Microcontroladores

Manual de asignatura

Sistema de Universidades Tecnológicas

ELECTRICIDAD Y ELECTRÓNICA INDUSTRIAL

Programa 2004

Créditos

Elaboró: **Ing. Eduardo Salinas Rodríguez.**

Revisó:

Colaboradores:

Autorizó:

Contenido

Objetivo general
Aplicar Microcontroladores en sistemas de control.

Habilidades por desarrollar en general
Escribir la habilidad propuesta que se debe desarrollar en esta asignatura.

I

Introducción a los Microcontroladores

Objetivo particular de la unidad
Describir la aplicación y características del Microcontrolador

Habilidades por desarrollar en la unidad
Conocer el uso y principios que rigen a los microcontroladores.

Saber en la Teoría (2 hrs.)

I.1 INTRODUCCION AL MICROCONTROLADOR.

- Explicar la evolución e historia del Microcontrolador.

Desde la invención del circuito integrado, el desarrollo constante de la electrónica digital ha dado lugar a dispositivos cada vez más complejos. Entre ellos los microprocesadores y los microcontroladores, los cuales son básicos en las carreras de ingeniería electrónica. Los microcontroladores están presentes en nuestro trabajo, en nuestra casa y en nuestra vida, en general. Se pueden encontrar controlando el funcionamiento de los ratones y teclados de los computadores, en los teléfonos, en los hornos microondas y los televisores de nuestro hogar. Pero la invasión acaba de comenzar y el nacimiento del siglo XXI será testigo de la conquista masiva de estos diminutos computadores, que gobernarán la mayor parte de los aparatos que fabriquemos y usemos los humanos.

Aunque el concepto de controlador ha permanecido invariable a través del tiempo, su implementación física ha variado frecuentemente. Hace tres décadas, los controladores se construían exclusivamente con componentes de lógica discreta, posteriormente se emplearon los microprocesadores, que se rodeaban con chips de memoria y E/S sobre una tarjeta de circuito impreso. En la actualidad, todos los elementos del controlador se han podido incluir en un chip, el cual recibe el nombre de microcontrolador.

- Enlistar los tipos de Microcontroladores.

Aunque en el mercado de la microinformática la mayor atención la acaparan los microprocesadores, lo cierto es que se venden cientos de microcontroladores por cada uno de ellos. Existe una gran diversidad de microcontroladores. Quizá la clasificación más importante sea entre microcontroladores de 4, 8, 16 ó 32 bits. Aunque las prestaciones de los microcontroladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los microcontroladores de 8 bits dominan el mercado y los de 4 bits se resisten a

desaparecer. La razón de esta tendencia es que los microcontroladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones, lo que hace absurdo emplear micros más potentes y consecuentemente más caros. Uno de los sectores que más tira del mercado del microcontrolador es el mercado automovilístico. De hecho, algunas de las familias de microcontroladores actuales se desarrollaron pensando en este sector, siendo modificadas posteriormente para adaptarse a sistemas más genéricos. El mercado del automóvil es además uno de los más exigentes: los componentes electrónicos deben operar bajo condiciones extremas de vibraciones, choques, ruido, etc. y seguir siendo fiables. El fallo de cualquier componente en un automóvil puede ser el origen de un accidente.

También los modernos microcontroladores de 32 bits van afianzando sus posiciones en el mercado, siendo las áreas de más interés el procesamiento de imágenes, las comunicaciones, las aplicaciones militares, los procesos industriales y el control de los dispositivos de almacenamiento masivo de datos.

- Definir los microcontroladores RISC y CISC

Existen tres orientaciones en cuanto a la arquitectura y funcionalidad de los procesadores actuales.

CISC: Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Computadores de Juego de Instrucciones Complejo). Disponen de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución. Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros.

RISC: Tanto la industria de los computadores comerciales como la de los microcontroladores están tendiendo hacia la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo. La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

SISC: En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es “específico”, o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Computadores de Juego de Instrucciones Específico).

- Reconocer las diferencias entre el microcontrolador y el microprocesador.

Una microcomputadora posee en su interior al microprocesador, que es un circuito integrado que contiene la Unidad Central de Proceso (CPU), también llamada procesador, de un computador; la CPU está formada por la Unidad de Control, que interpreta las instrucciones, y el Camino de Datos, que las ejecuta. Las patitas de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de E/S y configurar un computador implementado por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que

se destine (Fig. 1.1.). Un microcontrolador posee en su interior al microprocesador y todos los dispositivos periféricos de entrada/salida para que opere como una computadora en pequeño. (Fig. 1.1. y 1.2)

microcontrolador

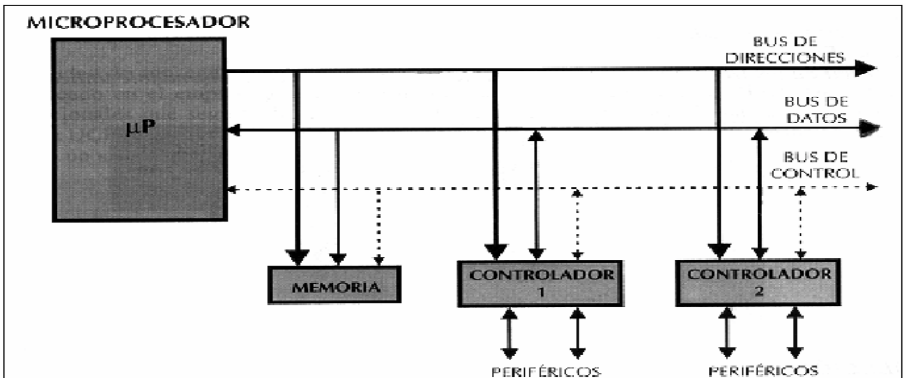


Fig. 1.1 Diferencia entre microprocesador y microcontrolador.

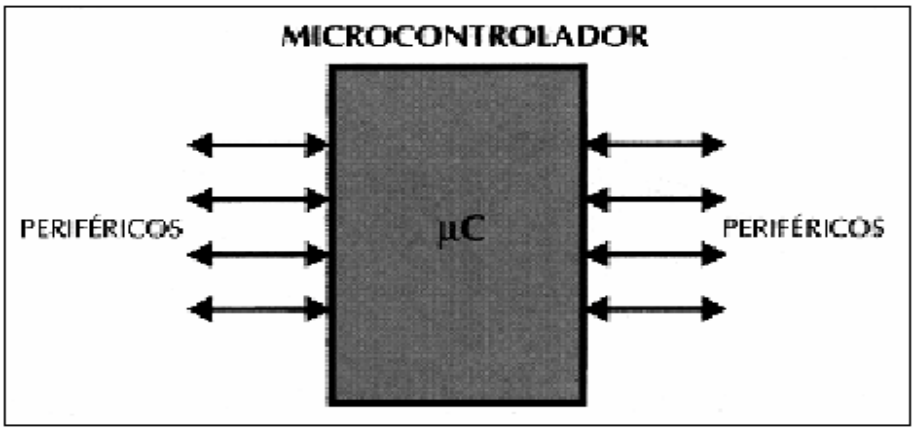


Fig. 1.2 Microcontrolador.

II

Arquitectura de Microcontroladores

Objetivo particular de la unidad
Identificar los componentes de la arquitectura de un microcontrolador y comprender sus principales funciones.

Habilidades por desarrollar en la unidad
Comprender su estructura interna de operación del microcontrolador y sus principios de funcionamiento.

Saber en la Teoría (2 hrs.)

I.1 ARQUITECTURA DEL MICROCONTROLADOR.

- Conocer la arquitectura: Entradas, salidas, buses de datos y direcciones, memoria, etc.

Un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador. Un microcontrolador dispone normalmente de los siguientes componentes:

- Procesador o UCP (Unidad Central de Proceso).
- Memoria RAM para Contener los datos.
- Memoria para el programa tipo ROM / PROM / EPROM
- Líneas de E/S para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, Puertas Serie y Paralelo, CAD: Conversores Analógico / Digital, CDA: Conversores Digital / Analógico, etc.).
- Generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema.

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de Von Neumann, en la actualidad se emplea más la arquitectura Harvard. La arquitectura de Von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control). La arquitectura Harvard dispone de dos memorias independientes una, que contiene sólo instrucciones y otra, sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones (lectura o escritura) simultáneamente en ambas memorias.

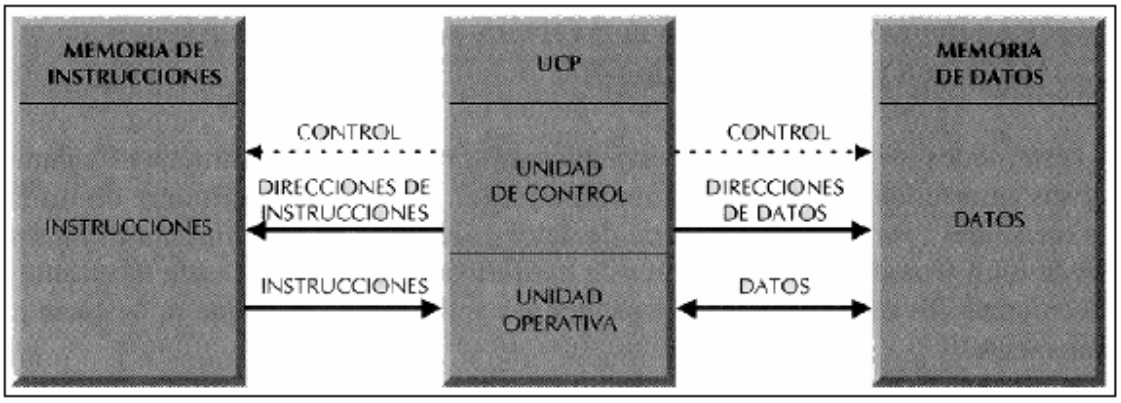


Fig. 2.1 La arquitectura Harvard dispone de dos memorias independientes para datos y para instrucciones, permitiendo accesos simultáneos.

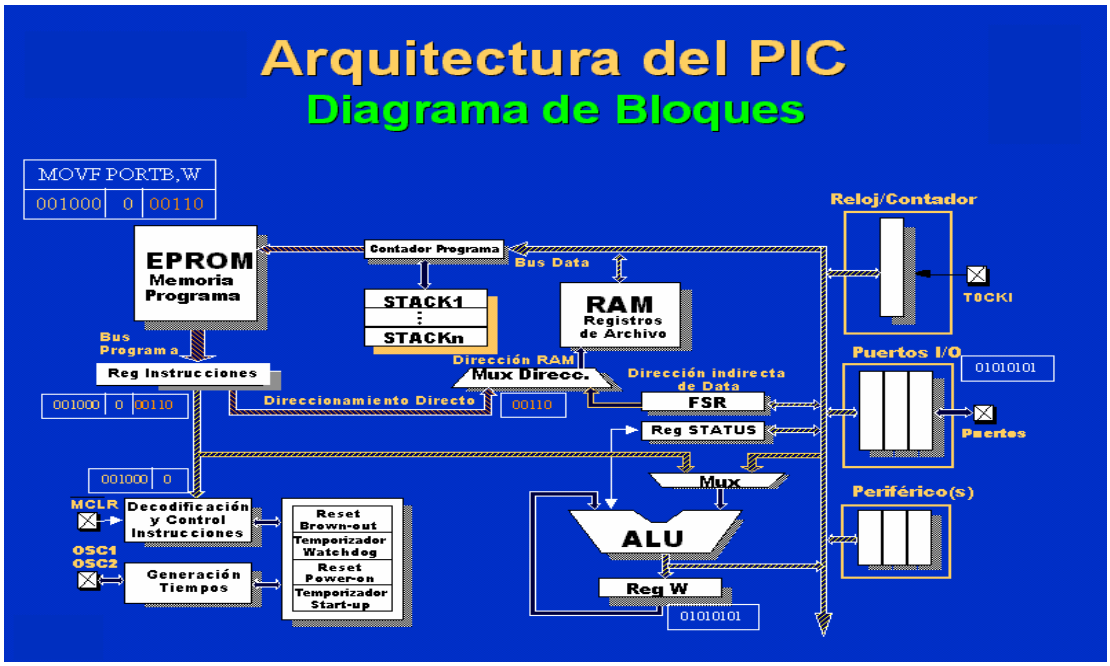


Fig. 2.2 La arquitectura de un microcontrolador PIC.

II.2 LA MEMORIA

- Nombrar los tipos de memoria del microcontrolador.

En los microcontroladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo ROM, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria será tipo RAM, volátil, y se destina a guardar las variables y los datos. Hay dos peculiaridades que diferencian a los microcontroladores de los computadores personales:

1. No existen sistemas de almacenamiento masivo como disco duro o disquetes.
2. Como el microcontrolador sólo se destina a una tarea en la memoria ROM, sólo hay que almacenar un único programa de trabajo.

La RAM en estos dispositivos es de poca capacidad pues sólo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como sólo existe un programa activo, no se requiere guardar una copia del mismo en la RAM pues se ejecuta directamente desde la ROM. Los usuarios de computadores personales están habituados a manejar Megabytes de memoria, pero, los diseñadores con microcontroladores trabajan con capacidades de ROM comprendidas entre 512 bytes y 8 k bytes y de RAM comprendidas entre 20 y 512 bytes. Según el tipo de memoria ROM que dispongan los microcontroladores, la aplicación y utilización de los mismos es diferente.

- MEMORIA DE PROGRAMA

La memoria de programa es del tipo flash, lo que facilita el desarrollo de programas y la experimentación. En ella se graba o almacena el programa de códigos que el microcontrolador debe ejecutar. Dado que el **PIC 16f84** tiene un contador de programa de 13 bytes, tiene una capacidad de direccionamiento de 8x14, pero solamente tiene implementado el primero (0000H hasta 03FFH).
La memoria de instrucciones puede tener una capacidad mínima de 512 palabras de 12 bits hasta una máxima de 2.048 palabras de la misma longitud. Durante la fase de búsqueda, la dirección de la instrucción la proporciona el PC, el cual normalmente se autoincrementa en la mayoría de las instrucciones, excepto en las de salto.

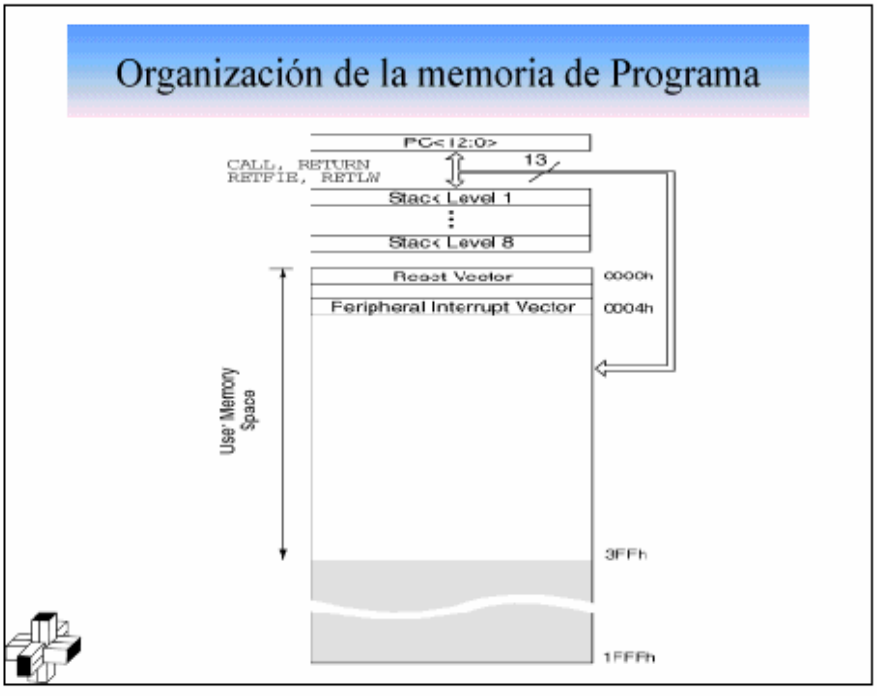


Fig. 2.3 Memoria de programa.

- MEMORIA DE DATOS

Los microcontroladores PIC funcionan con datos de 8 bits por lo que las posiciones de la memoria de datos tienen esa longitud. La capacidad de esta memoria varia entre 25 y 73 posiciones de un byte, en los modelos de la gama baja.
La memoria de datos se organiza en "bancos", pudiendo existir hasta cuatro en los modelos de mayor capacidad. El banco 0 tiene un tamaño de 32 bytes, mientras que todos los demás sólo disponen de 16. El banco 0 ocupa las 32 primeras posiciones de la memoria y lo poseen todos los modelos de PIC16C5X. Las 16 posiciones de menos peso de los bancos 1, 2 y 3 no son accesibles y cuando se direcciona desde el valor 00h al 0Fh, siempre se acude al banco 0. Cuando la dirección está comprendida entre la 10h y la 1Fh, hay que seleccionar a cuál de los cuatro posibles bancos corresponde con los bits 6 y 5 del registro FSR.
La memoria de datos funciona de forma similar al "banco de registros" de un procesador por lo cual sus posiciones implementan registros de propósito especial y propósito general. Las ocho primeras posiciones del banco 0 soportan registros específicos.

Dirección	BANCO 0	BANCO 1	Dirección
00	INDF	INDF	80
01	TMR0	OPTION	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	FSR	FSR	84
05	PORT A	TRIS A	85
06	PORT B	TRIS B	86
07			87
08			88
09			89
0A	PCLATH	PCLATH	8A
0B	INTCON	INTCON	8B
0C	PIR1	PIE1	8C
0D			8D
0E		PCON	8E
0F			8F
10			90
11			91
12			92
13			93
14			94
15			95
16			96
17			97
18			98
19			99
1A			9A
1B			9B
1C			9C
1D			9D
1E			9E
1F	CMCON	VRCON	9F
.	Reg. Prop. General	Reg. Prop. General	.
7F			FF

Fig. 2.4 Memoria de Datos

II.3 DISPOSITIVOS DE ENTRADA Y SALIDA

- Describir los modos de direccionamiento.

Los microcontroladores PIC tienen una arquitectura RISC, su juego de instrucciones es reducido (33 instrucciones en la gama baja), siendo éstas, además, sencillas y rápidas, puesto que casi todas se ejecutan en un único ciclo de máquina (equivalente a 4 del reloj principal). Sus operandos son de gran flexibilidad, pudiendo actuar cualquier objeto como fuente y como destino. Posee tres tipos bien diferenciados de direccionamiento, estos son:

1º Inmediato: El valor del dato está incluido en el propio código OP, junto a la instrucción.
2º Directo: La dirección del dato está incluido en el propio código OP, junto a la instrucción.
3º Indirecto: La dirección de la memoria de datos que guarda el operando está contenida en un registro.

- Definir los puertos de E/S, explicar timers y contadores, Explicar las interrupciones, Describir el convertidor interno analógico digital.

Puertos de Entrada y Salida

La principal utilidad de las patitas que posee la cápsula que contiene un microcontrolador es soportar las líneas de E/S que comunican al computador interno con los periféricos exteriores. Según los controladores de periféricos que posea cada modelo de microcontrolador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control. Un recurso imprescindible en los microcontroladores es el que soporta las Entradas y Salidas con los periféricos del mundo exterior. Los PIC16C5X sólo disponen de líneas de E/S digitales, cada una de las cuales tiene su correspondencia con una patita de la cápsula. Dichas líneas se agrupan en tres puertos Puerto A, Puerto B y Puerto C. El Puerto A actúa lo mismo que un registro de E/S de lectura y escritura. Sólo son válidos los 4 bits de menos peso que corresponden con las patitas RA3:RA0. Los 4 bits de más peso no están implementados y cuando se leen siempre valen 0. El Puerto B funciona como un registro de E/S de 8 bits bidireccionales, siendo accesibles todos ellos por las patitas RB7:RB0.

Vector de Reset

Cuando ocurre un reset al micro, el contador de programa se pone en ceros (0000H). Por esta razón en la primera dirección del programa se debe escribir todo lo relacionado con la inicialización del mismo.

Vector de Interrupción

Cuando el micro recibe la señal de una interrupción en el programa, el contador de programa apunta a la dirección 04H de la memoria de programa, por esta razón ahí se debe escribir toda la programación necesaria para atender dicha interrupción.

Reloj principal

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema. Generalmente, el circuito de reloj está incorporado en el microcontrolador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C.
Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía.

RECURSOS ESPECIALES

Cada fabricante oferta numerosas versiones de una arquitectura básica de microcontrolador. En algunas amplía las capacidades de las memorias, en otras incorpora nuevos recursos, en otras reduce las prestaciones al mínimo para aplicaciones muy simples, etc. La labor del diseñador es encontrar el modelo mínimo que satisfaga todos los requerimientos de su aplicación. De esta forma, minimizará el costo, el hardware y el software.
Los principales recursos específicos que incorporan los microcontroladores son:
• Temporizadores o “Timers”.
• Perro guardián o “Watchdog”.
• Protección ante fallo de alimentación o “Brownout”.
• Estado de reposo o de bajo consumo.
• Conversor A/D.
• Conversor D/A.
• Comparador analógico.
• Modulador de anchura de impulsos o PWM.
• Puertas de E/S digitales.
• Puertas de comunicación.

Temporizadores o “Timers”

Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores). Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso. Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las patitas del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

Perro guardián o “Watchdog

Cuando el computador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicializa el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.
Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización, “ladrará y ladrará” hasta provocar el reset.

Protección ante fallo de alimentación o “Brownout”

Se trata de un circuito que resetea al microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo (“brownout”). Mientras el voltaje de alimentación sea inferior al de brownout el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor.

Estado de reposo ó de bajo consumo

Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los microcontroladores disponen de una instrucción especial (SLEEP en los PIC), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se “congelan” sus circuitos asociados, quedando sumido en un profundo “sueño” el microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo.

Conversor A/D (CAD)

Los microcontroladores que incorporan un Conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde las patitas del circuito integrado.

Conversor D/A (CDA)

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por una de las patitas de la cápsula. Existen muchos efectores que trabajan con señales analógicas.

Comparador analógico

Algunos modelos de microcontroladores disponen internamente de un Amplificador Operacional que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por una de las patitas de la cápsula. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.
También hay modelos de microcontroladores con un módulo de tensión de referencia que proporciona diversas tensiones que se pueden aplicar en los comparadores.

Modulador de anchura de impulsos o PWM

Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patitas del encapsulado.

Puertas de E/S digitales

Todos los microcontroladores destinan algunas de sus patitas a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertas. Las líneas digitales de las Puertas pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el BIT correspondiente de un registro destinado a su configuración.

Puertas de comunicación

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos. Algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

• UART, adaptador de comunicación serie asíncrona.
• USART, adaptador de comunicación serie sincrona y asíncrona
• Puerta paralela esclava para poder conectarse con los buses de otros microprocesadores.
• USB (Universal Serial Bus), que es un moderno bus serie para los PC.
• Bus I2C, que es un interfaz serie de dos hilos desarrollado por Philips.
• CAN (Controller Area Network), para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles. En EE.UU. se usa el J185O.

III

Programación del Microcontrolador

Objetivo particular de la unidad

Programar un Microcontrolador y simular su operación.

Habilidades por desarrollar en la unidad

Aprender los pasos para poder programar y poner en operación al microcontrolador.

Saber en la Teoría (8 hrs.)

III.1 FORMATO DE DATOS.

- Enlistar tipos de datos

De acuerdo al set de instrucciones que tiene cada microcontrolador se dividirán las instrucciones de acuerdo a los tipos de dato de información que se manejen. Esta clasificación esta dividida en los siguientes formatos de datos:

- BIT
- BYTE
- PALABRA
- NUMERO FLOTANTE
- DOBLE PALABRA

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2

BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01	00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11	111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00	0000 0000 1001		
RETLW	k	Return with literal in W	2	11	01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00	0000 0000 1000		
SLEEP	-	Go into standby mode	1	00	0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010 kkkk kkkk	Z	

Fig. 3.1 Conjunto de instrucciones.

El set de instrucciones de la figura anterior permite hacer una clasificación de los datos orientados a byte, bit y a literales de control, pero sin embargo la clasificación que se tomara en cuenta será con respecto a los datos.
En el primer caso se maneja solo un bit de un registro, de tal forma que este solo puede tomar dos valores posibles (1 y 0 lógico) y solo se podrá afectar un solo bit a la vez de u registro, como se muestra en los siguientes ejemplos.

BCF DATO , 4
INSTRUCCIÓN REGISTRO BIT UTILIZADO

BTFSS DATO , 7
INSTRUCCIÓN REGISTRO BIT UTILIZADO

En estos ejemplos se puede observar como es que se ejecuta la instrucción de tal forma que aunque se dirija al contenido de un registro, solo afectara o revisara el contenido de un solo bit.

III.2 MODOS DE DIRECCIONAMIENTO DE DATOS DE 8 Y 16 BITS.

Las instrucciones de los PIC16CSX pueden especificar los datos u operandos mediante tres modos de direccionado:

1º Inmediato.
2º Directo.

3º Indirecto.

Cuando una instrucción utiliza un dato inmediato, su valor (literal) lo contiene su código operativo (OP) y en la ejecución se carga en el registro W para su posterior procesamiento.
En el direccionamiento directo se utilizan los 5 bits de menos peso del código OP de la instrucción para apuntar cualquiera de las 32 posiciones del banco 0. Si el dato no se halla en el banco 0, se usa un sistema de paginación para seleccionar el banco adecuado se emplean los bits 6 y 5 del registro FSR (Registro de Selección de Banco), que se denominan RAI y RA0, respectivamente. Como sólo son accesibles las 16 posiciones de más peso de los bancos 1, 2 y 3, cuando el BIT 4 del registro FSR vale 0, se ignoran los bits 6 y 5 y se accede al banco 0.
En las instrucciones con direccionamiento indirecto se usa como operando el registro INDF, que ocupa la posición 0 del banco 0. En tal caso, se accede a la posición que apunta el contenido del registro FSR ubicado en la posición 04 del área de datos. Sus 5 bits de menos peso apuntan la dirección del dato y los bits 6 y 5 seleccionan el banco. No tiene implementado el BIT 7, que siempre se lee como 1.
El registro INDF no se halla implementado físicamente. Cada vez que se le referencia, se utiliza el contenido del registro FSR para direccionar al operando.

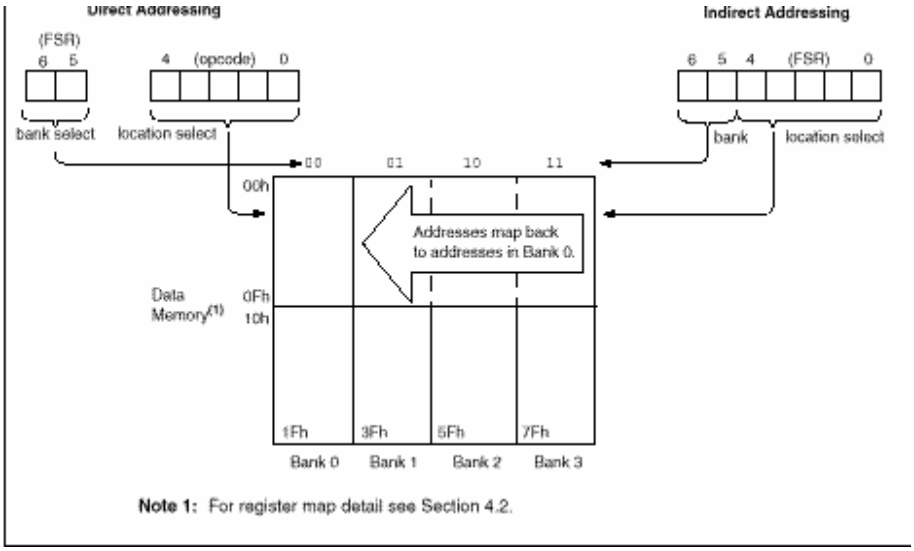


Fig. 3.2 Direccionamiento Directo e Indirecto

III.3 TRANSFERENCIA DE DATOS.

El manejo de datos en el microcontrolador esta basado en la forma en que se moverán los datos a otros recursos del microcontrolador ya sea:

- Registros
- Memoria
- La Pila
- El acumulador
- Dispositivos de entrada / salida

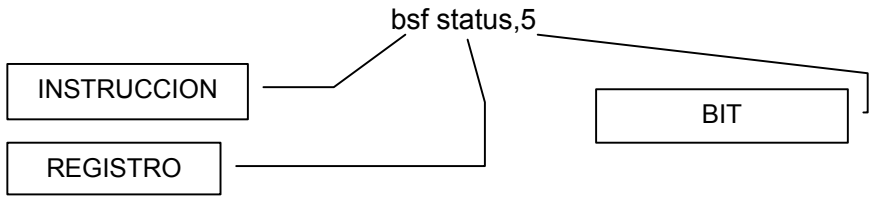
Para cada uno de estos se tendrá un medio particular del traslado de los datos, según las instrucciones que se ocupen.

Transferencia hacia Registros

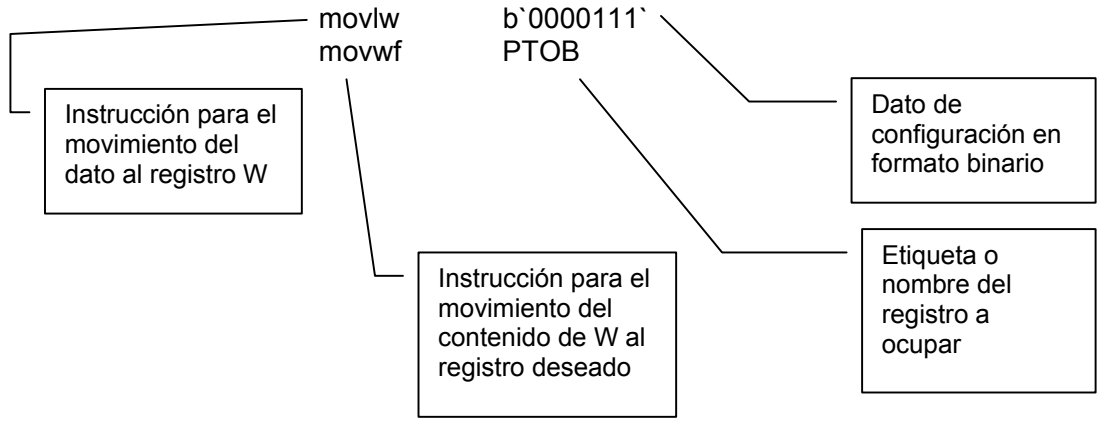
De acuerdo a los registros contenidos en la memoria de datos, esta memoria se encuentra particionada en un número determinado de áreas (bancos), y su uso depende de las necesidades del programador.

Tomando en cuenta que para los microcontroladores PIC de familia media, se encuentra la memoria de datos particionada en pagina 0 y pagina 1, la transferencia tendrá dos principios o el dato es para realizar una configuración o edición del registro o es para manejar un dato momentáneo sobre el registro.

Para la configuración del registro se tendrá que posicionarse en la pagina 1, haciendo uso del registro de estado (status) que de acuerdo a su composición de bits internos tiene la facultad de hacer el traslado de la pagina cero a la pagina 1. La secuencia de instrucciones será de la siguiente forma:



Ejecutando el anterior comando se podrá situar en la pagina uno y teniendo abierta la posibilidad de configurar el registro deseado. Para el movimiento de datos de configuración se utilizarán alguno de los modos de direccionamiento, como se muestra a continuación.



Para el manejo de datos sobre la pagina cero se realizara de la misma forma que en lo anterior, tomando las condiciones de uso para cada una de las paginas.

Transferencia hacia Memoria

El PIC contiene una memoria EEPROM de datos de 16 bits. Esta memoria no forma parte del espacio direccionable y solo es accesible para lectura y escritura a través de dos registros : un registro EEDATA en la dirección 08 para los datos y un registro EEADR en la dirección 09 para las direcciones. Y dos registros ECON. 1 y 2 que permiten definir el modo de funcionamiento de la memoria.

El principio de utilización de esta EEPROM en modo de lectura se basa en los siguientes puntos:

- Escritura de la dirección que hay que leer en el registro EEADR
- Puesta a 1 del bit RD del registro de control EECON1
- Lectura del dato así direccionado en el registro EEDATA

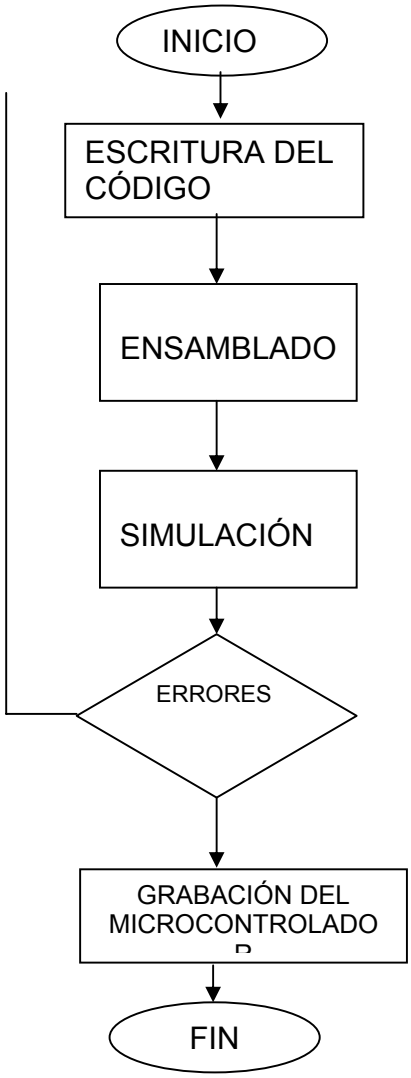
La escritura en esta memoria, que consiste en realidad en una programación, ya que se esta trabajando con una EEPROM, es algo mas complicado por evidentes razones de seguridad, este proceso se debe desarrollar de la manera siguiente:

- Escritura de la dirección que se desea escribir en el registro EEADR
- Escritura del dato en el registro EEDATA

De acuerdo al siguiente programa se

III.4 CONJUNTO DE INSTRUCCIONES.

Las fases de implementación para realizar un proyecto con el microcontrolador, en las que, de hecho, suponemos, una fase de estudio del problema, elegido ya el microcontrolador, y decidido el sistema de conexión son:



Programa fuente.

El programa fuente esta compuesto por una sucesión de líneas de programa. Cada línea de programa esta compuesta por 4 campos separados por uno o más espacios o tabulaciones. Estos campos son:

[Etiqueta] Comando [Operando(s)] [;Comentario]

La etiqueta es opcional. El comando puede ser un mnemónico del conjunto de instrucciones. El operando esta asociado al comando, si no hay comando no hay operando, e inclusive algunos comandos no llevan operando. El comentario es opcional para el compilador aunque es buena práctica considerarlo obligatorio para el programador. La etiqueta, es el campo que empieza en la primer posición de la línea. No se pueden insertar espacios o tabulaciones antes de la etiqueta sino será considerado comando. Identifica la línea de programa haciendo que el compilador le asigne un valor automáticamente. Si se trata de una línea cuyo comando es una instrucción de programa del microcontrolador, se le asigna el valor de la dirección de memoria correspondiente a dicha instrucción (location counter). En otros casos se le asigna un valor de una constante, o la dirección de una variable, o será el nombre de una macroinstrucción, etc.
El comando puede ser un código mnemónico de instrucción del microcontrolador, o una directiva o pseudoinstrucción para el compilador. En el primer caso será directamente traducido a código de maquina, en el segundo caso será interpretado por el compilador y realizara alguna acción en tiempo de compilación como ser asignar un valor a una etiqueta, etc. El campo de parámetros puede contener uno o más parámetros separados por comas. Los parámetros dependen de la instrucción o directiva. Pueden ser números o literales que representen constantes o direcciones.
El campo de comentario debe comenzar con el carácter punto y coma. No necesita tener espacios o tabulaciones separándolo del campo anterior, e incluso puede empezar en la primer posición de la línea. El compilador ignora todo el texto que contenga la línea después del carácter punto y coma. De esta manera pueden incluirse líneas que contengan solo comentarios, y es muy buena práctica hacer uso y abuso de esta posibilidad para que los programas resulten autodocumentados.

Conjunto de instrucciones.

El conjunto de instrucciones de los microprocesadores PIC 16F84 consiste en un pequeño repertorio de solo 35 instrucciones de 14 bits, que pueden ser agrupadas para su estudio en tres a cinco grupos. En este curso se ha optado por clasificarlas, desde el punto de vista del programador, en cinco categorías bien definidas de acuerdo con la función y el tipo de operandos involucrados. En primer lugar se agrupan las instrucciones que operan con bytes y que involucran algún registro de la memoria interna. En segundo lugar se analizaran las instrucciones que operan solo sobre el registro W y que permiten cargarle una constante implícita o incluida literalmente en la instrucción (literales). En tercer lugar se agrupan las instrucciones que operan sobre bits individuales de los registros de la memoria interna. En cuarto lugar se clasifican las instrucciones de control de flujo del programa, es decir las que permiten alterar la secuencia lineal de ejecución de las instrucciones. Por último se agrupan unas pocas instrucciones que llamaremos especiales, cuyas funciones o tipos de operandos son muy específicos y no encajan en ninguna de las clasificaciones anteriores.

Instrucciones de Byte que operan con Registros.

Estas instrucciones pueden ser de simple o doble operando de origen. El primer operando de origen será siempre el registro seleccionado en la instrucción, el segundo, en caso de existir, será el registro W. El destino, es decir donde se guardara el resultado, será el registro seleccionado o el W, según se seleccione con un bit de la instrucción.

Las instrucciones siguientes son las tres operaciones lógicas de doble operando :

```
ANDWF f,d ;operación AND lógica,  destino = W □ f
IORWF f,d  ;operación OR  lógica,  destino = W □ f
XORWF f,d ;operación XOR lógica,  destino = W □ f
```

Los nombres nemónicos de estas instrucciones provienen de : AND W con F, Inclusive OR W con F y XOR W con F.Las que siguen son las cuatro operaciones aritméticas y lógicas sencillas de simple operando :

```
MOVF f,d ;movimiento de datos,    destino = f
COMF f,d ;complemento lógico,     destino = NOT f
INCF f,d   ;incremento aritmético,  destino = f + 1
DECF f,d  ;decremento aritmético, destino = f - 1
```

Los nemónicos de estas instrucciones provienen de : MOVe File, COMplement File, INCrement File y DECrement File. En las siete instrucciones anteriores el único bit afectado de la palabra de estado del procesador es el Z, que se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor. A continuación siguen las dos instrucciones de rotación de bits a través del CARRY :

```
RLF f,d  ;rotación a la izquierda,   destino = f  ROT □
 RRF f,d  ;rotación a la derecha,    destino = f  ROT  □
```

En estas operaciones (Rotate Left File y Rotate Right File) los bits son desplazados de cada posición a la siguiente, en sentido derecho o izquierdo. El desplazamiento es cerrado, formando un anillo, con el bit C (CARRY) de la palabra de estado. En estas dos instrucciones, el único bit afectado de la palabra de estado del procesador es el bit C, que tomará el valor que tenia el bit 7 o el bit 0, según sea el sentido del desplazamiento. Estas instrucciones son muy útiles para la manipulación de bits, y además para realizar operaciones aritméticas, ya que en numeración binaria, desplazar un número a la izquierda es equivalente a multiplicarlo por 2, y hacia la derecha, a dividirlo por 2. La instrucción siguiente realiza el intercambio de posiciones entre los cuatro bits menos significativos y los cuatro más significativos (nibble bajo y nibble alto).

```
SWAPF f,d    ;intercambia nibbles,   destino = SWAP f
```

Esta instrucción (SWAP File) no afecta ninguno de los bits de la palabra de estado del procesador. Esta instrucción es muy útil para el manipuleo de números BCD empaquetados, en los que en un solo byte se guardan dos dígitos BCD (uno en cada nibble). Las dos operaciones que siguen son la suma y la resta aritméticas :

```
ADDWF f,d ;suma aritmética,  destino = f + W
SUBWF f,d ;resta aritmética,   destino = f  - W
```

Estas operaciones (ADD W a F y SUBstract W de F) afectan a los tres bits de estado C, DC y Z. El bit Z se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor. La suma se realiza en aritmética binaria pura sin signo. Si hay un acarreo del bit 7, es decir que el resultado es mayor que 255, el bit C (carry) resulta 1, en caso contrario resulta 0. Si hay un acarreo del bit 3, es decir que la suma de las dos mitades (nibbles) menos significativas (bits 0 a 3) resulta mayor que 15, se pone en 1 el bit DC (digit carry), en caso contrario se pone en 0.

Ejemplos :

```
  1010 0010              1101 0000
+ 0100 1111 C DC Z   + 0110 1111 C DC Z
  1111 0001  0   1   0     0011 1111  1   0   0
```

La resta se realiza sumando, en binario puro sin signo, el registro f más el complemento a dos de W (el complemento a 1, o complemento lógico, más 1)
Ejemplos :

```
f          0100 0100              0010 1000
W        - 0010 1000   C DC Z     - 0100 0100   C DC Z
           0001 1100   1   0  0     1110 0100   0  1   0
```

equivalente a :

```
      f      0100 0100              0010 1000
cmp.2 W    + 1101 1000  C DC Z    + 1011 1100  C DC Z
           0001 1100  1   0   0     1110 0100  0   1   0
```

Los bits de estado C y DC toman el valor normal correspondiente a la suma de f con el complemento a 2 de W. De esta manera el significado para la operación de resta resulta invertido, es decir que C (carry) es 1 si no hubo desborde en la resta, o dicho de otra manera, si el contenido de W es menor que el de f. El bit DC se comporta de manera similar, es decir que DC es 1 si no hubo desborde en la mitad menos significativa, lo que equivale a decir que el nibble bajo del contenido de W es menor que el del registro f.

Las instrucciones que siguen son de simple operando, pero son casos especiales ya que el destino es siempre el registro seleccionado :

```
CLRF f      ;borrado de contenido,    f = 0
MOVWF  f    ;copia contenido W □ f,  f = W
```

La instrucción CLRF (CLeaR File) afecta solo al bit Z que resulta siempre 0. La instrucción MOVWF (MOVe W a F) no afecta ningún bit de la palabra de estado.

Instrucciones de Byte que operan sobre W y Literales.

Estas instrucciones se refieren todas al registro W, es decir que uno de los operandos de origen y el operando de destino son siempre el registro W. En las instrucciones de este grupo que tienen un segundo operando de origen, este es siempre una constante de programa literalmente incluida en la instrucción, llamada constante literal o simplemente literal.

```
IORLW  k ; operación OR lógica,     W = W □ k
ANDLW k ; operación AND lógica,  W = W □ k
XORLW k ; operación XOR lógica,  W = W □ k
```

En estas tres instrucciones (Inclusive OR Literal W, AND Literal W y XOR Literal W) el único bit afectado de la palabra de estado del procesador es el Z, que se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor.
La instrucción que sigue sirve para cargar una constante de programa en el registro W:

```
MOVLW k   ;carga constante en W,    W = K
```

Esta (MOVe Literal W) instrucción no afecta ninguno de los bits de estado del procesador. La instrucción que sigue (CLeaR W) no correspondería incluirla en este grupo, y pertenece en realidad al primero, el de las instrucciones que operan sobre registros, ya que se trata de un caso especial de la instrucción CLRF, con destino W, y f = 0.

```
CLRW  ;borra el contenido de W, W = 0
```

Al igual que en la instrucción CLRF, el único bit de estado afectado es el Z que resulta 1.

Instrucciones de Bit

```
BCF f,b    ;borra el bit b de f              ;bit f(b) = 0
BSF f,b    ;coloca en uno el bit b de f   ;bit f(b) = 1
```

Estas instrucciones (Bit Clear File y Bit Set File) no afectan ningún bit de la palabra de estado del procesador.

Instrucciones de Control

```
GOTO k ;salto a la posición k (9 bits) del programa
```

Esta es la típica instrucción de salto incondicional a cualquier posición de la memoria de programa (que en la mayoría de los microprocesadores convencionales se llama JUMP). La constante literal k es la dirección de destino del salto, es decir la nueva dirección de memoria de programa a partir de la cual comenzarán a leerse las instrucciones después de ejecutar la instrucción GOTO. Esta instrucción simplemente carga la constante k en el registro PC (contador de programa). La única complicación de esta instrucción es que la constante k es de solo 9 bits, mientras que el registro PC es de 11 bits, ya que en el 16C57 debe permitir direccionar una memoria de programa de 2 K. Los dos bits faltantes, bit 9 y 10 del PC, son tomados respectivamente de los bits de selección de página PA0 y PA1 de la palabra de estado. El programador debe tener en cuenta que antes de ejecutar una instrucción GOTO es posible que haya que programar los bits PA0 y PA1.
La que sigue es la instrucción de llamado a subrutina:

```
CALL k ;salto a la subrutina en la posición k (8 bits)
```

Su comportamiento es muy similar al de la instrucción GOTO, salvo que además de saltar guarda en el stack la dirección de retorno de la subrutina (para la instrucción RETLW). Esto lo hace simplemente guardando en el stack una copia del PC incrementado, antes de que el mismo sea cargado con la nueva dirección k. La única diferencia con la instrucción GOTO respecto de la forma en la que se realiza el salto, es que en la instrucción CALL la constante k tiene solo 8 bits en vez de 9. En este caso también se utilizan PA0 y PA1 para cargar los bits 9 y 10 del PC, pero además el bit 8 del PC es cargado siempre con 0. Esto hace que los saltos a subrutina solo puedan realizarse a posiciones que estén en las primeras mitades de las paginas mencionadas. El programador debe tener en cuenta este comportamiento y asegurarse de ubicar las posiciones de inicio de las subrutinas en las primeras mitades de las paginas.
La instrucción que aparece a continuación es la de retorno de subrutina:

```
RETLW k       ;retorno de subrutina con constante k,     W = k
```

Esta (RETurn con Literal in W) instrucción produce el retorno de subrutina con una constante literal k en el registro W. La operación que realiza consiste simplemente en sacar del stack un valor y cargarlo en el PC. Ese valor es el PC incrementado antes de realizar el salto, de la ultima instrucción CALL ejecutada, por lo tanto es la dirección de la instrucción siguiente a dicho CALL.. Dado que el stack es de 11 bits, el valor cargado en el PC es una dirección completa, y por lo tanto se puede retornar a cualquier posición de la memoria de programa, sin importar como estén los bits de selección de pagina. Esta instrucción además carga siempre una constante literal en el registro W. Ya que esta es la única instrucción de retorno de subrutina de los PIC16C5X, no hay en estos microprocesadores forma de retornar de una subrutina sin alterar el registro W. Por otro lado, y con una metodología especial de programación, un conjunto de sucesivas

instrucciones RETLW puede ser usado como una tabla de valores constantes incluida en el programa (Ej. : tablas BCD/7 seg., hexa/ASCII, etc.).
A continuación se presentan las dos únicas instrucciones de “salteo” (skip) condicional. Estas instrucciones son los únicos medios para implementar bifurcaciones condicionales en un programa. Son muy generales y muy poderosas ya que permiten al programa tomar decisiones en función de cualquier bit de cualquier posición de la memoria interna de datos, y eso incluye a los registros de periféricos, los puertos de entrada / salida e incluso la palabra de estado del procesador. Estas dos instrucciones reemplazan y superan a todo el conjunto de instrucciones de salto condicional que poseen los microprocesadores sencillos convencionales (salto por cero, por no cero, por carry, etc.).

```
BTFSC f,b     ;salto si bit = 0,    bit =  f(0)  salta
BTFSS f,b     ;salto si bit = 1,    bit =  f(1)  salta
```

BTFSC (Bit Test File and Skip if Clear) saltea la próxima instrucción si el bit b del registro f es cero. La instrucción BTFSS (Bit Test File and Skip if Set) saltea si el bit es 1. Estas instrucciones pueden usarse para realizar o no una acción según sea el estado de un bit, o, en combinación con GOTO, para realizar una bifurcación condicional.

Las instrucciones que siguen son casos especiales de las de incremento y decremento vistas anteriormente. Estas instrucciones podrían categorizarse dentro del grupo de instrucciones orientadas a byte sobre registros (primer grupo), ya que efectivamente operan sobre los mismos, y el formato del código de la instrucción responde al de ese grupo, pero, a diferencia de las otras, pueden además alterar el flujo lineal del programa y por eso se les incluyó en este grupo.

```
DECFSZ f,d ; decrementa y salta sí 0, destino = f -  1, = 0  salta
INCFSZ f,d  ;incrementa y salta sí 0,  destino = f + 1, = 0  salta
```

Estas dos instrucciones (DECrement File and Skip if Zero, e INCrement File and Skip if Zero) se comportan de manera similar a DECF e INCF, salvo que no afectan a ningún bit de la palabra de estado. Una vez realizado el incremento o decremento, si el resultado es 00000000, el microprocesador salteara la próxima instrucción del programa. Estas instrucciones se utilizan generalmente en combinación con una instrucción de salto (GOTO), para el diseño de ciclos o lazos (loops) de instrucciones que deben repetirse una cantidad determinada de veces.

Instrucciones Especiales.
En este grupo se reunieron las instrucciones que controlan funciones específicas del microprocesador o que actúan sobre registros especiales no direccionados como memoria interna normal.
La instrucción que sigue es la típica NO OPERATION, existente en casi todos los microprocesadores.

```
NOP         ;no hace nada, consume tiempo
```

Esta instrucción solo sirve para introducir una demora en el programa, equivalente al tiempo de ejecución de una instrucción. No afecta ningún bit de la palabra de estado.
La siguiente es una instrucción especifica de control de los puertos de entrada / salida.

```
TRIS f ;carga el tristate control, TRISf = W
```

Esta instrucción (TRIState) carga el registro de control de los buffers tristate de un puerto de entrada salida (data dirección register), con el valor contenido en W. El parámetro f debe ser la dirección de memoria interna del puerto, aunque el valor W no será cargado en el puerto sino en el registro de tristate del mismo. Los valores validos para f son 4 y 5 en los 16C54/56 y 4, 5 y 6 en los 16C55/57. Esta instrucción no afecta ningún bit de la palabra de estado.
La siguiente instrucción sirve para programar el registro OPTION que controla el RTCC y prescaler

```
OPTION  ;carga el registro OPTION, OPTION = W
```

El registro OPTION no es accesible como memoria interna y solo se lo puede programar con esta instrucción. Esta instrucción no afecta ningún bit de la palabra de estado.
La instrucción que sigue borra el contador del watch dog timer. Este registro tampoco esta accesible como memoria, y esta es la única instrucción que lo modifica.

```
CLRWDT  ;borra el watch dog timer, WDT = 0
```

Esta instrucción, además, coloca en uno los bits PD (power down) y TO (time-out) de la palabra de estado.
La siguiente es una instrucción especial de control del microcontrolador que lo pone en el modo power down. En este modo el microprocesador se detiene, el oscilador se apaga, los registros y puertos conservan su estado, y el consumo se reduce al mínimo. La única forma de salir de este estado es por medio de un reset o por time-out del watch dog timer.

```
SLEEP  ;coloca el µC en modo sleep, WDT = 0
```

Esta instrucción, además, borra el bit PD (power down) y setea el bit TO (time-out) de la palabra de estado.

Operaciones Orientadas al Byte

MOVWF	f	Mueve W a f
CLRW	-	Limpia W
CLRF	f	Limpia f
SUBWF	f,d	Resta W de f
DECF	f,d	Decrementa f
IORWF	f,d	OR Inclusivo de W y f
ANDWF	f,d	AND de W con f
XORWF	f,d	OR Exclusivo de W y f
ADDWF	f,d	Suma W y f
MOVF	f,d	Mueve f
COMF	f,d	Complemento de f
INCF	f,d	Incrementa f
DECFSZ	f,d	Decrementa f, salta si es cero
RRF	f,d	Rota f hacia la derecha
RLF	f,d	Rota f hacia la izquierda
SWAPF	f,d	Intercambia mitades de f
INCFSZ	f,d	Incrementa f, salta si es cero

Operaciones Orientadas al BIT

BCF	f,b	Limpia registro f, bit b (coloca 0)
BSF	f,b	Activa registro f, bit b (coloca 1)
BTFSC	f,b	Prueba registro f, bit b salta si está en cero
BTFSS	f,b	Prueba registro f, bit b salta si está en uno

Operaciones con Literales y de Contro

NOP	-	No Operación
OPTION	-	Carga registro OPTION
SLEEP	-	Pasa a modo de espera
CLRWDT	-	Limpia temporizador Watchdog
TRIS	f	Configura Puerto f
CALL	a	Llamada a Subrutina
RETLW	k	Regreso, coloca Literal en W
GOTO	a	Salto (a =dirección)
MOVLW	k	Mueve Literal a W
IORLW	k	OR Inclusivo de Literal y W
ANDLW	k	AND de Literal con W
XORLW	k	OR Exclusivo de Literal y W
ADDLW	k	Suma Literal a W
SUBLW	k	Resta Literal de W
RETFIE	-	Regreso de una interrupción
RETURN	-	Regreso de una subrutina

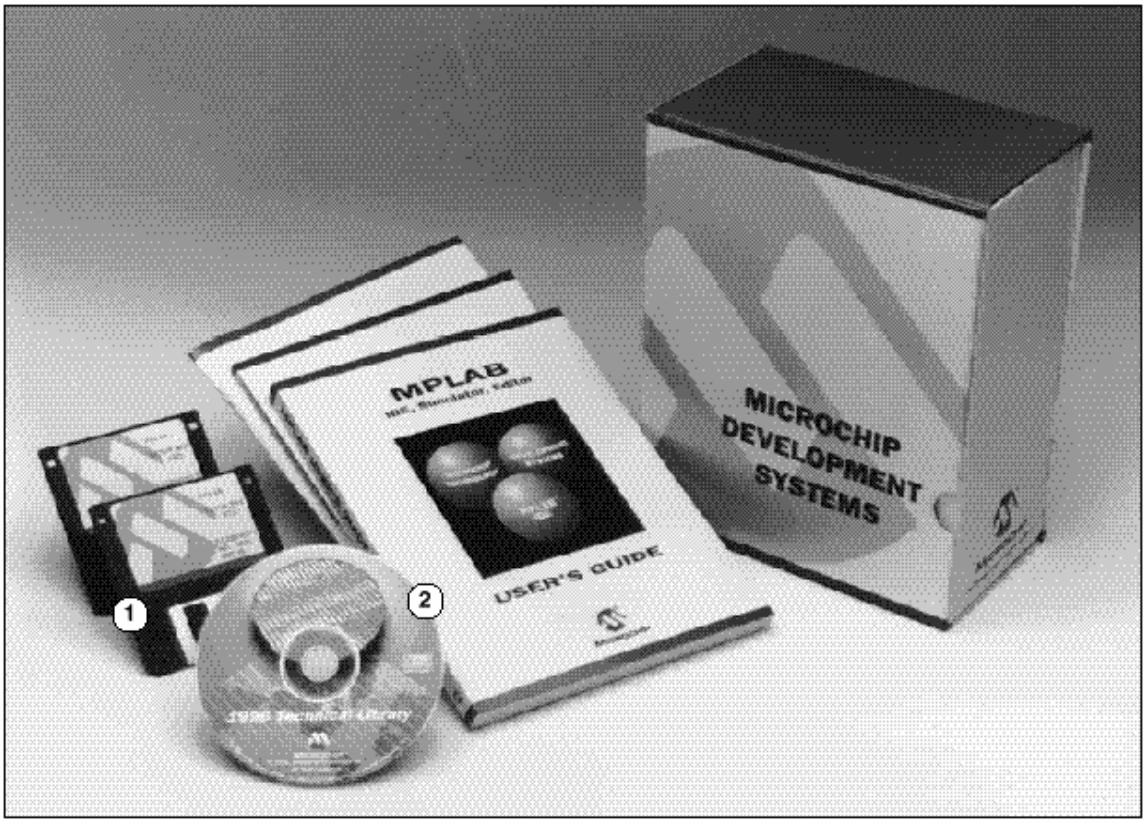
Núcleo de 12-bits: Resumen
Operaciones orientadas al Byte

NOP	-	No operación
MOVWF	f	Mueve W a f
CLRW	-	Limpia W
CLRF	f	Limpia f
SUBWF	f,d	Resta W de f
DECF	f,d	Decrementa f
IORWF f,d	OR Inclusivo de W y f	
ANDWF	f,d	AND de W con f
XORWF	f,d	OR Exclusivo de W y f
ADDWF	f,d	Suma W y f
MOVF	f,d	Mueve f
COMF	f,d	Complementa f
INCF	f,d	Incrementa f
DECFSZ	f,d	Decrementa f, salta si es cero
RRF	f,d	Rota f hacia la derecha
RLF	f,d	Rota f hacia la izquierda
SWAPF	f,d	Intercambia mitades de f
INCFSZ	f,d	Incrementa f, salta si es cero

III.5 SIMULACION DE MICROCONTROLADORES.

ENTORNO DE DESARROLLO INTEGRADO MPLAB

El MPLAB corre bajo Microsoft Windows. MPLAB proporciona funciones que permiten:
· Crear y Editar archivos fuentes
· Agrupar archivos fuente dentro de proyectos
· Depurar código fuente



Herramientas de MPLAB
MPLAB es un conjunto de herramientas para el desarrollo y depuración de aplicaciones en un proyecto. MPLAB incluye un editor de texto, un manejador de proyectos para mantener tu código organizado y un ambiente para depurar el software que desarrollas para tus propios diseños. Este ambiente incluye el simulador MPLAB-SIM, y soporta otras herramientas de Microchip tales como el emulador PICMASTER y el programador de dispositivos PICSTART Plus.

Manejador de Proyectos
El manejador de proyectos es la parte central del MPLAB. Sin la creación de un proyecto no se puede hacer depuración alguna. A través del manejador de proyectos se hacen las siguientes operaciones:
· Crear un proyecto
· Añadir un archivo de código fuente a un proyecto
· Ensamblar o compilar código fuente

· Editar código fuente
· Reconstruir todos los archivos fuente, o compilar un archivo sencillo
· Depurar el código fuente

El Editor MPLAB
El Editor de MPLAB permite a los programadores escribir y editar código fuente para las familias de microcontroladores PIC16/17, así como otros archivos de texto.

El Ensamblador MPASM
El Ensamblador Universal de Microcontroladores PIC16/17 MPASM ofrece grandes características completamente desarrolladas, ensamblado condicional y de diferentes fuentes y lista de formatos. MPASM permite generar varios formatos de código objeto que soportan las herramientas de desarrollo de Microchip así como los programadores relacionados sin salir de MPLAB.

El Simulador MPLAB-SIM
El simulador MPLAB-SIM permite aislar problemas de código y depurar diseños en los microcontroladores PIC16/17. Simula las funciones principales así como la mayoría de los periféricos de las familias de microcontroladores PIC16/17.

Otras Herramientas
El MPLAB soporta herramientas de desarrollo tales como programadores, compiladores y emuladores ya sea de Microchip o de otros diseñadores.

REQUERIMIENTOS DE SOFTWARE Y HARDWARE
Relativo al software el MPLAB funciona correctamente en un ambiente operativo de Windows, ya que soporta el entorno de multitareas. Debido a que MPLAB tiene capacidades de emulación de multiprocesador, soporta DDE (intercambio dinámico de datos) con programas cliente, los datos recolectados con MPLAB pueden ser intercambiados con mas programas.
Para tomar ventaja de las características de emulación del sistema, se debe instalar el software de MPLAB en una computadora con la siguiente configuración (mínimo):
· PC 386 o superior. Se recomienda Pentium
· 4 MB de memoria en RAM, 16MB recomendado
· 8 MB de espacio libre en disco duro, 20 MB recomendado
· Monitor VGA o Súper VGA
· Microsoft Windows 3.1 o superior

FUNCIONES DE MPLAB
Después de ajustar y compilar un proyecto en MPLAB, querrás ver como corre el código. Si se cuenta con un programador, se puede programar un microcontrolador y conectarlo en la aplicación actual para verificar que la aplicación funciona como se esperaba. Comúnmente, una aplicación no funciona correctamente la primera vez, y se tendrá que depurar el código. Se puede usar el MPLAB-SIM para simular el código o se puede usar el emulador PICMASTER para correr tu aplicación en la aplicación actual mientras lo depuras.
De otra manera, se pueden usar puntos de ruptura para ver como corre el código. Observar los valores de los registros en la ventana de Registros o la ventana de Registros de Funciones Especiales para ver el estado del procesador tal y como funcionaria tu código paso por paso.

El emulador PICMASTER corre el código en tiempo real en tu hardware objetivo, deteniéndose solamente en los puntos de ruptura que se hayan especificado. El MPLAB-SIM simula la ejecución de cualquier PIC16/17 y simula las condiciones de Entrada/Salida mas la velocidad depende de la velocidad de la PC en que se ejecuta.
Las siguientes funciones de depuración trabajan igual con el simulador o el emulador. Las funciones principales son:
· Emulación de la memoria (ventana Memoria de Programa)
· Puntos de Ruptura
· Paso-a-Paso
· Monitoreo de Registros (SFR y File Register)
Todas estas funciones utilizan información de un proyecto de MPLAB.

Ejecución en Tiempo-Real
La ejecución en Tiempo-Real solo es aplicable al emulador PICMASTER.

Ejecución en el Modo simulador MPLAB-SIM
Cuando se le dice al sistema que corre en tiempo real en el modo simulador, las instrucciones se ejecutan tan rápido como es posible por el software. Esto comúnmente es mas lento que lo que en realidad podrían corren los microcontroladores PIC16/17 en su ciclo de reloj.
La velocidad a la cual el simulador corre depende de la velocidad de la computadora en la que corre y de las demás tareas que se estén ejecutando al mismo tiempo. El software simulador debe de actualizar todos los registros simulados en RAM, también monitorear las Entradas / salidas, ajustar y limpiar las banderas, checar y buscar por puntos de ruptura en el software, y simular las instrucciones de los microcontroladores PIC16/17 tal y como están siendo ejecutadas en el CPU de la computadora.
Nota: Rutinas de tiempo pueden ser utilizadas en el código para generar retardos de tiempo. Cuando se utiliza el simulador, se podría decrementar el tiempo de esos retardos o condicionarlos removiendo esas secciones del código con el fin de incrementar la velocidad de simulación.
En general cuando este manual dice “tiempo real” y se esta en modo de simulador, quiere decir que la simulación del software del código de los PIC16/17 es ejecutada simulada tan rápido como la PC puede simular las instrucciones.

Modo Animado
El modo animado es un método en que el procesador camina automáticamente paso-paso. El simulador ejecuta pasos sencillos mientras corre, pero solamente actualiza los valores de los registros cuando es detenido. Para ir observando los cambios en las ventanas de Registros de Funciones Especiales y Archivos de Registros se utiliza en modo animado. El modo animado corre mas lento que la función RUN, pero permite ver los cambios en los valores de los registros.

Ambiente del Simulador MPLAB-SIM
El MPLAB-SIM es un simulador de eventos discretos para las familias de microcontroladores PIC16/17 y se encuentra integrado al MPLAB IDE. La herramienta de simulador MPLAB-SIM esta diseñada para:
· Modelar las operaciones de las familias de microcontroladores PIC16C5X, PIC16CXX y PIC17CXX de Microchip Technology
· Asistir al usuario en la depuración del software que utilizan los microcontroladores de Microchip.

Un simulador de eventos discretos, así como un emulador in-circuit como el emulador PICMASTER, están diseñados para depurar software. El MPLAB-SIM permite modificar código objeto e inmediatamente reejecutarlo, inyectarle estímulos externos al procesador simulado, y trazar la ejecución del código objeto. Un simulado difiere de un emulador incircuit en tres áreas importantes:
· Tiempo de las Entradas / salidas
· Velocidad de ejecución
· Costo

Tiempo de las Entradas / salidas
El tiempo en el MPLAB-SIM es procesado únicamente durante cada ciclo de instrucción. Las señales transitorias tales como los impulsos al MCLR, mas pequeñas que los ciclos de instrucción no pueden ser simuladas en un simulador pero si en un emulador in-circuit.

Velocidad de Ejecución
La velocidad de ejecución de un evento discreto en un software simulador es de acuerdo a la magnitud y a la orientación de la resolución del hardware. El usuario podría ver la velocidad de la ejecución tan lenta o tan rápido como la herramienta lo permita. MPLAB-SIM esta diseñado para proporcionar la simulación de ciclos los mas rápida posible, y depende de los modos de operación, puede operar en el orden de los milisegundos por instrucción.

Costo
La tecnología de Microchip ha desarrollado el simulador MPLAB-SIM para el ser la herramienta para depuración de software de aplicaciones mas efectiva por su costo. MPLAB-SIM no requiere ningún tipo de software externo a la PC, y opera en la mayoría de sus funciones exactamente igual al emulador PICMASTER. A menos que se necesite depurar la aplicación en tiempo real en el hardware actual, MPLAB-SIM puede ser utilizado para buscar y corregir la mayoría de los errores de codificación.

Herramienta de Depuración
MPLAB-SIM esta particularmente catalogado para la optimización de algoritmos. A diferencia de un emulador, el simulador hace visibles cualquier registro interno o puede proporcionar herramientas de software que son difíciles o caras de implementar en hardware con un emulador in-circuit. Para la mayoría de los casos MPLAB-SIM puede ser utilizado para depurar completamente el sistema a menos que se corra en situaciones donde un emulador in-circuit es requerido.

Consideraciones del Simulador
De cualquier manera, existen algunos eventos físicos que no pueden ser simulados apropiadamente. Estos caen dentro de las siguientes categorías:
· Eventos puramente asíncronos
· Eventos que tienen periodos mas cortos que un ciclo de instrucción (1 ciclo de instrucción = 4 ciclos de reloj)
En realidad, la red resultante simula únicamente ciclos de instrucción de todos los eventos sincronizados, y todos los eventos mas pequeños que un ciclo de instrucción no son reconocidos.
Los siguientes son una lista de operaciones de las familias de microcontroladores PIC16/17 las cuales se ven afectadas en la simulación:
· Entradas de reloj mas pequeñas que un ciclo de reloj utilizadas en el Prescaler
· Salidas PWM cuya resolución es menor que un ciclo de reloj

· Comparaciones mayores a 8 bits no se soportan
· En modo contador no sincronizado, las entradas mas pequeñas a un ciclo de reloj no se pueden usar.
· Las formas de onda del oscilador en RC0/RC1 no pueden mostrarse.

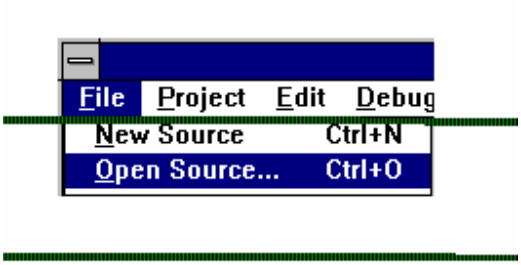
COMO USAR EL PROGRAMA MPLAB Y EL ENSAMBLADOR MPASM

Creación y desarrollo

MPLAB es un Entorno de Desarrollo Integrado (IDE) fácil de aprender y fácil de usar. La característica IDE proporciona a los desarrolladores de software para aplicaciones la flexibilidad para editar, compilar, emular, simular, desarrollar y depurar su propio software para las familias de microcontroladores PIC16/17 de Microchip.

Figura 7-1. Ventana del MPLAB IDE

El programa MPLAB es un software que contiene un editor, un ensamblador, un emulador y un simulador, todos ellos integrados en el mismo ambiente. El editor nos sirve para escribir un nuevo programa o modificarlo, para empezar a trabajar en el editor se necesita abrir o crear un nuevo archivo fuente como a continuación se muestra un ejemplo:

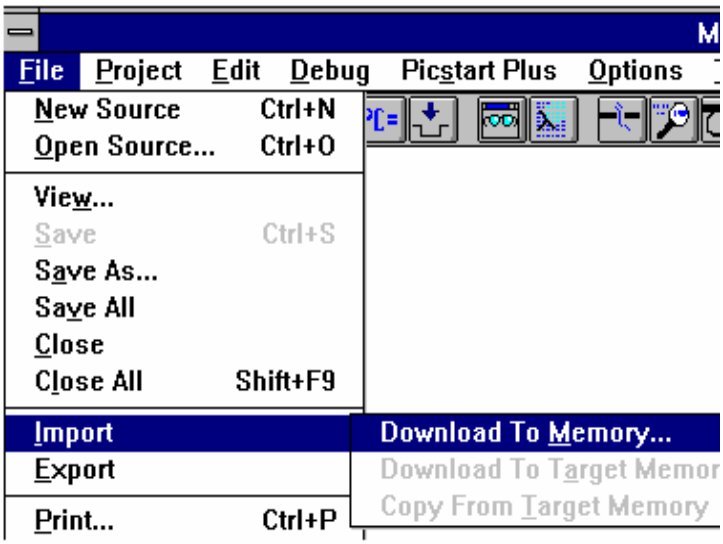


Esta figura nos muestra la manera de como abrir un archivo mas la opción NEW es la adecuada para comenzar a realizarlo.

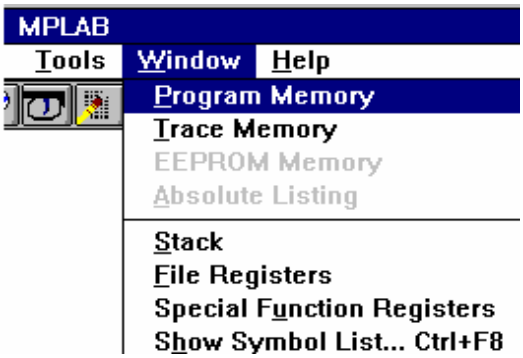
Una vez terminado de escribir el programa en el editor se procede a salvarlo (guardarlo) presionando CTRL+S, una vez ahí es tiempo de ponerle el nombre que uno desee, se recomienda que sea adecuado al programa o aplicación que se esta realizando, después de esto se procede a ejecutar el programa MPASM para pode ensamblar el archivo que acabamos de guardar / crear.

Esta figura nos muestra el programa MPASM que es un ensamblador para microcontroladores PIC de Microchip. Para comenzar a ensamblar se localiza con el mouse/teclado el botón que dice Browse y lo presionamos para localizar / indicar nuestro archivo que vamos a ensamblar.
Una vez que ya localizamos el archivo buscaremos el numero / modelo de PIC que usaremos para nuestra aplicación o para simularlo, una vez establecido el microcontrolador PIC que se utilizara se procede a ensamblar el archivo presionando el botón Assemble y aparecerá el siguiente cuadro:

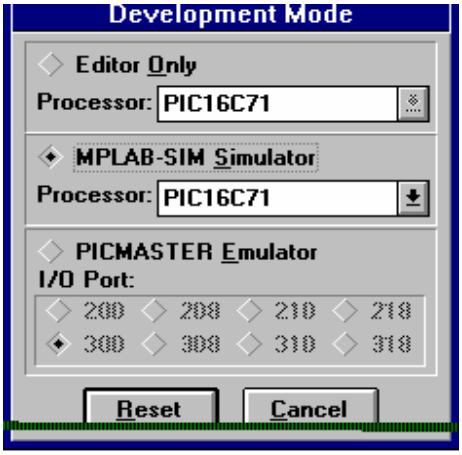
Este cuadro nos muestra los errores, warnings (advertencias), mensajes y líneas ensambladas, si no se tuvo ningún error el siguiente paso es simular y si no se procede a corregir los errores auxiliándose del archivo .ERR, que es generado por MPASM y se puede leer en el editor de MPLAB o en cualquier otro editor de texto, que nos muestra la línea en que nos equivocamos.
Para esto lo podemos abrir como cualquier otro archivo con el que se desea trabajar, indicado anteriormente, este archivo generalmente es generado junto con otros mas con extensión diferente, por ejemplo . COD, .HEX, y se localizan en el mismo directorio o ubicación que el archivo fuente.



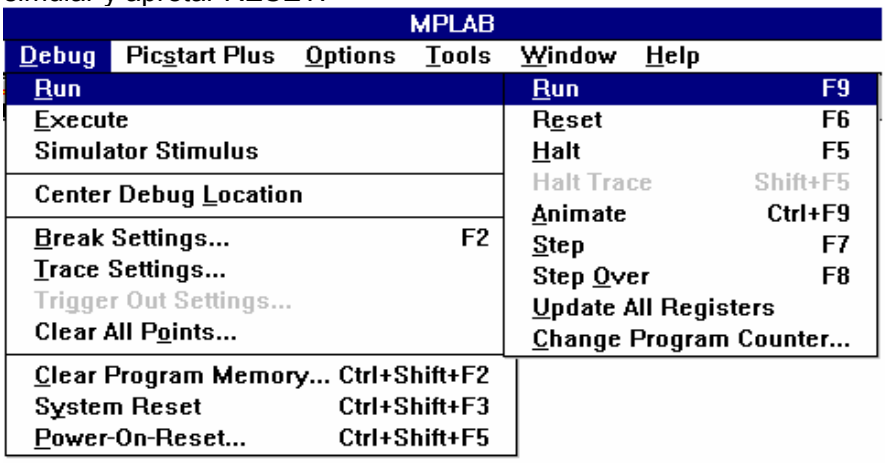
Para comenzar a simular el archivo después de haber sido correctamente ensamblado se procede a bajar el archivo .HEX del archivo que queremos simular a la memoria del simulador.



El siguiente paso es abrir una ventana con el programa ensamblado para poder simularlo.
También abrir File Registers y Special Function Registers



Para poder simular en este programa es necesario cambiar la configuración de modo Editor a modo Simulador, encontrar tu modelo de microcontrolador PIC con el que se va a simular y apretar RESET.

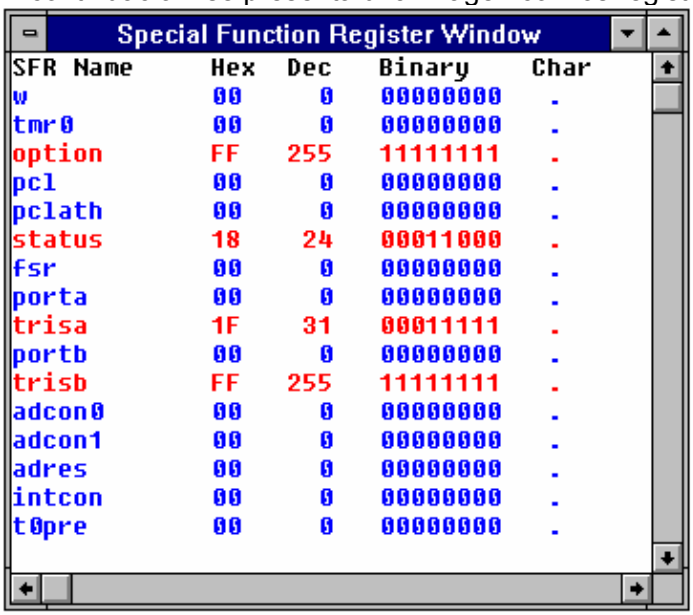


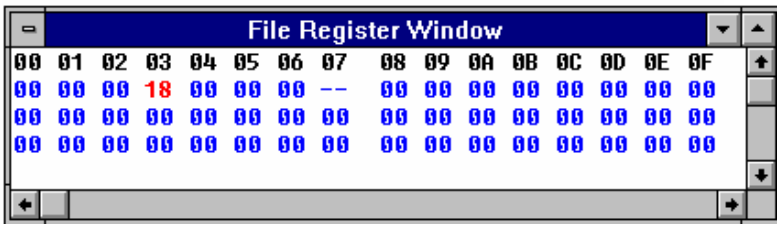
El siguiente paso es comenzar a simular nuestro programa, primero tendremos que mover el mouse hacia DEBUG y se verán todas las barras de dicha tarea:

Ahora explicare para que sirve cada barra, para correr un programa automáticamente presionaremos ANIMATE, para detenerlo esta HALT y el RESET para inicializar nuestro programa hasta PC 0x00 (inicio del contador de programa). STEP es para ir simulando paso a paso cada instrucción.

Para resetear el programa .HEX que esta en memoria en caso de que se desee simular otro o de que al programa existente se le haya hecho algún cambio y se haya ensamblado de nuevo, se utiliza CLEAR PROGRAM MEMORY y para limpiar todos los registros de memoria y registros especiales como: PORTB, PORTA, W(acumulador),TIMERS, etc., se utiliza CLEAR ALL POINTS.

A continuación se presenta una imagen con los registros especiales registro de memoria





Modificar.

Seleccione Window>Modify para desplegar o modificar los contenidos de la Memoria de Datos, Memoria de Programas, la Pila o la memoria EEPROM.

Modificar permite leer / escribir a una dirección especifica, leer/escribir mientras se esta incrementando a la siguiente dirección o llenar un bloque de direcciones. MPLAB permite dejar abierta la ventana de Modificar todo el tiempo. Existen cuatro maneras para abrir el cuadro de dialogo Modificar:

· Seleccionar Window>Modify

· Doble clic en un elemento en la ventana de registros de funciones especiales

· Doble clic en un elemento de la ventana Watch
· Seleccionar una dirección o un rango en la ventana File Register y dar un clic en el botón derecho del mouse para desplegar el botón Fill Register. Presione el botón Fill Register para desplegar el cuadro de dialogo Modify.

Elementos del cuadro de dialogo Modify

Address	Introduce la dirección en la cual el dato va a ser leído o modificado. Se puede introducir una dirección numérica o un símbolo (etiqueta)
Data/Opcode	Presione Read para desplegar el valor del dato/Opcode de la dirección seleccionada y el área de memoria. Presione en Write para escribir el valor del dato/Opcode de la dirección seleccionada y área de memoria en hexadecimal, decimal o mnemónico
Memory Area	Seleccione el área de memoria que se quiera modificar
Data Memory	Memoria RAM
Program Memory	Memoria ROM en el emulador
Stack	Memoria de la Pila del dispositivo
EEPROM	Datos de la memoria EEPROM
End Adr	La dirección final para llenar el rango
Fill Range	Llena el rango definido por las dos direcciones con el valor introducido en Data/Opcode
Auto Increment	Seleccione auto increment para incrementar a la siguiente dirección después de cada lectura/escritura

Nota: Auto Increment avanza a la siguiente dirección, despliega la siguiente dirección y lee el contenido de la dirección. Si se esta utilizando Auto Increment para leer un rango, introduzca la dirección del área de memoria menos uno, debido a que la primer lectura incrementara la dirección.

Write	Introduce un nuevo en el campo Data/Opcode, presione Write para modificar el dato de la dirección especifica. (Se pueden introducir datos en formato simbólico) Cuando un dato es modificado, todas las ventanas que lo utilizan se actualizan con la nueva información
Read	Presione Read para leer el dato de la dirección especificada
Close	Presione Close para salir de Modificar.

IV

Interfaseado con el Microcontrolador

Objetivo particular de la unidad

Utilizar y acoplar diferentes elementos para incrementar la capacidad de operación de las entradas y salidas del Microcontrolador.

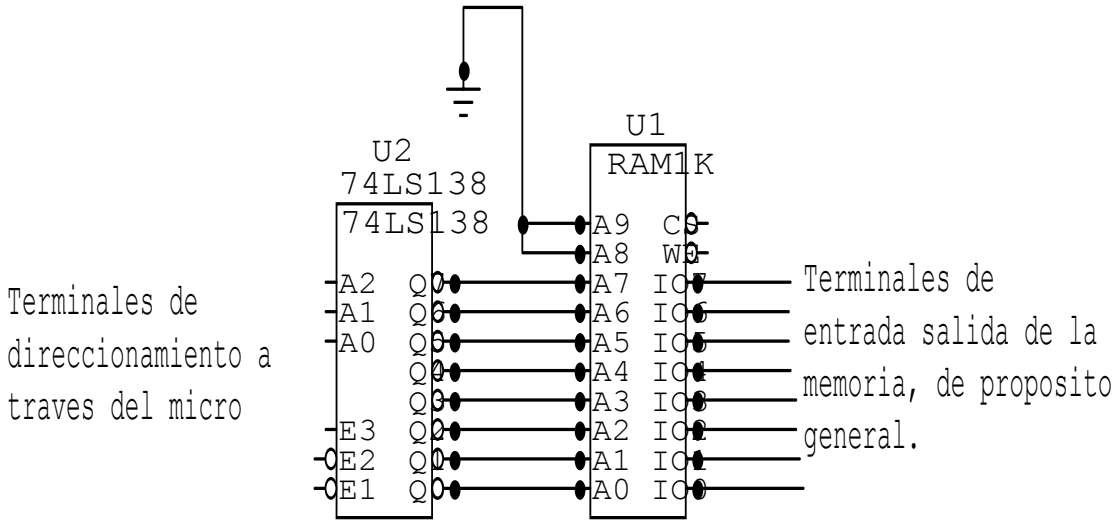
Habilidades por desarrollar en la unidad

Escribir la habilidad propuesta que se debe desarrollar en esta asignatura.

IV.1 INCREMENTO DE LA MEMORIA EXTERNA.

En algunos casos la implementación de circuitos o proyectos a través de microcontroladores se ve limitado por el numero finito de capacidad de memoria con la que cuenta el microcontrolador. En estos casos se recomienda el uso de memorias externas las cuales complementen la capacidad de memoria del micro sin omitir el uso de la misma que el contiene.

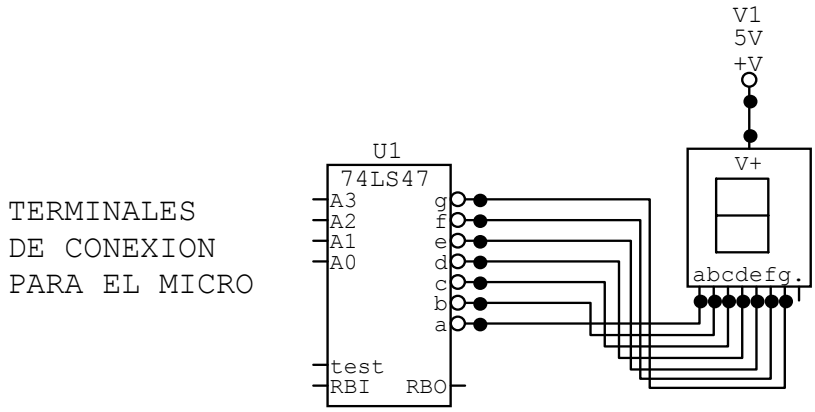
El tipo de memoria puede ser muy variado con respecto a las necesidades que presente el proyecto, tanto en tipo de memoria (prom, eporm, eeprom, etc) y su capacidad de almacenamiento. Al momento de hacer este tipo de implementaciones es necesario o se recomienda que en el diseño se tome en cuenta el no sacrificar el numero de terminales de entrada - salida, de tal forma que sea necesario incorporar elementos tales como multiplexores o decodificadores para hacer el direccionamiento de las memorias externas que se lleguen a conectar al micro, de acuerdo como se muestra en la siguiente figura.



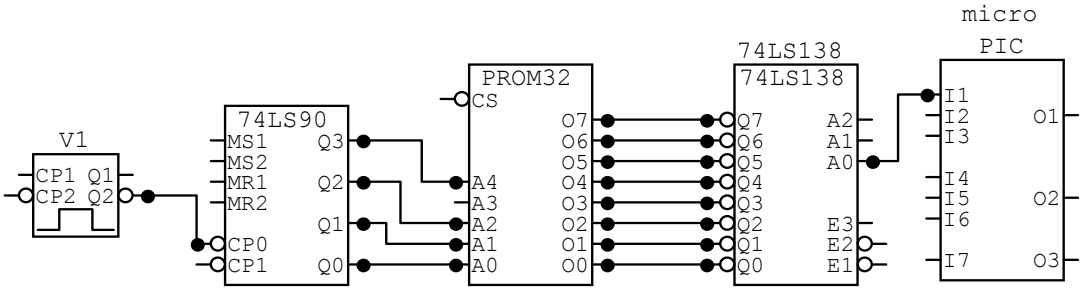
IV.2 INCREMENTO DE PUERTOS DE ENTRADA Y SALIDA

El manejo de los puertos es una tarea muy cotidiana en la programación e implementación de circuitos, el problema comienza en el momento que los puertos que contiene el microcontrolador no son los suficientes para poder controlar todas las actividades que se requieren para el sistema. En estos casos se tienen dos opciones ya sea cambiar el tipo de microcontrolador por uno de mayor capacidad y la otra es realizar la implementación de elementos adicionales al microcontrolador para que de esta forma se puedan anexar mas recursos al microcontrolador.

Unos de los casos mas sencillos y prácticos es la de implementar un decodificador, con el cual se permita direccionar dispositivos como displays de siete segmentos, alfanuméricos, matrices de leds, etc. Ver figura siguiente.



De igual forma se pueden ocupar circuitos extras para incrementar los puertos de entrada, como se muestra en la siguiente figura.



IV.3 CONVERTIDORES AD/DA

IV.4 ACOPLAMIENTO DE SEÑALES PARA ENTRADA Y SALIDA

IV.5 COMUNICACIÓN SERIAL

Saber en la Teoría (10 hrs.)

V

Proyecto de aplicación con Microcontroladores

Objetivo particular de la unidad
Integrar en un microcontrolador un control para un sistema o proceso sencillo.

Habilidades por desarrollar en la unidad
Escribir la habilidad propuesta que se debe desarrollar en esta asignatura.

V.1 APLICACIÓN PRACTICA CON UN MICROCONTROLADOR

Saber en la Teoría (0 hrs.)

Texto del contenido de la columna Saber (teoría) de la hoja de asignatura con desglose de unidades temáticas

Se desarrollará en esta sección, exclusivamente, la parte teórica que se indica. Utilizar este tipo de letra, tamaño y espacio entre párrafos. (arial, tamaño 11, justificado sin negrillas, interlineado sencillo)[1]

Saber Hacer en la practica (3 hrs.)

Texto del contenido de la columna Saber hacer (práctica) de la hoja de asignatura con desglose de unidades temáticas

Se desarrollará explicativamente la actividad de la práctica. Esto será un ejercicio o una práctica resueltos. Es importante que en esta parte el profesor enseñe a los alumnos como hacer o resolver la parte practica. Usar el mismo tipo de letra, tamaño y párrafo.[2]

Resolver las prácticas de la Guía de Prácticas de la asignatura.

[1] De esta forma se harán las referencias bibliográficas.
[2] De esta forma se harán las referencias bibliográficas.

Guía de Prácticas

Prácticas de la unidad 1

PRÁCTICA No. 1

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.
El alumno aprenderá a reconocer las diferencias existentes entre estos tipos de microcontroladores de 4, 8, 16 y 32 bits

2. Materiales y/o equipos.
Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Analizar la arquitectura interna de microcontroladores de 4,8,16 bits..
Analizar la arquitectura simplificada del Microcontrolador.
Analizar su arquitectura interna.

4. Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 1

PRÁCTICA No. 2

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno aprenderá a reconocer las diferencias entre las arquitecturas tipo CISC y RISC.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Analizar la arquitectura interna de los diferentes tipos de microcontroladores
Analizar la arquitectura de hardware y software de los Microcontrolador.

4. Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 1

PRÁCTICA No. 3

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno aprenderá a reconocer las diferencias existentes entre los microcontroladores y microprocesadores.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

- Analizar las prestaciones que presenta un microcontrolador contra un procesador.
- La fiabilidad
- La flexibilidad
- Las aplicaciones
- Los recursos en común que presentan

4. Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 2

PRÁCTICA No. 4

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno reconocerá los elementos que conforman al microcontrolador (registros, puertos E/S, temporizadores, interrupciones, etc.)

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

- Conocer los elementos específicos del microcontrolador.
- Analizar el funcionamiento de la arquitectura del microcontrolador y compararlo con el de una computadora.
- Ver el funcionamiento de los pines del micro.

4. Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 2

PRÁCTICA No. 5

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

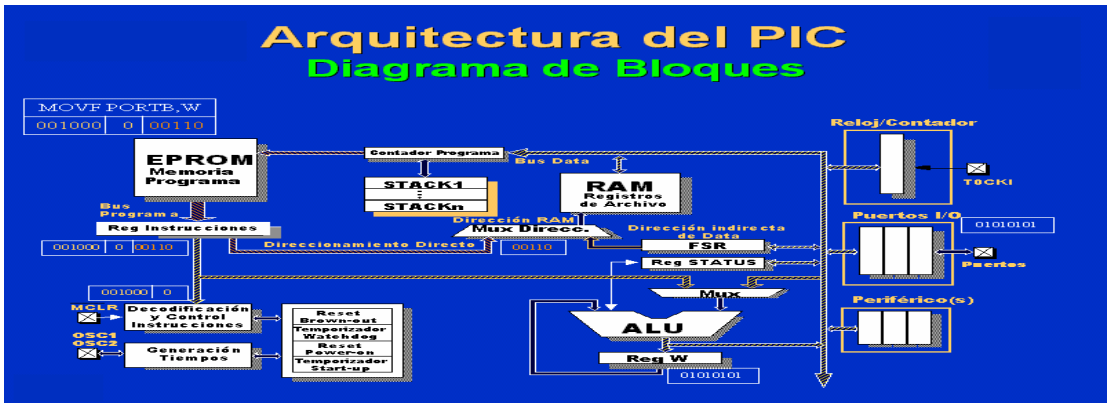
El alumno identificara físicamente la arquitectura interna del microcontrolador PIC.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

- Analizar las prestaciones que presenta un microcontrolador PIC e identificar el funcionamiento de sus partes que lo componen de acuero al siguiente diagrama.



4. Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 6

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno aprenderá la utilización de los diferentes nemónicos, operación y estados afectados de cada una de las instrucciones.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Conocer el juego de instrucciones del microcontrolador.
Comprensión del funcionamiento de las instrucciones con los modos de direccionamiento existentes.
Aplicar instrucciones aritméticas y lógicas.
Aplicar instrucciones para el control de programas.

4. Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 7

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno aprenderá a realizar la configuración de los puertos del microcontrolador así como el armado del circuito.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa el cual lea un numero en binario mediante unos interruptores conectados al puerto A, luego suma 2 unidades a ese valor visualizado y se visualiza. El resultado mediante 4 diodos led conectados al puerto B.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.

4. Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 8

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. **Objetivo.**

El alumno aprenderá a reconocer las aplicaciones que presta un microcontrolador, así como desarrollar sus habilidades en la elaboración de programas.

2. **Materiales y/o equipos.**

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. **Desarrollo general.**

Elaborar un programa el cual realice el encendido de un led al momento de presionar un botón y se mantendrá encendido solamente que el botón quede presionado. El led deberá de ir conectado a RA0 y el botón en RB0. Estructurar y analizar el propósito del programa.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa..

4. **Resultados y conclusiones de la práctica por parte del alumno.**

Prácticas de la unidad 3

PRÁCTICA No. 9

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno aprenderá a realizar el movimiento de datos de un puerto a otro puerto.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa el cual controle 5 interruptores conectados a través del puerto A y en el puerto B se visualiza la salida de los mismos datos

Por medio de unos leds conectados a este, encendiendo el correspondiente al interruptor.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 10

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno aplicara las subrutinas en el auxilio de manejo de tiempo de retardo con el manejo de mas datos de salida.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa en el que se realice una secuencia de leds encendiendo uno y al momento de apagarse encienda el otro led hasta que llegue al cuarto led se reiniciara empezando en el primer led.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 11

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno sabrá el uso de las instrucciones especiales de rotamiento de bits.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa que utilice las instrucciones RRF y RLF para realizar una secuencia de leds, encendiendo uno y al momento de apagarse encienda el otro led hasta que llegue al octavo led se reiniciara empezando en el primer led.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 12

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

Que el alumno desarrolle la utilización de instrucciones de chequeo de bits, para la realización de operaciones delimitadas.

2. Materiales y/o equipos.

Lápiz.
Cuaderno.
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Hacer un programa que por medio de la selección de tres push botón se seleccione una secuencia de leds con desplazamiento a la derecha, con otro botón el desplazamiento sea a la izquierda y con el ultimo botón se realizara el paro de la secuencia.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 13

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara la implementación de un proyecto y la carga de un programa en el software del simulador.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .asm
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 14

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara la interacción de dispositivos adicionales de visualización conectados al micro.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa el cual controle un display de 7 segmentos a través de la conexión directa con el microcontrolador. En el cual se visualice la cuenta de números hexadecimales (0 a la F), con un pequeño retardo entre los números.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 3

PRÁCTICA No. 15

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara la interacción de dispositivos adicionales conectados al micro y con la adición de dispositivos de potencia.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa el cual realice el control de un motor de corriente directa para hacer el control y la inversión de giro.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 16

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara la ampliación de la memoria del microcontrolador adicionando memorias externas.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa el cual controle una matriz de leds para la visualización de mensajes, ampliando la capacidad de memoria del micro. Adicionando una memoria externa la que contendrá los códigos para el control de la matriz de leds y el micro realizara el desplazamiento paulatino del mensaje.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 17

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara la interacción de dispositivos adicionales para incrementar el numero de terminales de los puertos de I / S.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa el cual controle un display de 7 segmentos a través de un decodificador 74LS47. En el cual se visualice la cuenta del 0 al 9 en el display con un pequeño retardo entre un número y el siguiente.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 18

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno identificara la forma de realizar el multiplexaje de información y se pueda controlar la visualización en dos displays.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar la decodificación de BCD a 7 segmentos por medio de software y multiplexar en él tiempo para dos dígitos de 7 segmentos.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 19

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara la interacción del microcontrolador con convertidores analógico – digital.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un programa el cual controle la variación de voltaje de 0 a 9 volts en un puente de wenston el que estará conectado a un convertidor A/D y este a su vez al microcontrolador. El microcontrolador manejara un decodificador que conectado a un display de 7 segmentos permitirá ver el voltaje de variación en el puente.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 20

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara la interacción del microcontrolador con convertidores digital – analógico.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un control para variar la velocidad de un motor de CD por medio de un DAC, conectado al microcontrolador. Las variaciones de velocidad se harán en base a software de programación y se empezara desde una velocidad mínima hasta una máxima.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 21

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno acoplara sensores y actuadores al microcontrolador, para la ejecución de un proceso determinado.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un control para la automatización de un estacionamiento en el cual para permitir el acceso a los vehículos se colocaran sensores de presencia e inductivos para detectar la presencia de un auto. Al momento de ser detectado el auto se activara un actuador que moverá una pluma para permitir el acceso. Pasando el umbral de la entrada se bajara automáticamente la pluma.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 22

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara el acoplamiento para la comunicación serial con una PC.
Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

2. Desarrollo general.

Realizar un programa par a realizar la comunicación serial con una PC y poder transmitir una serie de datos del microcontrolador hacia la PC. Los datos se almacenaran en la PC en base a un programa desarrollado en esta para recibir los datos

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 5

PRÁCTICA No. 23

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara el armado de un sistema mínimo para la operación del microcontrolador

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.
Placa protoboard
Microcontrolador
Capacitores de 22 pF
Cristal 4 Mhz
Push botón y resistencias

3. Desarrollo general.

Realizar el armado del sistema mínimo para la operación del microcontrolador .En el cual se pueda comprobar su operación ejecutando un programa que haga el encendido y apagado de leds conectados a los puertos del micro.

1. Estructurar y analizar el propósito del programa.
2. Realizar el diagrama de flujo o diagrama a bloques del programa.
3. En base a las instrucciones de programación desarrollar la implementación del programa.
4. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
5. Crear un archivo nuevo y teclear el programa realizado.
6. Guardar el archivo con su nombre respectivo en extensión .ASM.
7. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.
8. Grabar el programa en el microcontrolador.
9. Hacer el armado del sistema mínimo, colocando el micro ya programado.
10. Poner en marcha el circuito y comprobar la operación del programa.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 24

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

1. Objetivo.

El alumno desarrollara la interacción del microcontrolador con convertidores digital – analógico.

2. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

3. Desarrollo general.

Realizar un control para variar la velocidad de un motor de CD por medio de un DAC, conectado al microcontrolador. Las variaciones de velocidad se harán en base a software de programación y se empezara desde una velocidad mínima hasta una máxima.

4. Estructurar y analizar el propósito del programa.
5. Realizar el diagrama de flujo o diagrama a bloques del programa.
6. En base a las instrucciones de programación desarrollar la implementación del programa.
7. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
8. Crear un archivo nuevo y teclear el programa realizado.
9. Guardar el archivo con su nombre respectivo en extensión .ASM.
10. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Prácticas de la unidad 4

PRÁCTICA No. 20

Fecha		Grupo
No de alumnos por práctica	No. de alumnos por reporte	
Nombre y firma del profesor		
Nombre (s) del alumno (s)		
Tiempo estimado	Hrs	Calificación

11. Objetivo.

El alumno desarrollara la interacción del microcontrolador con convertidores digital – analógico.

12. Materiales y/o equipos.

Computadora
Software de simulación
Apuntes de clase.
Bibliografía adecuada.

13. Desarrollo general.

Realizar un control para variar la velocidad de un motor de CD por medio de un DAC, conectado al microcontrolador. Las variaciones de velocidad se harán en base a software de programación y se empezara desde una velocidad mínima hasta una máxima.

14. Estructurar y analizar el propósito del programa.
15. Realizar el diagrama de flujo o diagrama a bloques del programa.
16. En base a las instrucciones de programación desarrollar la implementación del programa.
17. Crear un nuevo proyecto en el software, el que contendrá todas las versiones del programa que se realice, dando el nombre con extensión .pjt.
18. Crear un archivo nuevo y teclear el programa realizado.
19. Guardar el archivo con su nombre respectivo en extensión .ASM.
20. Compilar el programa para revisar si no hay errores de sintaxis, si se presentan hacer la debida corrección.

Resultados y conclusiones de la práctica por parte del alumno.

Bibliografía

1 Escribir el Nombre del libro o manual
Escribir el Nombre del autor
Escribir la Editorial
Escribir la Edición

2 Escribir el Nombre del libro o manual
Escribir el Nombre del autor
Escribir la Editorial
Escribir la Edición

3 Escribir el Nombre del libro o manual
Escribir el Nombre del autor
Escribir la Editorial
Escribir la Edición

Etc, etc.