UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

AUTOMATIZACION DEL TRANSPORTE EN APLICACIONES EMPRESA-INDUSTRIA

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA (AREA INGENIERIA ELECTRONICA)

P R E S E N T A N :

MONICA MENDEZ ROBLES
MARCOS ANISLADO TOLENTINO

DIRECTOR DE TESIS: M.I. ALEJANDRO SOSA FUENTES

MEXICO, D. F. MARZO DE 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

D E D I C A T O R I A S

A MIS PADRES: Por su constante empuje por mantenerme en la línea de la superación, especialmente a ti Magui porque eres la mejor madre del mundo y la amiga mas grande del Universo.

A MIS HERMANOS: Blas, Vicente, Martín, Bety, Lety, Paty, y Elvirita por permitirme vivir y convivir con ustedes. Los amo.

A quien ha compartido conmigo momentos de tristeza y felicidad y por regalarme tu voz y tu presencia en los momentos más oscuros del camino: Ma. Eugenia.

A todos aquellos amigos y familiares a quienes estimo, amo y llevo eternamente en el corazón.

A DIOS: Por darme vida para lograr una de mis mas importantes metas en esta vida terrena.

MARCOS.

D E D I C A T O R I A S

Con todo mi amor
A mi hijita: Karen Gabriela
Por ser lo más bello que me
ha ocurrido en la vida, y porque
tu presencia da luz y alegría
a cada segundo de mi vida.

A mi esposo Fermín:
Por todo el amor y paciencia que
me has brindado y porque gozo de
la fortuna de haberte encontrado
en mi camino.

A Cristina:
Porque te quiero y admiro mucho.
Porque has compartido conmigo tu
fortaleza, tu esperanza y optimismo.
Y porque siempre he contado con tu
amor, tu apoyo y confianza.

¡Estoy orgullosa de ser tu hija!

A mis hermanos:

Zoraya: Porque tu madurez e inte
ligencia han sido una
guía para mi.

Rocio: Porque has compartido
conmigo tu sabiduría
y porque me contagias tu
alegría de vivir.

Felipe: Porque he contado con tu
apoyo en momentos impor-
tantes de mi vida.

Porque cada uno de ustedes sabe
lo especial que son para mi. Mu-
chas Gracias por todo el apoyo
que me han dado. Por mi parte se
que mi vida ha sido maravillosa
porque los he tenido a mi lado.

A Dios por permitirme lograr uno más de mis anhelos.

MONICA.

A G R A D E C I M I E N T O

A M.I. ALEJANDRO SOSA FUENTES: Por tus sabias opiniones que nos brindaste de forma incondicional y desinteresada y que motivaron la terminación de este Proyecto.

I N D I C E

1. INTRODUCCION

1.1 Necesidades del transporte de insumos en la industria

1.2 Medios existentes instalados

1.3 Medios existentes en el mercado

1 . INTRODUCCION

El transporte de insumos en una empresa industria es una actividad que tiene una influencia directa en las condiciones económicas de la misma, constituyendo un factor esencial dentro del proceso de producción.
En su concepto mecánico, el transporte significa desplazar personas o bienes de un punto a otro del espacio siguiendo una trayectoria determinada y bajo la acción de fuerzas exteriores.

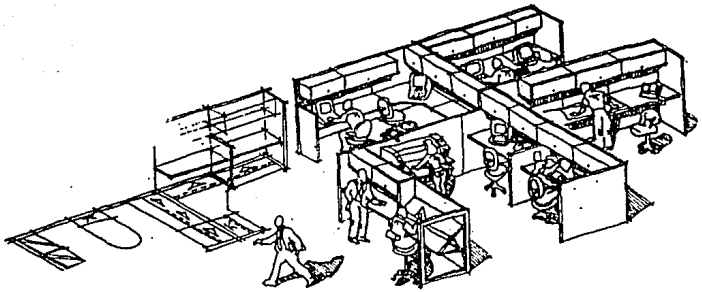
1.1 Necesidades del transporte de insumos en la industria.

Aplicado a la industria, el transporte de insumos interviene formando parte de las diferentes atapas del proceso de producción ya que dentro de las actividades de una empresa industria participa en:

a) Reunir los materiales y servicios requeridos para producir.
En esta etapa se requiere transportar las materias primas del almacen hacia una zona más próxima a las máquinas que las procesarán y transformarán. También se transporta el material de desecho reutilizable que se colocará en esta misma zona. La cantidad diaria de materias primas para el abastecimiento de la planta es de una gran proporción, para lo cuál se debe destinar un porcentaje del costo total del proceso de producción para el transporte local dentro de la empresa o industria.

b) Beneficiar o transformar esos materiales para convertirlos en productos acabados.
En esta parte del proceso, el transporte de insumos participa al ser trasladadas las materias primas, elaboradas en piezas, hacia otras unidades de trabajo dentro de la misma planta para ser ensambladas y/o tratadas de manera tal que se obtenga finalmente un producto terminado.

c) Distribución y venta de los productos acabados.
En esta última etapa se trasladan los productos hacia la zona de control de calidad, haciendose necesario también en esta parte un transporte automatizado local dentro de la planta. Por otra parte para los efectos de distribución a los puntos de venta se utiliza un transporte mecánico.



a) bodegas

b) control de producción

c) administración y gerencia

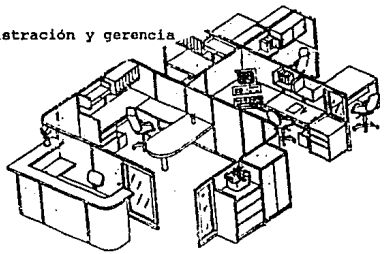


Fig. 1.1 Secciones de una planta productora de plásticos
(cortesía de DUMEX,S.A)

En las etapas mencionadas se puede hablar de un tipo de transporte fijo, entendiendose este, como aquel medio que realiza trayectorias definidas de manera repetitiva. En la tercera etapa se tiene además del transporte fijo otro tipo de transporte denominado transporte variable utilizado para la distribución a los puntos de venta de los productos acabados.

En general, los costos globales de transporte se descomponen en costos de transporte de las materias primas y costos de transporte de los productos, tomando en cuenta los dos principales factores importantes que son:

- Peso de acarrreo
- Distancia del recorrido

Respondiendo a las necesidades de un transporte de insumos se cuenta con alternativas en las cuales es inevitable el esfuerzo del hombre para su desarrollo, teniendose como consecuencia de esta situación:

- Un costo por sueldo en horas-hombre
- Mayor tiempo para ejecutar la actividad del transporte entre otras.

1.2 Medios existentes instalados.

El tipo de alternativas, en su mayoría mecanismos semiautomáticos con los que se cuenta en la actualidad (ver figura 1.2) son:

- planos inclinados
- elevadores
- escaleras rodantes
- teleféricos
- cintas transportadoras
- tubos neumáticos

Todos ellos aplicados comunmente en las diferentes empresas industria instaladas en el país, y de acuerdo al tipo de insumos que se esté manejando el proceso de producción de la misma.

La técnica de automatización tomando el concepto de automatización como toda operación continua e integrada de un sistema de producción que emplea equipo electrónico o de otra índole para regular y coordinar la cantidad y calidad de la producción (1) aún no ha sido utilizada en un gran número de empresas para reemplazar el trabajo realizado por el ser humano a lo largo del proceso de producción. Algunas empresas lo aplican unicamente de manera parcial.

(1) Walter Buckingham : El impacto de la automatización. Ed. Hobbs sudamericanas

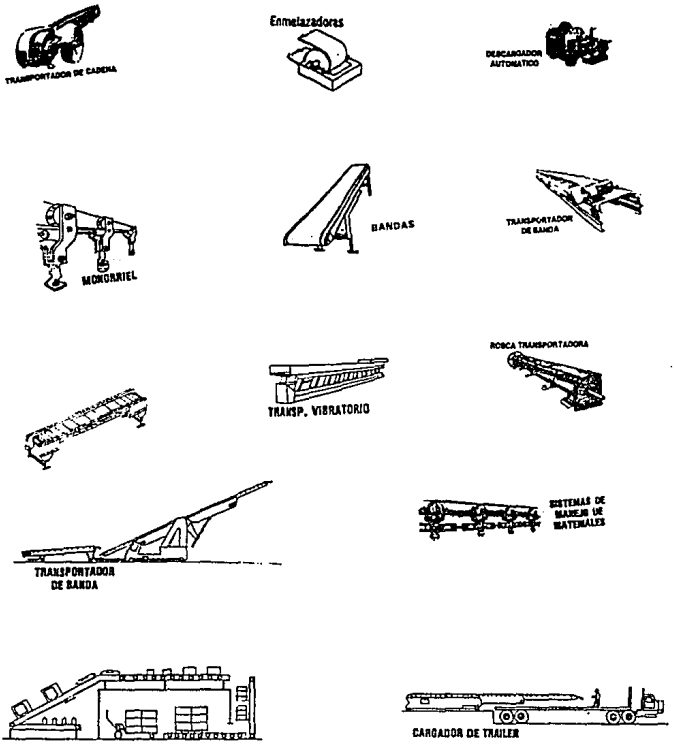


Fig. 1.2 Medios de transporte instalados en la industria

1.3.- Medios existentes en el mercado

Dentro de los equipos que se encuentran disponibles en el mercado se pueden encontrar los siguientes (ver figuras 1.3, 1.4 y 1.5):

- carros carretillas
- patines hidraulicos
- rampas de patio movible
- rampas niveladoras de anden
- elevadores para carga
- mesas
- plataformas
- tractores de arrastre
- triciclos eléctricos
- cargadores de barcos
- desviadores
- elevadores de cangilones
- cargadores frontales
- carretillas eléctricas
- polipastos
- transportadores de cadena
- transportadores de banda
- transportadores helicoloidales
- elevadores de banda y cubiletes
- cribas
- montacargas con motor gasolina
- montacargas con motor diesel
- montacargas eléctricos
- montacargas de pasillo angosto

entre las principales manufactureras de este tipo de equipo de transporte figuran : Mitsubishi, Allis Chalmer, Yale, Clark, etc.

Es importante mencionar que estos sistemas tienen un alto costo para la pequeñas y mediana industria y están limitados a las grandes empresas trasnacionales.

La automatización ofrece la oportunidad de:

- mayor producción
- menos horas de trabajo para la productividad
- creación de empleos especializados en mantenimiento diseño e ingenieria
- producción de nuevos y mejores bienes de calidad uniforme con un empleo más eficiente de las materias primas.

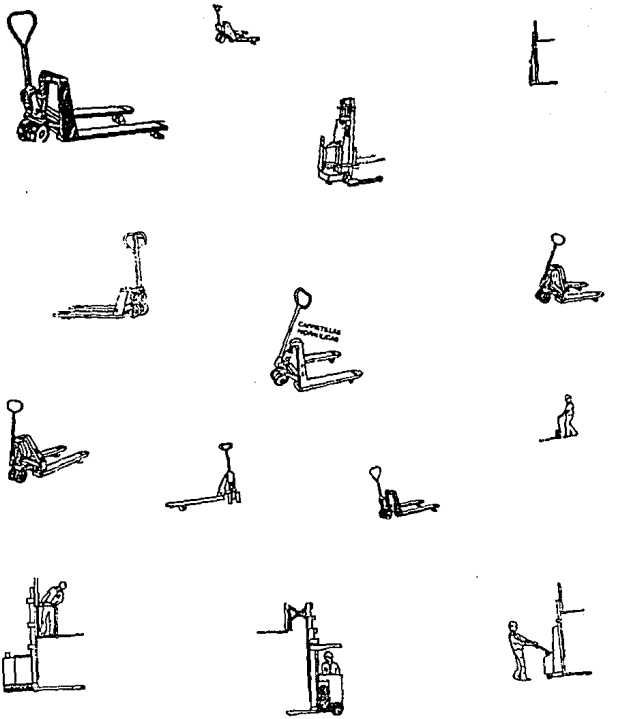


Fig. 1.3 Medios existentes en el mercado (carretillas y patines)

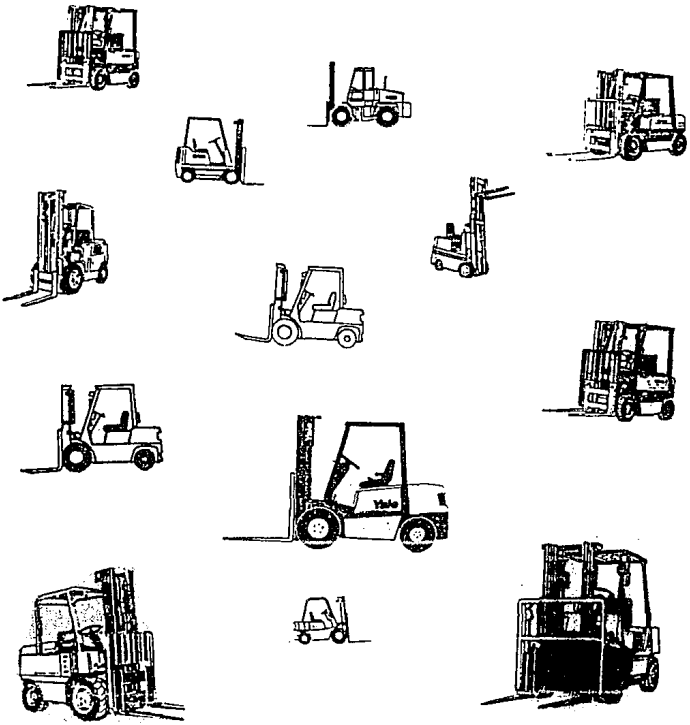


Fig. 1.4 Medios existentes en el mercado (montacargas)

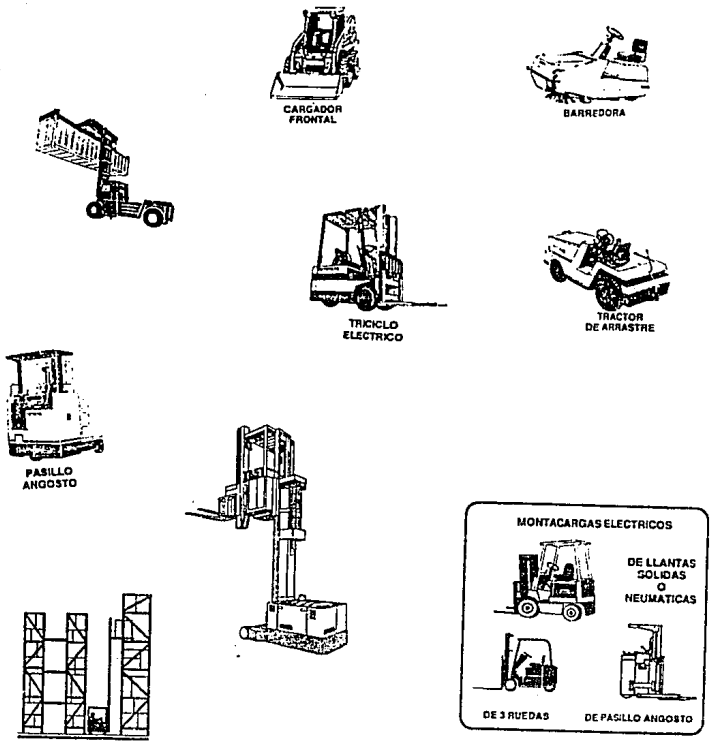


Fig. 1.5 Medios existentes en el mercado (mixtos)

2. OBJETIVO

2. OBJETIVO

Dentro de los equipos empleados en la automatización de empresas industria, el presente proyecto participa como un prototipo de equipo automático, con un sistema digital basado en microprocesador. Este sistema transportará productos acabados o semi-acabados de un lugar a otro en forma autonoma mediante una previa programación de la ruta.

Para el prototipo se contemplarán los siguientes puntos:

- El prototipo estará destinado a transportar insumos en una empresa industria de manera local, esto es, dentro de las mismas instalaciones.
- Tendrá capacidad de almacenar en memoria trayectorias programadas por el usuario.
- Selección y ejecución de alguna de las trayectorias almacenadas el número de veces seleccionado por el usuario
- La longitud que recorran las trayectorias será de acuerdo al local de la empresa industria
- Se contará con un teclado tanto para la programación de las trayectorias como para los comandos de selección y ejecución de la misma.
- El prototipo será un sistema digital basado en un microprocesador de propósito general.

3. ALCANCES DEL PROYECTO

3.1 Medio de operación

3.2 Trayectorias

3.3 Peso de los insumos a transportar

3. ALCANCES DEL PROYECTO

3.1 Medio de operación.

Las condiciones de operación del lugar para el correcto funcionamiento del sistema que se observarán son:

- El terreno no deberá de ofrecer una alta fricción al material de las llantas del dispositivo.
- Evitar en el terreno la presencia de sustancias que afecten la marcha del dispositivo (aceites,grasas,etc).
- Especial cuidado con objetos que obstruyan el paso del dispositivo durante su recorrido (cajas, bolsas, etc).
- La temperatura máxima del lugar será la temperatura máxima que soporten los dispositivos, pero se recomienda utilizar este dispositivo con temperaturas similares a la temperatura ambiental (25 grados centigrados)

3.2 Trayectorias.

- Se contará con un teclado tanto para la grabación de las trayectorias como para los comandos de selección y ejecución de la misma.
- Se podrán almacenar varias trayectorias en memoria de manera que el usuario pueda seleccionar alguna para ser ejecutada.
- Tras haber seleccionado alguna trayectoria, esta se podrá ejecutar un número de veces que será también seleccionado por el usuario.
- Dado que las trayectorias serán almacenadas en memoria, la longitud de estas dependerá de la capacidad de memoria utilizada.

3.3 Peso de insumos a transportar.

Puesto que éste proyecto persigue más que ser un mecánismo de transporte pesado, demostrar que es posible automatizar el transporte mediante la grabación de trayectorias programables, en el desarrollo e implantación del prototipo no se considera el peso de los insumos a transportar.

4. CONSIDERACIONES DEL DISEÑO

4.1 Hardware

4.1.1.- Selección del microprocesador

4.1.2.- Selección de memoria

4.1.3.- Selección de dispositivos periféricos

4.1.4.- Selección de la etapa de potencia

4.2 Software

4.2.1.- Lenguaje de programación

4.2.2.- Equipos de desarrollo

4.2.3.- Software para depuración

4.2.4.- Simuladores

4.2.5.- Metodología de programación

4.3 Mecánico

4.3.1.- Material

4.3.2.- Dimensiones

4. CONSIDERACIONES DEL DISEÑO

Después de tener bien definidos los objetivos y alcances del sistema, el siguiente paso es iniciar el desarrollo del sistema Hardware/Software. Algunos diseñadores utilizan comunmente los diagramas de flujo en esta etapa.

Actualmente, la técnica de diagramas de flujo es de gran ayuda cuando se describe la estructura del sistema y en la explicación del programa para otros usuarios. También son muy utilizados para la documentación, sin embargo pocos son los programas que resultan de un diagrama de flujo muy detallado, dado que el dibujo detallado del diagrama puede tener la misma dificultad que la escritura del programa realizado y es menos utilizado cuando el programa debe ser desarrollado a partir de éste.

Algunas técnicas para el desarrollo de sistemas referentes al Hardware/Software del sistema deben considerar los siguientes puntos principalmente:

* Selección del Microprocesador.
* Realizar una partición adecuada del Hardware y Software.
* Selección de un lenguaje de programación.
* Seleccionar las herramientas necesarias para el desarrollo.

La partición de tareas Hardware/Software deben ser evaluadas en en base al costo y a los aspectos referentes al desarrollo de tecnología o componentes a ser seleccionados.

La partición de Hardware/Software es una de las tareas más delicadas durante el diseño del sistema. La partición debera de ser reevaluada a través del diseño y sus relaciónes deben de ser consideradas cuidadosamente. La partición tiene un gran impacto, en el diseño de software.

Durante la etapa de diseño, la realización de una evaluación debe llevar un seguimiento como el que se muestra en la figura 4.0.1, y la partición que fué hecha originalmente puede ser reconsiderada.

Anteriormente los diseñadores trataban de minimizar el uso de memoria y hardware externo dado que la memoria y el hardware fueron más costosos que el software , actualmente el costo de la memoria y del hardware externo es menor que el costo del desarrollo del software.

Las tareas que habian sido realizadas por hardware actualmente se pueden realizar por software con una reducción de costos de procesadores y memoria. Recientemente existe un hardware adicional que puede substancialmente reducir la complejidad de programación y también se encuentra disponible a un costo relativamente bajo.

El gasto del software es único, dado que se desarrolla una solo vez y posteriormente se realizan copias de el para su distribución . En el caso del hardware el costo es proporcional al número de unidades requeridas. Esta relación entre el hardware y el software difiere actualmente de el pasado cuando el hardware fué más costoso.

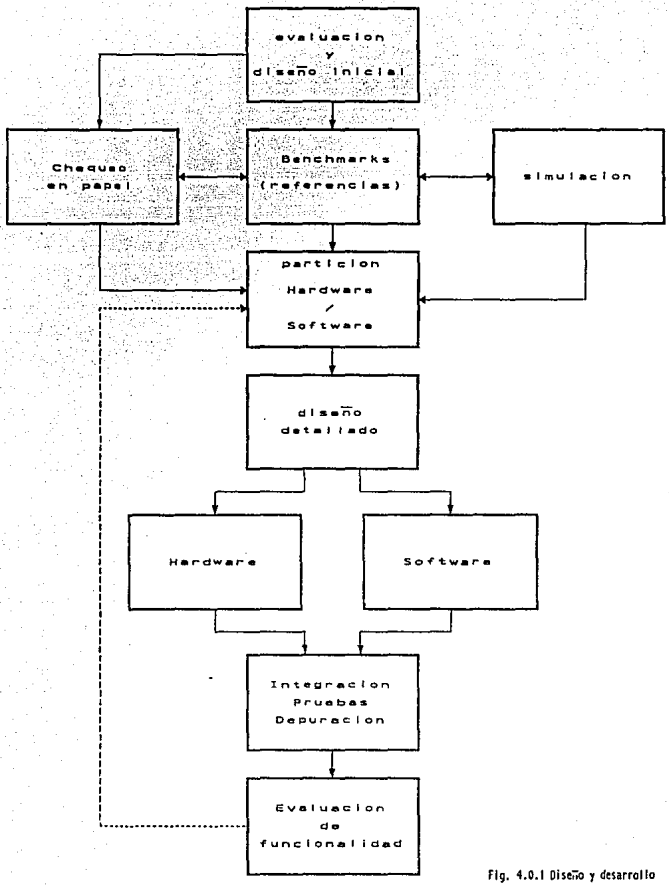


Fig. 4.0.1 Diseño y desarrollo

El tiempo y el costo requerido para el desarrollo del software simpre ha sido un factor muy importante. Los procesadores, memorias y otro tipo de hardware actualmente son menos costosos pero el costo de programación se ha incrementado, éste cambio relativo entre el costo relativo de hardware y software es una gran razón para enfocar la atención sobre las técnicas para desarrollar el software, como es la programación estructurada para incrementar la productividad del programador.

Un diseño, depuración, pruebas y métodos de documentación apropiados pueden reducir en gran cantidad el costo del desarrollo del programa. Escribiendo los programas en lenguajes de alto nivel se puede incrementar la productividad de la programación. Esto puede también simplificar parte del desarrollo, pero también existe una relación entre el costo del hardware y software involucrados.

El programador puede escribir y depurar rapidamente el programa en un lenguaje de alto nivel, pero al final el programa va a requerir más memoria que un programa escrito en lenguaje ensamblador.
Una solución es codificar primero en un lenguaje de alto nivel y editar entonces los ciclos repetitivos para reducir el requerimiento de memoria como se muestra en la figura 4.0.2 :

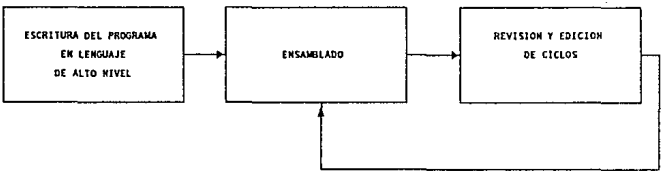


Fig. 4.0.2

La especificación del producto es usada para definir el problema, después los diagramas de bloques son usados para particionar el problema en secciones. El diseño de hardware y software así como la depuración es lo primero que se realiza en el nivel módulo depués en el nivel subsistema y finalmente en el nivel sistema. Siempre deberá verificarse el diseño de la solución en el ambiente de aplicación. Una característica importante de el diseño es que es modular, el nivel básico de los modulos de hardware es el módulo de circuitos. Un procedimiento básico es primero un diseño en papel y posteriormente podrá ser construido un prototipo de esa documentación para pruebas. Cuando esta prueba y depuración es completada totalmente, la documentación original se debe actualizar para la realización de los módulos.

Después de seleccionar el microprocesador y la técnica de desarrollo del sistema los otros componentes que deben ser evaluados son: memoria, entradas/salidas y dispositivos periféricos.

Las memorias del tipo ROM y RAM pueden reducir el número de partes en la memoria del sistema también como el consumo de potencia del sistema, la memoria EPROM es usada siempre durante la fase de desarrollo .

La expandibilidad del sistema es importante para la selección de los componentes del diseño.

Despúes del diseño original de software y hardware éste sistema debe ser combinado y checado juntos. En éste sistema la fase de integración puede presentar problemas en el nuevo circuito de hardware y del software aún no probado. Si ocurre un error en el software puede provocar en ocasiones que el hardware no funcione. La documentación del desarrollo, los circuitos emuladores y un lenguaje de alto nivel puede reducir estos problemas durante la fase de integración. Cuando el sistema está completo la documentación debe ser actualizada, ésta debe de incluir esquemas, diagramas de tiempo, listados de los programas realizados, diagramas de flujo, y procedimientos de mantenimiento.

4.1 Hardware

4.1.1.- Selección del microprocesador

La funcionalidad de las microcomputadoras es frecuentemente una función directa de la velocidad del microprocesador que sea utilizado, sin embargo la velocidad no siempre es el factor más importante en algunas aplicaciones. Esto no es considerado siempre como la más alta prioridad.

Otras consideraciones más importantes pueden incluir el número de líneas de entrada y salida, la capacidad de memoria, el tipo de interfaces para otros tipos de circuitería y de sistemas, así como la capacidad de expansión la cual puede ser tan necesaria según crezca el sistema.

En las aplicaciones, cuando el procesamiento demanda una microcomputadora, multiples microprocesadores pueden compartir ese requerimiento de procesamiento.

La expandibilidad del sistema y el hardware de soporte pueden permitir características a ser adicionadas y problemas de diseño a ser corregidos sin tener que rediseñar el sistema o cambiar a otro mocroprocesador.

La selección del microprocesador involucra la selección de un producto, el cual pueda funcionar eficientemente para la aplicación requerida, otra consideración que puede se incluida, es la disponibilidad de suplir componentes del sistema, así como un equipo de soporte para el desarrollo del sistema.

De acuerdo al tipo de aplicación existen dos tipos de microprocesador: los microprocesadores de propósito general y los microprocesadores de propósito particular.

Un microprocesador de propósito general es un dispositivo que puede ser programado independientemente del tipo de aplicación mientras que un microprocesador de propósito particular ha sido fabricado para un tipo de aplicaciones especificas. La siguiente gráfica que se muestra 4.1.1 señala las principales diferencias entre los dos tipos de microcesador.

A medida de que un microprocesador se hace más especial, su costo se incrementa.

Por otra parte, los microprocesadores se especifican por el numero de bits que maneja su bus de datos, teniendose microprocesadores de 8 bits, de 16 bits y de 32 bits. Para éste parámetro, según la precisión que se requiera en los cálculos que se realicen en la aplicación se selecciona un microprocesador de mayor número de bits.

Para éste proyecto se ha decidido utilizar el microprocesador de propósito general de 8 bits, Z-80 por las siguientes razones:

- Dado que no se realizarán cálculos intensivos, es suficiente un microprocesador que maneje 8 bits en su bus de datos.
- Bajo costo.
- Se encuentra disponible en el mercado.
- Compatibilidad con dispositivos de tecnología TTL, CMOS, etc.

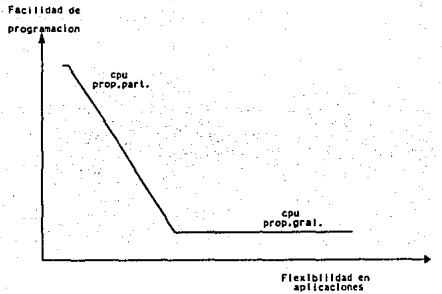


Fig. 4.1.1

4.1.2.- Selección de memoria

Existen en general dos tipos de memoria: memoria RAM y ROM. Las memorias ROM se graban y una vez grabadas no se borra la información almacenada en ellas al suspender el suministro de la fuente de alimentación. En cambio las memorias RAM almacenan la información mientras que se les suministre alimentación, pues al dejar de aplicar la alimentación la información se pierde.

Se utilizará memoria tipo ROM para almacenar el programa del sistema operativo ya que aún sin alimentar el sistema digital se debe conservar dicho programa. Para el almacenamiento de las trayectorias se utilizará memoria RAM puesto que estas deben de poder ser programadas en cualquier momento y no es necesario que sean permanentes.

4.1.3.- Selección de dispositivos periféricos

Se utilizará un teclado con adaptaciones, a partir de un teclado tipo calculadora debido a su tamaño reducido y bajo costo, pero sobre todo por que se adapta perfectamente a los requerimentos de ergonomía del sistema.

Además, se utilizará como interface con el mundo externo el puerto paralelo de la familia Intel, PPI 8255 por su compatibilidad con éste microprocesador ya que es de 8 bits, es bidireccional y posee 3 puertos.

El monitoreo de los estados de operación del sistema se hará mediante leds por su bajo costo, espacio reducido y por que son adecuados para el monitoreo.

4.1.4.- Selección de la etapa de potencia

Debido a que el presente proyecto es un prototipo de lo que puede ser un sistema con aplicación real, la potencia que se pretende manejar para el movimiento del mecánismo no es considerable ya que se utilizará un mecanismo activado por un motor de corriente directa que se alimenta con 3 volts (2 pilas de 1.5 volts en serie).

Se manejará un mecanismo que cuenta con dos pequeños motores, uno para avance y retrocesoy un motor para girar a la izquierda o a la derecha. Por consiguiente se tendran dos etapas de potencia con una configuracion igual. Esta configuracion debe, a partir de una entrada digital de un nivel "0" o "1" lógico proporcionar a su salida la potencia necesaria para activar el motor.

La figura 4.1.2 muestra el bloque de la etapa de potencia, así como sus entradas y salidas.

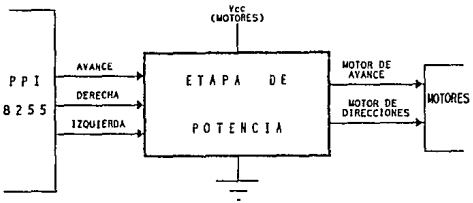


FIG. 4.1.2

4.2 Software

4.2.1. - Lenguaje de programación

El programa para esta aplicación será desarrollado en el lenguaje propio del microprocesador, lenguaje ensamblador Z-80, escrito inicialmente en mnemónicos del lenguaje ensamblador.

4.2.2.- Equipos de desarrollo

Dentro de los laboratorios se cuenta con el KIT del Z80, en el cual se captura el programa para trasladarlo posteriormente a la memoria eprom. También se puede escribir en cualquier paquete de edición de computadora personal (PC's).

4.2.3.- Software para depuración.

Un emulador es probablemente la herramienta de depuración de errores más poderosa con la que puede disponer el diseñador de microcomputadoras. El emulador es un componente de un sistema de desarrollo de microprocesador, se conecta directamente con el microprocesador del circuito prototipo que se va a emular. Esto permite al usuario monitorear y controlar al procesador

En la selección del procesador conviene revisar también la posibilidad de contar con un emulador, pues ofrece gran ayuda en la depuración de errores en el hardware, en el software y en la integración del hardware con el software. Algunas de las características de un emulador son las siguientes:

Puede funcionar a la velocidad total del procesador. En ocasiones el emulador permite utilizar todas las interrupciones del procesador en forma transparente. Se puede contar con capacidad de análisis lógico para el rasteo de la ejecución del programa en tiempo real, para lo cuál se configura o se programa con flexibilidad para puntos de ruptura, adquisición limitada, temporización y conteo. Todas estas características resultan de gran ayuda en la etapa de desarrollo del sistema.

Por otra parte, dentro del software para depuración se tienen algunos compiladores que son capaces de generar código adicional para facilitar la depuración de los programas de usuario. Estos compiladores se basan en el apoyo del sistema operativo de la computadora donde corre el programa. La capacidad de software para depuración son variadas, pero comunmente pueden ser capaces de seguir el flujo y mostrar los registros y la memoria.

4.2.4.-Simuladores.

Un simulador permite la ejecución de un programa de microprocesador sin contar con el hardware del microprocesador, las capacidades de depuración de errores es muy grande de manera que tiene una gran aplicación dentro del desarrollo del sistema.

Es posible encontrar simuladores para varios microprocesadores en grandes computadoras, y pueden encontrarse dentro de una red de tiempo compartido.

En éste sistema se está trabajando con el microprocesador de Zilog Z-80 por lo tanto se utilizará el simulador del Z-80.

Existen varios simuladores del Z-80 para PC's. El que se utilizará para éste desarrollo es el SIMZ80. Éste simulador, a partir de un programa fuente con mnemónicos, creado en cualquier editor, genera el programa de salida para verificar que no existan errores de sintáxis y el programa objeto, que es un programa código que contiene las instrucciones en el lenguaje del microprocesador y que está en hexádecimal.
El paquete SIMZ80 es un software tan completo que realiza las funciones de emulación, simulación y depuración, ofreciendo gran capacacidad para el desarrollo del software en el sistema.

4.2.5.- Metodología de programación

Las técnicas de diseño que son más utilizadas en la escritura de programas son:

1) Programación por módulos: Es una técnica en la que los programas son divididos en pequeños programas o módulos, los cuáles, son diseñados, codificados y depurados separadamente y ligados conjuntamente después.

2) Diseño "Top Down": Aquí se define primero una tarea principal para generar subtareas las cuales son particionadas en tareas completamente definidas, el proceso continúa hasta que todas las subtareas están definidas apropiadamente para comenzar a programarse. El método contrario a esta técnica es el "Botton Up" en él cuál todas las subtareas son primero codificadas y después se integran conjuntamente a un programa principal.

3) Diseño Estructurado: En éste método los programas son escritos de acuerdo a la definición especificada. Solo cierto tipo de lógica de programas puede ser permitida pero algunas rutinas. Pueden ser anidadas con algunas otras para realizar situaciones más complejas. La programación estructurada es siempre escrita en secciones con una sola entrada y una sola salida.

A partir de un diagrama de flujo de datos, la programación será desarrollada siguiendo el método de programación por módulos.

4.3 Mecánico

4.3.1- Material

De acuerdo a los alcances del proyecto, definidos en el punto 3.3, no se considera el peso de los insumos a transportar. Por lo tanto se desea que el mecánismo sea lo más ligero posible y se ha decidido que el mecanismo de transporte sea de plástico. Además se evita en lo posible que las adaptaciones que sufra el mecánismo sean piezas metálicas que incrementen el peso del mismo.

4.3.2.- Dimensiones

El presente proyecto pretende ser una aplicación práctica de la automatización del transporte. Se determinó utilizar un módelo de transporte que representara las características básicas de un medio de transporte real. Otros factores en la busqueda de éste módelo fueron:

- costo reducido
- fácil de adquirir en el mercado
- tamaño adecuado para el montaje de circuitos
- adaptaciones mecánicas mínimas

Las dimensiones del módelo seleccionado son: 10 cm (ancho) x 30 cm (largo) x 10 cm (altura) apróximadamente.

5. DISEÑO DEL PROTOTIPO

5.1 Diagrama de bloques
- 5.1.1.- Unidad Central de Procesamiento
- 5.1.2.- Memoria
  - 5.1.2.1.- Memoria Eprom
  - 5.1.2.2.- Memoria Ram
- 5.1.3.- Entrada/Salida
- 5.1.4.- Software

5.2 Sección de CPU
- 5.2.1.- Arquitectura del Z-80
- 5.2.2.- Señales de Entrada/Salida del Z-80

5.3 Sección de Memoria
- 5.3.1.- Mapa de memoria
- 5.3.2.- Direccionamiento de memoria
- 5.3.3.- Datos a memoria y desde memoria
- 5.3.4.- Ciclos de lectura y escitura
- 5.3.5.- Ejecución de una instrucción del Z-80

5.4 Sección de Entrada/Salida
- 5.4.1.- El PPI 8255
- 5.4.2.- Teclado
- 5.4.3.- Lineas de Salida

5.5 Software
- 5.5.1.- Definición de requerimientos
- 5.5.2.- Diagrama de flujo
- 5.5.3.- Desarrollo del software
- 5.5.4.- Implantación del software (programa)
- 5.5.5.- Grabación en eprom

5. DISEÑO DEL PROTOTIPO

5.1 Diagrama de bloques del sistema

El diagrama de bloques más general del sistema se puede observar en la fig. 5.1.1. En su forma más general, el sistema está constituido por un sistema digital que tiene interface con dispositivos de entrada/salida. Estos dispositivos son:

Un teclado como dispositivo de entrada. Mediante este teclado se darán tanto las instrucciones necesarias para la grabación en memoria, como las instrucciones o comandos para la ejecución de la trayectoria que se seleccione.

Una interface de salida con la etapa de potencia, para que dicha etapa de potencia active los motores, tanto de avance como de dirección.

Un mecánismo con adaptaciones para acoplar el sistema digital al teclado y a la circuiteria asociada.

Lo siguiente es una descripción general del sistema digital. Posteriormente se hará un análisis detallado de cada una de las etapas que lo confirman. La fig. 5.1.2 muestra la configuración del sistema digital en forma de diagrama de bloques.

5.1.1 Unidad Central de procesamiento

El CPU o Unidad Central de Proceso es el principal componente controlador del sistema. Sus funciones son:

-Busqueda y ejecución de instrucciones de memoria
-Almacenar y recuperar datos de memoria
-Almacenar y recuperar datos desde la sección de E/S
-Controlar todas las funciones del sistema.

DIAGRAMA DE BLOQUES DEL SISTEMA

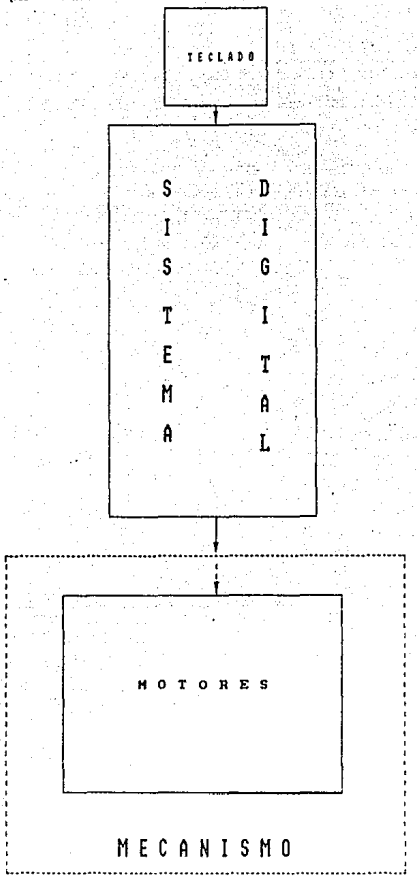


Fig. 5.1.1

DIAGRAMA DE BLOQUES DEL SISTEMA DIGITAL

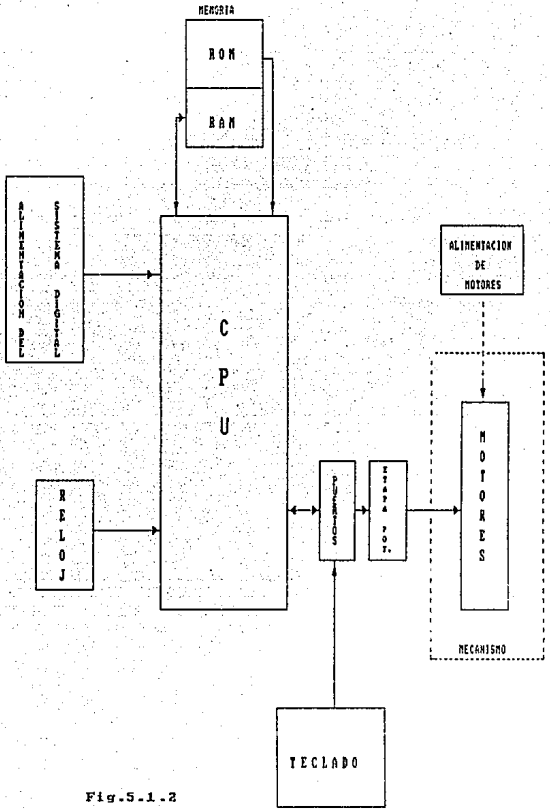


Fig.5.1.2

El CPU usado en este sistema es un microprocesador llamado Z-80 perteneciente a la compañia Zilog, que es el original manufacturero del Z-80. Este microprocesador puede sumar cientos de numeros por segundo y puede ejecutar una variedad de diferentes tipos de instrucciones.

El Set de intrucciones del CPU consta de alrededor de 200 instrucciones diferentes : para sumar 2 números, para restar 2 números, para guardar un número en memoria o para manejar un número del mundo exterior vía dispositivos de E/S.

Estas instrucciones pueden tener muchas aplicaciones diferentes construyendo una secuencia de instrucciones (programas).

El CPU opera a una razón constante llamada frecuencia del reloj. Cada acción en el CPU es particionada en incrementos de esta frecuencia de reloj. El microprocesador Z-80 es capaz de operar en una frecuencia de hasta 4 millones de ciclos por segundo. La frecuencia de reloj que se seleccionó para este sistema es de de 1 millón de ciclos por segundo.

El bloque llamado reloj del sistema es el circuito que genera un millón de ciclos por segundo, es decir, una frecuencia de 1 MHz.

frecuencia =1 ciclo /seg. = T(periodo)/tiempo
frecuencia =[Hertz]

El periodo de la frecuencia de reloj es 1 microseg.
Cada acción en el CPU ocurre en incrementos de la mitad del periodo. Cada acción que el CPU ejecuta varía desde 4 periodos de reloj hasta 20 periodos de reloj con lo que podremos ver que cada instruccion puede tener desde 4 microsegundos hasta 20 microsegundos de duracion.

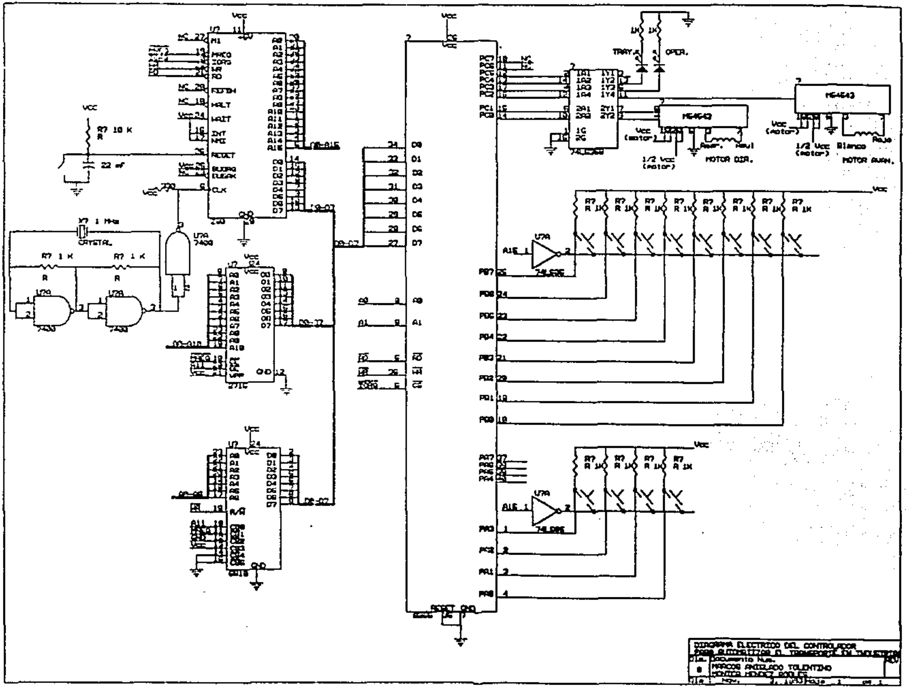


DIAGRAMA ELECTRICO DEL CONTROLADOR
Documento Num.
MARCOS ANICLADO TOLENTINO
Vcc
R? 10 K
22 nF
X? 1 MHz
CRYSTAL
R? 1 K
7400
74LS06
2716
TRAY.
OPER.
M54642
MOTOR DIR.
MOTOR AVAN.
1/2 Vcc (motor)
Vcc (motor)
Blanco
Rojo
Azul
CLK
RESET
HALT
WAIT
INT
NMI
D0-D7
A0-A15
A0-A10

5.1.2 Memoria

El bloque de memoria es otro de los principales componentes del sistema. Toda computadora tiene una memoria para guardar programas y datos. Los programas son secuencias de intrucciones que son ejecutadas. Dato es un término genérico que describe una variedad de tipos de información.

Cada instruccion es codificada como un valor numerico único, por ejemplo, una instruccion con código 09h realiza una operación, mientras una instrucción con código 67h realiza otra operación distinta.

Todas las instrucciones y datos son almacenados en segementos de memoria llamados bytes. Un byte es una colección de 8 bits. Bit es una contracción del término binary digit, siendo un bit un indicador de un "1" o de un "0" lógico.

Para manejar la memoria del sistema, se cuenta con dos chips. Estos chips tienen dos funciones que son: Almacenamiento de datos y recuperacion de datos.
Uno de los chips es la memoria EPROM, 2716, mientras que el otro chip es es la RAM 6810.

5.1.2.1 Memoria Eprom

La arquitectura de la 2716 EPROM es mostrada en la fig.5.1.3 Esta constituida por 2048 localidades, de la 0 a la 2047.

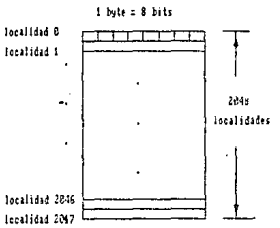


fig. 5.1.3 arquitectura EPROM

El CPU recupera el dato de un byte una vez que le ha enviado a la memoria una dirección a travez de las 16 líneas de direccion. La dirección es una colección de 16 bits, uno por cada línea. Para retomar el dato almacenado en una localidad EPROM, el CPU coloca ese valor en las líneas de dirección como se muestra en la figura 5.1.4. Al mismo instante que el CPU pone la dirección de la localidad, toma el dato de esa localidad de las 8 líneas de dato que se encuentran conectadas a la memoria. Las líneas de dato atrapan los 8 bits de datos de la memoria EPROM de la localidad que habia sido direccionada.

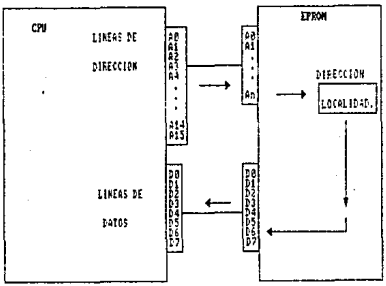


FIG.5.1.4 operacion EPROM

En el transcurso de la ejecucion del programa, la EPROM es direccionada para nuevos datos e instrucciones.

En este sistema, la EPROM toma juntos al programa dividido en secciones de 8 bits y a los datos constantes del mismo.

La EPROM es una memoria de sólo lectura, eso significa que el dato puede ser leido desde la memoria, pero nada puede ser escrito a la memoria.

La memoria EPROM es usada en el sistema para almacenar el programa y leer los datos constantes (todos aquellos que nunca cambiaran).

Para escribir datos en la EPROM es necesario remover la EPROM del sistema, asegurarse que no contenga datos u otro programa que no sea el de esta aplicación, y si es necesario borrarla mediante la exposicion de rayos ultravioleta, grabar el programa localidad por localidad o bien mediante un programa llamado simulador (por ejemplo UNIPRO de XelTek, que es un grabador universal de memoria eprom y que tiene interface con PCs) y regresar la EPROM al sistema con el programa y datos correctos.

5.1.2.1 Memoria Ram

La memoria RAM es usada en el sistema como una memoria de lectura/escritura.Los datos pueden ser conjuntamente escritos y leidos de la RAM. El número de localidades de la RAM es de 128. Este tamaño ha sido dictado por dos factores que son: el costo y el almacenamiento del sistema. El tamaño de la RAM podría haber sido más grande, pero 128 bytes de RAM conceden una cantidad adecuada de almacenamiento para esta aplicacion.

La arquitectura usada de la 6810 en el sistema es la mostrada en la figura 5.1.5

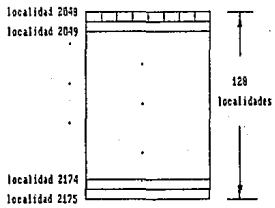


fig. 5.1.5 arquitectura RAM

Las direcciónes de las 128 localidades van de la 2048 a la 2175 (en binario de la 0000 1000 0000 0000 a la 0000 1000 0111 1111).

El CPU direcciona a la RAM colocando una dirección de 16 bits en este rango sobre la línea de direcciónes y entonces lee un dato de 8 bits de la RAM o escribe un dato de 8 bits a la RAM, vía las líneas de control.

Los datos son continuamente transferidos entre el CPU y la memoria RAM. La RAM es usada para tomar datos temporalmente, datos que son usados por el programa de la aplicación.

5.1.3 Entrada/Salida

Refiriendose a la fig.5.1.2, se puede observar otro elemento dentro un sistema de microcomputadora típico. La sección de entrada/salida o e/s, es el componente del sistema que permite la comunicación con el mundo exterior. En este sistema, el mundo exterior se conecta a travez de 24 líneas del chip de interface periférica programable PPI 8255. Las 24 líneas del PPI representan 24 bits de datos binarios. De las 24 líneas, se ocupan 12 para el teclado, 2 para los leds y 3 para la etapa de potencia para los motores.

El PPI es una interface e/s que es direccionada en forma similar a una localidad de memoria. El cpu puede enviar un dato de un byte hacia el PPI y el PPi entonces lo ruteará a el conjunto de líneas apropiado para encender un led o bién para activar motores a travez de la etapa de potencia. El cpu puede también leer un byte de datos representando el estado (1 o 0) de alguna de las líneas del teclado.

5.1.4 Software

Otro elemento mostrado en el diagrama de bloques del sistema digital es el Software. El sistema tiene una gran flexibilidad debido a que el software puede ser cambiado, ofreciendo una variedad de modificaciones en el sistema, como puede ser incluir más trayectorias a grabar y ejecutar, adición de leds indicadores, para un monitoreo más completo del sistema, etc. Más adelante se hablará del software y la programación del sistema digital.

5.2 SECCION CPU.

En esta sección se discutirá el CPU, que en este caso es el Z-80, el cuál es extremadamente popular en todo tipo de diseños que incluyen un microprocesador de propósito particular. El Z-80 requiere un mínimo de circuitería de soporte, esto significa que una microcomputadora puede ser implementada utilizando unos cuantos componentes adicionales al Z-80. Se describirá su estructura interna, las señales de salida o que se generan dentro del Z-80 y que van al exterior, y los ciclos de procesamiento del Z-80.

5.2.1 Arquitectura del Z-80

La figura 5.2.1 es un diagrama de bloques de la lógica interna del procesador Z-80. Este procesador tiene un registro de instrucciones, un decodificador de instrucciones, una unidad aritmética lógica, un arreglo de registros, buffers, y lógica de control.

Una vista expandida del arreglo de registros del Z-80 se encuentra en la fig.5.2.2.

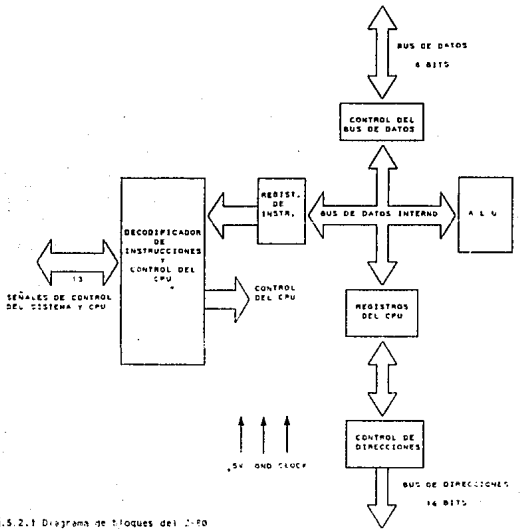


Fig.5.2.1 Diagrama de bloques del Z-80

El Z-80 tiene dos conjuntos de registros con 12 registros de propósito general. Los registros de el conjunto principal están designados con las letras B, C, D, E, H, y L. Un registro en el conjunto alterno es designado para cada uno de los registros, con un apostrofe después de la letra del registro. El acumulador y registros de banderas están agrupados con los arreglos de registros. El tener un número de registros mayor que el de otros procesadores, p.e., el 8080 y 8085 da al Z-80 una mayor flexibilidad; sin embargo, sólo un conjunto de registros puede estar áctivo al mismo tiempo. El Z-80 tiene instrucciones especiales para cambiar de un conjunto de registros a otro.

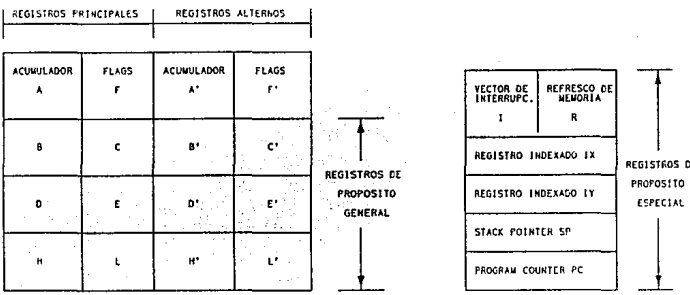


Fig. 5.2.2 Registros del Z-80

Además de los registros de propósito general, el Z-80 cuenta con dos registros de propósito especial. Estos son los registros de índice, IX e IY. Los registros de índice son muy usados en aquellas aplicaciones que involucran manipulacion de tablas de datos.

Adicionalmente, éstos registros expanden el número de modos de direccionamiento disponibles para el Z-80.

Los registros de 16 bits son dos registros de 8 bits. Los registros para refresco de memoria pueden ser utilizados para soportar memoria dinámica. El registro del vector de interrupción es parte del sistema de interrupciones del Z-80. Se combina con los flip-flop de interrupción y los flip-flop de modo de interrupción para determinar como el Z-80 manejará las interrupciones.

Las operaciones aritméticas y lógicas afectan a la banderas de signo S, de cero Z, de no cero N, de acarreo C, y a la bandera de paridad P/V. Registro de banderas, flags:

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	H	X	P/V	N	C

5.2.2 Señales de entrada y salida del Z-80

A continuación se estudian las señales de entrada y las señales de salida del microprocesador Z-80. La fig. 5.2.3 muestra como está conectado el Z-80 en el sistema.

Las entrada de fuente de alimentación del Z-80, están a +5 volts de dc y tierra (pins 11 y 29 respectivamente). Todas las señales de entrada y salida son compatibles con TTL, lo cuál significa que un 0 lógico es apróximadamente 0 volts y un 1 lógico es apróximadamente de 3 a 5 volts. La excepción a ésto es la entrada de reloj del pin 6 que debe de ser conectada en "pull-up" con una resistencia de 330 ohms como se muestra.

La entrada de reloj es una onda cuadrada que aparece como se muestra en la fig. 5.2.3. La frecuencia de la onda cuadrada es de 1 megahertz. Esta frecuencia fué escogida para dar suficiente tolerancia al tipo de circuito seleccionado. El CPU Z-80 acepta señales de reloj compatibles con TTL. La frecuencia generada por el circuito de la figura es controlada por el cristal. El Z-80 construye ciclos de máquina basados en la frecuencia del reloj. Tres o más estados de reloj son utilizados para construir ciclos de máquina, los cuáles son empleados para controlar el flujo de información a travez del bus de datos. La salida del circuito de reloj es amplificada en potencia mediante un inversor (74LS05) tipo "open collector" que permite a la onda cuadrada aproximarse a 5 volts y poder cumplir los requerimentos de las especificaciones del reloj del Z-80.

Cada instruccion ejecutada dentro del Z-80 está dividida en ciclos T, los cuales son esencialmente ciclos de reloj. Cada ciclo T es de un microsegundo de duración. El número de ciclos para ejecutar una instrucción varia de 4 a 23, ejecutandose la instrucción en un tiempo de 4 a 23 microsegundos.

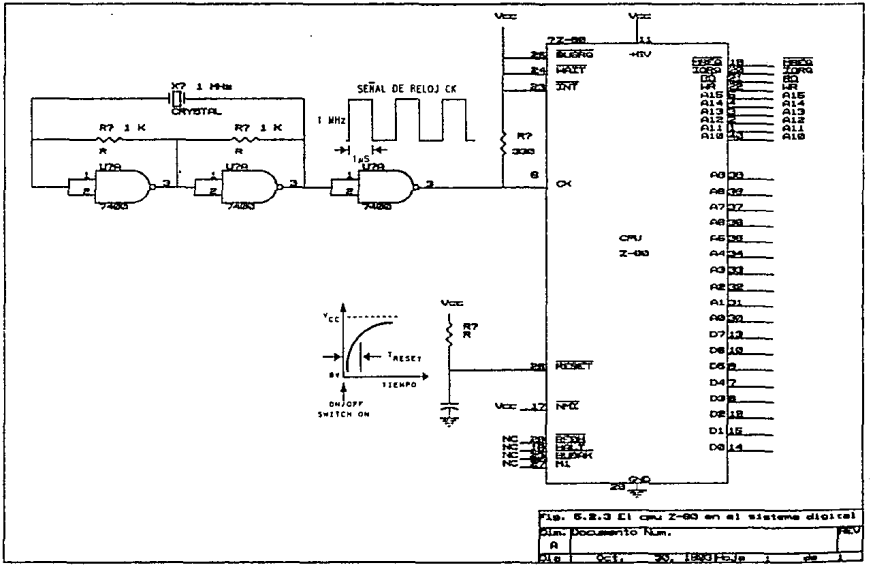
Hay 2 buses principales en el Z-80: El bus de datos y el bus de direcciones. Un bus es simplemente una colección de líneas. Las líneas del bus de datos son D0 a D7, siendo D7 el bit más significativo. Todos los datos que pasan entre el cpu y memoria, y entre el cpu y dispositivos entrada/salida son transferidos a travez del bus de datos. Esto incluye todos los bytes para instrucciones leidos desde memoria, todos los operandos que son transferidos entre cpu y memoria, y todos los datos de E/S que van y vienen del exterior. El bus de datos está orientado a byte, y todos los datos son transferidos un byte por vez.

El bus de direcciones es el segundo bus del Z-80. Está constituido por 16 líneas, de la A0 a la A15, siendo A15 la más significativa. El bus de direcciones es usado para direccionar memoria para busqueda de bytes de instrucciones y datos de operandos.

Asociado con la lectura de datos de memoria, está la señal RD. RD sirve para que la memoria sepa que un dato es leido o escrito a memoria. Entonces, todos los accesos de memoria son leidos. Cuando la señal es un cero, una lectura es indicada.

Otra señal, MREQ, requerimento de memoria, es un cero sólo cuando una lectura o escritura de memoria será realizada.

La señal WR es activa (0) cuando una escritura a alguna localidad de memoria está siendo hecha.


X? 1 MHz
CRYSTAL
R? 1 K
R? 1 K
R
R
U?A
U?B
U?C
7400
7400
7400
SEÑAL DE RELOJ CK
1 MHz
1uS
R?
330
Vcc
Z-80
BUSRQ
WAIT
INT
CK
CPU
Z-80
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
D0
D1
D2
D3
D4
D5
D6
D7
RESET
R?
330
Vcc
TRESET
TIEMPO
ON/OFF
SWITCH ON
Fig. 6.2.3 El cpu Z-80 en el sistema digital
Dim. Documento Num.
REV
A
Oct. 30, 1990 Hoja 1 de 1

La última señal de control asociada con memoria y operaciones de E/S es IORQ. IORQ se utiliza para indicar requerimientos de E/S y es una señal de lógica externa. En el caso de éste sistema, las operaciones de E/S transferirán datos entre un registro del cpu y la interface periférica programable, PPI 8255. IORQ está activa sólo cuando una instrucción de E/S es executada. Cuando una instrucción IN (entrada del PPI al cpu), un dato de 8 bits es leido del PPI al registro A del cpu y la señal IORQ está activa. El PPI utiliza las lineas de dirección A0 y A1 y transfiere el dato por el bus de datos. Cuando una instrucción OUT es realizada, un dato de 8 bits es transferido del registro A del cpu al PPI.

Leer y escribir a memoria y E/S del cpu, es entonces, manejado poniendo la dirección adecuada en el bus de datos, activando las señales de control apropiadas de RD o WR y de IORQ y MREQ, y transferir el dato a travez del bus de datos.

El Z-80 incluye además otras señales que no son usadas en este sistema. RFSH indica que el registro de refresh está disponible en las lineas de dirección. Como no se utiliza memoria dinámica, está señal no es requerida. M1 indica que un ciclo de busqueda de una instrucción esta siendo ejecutado en el cpu. Esta señal no es requerida en una configuración pequeña como ésta. BUSRQ y BUSAK (Bus Request y Bus Acknowledge) son usadas para transferir datos entre dispositivos de e/s y memoria sin pasar por el cpu, en un esquema de acceso directo a memoria (dma). Esto es usualmente utilizado es sistemas de larga escala y dispositivos de e/s de alta velocidad. HALT indica que una instrucción halt (detención del sistema) ha ocurrido. UNa instrucción halt puede ser usada para interrupciones, pero no se utilizará en este sistema. WAIT es una entrada que permite al Z-80 ser usado con memorias (o dispositivos de E/S) que operan mucho más lentamente que el cpu. INT es una entrada que indica que una interrupción externa ha ocurrido. La señal NMI, o interrupción no mascarable tiene mayor prioridad sobre la señal INT. En el sistema no se utiliza ninguna de las dos señales de interrupción debido a que no se considera que dispositivos externos interrumpan al sistema.

La señal de RESET es una señal de entrada al Z-80 que indica que la fuente de alimentación del Z-80 ha sido conectada, o que el cpu debería reestablecerse. El reset causa que los registros del cpu sean inicializados, y, después de un corto tiempo, se inicie la ejecución desde la localidad 0. Otras ciertas funciones son también inicializadas cuando la entrada de RESET se activa. Una entrada de RESET es obviamente necesaria para permitir que el cpu inicie desde un punto conocido. Cuando la fuente de alimentación es conectada, el voltaje en el pin 26 (RESET) es cerca de 0. Aumenta gradualmente hasta que el capacitor llega a cargarse completamente. El voltaje para éste tiempo se muestra en la fig. 5.2.3. Este esquema es usado para iniciar a la vez que el voltaje está en un valor estable.

5.3 SECCION DE MEMORIA

En este punto se detalla la interacción entre la memoria y el cpu. El sistema digital usa dos tipos de memoria: una EPROM 2716 y una RAM 6810. La secuencia para lectura y escritura de memoria es discutida aquí, con los requerimientos especiales para borrar y programar la EPROM.

5.3.1 Mapa de Memoria

En el punto 5.2 se habló del bus de direcciones del Z-80. Como el bus de direcciones tiene 16 líneas, pueden ser direccionadas hasta 65,536 localidades de memoria (sin esquemas especiales de bancos de memoria para cambiar de un banco a otro), con lo cuál se podrían tener en promedio hasta 30,000 instrucciones en memoria.

Uno de los principales criterios de diseño del sistema fué un costo razonable. Para reducir el número de componentes utilizados, la memoria fué limitada a 2048 bytes para el programay otros 128 bytes para almacenamiento de datos. A pesar de que la memoria EPROM 2716 es de 2048 localidades, resulta demasiado grande para el programa, pero debido a que es de un bajo costo, además de ser de fácil adquisición en el mercado, se optó por incluirla en el diseño del proyecto. Otro criterio que se tomó, fué que se previó tener memoria EPROM reservada para un posible desarrollo posterior de programación que incluya nuevas rutinas.

El mapa de memoria del sistema se muestra en la figura 5.3.1. Las localidades no usadas estan en areas sombreadas. Las localiades de memoria de la 0 a la 2047 son las localidades de memoria del chip 2716. Las localidaes de la 2048 a la 2175 son las localidaes de memoria del chip 6810. El resto de las localidades no son utilizadas.

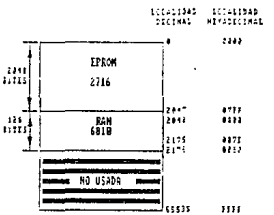


Fig. 5.3.1 Mapa de Memoria

Las localidades designadas están escritas en décimal y hexadécimal. El formato hexadecimal es una forma abreviada de escribir números binarios. En lugar de escribir 0000 0111 1111 1111 para la localidad 2047, se utiliza el hexadecimal 07FF. Para convertir de binario a hexadecima, se agrupa el número en binario en grupos de 4 bits, y cada grupo de 4 bits es cambiado a un digito hexadecimal. Valores hexadecimales pueden ser cambiados fácilmemte con el proceso inverso. El formato más comunmente utilizado en sistemas digitales es el hexadecimal.

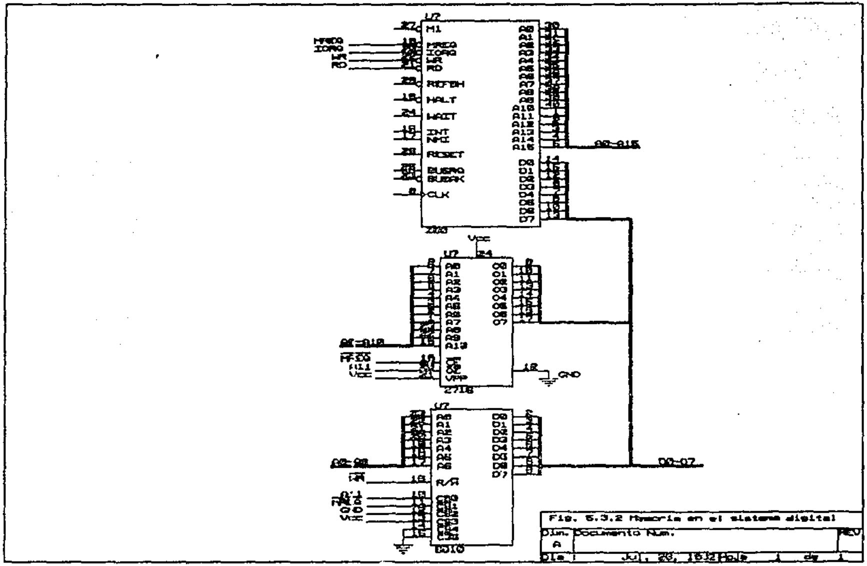


Fig. 5.3.2 Memoria en el sistema digital

5.3.2 Direccionamiento de Memoria

Los chips de memoria son direccionados por el cpu. La figura 5.3.2 muestra las conexiones del cpu y la memoria. Considerando primeramente la EPROM 2716, ésta tiene 11 entradas de dirección, de la A0 a la A10. Esto permite direccionar de la localidad 0000 0000 0000 0000 a la 0111 1111 1111 1111 (hexadecimal 0 a 7FF), 2048 localidades. Para ello se conectan las lineas A0 a A10 de la memoria al bus de direcciones del cpu. Por otra parte, la RAM 6810 utiliza las lineas A0 a A6 del bus de direcciones del cpu para direccionar las 128 localidades, de la 2048 a la 2175 (hexadecimal 0800 a 087F). Entonces para la RAM se conectan las lineas A0 a A6 de la RAM al bus de direcciones del cpu.

Esta conexión parecería ser suficiente, sin embargo esto causaría conflictos porque el cpu al poner en el bus de direcciones una dirección, ambas memorias entenderían que están siendo direccionadas. Para evitar ésta situación se necesitan lineas de dirección adicionales para seleccionar solamente una de las dos memorias. El mapa de memoria define el area de la EPROM de la localidad 0 a la 7FF y el area RAM de la localidad 800 a la 87F. Atendiendo a la figura 5.3.3, la linea de direción A11 nunca es un 1 para el area de direcciones de la EPROM y siempre es un 0 para el area de la RAM. Entonces, éste hecho se utiliza para la selección entre EPROM o RAM.

Esta selección se realiza conectando A11 al pin 10 de la RAM 6810. El pin 10 es el chip select o CS0 del chip 6810. La 6810 tiene otras 5 entradas de selección: CS1, CS2, CS3, CS4, CS5. Cuatro de ellas deben ser cero para que el chip sea seleccionado: CS1, CS2, CS4, CS5. La otra, CS3 debe ser un 1.

		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
E																	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
R	.									.							
	.									.							
O	.									.							
	07FF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
M																	
R	0800	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0801	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
	.									.							
A	.									.							
	.									.							
	087E	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	0
M	087F	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1

A11 = 0 para EPROM
1 para RAM

Fig. 5.3.3 Selección de EPROM o RAM

Para ésto se conectan los pin CS1, CS2, CS4, y CS5 de la RAM a tierra y el pin CS3 a Vcc.

Por que el bus de direcciones es usado no solamente por la memoria, sino también para direcciones de e/s, por lo que es necesario manejar un selector de memoria o dispositivo de e/s adicional. La señal MREQ del cpu se activa (es un cero) siempre que una localidad de memoria está siendo escrita o leída. Para diferenciar entre una dirección de e/s y una dirección de memoria, MREQ es usada para seleccionar la EPROM (pin 18) y la RAM (pin 11).

5.3.3 Datos a Memoria y desde Memoria

El bus de datos, D0 a D7, es usado para transferir datos leídos desde la memoria dentro del cpu. Es utilizado también para escribir datos dentro de la RAM 6810. Por supuesto, no pueden ser escritos datos dentro de la EPROM 2716. El bus de datos es un bus bidireccional, los datos fluyen en ambas direcciones, desde el cpu y hacía el cpu. El cpu sabe siempre la dirección del flujo. Durante una busqueda de instrucción, el dato es leído de la memoria y transferido dentro del cpu para decodificar la instrucción. Una vez que el cpu ha leído los uno a cuatro bytes de la instrucción, va dentro de la porción de execución de la instrucción y puede leer o escribir un (o dos) byte hacia o desde la memoria. Porque el bus de datos es bidireccional, el cpu genera una señal de lectura (RD) y una señal de escritura (WR) para indicar a la memoria o dispositivos de e/s cuando está ocurriendo una lectura o escritura, según sea el caso. Las señales RD y WR, entonces, son otros selectores que deben ser considerados dentro de la lógica de memoria.

La señal WR del cpu se conecta al pin 16 de la RAM 6810. Las especificaciones de la 6810 son tales que, para una operación de escritura de memoria, ésta señal es un cero, y para una operación de lectura de memoria es un uno.

La señal WR no es usada para la EPROM 2716. La razón de esto, es que cada vez que la EPROM es direccionada, la operación debe ser una lectura, mientras que una escritura sería imposible. Por lo tanto, para la EPROM no se conecta la señal WR ni la señal RD.

5.3.4 Ciclos de Lectura y Escritura

A continuación se estudia como se realizan las operaciones de lectura y escritura por el cpu Z-80. La fig.5.3.4 muestra la operación de lectura durante la busqueda del primer o dos primeros bytes de la instrucción. Cada perído T tiene una duración de 1 microsegundo. La primera acción que toma el cpu es poner la dirección (del program counter, PC) dentro de las lineas del bus de direcciones A15-A0. (la señal M1 indica que ésta es la busqueda del código de operación). Entonces, el cpu pone las señales MREQ y RD en cero, para indicar a los dispositivos externos que una lectura de memoria esta siendo hecha. En este punto, la memoria que está siendo direccionada tiene cerca de 1.5 estados T para responder, o cerca de 1.5 microsegundos.

La memoria responde como sigue: sabe por la señal MREQ que el cpu desea leer o escribir un dato vía el bus de datos. Si la memoria es la RAM 6810, sabe por el estado de la señal WR que es una escritura (WR = 0) o lectura (WR = 1) la operación que se requiere. Si A11 es un 1 (si el cpu está leyendo un byte del area 2048 a 2175), la señal CS0 de la RAM es un 1 y la RAM seleccionada. Si A11 es un 0, la señal OE de la EPROM se activa y la EPROM seleccionada.

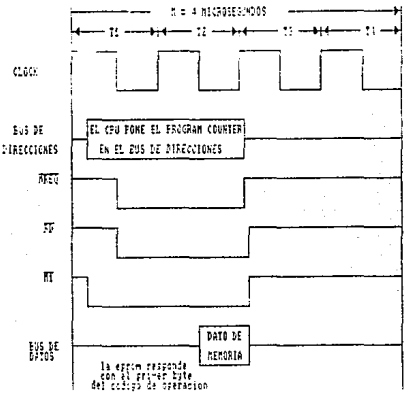


Fig. 5.3.4 Ciclo de busqueda

En el caso general, si es seleccionada memoria (MREQ = 0 y A11 = 1 para la 6810 y A11 = 0 para la 2716), ve la dirección presente en el bus de datos. Si la RAM 6810 o la EPROM 2716 es seleccionada para una lectura, los 8 bits en la localidad de memoria que es especificada en las lineas de dirección son puestos en las lineas de datos D0 - D7. Al final de T2 el cpu lee los bits de las lineas de datos dentro de un registro interno para decodificarlos.

El escribir a memoria es realizado solamente durante la ejecución de una instrucción, de manera que la busqueda del codigo de operación no es considerado durante un ciclo de escritura. Un ciclo de escritura comienza de manera similar al ciclo de lectura, como se muestra en la figura 5.3.5 . La dirección de la localidad de memoria donde se escribirá es puesta en el bus de direcciones A0 - A15. La señal MREQ es activada (es puesta en cero). Inmediatamente, el cpu manda el dato a ser escrito en memoria hacia el bus de datos D0 - D7. Al mismo tiempo la memoria comienza a hacer realmente una lectura de la localidad especificada. La señal WR es un cero, y el dato es escrito dentro de la localidad especificada, enseguida que el chip de la memoria detecta la transición de WR de uno a cero.

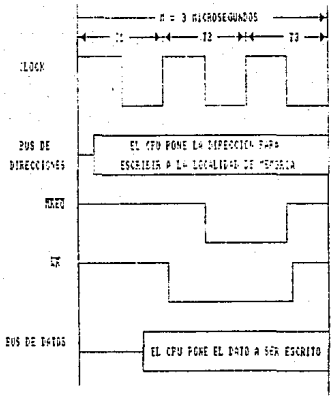


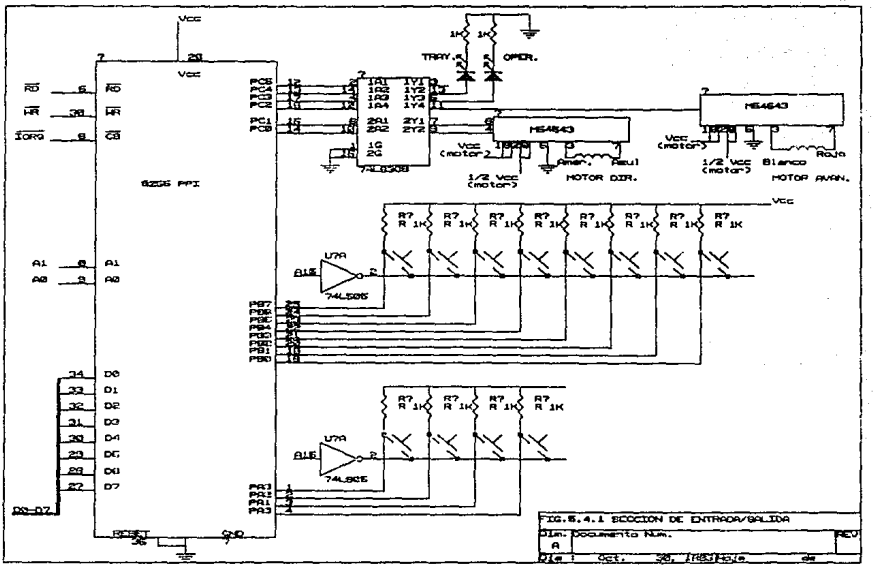
Fig.5.3.5 Ciclo de escritura

5.3.5 Ejecución de una instrucción del Z-80

La ejecución de una instrucción completa es una combinación de ciclos de lectura y escritura. Para cada instrucción, el cpu primero realiza un ciclo de lectura de un primer byte dentro del registro decodificador de instrucciones del cpu. El cpu reconoce que bytes adicionales deben ser leídos, si lo requiere la instrucción. El bus de direcciones tiene el contenido del program counter (PC) y es incrementado en uno por cada lectura realizada, para leer en el siguiente byte de instrucción. Es importante mencionar que ésto ocurre siempre que las instrucciones son de tipo "direccionamiento directo", pués cuando se trata de instrucciones con direccionamiento indirecto, el valor que el cpu pone en el bus de direcciones no es necesariamente el valor del program counter. En éste caso, la dirección que el cpu coloca en el bus de direcciones depende de la instrucción.

Durante los ciclos de escritura de instrucciones que mueven un valor al registro A, el cpu transfiere el valor correspondiente a este registro.

Al ejecutar un programa, el bus de direcciones y el bus de datos están muy áctivos con ciclos de lectura y de escritura cada vez que el cpu transfiere un operando del cpu a localidades de memoria y de memoria a registros del cpu.


VCC
TRAV.
OPER.
MS4643
MS4643
MOTOR DIR.
MOTOR AVAN.
Rojo
Blanco
Azul
8255 PPI
74LS08
74LS05
74LS05
RESET
GND
FIG.5.4.1 SECCION DE ENTRADA/SALIDA
Dim: Documento NUM.
REV
A
Oct. 28, 1988

5.4 SECCION ENTRADA SALIDA

La interface entre el microprocesador Z-80 y la interface periférica programable (PPI) 8255 es discutida en ésta sección. El PPI acepta comandos del Z-80 y maneja las tareas de recibir y transmitir datos de 24 líneas de entrada/salida. La lógica asociada con los led's, el teclado, y las líneas de salida para la etapa de potencia para activar los motores es discutida aquí también.

5.4.1 El PPI 8255

La figura 5.4.1 muestra la sección de entrada/salida del sistema. El PPI actúa como un buffer intermediario entre el CPU Z-80 y el mundo exterior de los led's y el teclado, además de las líneas de salida. El propósito del PPI es hacer compatible los datos de e/s con el mundo exterior en velocidad y en características eléctricas de las señales del Z-80.

El PPI consta de cuatro registros similares a los registros del Z-80. Cada registro es de 8 bits, tres de los registros se conectan a tres conjuntos de 8 líneas de e/s, como se muestra en la figura.5.4.2

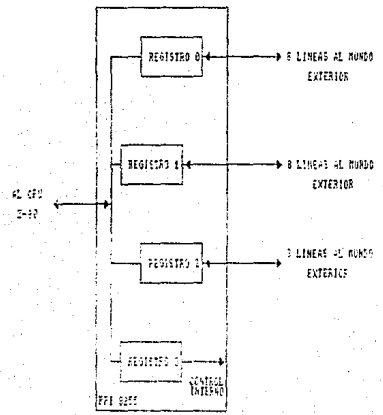


Fig.5.4.2 Registros del PPI

El CPU puede leer o escribir a cada uno de los cuatro registros en el PPI mediante una instrucción de lectura (IN) o escritura (OUT). Por ejemplo, al ejecutar la instrucción IN A,(2) lee del registro 2 del PPI y transmite los 8 bits del dato en el registro hacia el registro A del CPU. Al ejecutar la instrucción OUT (0),A escribe el contenido del registro A del cpu dentro del registro 0 del PPI.

Los 4 registros son direccionados con las direcciones 00, 01, 10, y 11. Dos líneas del bus de direcciones se conectan al PPI: A1 y A0 y esas dos líneas son todo lo requerido para accesar el registro del PPI especificado. Siempre que una instrucción de entrada/salida (IN o OUT) es ejecutada, el programa debe conversar con el PPI y no con ningún otro dispositivo. Para diferenciar entre comunicación con memoria y comunicación de e/s la señal IORQ esta conectada a el PPI (pin 6). La señal IORQ es un 0 sólo cuando una lectura (IN) o una escritura se está ejecutando. La dirección de transmisión es decodificada por el PPI con las señales WR y RD (pin 36 y 5).

Para un análisis más detallado de una operación de entrada/salida, las señales durante una instrucción de lectura (IN) son mostradas en la figura 5.4.3. Cuando una operación de lectura es ejecutada, el cpu pone primero la dirección del dispositivo de entrada/salida en las líneas del bus de direcciones A0-A7. El formato de la instrucción de entrada/salida especifica una dirección de sólo 8 bits, de manera que el número máximo de dispositivos que pueden ser direccionados es 256. En el sistema, solamente 4 direcciones son utilizadas: 0, 1, 2, 3.

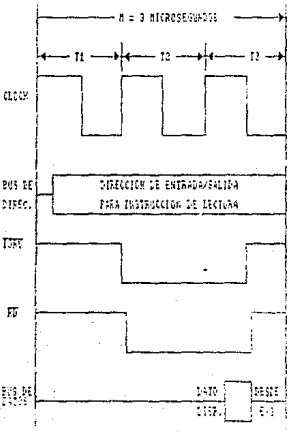


Fig.5.4.3 Ciclo de Lectura para E/S

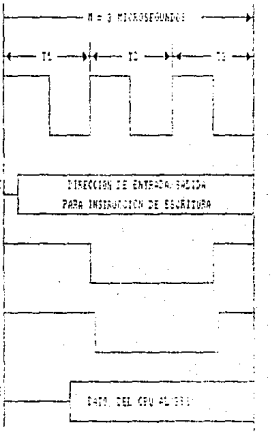


Fig.5.4.4 Ciclo de Escritura para E/S

Brevemente después de que la dirección es puesta en el bus de direcciones, el cpu activa las señales de IORQ y RD poniendolas en cero (0 volts). Esto es interpretado por el PPI como una operación de entrada/salida, especificamente una lectura (IN). El PPI ve las dos lineas de dirección A0 y A1 para determinar cuál de los 4 registros será leido. Entonces pone el contenido del registro seleccionado dentro de las líneas del bus de datos D7-D0. En la mitad del tercer ciclo T, el cpu introduce el contenido del bus de datos dentro del registro A. Como en el caso de la memoria, el tiempo entre el inicio de IORQ y RD y el paso de los datos a el registro A se debe a 2 razones. Primero, el CPU necesita el tiempo para secuenciar las operaciones internas. Segundo, éste intervalo da tiempo para que el dispositivo externo responda.

Una operación de escritura, para una instrucción OUT es similar (ver fig.5.4.4). La dirección y la señal IORQ se activan como en una lectura. La señal RD permanece inactiva mientras que la señal WR está en cero. Sin embargo, antes de que la señal WR se active el dato del registro A es puesto dentro de las líneas del bus de datos D7-D0. La señal WR efectivamente manda el dato hacia el registro de datos del PPI apropiado.

Cada uno de los cuatros registros de PPI puede entonces leer o escribir según la instrucción de entrada/salida especifique una dirección de 0,1,2,3. Los tres primeros registros están asociados con 3 conjuntos de 8 líneas que van al mundo exterior. El cuarto registro es un registro de control que toma el byte de control de modo para el PPI.

El PPI puede operar en distintos modos. El modo 0 es "Básico de Entrada/Salida", el modo 1 es "Strobed Entrada/Salida" y el modo 2 "Bus Bidireccional". El modo utilizado en este proyecto es el modo 0.

En modo 0 las líneas PA7-PA0 pueden ser todas entradas o todas salidas, pero no una mezcla. Las líneas PB7-PB0 pueden ser todas entradas o todas salidas. Las líneas PC7-PC0 están susbdivididas en dos conjuntos de 4 y cada conjunto puede ser de entrada o de salida.

En este proyecto hemos seleccionadao las siguientes combinaciones:

PA7-PA0	Todas Entradas	DIRECCION 0
PB7-PB0	Todas Entradas	DIRECCION 1
PC7-PC0	Todas Salidas	DIRECCION 2

La primera acción que debe de tomarse antes de realizar operaciones de entrada/salida con las 24 lineas del PPI es programar el PPI enviando la palabra de control hacia el registro de control del PPI. Esta palabra de control es almacenada en el registro de control y permanece en él mientras se le suministre alimentación al PPI. La palabra de control para modo 0 y la configuración de lineas anterior se muestra en la figura 5.4.5 con las instrucciones requeridas. Habiendo sacado la palabra de control apropiada hacia el registro de control del PPI, el PPI está listo para ser usado para transmitir datos entre el cpu y el mundo exterior. El "mundo exterior" está dividido en tres areas: Teclado, Led's, y lineas de salida para la etapa de potencia para motores.

En la figura 5.4.5 aparecen los 8 bits que componen la palabra de control. Cada bit tiene un significado para la configuración del PPI, dependiendo de su valor, uno o cero. Los bits de la palabra de control se encuentran divididos en dos conjuntos: grupo A y grupo B. El grupo B está compuesto por los bits D3, D4, D5, D6. El grupo B lo integran los bits D0, D1, D2. Dentro del grupo A se configuran el puerto A y la parte superior del puerto C, además del modo de operación de éste grupo (modo 0, modo 1, o modo 2). En el grupo B se configura el puerto B, así como la parte baja del puerto C y el modo de operación para éste. Por otra parte, si se requiere trabajar con el conjunto de banderas activo, es necesario que el bit D7 sea un uno.

Para ésta aplicación, atendiendo al diagrama eléctrico de la sección de entrada/salida que se muestra en la figura 5.4.1, se tienen los siguientes valores en los bits de la palabra de control para la configuración de los registros del PPI:

-El bit D7 es un uno, ya que pone las banderas en modo activo.
-Los bits D6 y D5 son ambos ceros para configurar el grupo A de la líneas del PPI en modo 0.
-El bit D4 es un uno para que los ocho bits del puerto A (puerto 0) sean de entrada, debido a que es en éste puerto donde se conecta una parte del teclado.
-El bit D3 es un cero, configurando de ésta manera la parte superior del puerto C (los 4 bits más significativos , PC4-PC7, del puerto 2) como líneas de salida. Esto es necesario, aunque solamanete una de las cuatro líneas será utilizada dentro del sistema, la línea del bit PC4. Esta línea activará mediante el buffer 74LS368 a un led que será parte del monitoreo del status del sistema.
-El bit D2 deberá de ser también un cero, pués el grupo B de las líneas del PPI trabajará en modo 0, al igual que el grupo A.
-El bit D1 será un uno, puesto que los ocho bits del puerto B (puerto 1) son conectados a la otra parte del teclado y por lo tanto las líneas del puerto B deben de ser configuradas como líneas de entrada al PPI.
-El bit D0 será un cero para que los cuatro bits menos significativos del puerto C (puerto 2) sean configurados como líneas de salida del PPI. Tres de éstas líneas activaran a travez del buffer 74LS368 a la etapa de potencia de los motores. La otra línea activará a un led que será parte del monitoreo del status del sistema.

La palabra de control para modo 0 y la configuración de líneas anterior se muestra en la figura 5.4.6 con las instrucciones requeridas. Habiendo sacado la palabra de control apropiada hacia el registro de control del PPI, el PPI está listo para ser utilizado para transmitir datos entre el cpu y el mundo exterior. El "mundo exterior" está dividido en tres areas:

-Teclado
-Led's
-Líneas de salida para la etapa de potencia para motores.

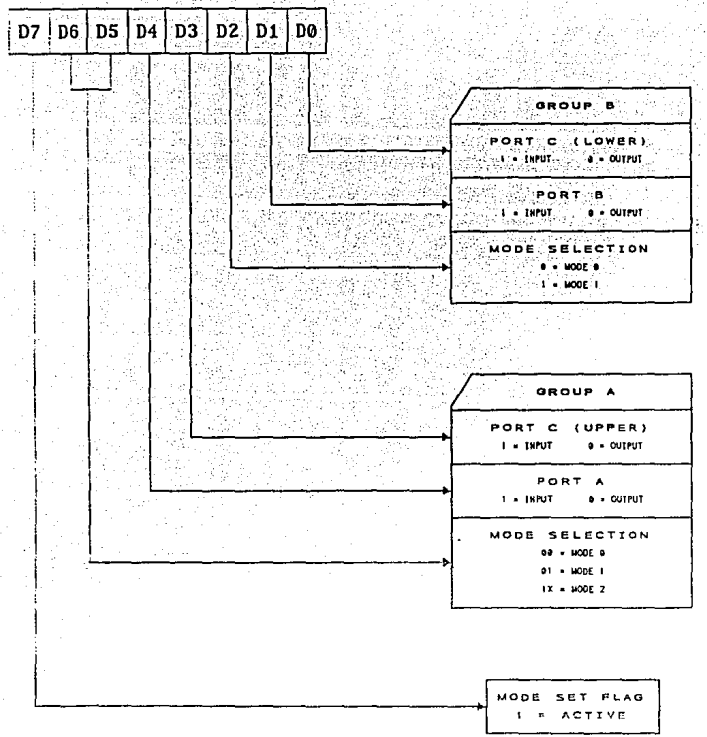


FIG. 5.4.5 OBTENCION DE LA PALABRA DE CONTROL DEL PPI

```
LD A,92H        ;CARGA PALABRA DE CONTROL
OUT (3),A       ;PONE EN EL PPI EN MODO-0
```

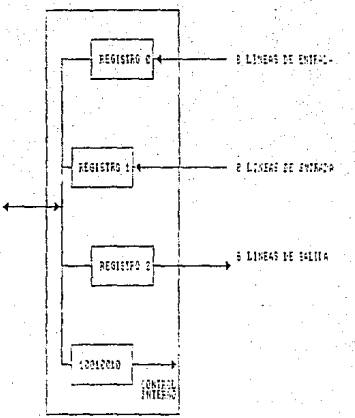


Fig.5.4 Palabra de Control y configuración del PPI

5.4.2 Teclado

El teclado utilizado es un teclado matricial muy simple para continuar con los bajos costos del proyecto. El teclado y circuiteria asociada se muestran en la figura 5.4.1. Este teclado está compuesto de dos partes: una parte para el registro 0 del PPI, de 1 renglon por 4 columnas; y una parte de 1 renglón por 8 columnas para el registro 1 del PPI. Al presionar alguna tecla, se conecta un renglón con una columna. Cuando ninguna tecla es presionada , las entradas del registro 0 en las lineas PA0, PA1, PA2 y PA3 y las entradas del registro 1 del PPI en las lineas PB0, PB1, PB2, PB3, PB5, PB6, PB7 estan a Vcc o 1 lógico. Cuando una tecla es presionada, dos lineas son conectadas y la linea de la columna asociada con la tecla es conectada a la linea asociada con el renglón de la tecla.

La linea de la columna va a una de las 4 entradas de los 4 bits menos significativos del registro 0 (PA0-PA3) del PPI o a una de las 8 entradas del registro 1 (PB0-PB7) del PPI. La linea de la columna refleja el estado del renglón conectado a la salida del inversor 74LS05. Si la salida del inversor para el renglón es un cero, entonces la entrada para el PPI será un cero; si la salida del inversor para el renglón es un uno, entonces la entrada para el PPI será un uno.

Para detectar la presión de una tecla, si la tecla es presionada y la salida del inversor es un cero, entoces la entrada del PPI de la columna asociada con la tecla presionada será un cero. Si la salida del inversor es cero, se puede saber con precisión cuál de las teclas ha sido presionada por la posición del bit de la columna asociada con la tecla.

Para comprender como funciona este circuito, supongase que en la fig. 5.4.1 la tecla GRAB es presionada y los dos puntos de contacto son conectados. Si se hace A15 un uno, la salida del inversor 74LS05 será un cero y la entrada al PPI, PA0, sera también cero. El resto de las entradas del PPI serán unos. Ahora, si se hace A15 un cero, la salida del inversor 74LS05 será un uno y la entrada al PPI, PA0, será ahora un uno. El resto de las entradas del PPI serán unos como en el caso anterior debido a que ninguna de esas teclas fué presionada. Esto mismo se aplica para la detección de la presión de cualquier otra tecla. La detección de una tecla se puede deducir observando cuál de los bits PA0, PA1, PA2, PA3, PB0, PB1, PB2, PB3, PB5, PB6, PB7 es un cero asegurando que la entrada del inversor sea un uno.

Este proceso es llamado "scanning" (inspección) del teclado y es una técnica común para deteccion y decodificacion de teclados. Se puede observar que éste principio puede ser aplicado para teclados de matrices de order superior a 1 por 4 y 1 por 8, como los utilizados en ésta aplicación. Nuevamente, se decidió utilizar el teclado de éste orden para reducir costos. Se implemento el teclado con un sólo renglón y su inversor correspondiente para facilitar el desarrollo del software del sistema, aunque el número de columnas es mayor al que hubiese resultado de haber empleado un número mayor de renglones. Esto se analizará en el punto 5.5, Desarrollo del Software.

La inspección en el sistema es implementada al habilitar simultaneamente la línea de direccion A15 y leer el estado de las entradas del PPI (PA0-PA3 y PB0-PB7). Si todas las entradas del PPI son unos, entonces ninguna tecla ha sido presionada.

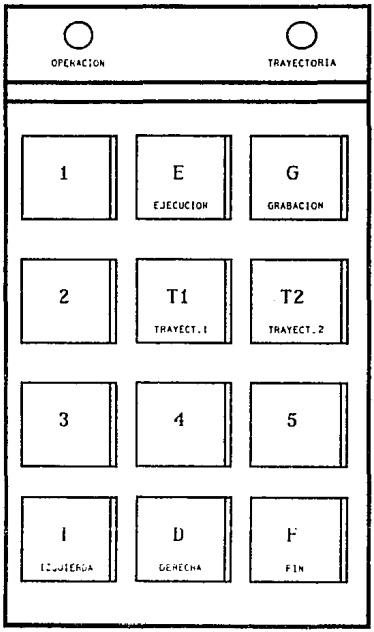


FIG. 5.4.7 TECLADO

Sí uno de los bits es un cero y además A15 es un uno, entonces hay una tecla presionada y la tecla que está siendo presionada se encuentra buscando el número de la columna.

La clave de ésta tecnica en el sistema es que la instrucción de entrada/salida IN A,(reg.PPI) pone el contenido del registro A del cpu en las líneas de dirección A0-A15 al mismo tiempo que la lectura de las líneas PA0-PA3 (en caso de IN A,(0)) o de las líneas PB0-PB7 (en caso de IN A,(1)) es realizada.

5.4.3 Líneas de salida

El sistema cuenta con seis líneas de salida de los seis bits de orden inferior del registro 2 del PPI, registro C, que son amplificadas en potencia (buffered) por los inversores del circuito integrado 74368. Este chip invierte la señal del registro del PPI, pero lo importante, es que proporciona una mayor capacidad para manejar corriente para el manejo de los dispositivos externos en el sistema. Estos dispositivos son:

a) Una etapa de potencia para activar los motores de avance y dirección.

b) Dos Led's para el monitoreo del status de la operación del sistema.

a) Etapa de Potencia.

Esta etapa esta basada en el circuito integrado M5454. El circuito M5454 es un circuito integrado monolitico capaz de manejar dispositivos que pueden cambiar entre dos estados (encendido o apagado). Transistores, SCR's, y triacs son algunos de los dispositivos que pueden ser controlados por el M5454. Este dispositivo puede tambien controlar relevadores, solenoides, motores (de avance y reversa de CD) y motores de paso de CD.

El M5454 contiene un par de circuitos controladores individuales, cada uno de ellos consiste de un circuito manejador de fuente con protección limitadora de corriente. El segundo controlador contiene un diodo que absorbe energía para proteger el dispositivo contra cualquier impulso inductivo durante los cambios de estado. El M5454 está protegido contra condiciones de sobrevoltaje en los manejadores de salida y condiciones de sobre temperatura.

Los niveles operativos de entrada son compatibles con TTL. Las salidas están en su condicion de apagado (off) cuando sus respectivas entradas están en un estado alto (HI), o en circuito abierto. Las salidas están en su estado de encendido (on) cuando sus respectivas entradas son bajas (LO). El M5454 tiene un montaje de forma horizontal.

Se han utilizado dos circuitos M5454, uno para activar el motor de avance y un segundo para activar el motor de direcciones (giros a la izquierda y a la derecha). Del M5454 que controla el motor de avance solamente se utiliza uno de sus dos circuitos controladores puesto que el carro unicamente avanza y no retrocede. Su entrada es el bit 2 del puerto C del PPI (PC2) amplificado en corriente por el circuito 74LS368.
Las entradas para el M5454 que controla el motor de direcciones son para giros a la derecha y para giros a la izquierda los bits 0 y 1 del puerto C del PPI (PC0 y PC1) amplificados en corriente por el 74LS368 respectivamente.
Estos dos circuitos controladores han sido configurados de maneras diferentes. Una de las terminales del motor de avance se conecta a la salida del controlador y la otra a tierra. Esto implica que el motor de avance no tendrá retroceso. Para el motor de direcciones se tiene que una de sus terminales se conecta a la salida 1 de su controlador mientras que la otra terminal se conecta a la salida 2. Esto permite que el motor se active en la dirección que corresponda con la entrada que se aplique al controlador (entrada 1 o entrada 2), lo cual obliga a que una y solo una entrada se active por el microprocesador al mismo tiempo.

b) Leds de monitoreo del sistema.
La señal del bit 3 del puerto C del PPI (PC3) es amplificada en corriente por el buffer 74LS368 para activar el led de monitoreo de Selección de operación (Grabación o Ejecución).
La señal del bit 4 del puerto C del PPI (PC4) es amplificada en corriente por el buffer 75LS368 para activar el led de monitoreo de Selección de trayectoria (Trayectoria 1 o Trayectoria 2).

5.5 SOFTWARE

5.5.1 Definición de requerimientos

Los requerimientos son todas aquellas funciones que debe de realizar el programa para que el sistema cumpla satisfactoriamente los objetivos y alcances del proyecto. Esta defición de requerimientos debe de quedar especificada en forma clara y precisa, pués de ésta manera se facilitará el desarrollo del software.

A partir del diagrama de bloques en la figura 5.1.1, se han definido los siguientes requerimientos:

a) Inicialmente el usuario puede elegir en el teclado una operación a realizar por el sistema. Esta operación podrá ser de:

- Grabación
- Ejecución

Un Led indica a el usuario que se encuentra en éste punto y que debe de elegir una de las 2 operaciones. En caso de teclear cualquier otra tecla que no sea Grabación o Ejecución se hará caso omiso de la tecla presionada. Podrá pasar al siguiente inciso hasta que se presione una de las 2 teclas de operación (Grabación o Ejecución).

b) Después de haber seleccionado una de las dos operaciones (Grabación o Ejecución), el Led indicador de selección de operación se apagará y se encenderá un Led que le indica a el usuario que debe seleccionar una de dos trayectorias:

- Trayectoria 1
- Trayectoria 2

Si el usuario presionó otra tecla diferente a Trayectoria 1 o Trayectoria 2, el sistema hará caso omiso y seguirá esperando la presión de una de las dos teclas de trayectoria.

c) En este punto no hay Leds indicadores, entonces al estar apagados los dos Leds con que cuenta el sistema, el usuario deberá asumir que se encuentra en éste punto. Dependiendo de la selección de operación (Grabación o Ejecución) que se haya realizado en el inciso a), se tienen las siguientes 2 opciones:

- Si se seleccionó la operación de Grabar, el sistema está listo para aceptar comandos de grabación de la trayectoria seleccionada en el inciso b). Esta grabación de la trayectoria se realiza a travez del teclado y se dará por terminada una vez que se presione la tecla "FIN". Entonces el sistema regresará al inicio del inciso a) para aceptar una nueva operación.
  Es importante mencionar que si el usuario seleccionó en el punto b) una trayectoria que tiene una trayectoria grabada anteriormente, dicha grabación se borrará y se sustituirá por la nueva.

- Si se seleccionó la operación Ejecutar, el sistema espera que el usuario presione en el teclado una de las teclas del "1" al "5". Esta tecla es el número de veces que la trayectoria seleccionada en el inciso b) será ejecutada por el sistema.
  Se tendrá un tiempo de espera de 5 segundos para que se comienze a ejecutar automáticamente la trayectoria, esto, para que el usuario tenga tiempo de retirarse del dispositivo antes de que avance.
  Cuando la trayectoria haya sido ejecutada el número de veces que se tecleó, el dispositivo se detendrá y el sistema regresará al inciso a) para una nueva operación.

Una vez definidos los requerimientos para el software del sistema, se tiene en una representación inicial, el diagrama de bloques que se muestra en la figura 5.5.1.1 . Esta es la forma más general en que se pueden mostrar gráficamente todos las funciones que debe realizar el sistema.

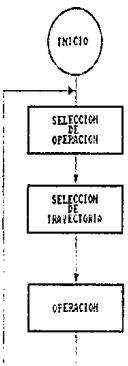


FIG. 5.5.1.1

Además de definir los requerimientos que debe cumplir la pieza de software a desarrollar, se deben de tomar en cuenta aquellos factores del hardware que tienen una relación directa con el desarrollo del software. Las consideraciones que se han hecho para este sistema son:

1.- Instrucciones para la configuración de los puertos del PPI.
Esto es indispensable para que se realicen las operaciones de entrada/salida entre el sistema digital y el mundo externo. Para configurar los puertos como se definió en el punto 5.4, es necesario sacar la Palabra de control hacia el puerto de control. El valor de la palabra de control se determinó de las figuras 5.4.5 y 5.4.6 : 92H. La secuencia de intrucciones para configurar los puertos se indica a continuación:

```
LD A, 92 H

OUT (3), A
```

Primeramente se carga el acumulador (registro A del cpu) con el valor de la palabra de control. El valor de la palabra de control almacenado en el acumulador se envia con una instrucción OUT hacia el puerto de control del PPI. Este puerto tiene la dirección 3.

Estas dos instrucciones, que configuran los puertos de entrada/salida, deben ser las primeras instrucciones que se ejecuten en el programa.

2.- Zona de memoria EPROM para la grabación del sistema operativo.
De acuerdo con el mapa de memoria, mostrado en la figura 5.3.1, para la grabación del programa se tienen las 2048 primeras localidades de memoria (de la localidad 0H a la 07FFH). Ciertas localidades han sido reservadas por el fabricante del cpu Z-80 para interrupciones. Tal es el caso de la localidad 102, reservada para la interrupcion no mascarable (NMI). Localidades abajo de la 102 son dedicadas para otras posibles interrupciones no implementadas. Sin embargo el programa inicia en la localidad 0 dado que no se manejan interrupciones. Entonces, la zona para la grabación de memoria está en el rango de la localidad 0 a la 2047.
Se debe preveer que el tamaño del programa no rebase la zona de memoria EPROM.

3.- Zonas de memoria RAM para la grabación de trayectorias.
En el mapa de memoria (ver figura 5.3.1) se definió que la zona para memoria RAM será de la localidad 2048 a la localidad 2175, teniendose un total de 128 localidades. Debido a que se tienen dos posibles trayectorias, este total se dividirá en dos rangos con igual número de localidades para cada una de las dos trayectorias. Esto se muestra a continuación:

TRAYECTORIA	AREA DE MEMORIA (DECIMAL)	AREA DE MEMORIA (HEXADECIMAL)
1	2048-2111	0800-0840
2	2112-2175	0841-087F

La selección de éstos rangos es necesaria para hacer referencia a las localidades iniciales de la trayectoria. Cuando se este en el proceso de Grabación o Ejecución, las instrucciones que graben o ejecuten una trayectoria incluirán como parámetro principal la localidad inicial que corresponda con la trayectoria que se este almacenando o ejecutando.

Localidad inicial para la trayectoria 1 ..... 0800H
Número de locals. para la trayectoria 1 00064

Localidad inicial para la trayectoria 2 ..... 0841H
Número de locals. para la trayectoria 2 00064

4.- En el punto 5.4 se definió la configuración del puerto de salida C para los dispositivos de salida (la etapa de potencia para los motores y los Leds para el monitoreo del status de operación del sistema). En la figura 5.4.1 se observa que el puerto de salida C está conectado a los dispositivos de salida a través de un buffer 74LS368 que invierte la señal. Los dispositivos de salida son activados con un uno lógico. Lo valores que serán enviados al exterior se muestran en la siguiente tabla:

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	VALOR	OPERACION
1	1	1	1	0	1	1	1	F7H	I
1	1	1	0	1	1	1	1	EFH	II
1	1	1	1	1	0	1	0	FAH	III
1	1	1	1	1	0	0	1	F9H	IV
1	1	1	1	1	0	1	1	FBH	V

I .- Enciende Led indicador de selección de operación
II .- Enciende Led indicador de selección de trayectoria
III.- Activa motor de dirección a la derecha
IV .- Activa motor de dirección a la izquierda
V .- Activa motor de avance

5.- Retardos necesarios para la ejecución de trayectorias.
Se incluirán rutinas de retardo de uno y cinco segundos. La rutina de 5 segundos cada vez que se ejecute una trayectoria que ha sido grabada anteriormente. Este es un tiempo que se ha considerado apropiado para que en este prototipo se tenga una pausa antes de que se inicie una nueva ejecución de la trayectoria.
Durante la ejecución de la trayectoria, por cada comando que fué grabado se almacenó en memoria RAM una clave que tiene un significado para la ejecución. Debido a que el tiempo en el que se ejecuta una instrucción de salida por parte del sistema digital es del orden de los microsegundos, el motor no puede ser activado, dado que los motores utilizados en el prototipo tienen un tiempo de respuesta lento para la velocidad con que opera el sistema digital. Es necesario incluir un retardo en las instrucciones de salida del sistema digital hacia los motores.
Este retardo acoplará la velocidad del sistema digital con la de los motores. De esta manera la señal enviada por el sistema digital activará al motor durante un segundo, tiempo que dura la rutina de retardo, para que éste ejecute el comando solicitado (un avance, un giro a la izquierda, o un giro a la derecha). Se ha considerado que un segundo es un periodo adecuado para que el motor reaccione y sea activado para ejecutar un comando.

5.5.2 Diagrama de flujo

En el diagrama de bloques de bloques de la figura 5.5.1.1 se observa que el programa permitirá efectuar 3 operaciones principales:

- Selección de operación
- Selección de trayectoria
- Operación

Este diagrama de bloques se ha trasladado al diagrama de flujo que se muestra en la figura 5.5.2.1.
Aplicando la técnica de diseño modular, cada uno de los 3 modulos anteriores es visto en el diagrama de flujo como una serie de submodulos. Los submodulos en que se ha subdividido cada modulo son:

Selección de operación:
- encender led de operación
- lectura de la operación de operación, via el teclado
- decisión de operación

Selección de trayectoria:
- encender led de trayectoria
- lectura de la trayectoria, via el teclado
- decisión de trayectoria

Operación:

(grabación)
- lectura de comando para grabar trayectoria, via el teclado
- grabación en memoria de los comandos
- asi sucesivamente hasta que se teclee FIN

(ejecución)
- lectura del número de veces a ejecutarse la trayectoria (1 a 5)
- retardo de 5 segundos (cada vez que se inicie la ejecución de una trayectoria)
- lectura del comando desde memoria
- envio de comando a dispositivo de salida (motor, Led)
- asi sucesivamente hasta leer de memoria el comando FIN
- asi sucesivamente hasta completar el número de trayectorias.

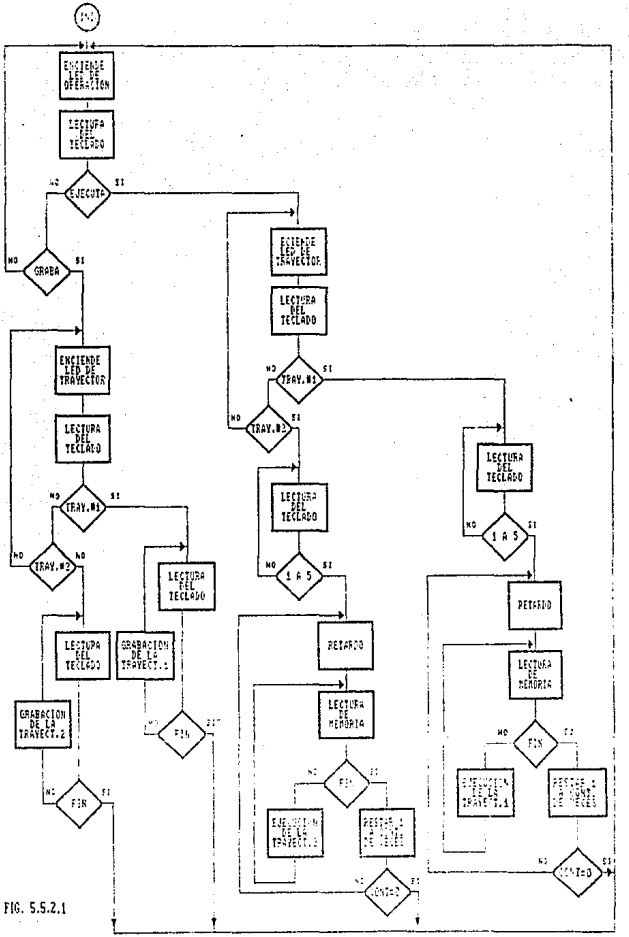

ENCIENDE LED DE OPERACION
LECTURA DEL TECLADO
EJECUTA
GRABA
ENCIENDE LED DE TRAYECTOR
LECTURA DEL TECLADO
TRAY.#1
TRAY.#2
1 A 5
LECTURA DE MEMORIA
FIN
GRABACION DE LA TRAYECT.1
GRABACION DE LA TRAYECT.2
EJECUCION DE LA TRAYECT.1
PRETARDO
CONTEO

FIG. 5.5.2.1

5.5.3 Desarrollo del software

Después de que cada uno de los tres modulos básicos ha sido dividido en submodulos más especificos, se continuará con la escritura del programa utilizando los mnemónicos del lenguaje ensamblador del Z-80.

Dado que el desarrollar un diagrama de flujo tan detallado, implica el mismo grado de dificultad que el escribirlo, se ha decidido pasar directamente del diagrama de flujo a escribir el programa, teniendo como apoyo además del diagrama de flujo, las especificaciones numeradas en el punto 5.5.2.1. Con ésto se ahorra el tiempo que se invirtiría en dibujar cada uno de los submoludos en forma detallada. El programa se anexa en la siguientes hojas.Este programa está editado en el procesador de palabras de microsoft, WORKS, versión 2.0. El programa se encuentra documentado para la interpretación del mismo y facilitar al programador posibles futuras modificaciones al sistema. El nombre de este archivo debe tener una extensión .asm. El nombre que se ha dado a este programa es PROG.ASM

5.5.4 Implantación del software

El programa editado en el procesador de textos es entonces "compilado" con el software de aplicación - SIM Z80 -. Este software corre bajo ambiente del sistema operativo MS DOS, versión 3.0 en adelante.

Desde el prompt del sistema se teclea el siguiente comando:

C:\ASMZ80

Una vez que se dió el comando anterior aparecerá la pantalla siguiente solicitando el nombre del archivo fuente, así como el nombre de dos archivos que se generarán con esta utilería:

A S M - Z 8 0

Nombre del archivo fuente ?

Nombre del archivo de salida ?

Nombre del archivo objeto ?

donde:

archivo fuente.- es el programa escrito en mnemonicos

archivo objeto.- es el programa convertido de código fuente (mnemonicos) a código objeto en hexadecimal (código de máquina)

archivo de salida.- es el programa fuente con comentarios sobre posibles errores durante la conversión del programa fuente a objeto

Los nombre que se han dado a los programas son:

archivo fuente	prog.asm
archivo de salida	prog.sal
archivo objeto	prog.cod

Es importante mencionar que sí el programa fuente tiene errores, el archivo objeto no se podrá generar. Sí esto ocurriera, el único archivo que se genera es el archivo de salida. Este archivo de salida es el mismo archivo archivo fuente escrito en mnemonicos, al cuál se le han añadido comentarios sobre los errores detectados durante la conversión del código fuente a código objeto. Los errores se encuentran plenamente identificados y localizados donde ocurrieron, debido a algun error de sintáxis. Entonces al mismo tiempo que ésta utilería (ASM80) del simulador SIMZ80 generá código objeto a partir de código fuente, detecta también los posibles errores que se tengan en el programa fuente.

5.5.4 Grabación en eprom

Para la grabación del programa se utilizó el grabador de memorias universal UNIPRO de la manufacturera XelTek. Este grabador tiene una interface con PC y su software de aplicación permite la captura en linea de código o bien leer archivos que contengan el código a ser grabado automáticamente a la memoria eprom. El archivo que contiene el código de máquina es: **PROG.COD**
El sofwtare de el grabador de memoria requiere que el código del archivo PROG.COD esté en código ascii. Dado que el archivo generado por la utileria ASMZ80 está en hexadecimal fué necesario realizar un programa que convirtiera un archivo en hexadecimal a ascii. Este programa fué escrito en el lenguaje de programación Pascal. Se anexa el programa hextoasc.pas. Una vez que se cargó el archivo ascii, se realizó la operación de grabar a memoria desde la opción Grabar en el menu principal.

```
;*********************************************************************
;**************          P R O G . A S M          ********************
;**************          ---------------          ********************
;**************  P R O G R A M A   P R I N C I P A L  ****************
;*********************************************************************



        .ORG 00H      ; El programa inicia en la loc. 0
         LD A, 92H    ; Carga palabra de control
         OUT(3), A    ; para configurar puertos
;---------------------------------------------------------------------
INICIO   LD A,00F7H   ; Enciende led para la
         OUT(2), A    ; selección de operación.
         LD A, 80H    ; Lectura del teclado para
         IN A, (0)    ; seleccionar operación.
         BIT 0, A     ; Si tecleó grabación
         JP Z, GRAB   ; salta a GRAB.
         BIT 1,A      ; Si tecleó ejecución
         JP Z, EJEC   ; salta a EJEC.
         JP INICIO    ; Si no tecleó grabación ni
                      ; ejecución regresa a INICIO
                      ; para reintentar selección
                      ; de operación (poll select).
```

```
;*********************************************************************
;*****       RUTINA GENERAL DE GRABACION DE TRAYECTORIAS       ********
;*********************************************************************


GRAB     LD A,00EFH      ; Enciende led para selección
         OUT (2),A       ; de trayectoria a grabar
GRABPOL  LD A,80H        ; Lectura del teclado para
         IN A,(0)        ; seleccionar trayectoria
         BIT 2,A         ; Si tecleo trayectoria 1
         JP Z,GRAB1      ; salta a GRAB1
         BIT 3,A         ; Si tecleo trayectoria 2
         JP Z,GRAB2      ; salta a GRAB2
         JP GRABPOL      ; Si no tecleo trayectoria 1
                         ; ni trayectoria 2 regresa a
                         ; GRAB para reintentar se-
                         ; leccionar trayectoria .


;*********************************************************************
;*****       RUTINA GENERAL DE EJECUCION DE TRAYECTORIAS       ******
;*********************************************************************


EJEC     LD A,00EFH      ; Enciende led para selección
         OUT (2),A       ; de trayectoria a ejecutar .
EJECPOL  LD A,80H        ; Lectura del teclado para
         IN A,(0)        ; seleccionar trayectoria .
         BIT 2,A         ; Si tecleo tryectoria 1
         JP Z,EJEC1      ; salta a EJEC1 .
         BIT 3,A         ; Si tecleo trayectoria 2
         JP Z,EJEC2      ; salta a EJEC2 .
         JP EJECPOL      ; Si no tecleo trayectoria 1
                         ; ni trayectoria 2 regresa a
                         ; EJEC para reintentar se-
                         ; leccionar trayectoria .
```

```
;*****************************************************************
;*****          RUTINA DE GRABACION DE TRAYECTORIA 1          ******
;*****************************************************************

GRAB1   LD A,00FFH      ; apaga leds indicando
        OUT (2),A       ; grabacion realizandose .
        LD HL,07FFH     ; Carga HL con loc.inic.tray1 .
        JP LEER         ; Salta a la rutina comun de
                        ; grabacion LEER .


;*****************************************************************
;*****          RUTINA DE GRABACION DE TRAYECTORIA 2          ******
;*****************************************************************

GRAB2   LD A,00FFH      ; Apaga leds indicando
        OUT (2),A       ; grabacion realizandose .
        LD HL,0840H     ; Carga HL con loc.inic.tray2 .
        JP LEER         ; Salta a la rutina comun de
                        ; grabacion LEER .


;---------------------------------------------------------------
;              RUTINA COMUN DE GRABACION DE TRAYECTORIAS        |
;---------------------------------------------------------------

LEER    INC HL          ; Incrementa HL .
RELEER  IN A,(1)        ; Lee teclado .
        BIT 0,A         ; Si tecleo I
        JP  Z,IZQG      ; salta a IZQG .
        BIT 1,A         ; Si tecleo 1
        JP  Z,UNOG      ; salta a UNOG .
        BIT 2,A         ; Si tecleo 2
        JP  Z,DOSG      ; salta a DOSG .
        BIT 3,A         ; Si tecleo 3
        JP  Z,TREG      ; salta a TREG .
        BIT 4,A         ; Si tecleo 4
        JP  Z,CUAG      ; salta a CUAG .
        BIT 5,A         ; Si tecleo 5
        JP  Z,CING      ; salta a CING .
        BIT 6,A         ; Si tecleo F
        JP  Z,FING      ; salta a FING .
        BIT 7,A         ; Si tecleo D
        JP  Z,DERG      ; salta a DERG .
        JP RELEER       ; Si no tecleo alguno
                        ; de los anteriores
                        ; regresa a RELEER .
```

```
IZQG    LD (HL),0       ; Almacena en memoria clave de
        JP RETARG       ; izquierda y salta a RETARG .

UNOG    LD (HL),1       ; Almacena en memoria clave de
        JP RETARG       ; avanza 1 y salta a RETARG .

DOSG    LD (HL),2       ; Almacena en memoria clave de
        JP RETARG       ; avanza 2 y salta a RETARG .

TREG    LD (HL),3       ; Almacena en memoria clave de
        JP RETARG       ; avanza 3 y salta a RETARG .

CUAG    LD (HL),4       ; Almacena en memoria clave de
        JP RETARG       ; avanza 4 y salta a RETARG .

CING    LD (HL),5       ; Almacena en memoria clave de
        JP RETARG       ; avanza 5 y salta a RETARG .

FING    LD (HL),6       ; Almacena en memoria clave de
        JP INICIO       ; fin y retorna a INICIO .

DERG    LD (HL),7       ; Almacena en memoria clave de
        JP RETARG       ; derecha y salta a RETARG .

;-----------------------------------------------------


RETARG  LD IX,100       ; IX = 100
        LD DE,-1        ; Rutina de ratardo
DELGA   LD B,04AH       ; de 100 ms para
        ADD IX,DE       ; anular rebotes al
        JP NC,LEER      ; leer del teclado.
DELGB   DJNZ DELGB      ;
        NOP             ;
        NOP             ;
        JR DELGA        ;
```

```
;*********************************************************************
;*****          RUTINA DE EJECUCION DE TRAYECTORIA 1          ******
;*********************************************************************

EJEC1   LD A,00FFH      ; Apaga leds indicando
        OUT(2),A        ; operación realizandose .
        LD HL,0840H     ; Loc. 0840H se almacena num.veces
LEEV1   IN A,(1)        ; Lectura del num.veces de ejec.
        BIT 1,A         ; Si tecleo 1
        JP Z,VECES11    ; salta a VECES11
        BIT 2,A         ; Si tecleo 2
        JP Z,VECES21    ; salta a VECES21
        BIT 3,A         ; Si tecleo 3
        JP Z,VECES31    ; salta a VECES31
        BIT 4,A         ; Si tecleo 4
        JP Z,VECES41    ; salta a VECES41
        BIT 5,A         ; Si tecleo 5
        JP Z,VECES51    ; salta a VECES51
        JP LEEV1        ; Si no tecleo alguno de los
                        ; anteriores salta a LEEV1


VECES11 LD (HL),1    ; Carga en HL el num.veces
        JP RETAR1       ; (1) y salta a RETAR1

VECES21 LD (HL),2     ; Carga en HL el num.veces
        JP RETAR1       ; (2) y salta a RETAR1

VECES31 LD (HL),3     ; Carga en HL el num.veces
        JP RETAR1       ; (3) y salta a RETAR1

VECES41 LD (HL),4     ; Carga en HL el num.veces
        JP RETAR1       ; (4) y salta a RETAR1

VECES51 LD (HL),5     ; Carga en HL el num.veces
        JP RETAR1       ; (5) y salta a RETAR1


RETAR1  LD A,00FFH      ; Saca por el puerto C la palabra que
        OUT (2),A       ; apaga los motores durante el retardo
        LD HL,2000      ; Carga HL con 2000 para
        LD DE,-1        ; que la rutina dure 2
DEL15   LD B,04AH       ; segundos y le de al
        ADD HL,DE       ; usuario tiempo para
        JP NC,INITR1    ; retirarse del auto.
DEL10   DJNZ DEL10      ; Al terminar la rutina
        NOP             ; de retardo salta a
        NOP             ; INITR1 para ejecutar
        JR DEL15        ; la trayectoria 1 .
```

```
INITR1  LD BC,07FFH     ; Carga BC con loc.inic.tray1.
ESCRI1  INC BC          ; Incrementa BC .
        LD A,(BC)       ; Carga A con clave de trayectoria .
        CP 7            ; Si clave es 7
        JP Z,DER1E      ; salta a DER1E
        CP 6            ; Si clave es 6
        JP Z,FINEJ1     ; salta a FINEJ1
        CP 5            ; Si clave es 5
        JP Z,CINCO1E    ; salta a CINCO1E
        CP 4            ; Si clave es 4
        JP Z,CUATRO1E   ; salta a CUATRO1E
        CP 3            ; Si clave es 3
        JP Z,TRES1E     ; salta a TRES1E
        CP 2            ; Si clave es 2
        JP Z,DOS1E      ; salta a DOS1E
        CP 1            ; Si clave es 1
        JP Z,UNO1E      ; salta a UNO1E
        CP 0            ; Si clave es 0
        JP Z,IZQ1E      ; salta a IZQ1E



CINCO1E LD HL,5000   ; Carga HL con 5000
        JP SACA1     ; Salta a SACA1

CUATRO1E LD HL,4000  ; Carga HL con 4000
         JP SACA1    ; Salta a SACA1

TRES1E  LD HL,3000   ; Carga HL con 3000
        JP SACA1     ; Salta a SACA1

DOS1E   LD HL,2000   ; Carga HL con 2000
        JP SACA1     ; Salta a SACA1

UNO1E   LD HL,1000   ; Carga HL con 1000
        JP SACA1     ; Salta a SACA1
```

```
DER1E   LD A,00FAH      ; Saca por el puerto
        OUT (2),A       ; C la palabra que
        LD HL,1000      ; activa el motor de
        LD DE,-1        ; direcciones a la
DELD15  LD B,04AH       ; derecha durante un
        ADD HL,DE       ; segundo.
        JP NC,REESCRI1  ; Al transcurrir un
DELD10  DJNZ DELD10     ; segundo retorna a
        NOP             ; ESCRI1 para ejecu-
        NOP             ; tar otro comando.
        JR DELD15       ;


IZQ1E   LD A,00F9H      ; Saca por el puerto
        OUT (2),A       ; C la palabra que
        LD HL,1000      ; activa el motor de
        LD DE,-1        ; direcciones a la
DELI15  LD B,04AH       ; izquierda durante un
        ADD HL,DE       ; segundo.
        JP NC,REESCRI1  ; Al transcurrir un
DELI10  DJNZ DELI10     ; segundo retorna a
        NOP             ; ESCRI1 para ejecutar
        NOP             ; otro comando.
        JR DELI15       ;


SACA1   LD A,00FBH      ; Saca por el puerto
        OUT (2),A       ; C la palabra que
        LD DE,-1        ; activa el motor de
DELA15  LD B,04AH       ; avance durante un
        ADD HL,DE       ; tiempo proporcional
        JP NC,REESCRI1  ; al valor de HL. p.e.
DELA10  DJNZ DELA10     ; si HL = 1000, durara
        NOP             ; 1 segundo. Al termi-
        NOP             ; nar retorna a ESCRI2
        JR DELA15       ; para ejecutar otro
                        ; comando.

REESCRI1  LD B,08H      ;RECUPERA LA LOC.INIC.
          JP ESCRI1     ;PARA SEGUIR LA TRAY1.


FINEJ1  LD HL,0840H     ; Termino una ejecucion
        DEC (HL)
        JP Z,INICIO     ; Si (HL) es cero, salta a INICIO
        JP RETAR1       ; Si (HL) <> 0 regresa a RETAR1
```

```
;*********************************************************************
;******          RUTINA DE EJECUCION DE TRAYECTORIA 2          ******
;*********************************************************************

EJEC2   LD A,00FFH      ; Apaga leds indicando
        OUT(2),A        ; operación realizandose .
        LD HL,087FH     ; Loc. 087FH almacena num.veces
LEEV2   IN A,(1)        ; Lectura del num.veces ejec.
        BIT 1,A         ; Si tecleo 1
        JP Z,VECES12    ; salta a VECES12
        BIT 2,A         ; Si tecleo 2
        JP Z,VECES22    ; salta a VECES22
        BIT 3,A         ; Si tecleo 3
        JP Z,VECES32    ; salta a VECES32
        BIT 4,A         ; Si tecleo 4
        JP Z,VECES42    ; salta a VECES42
        BIT 5,A         ; Si tecleo 5
        JP Z,VECES52    ; salta a VECES52
        JP LEEV2        ; Si no tecleo alguno de los
                        ; anteriores salta a LEEV2


VECES12 LD (HL),1       ; Carga en HL el num.veces
        JP RETAR2       ; (1) y salta a RETAR2

VECES22 LD (HL),2       ; Carga en HL el num.veces
        JP RETAR2       ; (2) y salta a RETAR2

VECES32 LD (HL),3       ; Carga en HL el num.veces
        JP RETAR2       ; (3) y salta a RETAR2

VECES42 LD (HL),4       ; Carga en HL el num.veces
        JP RETAR2       ; (4) y salta a RETAR2

VECES52 LD (HL),5       ; Carga en HL el num.veces
        JP RETAR2       ; (5) y salta a RETAR2



RETAR2  LD A,00FFH      ; Saca por el puerto C la palabra que
        OUT (2),A       ; apaga los motores durante el retardo
        LD HL,2000      ; Carga HL con 2000 para
        LD DE,-1        ; que la rutina dure 2
DEL25   LD B,04AH       ; segundos y le de al
        ADD HL,DE       ; usuario tiempo para
        JP NC,INITR2    ; retirarse del auto.
DEL20   DJNZ DEL20      ; Al terminar la rutina
        NOP             ; de retardo salta a
        NOP             ; INITR2 para ejecutar
        JR DEL25        ; la trayectoria 2 .          PROG.ASM  8/10
```

```
INITR2  LD BC,0840H     ; Carga BC con loc.inic.tray2.
ESCRI2  INC BC          ; Incrementa BC .
        LD A,(BC)       ; Carga A con clave de trayectoria .
        CP 7            ; Si clave es 7
        JP Z,DER2E      ; salta a DER2E
        CP 6            ; Si clave es 6
        JP Z,FINEJ2     ; salta a FINEJ2
        CP 5            ; Si clave es 5
        JP Z,CINCO2E    ; salta a CINCO2E
        CP 4            ; Si clave es 4
        JP Z,CUATRO2E   ; salta a CUATRO2E
        CP 3            ; Si clave es 3
        JP Z,TRES2E     ; salta a TRES2E
        CP 2            ; Si clave es 2
        JP Z,DOS2E      ; salta a DOS2E
        CP 1            ; Si clave es 1
        JP Z,UNO2E      ; salta a UNO2E
        CP 0            ; Si clave es 0
        JP Z,IZQ2E      ; salta a IZQ2E


CINCO2E LD HL,5000      ; Carga HL con 5000
        JP SACA2        ; Salta a SACA2

CUATRO2E LD HL,4000     ; Carga HL con 4000
         JP SACA2       ; Salta a SACA2

TRES2E  LD HL,3000      ; Carga HL con 3000
        JP SACA2        ; Salta a SACA2

DOS2E   LD HL,2000      ; Carga HL con 2000
        JP SACA2        ; Salta a SACA2

UNO2E   LD HL,1000      ; Carga HL con 1000
        JP SACA2        ; Salta a SACA2
```

```
DER2E   LD A,00FAH      ; Saca por el puerto
        OUT (2),A       ; C la palabra que
        LD HL,1000        activa el motor de
        LD DE,-1        ; direcciones a la
DELD25  LD B,04AH       ; derecha durante un
        ADD HL,DE       ; segundo.
        JP NC,REESCRI2    ; Al transcurrir un
DELD20  DJNZ DELD20     ; segundo retorna a
        NOP             ; ESCRI2 para ejecu-
        NOP             ; tar otro comando.
        JR DELD25       ;


IZQ2E   LD A,00F9H      ; Saca por el puerto
        OUT (2),A       ; C la palabra que
        LD HL,1000      ; activa el motor de
        LD DE,-1        ; direcciones a la
DELI25  LD B,04AH       ; izquierda durante un
        ADD HL,DE       ; segundo.
        JP NC,REESCRI2    ; Al transcurrir un
DELI20  DJNZ DELI20     ; segundo retorna a
        NOP             ; ESCRI2 para ejecutar
        NOP             ; otro comando.
        JR DELI25       ;


SACA2   LD A,00FBH      ; Saca por el puerto
        OUT (2),A       ; C la palabra que
        LD DE,-1        ; activa el motor de
DELA25  LD B,04AH       ; avance durante un
        ADD HL,DE       ; tiempo proporcional
        JP NC,REESCRI2    ; al valor de HL. p.e.
DELA20  DJNZ DELA20     ; si HL = 5000, durara
        NOP             ; 5 segundos. Al ter-
        NOP             ; minar retorna a ESCRI2
        JR DELA25       ; para ejecutar otro
                        ; comando.


REESCRI2  LD B,08H      ;RECUPERA LA LOC.INIC.
          JP ESCRI2     ;PARA SEGUIR LA TRAY2.


FINEJ2  LD HL,087FH
        DEC (HL)        ; Termino una ejecucion, decrementa IX
        JP Z,INICIO     ; Si HL es cero, salta a INICIO
        JP RETAR2       ; Si HL <> 0, regresa a RETAR2
```

```
{ Programa que convierte un archivo hexadecimal a ascii }


program hextoasc;
var
  asc: char;
 cod1: char;
 cod2: char;
f1,f2: text;
 Reg1: string;
 Reg2: string;
filename1: string;
filename2: string;
TamReg1: integer;
I: byte;
{************************************}
function hextodec(hex: char): byte;
var
  dec: byte;
begin
 case hex of
      '0': dec:=  0;    '1': dec:=  1;    '2': dec:=  2;    '3':
dec:=  3;
      '4': dec:=  4;    '5': dec:=  5;    '6': dec:=  6;    '7':
dec:=  7;
      '8': dec:=  8;    '9': dec:=  9;    'A': dec:= 10;    'B':
dec:= 11;
      'C': dec:= 12;    'D': dec:= 13;    'E': dec:= 14;    'F':
dec:= 15;
 end;
 hextodec:= dec;
end;
{*****************************************}
function convert(cod1,cod2: char): char;
var
   decimal: byte;
begin
 decimal:= 16 * hextodec(cod1) + hextodec(cod2);
 convert:= chr(decimal);
end;
{*************************************************}
```

```
begin
write('Enter Input FileName:');
readln(filename1);
{$I-}
Assign(f1,filename1);
reset(f1);
if IOResult <> 0 then
  begin
   writeln('File not found');
   halt(1);
  end;
{$I+}
write('Enter Output FileName:');
readln(filename2);
Assign(f2,filename2);
rewrite(f2);
while (not EOF (f1)) do
begin
 readln(f1,Reg1);
 Reg1:= copy(Reg1, 10, Length(Reg1)-11);
 TamReg1:= Length(Reg1);
 Reg2:= '';
for I:= 1 to TamReg1 do
 begin
   cod1:= Reg1[I];
   cod2:= Reg1[I+1];
   asc:= convert(cod1,cod2);
   {Reg2:= Reg2 + asc;}
   write(f2,asc);
   inc (I);
 end;
{write(f2,Reg2);}
end;
close(f2);

end.
```

6. PRUEBAS MODULARES

6.6 PRUEBAS MODULARES

La arquitectura del sistema permitió la realización de pruebas por módulo en forma independiente. Esto significa que al realizar pruebas a un módulo determinado no fué necesario que los módulos restantes estuvieran operando. Tampoco fué necesario seguir un orden preestablecido para las pruebas.

Las pruebas por módulo fueron las siguientes:

a) Reloj
Se armó el circuito de reloj y se hicieron mediciones de la frecuencia generada en un osciloscopio. Las lecturas obtenidas fueron satisfactorias, ya que la frecuencia de la señal fué de 1MHz y la forma de onda cuadrada. Cabe señalar que éste circuito aislado presentó estas características y al integrar todos los módulos se observó la presencia de ruido en la señal. Para reducir el ruido se colocó entre los pines de Vcc y tierra de cada uno de los chips un capacitor.

b) Teclado
A partir de una calculadora se modificó su circuitería para obtener el teclado del sistema. Se realizaron pruebas de continuidad a cada una de las teclas mediante un multímetro para garantizar su funcionamiento correcto al momento de integrar los módulos.

c) Mecanismo_Motores
La parte mecanica se construyó a partir de un carro de baterias con motor de avance y motor de direcciones. El cable que alimenta los motores estaba colocado a un portapilas especial. Por lo tantos se separó del portapilas para adaptarlo a nuestros requerimientos. Se realizaron pruebas de continuidad para identificar los cables de alimentación de cada uno de los motores. También se probó la operación de los motores mediante una pila de 1.5 volts como fuente de alimentación.

d) Etapa de Potencia
Se armó la etapa de potencia y con motores aislados (independientes a los del mecanismo) se probó la operación de los dos circuitos integrados M54543. Se simuló la generación de "1" y"0" del CPU a los motores enviando las entradas de la etapa de potencia a el Vcc y a tierra del sistema digital respectivamente. Se comprobó que los motores se activan de acuerdo con las entradas de la etapa de potencia que se simulaban.

e) Software del sistema (Programa)
El programa obtenido en el punto 5.5 se probó con el simulador AVSIMZ80. Este simulador permitió verificar la funcionalidad del programa que opera al sistema y que reside en la memoria Eprom del mismo.

7. IMPLANTACION DEL PROTOTIPO

7.1 Integración Hardware - Software

7.2 Montaje de los dispositivos

7. IMPLANTACION DEL PROTOTIPO

7.1 Integración Hardware-Software

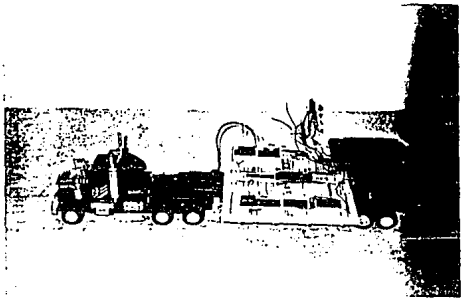
Durante la etapa de diseño del hardware y software, éstos fueron inicialmente particionados en módulos. Después del diseño y de las pruebas modulares es necesario integrar los diferentes módulos que componen el sistema.
Para aumentar el rango de confiabilidad de la operación del sistema se realizaron las siguientes pruebas interconectando más de un módulo:

a) Etapa de potencia - Motores
   Se conectaron los motores a la salida de la etapa de potencia y a la entrada de la etapa de potencia se simularon los "1" y "0" enviando al Vcc del sistema digital y a la tierra del sistema respectivamente dichas entradas. La activación de los motores cumplió con lo planteado en el diseño.

b) Circuitería CPU - Memoria - PPI - Reloj
   Después de haber armado y probado los módulos del CPU, Memoria, PPI, y Reloj en forma independiente, se integraron junto con la etapa de potencia y el teclado para la realización de pruebas del prototipo preliminares al montaje de los dispositivos.
   Estas pruebas consistieron en la grabación y ejecución de trayectorias (ver manual de usuario, apendice I) estando los circuitos sobrepuestos y simulando la activación de los motores con leds conectados a la entrada de la etapa de potencia para validar la lógica de la programación. Cabe destacar que se dejaron para pruebas futuras las consideraciones de velocidad de los motores.

7.2 Montaje de los dispositivos

En éste punto se ensambló el circuito digital al mecanismo de transporte. El circuito requirió dos tabletas, las cuáles se montaron en la parte superior del mecanismo.
La fuente de alimentación del sistema digital consta de una pila de 9 volts colocada en la parte inferior del mecanismo, éste voltaje es regulado a 5 volts mediante el circuito integrado LM7805.
Para la fuente de alimentación de los motores del mecanismo fué necesario un portapilas para las cuatro pilas de 1.5 volts tipo A. Este tipo de pilas fué seleccionado debido a que proporcionar una corriente adecuada para la etapa de potencia que alimenta los motores. El portapilas fué ensamblado al mecanismo mediante tornillos en su parte posterior y permite al usuario un acceso rápido para el reemplazo de las baterias.
El teclado fué colocado en la tapa del portapilas para facilitar al usuario la programación de las trayectorias. Los leds de monitoreo para la selección de operación y selección de trayectoria fueron montados en el teclado dando una ergonomía adecuada al sistema.



IMPLANTACION DEL PROTOTIPO

8. PRUEBAS DEL PROTOTIPO

8. PRUEBAS DEL PROTOTIPO

Una vez integrados todos los módulos del sistema y montados en el mecanismo se realizaron las siguientes pruebas al prototipo:

a) Grabación de la trayectoria 1
Al encender el sistema se encendió el led indicador de operación, lo cual permite elegir entre la operación de grabación o ejecución. Se eligió la operación de grabación. Se apagó el led de selección de operación y se encendió el led indicador de trayectoria. Se seleccionó la trayectoria 1 y automáticamente se apago el led. Como no había ningún led encendido se asumió que el sistema estaba listo para recibir y grabar comandos. Entonces se teclearon los comandos de una trayectoria predeterminada. Se presionó la tecla de Fin para dar por terminada la grabación de la trayectoria y se encendió el led de selección de operación.

a) Ejecución de la trayectoria 1
Después de haber grabado la trayectoria 1 en el inciso anterior se encontraba encendido el led de selección de operación. Se seleccionó la operación de ejecución, se apagó automáticamente el led de selección de operación y se encendió el led de selección de trayectoria. Se seleccionó la trayectoria 1 y se apagó el led de trayectoria. En este punto ningún led se encendió y se debía de teclear el número de veces (1 a 5) que se deseaba ejecutar la trayectoria. Para abreviar las pruebas se eligío ejecutar la trayectoria una vez.
Mediante los leds de monitoreo de la ejecución se observó que los comandos grabados en el inciso anterior correspondían con los ejecutados por el sistema. Sin embargo los motores no se activaban. El mismo procedimiento se realizó para la prueba de grabación y ejecución de la trayectoria 2 y se obtuvo el mismo resultado.

Se pensó que el problema podría haber estado en la etapa de potencia puesto que los leds para el monitoreo de la ejecución de trayectorias están a la entrada de la etapa de potencia. Se realizaron nuevamente pruebas a la misma pero se comprobó su correcta funcionalidad. Enfocando la atención en los leds de monitoreo se pudó percibir que éstos mostraban muy rápido los comandos grabados. Esto permitió deducir que las señales enviadas por el sistema a la entrada de la etapa de potencia tenían una duración corta la cuál no permitía la activación de los motores.
Se incrementó el tiempo de las rutinas de retardo en el programa. Fué necesario borrar y regrabar el programa en la eprom y repitir las pruebas de los incisos a) y b) varias ocasiones hasta obtener una respuesta adecuada de los motores.
Con ésto se dió por terminada la conformación definitiva del prototipo.

9. CONCLUSIONES

9. CONCLUSIONES

La automatización del transporte dentro de una industria basado en un sistema digital ofrece entre otras las siguientes ventajas:

- Incremento de la productividad
- Reducción de costos totales
- Alta calidad en servicio y produccción

La experiencia adquirida durante el desarrollo e implantación de éste prototipo facilitaría la realización de un sistema de acuerdo a las necesidades y requerimientos de cada empresa-industria en particular.

De acuerdo a lo anterior se tendría flexibilidad para adecuar las siguientes características del sistema.

- Las dimensiones, material, y forma del mecanismo, además del peso y naturaleza de los insumos a transportar son independientes de la circuitería del sistema, excepto por la etapa de potencia que deberá ser de acuerdo con la potencia de los motores del mecanismo a utilizar.
- El número de trayectorias se puede modificar sin grandes cambios en el diseño del software. El número de trayectorias tiene relación con el tamaño de la memoria.
- El número de comandos por trayectoria se puede modificar sin grandes cambios en el diseño del software. El número de comandos por trayectoria tiene una relación directa con el tamaño de la memoria.
- En caso de que las trayectorias sean fijas, estas pueden ser grabadas previamente por el desarrollador del sistema en memoria eprom, evitandole al usuario final la programación de trayectorias.

El prototipo presentado es una propuesta que permitiría a las empresas-industria (producción y servicios) obtener una infraestructura de transporte automatizado a bajo costo con una alta funcionalidad.

APENDICES

APENDICE I. MANUAL DE USUARIO

a) Inicialmente el usuario puede elegir en el teclado una operación a realizar por el sistema. Esta operación podrá ser de:

- Grabación
- Ejecución

Un Led indica a el usuario que se encuentra en éste punto y que debe de elegir una de las 2 operaciones. En caso de teclear cualquier otra tecla que no sea Grabación o Ejecución se hará caso omiso de la tecla presionada. Podrá pasar al siguiente inciso hasta que se presione una de las 2 teclas de operación (Grabación o Ejecución).

b) Después de haber seleccionado una de las dos operaciones (Grabación o Ejecución), el Led indicador de selección de operación se apagará y se encenderá un Led que le indica a el usuario que debe seleccionar una de dos trayectorias:

- Trayectoria 1
- Trayectoria 2

Sí el usuario presionó otra tecla diferente a Trayectoria 1 o Trayectoria 2, el sistema hará caso omiso y seguirá esperando la presión de una de las dos teclas de trayectoria.

c) En este punto no hay Leds indicadores, entonces al estar apagados los dos Leds con que cuenta el sistema, el usuario deberá asumir que se encuentra en éste punto.
Dependiendo de la selección de operación (Grabación o Ejecución) que se haya realizado en el inciso a), se tienen las siguientes 2 opciones:

- Sí se seleccionó la operación de Grabar, el sistema está listo para aceptar comandos de grabación de la trayectoria seleccionada en el inciso b). Esta grabación de la trayectoria se realiza a travez del teclado y se dará por terminada una vez que se presione la tecla "FIN". Entonces el sistema regresará al inicio del inciso a) para aceptar una nueva operación.
Es importante mencionar que sí el usuario seleccionó en el punto b) una trayectoria que tiene una trayectoria grabada anteriormente, dicha grabación se borrará y se sustituirá por la nueva.

- Sí se seleccionó la operación Ejecutar, el sistema espera que el usuario presione en el teclado una de las teclas del "1" al "5". Esta tecla es el número de veces que la trayectoria seleccionada en el inciso b) será ejecutada por el sistema. Se tendrá un tiempo de espera de 5 segundos para que se comienze a ejecutar automáticamente la trayectoria, esto, para que el usuario tenga tiempo de retirarse del dispositivo antes de que avance.
Cuando la trayectoria haya sido ejecutada el número de veces que se tecleó, el dispositivo se detendrá y el sistema regresará al inciso a) para una nueva operación.

Grabación de una trayectoria:

Con un avance unitario de 12 cm, las teclas 1 a 5 representan un avance proporcional a este numero. Esto sigifica que para la tecla 1 se tendrá un avance de 12 cm, para la tecla 2 el avance será de 24 cm., y así sucesivamente.
La tecla D representa un giro a la derecha con un angulo de 45 grados mientras que la tecla I representa un giro a la izquierda con un angulo de 45 grados. Se puede girar hasta 90 grados al grabar 2 veces seguidas la tecla correspondiente a la dirección del giro deseado.

Por ejemplo, supongase que se requiere que el carro se desplace del punto Inicial 1 al punto Destino 3 (ver fig.1) y que el punto 2 es un punto intermedio que el carro tiene que atravesar para salvar ciertos obstaculos en su camino. Para que el carro se desplace del punto 1 al punto 2 es necesario introducir el siguientes comando: **4** y del punto 2 al punto 3 los comandos: **D 5 5** .

Para que el carro regrese del punto 3 al punto inicial 1 y se ejecute tantas veces como sea necesario este recorrido se ha añadido a la trayectoria inicial del punto 1 al punto 3 una segunda parte (ver fig.2) que comprende un giro de 45 grados a la derecha en el punto 3, un avance de 36 cm, un giro de 90 grados a la derecha en el punto 4, un avance de 120 cm, un giro de 90 grados a la derecha en el punto 5, un avance de 132 cm y un giro de 90 grados a la derecha en el punto 1. Los comandos necesarios para esta segunda parte son: **D, 3, D D, 5 5, D D, 5 5 1, D D** respectivamente.

Entonces para la grabación de la trayectoria se tendrán que teclear las secuencias de comandos de ambas partes de la trayectoria total:

4 D 5 5 D 3 D D 5 5 D D 5 5 1 D D

Nota: cualquier comando de avance (teclas de 1 a 5) puede ser descompuesta en varios siempre que la suma de los comandos sea igual al comando.
Es importante mencionar que para desplazamientos menores a 12 cm o que no sean multiplos de 12 cm se modificaría el Software del sistema, especificamente en las rutinas de retardo que mantienen los motores activos por un tiempo proporcional al valor del comando de avance.

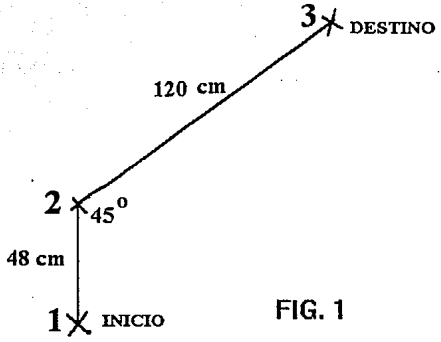

3
DESTINO
120 cm
2
45°
48 cm
1
INICIO

FIG. 1

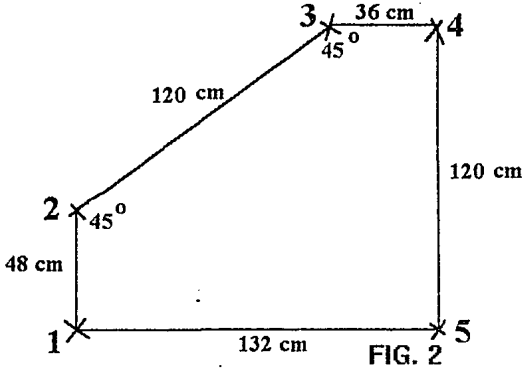

3
36 cm
4
45°
120 cm
120 cm
2
45°
48 cm
1
132 cm
5

FIG. 2

APENDICE II. LISTA DE MATERIAL

1 CPU Z-80 (1MHz)
1 Memoria Eprom 2716
1 Memoria Ram 6810
1 PPI 8255
1 Hex Bus Driver 74LS368
2 Bi-directional Motor Drive ECG1628
1 Hex Inverter 74LS05
2 Capacitores 300 pf
3 Capacitores 3 mf
2 Resistencias 10 Kohms
1 Teclado
5 Led's
4 Pilas 1.5 volts topo B
1 Portapilas
1 Pila 9 volts
1 Broche para pila cuadrada
2 Tabletas Project-Board
12 Resistencias 1 Kohms

Equipo requerido

Osciloscopio
Multimetro
Grabador de memoria
Borrador de memoria
Fuente de poder 5 volts
Pinzas

Software requerido

Simulador Z-80
Ensamblador
Editor

APENDICE III. GLOSARIO

ACUMULADOR: Registro de proposito especial que contiene los resultados de 8 bits de instrucciones aritmeticas y logicas.

ANALIZADOR LOGICO: Una pieza de equipo de texto que cierra y almacena el estado de estas entradas para cada transicion de reloj.

ASCII: (American Standard Code for Information Interchange). Un codigo de 7 bits que representa digitos decimales, letras del alfabeto, simbolos y caracteres de control.

BANDERAS: Bits en el registro de banderas que indican varias condiciones del resultado de una operacion.

BANDERA DE ACARREO (C): Un bit del registro de banderas, de las operaciones que afectan a la bandera de acarreo (C), C se coloca si un acarreo del bit mas significativo ocurre.

BANDERA DE CERO: Un bit del registro de banderas, una operacion que afecta a la bandera de cero, la bandera de cero se enciende si el resultado de la operacion es cero.

BANDERA DE MEDIO-ACARREO (H): Un bit del registro de banderas, dentro de las operaciones que afectan a la bandera H , H se coloca cuando resulta un acarreo de 3 a 4 bits.

BANDERA DE PARIDAD/OVERFLOW (P/V): Un bit del registro de banderas que monitorea el overflow (V) en complemento 2's, monitorea paridad (P) en operaciones logicas.

BANDERA DE SIGNO (S): Un bit del registro de banderas, sobre las operaciones que afectanb a la bandera de signo, la bandera de signo monitorea el bit mas significativo del resultado.

BCD (Binary Coded Decimal): Codigo Decimal Binario

BIT: Una contraccion de binary digit. Cada lugar en un numero binario es un bit; por ejemplo 1011 es un numero de cuatro bits.

BUS BIDIRECCIONAL: Un bus, semejante al bus de datos, que carga informacion en ambas direcciones.

BUS UNIDIRECCIONAL: Un bus, como el bus de direccion, que transfiere datos en una sola direccion.

BUFFER: Un circuito cuya funcion es suministrar mayor corriente de salida de la que maneja la entrada.

BUS: Un conjunto de lineas que transmiten informacion en paralelo, una linea por cada bit.

BUS DE DATOS: Los circuitos que generan, guardan, usan, introducen o tienen salidas de datos son conectadas al bus de datos. Se proporciona una linea para bit de dato.

BUS DE DIRECCION: Un juego de lineas en un sistema digital dedicadas a transmitir la direccion en memoria o puerto a ser usado.

BUS DRIVER (Manejador de Bus): Un buffer que proporciona un manejo de corriente suficiente de salida para manejar una linea del bus.

BYTE: Un numero binario de 8 bits de largo.

CHIP SELECT: Una señal utilizada para habilitar un circuito integrado.

CICLO DE MAQUINA: Conjunto de estados T necesarios para completar una funcion particular.

CICLO DE MAQUINA DE ESCRITURA DE ENTRADA/SALIDA: Un grupo de estados T usados para la salida del dato a un puerto de salida.

CICLO DE MAQUINA DE LECTURA DE ENTRADA/SALIDA: Un grupo de estados T usados para introducir un byte de datos a un puerto de salida.

CICLO DE MAQUINA DE LECTURA DE MEMORIA: Un grupo de estados T usados para cargar un byte de dato de memoria dentro de la CPU.

CICLO DE MAQUINA DE ESCRITURA DE MEMORIA: Un grupo de estados T usados para cargar un byte de dato de la CPU a memoria.

CICLO DE MAQUINA DE RECONOCIMIENTO DE INTERRUPCION: Grupo de estados T usados en la entrada de la siguiente instruccion de un puerto de entrada.

CICLO DE MAQUINA M1(Ciclo de maquina de instruccion Fetch): Un grupo de estados T usados para cargar el primer byte de una instruccion de memoria al CPU.

CLOCK (Reloj): Una forma de onda rectangular usada para conteo.

CMOS (Complementary metal-oxide semiconductor): Una de las tegnologias usadas en la manufactura de circuitos integrados.

CODIGO DE OPERACION: Un numero binario, usualmente representado en hexadecimal que es reconocido por el CPU como una instruccion.

CODIGO FUENTE: Un programa escrito en codigo nmemonico (ensamblador).

CODIGO NMEMONICO: Un codigo asignado para cada codigo de operacion en el conjunto de instrucciones que facilita programacion y memoria.

CODIGO OBJETO: Un programa escrito o listado en codigo de operacion.

CONTADOR DE PROGRAMA (PC): Registro de 16 bits en el Z-80 que contiene la direccion de la proxima instruccion del programa a ser implementada.

DEBUG: Descubrimiento y correccion de errores en programas.

DEBUGGER SOFTWARE: Un programa que simula la corrida de un programa bajo prueba en el ambiente hardware dentro del cual va a operar.

DECODIFICADOR DE DIRECCIONES: Un circuito que genera un pulso (chip select) de la direccion presente en el bus de direcciones.

DECREMENTO: Decremento en 1.

DESTINO: Localidad a la que el dato es transferido via una instruccion de carga.

DIRECCIONAMIENTO IMPLICITO: Uno de los modos de direccionamiento. El codigo de operacion implicito en el cual esta un registro interno para ser usado en la operacion.

DIRECCIONAMIENTO INDEXADO: Uno de los modos de direccionamiento. Un byte de desplazamiento en forma de complemento a 2's es adicionado a uno de los registros indexados y esta incluido en la instruccion.

DIRECCIONAMIENTO INMEDIATO: Uno de los modos de direccionamiento, la instruccion contiene 8 bits de datos que son usados como un operando.

DIRECCIONAMIENTO INMEDIATO EXTENDIDO: Uno de los modos de direccionamiento. La instruccion contiene 16 bits de datos que son usados como un operando.

DIRECCIONAMIENTO POR REGISTRO: Uno de los modos de direccionamiento, un codigo de tres bits que especifica cual registro interno es usado en la operacion.

DIRECCIONAMIENTO POR REGISTRO INDIRECTO: Uno de los modos de direccionamiento. Un registro o registro par apunta al operando utilizado.

DIRECCIONAMIEMTO RELATIVO: Uno de los modos de direccionamiento, la instruccion contiene un byte de desplazamiento el cual sera utilizado en un salto relativo.

DMA (Direct Memory Access): Acceso Directo a Memoria, Transferencia de datos directamente entre la memoria y los perifericos sin pasar a traves del CPU.

DRAM (Dinamic RAM): RAM dinanica que debe de ser refrescada periodicamente para mantener el almacenamiento de datos.

EEPROM (Electrically Erasable programable read only memory): Una ROM que puede ser borrada electricamente y reprogramada.

EMULADOR DE CPU: Un circuito utilizado para probar un sistema con procesador. El CPU es remplazado por el emulador para prueba.

ENSAMBLADOR: Un programa que convierte codigo de nmenicos en codigo de operacion (codigo fuente en codigo objeto).

ENTRADA ACTIVA ALTA: Un circuito de entrada que esta buscando o esperando un uno que origina la activacion del circuito o funcion.

ENTRADA ACTIVA BAJA: Un circuito de entrada que esta buscando o esperando un 0 que causa la activacion del circuito o funcion.

EPROM (Erasable Programable Read Only Memory): Una ROM que puede ser borrada con luz ultravioleta y reprogramada.

ESTADO T: Un periodo de reloj.

GRUPO DE REGISTROS PRINCIPALES: Este conjunto de registros en el Z-80 esta formado por A, F, B, C, D , E ,H ,L.

H (HEXADECIMAL): En este texto H es usado para indicar que el numero precedente esta escrito en hexadecimal.

HARDWARE: Los componentes fisicos de un sistema digital.

INCREMENTO: Incremento en 1

INSTRUCCION DE REINICIO: Una instruccion que provoca al CPU a tomar la instruccion de una localidad especifica en pagina cero (byte alto= a 00h).

INSTRUCCION FETCH (Busqueda): Ciclo de maquina M1. Un grupo de estados T para cargar el primer byte de una instruccion de memoria al CPU.

INTERFACE PARALELA: Un circuito utilizado para la comunicacion entre el bus de datos y un dispositivo externo, 8 bits a la vez.

INTERRUPCION: Señal de un dispositivo externo que requiere servicio.

INTERRUPCION MASCARABLE: Una interrupcion que puede ser inhibida por el programador.

INTERRUPCION NO MASCARABLE: Una interrupcion que no puede ser inhibida por el programador.

FIRMWARE: Un programa almacenado en una memoria de solo lectura.

FUENTE: Localidad de dato a ser transferida via una instruccion de carga.

FUENTE DE CORRIENTE: Proporciona un camino de corriente convencional para que fluya desde la fuente de energia.

LARGE-SCALE INTEGRATION (LSI): Un circuito integrado que contiene circuiteria equivalente a 100 compuertas o mas

LENGUAJE DE ALTO NIVEL: Un lenguaje de programacion en el cual una declaracion traduce una serie de codigos de operacion.

LENGUAJE ENSAMBLADOR: Un lenguaje de programacion compuesto de codigo de nmemonicos.

MEMORIA: Un arreglo de celdas que pueden almacenar numeros binarios.

MEMORIA NO VOLATIL: Memoria en la cual el contenido no es destruido cuando la energia es removida.

MEMORIA VOLATIL: Memoria que pierde sus datos almacenados cuando la energia es removida.

MODO 0 DE INTERRUPCION: Sistema de interrupcion en el cual el codigo de operacion de la proxima instruccion es para realizarse esta entrada en el bus de datos.

MODO 1 DE INTERRUPCION: Sistema de interrupcion en el cual el CPU toma la proxima instruccion de la localidad de memoria 0038H.

MODO 2 DE INTERRUPCION: Sistema de interrupcion en el cual el byte en el registro I junto con el byte de entrada del dispositivo de interrupcion apunta a la localidad de memoria que contiene el byte bajo de la proxima instruccion a ser implementada, la siguiente direccion contiene el byte alto.

MODO DE DIRECCIONAMIENTO: Metodo de especificacion de los operandos a ser usados en una operacion.

MOS (Metal-Oxide Semiconductor): Tipo de transistor usado en algunos circuitos integrados digitales.

MULTIPLEXOR: Un circuito logico que va a conmutar una de muchas entradas a una salida. La entrada que es conectada a la salida es seleccionada por un numero binario de entrada del circuito logico.

NIBBLE: Cuatro bits.

NMEMONICO: Propone auxiliar a la memoria.

OPEN COLLECTOR: Circuito con salida en circuito abierto, esto es, no conectada a alguna fuente de poder. Una resistencia externa "pull up" es usualmente conectada para suplir energia.

OPERANDO: El dato, registro o localidad de memoria que es operado por el CPU.

PALABRA LARGA: Palabras que son de 32 bits de largo.

PARIDAD: Un sistema utilizado para detectar errores en la transmision de datos binarios.

PARIDAD IMPAR: Sistema utilizado para detectar errores en datos binarios transmitidos, este usa un bit de paridad para poner el total de numeros 1's en una palabra impar.

PARIDAD PAR: Un sistema que es utilizado para detectar errores de transmision en datos binarios. Este usa un bit de paridad para poner el total de numeros 1's en una palabra par.

PERIFERICO: Dispositivo externo.

PROCESO DE CONTROL: El uso de un microprocesador para control de un sistema o manufacturade procesos.

PROM(Programmable Read Only Memory): PROM que puede ser programada una sola vez.

PUERTO: Un circuito que coloca datos en el bus de datos para uso externo, o que los puede registrar de un dispositivo externo al bus de datos cuando sea requerido.

PULL-UP: Una resistencia utilizada para proporcionar un camino a la energia suplente.

RAM (Random Acces Memory): Memoria de lectura/escritura en la cual el acceso de tiempo para un byte es constante.

RECONOCIMIENTO DE INTERRUPCION: Una señal del sistema al dispositivo externo, que indica que el CPU esta listo para procesar una interrupcion.

REFRESCO: Para actualizar el contenido de DRAM.

REGISTRO: Un juego de bits de memoria de lectura/escritura que trabajan juntos como una unidad.

REGISTROS ALTERNADOS: El juego de registros en el Z-80 consta de A',F', B',C', D', E', H', L'.

REGISTRO DE BANDERAS: Registro de proposito especial cuyos bits indican varias condiciones del resultado de una operacion.

REGISTRO DE REFRESCO DE MEMORIA: Registro interno de memoria de 8 bits, utilizado para refrescar a la RAM dinamica.

REGISTRO INDEXADO (IX): Registro interno de 16 bits que apunta a un ar de comienzo en memoria. Un byte de desplazamientoes adicionado al IX para formar la localidad exacta.

REGISTRO INDEXADO (IY): Registro interno de 16 bits que apunta a un lugar de comienzo en memoria. Un byte de desplazamiento es adicionado a IY fara formar la localidad exacta.

REGISTRO INTERNO: Registro contenido dentro del CPU.

REGISTRO VECTOR DE INTERRUPCION (I): Registro interno de 8 bits que contiene el byte alto de la direccion en memoria que contiene la proxima instruccion a ser usada despues de una interrupcion modo 2.

RESET: Una señal de entrada que causa al CPU a la busqueda de la proxima instruccion de 0000H.

RETURN: Una instruccion que causa al CPU a terminar la ejecucion de una subrutina y retornar a su posicion original en el programa principal.

ROM (Read Only Memory): Memoria de lectura durante el transcurso de un programa pero no de escritura.

RS-232: Estandar ampliamente utilizado para comunicaciones en serie.

SALIDA ACTIVA ALTA: Un cicuito de salida que es normalmente 0 y que conmuta a 1 cuando es activado por el circuito.

SALTO CONDICIONAL: Instruccion que causa al CPU a saltar a otra localidad en el programa para obtener la proxima instruccion si se encuentra cierta condicion.

SALTO INCONDICIONAL: Instruccion que causa que el cpu salte a otra localidad del programa para realizar la proxima instruccion.

SALTO RELATIVO: Una instruccion que causa al CPU a saltar hacia adelante o atras de la presente posicion en el programa para obtener la proxima instruccion.

SET DE INSTRUCCIONES: El conjunto de codigos de operacion que son reconocidos por el CPU.

SOFTWARE: La parte de programa del sistema digital.

SRAM (Static RAM): RAM que retiene sus datos sin nececidad de ser refrescada.

STACK: Una seccion de memoria donde datos en el registro pueden ser almacenados o apilados hasta que se requieran nuevamente.

STACK POINTER (SP): Un registro de 16 bits en el Z-80 que apunta a la localidad en memoria a ser usada como proxima parte del Stack.

STROBE: Una señal utilizada para habilitar un circuito integrado o circuito.

SUBRUTINA: Una porcion del programa que realiza una funcion particular.

TIEMPO DE ACCESO: Es el tiempo de cuando la memoria es seleccionada (CE va a bajo) hasta que el dato de salida estable esta presente.

TTL(Transistor-Transistor Logic): Una de las populares familias de circuitos integrados.

UNIDAD CENTRAL DE PROCESO (CPU): La unidad de control de la microcomputadora o sistema digital.

APENDICE IV. SET DE INSTRUCCIONES DEL Z-80.

(La literal d se muestra con el valor 05 en el codigo objeto)

CODIGO OBJETO	CODIGO FUENTE	CODIGO OBJETO	CODIGO FUENTE
8E	ADC A,(HL)	E620	AND n
DD8E05	ADC A,(IX+d)	CB46	BIT 0,(HL)
FD8E05	ADC A,(IY+d)	DDCB0546	BIT 0,(IX+d)
8F	ADC A,A	FDCB0546	BIT 0,(IY+d)
88	ADC A,B	CB47	BIT 0,A
89	ADC A,C	CB40	BIT 0,B
8A	ADC A,D	CB41	BIT 0,C
8B	ADC A,E	CB42	BIT 0,D
8C	ADC A,H	CB43	BIT 0,E
8D	ADC A,L	CB44	BIT 0,H
CE20	ADC A,n	CB45	BIT 0,L
ED4A	ADC HL,BC	CB4E	BIT 1,(HL)
ED5A	ADC HL,DE	DDCB054E	BIT 1,(IX+d)
ED6A	ADC HL,HL	FDCB054E	BIT 1,(IY+d)
ED7A	ADC HL,SP	CB4F	BIT 1,A
86	ADD A,(HL)	CB48	BIT 1,B
DD8605	ADD A,(IX+d)	CB49	BIT 1,C
FD8605	ADD A,(IY+d)	CB4A	BIT 1,D
87	ADD A,A	CB4B	BIT 1,E
80	ADD A,B	CB4C	BIT 1,H
81	ADD A,C	CB4D	BIT 1,L
82	ADD A,D	CB56	BIT 2,(HL)
83	ADD A,E	DDCB0556	BIT 2,(IX+d)
84	ADD A,H	FDCB0556	BIT 2,(IY+d)
85	ADD A,L	CB57	BIT 2,A
C620	ADD A,n	CB50	BIT 2,B
09	ADD HL,BC	CB51	BIT 2,C
19	ADD HL,DE	CB52	BIT 2,D
29	ADD HL,HL	CB53	BIT 2,E
39	ADD HL,SP	CB54	BIT 2,H
DD09	ADD IX,BC	CB55	BIT 2,L
DD19	ADD IX,DE	CB5E	BIT 3,(HL)
DD29	ADD IX,IX	DDCB055E	BIT 3,(IX+d)
DD39	ADD IX,SP	FDCB055E	BIT 3,(IY+d)
FD09	ADD IY,BC	CB5F	BIT 3,A
FD19	ADD IY,DE	CB58	BIT 3,B
FD29	ADD IY,IY	CB59	BIT 3,C
FD39	ADD IY,SP	CB5A	BIT 3,D
A6	AND (HL)	CB5B	BIT 3,E
DDA605	AND (IX+d)	CB5C	BIT 3,H
FDA605	AND (IY+d)	CB5D	BIT 3,L
A7	AND A	CB66	BIT 4,(HL)
A0	AND B	DDCB0566	BIT 4,(IX+d)
A1	AND C	FDCB0566	BIT 4,(IY+d)

CODIGO OBJETO	CODIGO FUENTE	CODIGO OBJETO	CODIGO FUENTE
A2	AND D	CB67	BIT 4,A
A3	AND E	CB60	BIT 4,B
A4	AND H	CB61	BIT 4,C
A5	AND L	CB62	BIT 4,D
CB63	BIT 4,E	EDB1	CPIR
CB64	BIT 4,H	EDA1	CPI
CB65	BIT 4,L	2F	CPL
CB6E	BIT 5,(HL)	27	DAA
DDCB056E	BIT 5,(IX+d)	35	DEC (HL)
FDCB056E	BIT 5,(IY+d)	DD3505	DEC (IX+d)
CB6F	BIT 5,A	FD3505	DEC (IY+d)
CB68	BIT 5,B	3D	DEC A
CB69	BIT 5,C	05	DEC B
CB6A	BIT 5,D	0B	DEC BC
CB6B	BIT 5,E	0D	DEC C
CB6C	BIT 5,H	15	DEC D
CB6D	BIT 5,L	1B	DEC DE
CB76	BIT 6,(HL)	1D	DEC E
DDCB0576	BIT 6,(IX+d)	25	DEC H
FDCB0576	BIT 6,(IY+d)	2B	DEC HL
CB77	BIT 6,A	DD2B	DEC IX
CB70	BIT 6,B	FD2B	DEC IY
CB71	BIT 6,C	2D	DEC L
CB72	BIT 6,D	3B	DEC SP
CB73	BIT 6,E	F3	DI
CB74	BIT 6,H	102E	DJNZ e
CB75	BIT 6,L	FB	EI
CB7E	BIT 7,(HL)	E3	EX (SP),HL
DDCB057E	BIT 7,(IX+d)	DDE3	EX (SP),IX
FDCB057E	BIT 7,(IY+d)	FDE3	EX (SP),IY
CB7F	BIT 7,A	08	EX AF,AF'
CB78	BIT 7,B	EB	EX DE,HL
CB79	BIT 7,C	D9	EXX
CB7A	BIT 7,D	76	HALT
CB7B	BIT 7,E	ED46	IM 0
CB7C	BIT 7,H	ED56	IM 1
CB7D	BIT 7,L	ED5E	IM 2
DC8405	CALL C,nn	ED78	IN A,(C)
FC8505	CALL M,nn	ED40	IN B,(C)
D48405	CALL NC,nn	ED48	IN C,(C)
C48405	CALL NZ,nn	ED50	IN D,(C)
F48405	CALL P,nn	ED58	IN E,(C)
EC8405	CALL PE,nn	ED60	IN H,(C)
E48405	CALL PO,nn	ED68	IN L,(C)
CC8405	CALL Z,nn	34	INC (HL)
CD8405	CALL nn	DD3405	INC (IX+d)
3F	CCF	FD3405	INC (IY+d)

CODIGO OBJETO	CODIGO FUENTE	CODIGO OBJETO	CODIGO FUENTE
BE	CP (HL)	3C	INC A
DDBE05	CP (IX+d)	04	INC B
FDBE05	CP (IY+d)	03	INC BC
BF	CP A	0C	INC C
B8	CP B	14	INC D
B9	CP C	13	INC DE
BA	CP D	1C	INC E
BB	CP E	24	INC H
BC	CP H	23	INC HL
BD	CP L	DD23	INC IX
FE20	CP n	FD23	INC IY
EDA9	CPD	2C	INC L
EDB9	CPDR	33	INC SP
DE20	SBC A,n	DB20	IN A,(n)
EDAA	IND	DD7E05	LD A,(IX+d)
EDBA	INDR	FD7E05	LD A,(IY+d)
EDA2	INI	3A8405	LD A,(nn)
EDB2	INIR	7F	LD A,A
C38405	JP nn	78	LD A,B
E9	JP (HL)	79	LD A,C
DDE9	JP (IX)	7A	LD A,D
FDE9	JP (IY)	7B	LD A,E
DA8405	JP C,nn	7C	LD A,H
FA8405	JP M,nn	ED57	LD A,I
D28405	JP NC,nn	7D	LD A,L
C28405	JP NZ,nn	3E20	LD A,n
F28405	JP P,nn	ED5F	LD A,R
EA8405	JP PE,nn	46	LD B,(HL)
E28405	JP PO,nn	DD4605	LD B,(IX+d)
CA8405	JP Z,nn	FD4605	LD B,(IY+d)
382E	JR C,e	47	LD B,A
302E	JR NC,e	40	LD B,B
202E	JR NZ,e	41	LD B,C
282E	JR Z,e	42	LD B,D
182E	JR e	43	LD B,E
02	LD (BC),A	44	LD B,H
12	LD (DE),A	45	LD B,L
77	LD (HL),A	0620	LD B,n
70	LD (HL),B	ED4B8405	LD BC,(nn)
71	LD (HL),C	018405	LD BC,nn
72	LD (HL),D	4E	LD C,(HL)
73	LD (HL),E	DD4E05	LD C,(IX+d)
74	LD (HL),H	FD4E05	LD C,(IY+d)
75	LD (HL),L	4F	LD C,A
3620	LD (HL),n	48	LD C,B
DD7705	LD (IX+d),A	49	LD C,C
DD7005	LD (IX+d),B	4A	LD C,D

CODIGO OBJETO	CODIGO FUENTE		CODIGO OBJETO	CODIGO FUENTE	
DD7105	LD	(IX+d),C	4B	LD	C,E
DD7205	LD	(IX+d),D	4C	LD	C,H
DD7305	LD	(IX+d),E	4D	LD	C,L
DD7405	LD	(IX+d),H	0E20	LD	C,n
DD7505	LD	(IX+d),L	56	LD	D,(HL)
DD360520	LD	(IX+d),n	DD5605	LD	D,(IX+d)
FD7705	LD	(IY+d),A	FD5605	LD	D,(IY+d)
FD7005	LD	(IY+d),B	57	LD	D,A
FD7105	LD	(IY+d),C	50	LD	D,B
FD7205	LD	(IY+d),D	51	LD	D,C
FD7305	LD	(IY+d),E	52	LD	D,D
FD7405	LD	(IY+d),H	53	LD	D,E
FD7505	LD	(IY+d),L	54	LD	D,H
FD360520	LD	(IY+d),n	55	LD	D,L
328405	LD	(nn),A	1620	LD	D,n
ED438405	LD	(nn),BC	ED5B8405	LD	DE,(nn)
ED538405	LD	(nn),DE	118405	LD	DE,nn
228405	LD	(nn),HL	5E	LD	E,(HL)
DD228405	LD	(nn),IX	DD5E05	LD	E,(IX+d)
FD228405	LD	(nn),IY	FD5E05	LD	E,(IY+d)
ED738405	LD	(nn),SP	5F	LD	E,A
0A	LD	A,(BC)	58	LD	E,B
1A	LD	A,(DE)	59	LD	E,C
7E	LD	A,(HL)	5A	LD	E,D
5B	LD	E,E	EDB3	OTIR	
5C	LD	E,H	ED79	OUT	(C),A
5D	LD	E,L	ED41	OUT	(C),B
1E20	LD	E,n	ED49	OUT	(C),C
66	LD	H,(HL)	ED51	OUT	(C),D
DD6605	LD	H,(IX+d)	ED59	OUT	(C),E
FD6605	LD	H,(IY+d)	ED61	OUT	(C),H
67	LD	H,A	ED69	OUT	(C),L
60	LD	H,B	D320	OUT	(n),A
61	LD	H,C	EDAB	OUTD	
62	LD	H,D	EDA3	OUTI	
63	LD	H,E	F1	POP	AF
64	LD	H,H	C1	POP	BC
65	LD	H,L	D1	POP	DE
2620	LD	H,n	E1	POP	HL
2A8405	LD	HL,(nn)	DDE1	POP	IX
218405	LD	HL,nn	FDE1	POP	IY
ED47	LD	I,A	F5	PUSH	AF
DD2A8405	LD	IX,(nn)	C5	PUSH	BC
DD218405	LD	IX,nn	D5	PUSH	DE
FD2A8405	LD	IY,(nn)	E5	PUSH	HL
FD218405	LD	IY,nn	DDE5	PUSH	IX
6E	LD	L,(HL)	FDE5	PUSH	IY

CODIGO OBJETO	CODIGO FUENTE		CODIGO OBJETO	CODIGO FUENTE	
DD6E05	LD	L,(IX+d)	CB86	RES	0,(HL)
FD6E05	LD	L,(IY+d)	DDCB0586	RES	0,(IX+d)
6F	LD	L,A	FDCB0586	RES	0,(IY+d)
68	LD	L,B	CB87	RES	0,A
69	LD	L,C	CB80	RES	0,B
6A	LD	L,D	CB81	RES	0,C
6B	LD	L,E	CB82	RES	0,D
6C	LD	L,H	CB83	RES	0,E
6D	LD	L,L	CB84	RES	0,H
2E20	LD	L,n	CB85	RES	0,L
ED4F	LD	R,A	CB8E	RES	1,(HL)
ED7B8405	LD	SP,(nn)	DDCB058E	RES	1,(IX+d)
F9	LD	SP,HL	FDCB058E	RES	1,(IY+d)
DDF9	LD	SP,IX	CB8F	RES	1,A
FDF9	LD	SP,IY	CB88	RES	1,B
318405	LD	SP,nn	CB89	RES	1,C
EDA8	LDD		CB8A	RES	1,D
EDB8	LDDR		CB8B	RES	1,E
EDA0	LDI		CB8C	RES	1,H
EDB0	LDIR		CB8D	RES	1,L
ED44	NEG		CB96	RES	2,(HL)
00	NOP		DDCB0596	RES	2,(IX+d)
B6	OR	(HL)	DDCB0596	RES	2,(IY+d)
DDB605	OR	(IX+d)	CB97	RES	2,A
B7	OR	(IY+d)	CB90	RES	2,B
B0	OR	A	CB91	RES	2,C
B1	OR	B	CB92	RES	2,D
B2	OR	C	CB93	RES	2,E
B3	OR	D	CB94	RES	2,H
B4	OR	E	CB95	RES	2,L
B5	OR	H	CB9E	RES	3,(HL)
F620	OR	L	DDCCB059E	RES	3,(IX+d)
EDBB	OR	n	FDCCB059E	RES	3,(IY+d)
CB9F	RES	3,A	ED4D	RETI	
CB98	RES	3,B	ED45	RETN	
CB99	RES	3,C	CB16	RL	(HL)
CB9A	RES	3,D	DDCB0516	RL	(IX+d)
CB9B	RES	3,E	FDCB0516	RL	(IY+d)
CB9C	RES	3,H	CB17	RL	A
CB9D	RES	3,L	CB10	RL	B
CBA6	RES	4,(HL)	CB11	RL	C
DDCB05A6	RES	4,(IX+d)	CB12	RL	D
FDCB05A6	RES	4,(IY+d)	CB13	RL	E
CBA7	RES	4,A	CB14	RL	H
CBA0	RES	4,B	CB15	RL	L
CBA1	RES	4,C	17	RLA	
CBA2	RES	4,D	CB06	RLC	(HL)

CODIGO OBJETO	CODIGO FUENTE		CODIGO OBJETO	CODIGO FUENTE	
CBA3	RES	4,E	DDCB0506	RLC	(IX+d)
CBA4	RES	4,H	FDCB0506	RLC	(IY+d)
CBA5	RES	4,L	CB07	RLC	
CBAE	RES	5,(HL)	CB00	RLC	B
DDCB05AE	RES	5,(IX+d)	CB01	RLC	C
DDCB05AE	RES	5,(IY+d)	CB02	RLC	D
CBAF	RES	5,A	CB03	RLC	E
CBA8	RES	5,B	CB04	RLC	H
CBA9	RES	5,C	CB05	RLC	L
CBAA	RES	5,D	07	RLCA	
CBAB	RES	5,E	ED6F	RLD	
CBAC	RES	5,H	CB1E	RR	(HL)
CBAD	RES	5,L	DDCB051E	RR	(IX+d)
CBB6	RES	6,(HL)	FDCB051E	RR	(IY+d)
DDCCB05B6	RES	6,(IX+d)	CB1F	RR	A
FDCCB05B6	RES	6,(IY+d)	CB18	RR	B
CBB7	RES	6,A	CB19	RR	C
CBB0	RES	6,B	CB1A	RR	D
CBB1	RES	6,C	CB1B	RR	E
CBB2	RES	6,D	CB1C	RR	H
CBB3	RES	6,E	CB1D	RR	L
CBB4	RES	6,H	1F	RRA	
CBB5	RES	6,L	CB0E	RRC	(HL)
CBBE	RES	7,(HL)	DDCB050E	RRC	(IX+d)
DDCB05BE	RES	7,(IX+d)	FDCB050E	RRC	(IY+d)
DDCB05BE	RES	7,(IY+d)	CB0F	RRC	A
CBBF	RES	7,A	CB08	RRC	B
CBB8	RES	7,B	CB09	RRC	C
CBB9	RES	7,C	CB0A	RRC	D
CBBA	RES	7,D	CB0B	RRC	E
CBBB	RES	7,E	CB0C	RRC	H
CBBC	RES	7,H	CB0D	RRC	L
CBBD	RES	7,L	0F	RRCA	
C9	RET		ED67	RRD	
D8	RET	C	C7	RST	00H
F8	RET	M	CF	RST	08H
D0	RET	NC	D7	RST	10H
C0	RET	NZ	DF	RST	18H
F0	RET	P	E7	RST	20H
E8	RET	PE	EF	RST	28H
E0	RET	PO	F7	RST1	30H
C8	RET	Z	FF	RST	38H
9E	SBC	A,(HL)	DDCB05E6	SET	4,(IX+d)
DD9E05	SBC	A,(IX+d)	FDCB05E6	SET	4,(IY+d)
FD9E05	SBC	A,(IY+d)	CBE7	SET	4,A
9F	SBC	A,A	CBE0	SET	4,B
98	SBC	A,B	CBE1	SET	4,C

CODIGO OBJETO	CODIGO FUENTE	CODIGO OBJETO	CODIGO FUENTE
99	SBC A,C	CBE2	SET 4,D
9A	SBC A,D	CBE3	SET 4,E
9B	SBC A,E	CBE4	SET 4,H
9C	SBC A,H	CBE5	SET 4,L
9D	SBC A,L	CBEE	SET 5,(HL)
ED42	SBC HL,BC	DDCB05EE	SET 5,(IX+d)
ED52	SBC HL,DE	FDCB05EE	SET 5,(IY+d)
ED62	SBC HL,HL	CBEF	SET 5,A
ED72	SBC HL,SP	CBE8	SET 5,B
37	SCF	CBE9	SET 5,C
CBC6	SET 0,(HL)	CBEA	SET 5,D
DDCB05C6	SET 0,(IX+d)	CBEB	SET 5,E
FDCB05C6	SET 0,(IY+d)	CBEC	SET 5,H
CBC7	SET 0,A	CBED	SET 5,L
CBC0	SET 0,B	CBF6	SET 6,(HL)
CBC1	SET 0,C	DDCB05F6	SET 6,(IX+d)
CBC2	SET 0,D	FDCB05F6	SET 6,(IY+d)
CBC3	SET 0,E	CBF7	SET 6,A
CBC4	SET 0,H	CBF1	SET 6,B
CBC5	SET 0,L	CBF2	SET 6,C
CBCE	SET 1,(HL)	CBF3	SET 6,D
DDCB05CE	SET 1,(IX+d)	CBF4	SET 6,E
FDCB05CE	SET 1,(IY+d)	CBF5	SET 6,H
CBCF	SET 1,A	CBF6	SET 6,L
CBC8	SET 1,B	CBFE	SET 7,(HL)
CBC9	SET 1,C	DDCB05FE	SET 7,(IX+d)
CBCA	SET 1,D	FDCB05FE	SET 7,(IY+d)
CBCB	SET 1,E	CBFF	SET 7,A
CBCC	SET 1,H	CBF8	SET 7,B
CBCD	SET 1,L	CBF9	SET 7,C
CBD6	SET 2,(HL)	CBFA	SET 7,D
DDCB05D6	SET 2,(IX+d)	CBFB	SET 7,E
FDCB05D6	SET 2,(IY+d)	CBFC	SET 7,H
CBD7	SET 2,A	CBFD	SET 7,L
CBD0	SET 2,B	CB26	SLA (HL)
CBD1	SET 2,C	DDCB0526	SLA (IX+d)
CBD2	SET 2,D	FDCB0526	SLA (IY+d)
CBD3	SET 2,E	CB27	SLA A
CBD4	SET 2,H	CB20	SLA B
CBD5	SET 2,L	CB21	SLA C
CBD8	SET 3,(HL)	CB22	SLA D
CBDE	SET 3,(IX+d)	CB23	SLA E
DDCB05DE	SET 3,(IY+d)	CB24	SLA H
FDCB05DE	SET 3,A	CB25	SLA L
CBDF	SET 3,B	CB2E	SRA (HL)
CBD9	SET 3,C	DDCB052E	SRA (IX+d)
CBDA	SET 3,D	FDCB052E	SRA (IY+d)

CODIGO OBJETO	CODIGO FUENTE
CB2F	SRA A
B28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB3E	SRL (HL)
DDCB053E	SRL (IX+d)
FDCB053E	SRL (IY+d)
CB3F	SRL A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
96	SUB (HL)
DD9605	SUB (IX+d)
FD9605	SUB (IY+d)
97	SUB A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
D620	SUB n
AE	XOR (HL)
DDAE05	XOR (IX+d)
FDAE05	XOR (IY+d)
AF	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE20	XOR n

APENDICE V. MICROPROCESADOR Z-80

DEFINICION FUNCIONAL DE LAS PATAS DE Z-80

A15-A0 (Bus de dirección)

Salidas de tres estados y activas en nivel alto. Las líneas A15-A0 constituyen un bus de dirección de 16 bits, en donde A15 es la línea más significativa y A0 es la línea menos significativa. La CPU Z-80 con estas 16 líneas tiene una capacidad para direccionar hasta 65,536 ó 64K localidades de memoria. Este bus de dirección tambien se usa para enviar el código de selección de dispositivos de Entrada/Salida (E/S). La dirección de un dispositivo de E/S utiliza las ocho líneas de más bajo orden que permiten al usuario selleccionar hasta 256 puertos de entrada y 256 puertos de salida. Durante cierto tiempo en la ejecución de cada intrucción, el contenido del registro R (Registro de Refrescar Memoria) se envía por las siete líneas de más bajo orden para función de refrescar memoria.

D7-D0 (Bus de Datos)

Entradas/Salidas de tres estados y activas en nivel alto. Proporcionan un bus de datos para la comunicación bidireccional entre la CPU Z-80 con los dispositivos de memoria y de E/S para la transferencia de instrucciones y datos. La línea D7 transmite el bit mas significativo y la línea D0 el bit menos significativo.

$\overline{M1}$ (Ciclo de Máquina Uno)

Salida y activo en nivel bajo, M1 indica que el ciclo de máquina en proceso es un ciclo Fetch, se utiliza para obtener el código de operación de la próxima instrucción a ejecutar. La señal M1 se activa junto con la señal IORQ para indicar un ciclo de reconocimiento de interrupción.

$\overline{MREQ}$ (Solicitud de Memoria)

Salida de tres estados y activo en nivel bajo, indica que el bus de dirección conserva una dirección válida para operaciones de leer y escribir en memoria.

$\overline{IORQ}$ (Solicitud de E/S)

Salida de tres estados y activo en nivel bajo, indica que las 8 líneas de dirección de más bajo orden tienen una dirección de E/S válida para operaciones de leer o escribir en dispositivos de E/S. Esta señal se genera cuando la Z-80 reconoce una interrupción que indica que se debe colocar un vector de interrupción en el bus de datos.

$\overline{RD}$ (Leer Memoria)

Salida de tres estados y activo en nivel bajo, indica que la CPU va a leer un dato de la memoria o un dispositivo de E/S. El dispositivo de E/S o la localidad de memoria direccionada deberá utilizar esta señal para enviar el dato al bus de datos de la CPU.

$\overline{WR}$ (Escribir en Memoria)

Salida de tres etados y activa en nivel bajo. Indica que el bus de datos tiene un dato válido que se debe almacenar en la memoria o en un puerto de E/S.

$\overline{RFSH}$ (Refrescar Memoria)

Salida y activa en nivel bajo, indica que las 7 lineas de más bajo orden presentes en el bus de dirección contienen una dirección de refresco para memorias dinámicas. Estas memorias dinámicas requieren periódicamente de una función de refrescar para poder conservar los datos almacenados en ellas.

$\overline{HALT}$ (Estado de Alto)

Salida y activa en nivel bajo, indica que la CPU ha ejecutado la instruccion HALT y esta esperando una solicitud de interrupción no-mascarable o mascarable antes de que continue en operación. Mientras está en estado de alto, la CPU ejecuta instrucciones NOP (no operación) para mantener la actividad de refrescar la memoria dinamica.

$\overline{WAIT}$ (Esperar)

Entrada y activa en nivel bajo, indica a la CPU que ha direccionado a una localidad de memoria o a un puerto de E/S que no tiene listo todavía el dato a transferir. La CPU tiene estado de espera mientras esta señal esta en nivel bajo. Esta señal permite a la memoria o a los dispositivos de E/S de cualquier velocidad a sincronizarse con la CPU.

$\overline{INT}$ (Solicitud de interrupción)

Entrada y activa en nivel bajo, una solicitud de interrupción generada por los dispositivos de E/S será atendida al final de la instrucción en proceso si el flip-flop interno de habilitar Interrupción (IFF) controlado por programación esta habilitado y si la señal BUSRQ no está activa. La CPU puede responder a una interrupción en tres diferentes modos.

$\overline{\text{NMI}}$ (Interrupción No-Mascarable)

Entrada, se dispara con el cambio de nivel alto-bajo. La línea de interrupción no-mascarable tiene una prioridad más alta que la entrada INT y se reconoce al final de la instruccion en proceso, de manera que el usuario puede regresar al programa que fue interrumpido. Una secuencia continua de ciclos WAIT puede evitar que la instrucción en proceso termine, y una atención a una señal BUSRQ ignorara a la señal NMI.

$\overline{\text{RESET}}$ (Limpiar)

Entrada y activa en nivel bajo, forza al contador del programa a cero e inicializa la CPU, lo que implica:
1.- Se deshabilita al flip-flop de habilitar Interrupción previniendo al sistema a aceptar interrupciones con excepción a las hechas a través de la línea NMI.
2.- El registro I,vector de interrupción, se limpia a 00H.
3.- El registro R, registro de refrescar, se limpia a 00H.
4.- Se pone el modo de interrupción 0.
5.- El bus de dirección pasa al estado de alta impedancia.
6.- El bus de datos pasa al estado de alta impedancia.
7.- Todas las señales de control pasan al estado inactivo

$\overline{\text{BUSRQ}}$ (Solicitud del Bus)

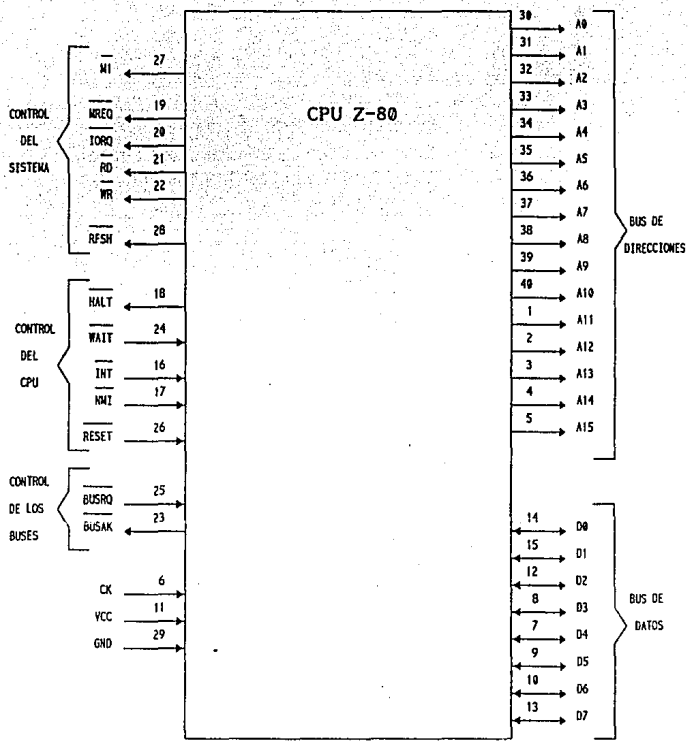
Entrada y activo en nivel bajo. Se usa para solicitar que el bus de dirección, el bus de datos y las salidas de control de tres estados de la CPU pasen al estado de alta impedancia de tal manera que otros dispositivos externos puedan tomar el control de los buses, generalmente, para realizar operaciones de acceso directo a Memoria (DMA). Cuando el dispositivo termina de hacer la transferencia, regresa la señal BUSRQ a nivel alto.

$\overline{\text{BUSAK}}$ (Reconocimiento del Bus)

Salida y activo en nivel bajo, cuando se indica que los buses ya estan libres se envía a nivel bajo la salida BUSAK. El dispositivo regresará la señal BUSRQ a nivel alto cuando termina de usar los buses, la Z-80 reconoce esta acción regresando la señal BUSAK a nivel alto y tomando el control de ellos.

CLOCK (Señal de reloj)

Entrada de señal de reloj que requiere unicamente de una resistencia de 330 ohms en pull-up con la alimentación de +5V para satisfacer todos los requerimientos.



DISTRIBUCION DE LOS PINES DEL CPU Z-80

BIBLIOGRAFIA

-- Z-80 Microcomputer Design Projects
William Barden Jr.
Howard W. Sams and Co., Inc.

-- The 8080, 8085, and Z80
Hardware, Software, Programming, Interfacing and Troubleshooting
David Lalond
Prentice Hall, Englewood Cliffs, New Jersey, 1988

-- Microprocessor Interfacing Techniques
Austin Lesea and Rodnay Zaks
Sibex, 1982

-- Design of Microprocessor sensor and Control Systems
Michael F.Hordeski

-- Peripherals
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051
1990

-- ECG Semiconductors
Master Replacement Guide
Philips ECG, Inc. 1989