UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES "ARAGÓN"

DISEÑO Y CONSTRUCCIÓN DE UN PROGRAMADOR PARA LABORATORIO A BASE DE UN MICROCONTROLADOR

T E S I S

QUE PARA OBTENER EL TITULO DE INGENIERO MECANICO ELECTRICISTA (AREA ELECTRONICA EN EL MODULO DE COMUNICACIONES)

P R E S E N T A:
EDGAR ALFREDO GONZALEZ GALINDO

ASESOR: ING. JOEL LOPEZ CONTRERAS

Agradecimiento

Si pudiera expresar la verdadera razón de existir haría todo por que el estuviera con migo, pero me ha dado a una mujer muy bella que amo tanto, eres la mujer que ha dado todo por tener a un ser como yo, ahora espero darte y brindarte todo de mi.

No existe el mejor agradecimiento que tu presencia en donde quiera que estés y en donde quiera que me veas ahí estaré con tigo. Tu presencia me fortaleció tu mi mejor amiga, tu mi mejor compañera que de los mejores consejos y de la mejor compañía hoy ya no estas, pero te llevo en mi en mi mente y en mi corazón

Quisiera escribirte los mejores agradecimiento pero lo mas importante es que solo te tengo a ti y daría por que mi otra madre estuviera con migo, hoy se que adonde quiera que este, esta orgullosamente de mi, por que me vio en vida y vio el sacrificio que tu madre has hecho por tu hijo.

Dedicada a Gudelia López Tino

Que descanses en paz

Índice

Introducción

Los microcontroladores han revolucionado en la parte de la electrónica además son sencillos de programar y mantiene una función especifica y se encuentra en gran parte de los equipos de electrónica que cotidianamente hacemos uso de ellos, Se pueden encontrar controlando el funcionamiento de los ratones y teclados de los computadoras, en los teléfonos, en los hornos “microondas” y los televisores de nuestro hogar por mencionar algunos.

En la red mundial se ha dado a conocer información sobre proyectos que día con día se realizan en diversas Universidades del mundo, la revolución de los microcontroladores es ya una necesidad ya que tiene varias aplicaciones y que empieza a cubrir en todas las áreas de la Ciencias.

En los siglos venideros serás testigos sobre la expansión de las diminutas computadoras que gobernaran gran parte de los equipos electrónicos que utilizamos en el hogar. La tecnología ya comenzó a tener una dimensión mas pequeña y la cual comenzara a tener una ventaja de espacio, sobre la revolución de esta nueva manera de usar tecnología.

CAPITULO 1

“ESTRUCTURA DE LOS MICROCONTROLADORES”

1.1 Antecedentes

Inicialmente cuando no existían los microprocesadores las personas se ingeniaban en diseñar sus circuitos electrónicos y los resultados estaban expresados en diseños que implicaban muchos componentes electrónicos y cálculos matemáticos. Un circuito lógico básico requería de muchos elementos electrónicos basados en transistores, resistencias, etc, lo cual desembocaba en circuitos con muchos ajustes y fallos; pero en el año 1971 apareció el primer microprocesador el cual originó un cambio decisivo en las técnicas de diseño de la mayoría de los equipos. Al principio se creía que el manejo de un microprocesador era para aquellas personas con un coeficiente intelectual muy alto; por lo contrario con la aparición de este circuito integrado todo sería mucho más fácil de entender y los diseños electrónicos serian mucho más pequeños y simplificados. Entre los microprocesadores mas conocidos tenemos el popular Z-80 y el 8085. Los diseñadores de equipos electrónicos ahora tenían equipos que podían realizar mayor cantidad de tareas en menos tiempo y su tamaño se redujo considerablemente; sin embargo, después de cierto tiempo aparece una nueva tecnología llamada microcontrolador que simplifica aun mas el diseño electrónico.

En 1965 GI formó una división de microelectrónica, destinada a generar las primeras arquitecturas viables de memoria EPROM y EEPROM. De forma complementaria GI Microelectronics División fue también responsable de desarrollar una amplia variedad de funciones digitales y analógicas en las familias AY3-xxxx y AY5-xxxx.

GI también creó un microprocesador de 16 bit, denominado CP1600, a principios de los 70. Este fue un microprocesador razonable, pero no particularmente bueno manejando puertos de e/s. Para algunas aplicaciones muy específicas GI diseñó un Controlador de Interfase Periférico (PIC) entorno a 1975. Fue diseñado para ser muy rápido, además de ser un controlador de e/s para una máquina de 16 bits pero sin necesitar una gran cantidad de funcionalidades, por lo que su lista de instrucciones fue pequeña.

No es de extrañar que la estructura diseñada en 1975 es, sustancialmente, la arquitectura del actual PIC16C5X. Además, la versión de 1975 fue fabricada con tecnología NMOS y sólo estaba disponible en versiones de ROM de máscara, pero seguía siendo un buen pequeño microcontrolador. El mercado, no obstante, no pensó así y el PIC quedó reducido a ser empleado por grandes fabricantes únicamente.

1.2 Definición de un Microcontrolador

Un Microcontrolador es un circuito integrado de muy alta escala de integración el cual contiene tres unidades básicas que lo identifican como tal y son *CPU* para procesar la información, *Memoria de datos* para guardar información y *Memoria de Programa* para almacenar las instrucciones.

Durante los 80, GI renovó su apariencia y se reestructuró, centrando su trabajo en sus principales actividades, semiconductores de potencia esencialmente, lo cual siguen haciendo actualmente con bastante éxito. GI Microelectrónica División cambió a GI Microelectrónica Inc (una especie de subsidiaria), la cual fue finalmente vendida en 1985 a Venture Capital Investors, incluyendo la fábrica en Chandler, Arizona. La gente de Ventura realizó una profunda revisión de los productos en la compañía, desechando la mayoría de los componentes AY3, AY5 y otra serie de cosas, dejando sólo el negocio de los PIC y de las memorias EEPROM y EPROM. Se tomó la decisión de comenzar una nueva compañía, denominada Arizona Microchip Technology, tomando como elemento diferenciador sus controladores integrados.

Como parte de esta estrategia, la familia NMOS PIC165X fue rediseñada para emplear algo que la misma compañía fabricaba bastante bien, memoria EPROM. De esta forma nación el concepto de basarse en tecnología CMOS, OTP y memoria de programación EPROM, naciendo la familia PIC16C5X.

Actualmente Microchip ha realizado un gran número de mejoras a la arquitectura original, adaptándola a las actuales tecnologías y al bajo costo de los semiconductores.

Un Microcontrolador es un circuito integrado o mas comúnmente llamado chip, que cumple las funciones de cerebro de cualquier aplicación, que puede ser desde encender un led hasta telecontrol y es responsable de la buena funcionalidad del circuito que gobierna. Como todo cerebro, este chip tiene que procesar alguna información que tiene en su memoria y de esta maneta decidir que hacer. A esta información que debe tener el chip se le llama software o programa de aplicación. Es responsabilidad nuestra enviar la adecuada información este chip para que trabaje bien en su funcionalidad.

Puede ser visto externamente como un circuito integrado TTL o CMOS normal, pero internamente dispone de todos los dispositivos típicos de un sistema microprocesador.

1.3 Diagrama de Bloques de un Microcontrolador

Para entender la configuración básica del Microcontrolador, describiremos una solicitud del CPU de un dato localizado en Memoria: El CPU solicita información a la memoria (o un Puerto) por un llamado con la dirección correspondiente a la localidad donde se encuentre el dato solicitado. La dirección (de la información solicitada) es almacenada en el CPU como un número binario en un registro temporal. Las salidas de este registro son mandadas por muchas vías (o una vía sencilla) a la memoria del Microcontrolador y periféricos. Al grupo de vías de comunicación que comparten una trayectoria común en forma paralela se le denomina “BUS”.

Es entonces cuando el registro de dirección almacena el dato recibido. El número de bits recibidos dependen del tipo de Microcontrolador. El dato o la información buscada es mandada al CPU por el bus de datos. El bus de datos es diferente al bus de direcciones ya que el bus de datos sirve únicamente para recibir o mandar datos a memoria o periféricos.

Las señales del bus de direcciones son controladas solamente por el CPU y la información va siempre del CPU a los bloques de memoria. Por otro lado, la información en el bus de datos puede ser de entrada o salida al CPU por medio del registro de datos.

En otras palabras, el bus de datos es bi-direccional y el bus de direcciones es unidireccional. El ancho de los buses de datos y de direcciones también pueden ser diferentes, dependiendo del tipo de microcontrolador y del tamaño de la memoria. A continuación se muestra a bloques el CPU, Memorias y Buses.

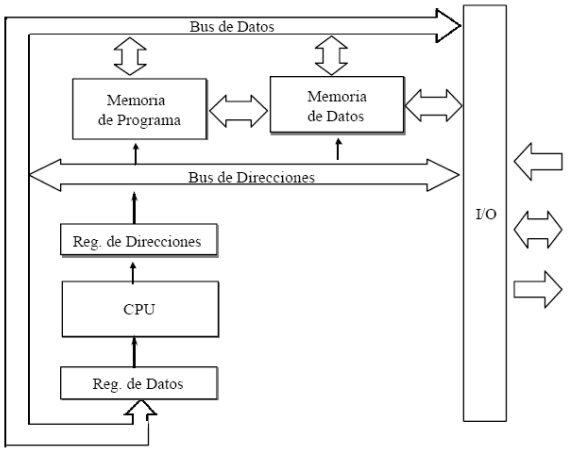


Figura 1. Diagrama a Bloques de un Microcontrolador

CPU

La Unidad de Procesamiento Central es el corazón del Microcontrolador y es aquí donde todas las operaciones aritméticas y lógicas son realizadas. Es decir, es la unidad que calcula todas las operaciones que son ordenadas por la memoria de programa.

Memoria de Programa

Contiene las instrucciones organizadas en una secuencia particular para realizar una tarea. Típicamente es denominada memoria de sólo lectura (ROM) o también OTP, EPROM o FLASH que son memorias que una vez programadas almacenan la información aunque el sistema no sea energizado. Esto permite que el Microcontrolador ejecute el programa almacenado en Memoria inmediatamente después de ser energizado.

La Memoria de Programa contiene el programa del Microcontrolador. Es decir es aquí donde se almacenan los comandos a ejecutar por el CPU para realizar una tarea de control determinados por el usuario dependiendo de los requerimientos de la aplicación. Existen muchos tipos de memoria de programa ROM, EPROM, OTP, EEPROM, FLASH.

Memoria de Datos

Esta es una memoria que puede ser escrita y leída según sea requerido por el programa. Tiene las funciones de almacenamiento de datos (pila) y como almacenamiento de variables. Este tipo de memoria es usualmente llamada memoria RAM (Memoria de Acceso Aleatorio). Cada localidad de memoria tiene una dirección única con la cual el CPU encuentra la información necesaria. Los microcontroladores actuales contienen ambas memorias (Datos y Programa) incluidas dentro del circuito integrado. Por otro lado, resulta necesario contar con otras unidades que hacen posible el funcionamiento mínimo de un Microcontrolador que son Circuitería de Temporización y Entradas/Salidas

Circuitería de Temporización

Los Microcontroladores usan señales de temporización llamadas Reloj que proveen una referencia en el tiempo para la ejecución del programa. Esta señal determina en qué momento los datos deben ser escritos o leídos de la memoria. Así mismo, provee la sincronía con los dispositivos conectados al Microcontrolador (Periféricos).

Entradas y Salidas

Los Microcontroladores requieren de una interface para comunicarse con la circuitería externa. Esta Interface es denominada comúnmente como Puerto. Existen Puertos de Entrada y Salida los cuales permiten que las señales (o datos) sean leídos del exterior o mandados al exterior del Microcontrolador. Los Puertos están formados de pines, (terminales del circuito integrado) los cuales, dependiendo de la aplicación, son conectados a un sin fin de dispositivos como teclados, interruptores, censores, relevadores, motores, etc.

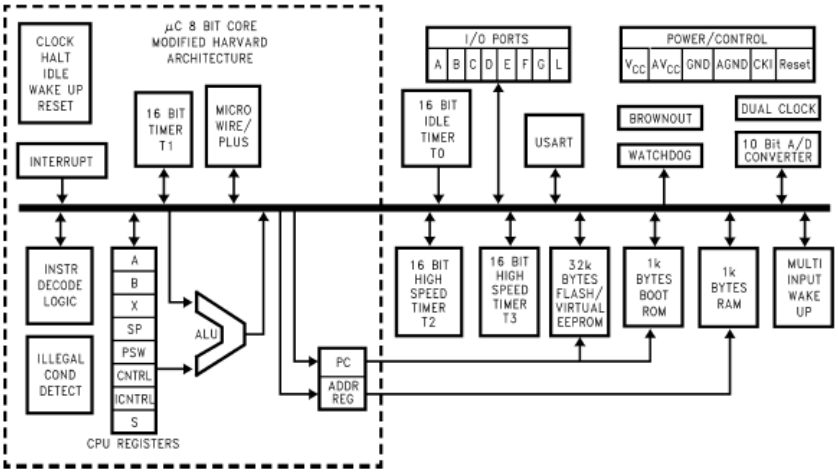


Fig. 3 Diagrama de bloques mas especifico de un COP8

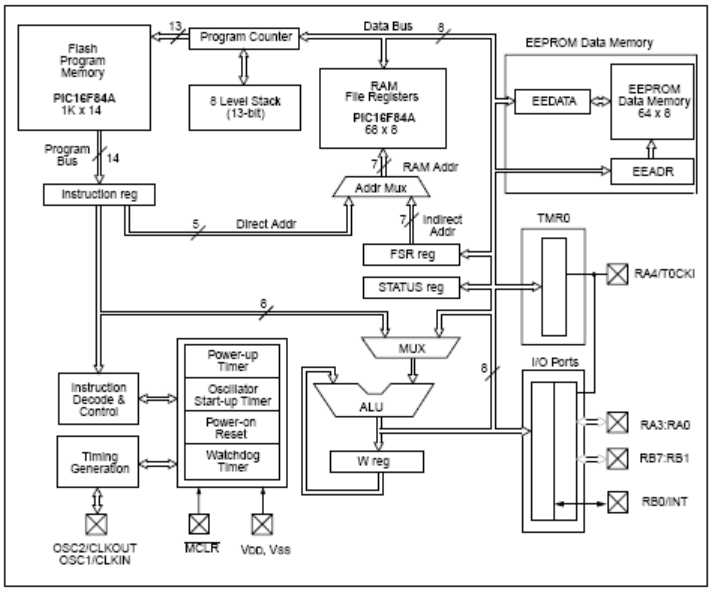


Fig. 3 Diagrama de bloques mas especifico de un PIC16F8X

1.3.1 Arquitectura

Arquitectura Harvard

La Arquitectura Harvard fue desarrollada en Harvard por Howard Aiken, otro pionero en las computadoras. Esta arquitectura se caracteriza por tener buses separados para la memoria de Programa y la memoria de Datos. Una de las ventajas de la arquitectura Harvard es que la operación del Microcontrolador puede ser controlada más fácilmente si se presentara una anomalía en el apuntador de programa. Existe otra arquitectura que permite accesos a tablas de datos desde la memoria de programa. Esta arquitectura es llamada Arquitectura Harvard Modificada.

Esta última arquitectura es la dominante en los microcontroladores actuales ya que la memoria de programa es usualmente ROM, OTP, EPROM o FLASH mientras que la memoria de datos es usualmente RAM.

Consecuentemente, las tablas de datos pueden estar en la memoria de programa sin que sean perdidas cada vez que el sistema es apagado. Otra ventaja importante en la arquitectura Harvard Modificada es que las transferencias de datos pueden ser traslapadas con los ciclos de decodificación de instrucciones. Esto quiere decir que la siguiente instrucción puede ser cargada de la memoria de programa mientras se está ejecutando una instrucción interviniendo la memoria de datos. La desventaja de la arquitectura Harvard Modificada podría ser que se requieren instrucciones especiales para acceder valores en RAM y ROM haciendo la programación un poco complicada.

1.3.2 Set de Instrucciones

Conjunto De Instrucciones

En la aplicación de los microcontroladores de 8 bit de hoy/ el rendimiento, la flexibilidad y el tiempo de comercializar son varios del los asuntos de tecla hacia los que los diseñadores de sistema miran intentar desarrollarse productos well- crear por ingeniería que compiten en el mercado. Muchos de estos asuntos pueden ser a través del direccionamiento de la manera en cuál maneja el procesamiento el conjunto de instrucciones de un microcontrolador que ejecuta tareas. Y ése es por qué la familia de COP8 se ofrece uno único de instrucción clave - eficiente fijada - uno que provee la flexibilidad, la funcionalidad los gastos reducidos y el tiempo más rápido de comercializar, y hoy día productos microcontrolador basados en estas características requieren la eficiencia de clave es importante porque permite que a diseñadores sea mas accesible para su diseño de desarrollo de este chip y para desarrollar programas en la memoria del microcontrolador espacio (el ROM, el OTP o el flash). Seleccionar un microcontrolador con menos el tamaño de memoria de programa traduce al sistema más bajo de gastos, y la seguridad adicional de saber que más clave puedan reducir espacio de memoria de programa disponible.

Esta sección define el conjunto de instrucciones de la familia de COP8 miembros. Contiene la información sobre el conjunto de instrucciones especiales, abordando los modos y los tipos.

Las características de instrucción la fuerza del conjunto de instrucciones está basada en lo siguiente Características:

• Principalmente las instrucciones de opcode solo - byte minimizan el programa tamaño.

• Un ciclo de instrucción para la memoria de instrucciones solo – byte para minimizar el tiempo de ejecución de programa.

• Muchas instrucciones de función solo - byte y múltiples como DRSZ.

• Tres memoria trazar un mapa de punteros: dos para el registro indirecto Se dirigir, y uno para el software se apilan.

• Dieciséis recuerdo correlacionó registros que lo permitían uno optimizar puesta en práctica de ciertas instrucciones.

• La habilidad de establecer, volver a poner, y evaluar cualquier bit individual en los datos el espacio de direccionamiento de memoria, incluyendo al memoria trazar un mapa puertos de E/S y registros.

• Registro - Indirecto cargue y cambie instrucciones con automático mensaje - incrementar opcional o disminuir del puntero de registro. Esto permite la eficiencia más grande (Tanto en la clave de tiempo de ciclo y programa) en la carga, Caminar a través y procesar campos en la memoria de datos.

• Las instrucciones únicas de optimizar el tamaño de programa y Eficiencia de caudal de proceso y transferencia. Algunas de estas instrucciones lo son: DRSZ, IFBNE, DCOR, RETSK, VIS y RRC.

Operando Que Direccionar Modos

El operando de una instrucción especifica qué ubicación de memoria es, ser afectado por esa instrucción. varios y diferentes son los modos operando direccionar están disponibles, admitiendo la memoria la ubicaciones ser especificado en una variante de maneras. Una instrucción puede especificar una dirección directamente proporcionando al específico dirección, o indirectamente especificando un puntero de registro. El Contenido del registro (o en algunas cajas, dos registros) señale a la ubicación de memoria deseada. En el inmediato el modo, el byte de datos de ser usado es contenido en la instrucción sí. Direccionar el modo EACH tiene sus propias ventajas y desventajas con respecto a la flexibilidad, la velocidad de ejecución, y compactibilidad de programa. No todos modos están disponibles con todos instrucciones. La instrucción de carga (LD) brinda el más grande número de direccionar modos. El disponible abordar los modos lo es:

• Dirija
• B de registro o Indirectamente de X
• B de registro o X desviado con Post – Incrementar/Disminuir
• Inmediato
• Cortocircuito inmediato
• Indirecto de la memoria de programa

El direccionar modos es descrito abajo. Cada descripción incluye un ejemplo de una instrucción de lenguaje de ensamblaje usar el describir abordando el modo. Dirija. La dirección de memoria es especificada directamente como un byte hacia dentro la instrucción.

En el lenguaje de ensamblaje, la dirección directa lo es escribir como un valor numérico (o una etiqueta que ha sido definida otra parte en el programa como un valor numérico).

Ejemplo: cargue directamente de memoria del Acumulador

LD A, 05

Reg/Data Memory	Contents Before	Contents After
Accumulator	XX Hex	A6 Hex
Memory Location 0005 Hex	A6 Hex	A6 Hex

Registre B o indirecto de X. La dirección de memoria es especificada por los contenido del registro de B o el registro de X (puntero registro). En el lenguajes de ensamblaje, la nota [B] o [X] especifican qué registro servir de la idea. Ejemplo: memoria de intercambio con acumulador, B indirecto

X A, [B]

Reg/Data Memory	Contents Before	Contents After
Accumulator	01 Hex	87 Hex
Memory Location 0005 Hex	87 Hex	01 Hex
B Pointer	05 Hex	05 Hex

B de registro o X desviado con Post - Incrementar/Disminuir. La dirección de memoria relevante es especificada por los contenido del registro de B o el registro de X (puntero Registro). El registro de puntero está automáticamente incrementado o disminuido después de la ejecución, admitir la manipulación fácil De bloques de memoria con bucles de software. En el lenguaje de ensamblaje, La nota [B+], [B-], [X +], o [X -] especifican cuál registre los saques como el puntero, y si el puntero lo incrementado o disminuido.

Ejemplo: memoria de intercambio con Acumulador, B indirecto Con Post – incrementa X A, [B+]

X A,[B+]

Reg/Data Memory	Contents Before	Contents After
Accumulator	03 Hex	62 Hex
Memory Location 0005 Hex	62 Hex	03 Hex
B Pointer	05 Hex	06 Hex

*M*edio. Los datos para la operación siguen la instrucción opcode en la memoria de programa. en el lenguaje de ensamblador, la calidad de señal de número (#) demuestra un operando inmediato.

Ejemplo: cargue Acumulador inmediato
LD A,#05
Reg/Data Contents Contents
Memory Before After
Acumulador XX Hex 05 Hex

*C*ortocircuito inmediato. Ésta es una caja especial de uno inmediato instrucción. En la instrucción de "Cargue B inmediato", el 4 – bit el valor inmediato en la instrucción es cargado en el más bajo mordisco del registro de B. el mordisco superior del registro de B lo es volver a poner a 0000 archivo binario

Ejemplo: cargue registro inmediatamente de B
LD B,#7
Reg/Data Contents Contents
Memory Before After
B Pointer 12 Hex 07 Hex

*I*ndirecto de la memoria de programa. Ésta es una caja especial de Una instrucción indirecta que admite el acceso para tablas de datos Guardar en la memoria de programa. en el "acumulador de carga indirecto" (Colocar) la instrucción, los el superior y bytes más bajos del Programar contador como el que (PCU y el PCL) es usado temporalmente uno Puntero para la memoria de programa. Para los propósitos de acceder al programa La memoria, los contenido del acumulador y el PCL lo son Cambiar. Los datos apuntados por el contador de programa lo son Intencionado en el acumulador, y simultáneamente, el original contenido del PCL es restaurado con el propósito de que el programa puede reanudarse Ejecución normal.

Ejemplo: Load Accumulator Indirect
LAID
Reg/Data Contents Contents
Memory Before After
PCU 04 Hex 04 Hex
PCL 35 Hex 36 Hex
Accumulator 1F Hex 25 Hex
Memory Location
041F Hex 25 Hex 25 Hex

Transferencia De Control Direccionar Modos

Las instrucciones de programa son ejecutadas en la orden secuencial generalmente. Sin embargo, las instrucciones de salto pueden ser use cambiar el Secuencia de ejecución normal. Varios transferencia - de - control abordar Los modos son disponer especificar direcciones de salto. Un cambio en la circulación de programa requiere uno non-incrementar
Cambio en los contenido de mostrador de programa. El programa Constó de dos bytes en sentido contrario, designar el byte superior (PCU) y byte más bajo (el PCL). El bit más significativo de PCU No es usado, dejando que 15 bits aborden la memoria de programa. Diferente direccionar modos es use especificar al nuevo Dirección para el mostrador de programa. La elección de se dirigir el modo depende de la distancia del salto principalmente. Más lejos Los saltos requieren más bytes de instrucción en orden a veces Especifique los nuevos contenido de mostrador de programa totalmente. Los modos transferencia - de - control abordar disponibles lo son:

- Salto Relativo
- Salto Absoluto
- Salto Absoluto Largo
- Salto indirecto

Los modos transferencia - de - control abordar son descritos abajo. Cada descripción incluye un ejemplo de una instrucción de salto Usar un detalle abordando el modo, y el efecto sobre los bytes de mostrador de programa de ejecutar esa instrucción. pariente de salto. En esta instrucción 1 byte, seis bits del opcode de instrucción especifica la distancia del salto del Ubicación de memoria de programa en curso. La distancia del salto Puede extenderse - 31 a +32. Un JP + 1 instrucción no es admitido. El programador debe usar uno NOP en vez.

JP 0A

Reg Contents	Contents Before	After
PCU	02 Hex	02 Hex
PCL	05 Hex	0F Hex

Salto Absoluta. En esta instrucción 2 byte, 12 bits del Opcode de instrucción especifica los nuevos contenido del programa Contesté. Las tres partes superiores del mostrador de programa quedan Igual, restringir la nueva dirección de mostrador de programa Al mismo espacio de direccionamiento de 4 Kbyte como la corriente Instrucción. (Esta restricción es relevante solamente en dispositivos usar Más de un espacio de memoria de programa de 4 Kbyte.)

Ejemplo: Jump Absolute
JMP 0125

Reg Contents	Before	After
PCU	0C Hex	01 Hex
PCL	77 Hex	25 Hex

Salto Absoluto Largo. En esta instrucción 3 byte, 15 bits de El opcode de instrucción especifica los nuevos contenido del contador del programa.

Example: Jump Absolute Long
JMP 03625

Reg/ Contents Memory	Before	After
PCU	42 Hex	36 Hex
PCL	36 Hex	25 Hex

Salto indirecto. En esta instrucción 1 byte, el byte más bajo de La dirección de salto es obtenida de una mesa guardada en el programa La memoria, con el posición del acumulador como el byte de orden bajo De un puntero en la memoria de programa. Para los propósitos de acceder Programe la memoria, los contenido del acumulador lo es Escribir al PCL (temporalmente). Los datos apuntados by el (PCH / PCL) de mostrador de programa es cargado en el PCL mientras PCH Se queda igual.

Ejemplo: Jump Indirect
JID

Reg/ Contents Memory	Before	After
PCU	01 Hex	01 Hex
PCL	C4 Hex	32 Hex
Accumulator	26 Hex	26 Hex
Memory Location 0126 Hex	32 Hex	32 Hex

La instrucción de VIS es una caja especial de la transferencia indirecta Del Control que direccional el modo, cuando el vector doble – byte Relacionar con la interrupción ser trasladado de adyacente Las direcciones en la memoria de programa respecto a el programa contestan hacia dentro Ordene para sacar precipitadamente la rutina del servicio de interrupción asociada.

Tipos De Instrucción

El conjunto de instrucciones contiene una gran variedad de instrucciones. El disponible debajo del que las instrucciones son puestas en una lista, organizar en Grupos relacionados. Algunas instrucciones evalúan una afección y pasan por alto la próxima instrucción Si la condición no es verdadera. Las instrucciones pasado por alto lo son Ejecutar como las instrucciones de no - operación (NOP).

Instrucciones De Aritmética

Las instrucciones de aritmética llevan a cabo aritmética binaria como Adición y resta, teniendo o prescindir el bit de acarreo.

Add (ADD)
Add with Carry (ADC)
Subtract with Carry (SUBC)
Increment (INC)
Decrement (DEC)
Decimal Correct (DCOR)
Clear Accumulator (CLR)
Set Carry (SC)
Reset Carry (RC)

Instrucciones De Transferencia De Control

Las instrucciones de transferencia de control cambian el habitual secuencial Circulación de programa modificando los contenido del programa Contesté. El salto para Subrutina que las instrucciones guardan p presiden del contenido de mostrador de programa sobre la pila antes de saltar; el Intro en el que las instrucciones abren rápidamente the top of la pila back el Mostrador de programa.

Jump Relative (JP)
Jump Absolute (JMP)
Jump Absolute Long (JMPL)
Jump Indirect (JID)
Jump to Subroutine (JSR)
Jump to Subroutine Long (JSRL)
Jump to Boot ROM Subroutine (JSRB)
Return from Subroutine (RET)
Return from Subroutine and Skip (RETSK)
Return from Interrupt (RETI)
Software Trap Interrupt (INTR)
Vector Interrupt Select (VIS)

Carga E Instrucciones De Intercambio

Las instrucciones de carga y lo intercambio escriben valores de byte registros o memoria. El se dirigir que el modo condiciona el origen de los datos.

Cargar (LD)
Cargar Acumulador indirecto (LAID)
El intercambio (X)

Las Instrucciones Lógicas

Las instrucciones lógicas efectúan el AND de operaciones, OR, y haga OR exclusivo de (el OR exclusivo). Las otras operaciones lógicas pueden serlo funcionó combinando estas operaciones básicas. Por ejemplo, complementar es logrado por exclusiva – OR El Acumulador con Hex de FF.

AND lógico (AND)
OR lógico (OR)
El OR exclusivo (XOR)

Las instrucciones de manipulación de bit de Acumulador las instrucciones de manipulación de bit de Acumulador admiten el usuario Cambiar las partes de Acumulador y intercambiar sus dos mordiscos.

Rotar en el alcance (RRC)
Girar a la izquierda a través del alcance (RLC)
Intercambie los mordiscos de Acumulador (SWAP)

Instrucciones De El Control De Pila

Empujar los datos en la pila (empujar)
Abrir rápidamente los datos fuera de la pila (la música pop)

Instrucciones De Las De Manipulación De Morder De Memoria

Las instrucciones de manipulación de bit de memoria permiten que el usuario lo establezca y vuelva a poner bits individuales en la memoria.
Poner mordió (SBIT)
Volver a quien un poner mordió (RBIT)
Volver a quien un hasta de poner mordió (RPND)

Condicionales De Instrucciones

La instrucción condicional evalúa una afección. Si la condición lo es verdadero, la próxima instrucción es ejecutada en la manera normal si la condición es falsa, la próxima instrucción es pasada por alto.

If Equal (IFEQ)
If Not Equal (IFNE)
If Greater Than (IFGT)
If Carry (IFC)
If Not Carry (IFNC)
If Bit (IFBIT)
If B Pointer Not Equal (IFBNE)
And Skip if Zero (ANDSZ)
Decrement Register and Skip if Zero (DRSZ)

Instrucción De No Operación

Instrucción de carril elevado de no - no nada de hace, habitar de hacerlo / serlo de para de operación espacio en la memoria de programa y la tiempo en la ejecución. La no - operación (NOP)

Nota: caja de una de es de VIS de carril elevado especial de la indirecta de transferencia de modo de carril elevado de direccionar de Control de carril elevado, cuando el vector de carril elevado de byte de doble Relacionar con la interrupción ser trasladado de adyacente Direcciones en la memoria de programa respecto a mostrador de carril elevado de programa
(PC) servicio de carril elevado de saltar de para de conectado de interrupción Rutina.

Definición de símbolo de y de registro

Para seguir la nomenclatura de representan de abreviaturas

Usado en la descripción de el ensamblador COP8 y de instrucción como se muestran a continuación

	Registers
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
S	8-Bit Segment Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt
	Enable
VU	Interrupt Vector Upper Byte
VL I	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data

Symbols	
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

Instrucción De Tiempo De Ejecución

La mayoría de las instrucciones son el byte solas (con inmediato se dirigir Instrucciones de modo tomar dos bytes). La mayoría de las instrucciones de byte solas tardan un tiempo de ciclo ejecutar. Las instrucciones pasado por alto requieren que la cantidad de x de los ciclos lo sea Pasar por alto, donde x es igual al número de bytes en el pasar por alto Opcode de instrucción. Ver los bytes y los ciclos por la tabla de instrucción por Detalles.

Bytes Y Ciclos Por La Instrucción

Lo siguiente para el que la tabla indica el número de bytes y ciclos Cada instrucción en el formato del byte / ciclo.

Arithmetic and Logic Instructions

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

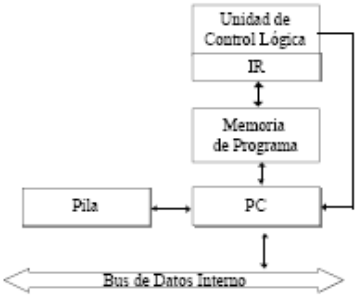
Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JSRB	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

1.4 Memoria de Programa

La memoria de programa consta de 32,768 byte de flash la memoria. Estos bytes pueden sujetar instrucciones de programa o constante los datos (tablas de datos para la instrucción de los vectores de salto y para la instrucción de JID, y los vectores de interrupción para el VIS la instrucción). La memoria de programa es direccionado por el bit 15 de la (PC) de mostrador de programa. Interrumpe todo el vector de dispositivo la dirección hex de 00FF de ubicación de memoria de programa. la memoria de programa lee hex de 00 en el estado borrado. La ejecución de programa empieza en 0 de ubicación después de RESET. Si una instrucción de INTRO es ejecutada cuando el SP contiene 6F (Hexadecimal), la instrucción que la ejecución continuará de programa 7FFF de ubicación de memoria (hexadecimal). Si la ubicación por la que 7FFF es accedido una extracción de instrucción, el destello del que la memoria devolverá un valor 00. Esto es el opcode para el INTR que instrucción y testamento causan una trampa de software. para el propósito de borrar y rescribir la memoria flash, es organizado en páginas de 128 bytes.

La Memoria de Programa contiene el programa del Microcontrolador. Es decir es aquí donde se almacenan los comandos a ejecutar por el CPU para realizar una tarea de control determinados por el usuario dependiendo de los requerimientos de la aplicación. Existen muchos tipos de memoria de programa ROM, EPROM, OTP, EEPROM, FLASH.



Existe un registro que se identifica como PC (Program Counter - registro de programa)) o Contador de Programa. El PC es requerido por el CPU para apuntar a la localidad de memoria que contiene las instrucciones del programa.

Cada vez que un Opcode (Representación binaria de una instrucción) es leído de la memoria, el registro PC es incrementado en uno para apuntar a la siguiente instrucción a ser ejecutada. Cada vez que el CPU realiza una operación de decodificación de instrucción el registro PC es incrementado en uno, por lo tanto, el registro PC siempre apunta a la siguiente instrucción por ser ejecutada. La longitud del registro PC es determinante en el tamaño de la Memoria de Programa. Ya que no se podría apuntar con el registro PC de 8 bits una Memoria de Programa de 1Kbytes.

En un procesador tipo HARVARD como el COP8SAX pueden identificarse como actores a los registros A, B, X y SP, los dispositivos de I/O, la memoria de datos (DM) y la memoria de programa (PM), y ciertos bloques funcionales necesarios para decodificación de instrucciones (ID) y generación de direcciones (AG). A diferencia del 6805, en un COP8SAX los registros B, X, SP y la I/O son también direccionables en el espacio de datos.

La memoria de programa y otras unidades: Un procesador requiere muchos otros módulos funcionales además de los descriptos, tales como la memoria de programa, y los bloques de generación de direcciones de programa, de decodificación de instrucciones, de manejo de ramificaciones, de manejo de interrupciones y de generación de señales de control

Device	Flash Program Memory (bytes)
COP8CBR9	32k
COP8CCR9	32k
COP8CDR9	32k

Con 32 kbytes enciende intermitentemente la memoria de programa con características de seguridad, usar memoria de programa virtual EEPROM

COP8	CB	R	9	H	VA	8
	Family and Feature Set Indicator	Program Memory Size	Program Memory Type	No. Of Pins	Package Type	Temperature
	CB = Low Brownout Voltage CC = High Brownout Voltage CD = No Brownout	R = 32k	9 = Flash	H = 44 Pin I = 48 Pin k = 56 Pin L = 68 Pin	LQ = LLP MT = TSSOP VA = PLCC	7 = -40 to +125˚C 8 = -40 to +85˚C

Programe la memoria. Esto es muy importante con moderno las aplicaciones microcontrolador basadas en, desde entonces programan la memoria Es generalmente ROM o EPROM mientras que la memoria de datos es generalmente RAM. Por consiguiente las tablas de datos constantes tienen que ser contenidas En la memoria non- imprevisible, por eso no están perdidos cuándo el microcontrolador es suministrado energía.

La eficiencia de clave es importante porque permite que a diseñadores lo haga las maletas más sobre funcionalidad de chip de - en menos memoria de programa Espacio (el ROM, el OTP o el flash). Seleccionar un microcontrolador con Menos el tamaño de memoria de programa traduce al sistema más bajo Gastos, y la seguridad adicional de saber que más clave puede Sea empaquetado en el espacio de memoria de programa disponible.

La familia de COP8 incluye una combinación única de la instrucción fije las características, que suministran a diseñadores con el óptimo eficiencia de clave y utilización de memoria de programa.

La eficiencia es debido a el hecho eso mejora las instrucciones Ser de la variedad de byte sola, resultando en el mínimo Espacio de programa. Porque polvera que la clave no habita uno Cantidad cuantiosa del espacio de memoria de programa, los diseñadores Poder integrar las características adicionales y la funcionalidad en el Espacio de memoria de programa de microcontrolador. También, la mayoría Las instrucciones ejecutadas por el dispositivo son el ciclo solas, dando como resultado En el tiempo de ejecución de programa mínimo. A decir verdad, 77 % del Las instrucciones son byte ciclo solo solas, proveyendo más grande Clave y eficiencia de E/S, y ejecución de clave más rápida.

La arquitectura del dispositivo es una arquitectura de Harvard modificada. Con la arquitectura de Harvard, la memoria de programa (El flash) estar separado de la memoria de la memoria de datos (el RAM). Tanto Memoria de programa y memoria de datos tienen ellos mismos se separar Se dirigir al espacio con autobuses de dirección distintos. La arquitectura,
sin embargo sobre la base de la arquitectura de Harvard, los permisos Transferencia de los datos de la memoria flash para el RAM.

La memoria flash y las usuario funciones de ISP (ver la sección 5.7), suministrar el usuario con la capacidad de usar el destello Programe la memoria para hacer una copia de seguridad de secciones usuario definir del RAM. Esto suministra el usuario con el mismo no es volátil eficazmente el almacenamiento de datos como EEPROM. La dirección, y ni siquiera la cantidad de la memoria usada, ser la responsabilidad del usuario, Sin embargo la memoria flash leyó y las funciones de escritura han ser suministrado en el ROM de bota. Un típico método de usar la característica de EEPROM virtual Ser para el usuario copiar los datos al RAM durante el sistema de inicialización, periódicamente, y si necesario, borrar la página de encienda y copie los contenido del RAM al regreso de el destello.

El registro de opción, ubicado en lo dirección 0x7FFF en el flash memoria de programa, es use arreglar al seleccionable por el usuario la seguridad, los WATCHDOG, y las opciones de HALT. El registro puede Sea programado solamente en la programación de memoria flash externa o modos de programación de ISP. Por lo tanto, el registro debe serlo programar al mismo tiempo as la memoria de programa. El Contenido del registro de opción enviar de la lectura de fábrica 00 Hex.

Este dispositivo suministra la capacidad de programar el programa la memoria mientras instalar en una tarjeta de aplicación. Esta característica es llamado en programación de sistema (ISP). Provee unos medios De ISP usando el MICROWIRE / signo más, o la usuario lata Suministre sí mismo la rutina de ISP personalizada. La fábrica instalar ISP usa el puerto de MICROWIRE / signo más. El usuario puede Provea su propia rutina de ISP que usa ningunas de las capacidad del dispositivo, como USART, ser paralelo el puerto, etcétera. La descripción funcional la organización de la característica de ISP consta del usuario destello La memoria de programa, el ROM de inicialización de fábrica, y algunos registros dedicado a efectuar la función de ISP. El ISP instalado en fábrica eso MICROWIRE / signo más de usos está ubicado en el ROM de bota. El tamaño del ROM de inicialización es bytes de 1K y también contiene la clave para facilitar la capacidad de emulación en el sistema. Si un usuario decide escribir su propia rutina de ISP, debe estar ubicado en el flash programe la memoria.

1.5 Memoria De Datos

El espacio de direccionamiento de memoria de datos incluye el on-chip que RAM y datos registran, los registros de E/S (la configuración, los datos y el alfiler), los registros de control, el registro de cambio de SIO de MICROWIRE / signo más, y los registros varios, y contadores se relacionaron con los relojes automáticos y el USART (con excepción de el reloj automático intermedio). La memoria de datos es direccionamiento directamente por la instrucción o indirectamente por el B, X y punteros de SP. La memoria de datos consta de 1024 byte de RAM. Dieciséis bytes de RAM es correlacionado como "Registrar" 0F0 en direcciones a Hex de 0FF. Estos registros pueden ser cargados inmediatamente, y también disminuyeron y evaluaron con el DRSZ (disminuya registro y contenedor si cero) la instrucción. El puntero de memoria registra X, SP, B y s son la memoria trazar un mapa de en este vacío en 0FC de ubicaciones de dirección a Hex de 0FF respectivamente. Con el otro registros estar disponible para el uso general.

El conjunto de instrucciones permite que cualquier bit en la memoria sea determinado, vuelto a poner o evaluado. Toda E/S y registros (excepto A y PC) son la memoria correlacionada; por lo tanto, las partes de E/S y partes de registro pueden ser directamente y por separado determinadas, vuelto a poner y evaluado. El acumulador (A) bits también pueden ser directamente y por separado evaluado. Nota: el contenido de RAM está no definido sobre lo poder -up.

Extensión de RAM de segmento de memoria de datos

0FF de dirección de memoria de datos es usado cuando una memoria trazó un mapa de la ubicación para el registro de dirección de segmento de datos (s). La memoria de la memoria de datos es direcionado directamente por una dirección de byte sola tampoco dentro de la instrucción, o indirectamente en comparación con la referencia del B, X, o los punteros de SP (cada uno contiene una dirección solo - byte). Este disco sencillo - byte al que la dirección permite uno abordar el alcance de 256 ubicaciones de las 00 a FF hexadecimal.

El bit superior de esta dirección solo - byte divide la memoria de la memoria de datos en dos secciones distintas como dar una idea general de antes. Con excepción de la memoria de registro de RAM de 00F0 de ubicaciones de dirección a 00FF, toda memoria de RAM es la memoria correlacionada con el bit superior de la dirección solo - byte ser igual al cero. Esto admite el bit superior del

La dirección solo - byte determinar si/no el alcance de dirección de base (de 0000 a 00FF) es extendido. Si esto superior

Bit es igual a uno (representar 0080 de alcance de dirección a 00FF), entonces/luego la extensión de dirección no tiene lugar. alternativo, si este bit superior iguala el cero, entonces/luego el s de registro de extensión de segmento de datos es use extender el alcance de dirección de base (de 0000 para 007F) de XX00 a XX7F, donde XX representa las 8 partes del registro de s. Por lo tanto, el segmento de datos 128 byte las extensiones están ubicadas de las direcciones hasta las que 0100 a 017F para los datos segmenta 1, 0200 para 027F para 2 de segmento de datos, etcétera, FF00 a FF7F para los datos segmente 255. El alcance de dirección de base de 0000 a 007F representa 0 de segmento de datos. Figure 7 ilustra cómo es usada in se extender la extensión de memoria de datos de registro de s el más bajo la mitad de la dirección de base

1.6 Entradas Y Salidas

El suministro de análogo para el convertidor analógico-digital. Debe estar conectado con VCC exteriormente. Este también requiere un nivel alto el convertidor digital-analógico presenta resistencia de escalera y dentro del convertidor analógico-digital.

El conector se fija a tierra convertidor analógico-digital. Debe estar conectado con GND exteriormente. Este también la parte inferior del convertidor digital-analógico presenta resistencia de escalera dentro del convertidor analógico-digital. El dispositivo contiene hasta seis puertos de E/S de 8 bits bidireccionales (A, B, C, E, G y L) y un puerto de E/S de 4 bits (F), donde cada bit individual puede ser arreglado por separado marca tanto directo una contribución (Schmitt las entradas de gatillo en adelante rescribe L y G), producto o control de nivel bajo TRISTATE. Tres fechas que ubicaciones de dirección de memoria que ara destinaron para cada uno de estos puertos de E/S.

Cada puerto de E/S tiene tres registros memoria trazar un mapa de 8 bits asociados, el registro de configuración, el dato de registro y fija el registro de entrada. Indica las configuraciones de puerto de E/S. Las fechas y los registros de configuración permiten cada parte de puerto de ser arreglados por separado

Inversor Trigger Schmitt: algunos censores no proporcionan señales digitales puras y es necesarios conformar dicha señal antes de aplicarla al microcontrolador, como en el ejemplo se muestra en la figura siguiente.

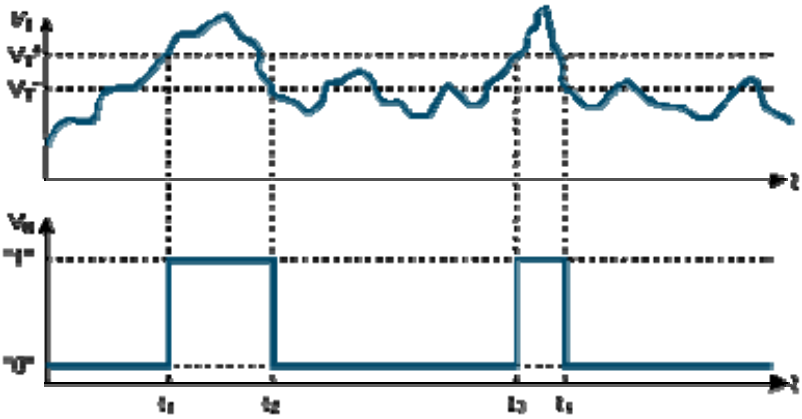
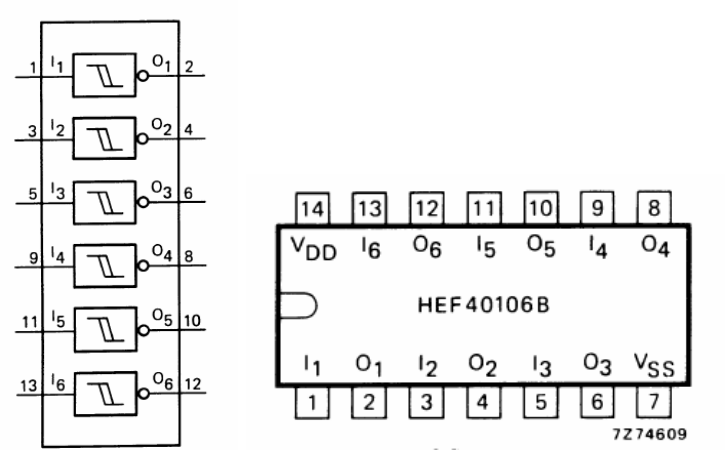


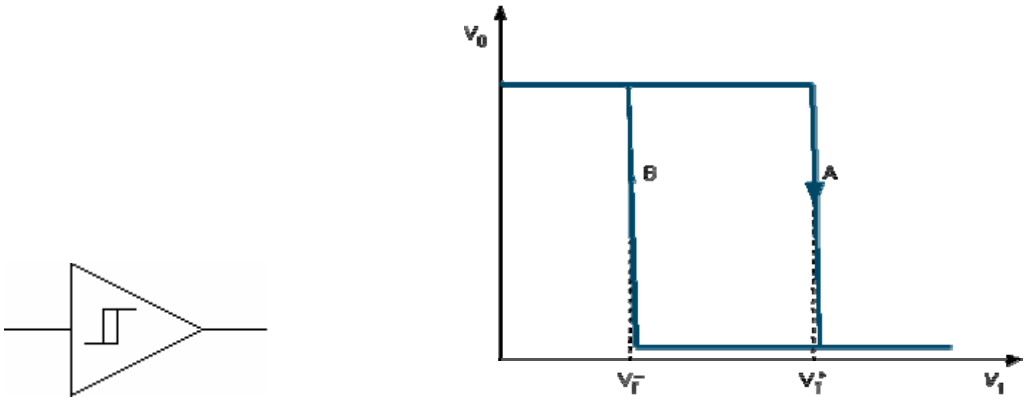
Figura de entrada y salida de un circuito Trigger Schmitt

Una forma sencilla de conformar una señal en digital es mediante puertas Trigger Schmitt, como las que tiene el circuito integrado 40106 este dispositivo Contiene 6 inversores Trigger Schmitt encapsulados según se indica en la figura siguiente



Estos Dispositivos tienen una característica de transferencia como se muestra en la figura siguiente, en esta cuerva se aprecia que si la tensión de entrada aumente desde 0V hasta el nivel alto de transición se produce siguiendo la cuerva A y conmuta para el valor V_T^+ denominado umbral superior por el contrario, si la entrada esta en nivel alto y disminuye hasta 0V la transición se siguiendo la cueva B cuando se alcanza el denominado umbral inferior V_T^- estos valores dependen de la tensión de alimentación.

La curva de transferencia del inversor Schmitt es importante observar que la transición de salida de Alto $\rightarrow$ Bajo es distinta que la de Bajo $\rightarrow$ Alto ha este fenómeno se le denomina como Histéresis



Curva de transferencia de un inversor Trigger Schmitt

Entradas

Un dispositivo externo otorga al Microcontrolador una señal en estado alto o bajo. El nivel lógico es leído por el Microcontrolador como un bit sencillo de información de entrada.

Salidas

El Microcontrolador forza uno de sus pines a un estado alto o bajo. El voltaje de salida en el pin corresponde a un bit sencillo de información.

Puertos

Un puerto es un grupo de pines utilizado para mandar o recibir información. Un puerto puede tener únicamente salidas, entradas o incluso una combinación de pines de entradas y salidas. Actualmente la mayoría de los puertos son bi-direccionales, es decir pueden ser configurados como pines de entrada o salida dependiendo de los requerimientos del usuario.

Usualmente cada puerto (grupo de pines) tiene asignada una dirección como si fuera un registro en memoria. La escritura a una dirección asignada a un puerto ocasiona que los pines asociados con la dirección del puerto sean forzados a un estado alto o bajo de acuerdo al valor escrito. Si los puertos no son mapeados en memoria, se tendrán instrucciones especiales de Entrada y Salida para accesarlos.

Rescriba un puerto de E/S de 8 bits a i. Todos alfileres de TO tienen gatillos de Schmitt en Las contribuciones. El 44 - alfiler que paquete no tiene al puerto de 8 bits completo Y contiene algunos pads unbonded y flotando interiormente en el Desportíllese.

El archivo binario valorado leyó i indeterminada de estas partes.

El software de aplicación debe ocultar éstos desconocidos Bits cuando leer el puerto para registro, o usos solamente bit – acceder Las instrucciones de programa cuando accede rescriben A..

Éstos desconectados Bits provoca el poder solamente cuando aran la tierra direccionamiento (i.e.., En zapatillas de clavos de resumen). rescriba I puerto de E/S de uno 8 bits de B.

Todos alfileres de B tienen gatillos de Schmitt on Las contribuciones. rescriba i puerto de E/S de uno 8 bits de C.

El dispositivo 44 alfiler no se ofrece rescriba C..

Los alfileres no disponibles que arados no limitaron. Para leer La operación sobre estos pines regresará imprevisible Valor.

Sobre este dispositivo, las citas de C de puerto asociadas y Los registros de configuración no deben ser usados. Todo que los alfileres de C tienen Gatillos de Schmitt sobre las contribuciones.

El puerto al que las atracciones de C no suministran energía cuándo Unbonded. Puerto e i puerto de E/S de uno 8 bits.

El dispositivo 44 alfiler no se ofrece rescriba E.. Los alfileres no disponibles que arados no limitaron. Para leer La operación sobre estos alfileres unterminateds regresará imprevisible Valor. Sobre este dispositivo, el puerto asociado y él salen y Los registros de configuración no deben ser usados. Todo AND que los alfileres tienen Gatillos de Schmitt sobre las contribuciones.

El puerto al que las atracciones de AND no suministran energía cuándo Unbonded. rescriba i de F al puerto de E/S de 4 bits. Todos alfileres de F tienen gatillos de Schmitt en Las contribuciones. El paquete 68 alfiler tiene menos de ocho alfileres de F de puerto, y Contiene pads unbonded y flotando interiormente sobre la desportilladura. El Valores binarios leyeron de estas brocas que araba indeterminado.

El software de aplicación debe ocultar estos bits desconocidos cuando leer el puerto al que registro de F, o usos solamente bit – acceder Las instrucciones de programa cuando acceder rescriben F..

El desconectado Bits provoca el poder solamente cuando aran la tierra direccionado (i.e.., En zapatillas de clavos de resumen). I puerto de uno 8 bits de G de puerto. G0 de alfiler, arados E/S bidireccional de G2.G5 Puertos. Pin G6 i siempre a entrada de Hi - Z para todo uso. Todos alfileres Tenga Schmitt Triggers sobre sus contribuciones. G1 el pin sirve realizar y dedicar el producto del guardián con tirón -up débil si el I de característica del guardián seleccionó por el registro de opción.

El de pin para la E/S para todo uso si de característica del guardián No seleccionar. Si la característica del guardián de la que escogió, morder 1 el El puerto que configuración de G y fechas registran no tiene ninguno Efecto sobre la instalación de G1 de alfiler.

Los saques de G7 hacen genial el producto dedicado Sujete con alfileres el producto de reloj para el CKO. Desde G6 i una contribución solamente alfiler y i de G7 el CKO dedicado Alfiler de producto de reloj, los bits asociados en las fechas y la configuración Se registra para G6 y arados de G7 usados para la determinación especial Las funciones de las que accedió una idea general abajo.

Leer las fechas de G6 y G7 Bits producirá los ceros. El dispositivo será puesto en el modo de HALT escribiendo en" 1" A bit 7 del G de puerto registra las fechas. De forma semejante el dispositivo sí Ser puesto en el modo de (capseq) escribiendo para realizar " 1 " a 6 de bit del Las fechas de G de puerto son recordadas.

CAPITULO 2

“ESTRUCTURA DE LOS MICROCONTROLADORES”

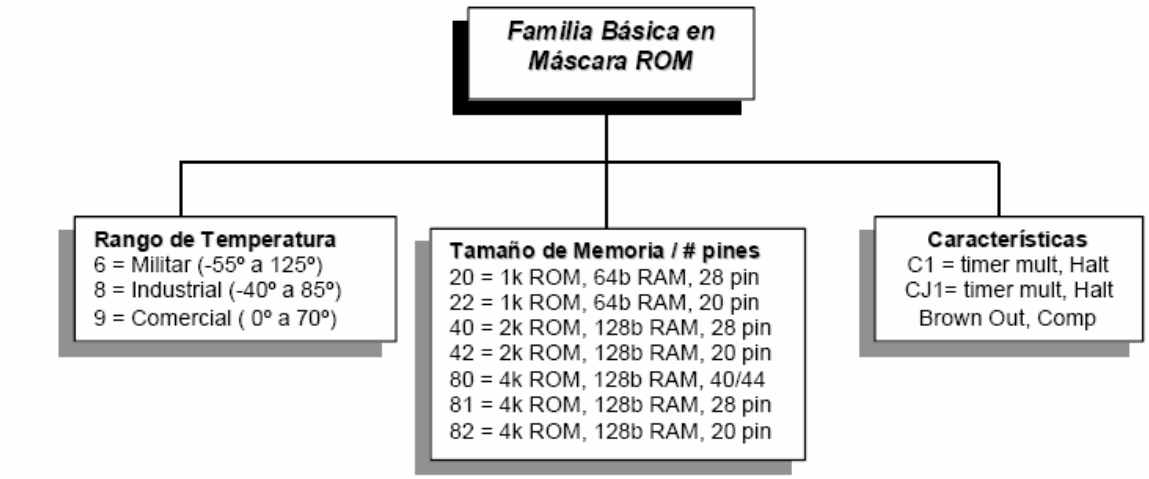
2.1 El microcontrolador COP8

Tamaño: La memoria de datos varía en tamaño dependiendo del Microcontrolador y/o la familia. (4-bit/8-bit/16-bit por 16, 32, 64, 128 bytes, etc.).

La familia del COP8 se divide en cuatro grandes grupos que son :
Familia Básica en Máscara
Familia Característica en Máscara
Familia OTP
Familia S

Familia Básica En Máscara

Como su nombre lo indica, son Microcontroladores con periféricos integradas de funciones sencillas . Podemos encontrar las siguientes variantes.

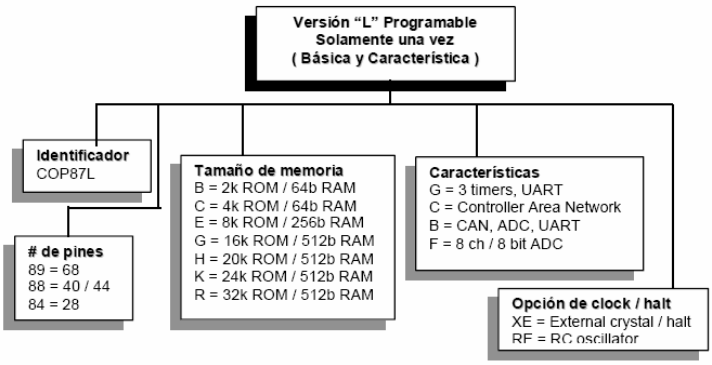


La notación del Microcontrolador se va dando de izquierda a derecha. Por ejemplo, en esta familia tenemos tres características (de Izq a Der) que son Rango de Temperatura, Tamaño de Memoria y # de Pines y por último tenemos la identificación de los periféricos integrados al controlador. Consecuentemente si tenemos un Microcontrolador COP842CJ1 identificaremos de forma inmediata que es un Microcontrolador de rango de operación en temperatura industrial, 2K en memoria ROM, 128 bytes en memoria RAM, 20 pines, un timer multifunciones, modo de ahorro de energía HALT, protección Brown Out y comparadores.

Estos Microcontroladores tienen como principal característica la máscara, es decir, que la compañía fabricante del Microcontrolador aplica en el útlimo proceso de manufactura del integrado una máscara que hace las veces de programación y por lo tanto nos ahorra este último paso. Todo esto es posible, bajo la condicionante de ordenar la manufactura en grandes volúmenes con nuestro programa residente en ROM.

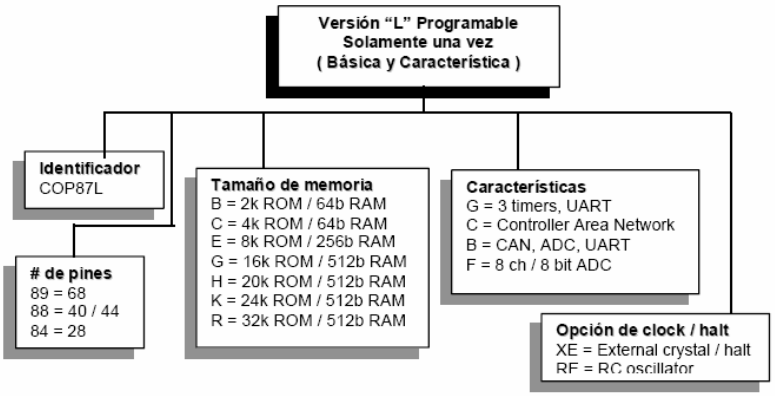
Familia Característica En Máscara

La segunda gran familia de COPs es la Familia Característica o en otras palabras la familia orientada a las Comunicaciones. En esta familia encotramos cuatro variantes de Izq a Der que son Temperatura, Número de Pines, Tamaño de Memoria y por último características en Periféricos. Al igual que la familia básica en máscara, estos Microcontroladores son programados por el fabricante. Aquí podríamos identificar todas las características del COP888RB (Controlador automotriz típico).



Familia OTP

La Familia OTP prácticamente involucra a todos los COP8. Ya que es la versión que se puede programar solamente una vez. Debemos tener especial cuidado con estos Microcontroladores ya que una programación mal realizada o la programación de un software no depurado hacen que el integrado ya no sea útil para realizar la tarea de control deseada. Esta familia tiene cinco variantes de Izq. a Der. Tenemos COP87L88RBXE que es un Microcontrolador de la familia OTP (identificador COP87L), puede ser de 40 o 44 pines, 32K en ROM, 512 bytes en RAM, Protocolo CAN y opción de cristal externo. De hecho, el COP87L88RBXE es la versión OTP del COP888RB de la Familia Característica en Máscara.

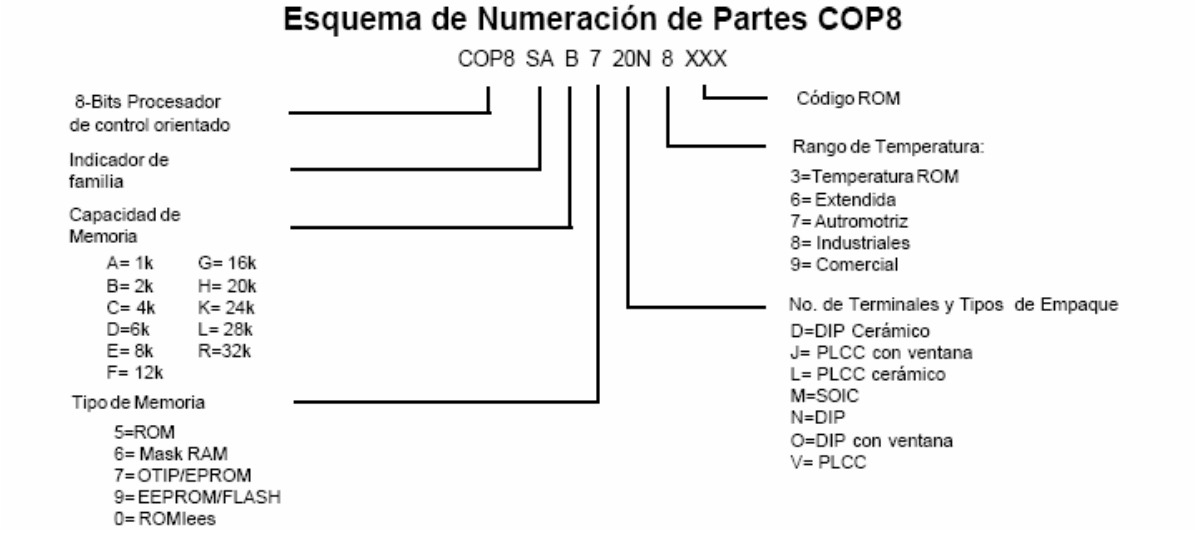


Familia S

La Familia S contiene los Microcontroladores más recientes y los de más variantes en características eléctricas y físicas (encapsulados). Aquí tenemos cuatro grupos que son :

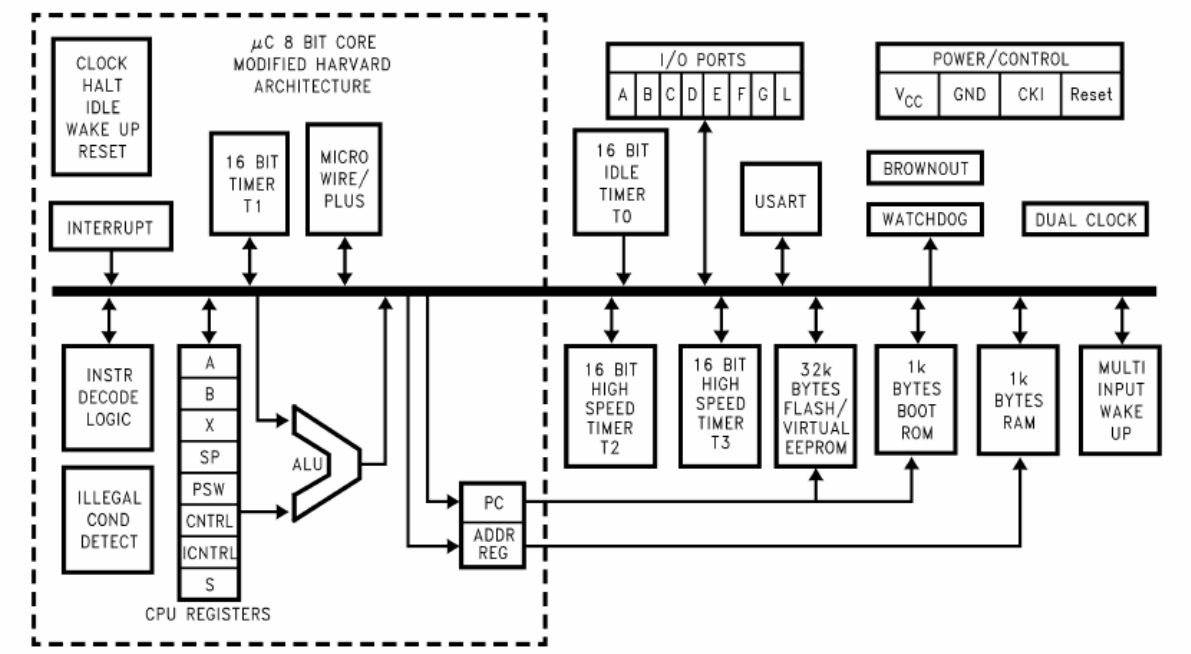
- Familia COP8SA
- Familia COP8SG
- Familia COP8AC
- Familia COP8SB/Familia COP8CB

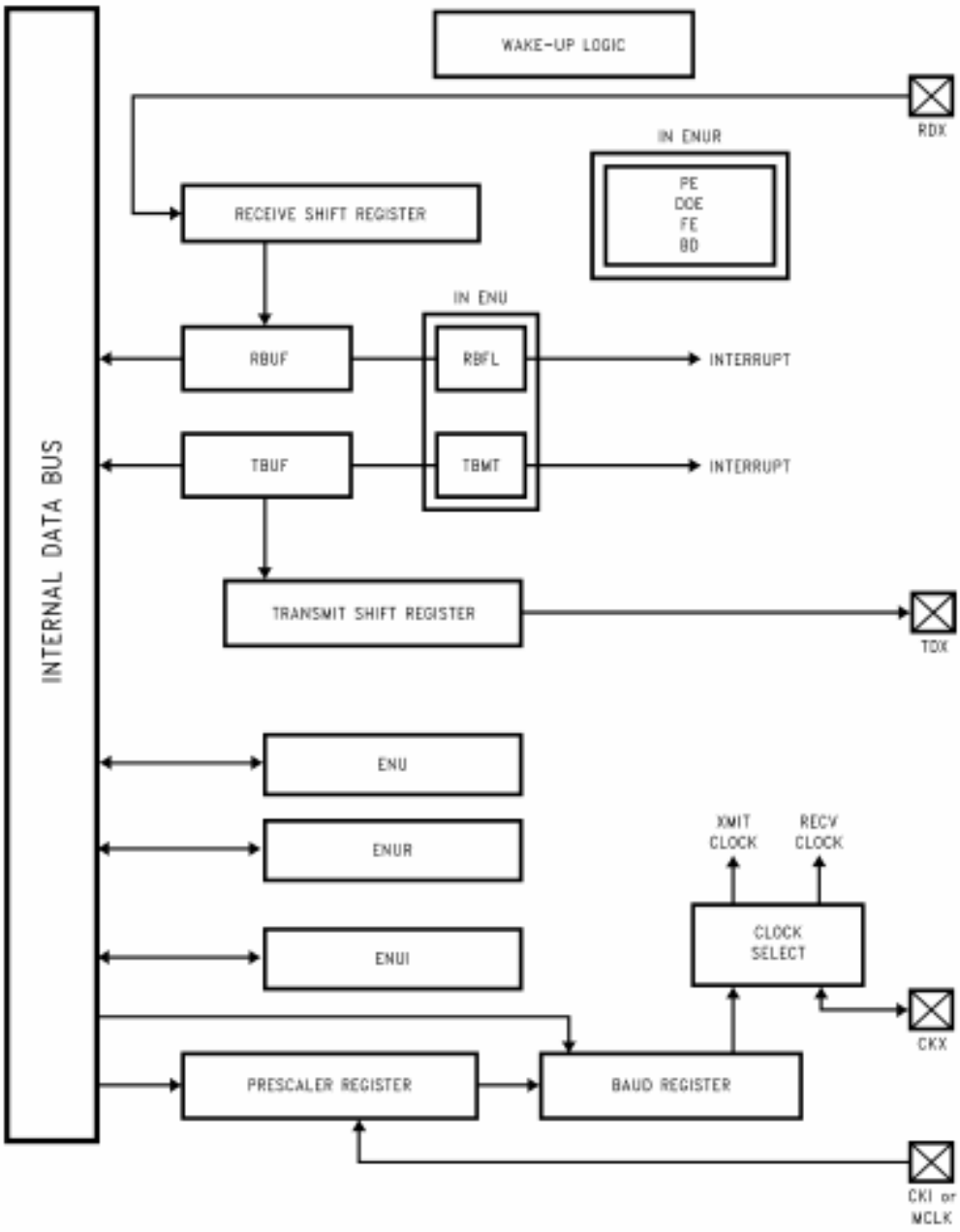
El COP8SA es un Microcontrolador de la familia caaracterística de 8 bits y un proceso EPROM de alta densidad, con una gran variedad en encapsulados, variedad en rangos de temperatura y voltaje, etc. Observar las variantes en el diagrama a la izquierda.



La Familia COP8SA se basa en la arquitectura Harvard Modificada, la cual permite que las tablas de datos se accesen directamente desde la memoria de programa. Consecuentemente las tablas de datos pueden estar en ROM o EPROM sin problemas que bajo alguna circunstancia los datos se puedan perder. El Microcontrolador COP8SA es un controlador de 5 puertos de propósito general (D, F, C, G, L), Memorias ROM que van desde 1K hasta los 4K, Memorias RAM desde los 128 Bytes hasta los 64 Bytes, funciones alternativas en algunos pines como “MultiInput Wakeup” orientados a “despertar” al COP8SA de los estados de bajo consumo de potencia. Por otro lado encotramos dos timers T0 y T1. El Timer de 16 bits T1 es capáz de funcionar en varios modos. En la figura se ilustra un diagrama a bloques del COP8SA (Figura 9). De cualquier forma, todas los bloques serán explicados en su funcionamiento a mas detalle en los capítulos siguientes.

Características de nuestro microcontrolador es basado en la arquitectura Harvard el Microcontrolador COP8SBR9 es un controlador de 8 puertos de propósito general, Por otro lado encotramos dos timers T0 , T1 , T3 y T2 una memoria Virtual de 32K EEPROM




WAKE-UP LOGIC
RDX
IN ENUR
PE
DOE
FE
BD
RECEIVE SHIFT REGISTER
IN ENU
RBUF
RBFL
INTERRUPT
TBUF
TBMT
INTERRUPT
INTERNAL DATA BUS
TRANSMIT SHIFT REGISTER
TDX
ENU
ENUR
ENUI
XMIT CLOCK
RECV CLOCK
CLOCK SELECT
CKX
PRESCALER REGISTER
BAUD REGISTER
CKI or MCLK

2.3 Set De Instrucciones del COP8

Cada Microcontrolador tiene un set de instrucciones. El usuario puede organizar las instrucciones en un orden lógico para crear un programa. El Microcontrolador ejecuta el programa para realizar una tarea específica. Para comprender completamente un set de instrucciones debemos entender los siguientes conceptos.

Opcode

El Opcode (Operation Code) es un código numérico de la instrucción que representa la operación a ser realizada por el CPU. Es decir, el Opcode es un grupo de bits los cuales le indican al Microcontrolador qué operación en particular hay que realizar. Por ejemplo un 64 podría significar “limpiar el Acumulador”.

Mnemonicos

Los Mnemonicos son nombres asigados a una operación en particular. Cada mnemonico es asociado con un opcode. El usuario puede hacer referencia a una operación por medio de un mnemonico “ADD” en lugar del Opcode “84”. Con el uso de los mnemonicos se hace más fácil escribir un programa. Por ejemplo, el mnemonico “LD” representa la operación “Cargar”.

En conclusión el usuario tendrá que crear una secuencia de tareas para realizar una acción de control. Esta secuencia de tareas será representada por instrucciones (mnemonicos). Luego entonces, es necesario convertir estos mnemónicos a lenguaje “entendible” para el Microcontrolador, por lo tanto, se translada esta secuencia de mnemónicos a una secuencia de Opcodes y datos adicionales.

2.4. Modo De Direccionamiento COP8

El set de instrucciones ofrece una amplia variedad de métodos para especificar la dirección de memoria. Cada método es llamado Modo de Direccionamiento.

Estos modos de direccionamiento son clasificados en dos categorías: Modos de direccionamiento con Operandos y Modos de direccionamiento de transferencia de control. Los modos de direccionamiento con operandos son 26 los métodos donde se especifica una dirección para accesar un dato (escribirlo o leerlo). Los modos de direccionamiento de transferencia de control son usados con las instrucciones de saltos para determinar la secuencia en la ejecución del programa.

2.5 Modo De Direccionamiento Con Operados

El operando de una instrucción determina la localidad de memoria que será afectada por la instrucción. Existen varios modos de direccionamiento de este tipo permitiendo que las localidades de memoria sean determinadas en diferentes formas. Una instrucción puede especificar una dirección de forma directa proporcionando una dirección específica, o indirectamente a través de un apuntador a memoria. En el modo inmediato, el dato a ser utilizado es el que es especificado en la propia instrucción. Cada modo de direccionamiento tiene sus ventajas y desventajas con respecto a la flexibilidad, velocidad de ejecución y lo compacto del programa. No todos los modos están disponibles con todas las instrucciones. Los modos direccionamientos disponibles son:

· Directo
· Indirecto con el registro B ó X
· Indirecto con el registro B ó X con un post Incremento/Decremento
· Inmediato
· Inmediato Corto
· Indirecto desde la Memoria de Programa.

Por ahora, sin detallar en las instrucciones, entenderemos la lógica de los modos de direccionamiento con un ejemplo.

2.5.1 Modo De Direccionamiento Directo

La dirección de memoria es especificada directamente como un byte en la instrucción. En lenguaje ensamblador, la dirección directa es escrita con un valor numérico (o alguna etiqueta que ha sido definida previamente en el programa con un valor numérico)

Ejemplo: LD A,05
(LD x,y) Carga x el contenido y Contenido antes Contenido después
ACUMULADOR (A) ? A6 hex
MEMORIA (05) A6 hex A6 hex
27

Esto significa que queremos mover al registro A lo que se encuentre en la localidad de Memoria RAM 05 hexadecimal. El contenido del registro A antes de la ejecución de la instrucción no es de nuestro interés mientras que el contenido de la localidad 05 de memoria RAM es A6 Hex. Por lo tanto, el nmemónico LD obliga a que el dato localizado en 05 se “copie” al registro Acumulador A. Ver el resultado en la tabla anterior.

2.5.2 Modo de direccionamiento indirecto con el registro B ó X

La dirección de memoria es especificada por el contenido del registro B ó el registro X (Son los registros punteros a RAM). En el lenguaje ensamblador, la notación que especifica qué registro es utilizado como apuntador es [B] ó [X].

Ejemplo: LD A,[B]
Carga al Acumulador Contenido antes Contenido después

ACUMULADOR (A) ? 87 hex
MEMORIA (05) 87 hex 87 hex
REGISTRO (B) 05 hex 05 hex

2.5.3 Modo De Direccionamiento Indirecto Con El Registro B Ó X Con Post Incrementos O Decrementos

La dirección de memoria es especificada por el contenido del registro B ó el registro X con la diferencia que después de tomar el valor apuntado por B ó X el registro apuntador es incrementado ó decrementado en uno automáticamente. Este tipo de direccionamiento es utilizado cuando hacemos referencia a localidades de memoria consecutivas. La notación en lenguaje ensamblador es [B+], [B-], [X+], [X-] donde se especifica que apuntador es utilizado y que operación realizar (incremento o decremento)

Ejemplo: LD A,[B+]
Carga al Acumulador Contenido antes Contenido después
ACUMULADOR (A) ? 87 hex
MEMORIA (05) 87 hex 87 hex
REGISTRO (B) 05 hex 06 hex
28

2.5.4 Modo De Direccionamiento Inmediato

En este modo el dato de la operación es agregado al Opcode de la instrucción en la memoria de programa. En el lenguaje ensamblador, el signo numérico (#) identifica un operando inmediato. Es decir, cuando anteponemos el signo # a un número identifica que estamos haciendo referencia a su valor mismo.

Ejemplo: LD A,#05
Carga al Acumulador Contenido antes Contenido después
ACUMULADOR (A) ? 05 hex

2.5.5 Modo De Direccionamiento Inmediato Corto.

Este es un caso especial de una instrucción inmediata. En la instrucción "carga inmediata al registro B", el valor de 4 bits inmediato de la instrucción es cargado en el nibble bajo del registro B. El nibble alto del registro B es limpiado al 0000 binario.

Ejemplo: LD B,#07
Carga al Acumulador Contenido antes Contenido después
Registro (B) 12 hex 07 hex

2.5.6 Modo De Direccionamiento Indirecto De Memoria De Programa

Este es un caso especial de una instrucción indirecta que permite el acceso a las tablas de datos guardadas en memoria de programa. En el caso de "Carga al Acumulador Indirecta" que es la instrucción LAID, los bytes altos y bajos del registro Contador de Programa son usados temporalmente como apuntador de memoria de programa con el propósito de accesar la información. El contenido del byte bajo del (PC) identificado como (PCL) es reemplazado por el contenido del registro (A), entonces el dato apuntado por la nueva dirección apuntada por (PCH) y por A es cargado en el acumulador y simultáneamente, el contenido original de (PCL) es recuperado para que el programa pueda continuar con la ejecución normal del programa.

Ejemplo: LAID
Carga al Acumulador Contenido antes Contenido después
ACUMULADOR (A) 1F hex 25 hex
Registro PCH 04 hex 04 hex
29
Registro PCL 35 hex 36 hex
Localidad de Memoria 04F1 hex 25 hex 25 hex

2.6 Modo De Direccionamiento De Transferencia De Control

Las instrucciones de un programa son ejecutadas usualmente en orden. Sin embargo, las instrucciones de salto pueden ser utilizadas para cambiar la secuencia de ejecución normal. Un cambio en la secuencia del programa no requiere de un cambio en el contenido del (PC). El bit más significativo del PC no es usado por lo que tenemos 15 bits para direccionar a la memoria de programa.

Existen diferentes modos de direccionamiento de transferencia que especifican una dirección nueva para el (PC). La elección del modo de transferencia depende, en primera instancia, de la distancia del salto. Saltos más lejanos algunas veces requieren más bytes para determinar completamente el contenido de (PC). Los modos de direccionamiento de Transferencia de Control son :

• Salto Relativo
• Salto Absoluto
• Salto Absoluto Largo
• Salto Indirecto

A continuación veremos como es afectado el (PC) con los direccionamientos de transferencia de control.

Salto Relativo

Esta instrucción es de un byte. Seis bits del Opcode especifican la distancia del salto desde la localidad donde se encuentre el (PC) apuntando en memoria de programa. La distancia del salto puede ser de un rango de –32 a +32. Un salto de una localidad no es permitido, el programador deberá utilizar una instrucción de No Operación “NOP”.

Ejemplo: JP 0A
Salto Relativo Contenido antes Contenido después
PCU 02 hex 02 hex
PCL 05 hex 0F hex
30

Salto Absoluto

Esta instrucción es de dos bytes de longitud, 12 bits del Opcode determinan el nuevo contenido del PC. Los tres bits más significativos del (PC) permanecen sin cambio, restringiendo que el nuevo valor del (PC) se encuentre en un espacio de 4Kbytes de la instrucción actual. (Esta restricción es relevante únicamente en dispositivos de más de 4K de memoria de programa.

Ejemplo: JMP 0125
Salto Relativo Contenido antes Contenido después
PCU 0C hex 01 hex
PCL 77 hex 25 hex

Salto Absoluto Largo

Esta instrucción es de tres bytes donde 15 bits del Opcode especifican el nuevo contenido del registro (PC).

Ejemplo: JMP 03625
Salto Relativo Contenido antes Contenido después
PCU 42 hex 36 hex
PCL 36 hex 25 hex

Salto Indirecto

Esta es una instrucción de un solo byte, el byte bajo de la dirección es obtenido de la tabla almacenada en la memoria de programa, con el Acumulador funcionando como byte bajo apuntando a la memoria de programa. Para propósito de accesos a la memoria de programa, el contenido del Acumulador es escrito en el (PCL) temporalmente. El dato apuntado por el registro Contador de Programa (PCH/PCL) es cargado a (PCL), mientras (PCH) permanece sin cambio.

La instrucción VIS es un caso especial de direccionamiento indirecto, donde el vector de dos bytes asociado con la interrupción es transferido al (PC) para saltar a la rutina de servicio asociada a la interrupción. 31

Ejemplo: JID
Salto Relativo Contenido antes Contenido después
PCU 01 hex 01 hex
PCL C4 hex 32 hex
Acumulador (A) 26 hex 26 hex
Localidad en Memoria 0126 32 hex 32 hex

CAPITULO 3

“DISEÑO DE CONSTRUCCION”

3.1 Propuesta De Un Diseño

3.1.1 Objetivos en el desarrollo de programador

1. Poner en practica los conocimientos adquiridos durante la estancia en la vida académica

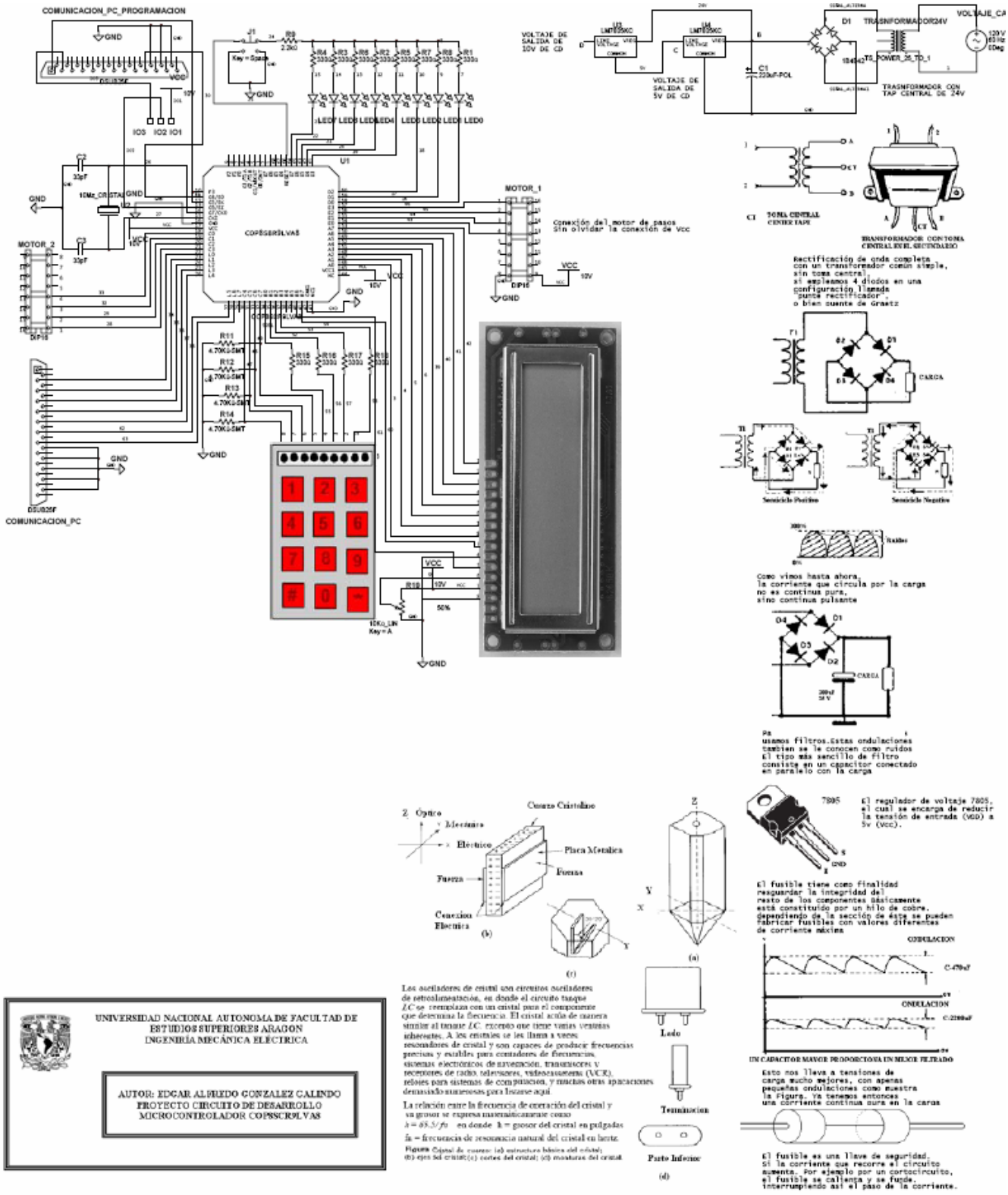
2. Desarrollar un circuito que nos permita manipular diversas aplicaciones

3. Aplicar un circuito para la implementarlo en el laboratorio de electrónica

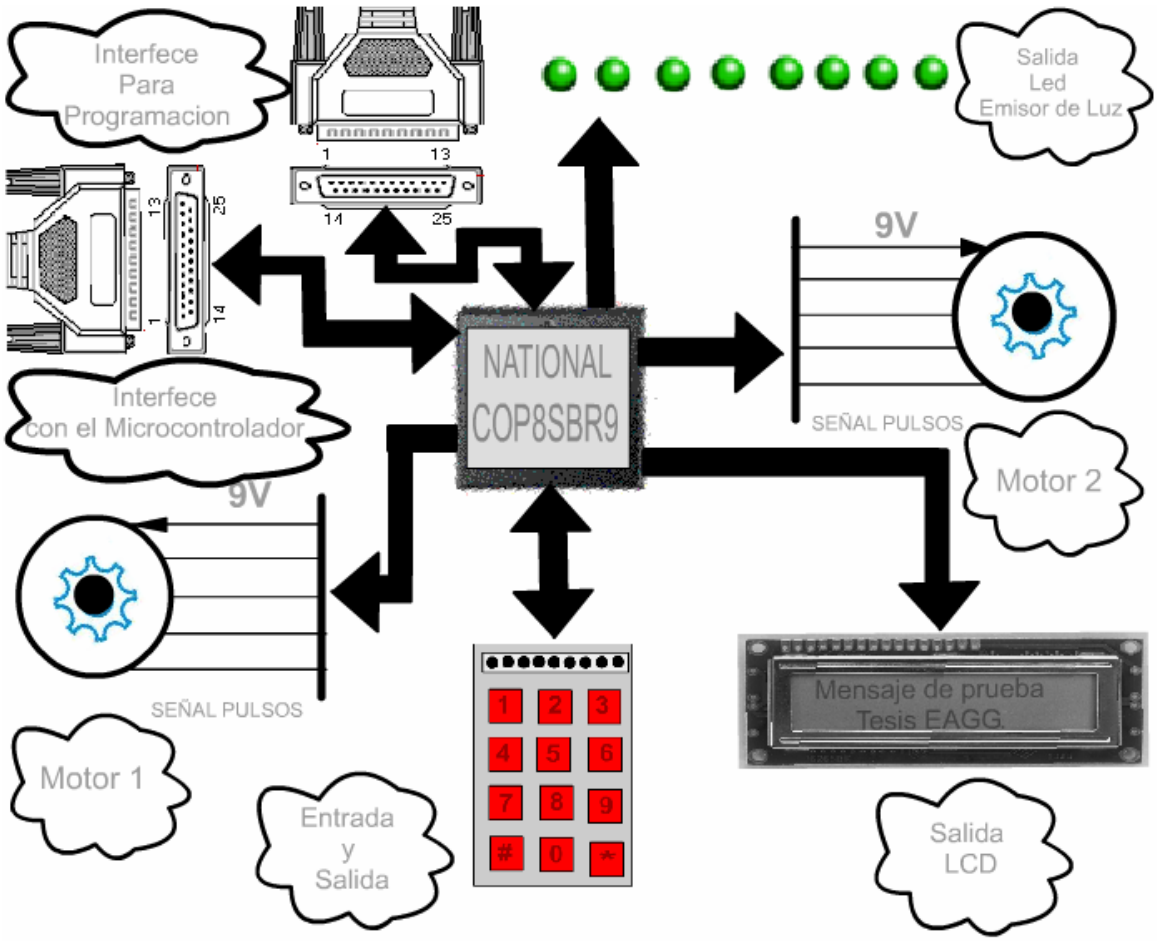
4. Que el Alumno le permite poner en práctica su creatividad en el desarrollo para crear proyectos que permitan su desarrollo académico.

Presentamos algunos empresas que desarrollan productos que nos servirán para auxiliarnos en el desarrollo del programador. La empresa Byte Craft desarrolla software que nos permitirá trabajar con un lenguaje C y con un compilador que nos permita traducir de un lenguaje escrito a un lenguaje de Maquina. En nuestro caso nos enfocaremos a las herramientas que nos ofrece el fabricante del microcontrolador

3.2 Circuito Esquemático



3.3 Animación del Microcontrolador

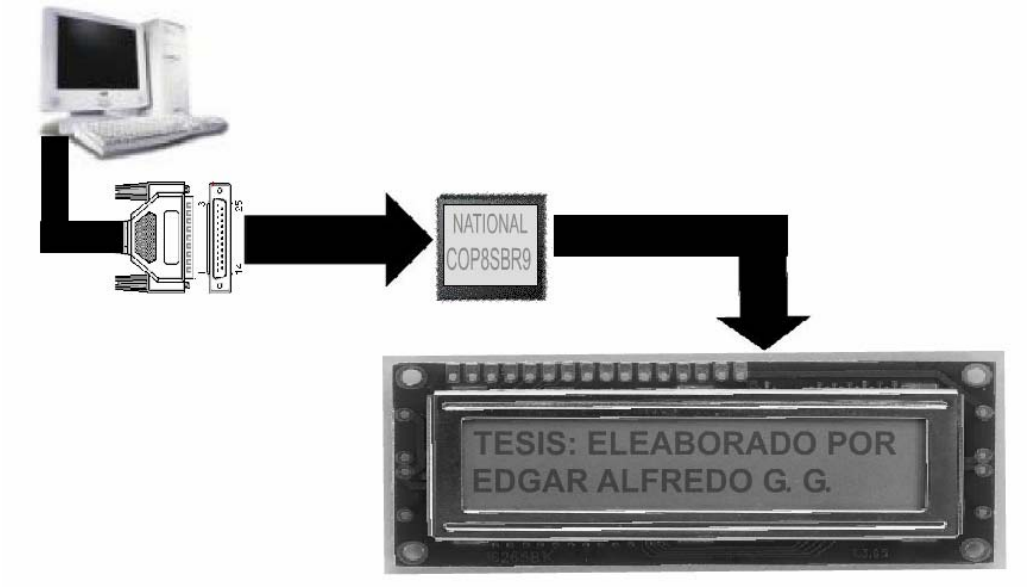


UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

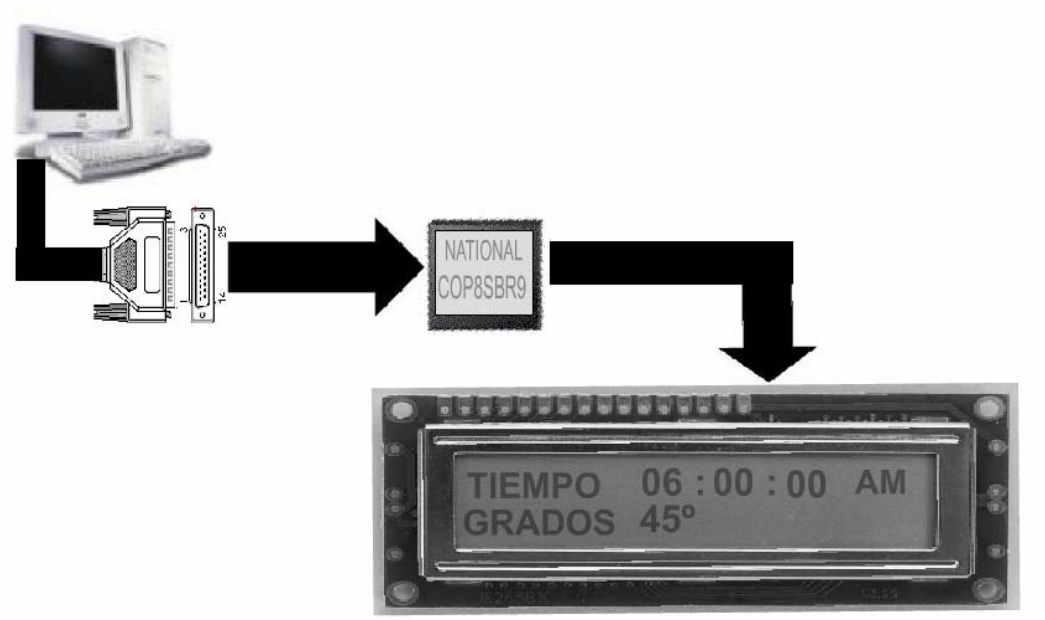
ANIMACIÓN

AUTOR: EDGAR ALFREDO GONZALEZ GALINDO ASESOR: ING. JOEL LOPEZ CONTRERAS
TESIS: DISEÑO Y CONSTRUCCIÓN DE UN PROGRAMADOR PARA LABORATORIO A BASE DE UN MICROCONTROLADOR

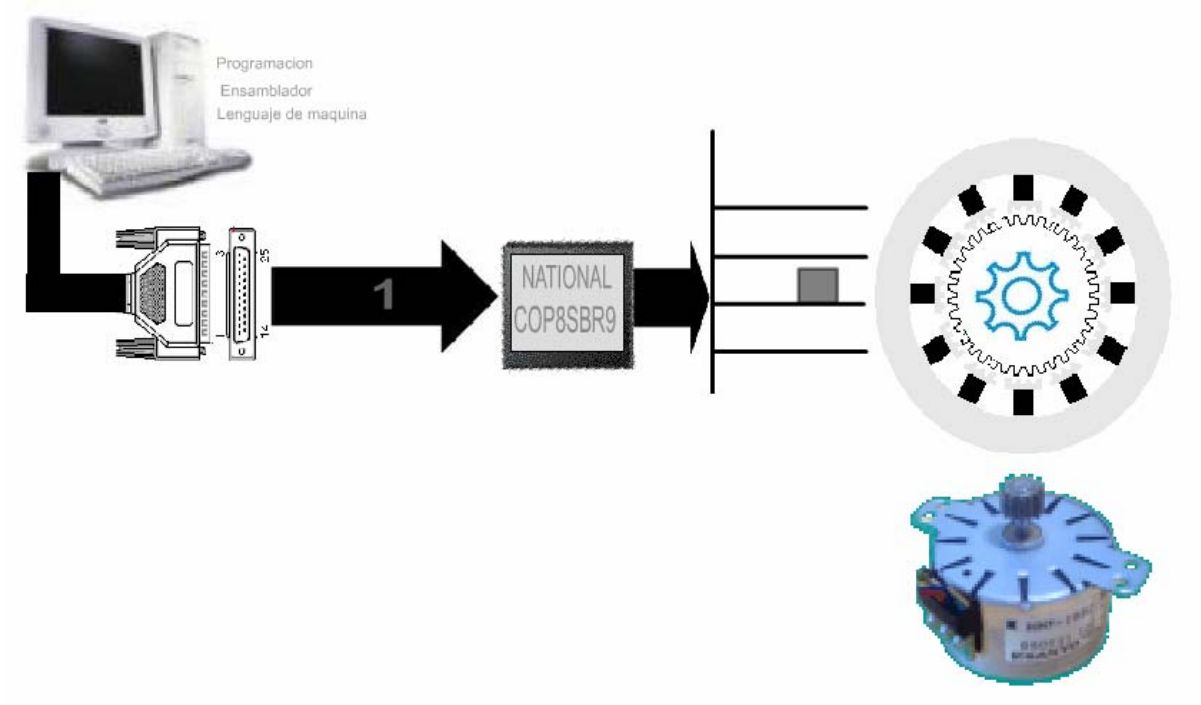
Simulación de un programa que es visualizado en el LCD en donde solamente envía caracteres



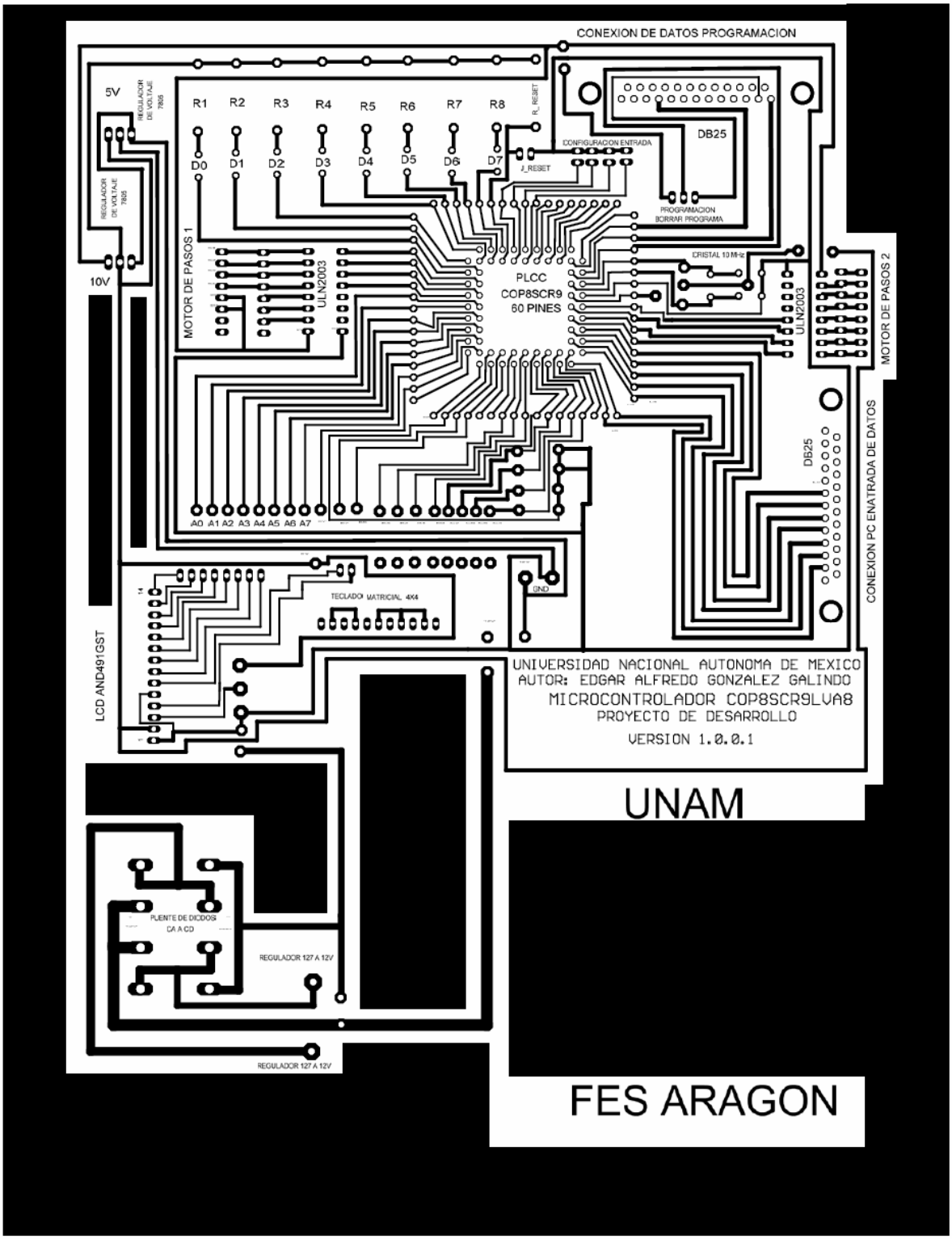
Simulación de un programa que es visualizado en el LCD en donde la aplicación puede ser en una antena en donde en cierto tiempo se desplaza a un cierto Angulo y otra aplicación puede ser un calentador de agua



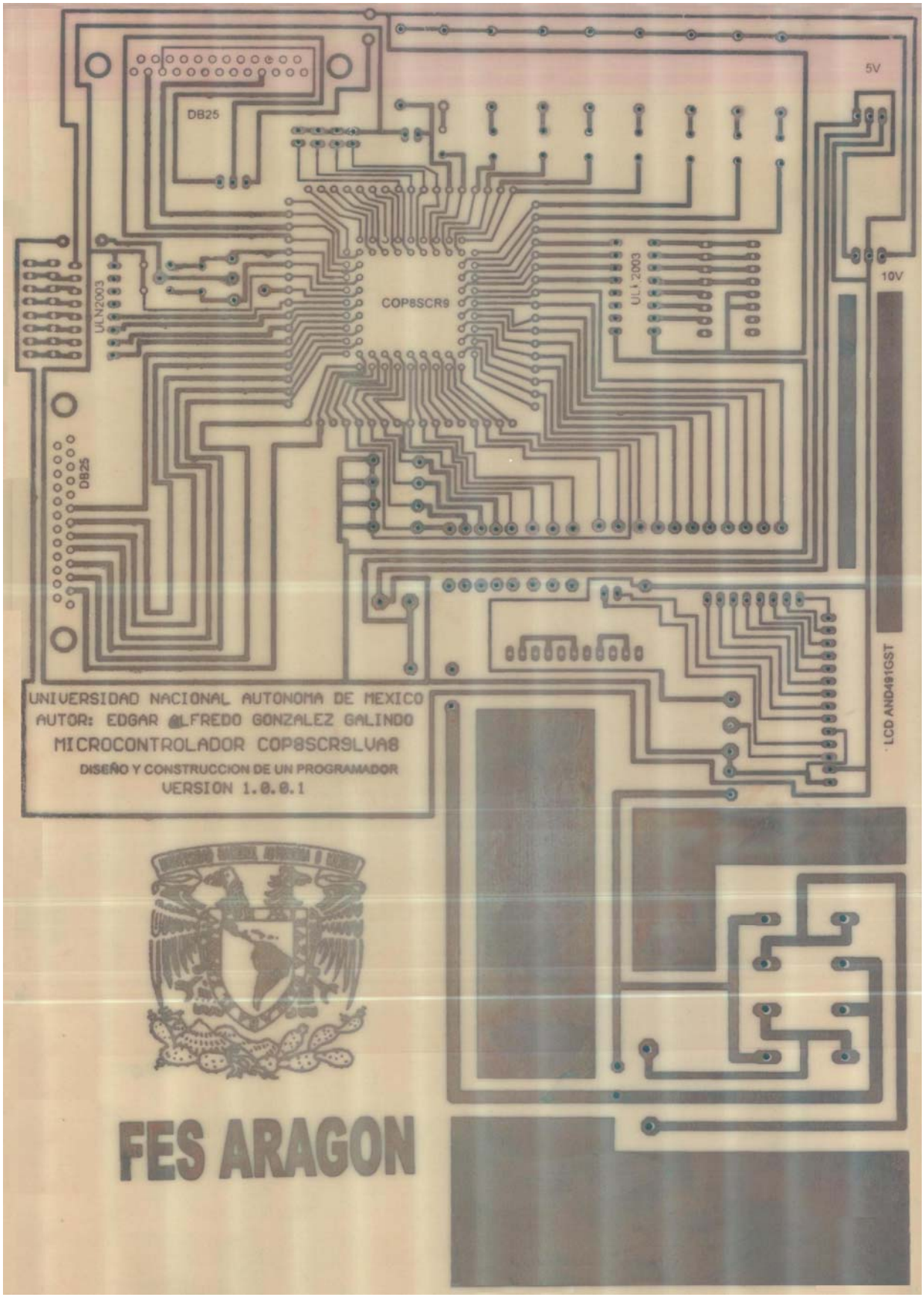
Simulación de un programa que manipula un motor de pasos por medio de pulsos y esta información es enviada por el microcontrolador.



3.4 Circuito Impreso Realizado con el software



3.5 Circuito Impreso Real del programador



3.6 Aplicación Rehabilitación En Terapias Físicas

3.7 Recursos De La WEB para el Programador

Para la construcción de nuestro programador utilizaremos las siguientes herramientas, y estas herramientas se encuentran disponibles en la WEB,

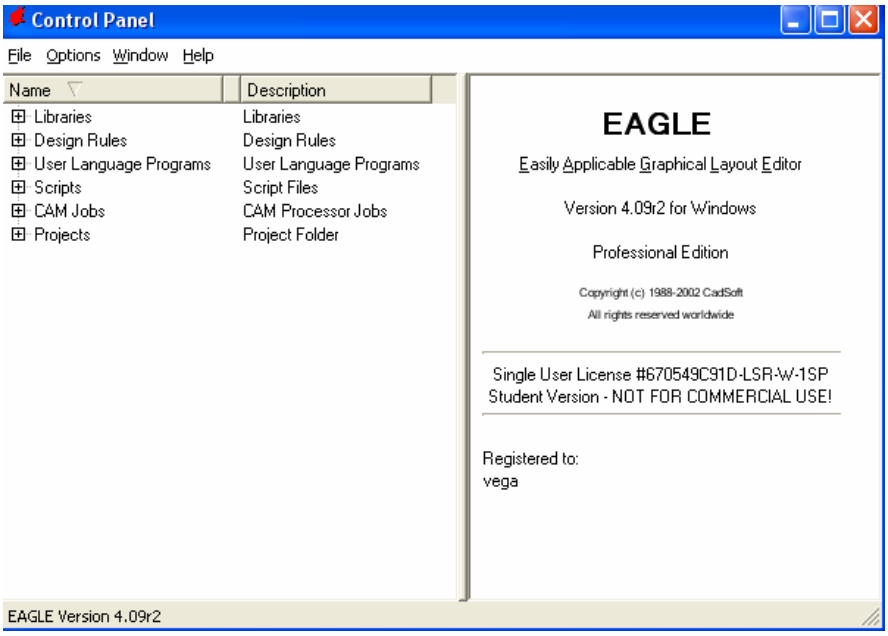
Los Recursos se mencionan a continuación

1. EAGLE
2. COP8FLASH
3. COP8 NSASM
4. MULTISIM Ver 9.0

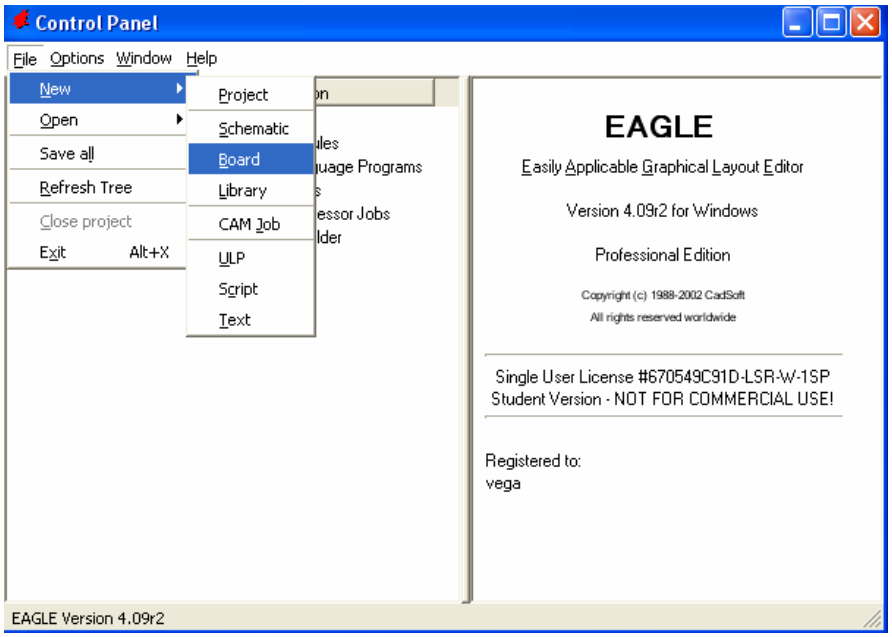
3.6.1 Eagle

Eagle es uno de los programas mas completos para elaborar circuitos impresos, para hacer uso de este programa se debe descargar el programa de la red como modo de prueba o en su caso comprar el paquete con la licencia para el uso al igual que el otro paquete nos dirigimos a Inicio – Programas - Eagle Ver 1.5 y ejecutamos o en su defecto damos clic únicamente en el acceso directo para ejecutarlo como se muestra en la figura 2.x

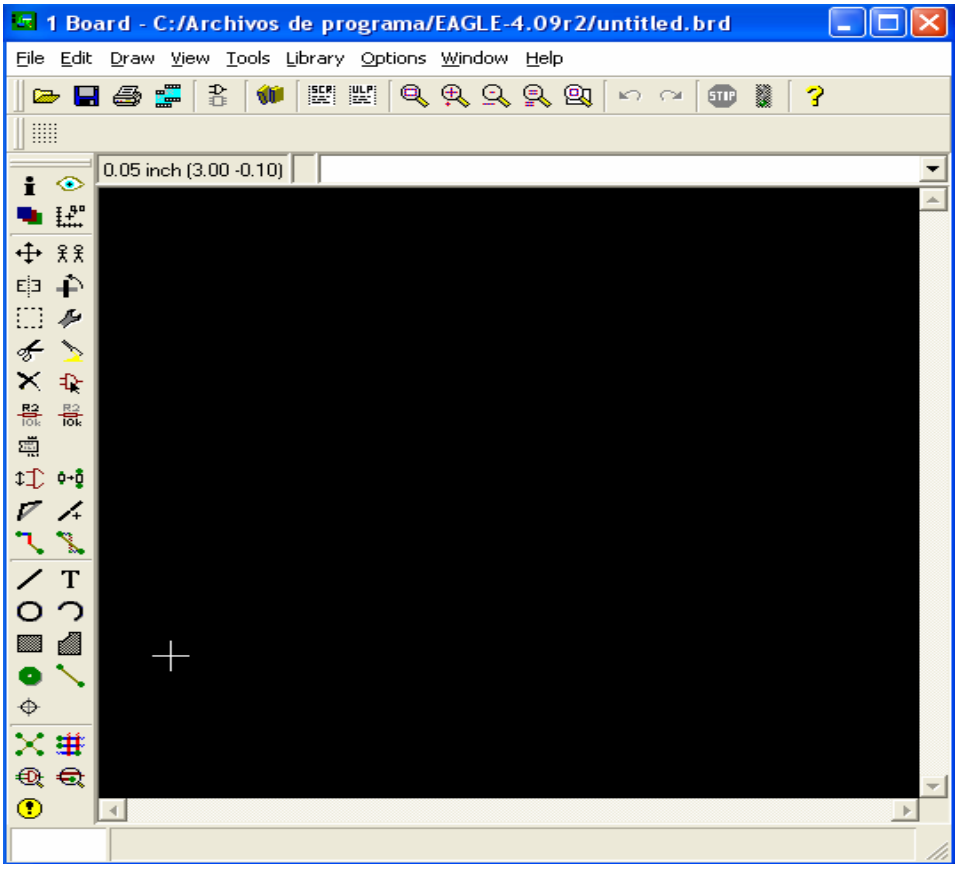
1. La ventana nos muestra la diversas herramientas que podemos hacer uso como son los diagramas esquemáticos, así como la realización de circuitos impresos o exportar de un Cto. Esquemático a el impreso.



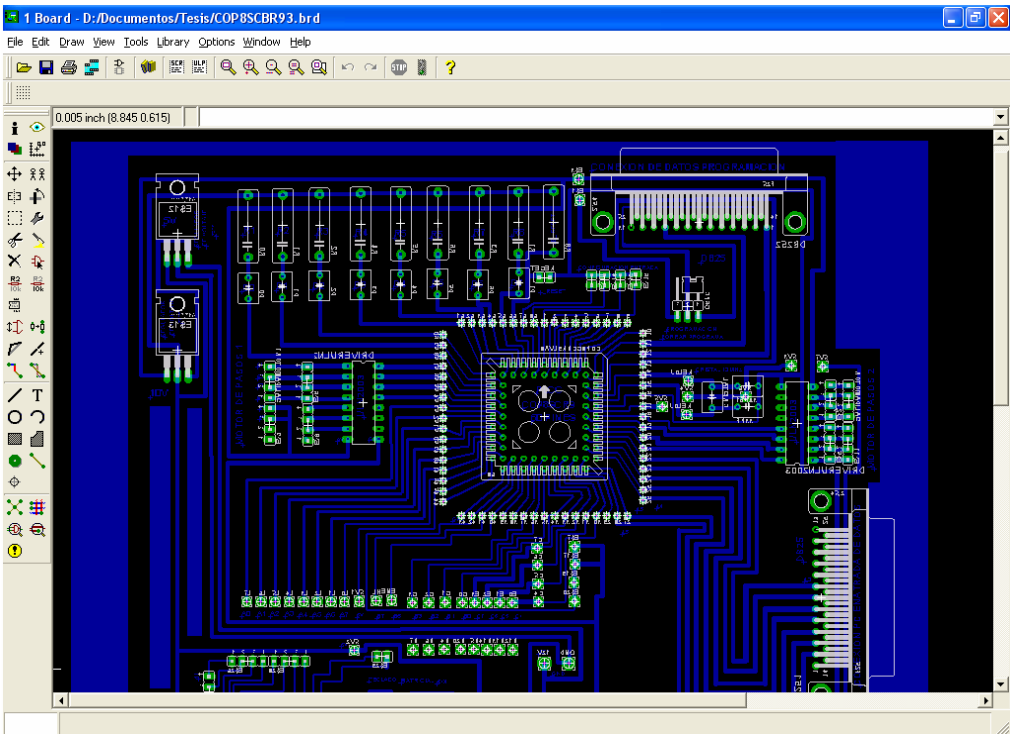
2. Para el comienzo del desarrollo del Cto. Impreso vemos la ventana como comenzamos a utilizarlo entramos a File – New – Board.



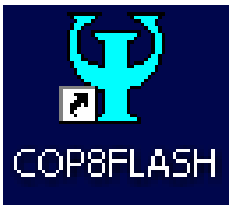
3. Inmediatamente se ejecuta la siguiente ventana que nos permitirá manipular las pistas así como los componentes electrónicos a utilizar.



4. En esta ventana como podemos observar hemos concluido la elaboración del cto. Impreso, este programa llamado Eagle que nos permite con todos sus recursos elaborar con creatividad la forma que uno desee y así como las dimensiones que uno requiera.



1. COP8FLASH



Es software es uno de los recursos que nos ofrece el fabricante para poder tener una comunicación con nuestro programador, aquí solo necesitamos tener el archivo en hexadecimal para poder transferir la información a través del puerto paralelo, este software tiene la ventaja de descargarlo en la red. Desde la pagina del fabricante.

Ejecutamos el software y veremos que la ventana qua aparece es el siguiente nos indica el fabricante figura 1.

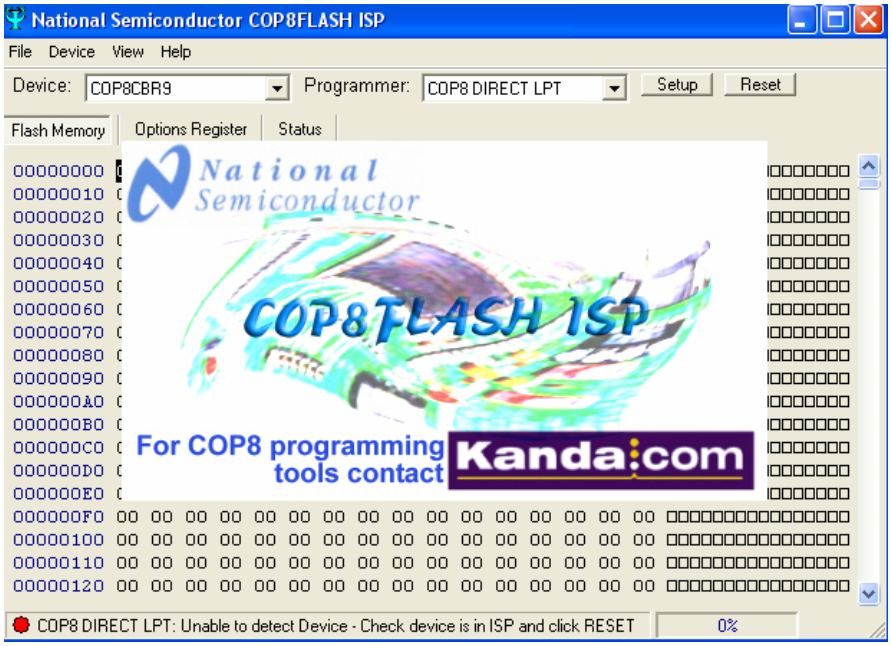


Figura 1.

Posteriormente que ya ejecutamos el software nos muestra la ventana siguiente y comenzaremos a realizar la transferencia de información figura 2.

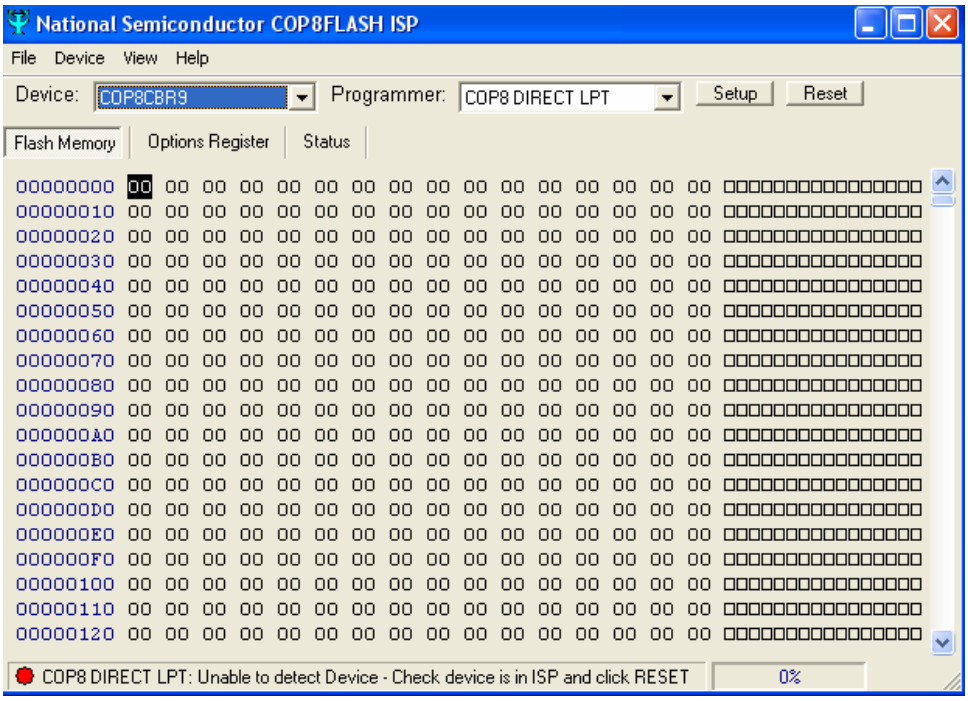


Figura 2.

Si ves la ventana anterior tiene un indicador en la parte inferior si presenta un color rojo eso quiere decir que no hay comunicación con nuestro programador, si este cambia de color verde veremos que en ese preciso momento ya tenemos una comunicación con nuestro programador figura 3.

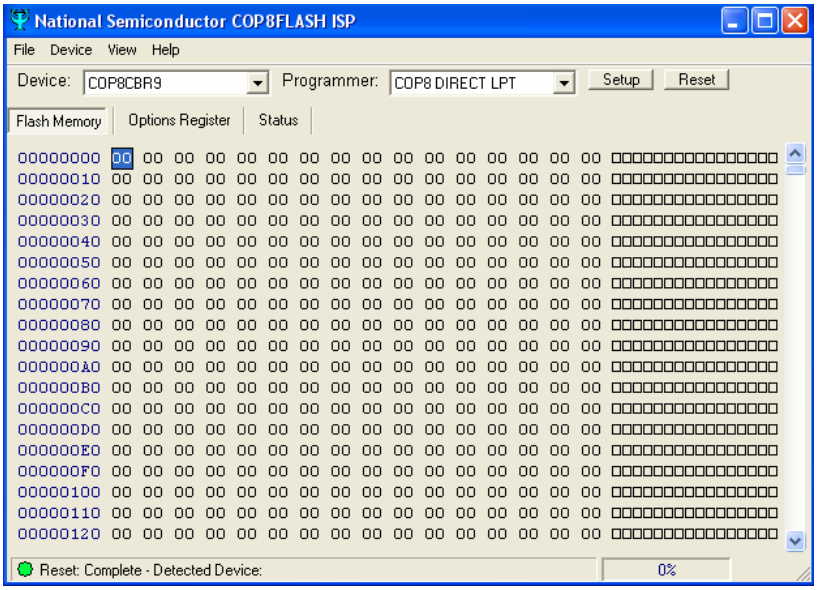


Figura 3.

Para comenzar a utilizar este programa entramos a File – Load – Flashfile estamos realizando la carga de el archivo en hexadecimal figura 4.

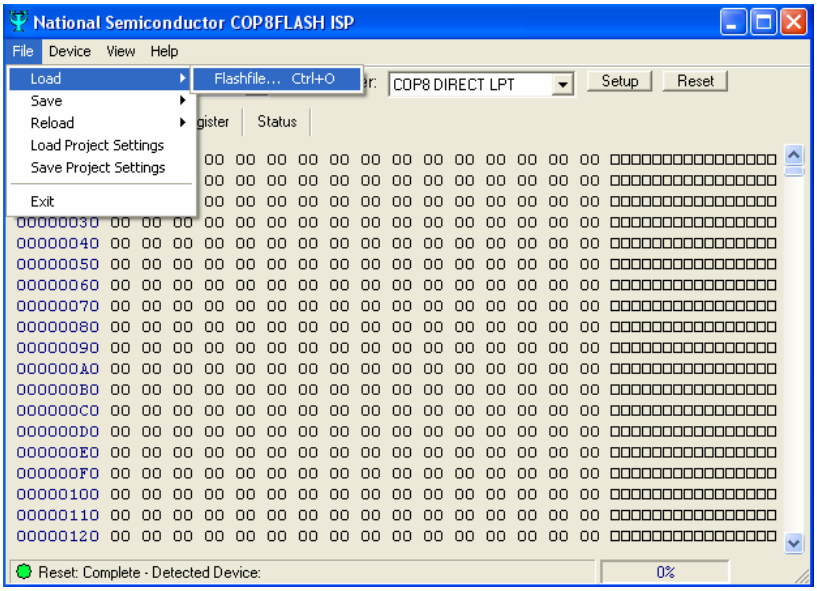


Figura 4.

Seleccionamos el archivo exploramos en la carpeta donde lo tenemos guardado, para cargarlo en el software y realizar su transferencia figura 5.

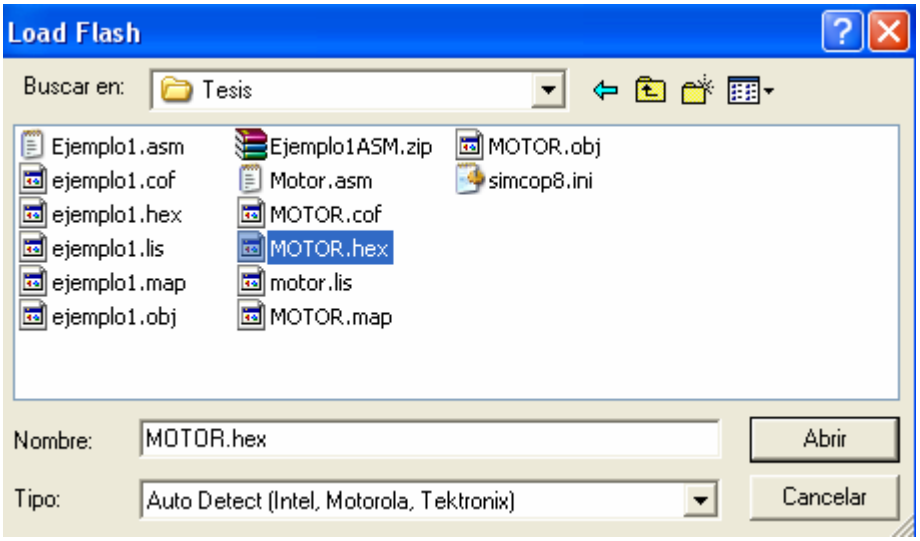


Figura 5.

En esta ventana ya cargamos en el software el archivo y la ventana nos muestra la siguiente imagen para comenzar a realizar la transferencia figura 6.

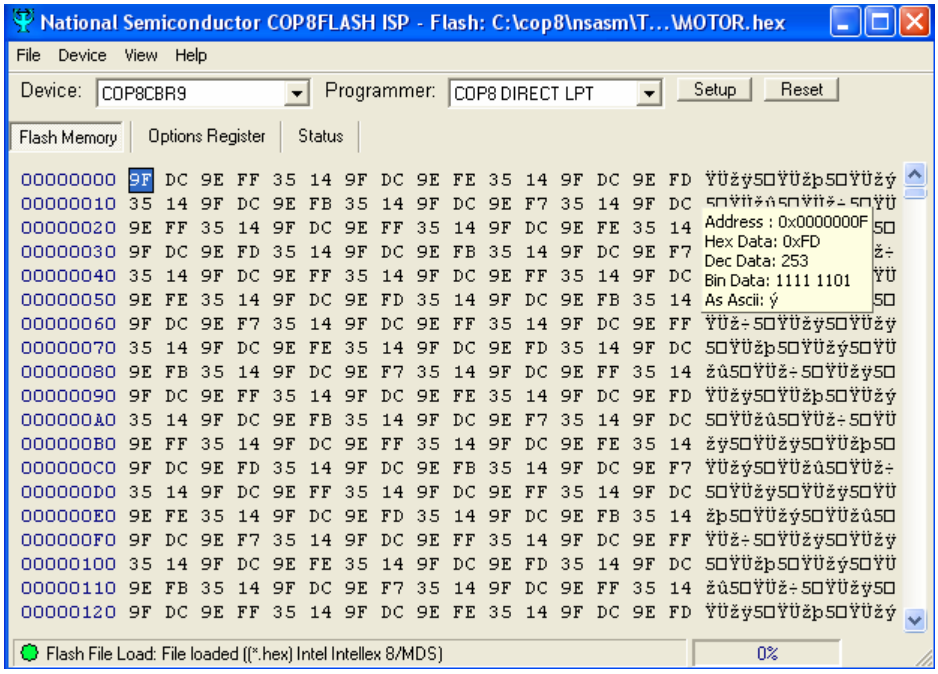


Figura 6.

Para realizar la transferencia entramos a dispositivos – Programa y Flash y realizamos la transferencia y ejecutamos como se muestran en las dos figuras 7 y 8 .

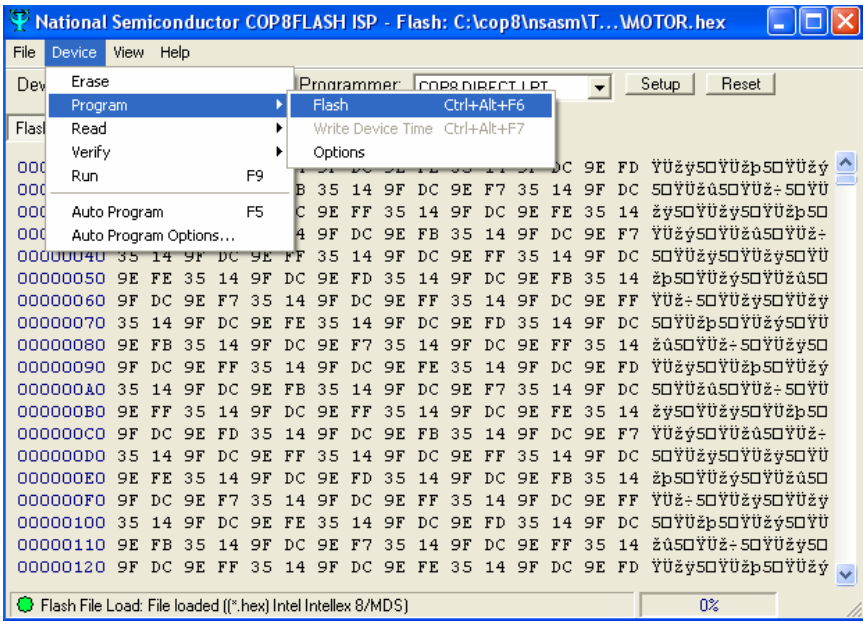


Figura 7.

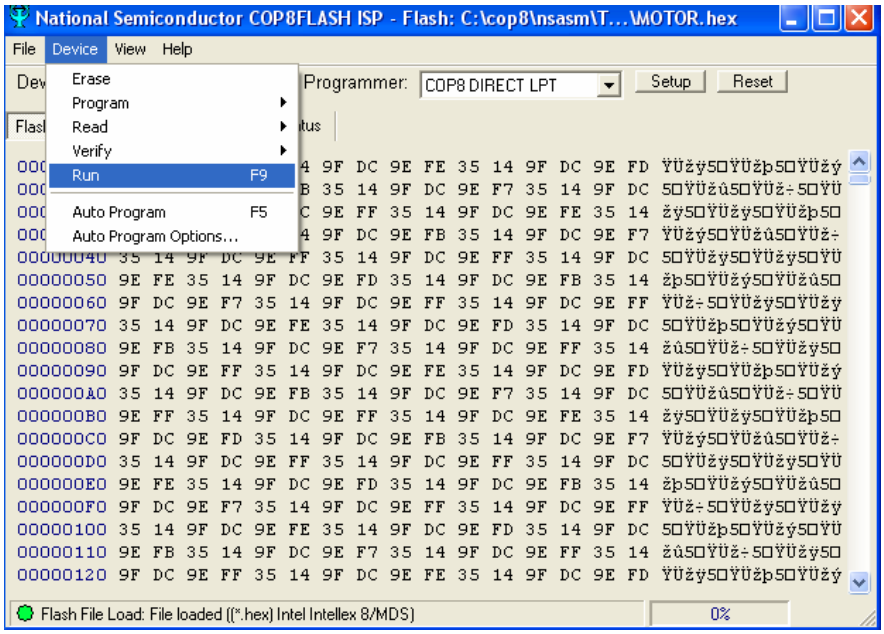


Figura 8.

Observa en la parte inferior como se va realizando la transferencia y no los indica en porcentajes, cuando realice el terminó de la transferencia comienza a ejecutarse nuestro programador, y cuando se finaliza la transferencia se desactiva la comunicación por lo que el software ya no detecta el programador, por lo que en ese instante cambia a su color inicial “Rojo como se muestra en la figura 9 y 10”

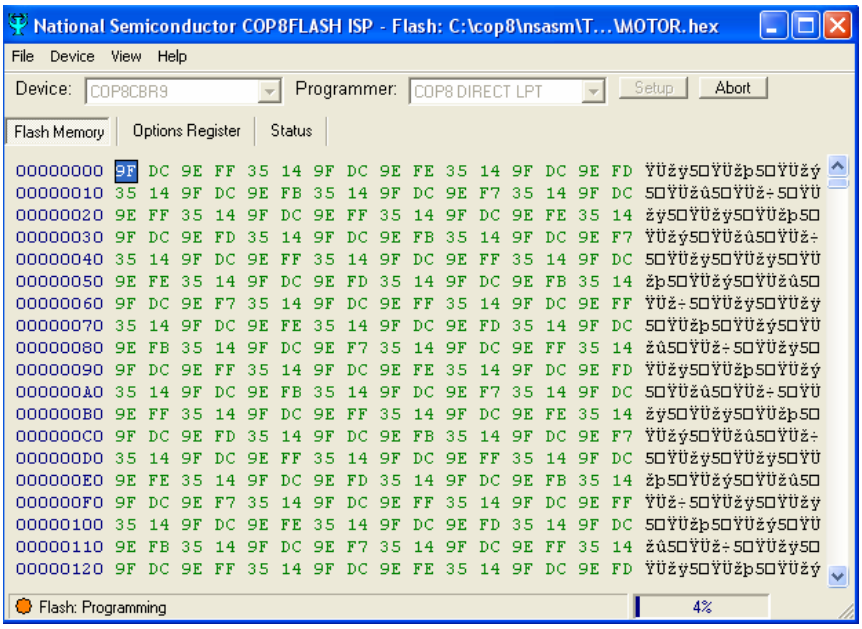


Figura 9.

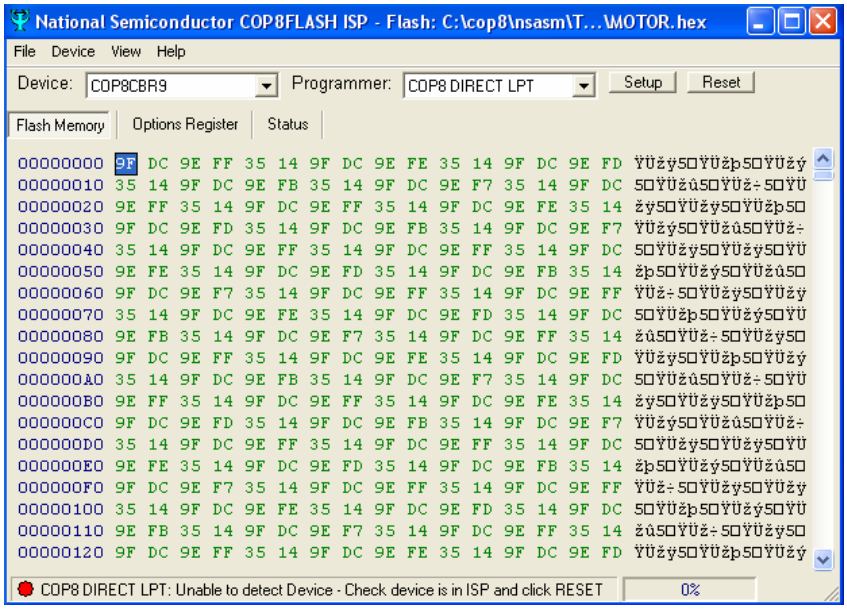


Figura 10.

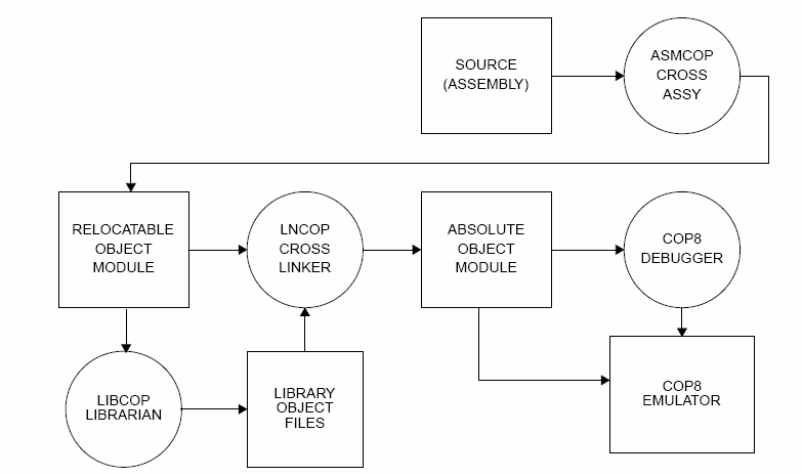
2. COP8 NSASM

ASMCOP traduce archivos fuente en módulos de objeto que contienen instrucciones en el archivo binario el lenguaje máquina. Estos módulos de objeto son conectados con LNCOP para generar uno total de módulo de objeto; el objeto total para el que el módulo puede ser cargado en el simulador del COP8 la depuración de programa y emulación. Los módulos de objeto de ASMCOP también pueden ser combinados una librería para el uso posterior en otros programas usando LIBCOP.

A la ayuda de la que el programador, el ensamblador opcionalmente generan un listado de producto la sentencias de fuente, el código máquina, las ubicaciones de memoria, los mensajes de errores, y el otro Información útil para depurar y verificar programas.

El ensamblador tiene varias directivas que controlan la operación del ensamblador. Para el ejemplo, el. Titulo sobre el que la directiva controla el título identificando que el ensamblador imprime Las páginas de los listados de producto; el. BYTE que la directiva ordena al ensamblador que generar Datos en bytes de memoria.

La Figura siguiente muestra cual es proceso de desarrollo del software.



Para el sistema operativo de MS-DOS, la instrucciones en la línea de comandos son los siguientes: asmcop, lncop y promcop, cada uno de estos comandos nos crearan archivos que nos van a servir para realizar otras tareas, si las instrucciones que realizamos están correctas no nos mandara mensajes de errores, y si las instrucciones son incorrectas el programa inmediatamente nos mandara el mensajes de error y el procedimiento es verificar nuevamente que el lenguaje este correcto, para ejecutar estos comandos tenemos que dirigirnos a la ruta donde se encuentra el archivo a compilar, como veremos en las siguiente ventanas que nos muestran las dos siguientes figuras 1 y 2.

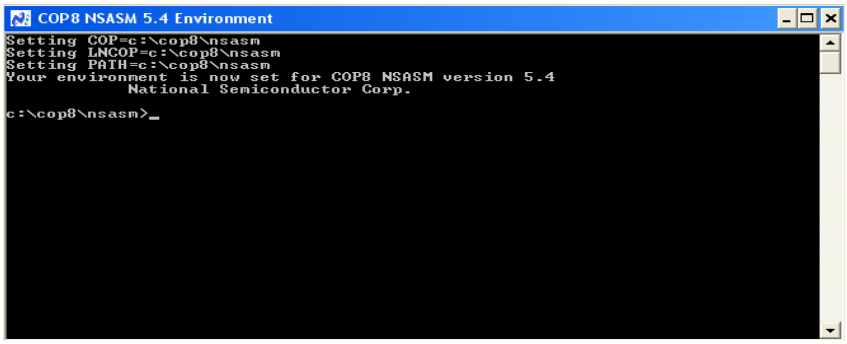
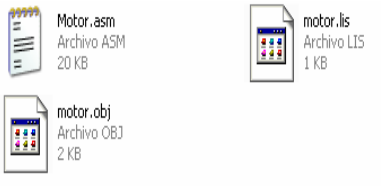


Figura 1.



En esta ventana vemos que no mando mensaje de error por obvias razones el lenguaje es correcto por lo que omitimos las ventanas de errores y nos genera archivos que nos van a auxiliar para generar otros archivos como a continuación se mostrara en las siguientes ventanas.

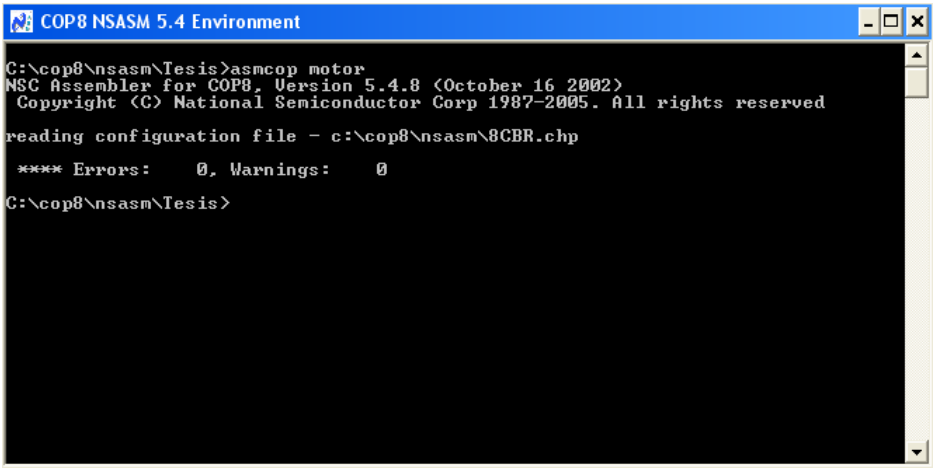


Figura 2.

Como podemos observar se generan dos archivos con extensión .obj y .lis estos archivos nos servirán para generar otros archivos y al mismo son auxiliares para realizar simulaciones, ahora ejecutamos el siguiente comando y nos genera dos archivos con extensión .cof y .mapcomo se muestra en la figura 3.

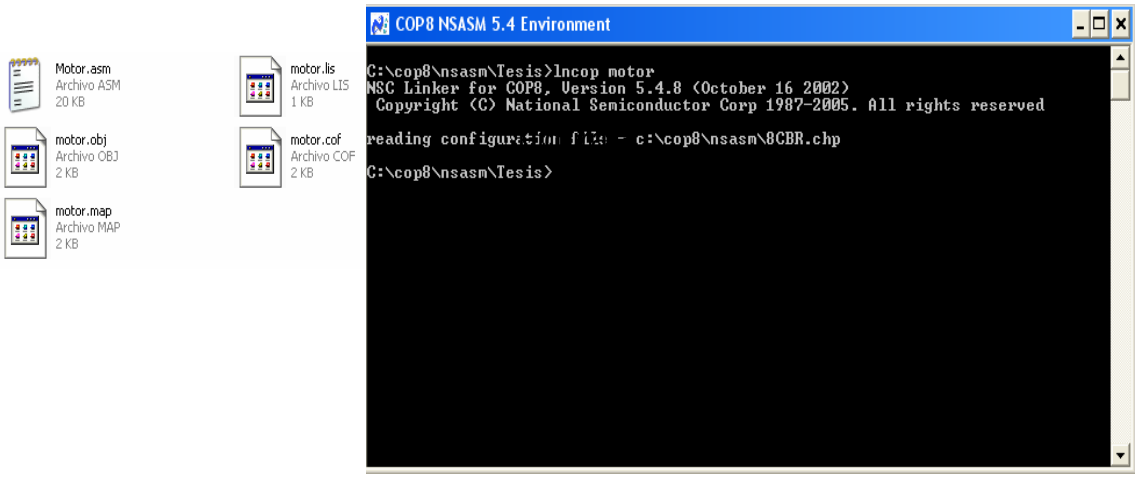


Figura3.

Por ultimo ejecutamos el comando promcop y nos genera el archivo en hexadecimal que nos servirá para la transferirlo a nuestro programador figura 4.

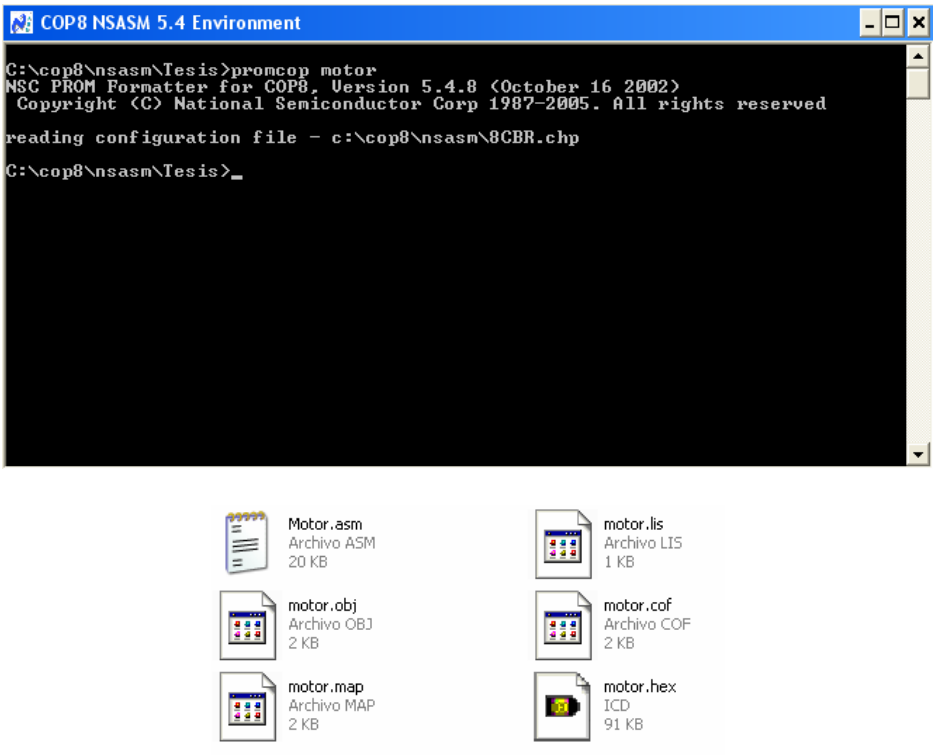


Figura 4.

3.8 Viabilidad De La Propuesta

La Viabilidad son las Condición que hace posible el funcionamiento del sistema, proyecto o idea al que califica, atendiendo a sus características tecnológicas y a las leyes de la naturaleza involucradas.

La **viabilidad técnica** se evalúa ante un determinado requerimiento o idea para determinar si es posible llevarlo a cabo satisfactoriamente y en condiciones de seguridad con la tecnología disponible, verificando factores diversos como resistencia estructural, durabilidad, operatividad, implicaciones energéticas, mecanismos de control, según el campo del que se trate.

Realmente el programador esta al alcance de todos ya que todos los componentes electrónicos son distribuidos en las sucursales y/o establecimientos dentro de la ciudad de México. Los componentes electrónicos que se mencionan continuación cada uno es acompañado por el precio.

Numero	Cantidad	Componente Electronico	Precio Sin IVA	Precio con IVA
1	1	Placa de Cobre	$72,00	$82,80
2	8	Led Emisor de Luz	$0,87	$1,00
3	1	Transformador	$42,61	$49,00
4	4	Diodos 1N4001	$0,35	$0,40
5	16	Resistencias 330 Ohms	$0,44	$0,50
6	2	Motor de Pasos	$50,00	$57,50
7	1	Teclado Matricial	$13,04	$15,00
8	2	DB25 Hembra	$25,00	$28,75
9	2	Driver ULN2003	$5,22	$6,00
10	1	Potenciómetro	$10,44	$12,00
11	1	Cristal de 10Mhz	$6,96	$8,00
12	1	Resistencias 1M Ohms	$0,44	$0,50
13	1	Push Boton	$1,30	$1,50
14	2	Regulador de voltaje 7805	$3,48	$4,00
15	2	Capacitor de 33 pF	$1,30	$1,50
16	1	Resistencia 1 K Ohms	$0,44	$0,50
17	1	LCD	$97,39	$112,00
18	1	Capacitor 0,1 uF	$1,30	$1,50
19	1	Base de 68 Pines PLCC	$13,91	$16,00
20	1	COP8SCR9LVAB	$149,57	$172,00
21	1	Tira de Pines	$2,50	$2,88
22	1	Cable de Puerto Paralelo Macho Macho	$35,00	$40,25
23	1	Cloruro Ferrico 1 Lt	$54,78	$63,00
24	1	Jumper	$0,87	$1,00
25	1	Capacitor de 220 uF	$2,61	$3,00
26	1	Cable puerto paralelo	$30.00	$34.5
Total	**Total de Piezas 55**		**$621,81**	**$715.58**

3.9 Imágenes del programador Propuesto

Presentamos las imágenes del programador para observar como se encuentran conectados los motores de pasos, el LCD, los LED, el teclado matricial.

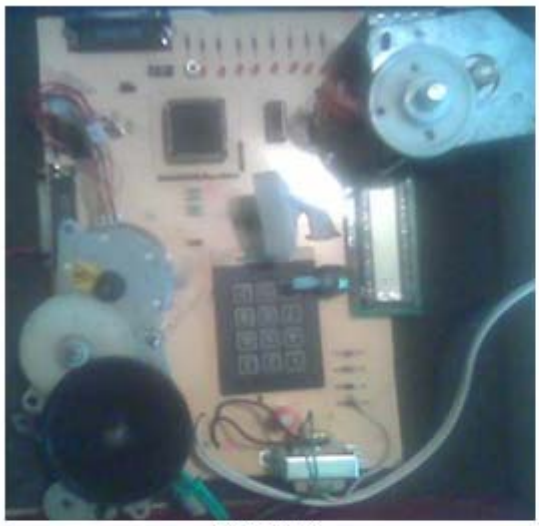


Figura 1

Figura 2

Figura 3

Figura 4

Figura 5

Figura 6

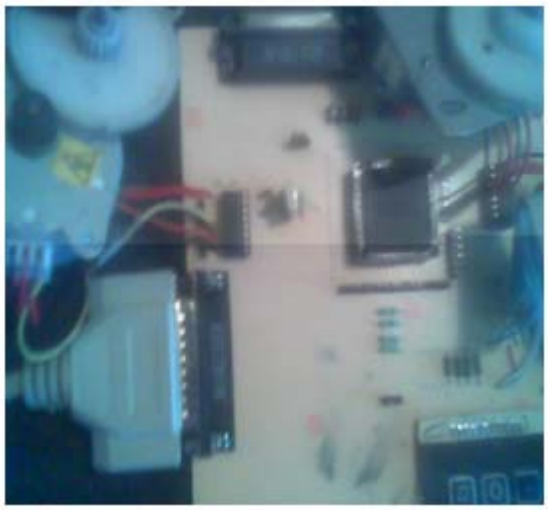


Figura 7

Figura 8

Figura 9

Figura 10

Figura 11

Figura 12

Conclusiones

El propósito de esta tesis es que el alumno se interese en la programación de los microcontroladores ya que hoy en día son muy útiles en la vida cotidiana, y es uno de los circuitos integrados que se pueden explotar para el uso de prototipos, así mismo también se pueden aplicar para el uso de investigación como una herramienta.

Con la realización de este trabajo se pretende que el alumno lo implemente dentro de sus practicas, y cubra las necesidades de los alumnos, así mismo como se ve dentro de la realización el circuito es sencillo y muy económico para su elaboración.

Con el tiempo veremos resultados que los alumnos se enfocaran a la realización de programas y manipulación de prototipos y les permitirá obtener un conocimiento amplio del uso de los microcontroladores.

La propuesta de este circuito puede ser modificado por el alumno ya que es muy enriquecedor el alumno pude adaptarles otros componentes electrónicos para realizar diversos prototipos como son relevadores, fotodiodos, censores, fotorresistencia, etc., pero es muy indispensable que el alumno entienda como esta estructurado y la función que realiza cada parte que integra al microcontrolador.

Glosario

MICROCONTROLADOR: Un Microcontrolador es un circuito integrado el cual, de no ser programado no realizará tarea alguna. Un Microcontrolador es un circuito integrado de muy alta escala de integración el cual contiene tres unidades básicas que lo identifican como tal y son CPU para procesar la información, Memoria de datos para guardar información y Memoria de Programa para almacenar las instrucciones.

MEMORIA DE PROGRAMA: Contiene las instrucciones organizadas en una secuencia particular para realizar una tarea. Típicamente es denominada memoria de sólo lectura (ROM) o también OTP, EPROM o FLASH que son memorias que una vez programadas almacenan la información aunque el sistema no sea energizado. Esto permite que el Microcontrolador ejecute el programa almacenado en Memoria inmediatamente después de ser energizado.

ROM: es una memoria de programa no volátil denominado ROM Flash que permite una programación muy sencilla, rápida y cómoda lo que representa gran facilidad en el desarrollo de diseños.

RAM: Se destina a guardar las variables y datos, es volátil es decir los datos almacenados se borran cuando desaparecen l alimentación.

ALU: (Arithmetic Logic Unic) La Unidad Aritmética Lógica. Esta se encarga de realizar operaciones lógicas y aritméticas que requiere la ejecución del programa con dos operandos, uno que proviene del registro W y otro que proviene de cualquier otro registro o en el propio código de instrucciones.

BUS: En Informática, bus es el conjunto de conductores eléctricos en forma de pistas metálicas impresas sobre la tarjeta madre del computador, por donde circulan las señales que corresponden a los datos binarios del lenguaje máquina con que opera el Microprocesador.

Hay tres clases de buses: Bus de Datos, Bus de Direcciones y Bus de Control. El primero mueve los datos entre los dispositivos del hardware: de Entrada como el Teclado, el Escáner, el Ratón, etc.; de salida como la Impresora, el Monitor o la tarjeta de Sonido; y de Almacenamiento como el Disco Duro, el Diskette o la Memoria-Flash.

DB25:

REGULADOR DE VOLTAJE: Un regulador tiene como función mantener la tensión de salida “Vo” en un valor predeterminado, sobre el rango esperado de corriente de carga, independientemente de las variaciones de la corriente de la carga, la tensión de entrada al regulador Vi y la temperatura T.

LED: (Light Emitting Diode). Pequeño diodo luminoso que se instala en los ordenadores y que se ilumina para indicar que el sistema está encendido o apagado.

PUERTO: Componente por donde se realiza la entrada y salida de datos de un ordenador:
tienes que configurar bien el puerto de la impresora.
SET DE INSTRUCCIONES: Cada Microcontrolador tiene un set de instrucciones. El usuario puede organizar las instrucciones en un orden lógico para crear un programa.

REGISTRO:

USART: Synchronous/Asynchronous Receiver/Transmitter (USART), la cual puede ser utilizada para transmitir y recibir datos de forma serial ya sea sincrónicamente o asincrónicamente.

ASMCOP: Es una instrucción que identifica el ensamblador y lo convierte a un código de maquina. Archivo. asm: Este archivo contiene el programa fuente que ha sido creado por el usuario.

LNCOP: es un acoplador y comunicador, el ASMCOP genera un archivo que conecta y este permite descargar y cargar al simulador COP8

LIBCOP: es un cruz –Librería que lee módulos de objeto producidos por ASMCOP y se combina, Ellos respecto a un archivo que llama a la librería para el uso posterior en otros programas de COP8.

DUMPCOFF: es un programa útil para mostrar el archivo COFF (generado por LNCOP) En un formulario de lectura fácil y amena.

PROMCOP: un archivo útil que nos servirá para que el archivo COFF genere uno o mas archivos que será útil para la programación de la memoria ROM

HEXLM y LMHEX: es útil para convertir - LNCOP - hexadecimales de LMHEX para formato de LM nacional, o LM Formato para Intel-hex. Archivo.hex: De la misma forma que el archivo .COF, este archivo es resultado del ligado del programa fuente pero en formato hexadecimal.

EMULADOR: A diferencia del Simulador, el Emulador es una herramienta basada en hardware que permite la prueba de nuestro circuito con la ventaja de poder visualizar la ejecución de nuestro programa y detenerla en el momento deseado así como la modificación de información en ciertas localidades en memoria y muchas otras cosas más. Todo esto puede ser realizado sin contar con el Microcontrolador.

OPCODE: El Opcode (Operation Code) es un código numérico de la instrucción que representa la operación a ser realizada por el CPU. Es decir, el Opcode es un grupo de bits los cuales le indican al Microcontrolador qué operación en particular hay que realizar. Por ejemplo un 64 podría significar “limpiar el Acumulador”.

MNEMONICOS: Los Mnemonicos son nombres asignados a una operación en particular. Cada mnemonico es asociado con un opcode. El usuario puede hacer referencia a una operación por medio de un mnemonico “ADD” en lugar del Opcode “84”. Con el uso de los mnemonicos se hace más fácil escribir un programa. Por ejemplo, el mnemonico “LD” representa la operación “Cargar”.

LA VIABILIDAD: Son las Condición que hace posible el funcionamiento del sistema, proyecto o idea al que califica, atendiendo a sus características tecnológicas y a las leyes de la naturaleza involucradas.

Apéndice 1

Set de Instrucciones

ADD	A, MemI	Suma	A <- A + MemI
ADC	A, MemI	Suma con Acarreo	A <- A + MemI + C, C <- Carry
SUBC	A, MemI	Resta con Acarreo	A <- A - MemI + C, C <- Carry
AND	A, MemI	Operación Lógica AND	A <- A and MemI
ANDS	A, Imm	Operación AND Imm, Saltar si Cero	Saltar la siguiente si (A and Imm) = 0
Z	A, MemI	Operación Lógica OR	A <- A OR MemI
OR	A, MemI	Operación Lógica OR Exclusiva	A <- A XOR MemI
XOR	A, MemI	SI es Igual	Compara A y MemI, ejecutar siguiente si A=MemI
IFEQ	MD, Imm	SI es Igual	Compara MD y Imm, ejecutar siguiente si MD=Imm
IFEQ	A, MemI	Si NO es Igual	Compara A y MemI, ejecutar siguiente si A≠MemI
IFNE	A, MemI	Si es mayor que	Compara A y MemI, ejecutar siguiente si A>MemI
IFGT	#	Si B NO es igual	Ejecutar siguiente si 4 bits bajos de B No = Imm
IFBNE	Reg	Decrementa Reg, Satar si Cero	Reg <- Reg - 1, saltar siguiente si Reg se hace cero
DRSZ	#, Mem	Poner Bit	1 a bit de Mem (bit = 0 a 7 inmediato)
SBIT	#, Mem	Limpiar Bit	0 a bit de Mem (bit = 0 a 7 inmediato)
RBIT	#, Mem	Si esta puesto un Bit	Si bit Mem es verdadero, ejecutar sig. Instrucción
IFBIT		Limpiar Bandera Pendiente	Limpiar Bandera de Software Interrupt Pending
RPND			
X	A, Mem	Intercambio de A con memoria	A <-> Mem
LD	B, Imm	Carga B con dato Inmediato	B <- Imm
LD	A, MemI	Carga A con memoria	A <- MemI
LD	Mem, Imm	Carga Memoria con dato inmediato	Mem <- Imm
LD	Reg, Imm	Carga Registro con dato inmediato	Reg <- Imm
X	A, [B+]	Intercambio de A con memoria [B]	A <-> [B] (B <- B + 1)
X	A, [X+]	Intercambio de A con memoria [X]	A <-> [X] (X <- X +1)
LD	A, [B+]	Carga A con memoria [B]	A <- [B] (B <- B +1)
LD	A, [X+]	Carga A con memoria [X]	A <- [X] (X <- X +1)
LD	[B+], Imm	Carga memoria [B] con dato inmediato	[B] <- Imm (B <- B +1)
CLRA		Lipiar A	A <- 0
INC	A	Incrementar A	A <- A + 1
DEC	A	Decrementar A	A <- A – 1
LAID		Carga indirecta en A de ROM	A <- ROM(PU, A)
DCOR	A	Corrección decimal en A	A <- BCD corrección (sigue de ADC, SUBC)
RRC	A	Rotar a la derecha con acarreo	C -> A7 -> .... -> A0 -> C, HC <- A0
RLC	A	Rotar a la izquierda con acarreo	C <- A7 <- .... <- A0 <- C, HC <- A3
SWAP	A	Intercambio de nibbles de A	A7...A4 <-> A3...A0
SC		Poner en Alto C	C <- 1
RC		Poner en Bajo C	C <- 0
IFC		Si C	Si C es uno, ejecutar siguiente instrucción
IFNC		Si NO C	Si C no es uno, ejecutar siguiente instrucción
POP	A	Salvar A en la Pila	SP <- SP + 1, A <- [SP]
PUSH	A	Restaurar A de la Pila	[SP] <- A, SP <- SP – 1
VIS		Salto a la Rutina de Servicio Corresp.	PCL <- [VL], PCU <- [VU]
JMPL	Addr.	Salto Absoluto Largo	PC <- ii (ii = 15 bits, 0 a 32K)
JMP	Addr.	Salto Absoluto	PC11...PC0 <- i (I = 12 bits) PC15...PC12 permanecen sin cambio
JP	Disp.	Salto Relativo Corto	PC <- PC + r (r es –31 a +32, no 1)
JSRL	Addr.	Salto Subrutina largo	[SP] <- PL, [SP - 1] <- PU, SP – 2, PC <- ii
JSR	Addr.	Salto Subrutina	[SP] <- PL, [SP - 1] <- PU, SP – 2, PC11..PC0 <- ii
JID		Salto Indirecto	PL <- ROM(PU, A)
RET		Regreso de Subrutina	SP + 2, PL <- [SP], PU <- [SP – 1]
RETS		Regreso de Subrutina y Salto	SP + 2, PL <- [SP], PU <- [SP – 1], Saltar siguiente instrucción
K		Regreso de Interrupción	SP + 2, PL <- [SP], PU <- [SP – 1], GIE <- 1
RETI		Generar una interrupción	[SP] <- PL, [SP - 1] <- PU, SP - 2, PC <- 0FF
INTR		Sin Operación	PC <- PC + 1
NOP			

Instruction Set (Continuo)

JSRL	Addr.	Jump SubRoutine Long	[SP]←PL, [SP–1]←PU,SP–2, PC←ii
JSR	Addr.	Jump SubRoutine	[SP]←PL, [SP–1]←PU,SP–2, PC9…0←i
JSRB	Addr	Jump SubRoutine Boot ROM	[SP]←PL, [SP–1]←PU,SP–2, PL←Addr,PU←00, switch to flash
JID		Jump InDirect	PL←ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL←[SP], PU←[SP–1]
RETSK		RETurn and SKip	SP + 2, PL←[SP],PU←[SP–1], skip next instruction
RETI		RETurn from Interrupt	SP + 2, PL←[SP],PU←[SP–1],GIE←1
INTR		Generate an Interrupt	[SP]←PL, [SP–1]←PU, SP–2, PC←0FF
NOP		No OPeration	PC←PC + 1

Fuentes de Consulta

MICROCONTROLADOR PIC16F84.
Desarrollo de proyectos
Editorial Ra-Ma
Alfa-Omega
Enrique Palacios Municio
Fernando Ramiro Domínguez
Lucas J. López Pérez

http://www.standardics.philips.com/products/hef/pdf/hef40106b.pdf (Hoja de dato de un circuito 40106 puertas Trigger Schmitt)
http://www.redeya.com/electronica/tutoriales/pic1.htm (Historia de los Microcontroladores)
http://www.monografias.com/trabajos12/microco/microco.shtml# (Historia de los Microcontroladores)
http://www.mailxmail.com/curso/informatica/electricidadpc/capitulo11.htm (Historia de los Microcontroladores)
http://lorien.die.upm.es/~juancho/lsed/practicas/laberinto2003/Memoria.pdf (Historia de los Microcontroladores)
http://grupos.unican.es/dyvci/ruizrg/postscript/Componentes/7805.pdf (Historia de los Microcontroladores)
http://formularios.123.cl/traductor/traductor_respuesta.php (Historia de los Microcontroladores)
http://endrino.cnice.mecd.es/~jhem0027/transformador/eltransformador.htm (Historia de los Microcontroladores)
http://www.jaycar.com.au/images_uploaded/ULN2003.PDF (Información del Motor de Pasos)
http://www.ortodoxism.ro/datasheets/nationalsemiconductor/COP8SCR9.pdf (Información del Microcontrolador)
http://www.national.com/nationaledge/may02/files/COP8CBR9_NN.pdf (Información del Microcontrolador)
http://cache.national.com/ds/CO/COP8CBR9.pdf (Información del Microcontrolador)
http://www.national.com/appinfo/mcu/files/ManualDeUsuario.PDF (Información del Microcontrolador)
http://www.national.com/packaging/folders/v68a.html (Información del Microcontrolador)
http://www.national.com/appinfo/mcu/0,3189,747,00.html#tools (Información del Microcontrolador)
http://www.national.com/appinfo/mcu/0,3189,751,00.html#software (Información del Microcontrolador)
http://www.national.com/appinfo/mcu/0,3189,584,00.html#CodeDev (Información del Microcontrolador)
http://www.todorobot.com.ar/proyectos/paralelo/paralelo.htm (Información del Microcontrolador)
http://es.wikipedia.org/wiki/Microcontrolador (Información del Microcontrolador)
http://www.ii.uam.es/~gdrivera/varios/notas_lpt.htm (Información Puerto Paralelo)
http://usuarios.lycos.es/tervenet/TUTORIALES/parallel01.htm(Información Puerto Paralelo)
http://www.modelo.edu.mx/univ/virtech/circuito/paralelo.htm(Información Puerto Paralelo)
http://www.definicion.org/puerto *(Información Puerto Paralelo)*
http://www.national.com/ (Fabricante del Microcontrolador)
http://www.agelectronica.com/ (Presupuesto)