87

# UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
CAMPUS ARAGON

## ANALISIS FUNCIONAL DE MICROPROCESADOR Z-80 Y MICROCONTROLADOR PIC 16F84 PARA EL CONTROL DE UN MOTOR A PASOS.

**TRABAJO DE TESIS**

PARA OBTENER EL TITULO DE

**INGENIERO MECANICO ELECTRICO**

P R E S E N T A :

**ORTEGA OLIVARES OMAR GERARDO**

ASESOR: ING. PINEDA DIAZ ELEAZAR MARGARITO

MEXICO, D.F. 2002

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

SECRETARÍA ACADÉMICA

**Ing. RAÚL BARRÓN VERA**
**Jefe de la Carrera de Ingeniería Mecánica Eléctrica,**
**Presente.**

En atención a la solicitud de fecha 18 de febrero del año en curso, por la que se comunica que el alumno OMAR GERARDO ORTEGA OLIVARES, de la carrera de Ingeniero Mecánico Electricista, ha concluido su trabajo de investigación intitulado "ANÁLISIS FUNCIONAL DE MICROPROCESADOR Z-80 Y MICROCONTROLADOR PIC 16F84 PARA EL CONTROL DE UN MOTOR A PASOS", y como el mismo ha sido revisado y aprobado por usted, se autoriza su impresión; así como la iniciación de los trámites correspondientes para la celebración del Examen Profesional.

Sin otro particular, reitero a usted las seguridades de mi atenta consideración.

**A t e n t a m e n t e**
**"POR MI RAZA HABLARÁ EL ESPÍRITU"**
**San Juan de Aragón, México, 19 de febrero del 2002**
**EL SECRETARIO**

**LIC. ALBERTO IBARRA ROSAS**

C.p. Asesor de Tesis.
C.p. Interesado.

AIR/RCC/vr

Recibí el oficio 27-Feb-02

# ÍNDICE

CONTENIDO PAGINA

# INTRODUCCION

En ingeniería muchos de los sistemas digitales emplean microprocesadores o microcontroladores para controlar movimiento o desplazamientos todos estos sistemas utilizan motores a pasos, dependiendo de la aplicación pueden ser con uno o más motores a pasos.

Es por eso que siempre al diseñar un sistema digital se persiguen tres objetivos básicos que son: costo, tiempo y confiabilidad. Ya que siempre un sistema se puede mejorar, un análisis funcional de las componentes utilizadas es necesario para poder decidir en que circunstancias se va a resolver por un determinado sistema.

El presente trabajo de investigación tiene como finalidad analizar dos sistemas diferentes: uno con un microprocesador Z-80 y otro con un microcontrolador PIC 16F84 con la finalidad de orientar y describir que sistema es mas recomendable. Para los dos sistemas se utiliza un motor de imán permanente de 6 fases.

Se analizara la funcionalidad de dos sistemas mínimos uno con un microprocesador Z-80 y otro con un microcontrolador PIC 16F84, entre los cuales evaluaremos costo, confiabilidad tamaño a fin de determinar las ventajas y desventajas de cada uno da los dos sistemas de control

Esta tesis se ha dividido en cinco capítulos y dos apéndices para su mejor entendimiento, y son:

En el capitulo primero se da una descripción del microprocesador Z-80, de las características que da el fabricante sobre esta componente, configuración, funcionamiento, las componentes adicionales que utiliza, así como de la programación y del repertorio de instrucciones que este maneja

En el capitulo dos se mencionan las características que el fabricante proporciona para el microcontrolador Pic 16F84, así como su configuración, funcionamiento, programación y el conjunto de instrucciones.

En el capitulo tres se da una descripción de los tipos de motores a pasos que existen en el mercado, de los pulsos que se necesitan para ser manipulados, también se dan las características y diferencias entre ellos.

En el capitulo cuatro se da una descripción de cómo implementar un circuito de control con un microprocesador Z-80, para el funcionamiento de un motor a pasos, el programa que utiliza el microprocesador, así como la descripción de las componentes extras que son necesarias en el diseño, y el análisis de cada uno de los puntos del diseño.

En el capitulo cinco se da la descripción del circuito de control con un microcontrolador PIC 16F84, la programación del microcontrolador y el análisis de cada uno de los puntos del diseño

Al final se dan las conclusiones, propuestas para mejorar este trabajo de control con dispositivos programables. en los apéndices se proporcionan las descripciones de las instrucciones de cada una de las componentes utilizadas para su programación

# Capitulo 1

# CARACTERÍSTICAS DEL MICROPROCESADOR Z-80

## 1.1 QUE ES UN MICROPROCESADOR

Es un circuito integrado de alta escala de integración el cual esta constituido internamente por un CPU (unidad central de proceso) que realiza las operaciones de transferencia de datos, control, aritméticas, lógicas y tratamiento de las interrupciones mediante la ejecución de instrucciones obtenidas de la memoria (cualquier dispositivo que puede almacenar bits lógicos en estado de 0 ó 1 tales como un solo bit o grupos de bits).

Las terminales del microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la memoria y los módulos de entrada / salida 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. Figura 1.1 estructura de un sistema abierto basado en microprocesador.

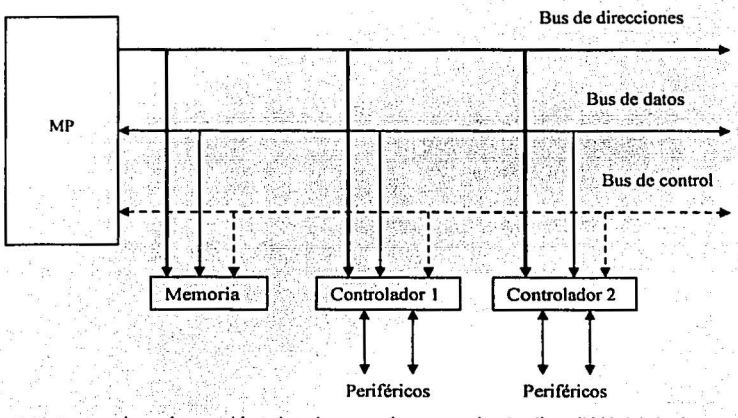



Figura 1.1 estructura de un sistema abierto basado en un microprocesador. La disponibilidad de los buses en el exterior permite que se configure a la medida de la aplicación.

## 1.2 DATOS DEL FABRICANTE.

Arquitectura del microprocesador Z-80 de 8 bits, se encuentran versiones mejores del mismo tales como Z80A, Z80B, Z80H, éstas se caracterizan por trabajar a frecuencias superiores de 4 MHz, 6.5 MHz y 8 MHz. Respectivamente las características fundamentales del Z-80 son:
El transporte de señales se realiza sobre tres buses, el bus de direcciones, el bus de datos, así como el bus de control.
Régimen de interrupción uniforme, con la posibilidad de encadenar las prioridades de los circuitos periféricos.
Alto grado de programabilidad.
Reloj único.
Fuente de voltaje única de +5 Volts.

Un sistema con Z80 se completa con el empleo de memorias estándar de lectura y memorias estáticas o dinámicas de lectura y escritura, además pertenecen al sistema, puertos de entrada y salida paralelo, interfaces de comunicación serie, sistemas contadores temporizadores y circuitos de acceso directo a memoria.

El funcionamiento del sistema consiste en que las instrucciones del microprocesador, que están en la memoria ROM, se ejecutan en una forma secuencial de operación, la fuente de datos es, la propia CPU, los periféricos o las memorias, la transferencia interna de datos es a través del CPU, exceptuando la transferencia de datos en el proceso de acceso directo a memoria.

El microprocesador Z-80 es una versión apreciablemente mejorada tanto en circuiteria como en características de programación del antiguo modelo INTEL 8080, el Z-80 resulta ser un microprocesador más rápido y sencillo en el desarrollo de sistemas ya que solo usa una fuente de alimentación de +5 Volts, contiene íntegramente todo el conjunto de instrucciones del 8080, lo cual le permite ejecutar todos los programas escritos para el CPU 8080, además contiene el Z-80 una expansión adicional de 80 instrucciones de ahí se deriva su nombre, su repertorio suma un total de 156 instrucciones.

El microprocesador Z80 contiene las siguientes unidades funcionales;

- Unidad aritmética y lógica
- El contador de programa
- El apuntador del stack (pila)
- Registros de propósito general
- Registros de índice
- Registros de interrupciones
- Registro de banderas
- Registro para refrescar memorias dinámicas

## 1.3 DIAGRAMA ESQUEMÁTICO

La figura 1.2 muestra la configuración y distribución de los pines del microprocesador Z-80

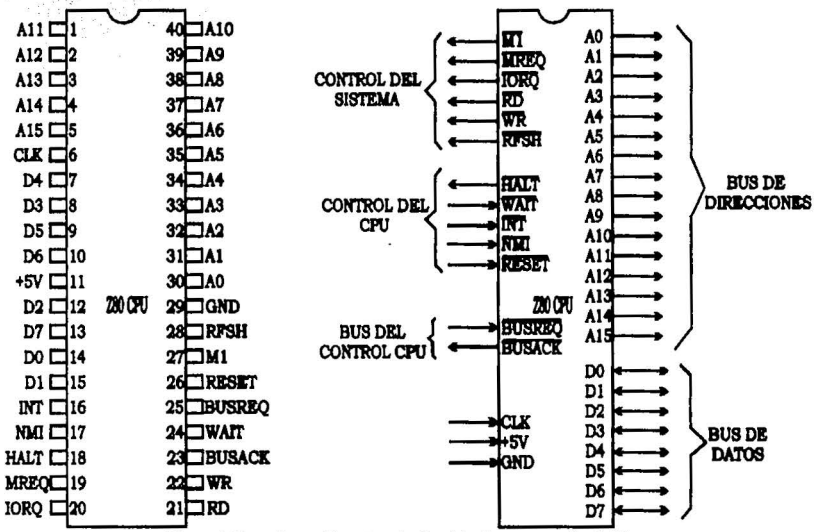



Figura 1.2 configuración y terminales del microprocesador Z-80.

El circuito integrado del microprocesador Z80 se deposita en un encapsulado de dos líneas (Dual In Line) de 40 terminales, de las cuales 5 son únicamente de entrada, 24 son de salida y 8 son bidireccionales, además, existen dos que sirven para conectar la fuente de alimentación y otra más que es la entrada del reloj.

Todas ellas tienen características eléctricas compatibles con la tecnología TTL, el intervalo de voltajes de entrada para un cero lógico es; 0.3 volts a 0.8 volts, el intervalo del voltaje de entrada para un uno lógico es; 2.0 volts a Vcc, el voltaje máximo de salida para un cero lógico es; Vsal = 0.4 Volts, y el voltaje mínimo de salida para un uno lógico es; Vsal = 2.4 Volts.

El microprocesador Z-80 opera con una fuente de alimentación de +5.0 Volts conectada a la terminal 11, (pin 11), la tierra de la fuente se conecta a la terminal GND, (pin 29), el consumo típico de corriente para el Z-80 es de 90 miliamperes.

Bus de Direcciones (A0-A15), pines 30-40 y 1-5, se forman con 16 líneas de direcciones, tienen la facultad de establecerse en tercer estado, estás señales proporcionan las direcciones correspondientes a intercambios de datos entre la memoria, la CPU y los puertos de los periféricos, la capacidad de direccionamiento con 16 bits es de 64 Kbytes y 256 puertos de entrada y salida, son activas en estado alto, los 8 bits menos significativos se usan para permitirle al usuario seleccionar los 256 puertos E / S, (A0-A7), en donde A0 es el bit menos significativo.

Líneas de datos (DO - D7), pines 13, 14, 15, 12, 7, 8, 9, y 10. Se forman con 8 líneas de datos bidireccionales con capacidad del tercer estado, son activas en nivel alto, se utilizan para el intercambio de datos con la memoria, y periféricos de E / S.

Requerimiento de E / S ($\overline{IORQ}$), pin 20. Es salida triestado activa en nivel bajo, está señal indica que la mitad baja del bus de direcciones mantiene una dirección válida de E / S, para efectuar una operación de lectura o escritura de E / S, se genera está señal cuando el ciclo de maquina 1 (M1) reconoce una interrupción, indican que el vector de respuesta de la interrupción se coloca en el bus de datos, las operaciones de reconocimiento de interrupción ocurre durante el ciclo de maquina 1, mientras que las operaciones de E / S nunca se producen durante este ciclo.

Ciclo de maquina uno ($\overline{M1}$), pin 27. Salida activa en nivel bajo, indica que en este ciclo de máquina uno el microprocesador va a obtener el código operacional de una instrucción, en las instrucciones que tienen un código operacional de 2 bytes está señal se opera al obtener cada uno de los bytes del código operacional, al igual que para indicar el reconocimiento de un ciclo de interrupción cuando ocurre IORQ'.

Requerimiento de memoria ($\overline{MREQ}$), pin 19. Salida activa en nivel bajo, está señal indica una petición que interrelaciona a la memoria con la CPU, obtiene una dirección valida de las líneas de direccionamiento, está terminal tiene capacidad del tercer estado.

Lectura ($\overline{RD}$), Pin 21. Salida triestado activa en nivel bajo, indica que la CPU desea leer datos desde la memoria de un dispositivo externo de E / S, el dispositivo E / S se direcciona a la memoria o al periférico, se usa está terminal para dirigir los datos al bus de datos de la CPU.

Escritura ($\overline{WR}$), pin 22. Salida triestado activa en nivel bajo, indica que el bus de datos de la CPU va a obtener datos válidos para ser almacenados en la memoria o en algún dispositivo de E / S.

Refresco de la memoria dinámica ($\overline{RFSH}$), pin 28. Salida activa en nivel bajo, indica que los siete bits inferiores de las líneas de direccionamiento contienen una dirección válida de refresco de memoria, se utiliza para el mantenimiento de datos en memorias dinámicas, con está se efectúa una lectura de refrescamiento para todas las memorias dinámicas.

Paro ($\overline{\text{HALT}}$), pin 18. Salida que activa en nivel bajo, indica que la CPU realiza una instrucción por software de paro (HALT), y que espera una interrupción (NMI) o (INT) antes de que continuara la operación, mientras permanezca en este estado la CPU ejecuta operaciones NOP, para mantener activo el refresco de las memorias dinámicas, al aplicarse un reset se continua con la operación.

Espera ($\overline{\text{WAIT}}$), pin 24. Es una entrada activa en nivel bajo, le indica al microprocesador que la memoria direccionada o los dispositivos periféricos de E / S no son tan rápidos como para realizar una transferencia de datos a la velocidad de la CPU, o no están listos para una transferencia de información, la CPU continua con el estado de espera durante todo el tiempo que está terminal es activa, esto les permite a los otros dispositivos sincronizarse con la CPU.

Requisición de interrupción mascarable ($\overline{\text{INT}}$), pin 16. Entrada activa en nivel bajo, está terminal se acciona con dispositivos E / S externos, una requisición (INT) se atiende al final de la instrucción que se ejecuta, si el enable interno del Flip Flop de interrupción IFF1 controlado por software se encuentra habilitado, y si la requisición de bus no está activa, al aceptar la CPU una interrupción envía una señal de reconocimiento, la petición de E / S se realiza durante el ciclo de máquina 1, al principio del siguiente ciclo de instrucción, está petición solo es valida bajo control del programa interno, reconociendo la CPU tres modos diferentes de interrupción.

Interrupción no mascarable ($\overline{\text{NMI}}$), pin 17. Entrada que se activa con un flanco de bajada mediante un impulso que identifica una interrupción obligada, posiciona al contador de programa (PC) en la dirección 0066h desde donde continua el proceso, está tiene una prioridad más alta que la interrupción (INT) y siempre se reconoce al final de la instrucción que se ejecuta, independientemente del estado del Flip Flop IFF1, el contador de programa PC se almacena automáticamente en el stack pointer (puntero de pila) externo de forma que el usuario regrese al programa en el mismo punto del que fue interrumpido.

Rehabilitación ($\overline{\text{RESET}}$), pin 26. Entrada que se activa con un flanco de bajada mediante un impulso, obliga a la CPU a reiniciar su actividad, coloca al contador de programa (PC) en la localidad de inicio de memoria 0000h, desde donde empieza el proceso, durante este tiempo el bus de direcciones y el bus de datos adquieren el estado de alta impedancia y todas las terminales de control de salida adquieren el estado inactivo.

Entrada del bus de control del CPU ($\overline{\text{BUSRQ}}$), pin 25. Esta entrada se activa en nivel bajo, le indica a la CPU que coloque todas sus líneas en estado de alta impedancia, (tan pronto el ciclo de maquina 1 actual termine), a petición del periférico externo que desea tomar el control del sistema, regresa el control a la CPU cuando está señal (BUSRQ)' pasa al nivel alto, se utiliza para pedir que el bus de direcciones, el bus de datos y las terminales de salida triestado del bus de control vayan a un estado de alta impedancia de tal forma que otros dispositivos controlen esos buses.

Salida hacia el bus de control del CPU ($\overline{BUSAK}$)', pin 23. Salida activa en nivel bajo, es una indicación para el periférico que efectúa una petición (BUSRQ)' de que su petición ha sido concedida por parte del microprocesador, sirve para indicar al dispositivo que solicita este reconocimiento, que el bus de direcciones, el bus de datos y el bus de las terminales de control triestado han sido puestos en su estado de alta impedancia y que el dispositivo externo puede ahora controlar estás terminales.

Reloj (CLK), Pin 6. Entrada configurada por un tren de impulsos útiles, es la diferencia que permite la secuencia de tiempos de operación, se implanta físicamente con un oscilador de onda cuadrada cuya frecuencia depende del tipo de características de la CPU Z80, requiere oscilación de una fase con niveles TTL, una forma de satisfacer todos los requerimientos de voltaje es por medio de una resistencia de activación "pull up" de 330 ohms conectada entre +Vcc y la terminal de salida de un oscilador implantado con circuitos TTL que generen oscilaciones.

## 1.4. DIAGRAMA A BLOQUES

Este diagrama se muestra en la figura 1.3 donde se observa los 7 módulos.

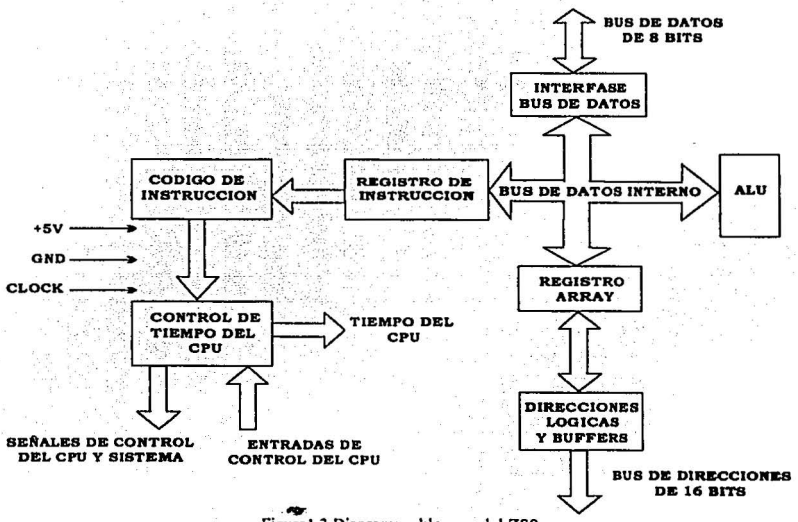



Figura1.3 Diagrama a bloques del Z80

La unidad aritmética y lógica (ALU), las operaciones del CPU Z80 se realizan con un grupo de dispositivos lógicos conocidos comúnmente como unidad aritmética y lógica (ALU) está efectúa las siguientes operaciones;

- Suma binaria.
- Operaciones lógicas.
- Complementar a dos.
- Corrimiento de un bit a la derecha o a la izquierda.
- Registro de resultados importantes como el acarreo, signo, acarreo auxiliar, paridad o si el resultado es cero.
- Comparaciones.
- Poner, Limpiar o probar un bit.

El contador de programa (PC), es un registro de 16 bits, que continuamente tiene la dirección de la localidad de memoria siguiente que se va a accesar, de esa localidad obtiene el código de la instrucción a ejecutarse, en la CPU el PC se incrementa en uno, cada vez que el microprocesador lee el código de la instrucción contenida en la localidad direccionada, de está forma el contador del programa direcciona secuencialmente las localidades de la memoria ROM, donde se encuentra almacenado el programa.

El apuntador de pila o stack (SP), el microprocesador Z80 cuenta con el registro de pila (SP) o stack pointer que contiene una dirección de memoria RAM a partir de la cual y en forma descendente, se almacenan los contenidos de un par de registros, o a partir del cual en forma ascendente se obtienen los últimos dos datos de 8 bits almacenados en esa área, el SP es un registro de 16 bits, para almacenar en el stack el contenido de un par de registros se utiliza la instrucción PUSH y para cargar a un par de registros con los dos últimos bytes del stack se utiliza la instrucción POP.

Registros de propósito general. El microprocesador Z-80 contiene 14 registros de 8 bits separados en dos grupos;

- GRUPO 1; A, B, C, D, E, H, y L.
- GRUPO 2; A', B', C', D', E', H' y L.

Todas las instrucciones trabajan con los registros del grupo 1, con las instrucciones EX y EXX se logra el intercambio entre los contenidos de los registros del grupo 1 con los contenidos de los registros del grupo 2, el grupo 2 se utiliza en cierta forma como stack del grupo 1, dentro de la propia CPU.

Con los 14 registros de propósito general se efectúan por medio de las instrucciones las siguientes funciones;

- Recibir datos desde la memoria.
- Enviar datos hacia la memoria.
- Incrementar o decrementar en uno su contenido.
- Formar una dirección con el contenido de un par de registros.
- Transferir datos entre los registros.
- Obtener un operando durante las funciones de la ALU.

Registros de índice IX e IY. Estos son registros de 16 bits cada uno y conservan direcciones base que se usan para modo de direccionamiento indexado, en este modo un registro de índice se usa como base para apuntar a una región de la memoria. La dirección efectiva de la localidad de memoria a donde se va a depositar el dato o de donde se va a leer, se obtiene, al sumar el contenido del registro de índice y el valor de 8 bits contenido en el campo de "desplazamiento" de las instrucciones que emplean direccionamiento con índice, estos desplazamientos se especifican con números enteros signados con el complemento a dos.

Registro de interrupciones. El microprocesador Z-80 opera en modo de interrupción en el que responde como una llamada indirecta en respuesta a una solicitud de interrupción. El registro I se usa para este propósito almacenando los 8 bits más significativos de la dirección indirecta mientras que el dispositivo que interrumpe proporciona los 8 bits menos significativos de la dirección índice, está característica permite que las rutinas servicio de las interrupciones se localicen en cualquier parte de la memoria y que se puedan accesar en un tiempo muy corto.

Banderas de estado, El microprocesador Z-80 tiene un registro de 8 Flips Flops, para monitorear ciertos resultados de las operaciones de la ALU, a la información que almacenan estos Flips Flops se conoce como banderas de estado, las banderas se actualizan después de cada operación con alguno de los registros, no todas las operaciones modifican a todas las banderas, de los 8 bits del registro de banderas, únicamente seis registran información útil para el programador, cuatro de estás banderas se prueban, esto es, se usan como condiciones de salto (JP), llamada (CALL), o regreso (RET), estás banderas son;

La paridad o sobre flujo (P / V), el registro P se utiliza para realizar funciones auxiliares necesarias para el usuario, le sirven para interpretar los resultados, es uno cuando el resultado de la operación lógica del complemento a dos produce un acarreo, de otro forma es un cero lógico.

La señal (S), muestra si en el resultado de funcionamiento de la ALU el bit más significativo es igual a 1, si no se produce restablezca.

Zero (Z), muestra si el resultado de la operación es cero en la ALU de otro forma ocurre un reset.
Acarreo auxiliar ocurre cuando del resultado del bit 3 está dentro del bit4 de otra forma ocurre un reset.

Carry (C), muestra si el resultado de las operaciones de suma o de la resta tomara el bit de mas alto orden, de otra forma ocurre un reset

Ciclo de lectura o escritura a la memoria, la figura 1.4 muestra los ciclos de escritura o de lectura de la memoria en el microprocesador Z-80

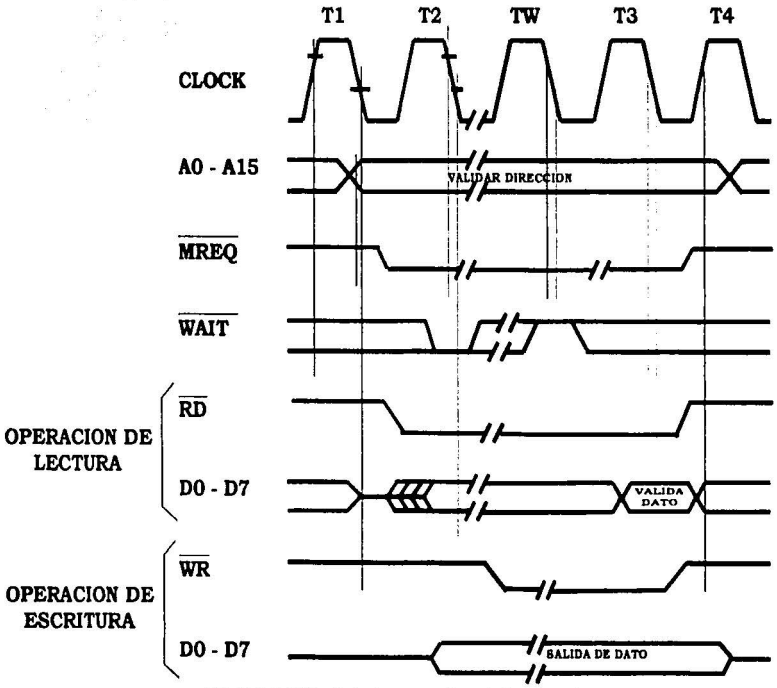



Figura 1.4. Ciclo de lectura o escritura de la memoria

Ocupa 3 ciclos de reloj. Las señales MREQ y RD se usan de la misma forma que en el ciclo de búsqueda.

En el caso de escritura la señal MREQ también se activa en nivel lógico bajo (0) cuando el bus de direcciones está estable para que pueda ser usado directamente como chip select de las memorias. La señal WR se activa en nivel lógico bajo (0) cuando el dato sobre el bus de datos está estáble para que pueda usarse directamente como pulso de R/W.

Ciclo de Lectura o Escritura a Dispositivos I / O, la figura 1.5. Muestra el ciclo de lectura o escritura de un dispositivo de entrada o salida

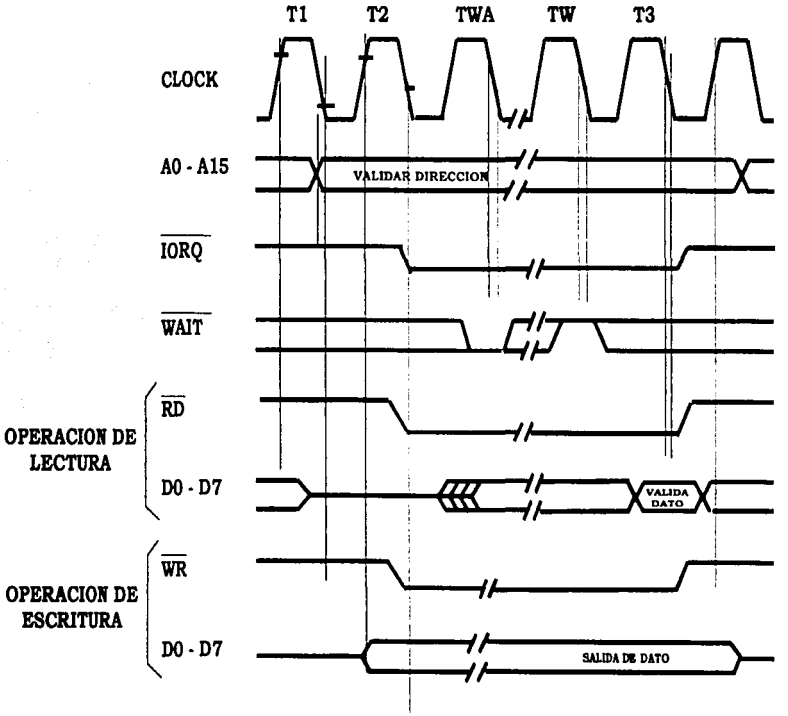



Figura 1.5. Ciclo de lectura o escritura de un dispositivo de salida o entrada

Durante las operaciones I / O se inserta automáticamente un estado de espera, ya que el tiempo desde que la señal IORQ se activa en nivel lógico bajo (0) hasta que el Z80 deba samplear la entrada WAIT es muy corto y sin este estado de espera extra no hay suficiente tiempo para que un I / O decodifique su dirección y mande un WAIT.

Requerimiento de interrupción (Interrupt Request / Acknowledge) (INT), En la figura1.6 se muestra el ciclo de requerimiento de interrupción INT

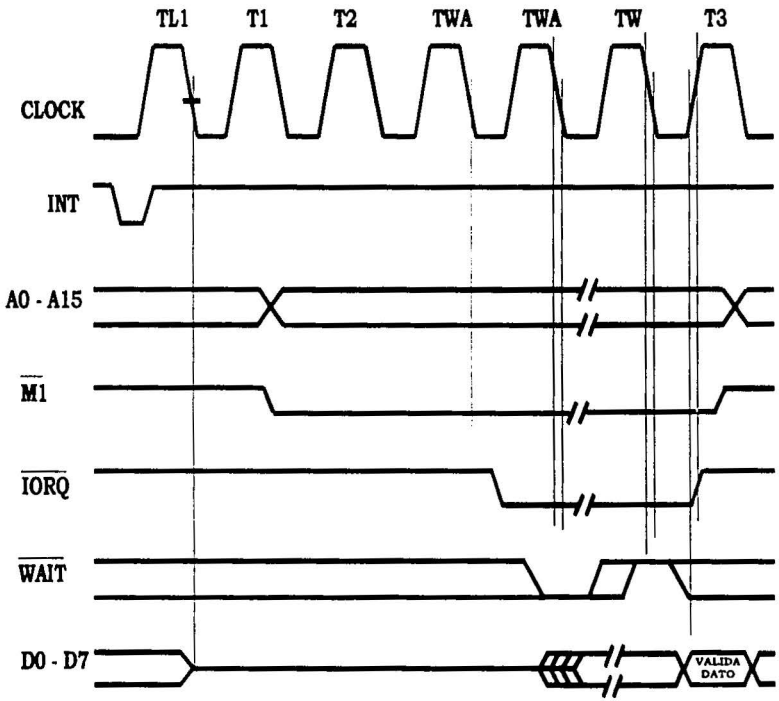



Figura 1.6 ciclo de requerimiento de interrupción

La señal de INT sé muestrea en el flanco positivo del último pulso de clock de una instrucción.
Cuando se acepta la señal se genera un ciclo M1 especial durante el cual la señal IORQ pasa a un nivel activo (0) en lugar de la señal MREQ en un ciclo M1 normal para indicar que el periférico que interrumpió puede poner un vector de 8 bits en el bus de datos.

Se agregan automáticamente 2 estados de wait para que el periférico pueda poner el vector y la estructura de prioridades decida quién interrumpe (está pensado para daisy-chain de 4 periféricos). También tiene un ciclo de refresco.

Requerimiento de interrupción (Request / Acknowledge) $\overline{NMI}$, en la figura1.7 se muestra el ciclo de interrupción NMI.

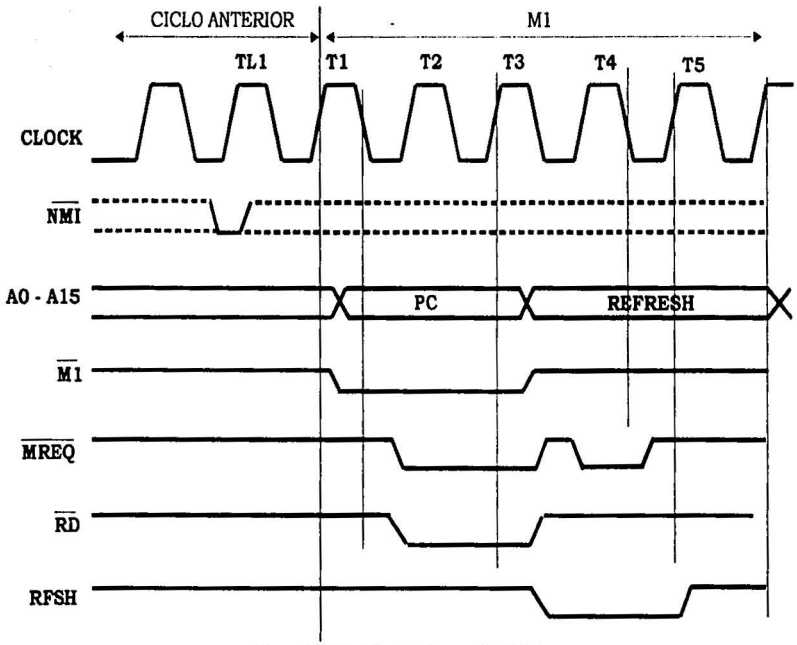



Figura1.7. El ciclo de interrupción NMI.

Se muestrea igual que la INT pero tiene más prioridad. La respuesta del Z80 es similar a un ciclo de lectura a memoria normal con la única diferencia de que el contenido del bus de dato se ignora mientras el Z80 guarda automáticamente el PC en la pila y salta a la dirección 0066 donde debe estar ubicado el comienzo de la rutina de atención.

## 1.5 Instrucciones de Programación

El procesador Z-80 posee un conjunto de instrucciones en las cuales podemos realizar diferentes operaciones como son mover bloque de bits, transferir datos entre la memoria entre la memoria dispositivos de entrada y salida. Las instrucciones están compuestas por nemónicos los cuales hay que pasar a lenguaje ensamblador y que se dividen en las siguientes categorías:

- Carga de 8 bits
- Carga de 16 bits
- Intercambio de registros y búsqueda
- Operaciones lógicas y aritméticas de 8 bits
- Aritmética de propósito general y control de CPU
- Operaciones aritméticas de 16 bits
- Rotación y cambio
- Prueba de bit y reset
- Llamadas, retornos y restaurar.
- Operaciones de entrada y salida

En el apéndice 1 se describen cada una de las instrucciones del microprocesador Z-80, así como su nemónico número de ciclos, símbolo de operación, código de operación

# Capitulo 2

# CARACTERISTICAS DEL MICROCONTROLADOR PIC 16F84

## 2.1 QUE ES UN MICROCONTROLADOR.

Es un circuito integrado programable que contiene todos los componentes de un computador. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño suele ir incorporado en el propio dispositivo que lo gobierna.

El microcontrolador es un computador dedicado. En su memoria solo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada y salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar y todos los recursos complementarios disponibles tienen como única finalidad atender sus requerimientos. Una vez programado y configurado el microcontrolador solo sirve para gobernar la tarea asignada.

## 2.2 DATOS DEL FABRICANTE.

El PIC 16F84 es un microcontrolador con memoria de programa tipo FLASH, lo que representa gran facilidad en el desarrollo de prototipos y en su aprendizaje ya que no se requiere de borrado con luz ultravioleta como las versiones EPROM sino, permite reprogramarlo nuevamente sin ser borrado con anterioridad. Por está razón, lo usaremos en la mayoría de aplicaciones que se desarrollan a lo largo del estudio.

Este microcontrolador se basa en la Arquitectura Harvard, representado en la figura 2.1, dispone de dos memorias:

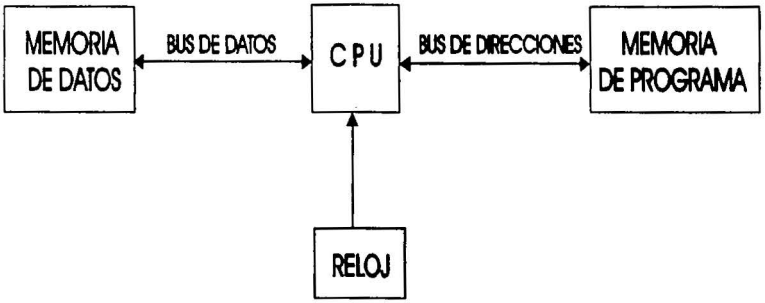



Figura 2.1 arquitectura harvad

- Memoria de datos.
- Memoria de Programa.

En la cual el programa y los datos se pueden trabajar desde memorias separadas, lo que posibilita que las instrucciones del programa y los datos tengan longitudes diferentes. Esta misma estructura es la que permite la superposición de los ciclos de búsqueda y ejecución de las instrucciones, lo cual se ve reflejado en una mayor velocidad del microcontrolador.

Además cada memoria dispone de su respectivo bus, lo que permite, que la CPU pueda accesar de forma independiente y simultánea a la memoria de datos y a la de instrucciones. Como los buses son independientes éstos pueden tener distintos contenidos en la misma dirección.

El CPU es un procesador tipo RISC (Computadores con un juego de instrucciones reducido), las instrucciones son muy simples y suelen ejecutarse en un ciclo de máquina. Además los RISC deben tener una estructura pipeline y ejecutar todas las instrucciones a la misma velocidad.

Procesador segmentado "pipe-line": quiere decir que aplica la técnica de segmentación que permite al procesador realizar simultáneamente la ejecución de una instrucción y la búsqueda de código de la siguiente. De esta manera, se puede ejecutar una instrucción en un ciclo. Cada ciclo de instrucción son cuatro ciclos de reloj.

En el momento de programar el microcontrolador, el interruptor de selección del temporizador de arranque (Power Up Timer) trabaja de forma inversa, es decir, se selecciona la opción "Low" para activarlo.

Los puertos son el puente entre el microcontrolador y el mundo exterior. Son líneas digitales que trabajan entre cero y cinco voltios y se pueden configurar como entradas o como salidas.

## 2.3 CONFIGURACION

La figura 2.2 muestra la distribución de los pines del microcontrolador PIC 16F84

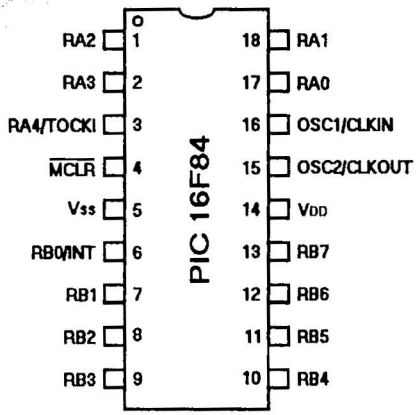



Figura 2.2. Muestra la distribución de los pines del microcontrolador PIC 16F84

El PIC 16F84 tiene dos puertos. El puerto A con 5 líneas y el puerto B con 8 líneas. Cada pin se puede configurar como entrada o como salida independiente programado por un par de registros diseñados para tal fin. En ese registro un "0" configura el pin del puerto correspondiente como salida y un "1" lo configura como entrada.

### 2.3.1. EL PUERTO A

Pin 17, RA0 = Pin de Entrada / salida compatible con TTL.
Pin 18, RA1 = Pin de Entrada / salida compatible con TTL.
Pin 1, RA2 = Pin de Entrada / salida compatible con TTL.
Pin 2, RA3 = Pin de Entrada / salida compatible con TTL.
Pin 3, RA4/TOCKI = Pin de Entrada / salida o entrada de Reloj Externo para el TMR0, cuando este pin se configura como salida es de tipo Open Drain (ST), cuando funciona como salida se debe conectar a Vcc (+5V) a través de una resistencia.

El pin RA4/TOCKI del puerto A puede ser configurado como un pin de entrada / salida como se mencionaba anteriormente o como entrada del temporizador / contador. Cuando este pin se programa como entrada digital, funciona como un disparador de Schmitt (Schmitt trigger, ST), esto quiere decir que puede reconocer señales un poco distorsionadas

y llevarlas a niveles lógicos (cero y cinco volts), cuando se usa como salida digital se comporta como colector abierto, por lo tanto se debe poner unas resistencias de pull-up (resistencia externa conectada a un nivel lógico de cinco volts), como salida, la lógica es inversa un 0 escrito al pin del puerto entrega en el pin un 1 lógico. Además como salida no puede manejar cargas como fuente, solo en el modo sumidero.
Como este dispositivo es de tecnología CMOS, todos los pines deben estar conectados a alguna parte, nunca dejarlos al aire por que se puede dañar el integrado. Los pines que no se estén usando se deben conectar la fuente de alimentación +5V con una resistencia de menor que 5 Kilo Ohmos.

## 2.3.2. PUERTO B

Pin 6 RB0/INT = Pin de Entrada / salida o entrada de interrupción externa. (TTL/ST).
Pin 7, RB1 = Pin de Entrada / Salida compatible con TTL.
Pin 8, RB2 = Pin de Entrada / salida compatible con TTL.
Pin 9, RB3 = Pin de Entrada / Salida compatible con TTL.
Pin 10, RB4 = Pin de Entrada / Salida con Interrupción por cambio de Flanco compatible con TTL.
Pin 11, RB5 = Pin de Entrada / Salida con Interrupción por cambio de Flanco compatible con TTL.
Pin 12, RB6 = Pin de Entrada / Salida con Interrupción por cambio de Flanco compatible con TTL / ST.
Pin 13, RB7 = Pin de Entrada / salida con Interrupción por cambio de Flanco compatible con TTL /ST.
El Puerto B tiene Internamente unas resistencias de pull-up conectadas a sus pines (sirven para fijar el pin a un nivel de cinco voltios), su uso puede ser habilitado o deshabilitado bajo control del programa. Todas las resistencias de pull-up conectan o desconectan a la vez. La resistencia de pull-up es desconectada automáticamente en un pin si este se programa como salida. El pin RB0/INT se puede configurar por software para que funcione como interrupción externa.

Pin 4, MCLR = Pin de Reset del Microcontrolador (Master Clear). Se activa (el pic sé resetea) cuando tiene un "0" lógico en su entrada.
Pin 5, Vss = Ground o Tierra
Pin 14, VDD = Fuente Positiva (+5V)
Pin 15, OSC2/CLKOUT = Entrada del Oscilador del Cristal. Se conecta al Cristal o Resonador en modo XT (Oscilador de Cristal). En modo RC (Resistencia-Condensador), este pin actúa como salida el cual tiene 1/4 de la frecuencia que entra por el pin OCS1/CLKIN.
Pin 16, OSC1/CLKIN = Entrada del Oscilador del Cristal / Entrada de reloj de una Fuente Externa.

La máxima capacidad de corriente de cada uno de los pines de los puertos en modo sumidero (sink) es de 25 mA y en modo fuente (source) es de 20 mA.

El consumo de corriente del microcontrolador para su funcionamiento depende del voltaje de operación, la frecuencia y de las cargas que tengan sus pines.
Por Ejemplo: Para un reloj de 4 MHz el consumo es de aproximadamente de 2mA; aunque este se puede reducir a 40 microamperios cuando está en el modo sleep (en este modo el micro se detiene y disminuye el consumo de potencia). Se sale de este estado cuando se produce alguna condición

## 2.4 DIAGRAMA A BLOQUES.

En la figura 2.3. Se muestra el diagrama a bloques del microcontrolador PIC 16F84.

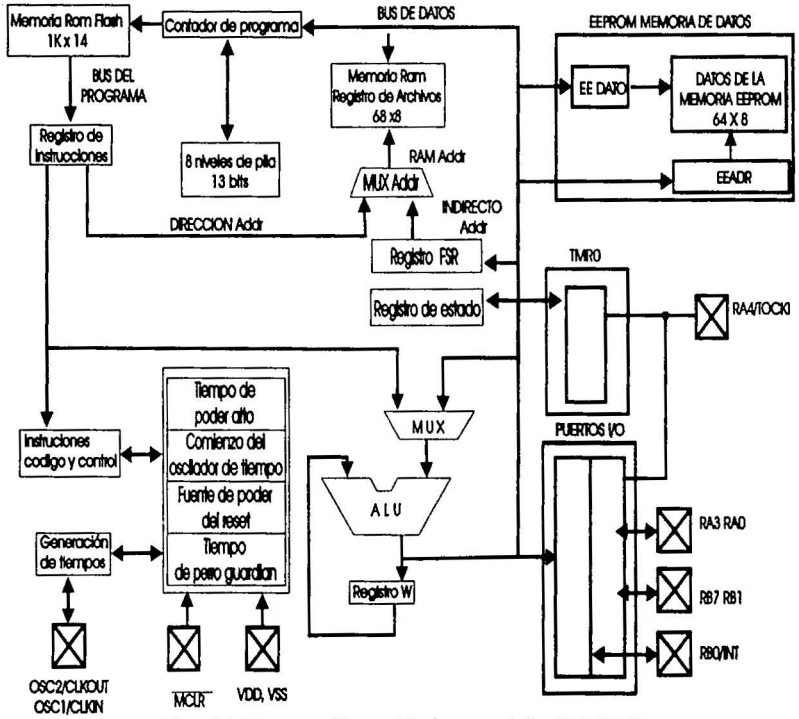



Figura 2.3. Diagrama a bloques del microcontrolador PIC16F84F

## 2.4.1 MEMORIA DE PROGRAMA

Es una memoria de 1 Kbyte de longitud con palabra de 14 bits. Como es del tipo FLASH se puede programar y borrar eléctricamente, en otras palabras, se puede programar o borrar sin necesidad de un borrador de luz ultravioleta, lo que facilita el desarrollo de programas y la experimentación. Como el PIC 16F84 tiene un contador de programa de 13 bits, tiene una capacidad de direccionamiento de 8K x 14, pero solamente tiene implementado el primer 1K x 14 (000h hasta 03FFh). Si se direccionan posiciones de memoria superiores a 3FFh se causará un solapamiento o desborde con el espacio del primer 1K. Figura 2.4 mapa de memoria de programa

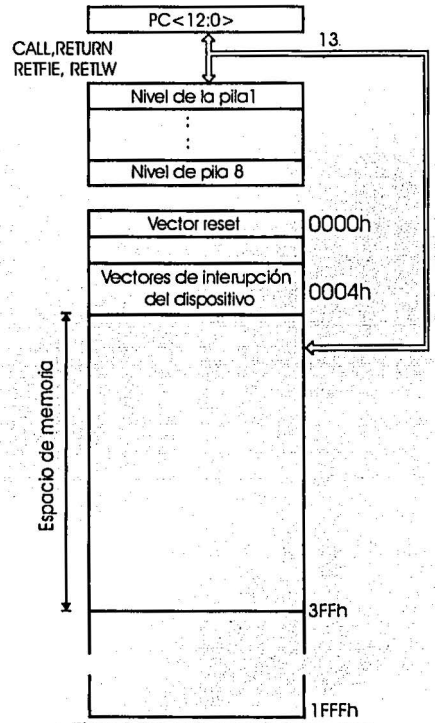



**Figura 2.4 mapa de memoria de programa**

## 2.4.2 REGISTROS (MEMORIA RAM DE DATOS)

El PIC 16F84 puede direccionar 128 posiciones de memoria RAM, pero solamente tiene implementado fisicamente los primeros 80 (0 a 4Fh). De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 68 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se está ejecutando. Los registros están organizados como dos bancos (paginas) de 128 posiciones de 8 bits cada una (128 x 8); todas las posiciones se pueden accesar directa o indirectamente (estás ultimas a través del registro FSR). Para seleccionar que pagina de registro se trabaja en un momento determinado se utiliza el bit RP0 del registro de estado. En la figura 2.5 se muestra el mapa de memoria RAM de datos

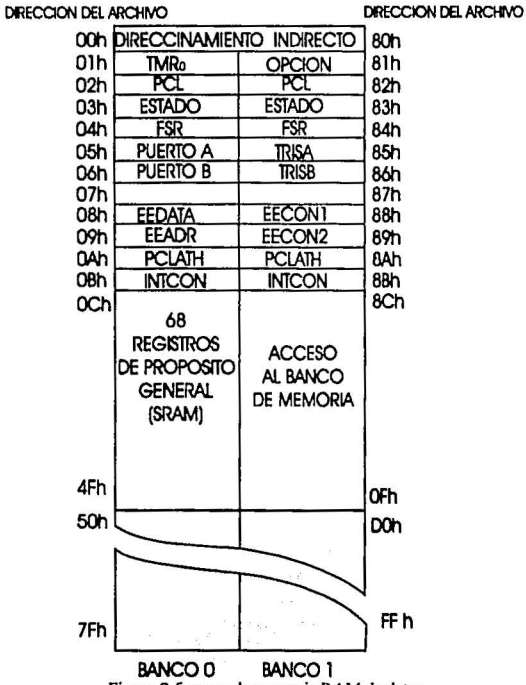


Figura 2.5 mapa de memoria RAM de datos

Vector de reset, cuando ocurre un reset o se enciende el microcontrolador, el contador de programa se pone en ceros (000h). Por esta razón, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo.

El registro 00h o INDF para el direccionamiento indirecto de datos. Este no es un registro disponible físicamente; utiliza el contenido del FSR y el bit RP0 del registro estado para seleccionar indirectamente la memoria de datos o RAM del usuario; la instrucción determinara que se debe señalar con el registro señalado. En la figura 2.6. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 00h | INDF | Contenido utilizado de FSR a la dirección de memoria de datos (no es un registro físico) | | | | | | | |

Figura 2.6. Registro de direccionamiento directo de datos

El registro 01h o TMR0 Temporizador / contador de 8 bits. Este se puede incrementar con una señal externa aplicada al pin RA4/TOCK1 o de a cuerdo a una señal interna proveniente del reloj de instrucciones del microcontrolador. La rata o tasa de incremento del registro se puede determinar por medio de un pre-escalador, localizado en el registro opción. Los anteriores microcontroladores no contaban con la generación de una interrupción cuando se rebasaba la cuenta (el paso de 0FFh a 00h). En la figura 2.7. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 01h | TMR0 | Reloj / Contador de 8 bit en tiempo real | | | | | | | |

Figura 2.7. Temporizador / contador de 8 bits

El CONTADOR DE PROGRAMA es el registro 02h o PCL. Se utiliza para direccionar las palabras de 14 bits del programa del usuario que se encuentra almacenado en la memoria ROM; este contador tiene un tamaño de 13 bits. Sobre el byte bajo, se puede escribir o leer a voluntad directamente, mientras que en el byte alto, no. El byte alto se maneja mediante el registro PCLATH (0Ah). A diferencia de los PIC de primera generación el 16F84 ante una condición de reset inicia el contador de programa con todos sus bits en "cero". Durante la ejecución normal del programa, y dado que todas las instrucciones ocupan solo una posición de memoria, el contador se incrementa con cada instrucción, a menos que se trate de alguna instrucción de salto. En la figura 2.8. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 02h | PCL | 8 bit más bajos del Contador de Programa | | | | | | | |

Figura 2.8. Contador de programa

El REGISTRO DE ESTADO es el 03h Y 83h o STATUS:. Contiene el estado Aritmético de la ALU, la causa de reset y los bits de preselección de pagina para la memoria de datos. En la figura 2.9. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 03Hy 83H | STATUS | IRP | RP1 | RP0 | T0# | PD# | Z | DC | C |

bit 0 C : flag (O) de acarreo en el octavo bit
1: Acarreo en la suma y no en la resta
0: Acarreo en la resta y no en la suma
Este bit también se utiliza en las instrucciones de rotación
bit 1 DC: flag (O) de acarreo en el 4º bit de menos peso.
1: Acarreo en la suma
0 : No acarreo en la suma. En la resta lo contrario
bit 2 Z: flag (O) de cero
1: el resultado de la última operación aritmética o lógica es cero
0: El resultado de la última operación es distinto de cero
bit 3 PD#: flag (O) Power Down
1: Tras conectar VDD o al ejecutar la instrucción CLRWDT
0: Al ejecutar la instrucción SLEEP
bit 4 TO#: flag (O) Timer Out
1:Tras conectar VDD o ejecutar CLRWDT o SLEEP
0 :Al desbordar el temporizador de WDT
bit 5-6 RP<1:0>:Selección del banco para el direccionamiento directo
00 Banco 0 (00h-7Fh)
01 Banco 1 (80h-FFh)
10 Banco 2 (100h-17Fh)
11 Banco 3 (180h-1FFh)
bit 7 IRP: Selección de bancos para el direccionamiento indirecto
1: el resultado de la última operación aritmética o lógica es cero
0: Bancos 0 y 1 (00h-FFh)

Figura 2.9. Registros de estado

Los bits 5 y 6 (RP0 y RP1) son los bits de selección de pagina (Bank 0 y Bank 1), para el direccionamiento directo de la memoria de datos; solamente RP0 se usa en los PIC 16F84. RP1 se puede utilizar como un bit de propósito general de lectura / escritura. Los bits TO y

PD no se pueden modificar por un proceso de escritura; ellos muestran la condición por la cual se ocasiono el ultimo reset.

Vector de interrupción, cuando el microcontrolador recibe una señal de interrupción el contador de programa apunta a la dirección 04h de la memoria de programa, por eso allí se debe escribir toda la programación necesaria para atender dicha interrupción.

El registro FSR es el SELECTOR DE REGISTROS con valor 04h. En asociado con el registro INDF, se utiliza para seleccionar indirectamente los otros registros disponibles. Mientras que los antecesores del PIC 16F84 solo poseían 5 bits activos, en este microcontrolador se poseen solo 8 bits. Si en el programa no se utilizan llamadas indirectas, este registro se puede utilizar como un registro de propósito general. En la figura 2.10. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 04h | FSR | Puntero indirecto de direccionamiento de datos | | | | | | | |

Figura 2.10. Selector de registros

El puerto "A" RA0 y es de 5 bits para entrada / salida con valor 05h. Este puerto al igual que todos sus similares en los PIC, puede leerse o escribirse como si se tratara de un registro cualquiera. El registro que controla el sentido (entrada o salida) de los pines de este puerto está localizado en la pagina 1 (Banco 1), en la posición 85h y se llama trisa. En la figura 2.11. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 05h | PORTA | -- | -- | -- | RB4/TOCKI | RA3 | RA2 | RA1 | RA0 |
| 85h | TRISA | -- | -- | -- | Registro de direccionamiento de datos del puerto A | | | | |

Figura 2.11. Puerto A de entrada / salida de 5 bits

El puerto "B" RB0 es de 8 bits es de entrada / salida con valor 06h RB0-RB7. Al igual que en todos los PIC, este puede leerse o escribirse como si se tratara de un registro cualquiera; algunos de sus pines tienen funciones alternas en la generación de interrupciones. El registro de control para la configuración de la función de sus pines se localiza en la pagina 1 (Banco 1), en la dirección 86h y se llama trisb. En la figura 2.12. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 06h | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| 86h | TRISB | Registro de direccionamiento de datos del puerto B | | | | | | | |

Figura 2.12. Puerto B de entrada / salida de 8 bits

El REGISTRO DE DATOS DE LA EEPROM (EEDATA) con valor 08h. Este registro contiene el dato que se va a escribir en la memoria EEPROM de datos o el que se leyó de ésta. En la figura 2.13. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 08h | EEDATA | Registro de datos EEPROM | | | | | | | |

bit 0: RD, Lectura
Se pone a 1 cuando se va a realizar un ciclo de lectura de la EEPROM, luego pasa a 0 automáticamente.
bit 1: WR, Escritura
Se pone a 1 cuando comienza el ciclo de escritura de la EEPROM
Se pone a 0 cuando finaliza el ciclo de escritura de la EEPROM
bit 2: WREN, Permiso de escritura
1 = Permite la escritura de la EEPROM
0 = Prohibe la escritura de la EEPROM
bit 3:WRWRR, Señalizador de error de escritura
1 = Se pone a 1 cuando una operación de escritura ha terminado prematuramente
0 = La operación de escritura se ha completado correctamente
bit 4:EEIF, Señalizador de final de operación de escritura
1 = La operación de escritura se ha completado con éxito
0 = La operación de escritura no se ha completado

Figura 2.13. Registro de datos EEPROM

El REGISTRO DE DIRECCION DE LA EEPROM (EEADR) con valor 09h. Aquí se mantiene la dirección de la EEPROM de datos que se van a trabajar, bien sea para una operación de lectura o para una de escritura. En la figura 2.14. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 09h | EEADR | Registro de direcciones EEPROM | | | | | | | |

Figura 2.14. Registros de dirección de la EEPROM

El REGISTRO PARA LA PARTE ALTA DE LA DIRECCION (PCLATH) con valor 0Ah. Este registro contiene la parte alta del contador de programa y no se puede acceder directamente. En la figura 2.15. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 0Ah | PCLATH | -- | -- | -- | | | | | |

Figura 2.15. Registro de la parte alta de la dirección

El REGISTRO PARA EL CONTROL DE INTERRUPCIONES (INTCON) con valor 0Bh. Es el encargado del manejo de las interrupciones y contiene los bits que se muestran. En la figura 2.16. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 0Bh | INTCON | GIE | EEIE | TOIE | INTE | RBIE | T0IF | INTF | RBIE |

bit 0 RBIF: flag (R/W) de estado del Port B
1:Cuando cambia de estado cualquier línea de PB (RB<7:4>). Se borra por software
0:Ninguna entrada de PB ha cambiado
bit 1 INTF: flag (R/W) de estado de la interrupción externa INT
1:La entrada de la interrupción se ha activado. Se borra por software
0:No hay interrupción externa
bit 2 T0IF: flag (R/W) de rebosamiento del TMR0
1: El TMR0 se ha desbordado. Se borra por software
0: El TMR0 no se ha desbordado
bit 3 RBIE: Activación de la interrupción del Port B
1: Interrupción activada
0: Interrupción desactivada
bit 4 INTE: Activación de la interrupción externa INT
1: Interrupción activada
0: Interrupción desactivada
bit 5 T0IE: Activación de la interrupción del TMR0
1: Interrupción activada
0: Interrupción desactivada
bit 6 EEIE: Activación de la interrupción de la memoria EEPROM
1: Interrupción activada
0: Interrupción desactivada
bit 7 GIE: Activación Global de Interrupciones
1: Concedido el permiso de interrupciones
0: No hay posibilidad de interrupciones

Figura 2.16. Registros para el control de interrupciones

El REGISTRO DE CONFIGURACION MULTIPLE (OPTION) con valor 81h. Posee varios bits para configurar el preescalador, la interrupción externa, el Timer y las características del Puerto B. Los bits que contiene y las funciones que realiza este registro se muestran. En la figura 2.17. El preescalador es compartido entre el TMR0 y el WDT; su asignación es mutuamente excluyente ya que solamente puede uno de ellos ser preescalado a la vez.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 81h | OPTION | RBPU# | INTEDG | TOSC | TOSE | PSA | PS2 | PS1 | PS0 |

bit 2-0 PS2:PSO: Rango con el que actúa el Divisor de frecuencia

| PS2 PS1 PS0 | Divisor del TMRO | Divisor del WDT |
|---|---|---|
| 0 0 0 | 1:2 | 1:1 |
| 0 0 1 | 1:4 | 1:2 |
| 0 1 0 | 1:8 | 1:4 |
| 0 1 1 | 1:16 | 1:8 |
| 1 0 0 | 1:32 | 1:16 |
| 1 0 1 | 1:64 | 1:32 |
| 1 1 0 | 1:128 | 1:64 |
| 1 1 1 | 1:256 | 1:128 |

bit 3 PSA: Asignación del divisor de frecuencia
1: El divisor de frecuencia se asigna al WDT
0: El divisor de frecuencia se asigna al TMR0
bit 4 TOSE: Tipo de flanco en TOCK1
1: Incremento de TMR0 cada flanco descendente
0: Incremento de TMR0 cada flanco ascendente
bit 5 TOCS: Tipo de Reloj para TMR0
1: Pulsos introducidos a través de T0CK1 (Contador)
0: Pulsos de reloj interno Fosc/4 (Temporizador)
bit 6 INTEDG: Flanco activo control de interrupciones
1: Flanco Ascendente
0: Flanco Descendente
bit 7 BPRU : Resistencia Pull-up Puerto B
1: Desactivadas
0: Activadas

Figura 2.17. Registro de configuración simple

El REGISTRO DE PARA EL CONTROL DE LA MEMORIA EEPROM DE DATOS (EECON1) con valor 88h. Este es el registro de control para la memoria de datos y solo destina cinco bits para ello, los más bajos; los tres bits superiores permanecen sin implementar. En la figura2.18. Se muestran las funciones de estos bits.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 88h | EECON1 | u | u | u | EEIF | WRERR | WRWN | WR | RD |

Unimplemented. No implementados.
EEIF
EEPROM Write Completion Interrup Flag o Bandera de finalización de la escritura. Se coloca en "1" cuando finaliza con éxito la escritura de la EEPROM de datos; se debe colocar en "0" por programa. El bit de habilitación correspondiente es el EEIE, localizado en el registro INTCON.
WRERR
Write Error Flag o Bandera de error de escritura. Si se coloca en "1" cuando la operación de escritura termina prematuramente, debido a cualquier condición de reset.
WREN
Write Enable o habilitación de escritura. Si se coloca en "0" no permite las operaciones de escritura; en "1" las habilita.
WR
Write Control o Control de escritura. Al colocarse en "1" inicia un ciclo de escritura. Este bit sólo es puesto a "0" por hardware, una vez la escritura termina.
RD
Read Control o Control de lectura. Al colocarse en "1" se inicia una lectura de la EEPROM de datos, la cual toma un ciclo de reloj de instrucciones. Este bit sólo se limpia (colocar en "0") por hardware, al finalizar la lectura de la posición de la EEPROM.

Figura 2.18. Registros para el control de la memoria

El REGISTRO DE CONFIGURACION DEL PUERTO A (TRISA) con valor 85h. Es el registro de control para el puerto A. Un "cero" en el bit correspondiente al pin lo configura como salida, mientras que un "uno" lo hace como entrada. En la figura 2.19. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 85h | TRISA | -- | -- | -- | Registro de direccionamiento de datos del PORTA | | | | |

Figura 2.19. Registro de configuración del puerto A

El REGISTRO DE CONFIGURACION DEL PUERTO B (TRISB) con valor 86h. Es el registro de control para el puerto B. Un "cero" en el bit correspondiente al pin lo configura como salida, mientras que un "uno" lo hace como entrada. En la figura 2.20. Se muestran los bits correspondientes a este registro.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 86h | TRISB | Registro de direccionamiento de datos del PORTB | | | | | | | |

Figura 2.20. Registro de configuración del puerto B

El REGISTRO AUXILIAR PARA EL CONTROL DE LA MEMORIA EEPROM DE DATOS (EECON2) con valor 89h. Este registro no es implementado físicamente por el microcontrolador, pero que es necesario en las operaciones de escritura en la EEPROM de datos; ante cualquier intento de lectura se tendrán "ceros". En la figura 2.21. Se muestran las funciones de estos bits.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 89h | EECON2 | Registro de control de EEPROM (no es un registro físico) | | | | | | | |

Figura 2.21. Registro auxiliar para el control dela memoria EEPROM de Datos

Los REGISTRO DE PROPOSITO GENERAL 0Ch a 4Fh. Estás 68 posiciones están implementadas en la memoria RAM estática, la cual conforma el área de trabajo del usuario; a ellas también se accede cuando en la pagina 1 (Banco 1) se direccionan las posiciones 8Ch a CFh. Esto se ha diseñado así para evitar un excesivo cambio de paginas en el manejo de la RAM del usuario, agilizando los procesos que se estén llevando a cabo y simplificando la labor del programador.

REGISTRO DE TRABAJO W. Este es el registro de trabajo principal, se comporta de manera similar al acumulador en los microprocesadores. Este registro participa en casi todo el programa y por consiguiente en la mayoría de las instrucciones.

PILA (STACK). Estos registros no forman parte de ningún banco de memoria y no permiten el acceso por parte del usuario. Se usan para guardar el valor del contador de programa cuando se hace un llamado a una subrutina (CALL ), o cuando se atiende una interrupción; luego, cuando el microcontrolador regresa a seguir ejecutando su tarea normal, el contador de programa recupera su valor leyéndolo nuevamente desde la pila. El PIC 16F84 tiene una pila de 8 niveles, esto significa que se pueden anidar 8 llamados a subrutina sin tener problema alguno.

## 2.4.3. EL OSCILADOR EXTERNO

Todo Microcontrolador requiere un circuito externo que le indique la velocidad a la que debe trabajar. Este circuito, que se conoce con el nombre de oscilador o reloj, es muy simple pero de vital importancia para el buen funcionamiento del sistema. El PIC 16F84 puede utilizar cuatro tipos de oscilador diferentes. Estos tipos son:

- RC. Oscilador con resistencia y condensador.
- XT. Cristal de cuarzo.
- HS. Cristal de alta velocidad.
- LP. Cristal para baja frecuencia y bajo consumo de potencia.

En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace a través de unos fusibles llamados "fusibles de configuración".
En la mayoría de las practicas que realizaremos se sugiere el cristal de 4 MHz, por que garantiza una mayor precisión y un buen arranque del microcontrolador. Internamente está frecuencia está dividida por cuatro, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz, por lo que cada instrucción se realiza en un microsegundo (1 µS). El cristal debe ir acompañado de dos condensadores y se conecta como se muestra en la figura 2.22.

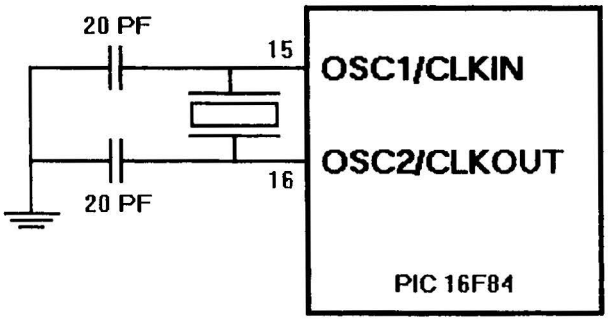



Figura 2.22. Conexión del cristal de 4 MHz

Dependiendo de la aplicación, se pueden utilizar cristales de otras frecuencias; por ejemplo se usa el cristal de 3.579545 MHz por que es muy económico, el de 32.768 Khz. cuando se necesita crear bases de tiempo de un segundo muy precisas. El límite de velocidad de este microcontrolador es de 10 MHz.
Si no se requiere mucha precisión en el oscilador y se requiere economizar dinero, se puede utilizar una resistencia y un condensador, como se muestra en la figura 2.23.

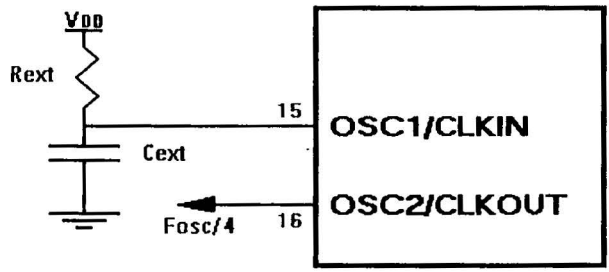



Figura 2.23 Conexión RC en el oscilador

Los valores recomendados para este tipo de oscilador son: 5 Kohms < Rext < 100 Kohms y Cext > 20 pF.

Nota: Cuando el oscilador del dispositivo está en modo RC, no maneje el pin OSC1 con un reloj externo por que puede dañar el dispositivo.

La frecuencia del oscilador dividida por cuatro está disponible en el pin OSC2/CLKOUT, y puede ser usada para chequear propósitos o para sincronizar otra lógica.

RESET

En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando sea necesario, ya sea por una falla que se presente o por que así se halla diseñado el sistema. El pin de reset en los PIC es llamado "Master Clear". El PIC 16F84 admite diferentes tipos de reset:

- Al encendido (Power On Reset)
- Pulso en el pin Master Clear durante operación normal
- Pulso en el pin Master Clear durante el modo de bajo consumo (modo sleep)
- El rebase del conteo del circuito de vigilancia (watchdog) durante operación normal.

- El rebase del conteo del circuito de vigilancia (watchdog) durante el modo de bajo consumo (sleep)

El reset al encendido se consigue gracias a dos temporizadores. El primero de ellos es el OST (Oscillator Star-Up Timer: Temporizador de encendido del oscilador), orientado a mantener el microcontrolador en reset hasta que el oscilador de cristal es estáble. El segundo es el PWRT (Power-Up Timer: Temporizador de encendido), que provee un retardo fijo de 72 mS (nominal) en el encendido únicamente, diseñado para mantener el dispositivo en reset mientras la fuente se estábiliza. Para utilizar estos temporizadores, solo basta con conectar el pin Master Clear a la fuente de alimentación evitándose utilizar las tradicionales redes RC externas en el pin de reset.

El reset por Master Clear se consigue llevando momentáneamente este pin a un estádo lógico bajo, mientras que el watchdog WDT produce un reset cuando su temporizador rebasa la cuenta, o sea que pasa de 0FFh a 00H. Cuando se quiere tener control sobre el reset del sistema se puede conectar un botón como se muestra en la siguiente figura 2.24.

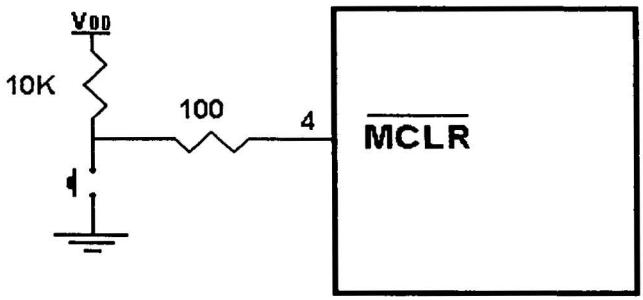



Figura 2.24. Reset por Master Clear

Reset por Brown-out: Un brown-out es una condición en donde la alimentación del dispositivo (Vdd) baja a un valor mínimo, pero no a cero y luego se normaliza. El dispositivo debe resetearse en caso de presentarse un brown-out. Para resetear un PIC 16F84 cuando un brown-out ocurre se debe construir un circuito de protección externo como el de la siguiente figura 2.25.

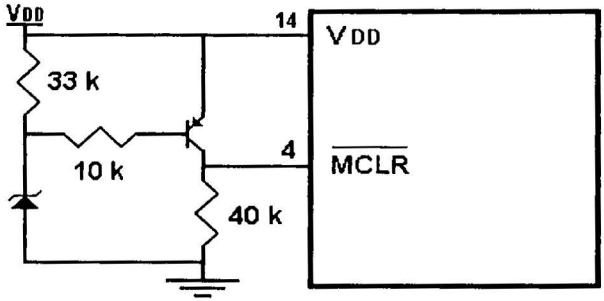



Figura2.25. Circuito de Protección # 1.

Este circuito entrará en un reset activo cuando VDD baja por debajo de Vz + 0.7, en donde Vz = Voltaje del Zener. Figura 2.26.

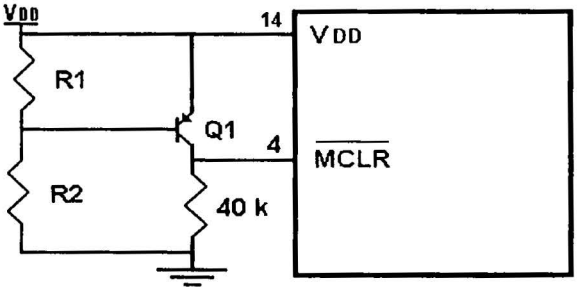



Figura 2.26. Circuito de Protección # 2.

Este circuito es más económico, aunque menos eficaz. El transistor Q1 pasará a un estado de corte cuando VDD está por debajo de un cierto nivel tal que:

$$VDD * (R1 / (R1 + R2)) = 0.7\ V$$

## 2.4.4. CIRCUITO DE VIGILANCIA (Watchdog Timer)

Su función es restablecer el programa cuando éste se ha perdido por fallas en la programación o por alguna razón externa. Es muy útil cuando se trabaja en ambientes con mucha interferencia o ruido electromagnético. Está conformado por un oscilador RC que se encuentra dentro del microprocesador. Este oscilador corre de manera independiente al oscilador principal. Cuando se habilita su funcionamiento, dicho circuito hace que el microcontrolador sufra un reset cada determinado tiempo (que se puede programar entre 18 mS y 2 segundos). Este reset lo puede evitar el usuario mediante una instrucción especial del microcontrolador (CLRWT: Borra el contenido del watchdog), la cual se debe ejecutar antes de que termine el periodo nominal de dicho temporizador. De está manera si el programa se ha salido de su flujo normal, por algún ruido o interferencia externa, el sistema se reiniciará (cuando se acabe el tiempo programado y no se haya borrado el contador) y el programa puede restablecerse para continuar con su funcionamiento normal.
En las primeras practicas no se utiliza el circuito de vigilancia para facilitar el trabajo; por eso, en el momento de programar el microcontrolador se debe seleccionar en los fusibles de configuración "watchdog timer off". temporizador de encendido (power-up timer)
Este proporciona un reset al microcontrolador en el momento de conectar la fuente de alimentación, lo que garantiza un arranque correcto del sistema. En el momento de grabar el microcontrolador se debe habilitar el fusible de configuración "Power-up Timer", para ello se debe seleccionar "ON". Su tiempo de retardo es de 72 milisegundos.

## 2.4.5 MODO DE BAJO CONSUMO (Sleep)

Esta característica permite que el microcontrolador entre en un estado pasivo donde consume muy poca potencia. Cuando se entra en este modo el oscilador principal se detiene, pero el temporizador del circuito de vigilancia (watchdog) se reinicia y empieza su conteo nuevamente. Se entra en ese estado por la ejecución de una instrucción especial (llamada SLEEP) y se sale de él cuando el microcontrolador sufre un reset por un pulso en el pin MCLR, por que el watchdog hace que se reinicie el sistema o por que ocurre una interrupción al sistema.

## 2.4.6 INTERRUPCIONES

Este microcontrolador incluye el manejo de interrupciones, lo cual representa grandes ventajas. El PIC16F84 posee cuatro formas de interrupción que son:

- Interrupción externa en el pin RB0/INT
- Finalización del temporizador / contador TMR0
- Finalización de escritura en la EEPROM de datos
- Cambio de estado en los pines RB4 a RB7

El registro 0Bh o INTCON contiene las banderas de las interrupciones INT, cambio en el puerto B y finalización del conteo del TMR0, al igual que el control para habilitar o deshabilitar cada una de las fuentes de interrupción, figura 2.27.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 0Bh | INTCON | GIE | EEIE | TOIE | INTE | RBIE | TOIF | INTF | RBIE |

Figura 2.27. Registros para el control de interrupciones

Incluida la de escritura de la memoria EEPROM. Sólo la bandera de finalización de la escritura reside en el registro 88h o EECON1.

Si el bit GIE (Global Interrup Enable) se coloca en 0, deshabilita todas las interrupciones. Cuando una interrupción es atendida, el bit GIE se coloca en 0 automáticamente pare evitar interferencias con otras interrupciones que se pudieran presentar, la dirección de retorno se coloca en la pila y el PIC se carga con la dirección 04h. Una vez en la rutina de servicio, la fuente de interrupción se pueden determinar examinando las banderas de interrupción. La bandera respectiva se debe colocar, por software, en cero antes de regresar de la interrupción, para evitar que se vuelva a detectar nuevamente la misma interrupción. La instrucción RETFIE permite al usuario retornar de la interrupción, a la vez que habilita de nuevo las interrupciones, al colocar el bit GIE en uno. Debe tenerse presente que solamente el contador de programa es puesto en la pila al atenderse la interrupción; por lo tanto, es conveniente que el programador tenga cuidado con el registro de estádos y el de trabajo, ya que se pueden introducir resultados inesperados si dentro de ella se modifican.
Interrupción Externa. Actúa sobre el pin RB0/INT y se puede configurar para activarse con el flanco de subida o el de bajada, de acuerdo al bit INTEDG (Interrup Edge Select Bit), localizado en el registro OPTION. Figura 2.28.

| Dirección | Nombre | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 81h | OPTION | RBPU# | INTEDG | TOSC | TOSE | PSA | PS2 | PS1 | PS0 |

Figura 2.28. Registro de configuración simple

Cuando se presenta un flanco valido en el pin INT, la bandera INTF (INTCON) se coloca en uno. La interrupción se puede deshabilitar colocando el bit de control INTE del registro INTCON en cero. Cuando se atiende la interrupción, a través de la rutina de servicio, INTF se debe colocar en cero antes de regresar al programa principal. La interrupción puede reactivar al microcontrolador después de la instrucción SLEEP, si previamente el bit INTE fue habilitado

Interrupción por finalización de la temporización. La superación del conteo máximo (0FFh) en el TMR0 colocara el bit TOIF del registro INTCON en uno. El bit de control respectivo es TOIE del registro INTCON.
Interrupción por cambio en el puerto RB. Un cambio en los pines del puerto B (RB4 a RB7) colocará en uno el bit RBIF del registro INTCON. El bit de control respectivo es RBIE del registro INTCON.
Interrupción por finalización de escritura. Cuando la escritura de un dato en la EEPROM finaliza, se coloca en 1 el bit EEIF (EECON1), el bit de control respectivo es EEIE del registro INTCON.

### 2.4.7 FUSIBLES DE CONFIGURACIÓN

El PIC 16F84 posee cinco fusibles, cada uno de los cuales es un bit. Estos fusibles se pueden programas para seleccionar varias configuraciones del dispositivo: tipo de oscilador, protección de código, habilitación del circuito de vigilancia y el temporizador al encendido. Los bits se localizan en la posición de memoria 2007h, posición a la cual el usuario sólo tiene acceso durante la programación del microcontrolador. Cuando se programa la protección del código, el contenido de cada posición de la memoria no se puede leer completamente, de tal manera que el código del programa no se puede reconstruir. Adicionalmente, todas las posiciones de memoria del programa se protegen contra la reprogramación.
Una vez protegido el código, el fusible de protección solo puede ser borrado (puesto a 1) si se borra toda la memoria del programa y la de datos.

### 2.4.8 LAS PULL-UP INTERNAS

Cada uno de los pines del puerto B tiene un elemento débil pull-up interno (250 uA típico); este elemento es automáticamente desconectado cuando el pin se configura como salida. Adicionalmente, el bit RBPU (OPTION) controla todos estos elementos, los cuales están deshabilitados frente a una condición de reset. Estos elementos pull-up son especialmente útiles cuando el microcontrolador va a colocarse en el modo de bajo consumo, ya que ayudan a no tener las entradas flotantes, significado una reducción en el consumo de corriente.

## 2.5. INSTRUCCIONES DE PROGRAMACION

Para la programación solo es necesario conocer las 35 instrucciones que maneja el microcontrolador y el software y hardware que el fabricante proporciona para la grabación del microcontrolador.

Estas instrucciones se clasifican en orientadas a registros, orientadas a bit y operaciones literales y de control. Cada instrucción es una palabra de 14 bits, dividida en un código de operación (el cuál especifica la orden a ejecutar) y uno o más operándos sobre los que se actúa.
Como se puede observar en la tabla 2.1, existe un total de 35, instrucciones; las cuales tardan un ciclo de máquina a excepción de los saltos que ocupan 2 ciclos.

| MNEMÓNICO | DESCRIPCIÓN | CÓDIGO OP | FLAGS AFECTADOS | NOTAS |
|---|---|---|---|---|
| **Instrucciones Orientadas a Registros** | | | | |
| ADDWF f,d | (W)+(f) [] (destino) | 00 0111 dfff ffff | C, DC, Z | 1,2 |
| ANDWF f,d | (W) AND (f) [] (destino) | 00 0101 dfff ffff | Z | 1,2 |
| CLRF f | Borra f | 00 0001 1fff ffff | Z | 2 |
| CLRW | Borra W | 00 0001 0000 0011 | Z | 1,2 |
| COMF f,d | Complemento de f [(#f)[] (destino)] | 00 1001 dfff ffff | Z | 1,2 |
| DECF f,d | Decremento de f [] destino | 00 0011 dfff ffff | Z | 1,2,3 |
| DECFSZ f,d | Decremento de (f)-1 [] destino y si resultado es 0 salta | 00 1011 dfff ffff | Ninguno | 1,2 |
| INCF f,d | Incremento de (f)+1 [] destino | 00 1010 dfff ffff | Z | 1,2,3 |
| INCFSZ f,d | Incremento de (f)+1 [] destino y si resultado es 0 salta | 00 1111 dfff ffff | Ninguno | 1,2 |

Continua

| | | | | |
|---|---|---|---|---|
| IORWF f,d | Or inclusiva OR (f) □ destino | 00 0100 dfff ffff | Z | 1.2 |
| MOVF f,d | Mueve f □ destino | 00 1000 dfff ffff | Z | |
| MOVWF f | Mueve (W) □ (f) | 00 0000 1fff ffff | Ninguno | 1.2 |
| NOP | No operación | 00 0000 0xx0 0000 | Ninguno | 1.2 |
| RLF f,d | Rota f a la izq a través del carry □ destino | 00 1101 dfff ffff | C | 1.2 |
| RRF f,d | Rota f a la dcha a través del cary □ destino | 00 1100 dfff ffff | C | 1.2 |
| SUBWF f,d | Substrae (f)-(W) □ (destino) | 00 0010 dfff ffff | C,DC,Z | 1.2 |
| SWAPF f,d | Intercambia los nibbles de f □ destino | 00 1110 dfff ffff | Ninguno | |
| XORWF f,d | Or exclusiva(W) XOR (f) □ (destino) | 00 0110 dfff ffff | Z | |
| **Instrucciones orientadas a Bit** | | | | |
| BCF f,b | Pone a 0 el bit b del registro f | 01 00bb bfff ffff | Ninguno | 1.2 |
| BSF f,b | Pone a 1 el bit b del registro f | 01 01bb bfff ffff | Ninguno | 1.2 |
| BTFSC f,b | Skip si el bit b del reg. f es 0 | 01 10bb bfff ffff | Ninguno | 3 |
| BTFSS f,b | Skip si el bit b del reg. f es 1 | 01 11bb bfff ffff | Ninguno | 3 |
| **Instrucciones con literales y de control** | | | | |
| ADDLW K | (W)+ K à (W) | 11 111x kkkk kkkk | C,DC,Z | |
| ANDLW K | (W) AND K à (W) | 11 1001 kkkk kkkk | Z | |
| CALL K | Llamada a subrutina | 10 0kkk kkkk kkkk | | |

Continua

| | | | | |
|---|---|---|---|---|
| CLRWDT | Clear del temporizador del WD | 00 0000 0110 0100 | Ninguno | |
| GOTO K | Go To dirección | 10 1kkk kkkk kkkk | Z | |
| IORLW K | (W) OR K à (W) | 11 1000 kkkk kkkk | Ninguno | |
| MOVLW K | K à (W) | 11 00xx kkkk kkkk | Ninguno | |
| RETFIE | Retorno de una interrupción | 00 0000 0000 1001 | Ninguno | |
| RETLW K | Retorno con un literal en W | 11 01xx kkkk kkkk | Ninguno | |
| RETURN | Retorno de una subrutina | 00 0000 0000 1000 | C,DC,Z | |
| SLEEP | Modo Standby | 00 0000 0110 0011 | Z | |
| SUBLW K | K – (W) à W | 11 110x kkkk kkkk | | |
| XORLW K | (W) XOR K à (W) | 11 1010 kkkk kkkk | | |

Tabla 2.1 conjunto de 35 instrucciones del microcontrolador PIC 16F84

Notas.

1. Al modificar un registro de E/S con una operación sobre él mismo (por ejemplo MOVF PORTB,1), el valor utilizado es el que se halle presente en los pines del PORTB. Por ejemplo, si el biestáble tiene un "1" para una patilla configurada como entrada y se pone a nivel bajo desde el exterior, el dato se volverá a escribir como "0".
2. Si se ejecuta está instrucción sobre el TMR0 y d=1, será borrado el divisor de frecuencia (preescaler), si está asignado al TMR0.

Si se modifica el Contador de Programa (PC) o una condición de prueba es verdadera, la instrucción requiere dos ciclos máquina. El segundo ciclo se ejecuta como un NOP

Descripción de las instrucciones

En el apéndice II de describe cada una de las 35 instrucciones del microcontrolador donde se indica el nemónico de la instrucción, la descripción de su operación, la operación que realiza la sintaxis dela misma y un ejemplo de su funcionamiento.

# Capitulo 3

# EL MOTOR A PASOS

También conocidos como "stepper motor" pueden girar en un ángulo preciso llamado "paso" o "step". Debido a que resultan muy precisos y confiables, se emplean comúnmente en aplicaciones donde el posicionamiento mecánico resulta ser muy importante. Son ideales cuando se tiene que girar un eje y detenerlo en ciertas posiciones con una precisión de hasta centésimas de milímetros, y dicha posición permanece bloqueada hasta que no se le de una nueva orden para hacerla girar en el sentido que queramos. Los motores de paso son conocidos en Alemania como Schrittmotoren y en Francia como moteurs pas à pas.

## 3.1 CARACTERÍSTICAS GENERALES

Los motores de pasos se diferencian de los otros tipos de motores por las siguientes características:

Convierten pulsos eléctricos en movimientos rotacionales discretos.
No son muy rápidos en términos de RPM (revoluciones por minuto), en comparación con los demás tipos de motores, por ejemplo para un motor de 1000 pasos por segundos, tiene un RPM de 150 y 1.8° por paso.

Siempre necesitan de un circuito especial externo para controlarlo (driver o manejador) debido a que no se le puede conectar directamente a una fuente de alimentación.
Son ideales para el posicionamiento, ya que son de fácil manejo y normalmente no necesitan una constante realimentación (lazo cerrado de control) o monitoreo. Lo único que se requiere es transmitir un número exacto de pasos para llevarlo a una posición exacta y repetible. El motor de paso de lazo abierto es ideal para sistemas que operan a bajas aceleraciones y cargas estáticas, pero un sistema de lazo cerrado seria esencial para altas aceleraciones y cargas variables.

Alcanzan una gran precisión y pueden moverse en incrementos muy pequeños, característica difícil de lograr en los motores DC pues aunque se desenergize el motor muy rápido, la inercia del rotor continuará girando el eje hasta una posición casual.
Debido a su bajo costo y pequeño tamaño en comparación con los demás tipos de motores, son empleados en disk - drives, impresoras, plotters, etc.
Entre las características comunes en los motores de pasos, tenemos los siguientes elementos:

Voltaje: Los motores de pasos usualmente tienen un rango de voltaje, que va indicado en el mismo motor o en las hojas de datos. A veces es necesario exceder el rango de voltaje para obtener el torque deseado de un motor dado, pero esto puede sobrecalentar y/o disminuir el
Resistencia: Una característica común es la resistencia por bobina. Esta resistencia

determinará la corriente que pase por el motor, también como la curva de torque del motor y la máxima velocidad de operación.

Grados por paso: Este es el factor mas importante al momento de escoger un motor para una determinada aplicación. Este factor especifica el número de grados que el rotor girará por cada paso. En la operación de medio paso del motor, el número de pasos por revolución es el doble y los grados por revolución se reducen a la mitad. Hay motores de 0.72°, 1.8°,3.6°, 7.5°, 15°, y hasta 90° por paso. Los grados por paso es comúnmente referido como la resolución del motor. Si un motor tiene el solo número de pasos / revolución, basta dividir 360° por este número para obtener el valor de grados / pasos.

## 3.2 CLASIFICACIÓN

Existen dos tipos de motores de paso, los de imán permanente y los de reluctancia variable. Existen también los híbridos, de imán permanente desde el punto de vista del controlador.

Los de imán permanente muestran resistencia cuando intentamos girar el eje con los dedos, mientras que los de reluctancia variable, casi siempre giran libremente o con menos dificultad. Sin embargo pueden mostrar cierta resistencia debido a una magnetización residual en el rotor.

Hay dos maneras para averiguar la distribución de los cables o los bobinados y el cable común en un motor de paso unipolar de 5 o 6 cables:

1. Aislando el cable(s) común que va a la fuente de alimentación usando un ohmetro para chequear la resistencia entre pares de cables. El cable común será el único que tenga la mitad del valor de la resistencia entre ella y el resto de los cables, como se muestra en la figura 3.1.

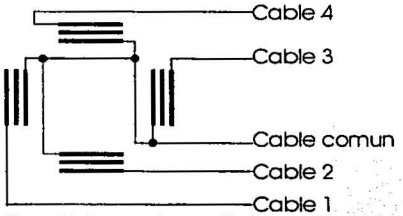


Figura 3.1 diagrama de conexión de un motor unipolar

Esto es debido a que el cable común tiene una bobina entre ella y cualquier otro cable, mientras que cada uno de los otros cables tienen dos bobinas entre ellos. De ahí la mitad de la resistencia.

2. Identificando los cables de las bobinas aplicando un voltaje al cable común y manteniendo uno de los otros cables a tierra mientras vamos poniendo a tierra cada uno de los demás cables de forma alternada y observando los resultados ver la tabla 3.1.

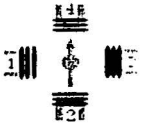
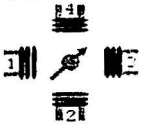
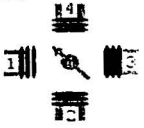
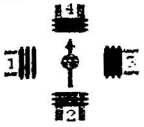
| | |
|---|---|
| Seleccionar un cable y conectarlo a tierra.<br>Elegimos el cable 4. |  |
| Manteniéndolo a tierra, poner el resto de los 3 cables a tierra uno por uno | |
| Poniendo uno a tierra haremos girar el rotor en sentido horario. Ese será el cable 3. |  |
| Poniendo otro a tierra haremos girar el rotor en sentido antihorario. Ese será el cable 1. |  |
| Poniendo otro a tierra el rotor no va a girar.<br>Ese será el cable 2. |  |

Tabla 3.1 Identificando los cables de las bobinas aplicando un voltaje al cable común

## 3.3 PRINCIPIO DE FUNCIONAMIENTO

Básicamente estos motores están constituidos normalmente por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas excitadoras bobinadas en su estator.

Los motores de pasos pueden ser vistos como motores eléctricos sin conmutadores. Las bobinas son parte del estator y el rotor es un imán permanente o, en el caso de los motores de paso de reluctancia variable, una pieza dentada hecha de una material magnético. Toda la conmutación (o excitación de las bobinas) deber ser externamente manejado por un controlador o driver.

Para comprender de una forma mas sencilla el funcionamiento del motor, representaremos un motor teórico de 4 bobinas con un imán y realicemos algunos ejemplos.

Si se quiere girar el imán a la posición A, se tiene que alimentar las bobinas 4 y 3, Figura 3.2 motor de 4 bobinas con imán

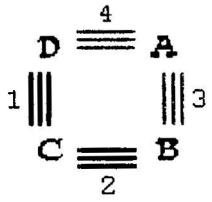



Figura 3.2. Motor de 4 bobinas con imán

Si se quiere girar el imán a la posición B, se tiene que alimentar las bobinas 3 y 2 Figura 3.3 motor de 4 bobinas con imán

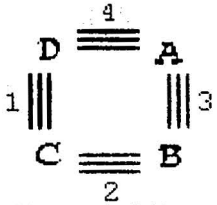



Figura 3.3. Motor de 4 bobinas con imán

Si se quiere girar el imán entre B y C, se tiene que alimentar la bobina 2 Figura 3.4 motor de 4 bobinas con imán

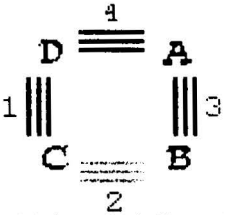



Figura 3.4. Motor de 4 bobinas con imán

Si se quiere girar el imán entre C y D, se tiene que alimentar la bobina 1 Figura 3.5 motor de 4 bobinas con imán

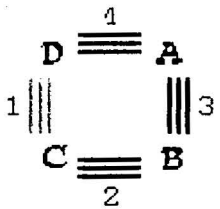



Figura 3.5. Motor de 4 bobinas con imán

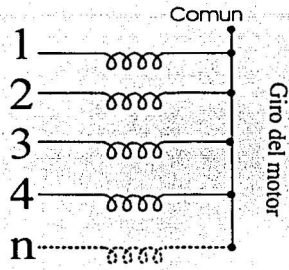
Cabe recordar que el motor se puede hacer girar en sentido horario y antihorario, y que en un motor de paso real existe un número mayor de imanes y bobinas excitadoras.

En general, Pasos por revolución = (Número de fases -o bobinas-) x (Número de dientes en el rotor)

Para algunos motores de pasos, el número de bobinas no es igual al número de fases. Mientras más grande sea el número de polos magnéticos y bobinas, menos grados girará el eje por paso

## 3.4 CARACTERÍSTICAS DEL MOTOR CON RELUCTANCIA VARIABLE.

Los motores de paso de reluctancia variable usualmente tienen 3 o 4 devanados con un retorno común, Son los motores de pasos más simples de controlar. La secuencia de control es simplemente energizar cada una de las bobinas en orden, una después de otra. Tiene un cable de alimentación que es común a un extremo de cada bobina. Este tipo de motor se siente como un motor DC cuando el rotor el girado manualmente, gira libremente y no se sienten los pasos debido a que no está permanentemente magnetizado como sus contrapartes, los unipolares y bipolares. En la figura 3.6 se muestra la secuencia de control del motor de reluctancia variable.




| Paso | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 |
| 8 | 0 | 0 | 0 | 1 |

Figura 3.6 secuencia de control de un motor paso a paso de reluctancia variable

Si el motor tiene 3 bobinas, generalmente se presentan 4 cables (uno es común a las demás bobinas) y puede ser de reluctancia variable. El cable común va a la fuente de alimentación positiva. Figura 3.7 motor de reluctancia variable

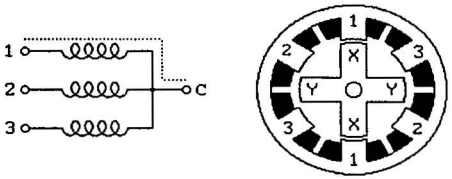



Figura. 3.7 motor de reluctancia variable de 30 grados

El motor mostrado es de 30° por paso y de reluctancia variable. El rotor tiene 4 dientes y el estator es de 6 polos. Cada bobina es enrollada alrededor de dos polos opuestos. si excitamos la bobina 1, el diente del rotor marcado con X es atraído hacia los polos excitados 1. Ahora si la corriente fluye por la bobina 2, el rotor girara 30° en sentido horario y los dientes marcados con Y se alinearan con los polos marcados con 2.
Para mover el motor se deben energizar las bobinas secuencialmente. Asumiendo lógica positiva, donde 1 es para energizar la bobina. La figura 3.8 muestra la secuencia de control del motor.

Bobina 1 100100100100100100100100
Bobina 2 010010010010010010010010
Bobina 3 001001001001001001001001
time --->

Figura 3.8. Secuencia de control del motor de reluctancia variable

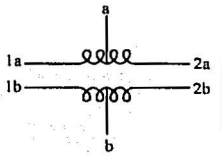
También hay motores de reluctancia variable con 4 y 5 bobinas que requieren 5 o 6 cables, pero el principio de manejo es el mismo.

## 3.5 CARACTERISTICAS DEL MOTOR CON IMÁN PERMANENTE

los motores de imán permanente tienen 2 devanados independientes, sin tap central en motores bipolares, y con tap central en motores unipolares.

### 3.5.1 MOTORES DE IMÁN PERMANENTE UNIPOLARES

Motores unipolares son relativamente fáciles de controlar. Su característica principal es tener un tap central, de manera que el esquema de cableado es tomar el (los) tap (s) centrales y conectarlos a la fuente de alimentación positiva. El circuito controlador se encargará de poner cada bobina a tierra para energizarla de manera secuencial. El número de fases es el doble al número de bobinas, ya que cada bobina es dividida en dos por medio del tap central. Figura 3.9 circuito de un motor unipolar y secuencia de control.




| Paso | 1a | 1b | 2a | 2b |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 |
| 8 | 0 | 0 | 0 | 1 |

Figura 3.9. Circuito de un motor unipolar de 4 fases y su secuencia de control

Otra manera a la secuencia de manejo estándar, es posible una secuencia de manejo a medio paso y otra de gran torque. En la secuencia de gran torque, 2 bobinados están activados al mismo tiempo para cada paso del motor. En este caso el torque es 1.5 veces mas que el entregado en una secuencia estándar, pero maneja el doble de corriente. La secuencia de manejo a medio paso es la combinación de las 2 anteriores. Primero una bobina es activado, luego dos bobinas, luego una, etc. Esto hace que le número de pasos por revolución sea el doble y el ángulo por paso se reduzca a la mitad en la figura 3.10 se muestran las secuencias de control de estos dos métodos

Giro del motor

| Paso | 1a | 1b | 2a | 2b |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 0 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 0 |
| 8 | 0 | 0 | 1 | 1 |

(a).
secuencia para un mayor torque

Giro del motor

| Paso | 1a | 1b | 2a | 2b |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 12 | 0 | 1 | 1 | 0 |
| 13 | 0 | 0 | 1 | 0 |
| 14 | 0 | 0 | 1 | 1 |
| 15 | 0 | 0 | 0 | 1 |
| 16 | 1 | 0 | 1 | 1 |

(b)
secuencia de medio torque

Figura 3.10 se muestran las secuencias de control de estos dos métodos

Los motores de paso unipolar, ya sean de imán permanente o híbridos con 5 o 6 cables, son cableados como lo muestra la figura 3.11

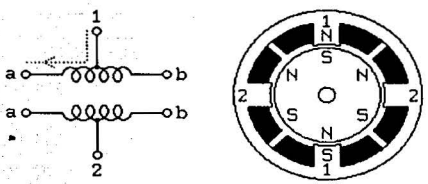



Figura 3.11 Diagrama de un motor de paso unipolar

con un tap central en cada uno de los bobinados. Los tap centrales típicamente son conectados a al fuente de alimentación positiva, y los extremos de cada bobinado son alternativamente puestos a tierra para invertir la dirección del campo entregado por el bobinado.

El motor mostrado en la figura 3.12. Es de un paso de 30°. El bobinado 1 está distribuido entre la parte superior e inferior del estator y el bobinado 2 entre la izquierda y derecha del estator. El rotor es un imán permanente de 6 polos, 3 norte y 3 sur, arreglados alrededor de su circunferencia. Para altas resoluciones el rotor debe tener mas polos.

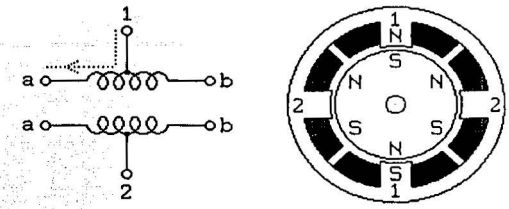



Figura 3.12 motor de paso unipolar de 30 grados

Por ejemplo, para el motor mostrado, si fluye corriente por un lado de la bobina 1, la parte de arriba del estator está en N y la de abajo en S. Esto atrae el rotor a la posición mostrada en la figura 3.6. Si ahora se deja de alimentar la bobina 1 y se alimenta un lado de la bobina 2, el rotor girara o dará un paso de 30 grados.

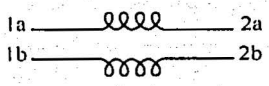
Una secuencia para el caso de la figura 3.12, asumiendo la lógica positiva, es la siguiente:

Bobina 1a 11000001110000011100000111
Bobina1b 00011100000111000001110000
Bobina 2a 01110000011100000111000001
Bobina 2b 00000111000001110000011100

## 3.5.2 MOTORES DE IMÁN PERMANENTE BIPOLARES

Los motores de paso bipolares, ya sean de imán permanente o híbridos, están diseñados de la misma forma que un motor unipolar, la única diferencia es que ya no tienen un tap central. Están diseñados por dos bobinas separadas y cuyas polaridades necesitan ser invertidas o cambiadas en cada operación para que haga girar el rotor.

El motor mismo es simple, pero el circuito manejador o driver que necesita invertir la polaridad de cada par de polos del motor es mas complicado. Dicho circuito permite que la polaridad de la fuente de alimentación aplicado al final de cada bobinado sea controlado de manera independiente. Figura 3.13 circuito y secuencia de control de un motor bipolar




Giro del motor

| Paso | 1a | 1b | 2a | 2b |
|---|---|---|---|---|
| 1 | + | - | - | - |
| 2 | - | + | - | - |
| 3 | - | - | + | |
| 4 | - | - | - | + |
| 5 | + | - | - | - |
| 6 | - | + | - | - |
| 7 | - | - | + | - |
| 8 | - | - | - | + |

Figura 3.13 circuito y secuencia de control de un motor bipolar

Los motores bipolares manejan la misma secuencia que un motor unipolar, pero los '0' y '1' corresponden a la polaridad del voltaje aplicado a las bobinas, ya no son simplemente signos de "on-off". Cada bobina necesita su propio circuito controlador llamado H-bridge (puente H). Un motor bipolar típicamente tiene 4 cables, las cuales van conectadas a los extremos de las dos bobinas aisladas. Figura 3.14. Diagrama de un motor bipolar.

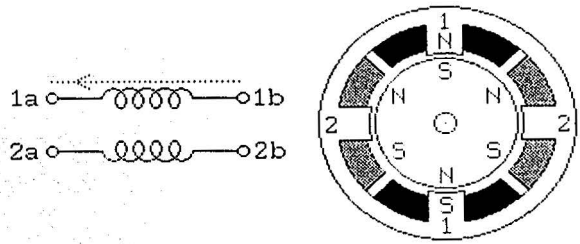



Figura 3.14. Diagrama de un motor bipolar

La secuencia de control, usando + y - para indicar la polaridad del poder aplicado por la fuente de alimentación a cada terminal del motor es la siguiente.

Terminal 1a +---+---+---+--- ++--++--++--++--
Terminal 1b --+---+---+---+- --++--++--++--++
Terminal 2a -+---+---+---+-- -++--++--++--++-
Terminal 2b ---+---+---+---+ +--++--++--++--+

### 3.5.3 MOTOR DE IMÁN PERMANENTE MULTIFASE

Motores de múltiples fases. es un tipo de motor no muy común. Todas las bobinas están conectadas en serie con un tap entre cada par de bobinas. figura 3.15. Diagrama de un motor multifase.

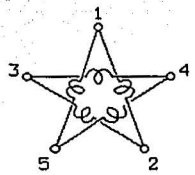



Figura 3.15. Diagrama de un motor multifase

La mayoría es de 3 y/o 5 fases y pueden dar un mas torque debido a que todos o todos menos uno de las bobinas del motor son energizadas en cada punto en un ciclo de manejo o paso, o sea , que para realizar un paso hay que solamente cambiar la polaridad de un terminal en forma secuencial.

Por ejemplo, para un motor de 5 fases, aquí hay una secuencia de 10 pasos que se repite para hacer girar el motor. Notar que para ejecutar un paso basta cambiar la polaridad de un terminal, el paso siguiente energizara la bobina que no estaba energizada en el paso anterior.

Terminal 1 +++-----+++++-----++
Terminal 2 --+++++-----+++++---
Terminal 3 +-----+++++-----++++
Terminal 4 +++++-----+++++-----
Terminal 5 ----+++++-----+++++-

# CAPITULO 4

# ANÁLISIS FUNCIONAL DEL CONTROL CON MICROPROCESADOR Z-80

En el sistema mínimo con microprocesador, observaremos la cantidad de componentes externas, que este requiere para su funcionamiento; como son Memoria, decodificadores circuitos de reloj, y de reset, y dispositivos de entrada y salida, de datos y un circuito de potencia así como también nos referiremos al programa que lleva la memoria EPROM del tamaño que este va ocupar a la hora de programar y numero de instrucciones a realizar, y el numero de conexiones para que el microprocesador haga funcionar el motor a pasos.

El primer punto será un análisis del diagrama a bloques del sistema mínimo, después se analizara el diagrama de componentes utilizadas y el análisis del funcionamiento de cada una de ellas, y por ultimo se hará un análisis de programa que hace funcionar el motor a pasos

## 4.1 IMPLEMENTACIÓN DE UN DIAGRAMA A BLOQUES PARA UN CONTROL CON MICROPROCESADOR Z-80

En la figura 4.1 se muestra el diagrama a bloques del circuito de control con un microprocesador Z-80 para un motor a pasos

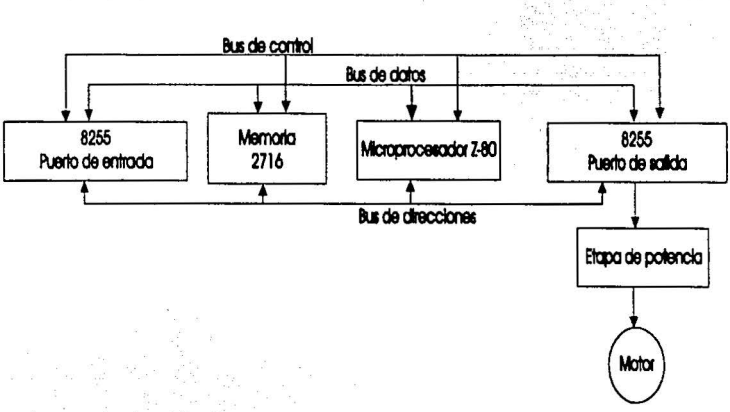



Figura 4.1 Diagrama a bloques del circuito de control con microprocesador Z-80

Como se puede ver el bus de control está conectado con el microprocesador, la memoria y el dispositivo de entrada / salida; así como el bus de direcciones y el bus de datos, para el sistema mínimo, la etapa de potencia permite accionar el motor de pasos.

## 4.2 ANÁLISIS DEL DIAGRAMA A BLOQUE.

Como se puede ver en el diagrama a bloques de la figura 4.1, el bus de control esta conectado con los dispositivos de entrada, memoria, microprocesador, y salida, este como su nombre lo indica transporta las señales que van ha comunicar a los dispositivos e indican en que momento deben interactuar entre ellos.

El bus de direcciones, que también está conectado con los dispositivos antes mencionados, indica en que momento se va ha ejecutar la instrucción que es transportada entre los dispositivos

El bus de datos, que también esta conectado con los mismos dispositivos antes mencionados, es el que se va a encargar de transportar la información que se intercambia entre los dispositivos

El puerto de entrada es una interfaz que comunica a los dispositivos del sistema mínimo, (memoria, microprocesador puerto de entrada / salida) con el mundo exterior y cuenta con tres puertos. Pueden recibir 4 bits ó 8 bits, según el modo de operación de este dispositivo, dichos modos son el 1, 2 y 3; se detallan en la tabla 4.1.

| | | |
|---|---|---|
| Modo 0 | Entradas y salidas básicas | Dos puertos de 8 bits y dos puertos de 4 bits.<br>Un puerto se puede ser de entrada o de salida.<br>16 diferentes configuraciones de este modo |
| Modo 1 | Entradas salidas controladas | Dos puertos de grupo A y B.<br>Cada grupo tiene 8 bits de datos y 4 de control.<br>Los 8 bits pueden ser de entrada o salida<br>Los 4 bits de control son usados para el control de estado de los puertos de 8 bits |
| Modo 2 | entradas bidireccionales | Usa un solo puerto A.<br>un bus bidireccional de 8 bits y 5 bits de control por el puerto C |

Tabla 4.1 modos de operación del 8255

Se puede ver los 3 modos de operación de este dispositivo; donde el modo cero, trabaja con entradas y salidas básicas, 2 puertos de 8 bits y 2 puertos de 4 bits.
El modo uno, trabaja con entradas y salidas controladas, dos grupos A y B de 8 bits.
El modo dos trabaja con un puerto bidireccional A de 8 bits y 5 bits de control.

La operación básica del dispositivo de entrada / salida 8255 se puede ver en la tabla 4.2

| A1 | A0 | RD | WR | CS | OPERACION DE LECTURA |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | DATOS DEL PUERTO A |
| 0 | 1 | 0 | 1 | 0 | DATOS DEL PUERTO B |
| 1 | 0 | 0 | 1 | 0 | DATOS DEL PUERTO C |
| | | | | | OPERACIÓN DE ESCRITURA |
| 0 | 0 | 1 | 0 | 0 | DATOS AL PUERTO A |
| 0 | 1 | 1 | 0 | 0 | DATOS AL PUERTO B |
| 1 | 0 | 1 | 0 | 0 | DATOS AL PUERTO C |
| 1 | 1 | 1 | 0 | 0 | DATOS AL BUS DE CONTROL |

Tabla 4.2 operación básica del 8255

La operación del 8255 consiste en una combinación de las señales del bus de direcciones y de las señales de control RD, WR y CS. Donde la combinación, 00010 lee datos en el puerto A, la combinación 01010, escribe datos en el puerto B, la combinación 10010, escribe datos en el puerto C. Y las combinaciones 00100, 01100, 10100, 11100, escriben en los puertos A, B, C, y control de datos respectivamente, este ultimo configura los puertos en el modo cero dependiendo de la palabra de control, en el apéndice 3 se muestran los 16 modos de configurar los puertos.

Memoria EPROM. Este dispositivo es el encargado de almacenar el programa que hace funcionar al microprocesador Z80

Microprocesador Z80. Este dispositivo es el encargado de accesar a la memoria, ejecutar las instrucciones que se encuentran almacenadas en ella, es el que reconoce las señales de control, proporciona las direcciones, y los datos entre los dispositivos para su correcto funcionamiento que harán funcionar al motor a pasos

El puerto de salida. Este dispositivo se encarga de enviar los bits que vienen del microprocesador a la etapa de potencia.

Etapa de potencia. Está etapa solo se encarga de aumentar la corriente que alimenta a los motores a pasos de acuerdo a los bits que recibe del puerto de salida

## 4.3 IMPLEMENTACIÓN DE UN SISTEMA MÍNIMO CON Z-80

Sistema mínimo. El sistema mínimo está constituido por 4 dispositivos esenciales Microprocesador, Memoria, puerto de entrada / salida y un decodificador; los cuales están inter conectados por tres buses importantes llamados bus de direcciones, bus de datos, y bus de control. También la etapa de potencia que acciona el motor a pasos

En la figura 4.2 se muestra el diagrama de conexión de las componentes del circuito de control que acciona el motor a pasos

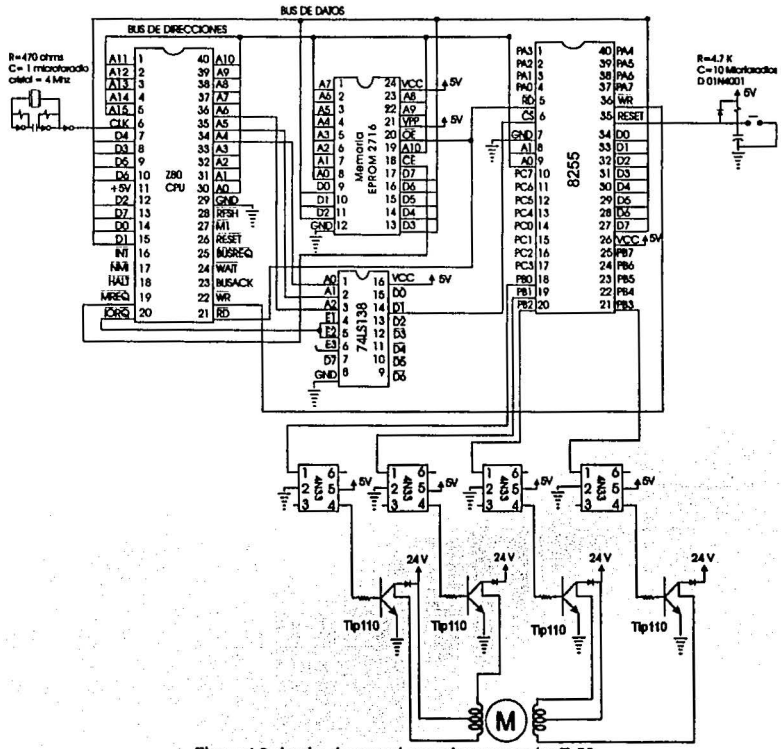



Figura 4.2 circuito de control con microprocesador Z-80

## 4.4 ANÁLISIS DEL SISTEMA MÍNIMO

Microprocesador Z-80. Como se puede ver en la figura 4.2 el microprocesador está conectado a la memoria por medio del bus de direcciones de A0 a A15 que corresponden a las terminales 30 hasta 40 y 1 a la 5 donde (A0 corresponde a la pata 30 y A15 a la pata 5), por el bus de control, MREQ (acceso a memoria) terminal 19, IORQ (acceso a los puertos) terminal 20, y RD (acceso a lectura) terminal 21. Y al bus de datos, D0 terminal 14, D1 terminal 15, D2 terminal 12, D3 terminal 8, D4 terminal 7 D5 terminal 9, D6 terminal 10 y D7 terminal 13. El microprocesador está conectado en su pata 6 a un reloj de 4 MHz, el cual envía los pulsos de reloj para que empiece a trabajar, el primer paso que realiza el microprocesador es activar las señales MREQ (petición de acceso a memoria) y RD (lectura de memoria), y poner el bus de direcciones con la dirección 0000H, para indicar que accesa a la primera localidad de memoria y recibir por el bus de datos la primera instrucción que ahí se encuentra, para ser ejecutada por el microprocesador, y continua así solo que seguiría la localidad 0001H de la memoria para ejecutar la siguiente instrucción.

Memoria EPROM. Este es un dispositivo de almacenamiento de datos borrable con luz ultravioleta, como se puede ver este dispositivo está conectado al bus de direcciones en sus terminales 1 a la 8 y 19, 22, y 23 correspondiente a A0 pata 8, A7 pata 1 y A10, A9, y A8. al bus de control, terminales 18 CE (habilita chip) y OE (habilita salida) pata 20, y el bus de datos terminales 9 a la 11 y 13 a la 17 donde D0 es pata 9 y D7 pata 17.este dispositivo se encarga de proporcionar las instrucciones al microprocesador a través del bus de datos

Puerto de entrada / salida (8255). Este dispositivo está conectado al bus de direcciones en A1 terminal 8 y A0 terminal 9, al bus de datos D0 hasta D7 terminales 34 a la 27, (Do en 34 y D7 en 27), y al bus de control con RD (petición de lectura) terminal 5, CS (habilitar comunicación), terminal 6 y WR (petición de escritura) terminal 36, puerto de entrada con PA0 hasta PA7, terminales 1 a la 4 y 37 a la 40, (PA0 pata 4 y PA7 pata 37) que da acceso a los datos del exterior, y puerto de salida con PB0 hasta PB7 terminales 18 a la 25 ( PB0 terminal 18 y PB7 terminal 25),

Decodificador 74LS138. Este dispositivo esta conectado al bus de direcciones con A0 hasta A2 terminales 1 a la 3 respectivamente (nótese que no están conectados a las direcciones A0, A1, y A2, del microprocesador), al bus de control en E1 Y E2 terminales 4 y 5 respectivamente. Y la salida del dispositivo en D1 terminal 14, este dispositivo se encarga de direccionar el puerto de entrada / salida con el microprocesador Z-80

Etapa de potencia. Los dispositivos 4N33 son opto acopladores para proteger el dispositivo de entrada / salida de los transistores de potencia Tip 110 que se encargan de incrementar la corriente por medio de la fuente de 24V para el funcionamiento del motor

Reloj. Este pequeño circuito proporciona los pulsos de reloj de 4 MHz al microprocesador en la terminal 6

Circuito de reset. Que se encarga de poner a todo el sistema en inicio, a través de las terminales 26 del microprocesador y 35 del 8255.

## 4.5 PROGRAMA PARA MANIPULAR UN MOTOR A PASOS CON Z-80

El siguiente programa de la tabla 4.1. Manipula un motor a pasos realizando 16 pasos a la derecha, posteriormente realiza 16 pasos a la izquierda, y comienza de nuevo el ciclo.

| LOCALIDAD DE MEMORIA | NEMONICO | CODIGO OPERACION | CODIGO HEX |
|---|---|---|---|
| 0000 | LD A,90H | 0011,1110,1001,0000 | 3E90 |
| 0001 | OUT (43H),A | 1101,0011,0100,0011 | D343 |
| 0002 | IN A,(40H) | 1101,1011,0100,0000 | DB40 |
| 0003 | BIT 0,A | 1100,1011,0100,0111 | CB47 |
| 0004 | JP NZ,0035H | 1100,0010,0011,0101,0000,0000 | C23500 |
| 0005 | LD A,03H | 0011,1110,0000,0011 | 3E03 |
| 0006 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0007 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0008 | LD A,06H | 0011,1110,0000,0110 | 3E06 |
| 0009 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 000A | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 000B | LD A,0CH | 0011,1110,0000,1100 | 3E0C |
| 000C | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 000D | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 000E | LD A,09H | 0011,1110,0000,1001 | 3E09 |
| 000F | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0010 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0011 | LD A,03H | 0011,1110,0000,0011 | 3E03 |
| 0012 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0013 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0014 | LD A,06H | 0011,1110,0000,0110 | 3E06 |
| 0015 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0016 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0017 | LD A,0CH | 0011,1110,0000,1100 | 3E0C |
| 0018 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0019 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 001A | LD A,09H | 0011,1110,0000,1001 | 3E09 |
| 001B | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 001C | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 001D | LD A,03H | 0011,1110,0000,0011 | 3E03 |

Continua

| 001E | OUT (41H),A | 1101,0011,0100,0001 | D341 |
|---|---|---|---|
| 001F | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0020 | LD A,06H | 0011,1110,0000,0110 | 3E06 |
| 0021 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0022 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0023 | LD A,0CH | 0011,1110,0000,1100 | 3E0C |
| 0024 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0025 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0026 | LD A,09H | 0011,1110,0000,1001 | 3E09 |
| 0027 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0028 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0029 | LD A,03H | 0011,1110,0000,0011 | 3E03 |
| 002A | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 002B | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 002C | LD A,06H | 0011,1110,0000,0110 | 3E06 |
| 002D | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 002E | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 002F | LD A,0CH | 0011,1110,0000,1100 | 3E0C |
| 0030 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0031 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0032 | LD A,09H | 0011,1110,0000,1001 | 3E09 |
| 0033 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0034 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0035 | LD A,09H | 0011,1110,0000,1001 | 3E09 |
| 0036 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0037 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0038 | LD A,0CH | 0011,1110,0000,1100 | 3E0C |
| 0039 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 003A | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 003B | LD A,06H | 0011,1110,0000,0110 | 3E06 |
| 003C | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 003D | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 003E | LD A,03H | 0011,1110,0000,0011 | 3E03 |
| 003F | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0040 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0041 | LD A,09H | 0011,1110,0000,1001 | 3E09 |
| 0042 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0043 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0044 | LD A,0CH | 0011,1110,0000,1100 | 3E0C |
| 0045 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0046 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 0047 | LD A,06H | 0011,1110,0000,0110 | 3E06 |

Continua

| | | | |
|---|---|---|---|
| 0048 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0049 | CALL 0066H | 1100,1101,0110,0110,0000,0000 | CD6600 |
| 004A | LD A,03H | 0011,1110,0000,0011 | 3E03 |
| 004B | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 004C | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 004D | LD A,09H | 0011,1110,0000,1001 | 3E09 |
| 004E | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 004F | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 0050 | LD A,0CH | 0011,1110,0000,1100 | 3E0C |
| 0051 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0052 | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 0053 | LD A,06H | 0011,1110,0000,0110 | 3E06 |
| 0054 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0055 | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 0056 | LD A,03H | 0011,1110,0000,0011 | 3E03 |
| 0057 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0058 | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 0059 | LD A,09H | 0011,1110,0000,1001 | 3E09 |
| 005A | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 005B | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 005C | LD A,0CH | 0011,1110,0000,1100 | 3E0C |
| 005D | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 005E | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 005F | LD A,06H | 0011,1110,0000,0110 | 3E06 |
| 0060 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0061 | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 0062 | LD A,03H | 0011,1110,0000,0011 | 3E03 |
| 0063 | OUT (41H),A | 1101,0011,0100,0001 | D341 |
| 0064 | CALL 0066H | 1100,1101,0000,0000,0110,0110 | CD0066 |
| 0065 | JP 0002 | 1100,0011,0000,0000,000,0010 | C30002 |
| 0066 | LD D,FFH | 0001,0110,1111,1111 | 16FF |
| 0067 | DEC D | 0001,0101 | 15 |
| 0068 | JP NZ,0067 | 1100,0010,0110,0111,0000,000 | C26700 |
| 0069 | RET | 1100,1001 | C9 |

Tabla 4.3. Programa para manipular un motor a pasos

## 4.6 ANÁLISIS DEL PROGRAMA

En la tabla 4.3 el primer renglón corresponde a la primera localidad de memoria (0000) que el microprocesador Z-80 va a ejecutar, con el nemonico LD A;90 la cual indica que cargué LD el registro A con el valor 90H, con esto se guarda en el registro A el dato que corresponde al numero 90H, el código hex es el que está grabado en la memoria y es el que va a leer el microprocesador

La siguiente instrucción OUT (43H);A en la localidad de memoria (0001), indica al microprocesador Z-80 que envié los datos (90H), contenidos en él registró A al puerto de salida con él numero 43H. está operación configura en el dispositivo 8255 los puertos de salida / entrada.

La siguiente instrucción IN A;(40H) en la localidad de memoria (0002) indica al microprocesador Z-80 que lea los datos que estén contenidos en el puerto de entrada 40H y que la almacene en el registro A.

La instrucción bit 0,A en la localidad de memoria (0003) pone en las banderas Z y H del microprocesador en uno para indicar una condición que se va a cumplir en la siguiente instrucción

La instrucción JPNZ,3500 en la localidad de memoria (0004) es un brinco condicional en caso de que la bandera Z sea un uno, brinca a la dirección de memoria 0035, pero si no es un cero continua con la siguiente localidad de memoria.

La siguiente instrucción LD A;03H en la localidad de memoria (0005) es otra de carga LD en el registro A, con él numero 03H, el microprocesador almacena en el registro A, el numeró 03H.

La instrucción OUT (41H),A indica al microprocesador Z-80 que envié los datos (03H), que están en el registro A, al puerto de salida (41H), con esto se ejecuta el primer paso del motor a pasos.
la instrucción CALL 0066H es una llamada de una subrutina ubicada en la localidad de memoria 0066H, está subrutina es un retardo para poder apreciar cada uno de los pasos que da el motor.

La subrutina de la localidad 0066H, tienen la instrucción (LD D,FFH), que como la anterior es una carga LD de un numero FFH en el registro D,

la siguiente instrucción (DEC D) en la localidad de memoria (0067) es un decremento y resta un bit al registro D, teniendo ahora el valor FEH

la siguiente instrucción (JP NZ,0067) en la localidad de memoria (0068) es un brinco condicional, de no cero la cual hace que brinque a una localidad de memoria (0067), si la bandera Z es un cero salta a la localidad de memoria (0067), y cuando la bandera Z sea un 1 dejara de saltar, y continuara con la siguiente localidad de memoria.

La siguiente instrucción RET en la localidad de memoria (0069) es un retorno a la localidad de memoria (0008) que es la localidad siguiente después de que se ejecuto la primera instrucción CALL 0066H.

Ahora se ejecuta la instrucción LD A,(06H), en la localidad de memoria (0008) que es otra carga pero con un valor diferente y así se continua con la localidad de memoria (0009) que

es salida de datos al puerto (41H), y otra vez la llamada a una subrutina, hasta terminar 16 pasos a la derecha.

En la localidad de memoria (0035) se encuentra la misma secuencia solo que los valores de carga se invierten esto es para el giro de la derecha es 03H, 06H, 0CH, 09H, para el de la izquierda es 09H, 0CH, 06H, 03H, al terminar estos 16 pasos a la izquierda, en la localidad de memoria (0065) hay un brinco a la localidad de memoria (0002), comenzando de nuevo el programa

# Capitulo 5

# ANÁLISIS FUNCIONAL DEL CONTROL CON MICROCONTROLADOR PIC 16F84

En el sistema mínimo con microcontrolador PIC 16F84, veremos como este dispositivo, ocupa menor numero de componentes externos ya que solo basta con un circuito de reloj, de reset, y un circuito de potencia observaremos la diferencia en el tamaño del programa que se utiliza para el control del motor, así como el numero de instrucciones utilizadas durante la programación

## 5.1 IMPLEMENTACIÓN DE UN CIRCUITO DE CONTROL CON UN MICROCONTROLADOR PIC 16F84

En la figura 4.3 se muestra el diagrama a bloques del circuito de control con microcontrolador PIC 16F84


Figura 4.3 diagrama a bloques del circuito de control con microcontrolador pic 16F84

Como se puede observar en el diagrama a bloques el microcontrolador realiza la mayor parte del control de señales de entrada y de salida las cuales comunican a la etapa de potencia los pasos que debe realizar el motor.

## 5.2 ANÁLISIS DEL DIAGRAMA A BLOQUES CON MICROCONTROLADOR PIC 16F84.

En el diagrama a bloque de la figura 4.3, se observa que tres de los bloques están contenidos en un bloque mas grande, de estos el primer bloque llamado puerto de entrada RA0 RA4 es el que recibe la información del mundo exterior para y que es enviado al segundo bloque.

El segundo bloque llamado de control es donde se encuentra el programa y la rutina de control de la misma información que llevara acabo para el funcionamiento del motor a pasos.

Y el tercer bloque llamado puerto de salida RB7-RB4 que es el encargado de enviar la información del microcontrolador a la etapa de potencia la cual hará funcionar al motor a pasos

La etapa de potencia en esta solo se encarga de aumentar la corriente para hacer funcionar al motor a pasos.

## 5.3 IMPLEMENTACIÓN DE UN CIRCUITO DE CONTROL

En la figura 4.4 se muestra el diagrama de las componentes del circuito de control con microcontrolador Pic 16F84.

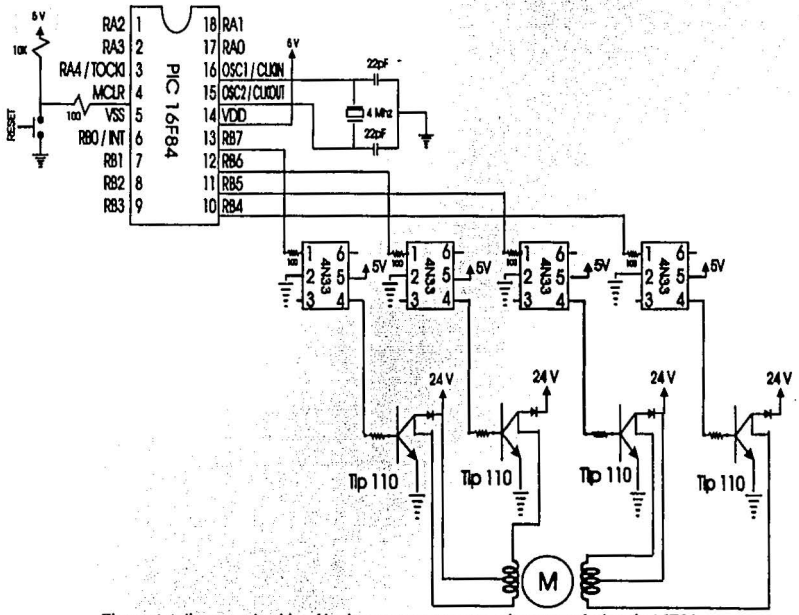



Figura 4.4 diagrama de ubicación de componentes con microcontrolador pic 16F84

En el diagrama de la figura 4.4, se puede ver un dispositivo principal que se encargara del control del motor de las señales de entrada / salida y una etapa de potencia que se encargara de alimentar al motor con la corriente necesaria para su funcionamiento.

## 5.4 ANÁLISIS DEL CIRCUITO

Como se puede observar en la figura 4.4. los pines 14 y 5 del microcontrolador es por donde se polariza con +5 Volts en VDD y tierra en VSS.

El microcontrolador pic 16F84 tiene conectados un cristal de cuarzo de 4 MHz con dos capacitores de 22 picó faradios en sus entradas 16 y 17 los cuales se encargan de proporcionar la frecuencia de trabajo o los pulsos de reloj al microcontrolador.

También tiene un una resistencia de 10K y un push botón conectados a una resistencia de 100 ohms en la entrada 4 del microcontrolador este pequeño circuito esta encargado de proporcionar un reset al dispositivo con el fin de inicializarlo en sus valores óptimos.

Y en los pines 10 a 13 correspondientes a RB4, RB5, RB6 y RB7 los cuales forman parte del puerto de salida B, por estos pines saldrán la secuencia de señales que harán funcionar a los optoacopladores 4N33.

Los optoacopladores reciben la señal del puerto B del Microcontrolador y lo envían al transistor de potencia Tip 110 los cuales están encargados de aumentar la potencia y la corriente para que el motor a pasos gire

## 5.5 PROGRAMA DE FUNCIONAMIENTO PARA UN MOTOR A PASOS

En la tabla 4.4 se muestra el programa para el funcionamiento de un motor a pasos.

```
LIST        P=16f84              ;selección de microcontrolador
RADIX       HEX                  ;formato hexadecimal

;-------------------------------------------------------------------------
ESTADO              EQU  0x03
PUERTOB             EQU  0x06
PASO1               EQU  0x0C
PASO2               EQU  0x0D
PASO3               EQU  0x0E
PASO4               EQU  0x0F
TMR0_OPT            EQU  0x01
conta1      EQU     0x10
conta2      EQU     0x11
;-------------------------------------------------------------------------
ORG  0      ; El programa comienza en la dirección 0 y
     goto   inicio  ; salta a la dirección 5 para sobrepasar la interrupción
```

continua

```
inicio  bsf     ESTADO,5    ; Selecciona el banco 1
        movlw   0x00        ; carga en el registro W en numero cero
        movwf   PUERTOB     ; Se configura Puerto B como salida
        movlw   0xD6        ;carga el numero D6 en el registro W
        movwf   TMR0_OPT    ;carga en TMR_OPT lo que esta en el registro W
        bcf     ESTADO,5    ; Selección del banco 0
        movlw   0x03        ;carga el numero 3 en el registro W
        movwf   conta1 ;carga en conta1 la que esta en el registro W
        movwf   conta2 ;carga en conta2 lo que esta en el registro W
        movlw   0xC0        ;carga el numero C0 en el registro W
        movwf   PASO1       ;carga en paso1 lo que esta en el registro W
        movlw   0x60        ;carga el numero 60 en el registro W
        movwf   PASO2       ;carga en paso2 lo que esta en W
        movlw   0x30        ;carga el numero 30 en el registro W
        movwf   PASO3       ;carga en paso3 lo que esta en el registro W
        movlw   0x90        carga el numero 90 en el registro W
        movwf   PASO4       ;carga en paso4 la que esta en el registro W
;-------------------------------------------------------------------------------
der     movfw   PASO1       ;carga en el registro W lo que esta en paso1
        movwf   PUERTOB     ;saca por el puerto B lo que esta en el registro W
        call    reta        ;llama a la subrutina reta (retardo)
        movfw   PASO2       ;carga en el registro W lo que esta en paso2
        movwf   PUERTOB     ;saca por el puerto B lo que esta en el registro W
        call    reta        ;llama a la subrutina reta (retardo)
        movfw   PASO3       ;carga en el registro W lo que esta en paso 3
        movwf   PUERTOB     ;saca por el puerto B lo que esta en el registro W
        call    reta        ;llama a la subrutina reta (retardo)
        movfw   PASO4       ;carga en el registro W lo que esta en paso4
        movwf   PUERTOB     ;saca por el puerto B lo que esta en el registro W
        call    reta        ;llama a la subrutina reta (retardo)
        decfsz  conta1 ;verifica si conta1=0,si no decrementa y ,
                            ;continua si conta1=0 brinco una
                            ;instrucción
        goto    der         ;brinca a la etiqueta der
;-------------------------------------------------------------------------------
izq     movfw   PASO4       ;carga en el registro W lo que esta en paso4
        movwf   PUERTOB     ;saca por el puerto B lo que esta en el registro W
        call    reta        llama a la subrutina reta (retardo)
        movfw   PASO3       ;carga en el registro W lo que esta en paso3
        movwf   PUERTOB     ;saca por el puerto B lo que esta en W
        call    reta        ;llama a la subrutina reta (retardo)
        movfw   PASO2       ;carga en el registro W lo que esta en paso2
```

continua

```
        Movwf   PUERTOB     ;saca por el puerto B lo que esta en el registro W
        call    reta        llama a la subrutina reta (retardo)
        movfw   PASO1       ;carga en el registro W lo que esta en paso1
        movwf   PUERTOB     ;saca por el puerto B lo que esta en el registro W
        call    reta        llama a la subrutina reta (retardo)
        decfsz  conta2      ;verifica si conta1=0,si no decrementa y ,
                            ;continua si conta1=0 brinco una
                            ;instrucción
        goto    lzq         ;brinco a la etiqueta lzq
        goto    fin         ;brinca a la etiqueta fin
        reta    clrf    TMR0_OPT
exp     btfss   TMR0_OPT,4
        goto    exp         ;brinca a la etiqueta exp
        return
        fin     nop
        end
```

Tabla 4.4 programa para el funcionamiento de un motor a pasos para el microcontrolador pic 16F84

## 5.6 ANÁLISIS DEL PROGRAMA

Como primer paso se explicará el formato de edición del programa, se puede usar cualquier editor que use caracteres ASCII nombrándolo siempre con la extensión asm, y los enunciados después de (;) solo son para indicar lo que realiza el programa.

En el primer renglón indicamos el tipo de dispositivo que se usa y el sistema de base numérico, de los cuales son decimal hexadecimal o binario.

Los siguientes renglones donde sé allá contenido la expresión EQU es la denominada zona de etiquetas

Y por ultimo los nemónicos que se emplearan para la programación, el uso de mayúsculas y minúsculas obedece a una serie de regla o normas de estilo, comunes entre los programadores en ensamblador, que aunque no son obligatorios, facilitan la lectura del código fuente, un resumen de las reglas empleadas es el siguiente:

Directivas del compilador en mayúsculas
Nombres de variables en mayúsculas
Nemónicos en minúsculas
Programa bien tabulado

Como ya se menciono el primer renglón contiene la información del tipo de microcontrolador que se va a emplear, seguido de la base numérica del sistema, las cuales son las siguientes:

Decimal d'12'
Hexadecimal 0x0c / h'0c' / 0c / 0ch
Binario b'1010'

En nuestro caso usamos el sistema hexadecimal escribiendo RADIX HEX.

En la zona de etiquetas llamada asi porque aquí es donde se nombra la localidad de memoria de datos donde se guardaran los datos por ejemplo;

| Nombre de etiqueta | EQU | localidad de la memoria de datos |
|---|---|---|
| PUERTOB | Directiva al microprocesador | 0x05 |

En general lo que se hace es dar un nombre a la localidad de memoria que se va a utilizar debido a que es más fácil poner el nombre que tratar de recordar una localidad de memoria, mas adelante veremos la ventaja de esto.

La instrucción ORG 0 es un comando que indica al ensamblador la dirección de la memoria de programa donde se situará la siguiente instrucción que en este caso en el programa comienza en la posición cero donde se encuentra el reset del sistema, el porque de esta instrucción es para la el reset del dispositivo y se carguen los valores por defaul.

La instrucción goto inicio es un brinco para buscar la etiqueta inicio y ejecuta la instrucción que se encuentra en esa localidad que en este caso es la posición 05 que es donde se encuentra el puerto A.

La instrucción bsf ESTADO,5 se utiliza para cambiar de banco de datos en este caso el microcontrolador siempre trabaja en el banco cero, y esta es la instrucción que selecciona el banco para hacer esto es necesario cambiar el bit 5 del ESTADO.

Pero como ya sé había declarado la etiqueta ESTADO la cual se encuentra en la localidad 0x03es más fácil poner el nombre de la etiqueta que manejamos que tratar de recordar la posición de memoria donde se encuentra esta.
La razón por la que se selecciona el banco 1 es por que en este se hayan las configuraciones que se le deben dar a los puertos.

La instrucción movlw0x00 es una instrucción mov tipica en todos los procesadores y su misión es transferir el contenido de un registro fuente a un registro destino, en este caso se carga un valor numerico de cero en el registro W.

La instrucción movwf es otra variante de la anterior solo que es esta caso copia lo que esta en el registro W y lo deposita en la etiqueta PUERTOB el cual ya sabemos que ocupa la dirección 0x06 de la memoria pero por estar trabajando en el banco uno lo que hace es configurar la instrucción trisb, también podemos decir que pines queremos como entradas y

cuales como salidas podemos poner 0xf0 y el puerto B se configuraría los primeros 4 pines como salidas y los 4 pines siguientes como entradas, pero lo que se realiza es la configuración del PUERTOB como salida de datos en todos sus pines

La instrucción movlw 0xd6, al igual que la anterior carga en valor numérico d6 en el registro W, para posteriormente ser depositado en la etiqueta TMR_OPT por medio de la instrucción "movwf TMR_OPT", que por estar trabajando todavía en el banco uno configura la instrucción OPTION para asignar el divisor a TMR0 y hacer que divida por 128 los impulsos del reloj interno

La instrucción bcf ESTADO,5 selecciona el banco cero, esto es porque los 36 registros de propósito general se encuentran en el banco cero.

La instrucción movlw 0x03"carga el valor numérico 03 en el registro W, que posteriormente se le asigna a la instrucción movwf conta1 y a la instrucción movwf conta2, esta operación carga en la localidad de memoria ya designadas por estas etiquetas el valor de 03 con el propósito de dos contadores de 3 números.

La instrucción movlw 0xC0 carga el valor numérico en el registro W que posteriormente se almacena en la memoria con la etiqueta paso 1, y se continua cargando los valores numéricos de las siguientes etiquetas de paso 2 , paso3 , y paso 4, con la secuencia 60, 30, y 90 respectivamente, en este conjunto de instrucciones se programo la memoria con la secuencia de datos que hacen funcionar al motor a pasos

En la instrucción movfw paso1 se carga el valor numérico en el registro W, que esta contenido en la memoria en la etiqueta paso1 la cual es C0 que fue la que se cargo anteriormente, posteriormente este dato que se encuentra en el registro w se enviara al puertoB con la instrucción "movwf PUERTO"B el cual es el dato C0 ejecutándose el primer paso en el motor a pasos.

La siguiente instrucción "call reta" es una subrutina de retardo para poder apreciar el paso que da el motor a pasos esta instrucción busca la etiqueta llamada call y ejecuta la instrucción que ahí se encuentra mas adelante explicaremos esta subrutina.

Las siguientes instrucciones ejecutan los siguientes pasos enviando por el PUERTOB la secuencia 60, 30, y 90 con su respectiva subrutina de retardo para lograr que gire a la derecha, y al llegar a la instrucción decfsz conta1 aquí el programa verifica que el dato contenido en conta1 sea un cero, si no es un cero decrementa el valor que ahí se encuentre que en este caso pasaría de 03 a 02, y continua con la instrucción siguiente que es goto der que es un brinco de localidad de memoria y ejecutara la instrucción que ahí se encuentre que en este caso es volver a dar un paso1 y va a continuar así hasta que llegue a la instrucción decfsz conta1 y que al verificar su valor sea un cero entonces el programa se brincara la instrucción de "goto der" y comenzara un ciclo de pasos o giros a la izquierda, en total el programa debe realizar 12 pasos o giros a la derecha y después va a realizar 12 pasos o giros a la izquierda

La subrutina de retardo comienza con la instrucción clrf TMR_OPT, la cual limpia el registro TMR0 y carga puros ceros, posteriormente la instrucción btfss TMR0_OPT,4 verificara que el bit 4 del temporizador TMR0 sea un si no es un continua con la instrucción goto exp y va a verificar de nuevo la instrucción btfss TMR0_OPT,4 esto hasta que el temporizador TMR0 en su bit numero 4 exista un uno, entonces el programa salta el goto y ejecuta return, y retorna la posición de memoria de donde fue realizada la llamada call.

El temporizador TMR0 tiene un contador que al pasar 128 ciclos de reloj el contador se incrementa en uno la siguiente formula :

Temporización = 4*Tosc * valor cargado en TMR0 (bit 4)* rango del divisor

Empleando la formula anterior tenemos que:

Temporización = 4 *250ns*16*128 = 512microseg.

El cual, al no ser muy apreciable se tendría que modificar los valores del temporizador a los siguientes:

Temporización = 4 *250ns*128*255 = 32.64miliseg.

Este valor sigue siendo muy pequeño usando el 7 bit, una solución es bajar la frecuencia de reloj o usar el señalizador que nos indica cuando se produce el desbordamiento del temporizador. Que esta en el registro INTCON dirección 0x0b y 0x8b.

En concreto es el bit T0IF,que es el bit 2. el señalizador T0IF debe ser borrado por software; de ahí la instrucción bcf INTCON,2 antes de retornar de la subrutina
La subrutina de retardo quedaría entonces de la siguiente manera:

```
INTCON          EQU   0x0b          ;etiqueta de configuracion

        movlw   0xd7                ;valor a cargar en OPTION
reta    clrf    TMR0_OPT    ;       TMR0 = 0 y Comienza su incremento
exp     btfss   INTCON,2            ;TMR= <6> = 1?
        Goto    exp                 ;no a llegado TMR0 a 255
        bcf     INTCON,2            ;pone el flag a o por software
        Return                      ;ha llegado a TMR0 al valor 255 y retorna
End
```

# Conclusiones

El microprocesador Z-80 es un dispositivo de gran compatibilidad con otras componentes, también funciona como dispositivo de experimentación ya que es muy practico y sencillo. Para la manipulación de la familia de estos dispositivos, que cada día son mas y más avanzados, se ocupa con un repertorio de 156 instrucciones y puede trabajar a frecuencias de 4, 6.5, y 8 MHz. Con el manejo de tres buses: uno de datos uno de control y uno de direcciones, se pueden implementar sistemas muy grandes o muy sencillos, dependiendo de la aplicación.

En particular para el control de un motor a pasos con microprocesador Z-80 se requirieron de dispositivos adicionales: una memoria ROM de 2K para el programa y un dispositivo de puertos de salida / entrada, los cuales forman un sistema mínimo especifico. Una ventaja al usar este sistema es que se pueden conectar un total de 6 motores a pasos para su experimentación, no importando su funcionamiento en especial ya que son controlados por el programa que se emplee. Otra ventaja de este sistema es que la memoria se puede sustituir por otra de mayor tamaño, con un programa diferente para hacer funcionar al motor en tareas especificas o determinadas estas tareas podrían ser la velocidad con que se quiere hacer el giro, aumentar o disminuir el número de giros; sin tener que volver a reprogramar todo de nuevo, es decir que solo se guardarían los programas en una memoria diferente

Entre las desventajas de este empleo de sistemas es el costo, ya que un sistema seria muy caro si solo se quiere controlar un solo motor. Ya que se necesitan otros dispositivos externos para el funcionamiento del microprocesador. Pero es muy eficiente si se desea controlar más de tres o cuatro motores, porque no se tendrían que hacer mas circuitos de control ya que este sistema no tendría que crecer más que en solo las etapas de potencia respectivas de cada motor.

Otra desventaja es la programación porque los datos o información para el manejo del motor a pasos se debe de almacenar en una y solo una localidad de memoria, aunque el dato sea el mismo, esto nos lleva a repetir las instrucciones o datos haciendo muy grande el programa, esto se vio en el capitulo 4, pero en general las instrucciones son fáciles de entender y se puede editar el programa en cualquier editor de textos

Y por ultimo otra desventaja en este sistema es su número de conexiones que se tienen que hacer para su funcionamiento, pues se tienen que realizar conexiones en un bus de datos, uno de control, otro de direcciones y un último en sus dispositivos de alimentación y circuitos de reloj

Un microcontrolador es un dispositivo más económico y sencillo de manipular, existe una gran familia de estos dispositivos dependiendo de la aplicación que se les desee emplear. En el control de un motor a pasos con un microcontrolador se emplearon un menor numero de componentes externos ya que este dispositivo se encarga casi en su totalidad del control

de la memoria y de las instrucciones que dan origen a los giros del motor, por estar estas integradas en el mismo.

Este sistema es muy práctico para el control de uno, dos, o tres motores a pasos a la vez. Entre las ventajas de este sistema tenemos que es barato, es reducido en su circuitería, ya que solo se debe de conectar la alimentación, su circuito de reloj y los circuitos que va a controlar, no teniendo que realizar más conexiones.

Otra ventaja mas es su programación ya que con solo 35 instrucciones el microcontrolador realiza muchas tareas específicas y si un dato que se emplea en el control debe ser repetido solo basta con guardar ese dato en una localidad de memoria y llamar a ese dato las veces que sea necesario, reduciendo en cierta manera el trabajo de estar repitiendo instrucciones. También se puede utilizar cualquier editor de textos para realizar el programa.

Entre las desventajas que se pueden mencionar es que no se puede ampliar su capacidad para poner mas de tres motores a pasos, ya que solo tiene 13 lineas de salida, y si tomamos en cuenta que los motores presentan cuatro cables de conexión para su funcionamiento, no alcanzarían.

También se debe de reprogramar cada vez que se desee que el motor haga algo diferente, ya sea mayor numero de pasos o mayor velocidad, esto implica tener que borrar el programa anterior y grabar el nuevo programa en la memoria, la cual es limitada y solo se pueden usarlos registros que se tienen disponibles (68).

Sobre los motores a pasos, se puede concluir que cada día son mas utilizados este tipo de motores en diferentes áreas de la ingeniería, por ser un tipo de motor preciso y de fácil control, además de que existen varios tipos diferentes en tamaño, consumo, y fuerza.

En general los dos sistemas de control son útiles porque si se desea diseñar con microprocesador se tiene un sistema potente de altas prestaciones pero ligeramente caro aproximadamente unos 180 pesos si la aplicación solo se limita a un o dos motores.

Y si se desea un sistema con microcontrolador tenemos un sistema practico económico aproximadamente unos 90 pesos pero reducido en prestaciones de mayor aplicación

Una propuesta interesante seria controlar tres motores a pasos para que movieran una grúa en el espacio de tres dimensiones, cada uno en su respectivo eje cartesiano X,Y,Z, ya sea con un microcontrolador o con el microprocesador.

También como mejora del trabajo se podría usar las interrupciones del microcontrolador mediante sensores para que el motor a pasos se detenga y haga funcionar otro motor mientras este se regresa a la posición inicial.

# Apéndice A

## INSTRUCCIONES DEL MICROPROCESADOR Z-80.

Instrucciones de carga de 8 bits

| Instrucción | LD r,r' | Comentario | |
|---|---|---|---|
| Descripción | Carga el contenido del registro r' en el registro r | r,r' | Reg. |
| Código de operación | 01 r r' | 000 | B |
| | | 001 | C |
| Instrucción | LD r,n | 010 | D |
| Descripción | Carga el valor n en el registro r | 011 | E |
| Código de operación | 00 r 110 | 100 | H |
| | < n > | 101 | L |
| Instrucción | LD r,(HL) | 111 | A |
| Descripción | Carga el registro r con el valor de la localidad (HL) | | |
| Código de operación | 01 r 110 | | |
| Instrucción | LD r,(IX+d) | | |
| Descripción | Carga el registro r con el valor de la localidad (IX+d) | | |
| Código de operación | 11 011 101 | | |
| | < d > | | |
| Instrucción | LD r,(IY+d) | | |
| Descripción | Carga el registro r con el valor de la localidad (IY+d) | | |
| Código de operación | 11 111 101 | | |
| | < d > | | |
| Instrucción | LD (HL),r | | |
| Descripción | Carga la localidad (HL) con el registro r | | |
| Código de operación | 01 100 r | | |
| Instrucción | LD (IX+d),r | | |
| Descripción | Carga la localidad (IX+d) con el registro r | | |
| Código de operación | 11 011 101 | | |
| | 01 110 r | | |
| Instrucción | LD (IY+d),r | | |
| Descripción | Carga la localidad (IY+d) con el registro r | | |
| Código de operación | 11 111 101 | | |
| | 01 110 r | | |

Continua

| Instrucción | LD (HL),n |
|---|---|
| Descripción | Carga la localidad (HL) con el valor de n |
| Código de operación | 00 110 110<br>< n > |
| Instrucción | LD (IX+d),n |
| Descripción | Carga la localidad (IX+d) con el valor n |
| Código de operación | 11 011 101<br>00 110 110<br>< d ><br>< n > |
| Instrucción | LD (IY+d),n |
| Descripción | Carga la localidad (IY+d) con el valor n |
| Código de operación | 11 111 101<br>00 110 110<br>< d ><br>< n > |
| Instrucción | LD A,(BC) |
| Descripción | Carga el registro A con el valor de la localidad (BC) |
| Código de operación | 00 001 010 |
| Instrucción | LD A,(DE) |
| Descripción | Carga el registro A con el valor de la localidad (DE) |
| Código de operación | 00 011 010 |
| Instrucción | LD A,(nn) |
| Descripción | Carga el registro A con el valor de la localidad nn. |
| Código de operación | 00 111 010<br>< n ><br>< n > |
| Instrucción | LD A,I |
| Descripción | Carga el registro A con I.(vector de interrupción) |
| Código de operación | 11 101 101<br>01 010 111 |
| Instrucción | LD A,R |
| Descripción | Carga el registro A con R.(registro de refresco de memoria) |
| Código de operación | 11 101 101<br>01 011 111 |
| Instrucción | LD I,A |
| Descripción | Carga I con el registro A |
| Código de operación | 11 101 101<br>01 000 111 |

| r,r' | Reg. |
|---|---|
| 000 | B |
| 001 | C |
| 010 | D |
| 011 | E |
| 100 | H |
| 101 | L |
| 111 | A |

Continua

| Instrucción | LD R,A | |
|---|---|---|
| Descripción | Carga R con el registro A | |
| Código de operación | 11 101 101<br>01 001 111 | |

Instrucciones de carga de 16 bits

| Instrucción | LD dd,nn | dd Registro |
|---|---|---|
| Descripción | Carga los registros dd con el valor nn | 00 BC |
| Código de operación | 00 dd0 001<br>< n ><br>< n > | 01 DE<br>10 HL<br>11 SP |
| Instrucción | LD IX,nn | |
| Descripción | Carga IX con el valor nn | |
| Código de operación | 11 011 101<br>00 100 001<br>< n ><br>< n > | |
| Instrucción | LD IY,nn | |
| Descripción | Carga IY con el valor nn. | |
| Código de operación | 11 111 101<br>00 100 001<br>< n ><br>< n > | |
| Instrucción | LD HL,(nn) | |
| Descripción | Carga HL con el valor de la localidad nn | |
| Código de operación | 00 101 010<br>< n ><br>< n > | |
| Instrucción | LD dd,(nn) | |
| Descripción | Carga el par de registros dd con el valor de la localidad nn | |
| Código de operación | 11 101 101<br>01 dd1 011<br>< n ><br>< n > | |
| Instrucción | LD IX,(nn) | |
| Descripción | Carga IX con el valor de la localidad (nn) | |
| Código de operación | 11 011 101<br>00 101 010<br>< n ><br>< n > | |

Continua

| Instrucción | LD IY,(nn) | dd | Registro |
|---|---|---|---|
| Descripción | Carga IY con el valor de la localidad (nn) | 02 | BC |
| Código de operación | 11 111 101<br>00 101 010<br>< n ><br>< n > | 03 | DE |
| | | 12 | HL |
| | | 13 | SP |
| Instrucción | LD (nn),HL | | |
| Descripción | Carga la localidad (nn) con HL | | |
| Código de operación | 00 100 010<br>< n > | | |
| Instrucción | LD (nn),dd | | |
| Descripción | Carga la localidad nn con el par de registros dd | | |
| Código de operación | 11 101 101<br>01 dd0 011<br>< n ><br>< n > | | |
| Instrucción | LD (nn),IX | | |
| Descripción | Carga la localidad (nn) con el registro IX | | |
| Código de operación | 11 011 101<br>00 100 010<br>< n ><br>< n > | | |
| Instrucción | LD (nn),IY | | |
| Descripción | Carga la localidad (nn) con el registro IY | | |
| Código de operación | 11 111 101<br>00 100 010<br>< n ><br>< n > | | |
| Instrucción | LD SP,HL | | |
| Descripción | Carga SP con el registro HL | | |
| Código de operación | 11 111 001 | | |
| Instrucción | LD SP,IX | | |
| Descripción | Carga SP con el registro IX | | |
| Código de operación | 11 011 101<br>11 111 001 | | |
| Instrucción | LD SP,IY | | |
| Descripción | Carga SP con el registro IY | | |
| Código de operación | 11 111 101<br>11 111 001 | | |

Continua

| Instrucción | PUSH qq | Qq par |
|---|---|---|
| Descripción | Carga el par de registros qq hacia la pila | 00 BC |
| Código de operación | 11 qq0 101 | 01 DE<br>10 HL |
| Instrucción | PUSH IX | 11 AF |
| Descripción | Carga el registro IX hacia la pila | |
| Código de operación | 11 011 101<br>11 100 101 | |
| Instrucción | PUSH IY | |
| Descripción | Carga el registro IY hacia la pila stack | |
| Código de operación | 11 111 101<br>11 100 101 | |
| Instrucción | POP qq | |
| Descripción | Carga el par de registro qq con la parte alta de la pila | |
| Código de operación | 11 qq0 001 | |
| Instrucción | POP IX | |
| Descripción | Carga IX con parte alta de la pila | |
| Código de operación | 11 011 101<br>11 100 001 | |
| Instrucción | POP IY | |
| Descripción | Carga IY con la parte alta de la pila | |
| Código de operación | 11 111 101<br>11 100 001 | |

Instrucciones transferencia intercambio y búsqueda

| Instrucción | EX DE,HL | |
|---|---|---|
| Descripción | Intercambia los contenidos de DE y HL | |
| Código de operación | 11 101 011 | |
| Instrucción | EX AF,AF | |
| Descripción | Intercambia los contenidos de AF y AF' | |
| Código de operación | 00 001 000 | |
| Instrucción | EXX | El banco de registros y el banco de registros auxiliares se intercambian |
| Descripción | Intercambia los contenidos de BC, DE, HL con BC',DE',HL' | |
| Código de operación | 11 011 001 | |

Continua

| | | |
|---|---|---|
| Instrucción | EX (SP),HL | |
| Descripción | Intercambia la localidad (SP) y el registro HL. | |
| Código de operación | 11 100 011 | |
| Instrucción | EX (SP),IX | |
| Descripción | intercambia la localidad (SP) y el registro IX | |
| Código de operación | 11 011 101 | |
| Instrucción | EX (SP),IY | |
| Descripción | Intercambia la localidad (SP) y el registro IY | |
| Código de operación | 11 100 011 | |
| Instrucción | LDI | Carga (HL) dentro de (DE), incrementa el puntero y decrementa el contador de byte (BC) |
| Descripción | Carga la localidad (DE) con la localidad (HL), incrementa DE, HL; y decrementa BC. | |
| Código de operación | 11 101 101<br>10 100 000 | |
| Instrucción | LDIR | |
| Descripción | Realice un LDI y repite hasta que BC = 0 | |
| Código de operación | 11 101 101 si BC diferente 0<br>10 110 000 si BC = 0 | |
| Instrucción | LDD | |
| Descripción | Carga la localidad (DE) con la localidad (HL), y decrementa DE, HL, BC | |
| Código de operación | 11 101 101<br>10 101 000 | |
| Instrucción | LDDR | |
| Descripción | Realiza un LDD y repite hasta que BC = 0 | |
| Código de operación | 11 101 101 si BC diferente 0<br>10 111 000 si BC = 0 | |
| Instrucción | CPI | |
| Descripción | Compara la localidad (HL) incrementa HL, y decrementa BC. | |
| Código de operación | 11 101 101<br>10 100 001 | |
| Instrucción | CPIR | |
| Descripción | Realiza un CPI y repite hasta que BC = 0 | |
| Código de operación | 11 101 101 si BC diferente 0 y A diferente (HL)<br>10 110 001 si BC = 0 o A = (HL) | |

Continua

| Instrucción | CPD | |
|---|---|---|
| Descripción | Compare la localidad (HL) decrementa HL y BC | |
| Código de operación | 11 101 101<br>10 101 001 | |
| Instrucción | CPDR | |
| Descripción | Realiza un CPD y repite hasta que BC = 0. | |
| Código de operación | 11 101 101 si BC diferente 0 y A diferente (HL)<br>10 111 001 si BC = 0 o A = (HL) | |

Instrucciones aritméticas y lógicas de 8 bits

| Instrucción | ADD A,r. | r registro<br>000 B<br>001 C<br>010 D<br>011 E<br>100 H<br>101 L<br>111 A |
|---|---|---|
| Descripción | Suma el registro r al registro A | |
| Código de operación | 10 [000] r | |
| Instrucción | ADD A,n | |
| Descripción | Suma el valor de n al registro A | |
| Código de operación | 11 [000] 110<br>< n > | |
| Instrucción | ADD A,(HL) | |
| Descripción | Suma la localidad (HL) al registro A | |
| Código de operación | 10 [000] 110 | |
| Instrucción | ADD A,(IX+d) | |
| Descripción | Suma la localidad (IX+d) al registro A | |
| Código de operación | 11 011 101<br>10 [000] 110 | |
| Instrucción | ADD A,(IY+d) | |
| Descripción | Suma la localidad (IY+d) al registro A | |
| Código de operación | 11 111 101<br>10 [000] 110<br>< d > | |
| Instrucción | ADC A,s. | El S es uno del registros del los r, n, (HL), (IX+d), (IY+d) como mostrados en las instrucciones de ADD la instrucción. Los reemplazar los bits indicados 000 en el ADD de la las instrucciones anteriores |
| Descripción | Suma con acarreo el registro s en el registro A | |
| Código de operación | [001] | |
| Instrucción | SUB s | |
| Descripción | Substraiga el registro s desde el registro A | |
| Código de operación | [010] | |

Continua

| | | |
|---|---|---|
| Instrucción | SBC A,s | El S es uno del registros del los r, n, (HL), (IX+d), (IY+ d) como mostrados en las instrucciones de ADD solo reemplaze los bits indicados 000 en el ADD de la las instrucciones anteriores |
| Descripción | Substraiga el registro s del registro A con acarreo | |
| Código de operación | 011 | |
| Instrucción | AND s | |
| Descripción | AND lógico del registro r al registro A | |
| Código de operación | 100 | |
| Instrucción | OR s | |
| Descripción | OR lógico del registro r y el registro A | |
| Código de operación | 110 | |
| Instrucción | XOR s | |
| Descripción | OR Exclusivo del registro r y el registro A | |
| Código de operación | 101 | |
| Instrucción | CP s | |
| Descripción | Compara el registro r con el registro A | |
| Código de operación | 111 | |
| Instrucción | INC r | |
| Descripción | Incrementa el registro r | |
| Código de operación | 00 r 100 | |
| Instrucción | INC (HL) | |
| Descripción | Incrementa la localidad (HL) | |
| Código de operación | 00 110 100 | |
| Instrucción | INC (IX+d) | |
| Descripción | Incrementa la localidad (IX+d) | |
| Código de operación | 11 011 101<br>00 110 100<br>< d > | |
| Instrucción | INC (IY+d) | |
| Descripción | Incrementa la localidad (IY+d) | |
| Código de operación | 11 111 101<br>00 110 100<br>< d > | |

Continua

| Instrucción | DEC m | m es uno de los registros r, (HL), (IX+d), (IY+d) como se mostró en INC DEC solo reemplace 100 con 101 |
|---|---|---|
| Descripción | Decrementa el registro m | |
| Código de operación | 101 | |

Instrucciones aritméticas de propósito general y de control del CPU

| Instrucción | DAA | Ajusta el registro A con complemento decimal a unos Complemento negativo a dos Completentas los acarreos y banderas en paquetes BCD |
|---|---|---|
| Descripción | Ajusta a decimales el registro A | |
| Código de operación | 00 100 111 | |
| Instrucción | CPL | |
| Descripción | Complementa el registro A a unos | |
| Código de operación | 00 101 111 | |
| Instrucción | NEG | |
| Descripción | Niega el registro A a complemento de 2 | |
| Código de operación | | |
| Instrucción | CCF | |
| Descripción | Complementa las banderas de acarreo | |
| Código de operación | 00 111 111 | |
| Instrucción | SCF | |
| Descripción | Acarrea el juego de banderas a complemento de uno | |
| Código de operación | 00 110 111 | |
| Instrucción | NOP | |
| Descripción | No operación | |
| Código de operación | 00 000 00 | |
| Instrucción | HALT. | |
| Descripción | Para el CPU y espera la interrupción | |
| Código de operación | 01 110 110 | |
| Instrucción | DI | Indica las interrupciones a realizar |
| Descripción | Desactiva las interrupciones (excepto NMI en 0066h) | |
| Código de operación | 11 110 011 | |

Continua

| Instrucción | EI | | | Indica las interrupciones a realizar |
|---|---|---|---|---|
| Descripción | Activa las interrupciones | | | |
| Código de operación | 11 | 111 | 011 | |
| Instrucción | IM 0 | | | |
| Descripción | Modo de interrupción 0 (interrupción en el bus de datos y por interrupción de un dispositivo) | | | |
| Código de operación | 11 | 101 | 101 | |
| | 01 | 000 | 110 | |
| Instrucción | IM 1 | | | |
| Descripción | Modo de interrupción uno | | | |
| Código de operación | 11 | 101 | 101 | |
| | 01 | 010 | 110 | |
| Instrucción | IM 2 | | | |
| Descripción | Modo de interrupción dos (vector de salto) | | | |
| Código de operación | 11 | 101 | 101 | |
| | 01 | 011 | 110 | |

Instrucciones aritméticas de 16 bit

| Instrucción | ADD HL,ss | | | Ss | registro |
|---|---|---|---|---|---|
| Descripción | Suma par de registros ss con HL | | | 00 | BC |
| Código de operación | 00 | ss1 | 001 | 01 | DE |
| | | | | 10 | HL |
| Instrucción | ADC HL,ss | | | 11 | SP |
| Descripción | Suma con acarreo el par de registros ss con HL | | | | |
| Código de operación | 11 | 101 | 101 | | |
| | 01 | ss1 | 010 | | |
| Instrucción | SBC HL,ss | | | | |
| Descripción | Subtrae con acarreo el par de registros ss con HL | | | | |
| Código de operación | 11 | 101 | 101 | | |
| | 01 | ss0 | 010 | | |
| Instrucción | ADD IX,pp | | | pp | registro |
| Descripción | Suma el par de registros pp con IX | | | 00 | BC |
| Código de operación | 11 | 011 | 101 | 01 | DE |
| | 01 | pp1 | 001 | 10 | IX |
| | | | | 11 | SP |

Continua

| Instrucción | ADD IY,rr | rr registros |
|---|---|---|
| Descripción | Suma el par de registros rr con IY | 00 BC |
| Código de operación | 11 111 101<br>00 rr1 001 | 01 DE<br>10 IY<br>11 SP |
| Instrucción | INC ss | |
| Descripción | Incrementa el par de registros ss | |
| Código de operación | 00 ss0 011 | |
| Instrucción | INC IX | |
| Descripción | Incrementa IX | |
| Código de operación | 11 011 101<br>00 100 011 | |
| Instrucción | INC IY | |
| Descripción | Incrementa IY | |
| Código de operación | 11 111 101<br>00 100 011 | |
| Instrucción | DEC ss | |
| Descripción | Decrementa el par de registros ss | |
| Código de operación | 00 ss1 011 | |
| Instrucción | DEC IX | |
| Descripción | Decrementa IX | |
| Código de operación | 11 011 101<br>00 101 011 | |
| Instrucción | DEC IY | |
| Descripción | Decrementa IY | |
| Código de operación | 11 111 101<br>00 101 011 | |

Instrucciones que mueven y recorren bits

| Instrucción | RLCA | |
|---|---|---|
| Descripción | Gira a la izquierda en forma circular los bits del registro A | |
| Código de operación | 00 000 111 | |
| Instrucción | RLA | |
| Descripción | Gira a la izquierda con el acarreo en forma circular los bits del registro A | |
| Código de operación | 00 010 111 | |

Continua

| Instrucción | RRCA | |
|---|---|---|
| Descripción | Gira a la derecha en forma circular los bits del registro A | |
| Código de operación | 00 001 111 | |
| Instrucción | RRA | |
| Descripción | Gira a la derecha en forma circular con el acarreo los bits del registro A | |
| Código de operación | 00 011 111 | |
| Instrucción | RLC r 8 Rotate register r left circular. | |
| Descripción | Gira a la izquierda en forma circular el registro r | |
| Código de operación | 11 001 011<br>00 000 r | |
| Instrucción | RLC (HL) | r registro<br>000 B<br>001 C<br>010 D<br>011 E<br>100 H<br>101 L<br>111 A |
| Descripción | Gira ala izquierda en forma circular la localidad (HL) | |
| Código de operación | 11 001 011<br>00 000 110 | |
| Instrucción | RLC (IX+d) | |
| Descripción | Gira a la izquierda en forma circular la localidad (IX+d) | |
| Código de operación | 11 011 101<br>11 001 011<br>< d ><br>00 000 110 | |
| Instrucción | RLC (IY+d) | |
| Descripción | Gira a la izquierda en forma circular la localidad (IY+d) | |
| Código de operación | 11 111 101<br>11 001 011<br>< d > | |
| Instrucción | RL m | Instrucción formada por los estados mostrados en RLCs desde el nuevo codigo remplaza 000 como se mostró en los códigos RLCs |
| Descripción | Gira a la izquierda con acarreo en forma circular el registro m | |
| Código de operación | 010<br>m = r,(HL),(IX+d),(IY+d) | |
| Instrucción | RRC m | |
| Descripción | Gira a la derecha en forma circular el registro m | |
| Código de operación | 001<br>m = r,(HL),(IX+d),(IY+d) | |

Continua

| Instrucción | RR m | Instrucción formada por los estados mostrados en RLCs desde el nuevo codigo remplaza 000 como se mostró en los códigos RLCs |
|---|---|---|
| Descripción | Gira a la derecha con acarreo en forma circular el registro m | |
| Código de operación | 011<br>m = r,(HL),(IX+d),(IY+d) | |
| Instrucción | SLA m | |
| Descripción | Desplaza a la izquierda con acarreo el registro m | |
| Código de operación | 100<br>m = r,(HL),(IX+d),(IY+d) | |
| Instrucción | SRA m | |
| Descripción | Desplaza a la derecha con acarreo el bit mas significativo | |
| Código de operación | 101<br>m = r,(HL),(IX+d),(IY+d) | |
| Instrucción | SRL m | |
| Descripción | Desplaza a la derecha con acarreo el registro m | |
| Código de operación | 111<br>m = r,(HL),(IX+d),(IY+d) | |
| Instrucción | RLD | |
| Descripción | Gira el digito de la izquierda y el derecho entre el registro (HL) y el registro A | |
| Código de operación | 11 101 101<br>01 101 111 | |
| Instrucción | RRD | |
| Descripción | Gira el digito de la derecha y el izquierdo entre el registro (HL) y el registro A | |
| Código de operación | 11 101 101<br>01 100 111 | |

Instrucciones de prueba y reset de bits

| Instrucción | BIT b,r |
|---|---|
| Descripción | Prueba el bit b del registro r. |
| Código de operación | 11 001 011<br>01 b r |
| Instrucción | BIT b,(HL) |
| Descripción | Prueba el bit b de la localidad (HL) |
| Código de operación | 11 011 101<br>01 b 110 |

| r | registro |
|---|---|
| 000 | B |
| 001 | C |
| 010 | D |
| 011 | E |
| 100 | H |
| 101 | L |
| 111 | A |

Continua

| Instrucción | BIT b,(IX+d) | B bit de prueba |
|---|---|---|
| Descripción | Prueba el bit b de la localidad (IX+d) | 000 0 |
| Código de operación | 11 011 101<br>11 001 011<br>< d ><br>01 b 110 | 001 1<br>010 2<br>011 3<br>100 4<br>101 5<br>110 6<br>111 7 |
| Instrucción | BIT b,(IY+d) 20 | |
| Descripción | Prueba el bit b de la localidad (IY+d) | |
| Código de operación | 11 111 101<br>11 001 011<br>< d ><br>01 b 110 | |
| Instrucción | SET b,r | |
| Descripción | Ponga el bit b a uno del registro r | |
| Código de operación | 11 001 011<br>[11] b r | |
| Instrucción | SET b,(HL) | |
| Descripción | Ponga el bit b a uno de la localidad (HL) | |
| Código de operación | 11 011 101<br>[11] 001 011 | |
| Instrucción | SET b,(IX+d) | |
| Descripción | Ponga el bit b a uno de la localidad (IX+d) | |
| Código de operación | 11 011 101<br>11 001 011<br>< d ><br>[11] b 110 | |
| Instrucción | SET b,(IY+d) | |
| Descripción | Ponga el bit b a uno de la localidad (IY+d) | |
| Código de operación | 11 111 101<br>11 001 011<br>< d ><br>11 b 110 | |
| Instrucción | Res b,m | Estos forman un nuevo codigo ramplasa [11] de Set b, s, con [10] |
| Descripción | Ponga el bit b a cero de m | |
| Código de operación | 10 | |

Continua

Instrucciones de brinco o salto

| | | |
|---|---|---|
| Instrucción | JP nn | cc condición<br>000 NZ cero<br>001 Z cerop<br>010 NC no acarreo<br>011 C acarreo<br>100 PO + paridad<br>101 PE paridad<br>110 P signo positivo<br>111 M signo negativo |
| Descripción | Brinco incondicional hacia la localidad nn | |
| Código de operación | 11 000 011<br>< n ><br>< n > | |
| Instrucción | JP cc,nn | |
| Descripción | Brinca hacia la localidad nn si la condición cc es verdadera | |
| Código de operación | 11 cc 010<br>< n ><br>< n > | |
| Instrucción | JR e | e representa la extensión del modo de direccionamiento relativo |
| Descripción | Brinco incondicional relativo hacia el PC+e | |
| Código de operación | 00 011 000<br>< e – 2 > | |
| Instrucción | JR C,e | e es un signo numérico de complemento a dos in un rango de (–126,129)<br>e – 2 dirección efectiva del PC + e el PC se incrementa por 2 prioridad de la suma e |
| Descripción | Brinco relativo hacia PC+e si el acarreo = 1 | |
| Código de operación | 00 111 000<br>< e - 2 > | |
| Instrucción | JR NC,e | |
| Descripción | Brinco relativo hacia el PC+e si el acarreo = 0. | |
| Código de operación | 00 110 000<br>< e - 2 > | |
| Instrucción | JP Z,e | |
| Descripción | Brinco relativo hacia PC+e si zero es Z = 1 | |
| Código de operación | 00 101 000<br>< e – 2 > | |
| Instrucción | JR NZ,e | |
| Descripción | Brinco relativo hacia PC+e si zero es Z = 0 | |
| Código de operación | 00 100 000<br>< e – 2 > | |
| Instrucción | JP (HL) | |
| Descripción | Brinco incondicional hacia la localidad | |
| Código de operación | 11 101 001 | |
| Instrucción | JP (IX) | |
| Descripción | Brinco incondicional hacia la localidad (IX) | |
| Código de operación | 11 011 101<br>11 101 001 | |

Continua

| Instrucción | JP (IY) | |
|---|---|---|
| Descripción | Brinco incondicional hacia la localidad (IY). | |
| Código de operación | 11 111 101<br>11 101 001 | |
| Instrucción | DJNZ e | |
| Descripción | Decrementa B y realiza un salto relativo si B es diferente de 0 | |
| Código de operación | 00 010 000<br>< e – 2 > | |

Instrucciones de llamadas y retornos

| Instrucción | CALL nn | |
|---|---|---|
| Descripción | Llama a una subrutina en una localidad | |
| Código de operación | 11 001 101<br>< n ><br>< n > | |
| Instrucción | CALL cc,nn | |
| Descripción | Llama a una subrutina de una localidad nn si la condición CC es verdadera | |
| Código de operación | 11 cc 100<br>< n ><br>< n > | |
| Instrucción | RET | |
| Descripción | Retorna de una subrutina | |
| Código de operación | 11 001 001 | |
| Instrucción | RET cc | cc condición<br>000 NZ cero<br>001 Z cerop<br>010 NC no acarreo<br>011 C acarreo<br>100 PO + paridad<br>101 PE paridad<br>110 P signo positivo<br>111 M signo negativo |
| Descripción | Retorna desde una subrutina si la condición cc es verdadera | |
| Código de operación | 11 cc 000 | |
| Instrucción | RETI | |
| Descripción | Retorna desde una interrupcion | |
| Código de operación | 11 101 101<br>01 001 101 | |
| Instrucción | RETN | |
| Descripción | Retorna desde una interrupción no enmascarable | |
| Código de operación | 11 101 101<br>01 000 101 | |

Continua

| Instrucción | RST P | T p |
|---|---|---|
| Descripción | Reinicia hacia la localidad P | 000 00H |
| Código de operación | 11 t 111 | 001 08H<br>010 10H<br>011 18H<br>100 20H<br>101 28H<br>110 30H<br>111 38H |

Instrucciones de entrada y salida

| Instrucción | IN A,(n) | |
|---|---|---|
| Descripción | Carga el registro A con n desde el dispositivo de entrada | |
| Código de operación | 11 011 01<br>< n > | |
| Instrucción | IN r,(c) | |
| Descripción | Carga el registro r con (c) desde el dispositivo de entrada | |
| Código de operación | 11 101 101<br>01 r 000 | |
| Instrucción | INI | |
| Descripción | (HL) = entrada desde el puerto (C) HL = HL + 1 B = B-1. | |
| Código de operación | 11 101 101<br>10 100 010 | |
| Instrucción | INIR | |
| Descripción | Realiza un INI y repite hasta B = 0 | |
| Código de operación | 11 101 101<br>10 110 010 | |
| Instrucción | IND | |
| Descripción | (HL) = entrada desde el puerto (C). Y decrementa HL y B. | |
| Código de operación | 11 101 101<br>10 101 010 | |
| Instrucción | INDR | |
| Descripción | Realiza un IND y repite hasta que B = 0. | |
| Código de operación | 11 101 101<br>10 111 010 | |

Continua

| | | |
|---|---|---|
| Instrucción | OUT (n),A | |
| Descripción | Carga el puerto de salida (n) con lo que tiene el registro A | |
| Código de operación | 11 010 011<br>< n > | |
| Instrucción | OUT (C),r | |
| Descripción | Carga el puerto de salida (c) con lo que tiene el registro r | |
| Código de operación | 11 101 101<br>01 r 001 | |
| Instrucción | OUTI | |
| Descripción | Carga el puerto de salida (c) con (HL),incrementa HL, y decrementa B. | |
| Código de operación | 11 101 101<br>10 100 011 | |
| Instrucción | OTIR | |
| Descripción | Realiza un OTI y repite hasta que B = 0 | |
| Código de operación | 11 101 101<br>10 110 011 | |
| Instrucción | OUTD | |
| Descripción | Carga el puerto de salida (C) con (HL), decrementa HL y B. | |
| Código de operación | 11 101 101<br>10 101 011 | |
| Instrucción | OTDR | |
| Descripción | Realiza un OUTD y repite hasta que B = 0 | |
| Código de operación | 11 101 101<br>10 111 011 | |

Continua

Apéndice B

DESCRIPCIÓN DE LAS INSTRUCCIONES DE MICROCONTROLADOR PIC 16F84.

| Instrucción | ADDWF |
|---|---|
| Descripción | Suma el contenido del registro W al contenido del registro f, y almacena el resultado en W si d = 0, y en el registro f si d = 1.Si se produce acarreo el flag C se pone a "1" |
| Operación | (W) + (f) ----> (destino) |
| Sintaxis | [Etiqueta] ADDWF f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | ADDWF FSR,0<br>Si antes de la instrucción. W = 17 h y FSR = C2 h como d=0<br>Al ejecutarse: W = 17 h+ C2 h = D9 h<br>FSR = C2 h |
| Instrucción | ANDLW |
| Descripción | Efectúa la operación AND lógico entre el contenido del registro W y el literal k, y almacena el resultado en W. |
| Operación | (W).AND. (k) ----> (W) |
| Sintaxis | [Etiqueta] ANDLW k |
| Operadores | 0 < f < 255 |
| Ejemplo | ANDLW 0x5F<br>Si antes de la instrucción. W = A3 h<br>Al ejecutarse: W = 0101 1111 b AND 1010 0011 b = 0000 0011 B = 03 h |
| Instrucción | CLRF f |
| Descripción | Se borra el contenido del registro f y el flag Z se activa |
| Operación | 00h --> f<br>1 ---> Z |
| Sintaxis | [Etiqueta] CLRF f |
| Operadores | 0 < f < 127 |
| Ejemplo | CLRF REG<br>Si antes de la instrucción. REG = 5A h<br>Al ejecutarse: REG = 00 h<br>flag Z = 1 |

Continua

| Instrucción | CLRW |
|---|---|
| Descripción | El registro de trabajo W se carga con 00h. El flag Z se pone a 1 |
| Operación | 00h -->(W)<br>1 ---> Z |
| Sintaxis | [Etiqueta] CLRW |
| Operadores | No tiene |
| Ejemplo | CLRW<br>Si antes de la instrucción. W= 5Ah<br>Al ejecutarse: W = 00<br>flag Z = 1 |
| Instrucción | COMF |
| Descripción | Hace el complemento del contenido del registro f bit a bit. El resultado se almacena en el registro f si d=1 y en el registro W si d=0, en este caso f no varía. |
| Operación | (f#) -----> (dest) |
| Sintaxis | [Etiqueta] COMF f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | COMF REG1,0<br>Si antes de la instrucción. REG1 = 13 h como d= 0<br>W = EC h = 1110 1100 b<br>flag Z = 0 |
| Instrucción | DECF |
| Descripción | Se decrementa el contenido del registro f en una unidad. El resultado se almacena en f si d=1 y en W si d=0, en este caso f no varía. |
| Operación | (f)-1 --> (dest) |
| Sintaxis | [Etiqueta] DECF f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | DECF CNT,1<br>Si antes de la instrucción. CNT = 01 h<br>Z = 0<br>Al ejecutarse: CNT = 00 h<br>bit Z = 1 |

Continua

| Instrucción | DECFSZ |
|---|---|
| Descripción | Decrementa el contenido del registro f en una unidad, el resultado se almacena en f si d=1 y en W si d=0, en este caso, f no varía. Si el resultado es cero, se ignora la siguiente instrucción y, en ese caso la instrucción tiene una duración de dos ciclos. |
| Operación | (f) -1 --> (dest) ; skip if result =0 |
| Sintaxis | [Etiqueta] DECFSZ f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | EJEMPLO: HERE DECFSZ CNT,1<br>GOTO LOOP<br>CONTINUE<br>Si antes de la instrucción. PC = dirección HERE<br>Al ejecutarse: CNT = CNT –1<br>Si CNT = 0 entonces PC = dirección CONTINUE<br>Si CNT no = 0 entonces PC = dirección HERE + 1 |
| Instrucción | INCF |
| Descripción | Se incrementa en una unidad el contenido del registro f, si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W, en este caso el resultado de f no varía |
| Operación | (f) + 1 --> (dest) |
| Sintaxis | [Etiqueta] INCF f,d |
| Operadores | 0 < f < 127<br>d [0,1]<br>(f) + 1 ---> (dest) |
| Ejemplo | INCF CNT,1<br>Si antes de la instrucción:<br>CNT = FF h<br>flag Z = 0<br>Al ejecutarse: FF h +1 h = 00 h<br>CNT = 00<br>flag Z = 1 |

Continua

| Instrucción | INCFSZ |
|---|---|
| Descripción | Incrementa el contenido del registro f en una unidad, el resultado se almacena de nuevo en f si d=1, y en W si d=0, en este caso, f no varía. Si el resultado es cero, se ignora la siguiente instrucción y, en ese caso la instrucción tiene una duración de dos ciclos. |
| Operación | (f) +1 --> (dest) , skip if result = 0 |
| Sintaxis | [Etiqueta] <INCFSZ f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | HERE INCFSZ CNT,1<br>GOTO LOP<br>CONTINUE<br>Si antes de la instrucción. PC = dirección HERE<br>Al ejecutarse: CNT = CNT+1<br>Si CNT = 0<br>Entonces PC = dirección CONTINUE<br>Si CNT no = 0<br>Entonces PC = dirección HERE + 1 |
| Instrucción | IORLW |
| Descripción | Se realiza la operación lógica OR entre el registro W y el literal k. El resultado se almacena en el registro W. |
| Operación | (W).OR.k ---> (W) |
| Sintaxis | [Etiqueta] IORLW k |
| Operadores | 0 < k < 255 |
| Ejemplo | Z Se pone a 1 si el resultado de la operación es cero<br>IORLW 0x35<br>Si antes de la instrucción. W = 9A h<br>Al ejecutarse: W = 1001 1010 b + 0011 0101 b = 1011 1111 b = BFh |
| Instrucción | MOVF |
| Descripción | El contenido del registro f se carga en el registro destino dependiendo del valor de d. Si d=0 el destino es el registro W, si d=1 el destino es el propio registro f . Esta instrucción permite verificar dicho registro ya que el flag Z queda afectado. |
| Operación | (f) --> (dest) |
| Sintaxis | [Etiqueta] MOVF f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | Z Se pone a 1 si el resultado de la operación es cero<br>MOVF FSR,0<br>Al ejecutarse: W = al valor del FSR |

Continua

| | |
|---|---|
| Instrucción | MOVWF |
| Descripción | Mueve el contenido del registro W al registro f |
| Operación | (W)--> (f) |
| Sintaxis | [Etiqueta] MOVWF f |
| Operadores | 0 < f < 127 |
| Ejemplo | MOVWF OPTION<br>Si antes de la instrucción. OPTION = FF h<br>W = 4F h<br>Al ejecutarse: OPTION = 4F h<br>W = 4F h |
| Instrucción | NOP |
| Descripción | No realiza operación alguna. En realidad, se consume un ciclo de instrucción sin hacer nada. |
| Operación | no operación |
| Sintaxis | [Etiqueta] NOP |
| Operadores | No tiene |
| Ejemplo | NOP |
| Instrucción | RLF |
| Descripción | Rotación de un bit a la izquierda del contenido del registro f, pasando por el bit de acarreo C. Si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W. |
| Operación | |
| Sintaxis | [Etiqueta] RLF f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | RLF REG1,0<br>Si antes de la instrucción. REG1 = 1110 0110 b<br>flag C = 0<br>Como d= 0 el resultado queda en W<br>Al ejecutarse: REG1 = 1110 0110 b<br>W = 1100 1100 b<br>flag C = 1 |

Continua

| Instrucción | RRF |
|---|---|
| Descripción | Rotación de un bit a la derecha del contenido del registro f, pasando por el bit de acarreo C. Si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W |
| Operación | |
| Sintaxis | [Etiqueta] RRF f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | RRF REG1,0<br>Si antes de la instrucción. REG1 = 1110 0110 b<br>flag C = 1<br>Como d= 0 el resultado queda en W<br>Al ejecutarse: REG1 = 1110 0110 b<br>W = 0111 0011 b<br>flag C = 0 |
| Instrucción | SUBWF |
| Descripción | Resta en complemento a dos el contenido del registro f menos el contenido del registro W almacena el resultado en W si d=0 y en f si d=1. |
| Operación | (f) - (W) ---> (dest) |
| Sintaxis | [Etiqueta] SUBW f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | SUBWF REG1,1<br>a) Si antes de la instrucción. REG1 = 03 h<br>W = 02 h<br>flag C = ?<br>Al ejecutarse REG1 = 01h<br>W = 02 h<br>flag C = 1 ; el resultado es positivo<br>b) Si antes de la instrucción. REG1 = 02 h<br>W = 02 h<br>flag C = ?<br>Al ejecutarse REG1 = 00h<br>W = 02 h<br>flag C = 1 ; el resultado es cero<br>flag Z = 1 ; el resultado es cero<br>c) Si antes de la instrucción. REG1 = 01 h<br>W = 02 h<br>bit C = ?<br>Al ejecutarse REG1 = 00h<br>W = FF h<br>flag C = 0 ; el resultado es negativo<br>flag Z = 1 ; el resultado es cero |

Continua

| Instrucción | SWAPF |
|---|---|
| Descripción | Los cuatro bits de más peso del registro f se intercambian con los 4 bits de menos peso del mismo registro. Si d=0 el resultado se almacena en W, si d=1 el resultado se almacena en f. |
| Operación | (f<3:0>) ---> (dest <7:4>)<br>(f<7:4>) ---> (dest <3:0>) |
| Sintaxis | [Etiqueta] SWAPF f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | SWAPF REG1,0<br>Si antes de la instrucción. REG1 = A5 h = 1010 0101 h<br>Como d=0 el resultado se almacenará en W<br>Al ejecutarse la instrucción: REG1 = A5 h = 1010 0101 b<br>W = 5A h = 0101 1010 b |
| Instrucción | XORWF |
| Descripción | Realiza la función OR-Exclusiva entre el contenido del registro W y el contenido del registro f, y almacena el resultado en f si d=1 y en W si f=0 |
| Operación | (W).XOR.(f) ---> (des) |
| Sintaxis | [Etiqueta] XORWF f,d |
| Operadores | 0 < f < 127<br>d [0,1] |
| Ejemplo | XORWF REG1,1<br>Si antes de la instrucción. REG1 = AF h = 1010 1111 b<br>W = B5 h = 1011 0101 b<br>Como d=1, el resultado se almacena en REG1<br>Al ejecutarse: REG1 = 1010 1111 Å 1011 0101 =0001 1010 = 1A h<br>W = B5 h |

INSTRUCCIONES ORIENTADAS A BIT

| Instrucción | BCF |
|---|---|
| Descripción | Pone a cero el bit número b del registro f |
| Operación | 0 --> (f<b>) |
| Sintaxis | [Etiqueta] BCF f,b |
| Operadores | 0 < f < 127<br>0 < b < 7 |
| Ejemplo | BCF FLAG_REG, 7<br>Si antes de la instrucción el registro<br>FLAG_REG = C7 h = 1100 0111 b<br>Al ejecutarse la instrucción, el registro queda con el valor:<br>FLAG_REG = 47b = 0100 0111 b |

Continua

| Instrucción | BSF |
|---|---|
| Descripción | Pone a 1 el bit b del registro f |
| Operación | 1 --> (f<b>) |
| Sintaxis | [Etiqueta] BSF f,b |
| Operadores | 0 < f < 127<br>0 < b < 7 |
| Ejemplo | BSF FLAG_REG, 7<br>Si antes de la instrucción el registro tiene el valor.<br>FLAG_REG = 0A h = 0000 1010 b<br>Al ejecutarse la instrucción, el registro queda con el valor:<br>FLAG_REG = 8A h = 1000 1010 b |
| Instrucción | BTFSC |
| Descripción | Si el bit número b del registro f es cero, la instrucción que sigue a ésta se ignora y se trata como un NOP (skip). En este caso, y sólo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse. |
| Operación | skip if (f<b>) = 0 |
| Sintaxis | [Etiqueta] BTFSC f,b |
| Operadores | 0 < f < 127<br>0 < b <7 |
| Ejemplo | AQUI BTFSC FLAG,1<br>FALSE GOTO PROCESS_CODE<br>TRUE<br>Si antes de la instrucción. PC = dirección AQUÍ<br>Al ejecutarse:<br>if FLAG<1> = 0,<br>PC = dirección TRUE<br>if FLAG<1> = 1,<br>PC = dirección FALSE |

Continua

| Instrucción | BTFSS |
|---|---|
| Descripción | Si el bit número b del registro f está a 1, la instrucción que sigue a ésta se ignora y se trata como un NOP (skip). En este caso, y sólo en este caso, la instrucción BTFSS precisa dos ciclos para ejecutarse |
| Operación | skip if (f<b>) = 1 |
| Sintaxis | [Etiqueta] BTFSS f,b |
| Operadores | 0 < f < 127<br>0 < b <7 |
| Ejemplo | HERE BTFSS FLAG,1<br>FALSE GOTO PROCESS_CODE<br>TRUE .<br>Si antes de la instrucción. PC = dirección HERE<br>Al ejecutarse:<br>if FLAG<1> = 0,<br>PC = dirección FALSE<br>if FLAG<1> = 1,<br>PC = dirección TRUE |

INSTRUCCIONES CON LITERALES Y DE CONTROL

| Instrucción | ADDLW |
|---|---|
| Descripción | Suma el contenido del registro W al literal k, y almacena el resultado en W.Si se produce acerreo el flag C se pone a "1" |
| Operación | (W) + k ----> (W) |
| Sintaxis | [Etiqueta] ADDLW k |
| Operadores | 0 < k < 255 |
| Ejemplo | ADDLW 0x15<br>Si antes de la instrucción:<br>W = 10h = 0001 0000 b<br>Al ejecutarse la instrucción<br>W = 10 h + 15 h = 25 h<br>W = 0001 0000 b + 0001 0101 b = 0010 0101 b |
| Instrucción | ANDLW |
| Descripción | Efectúa la operación AND lógico entre el contenido del registro W y el literal k, y almacena el resultado en W. |
| Operación | (W).AND. (k) ----> (W) |
| Sintaxis | [Etiqueta] ANDLW k |
| Operadores | 0 < f < 255 |
| Ejemplo | ANDLW 0x5F<br>Si antes de la instrucción. W = A3 h<br>Al ejecutarse: W = 0101 1111 b AND 1010 0011 b = 0000 0011 B = 03 h |

Continua

| Instrucción | CALL |
|---|---|
| Descripción | Salvaguarda la dirección de vuelta en la Pila y después llama a la subrutina situada en la dirección cargada en el PC.<br>El modo de cálculo de la dirección efectiva difiere según la familia PIC utilizada. También hay que posicionar PA2, PA1 y PA0 (PIC 16C5X) o el registro PCLATCH (En los demás PIC) antes de ejecutarse la instrucción. |
| Operación | 0 = k = 2047 |
| Sintaxis | [Etiqueta] CALL k |
| Operadores | (PC)+1 ---> Top of Stack<br>k ---> PC <10:0>;<br>PCLATCH (<4:3>) ---> PC (<12,11>) |
| Ejemplo | HERE CALL THERE<br>Si antes de la instrucción.<br>PC = dirección HERE<br>Al ejecutarse:<br>PC = dirección (THERE)<br>TOS = dirección (HERE +1) |
| Instrucción | CLRWDT |
| Descripción | Se borra tanto el registro WDT (Watchdog) como su preescaler. Los bits T0# y PD# del registro de estado se ponen a "1". |
| Operación | 00h --> WDT<br>1 --> T0#<br>1 --> PD# |
| Sintaxis | [Etiqueta] CLRWDT |
| Operadores | No tiene |
| Ejemplo | CLRWDT<br>Si antes de ejecutarse la instrucción<br>WDT = ?<br>Al ejecutarse: WDT = 00 h<br>Preescaler WDT = 0<br>bit de estado T0 = 1<br>bit de estado PD = 1 |
| Instrucción | GOTO |
| Descripción | Salto incondicional, normalmente se utiliza para llamar a la subrutina situada en la dirección que se carga en PC.<br>El modo de cálculo de la instrucción caga de bit 0 al 10 de la constante k en el PC y los bits 3 y 4 del registro PCLATH en los 11 y 12 del PC |
| Operación | K --> PC <10:0><br>(PCLATH <4:3>) ---> (PC <12:11>) |
| Sintaxis | [Etiqueta] GOTO k |
| Operadores | 0 < k < 2047 |
| Ejemplo | GOTO THERE<br>Al ejecutarse: PC = dirección THERE |

Continua

| | |
|---|---|
| Instrucción | IORLW |
| Descripción | Se realiza la operación lógica OR entre el registro W y el literal k. El resultado se almacena en el registro W. |
| Operación | (W).OR.k ---> (W) |
| Sintaxis | [Etiqueta] IORLW k |
| Operadores | 0 < k < 255 |
| Ejemplo | IORLW 0x35<br>Si antes de la instrucción. W = 9A h<br>Al ejecutarse: W = 1001 1010 b + 0011 0101 b = 1011 1111 b = BF h |
| Instrucción | MOVLW |
| Descripción | El registro W se carga con el valor de 8 bits del literal k |
| Operación | k --> (W) |
| Sintaxis | [Etiqueta] MOVLW k |
| Operadores | 0 < f < 255 |
| Ejemplo | MOVLW 0x5A<br>Al ejecutarse: W = 5A h |
| Instrucción | RETFIE |
| Descripción | Carga el PC con el valor que se encuentra en la parte alta de la Pila, asegurando así la vuelta de la interrupción. Pone a 1 el bit GIE, con el fin de autorizar de nuevo que se tengan en cuenta las interrupciones. |
| Operación | TOS --> PC<br>1 --> GIE |
| Sintaxis | [Etiqueta] RETFIE |
| Operadores | No tiene |
| Ejemplo | RETFIE<br>Al ejecutarse: PC = TOS<br>GIE = 1 |

Continua

| | |
|---|---|
| Instrucción | RETLW |
| Descripción | Carga el registro W con el literal k, y después carga el PC con el valor que se encuentra en la parte superior de la PILA, efectuando así un retorno de subrutina. |
| Operación | k --> (W)<br>TOS ---> PC |
| Sintaxis | [Etiqueta] RETLW k |
| Operadores | 0 < K < 255 |
| Ejemplo | CALL TABLA; W contiene tabla<br>el valor offset<br>W nuevo valor de tabla<br>TABLA: ADDWF PC; W = offset<br>RETLW k1; Nueva Tabla<br>RETLW k2<br>RETLW kn;<br>Fin de tabla<br>Antes de ejecutarse la instrucción W = 07 h<br>Al ejecutarse la instrucción W = Toma el valor de k7 |
| Instrucción | RETURN |
| Descripción | Carga el PC con el valor que se encuentra en la parte superior de la PILA, efectuando así un retorno de subrutina |
| Operación | TOS ---> PC |
| Sintaxis | [Etiqueta] RETURN |
| Operadores | No tiene |
| Ejemplo | |
| Instrucción | SLEEP |
| Descripción | Pone al circuito en modo Sleep (bajo consumo) con parada del oscilador. Pone a 0 el flag PD# (Power Down) y el flag TO# (Timer Out) se pone a 1. Se puede salir de este estado por:<br>1. Activación de MCLR para provocar un Reset<br>2. Desbordamiento del Watchdog si quedó operativo en el modo reposo<br>3. Generación de una interrupción que no sea TMR0 ya que ésta se desactiva con la instrucción SLEEP. |
| Operación | 00h ---> WDT<br>1 ---> TO#<br>0 --> PD# |
| Sintaxis | [Etiqueta] SLEEP |
| Operadores | No tiene |
| Ejemplo | SLEEP |

Continua

| Instrucción | SUBLW |
|---|---|
| Descripción | Resta en complemento a dos del contenido del literal k el contenido del registro W, y almacena el resultado en W. |
| Operación | k - (W) ---> (W) |
| Sintaxis | [Etiqueta] SUBLW k |
| Operadores | 0 < k < 255 |
| Ejemplo | SUBLW 0x 02<br>Si antes de la instrucción. W = 01 h<br>flag C = ?<br>Al ejecutarse: W = 01<br>flag C = 1 ; el resultado es positivo<br>Si antes de la instrucción. W = 02 h<br>flag C = ?<br>flag Z = ?<br>Al ejecutarse: W = 00 h<br>flag C = 1 ; el resultado es cero<br>flag Z = 1<br>Si antes de la instrucción. W = 03 h<br>flag C = ?<br>Al ejecutarse: W = FF h<br>flag C = 0 ; el resultado es negativo |
| Instrucción | XORLW |
| Descripción | Realiza la función OR-Exclusiva entre el contenido del registro W y la constante k de 8 bits. El resultado se almacena en W |
| Operación | (W).XOR.k ---> (W) |
| Sintaxis | [Etiqueta] XORLW k |
| Operadores | 0 < f < 255 |
| Ejemplo | XORLW 0xAF<br>Si antes de la instrucción. W = 1011 0101 b = B5 h<br>Al ejecutarse la instrucción: W = 1011 0101 b Å 1010 1111 b = 0001 1010 b = 1A h |

# Bibliografía


Microprocesador Z-80 programación e interfaces.
Joseph c. Nichols. Elizabet A. Peter R. Rony
Ed. publicaciones marcambo, S.A 1979

Microcontroladores PIC
Diseño práctico de aplicaciones.
José Ma. Angulo Usategui.
Ignacio Angulo Martínez.
Ed. Mc Graw Hill. 2° edicion

Manual de microchip technology 1996
PIC 16F84

Manual del fabricante Zilog
Microprocesador Z-80


REFERENCIAS


Stepper motors, de Ian Harris
http://www.doc.ic.ac.uk/~ih/doc/stepper/
http://www.cs.uiowa.edu/~jones/step/
http://www.cs.uiowa.edu/~jones/step/#introduction
http://www.eio.com/jasstep.htm
http://www.eio.com/jasstep.htm#characteristics