Ref. 5-A

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"DISEÑO Y CONSTRUCCION DE UN SISTEMA DE DESARROLLO BASADO EN UN MCU-Z8"

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N :

CLARA CAMACHO CAMPOS

EDUARDO CRUZ JIMENEZ

EDUARDO REYES HERNANDEZ

DIRECTOR

ING. MANUEL CORREA SINTORA

México, D. F. 1987.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

PROLOGO

Cualquier proyecto profesional ingenieril, debe tener una justificación. Esto es un motivo para invertir recursos humanos y materiales, que implícitamente son necesarios para el desarrollo de una aplicación determinada de la ingeniería.

En términos éticos, es claro, que ésta motivación debe ser la de finalmente satisfacer cualquier necesidad de una comunidad o entidad.

Cuando a un proyecto le hace falta esta justificación, sin importar la calidad del mismo o el monto de los recursos empleados, éste resulta en un vano esfuerzo, que por su propio peso, cae en el olvido y el abandono.

Es común escuchar de los estudiantes de la Facultad de Ingeniería, que la tesis es solo un trámite inútil, puesto que finalmente es un trabajo irrelevante que solo terminará archivado. Quiza éste se deba a que los proyectos de tesis profesionales de ingeniería tienden a perder su justificación, para convertirse en un trabajo más de la carrera, necesario únicamente para adquirir un título. Lo anterior, es algo más serio, que un problema exclusivo de una coordinación o institución, se trata del problema de una sociedad que no quiere o no sabe aprovechar los recursos, técnicos y creativos, disponibles. Es un síntoma más de la dependencia tecnológica de México.

Ha sido el deseo de quienes colaboramos en el presente trabajo, que el Sistema de Desarrollo para el MCSC Z8 (SIS Z8), no adoleciera de una justificación, es decir que el elemento de trascendencia del proyecto no sea el SIS Z8 por sí solo. El SIS Z8 es un proyecto auspiciado por una entidad de la iniciativa privada, cuyo objetivo es coadyuvar en la elaboración de nuevos proyectos comerciales basados en el MCSC Z8.

Tenemos la certeza, que cualquier desarrollo tecnológico es útil, sí indirecta o directamente, satisface requerimientos de

una sociedad. La nuestra es una sociedad que exige la creación de dispositivos tecnológicos con recursos propios, que tiendan a sustituir importaciones y/o generar riqueza económica. Es cierto que en la U.N.A.M. se realizan grandes esfuerzos para crear tecnología nacional, pero también es cierto que mientras la iniciativa privada no absorba los resultados de la investigación científica de las instituciones nacionales, o bién, sea capaz de crear tecnología propia, entonces todo el aparato productivo seguirá en el retraso permanente.

El SIS Z8 queda justificado por el solo deseo verdadero de ser la herramienta útil para otros proyectos comerciales y competitivos, que aunque aún están por verse, el hecho es que si se llevaran a cabo muchos proyectos (de tesis, por ejemplo) con este enfoque, seguramente algunos de ellos lograrían su objetivo. Y èsto si seria un avance...

En cuanto al texto, se ha buscado lograr una lectura àgil del mismo, para que las ideas y conocimientos que en el se encuentran sean fàcilmente aprovechables y que de alguna manera sean semilla de proyectos futuros.

Con esta idea en mente, se han traducido y corregido los manuales del MCSC Z8, para hacer comprensible el resto del texto, así tambièn para proveer una fuente con información de primera mano, para aquellos que deseen aplicar èste dispositivo.

En cuanto a la aplicación de los tèrminos tècnicos en inglès, para evitar confusiones, estos fueron empleados libremente sin atentar hacer una traducción de muchos de ellos, ya que estamos conscientes de que esto requiere de un trabajo profundo que tenga el concenso de una institución o comunidad ampliamente reconocida. Además de que, tristemente, son escasos los textos tècnicos en español, referentes a la microcomputación y por lo tanto, los ingenieros en electrónica y computación, aprenden de textos en inglès, por lo cual se encuentran más familiarizados con los tèrminos tècnicos en este idioma.

Así que la idea es hacer de este trabajo un escrito amigable, para poder transmitir la experiencia del SIS Z8 fàcilmente a aquellos que la encuentren necesaria , o simplemente interesante, su lectura.

En resumen, es nuestra esperanza, que el SIS Z8 y el presente trabajo sean útiles como semillas para nuevas y mejores ideas, y no solamente el efimero tema de un examen profesional.

CAPITULO I

INTRODUCCION

Han transcurrido casi 15 años desde Noviembre de 1971, fecha en que apareció en el mercado el 4004, considerado como el primer microprocesador comercial. De aquel primer microprocesador a los existentes en la actualidad hay una gran distancia tecnológica producto del incremento de novedosas aplicaciones que han llevado a los fabricantes de semiconductores a producir gran cantidad de microprocesadores cuyas variantes los hacen idóneos para determinadas aplicaciones específicas. Así hemos visto aparecer una gran variedad de microprocesadores, algunos de los cuales han quedado obsoletos con el paso del tiempo, mientras que otros han permanecido y sus arquitecturas se han convertido en clásicas, tales como el 8080 de INTEL y el Z80 de ZILOG. Así dentro de este desfile de arquitecturas distintas una parece destinada a establecer una nueva tendencia con aplicaciones bien definidas, que con el tiempo probablemente será un campo distinto al de los microprocesadores, nos referimos a los microcomputadores de un solo chip.

Esta nueva tendencia tiene su punto de partida en la arquitectura del 8748 de INTEL, introducido en Febrero de 1977. A este siguió toda una familia de microcomputadoras de un solo chip: la MCS 48. Para principios de 1980 prácticamente todos los fabricantes mayores de microprocesadores habían imitado e incluso superado al 8748 con el 6801 de MOTOROLA y el Z8 de ZILOG, siendo indiscutiblemente las microcomputadoras de un solo chip más poderosas y versátiles existentes en la actualidad.

Pero ¿Que es un microcomputador de un solo chip y cuales son las diferencias con los microprocesadores?

Entendamos por microprocesador como la Unidad Central de Proceso (CPU) de un microcomputador, llevado al nivel de circuito integrado. Si juntamos un microprocesador con memorias y a esto le damos capacidad de Entrada/Salida estamos hablando de un microcomputador. El prefijo micro indica el pequeño tamaño físico de los elementos involucrados. Así, para hacer un microcomputador

con base en un microprocesador necesitamos de varios elementos más en forma de circuitos integrados, tales como Puertos E/S en Paralelo, Puertos E/S Serie, Memorias RAM, Memorias ROM, Temporizadores, Contadores, etc. La idea del microcomputador de un solo chip, es llevar al mínimo el número de elementos para la realización de un sistema computacional : solamente un circuito integrado. Esto es fundamentalmente un microcomputador de un solo chip y su característica más sobresaliente es su versatilidad y la capacidad para adaptarse a una gran variedad de aplicaciones diversas.

El Z8 encaja perfectamente en este esquema y para dar una idea más clara, la Figura 1 indica los elementos necesarios de un microcomputador y cuales de ellos separados por la línea punteada, constituyen al Z8.

Externamente el Z8 es un circuito integrado de 40 pins, de los cuales 32 son destinados a E/S agrupados en 4 puertos de 8 líneas cada uno, que pueden ser programados como entrada, salida, bidireccionales en serie o paralelo, con o sin señales de enlace (Handshake), o bien, proveedores de señales de status, direccionamiento y temporización.

Internamente el Z8 consta de los siguientes elementos:

1.- Una memoria de solo lectura (ROM) que según la versión puede ser de 2048 o 4096 bytes.

2.- Un archivo de registros de ocho bits, de acceso aleatorio, organizado en 124 registros de propósito general, 16 de control y 4 registros correspondientes a los puertos de E/S. Cualquiera de los registros de propósito general puede trabajar como acumulador, registro índice, o formar parte de una pila (stack) interna. Los 144 registros han sido agrupados en 9 bloques de 16 registros de trabajo cada uno. Uno de los registros de control trabaja como apuntador a grupos de registros , lo cual conduce a un formato corto en las instrucciones y un acceso rápido a cualquiera de los 16 registros del grupo designado por el registro apuntador.

3.- Un mecanismo que protege al archivo de registros contra fallas de energía.

4.- Un oscilador interno que acepta diversas formas de bases de tiempo.

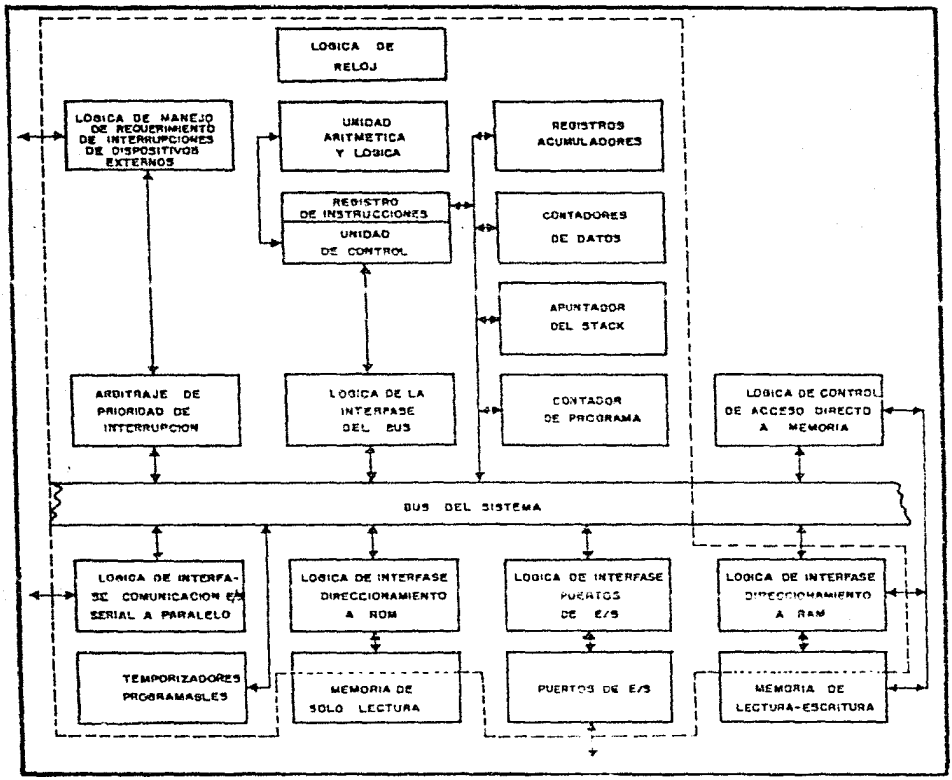
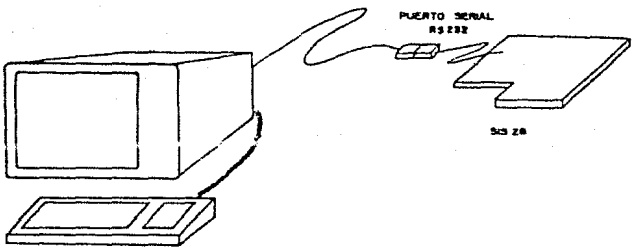
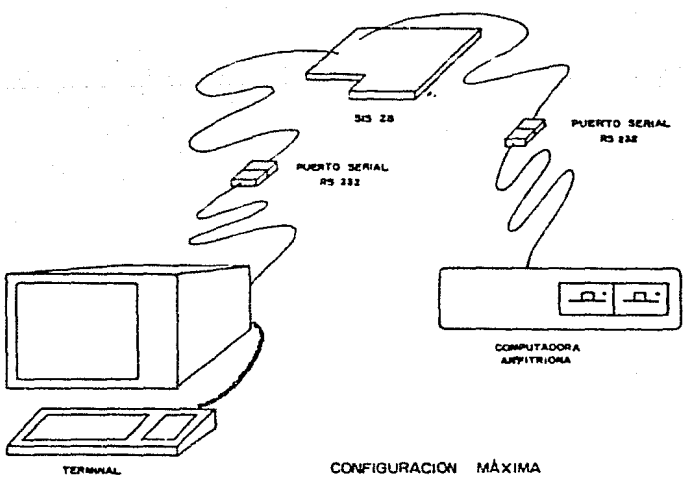


FIGURA (1.1)

En cuanto a la programación de Z8, este cuenta con 48 instrucciones que pueden trabajar distintos formatos, incluyendo bits individuales, digitos BCD, bytes y palabras de 16 bits. Trabajando con un reloj de 8 MHz, la mayoria de las instrucciones se ejecuta en 1.5 a 2.5 microsegundos.



CONFIGURACION SENCILLA



CONFIGURACION MÁXIMA

El espacio de direccionamiento no queda limitado a los registros y la memoria interna--, puesto que la memoria de programa puede ser ampliada con memoria externa hasta 64 KB, además de la opción de hasta 62KB de memoria exclusiva de datos

El Z8 puede aceptar 3 interrupciones internas y 3 externas, cuya prioridad es programable y su formato es a base de vectores de interrupción.

Todo lo anterior, regido por un programa de control adecuado, hace posible configurar al Z8 para adaptarse a cualquiera de las necesidades de aplicación. Particularmente, ha sido diseñado para realizar gran cantidad de combinaciones Memoria y Entrada/Salida. En un extremo el Z8 puede configurarse como una microcomputadora con uso intensivo de E/S mientras que en el otro puede enfocarse al uso intensivo de memoria.

Las aplicaciones más importantes son probablemente en instrumentación, sistemas donde el número de circuitos integrados es una limitante, y en procesos de control en tiempo real donde la velocidad es una consideración importante.

¿QUE ES EL SIS Z8?

Tal vez la desventaja más importante del Z8 es que la ROM interna no es programable por el usuario. Sin embargo, en la familia Z8 se ofrecen versiones que atenúan esta desventaja. Existen versiones del Z8 con 2 y 4 KB de memoria interna , el Z8613, por ejemplo, ha sido diseñado para prototipos y es empacado con una base para EPROM, que sustituye a la memoria interna. El Z8612 es una versión de 64 pins sin memoria interna destinada al desarrollo. ZILOG también dispone de herramientas para desarrollo tales como el lenguaje ensamblador del Z8, el Z8 SCAN y el Z8DM, aunque en realidad, estas últimas dos son inaccesibles para diseñadores en México.

El Z8613 permite la elaboración de nuevos productos, pero no elimina la tarea de grabar y borrar EPROMS en cantidades fenomenales. (Durante el desarrollo del SIS Z8 se hizo esta operación por lo menos dos centenares de veces). Surge entonces la necesidad de una herramienta que facilite grandemente la elaboración de proyectos con el Z8 con las siguientes carcterísticas

a) La mayoría de los componentes del Z8 puedan ser empleados por el usuario en sus proyectos.

b) Capacidad para interrumpir la ejecución de un programa y muestrear los registros del Z8 para comprobar la secuencia del mismo.

c) Memoria RAM accesible en el mismo sistema, tanto en datos como en programa.

d) Sencillez de operación.

e) Facilidades para modificar errores y examinar el programa en tiempo real.

Esta herramienta es lo que consideramos un sistema de desarrollo y èsto es precisamente el SIS Z8.

El sistema de desarrollo SIS Z8 cuenta con los siguientes elementos:

1.- Dos puertos seriales.

2.- 16 KB de memoria RAM para el usuario, que pueden configurarse en cualquiera de las formas siguientes:

- 16 KB de memoria de programa

- 8 KB de memoria de programa y 8 KB de memoria de datos.

3.- Una memoria RAM que simula a la memoria interna del Z8, lo cual permite la modificación de programas en tiempo real.

4.- Un programa monitor/debug que contiene varios comandos de alto nivel para depurar programas, así también dos herramientas para el control de secuencia.

5.- Un módulo para memorias de solo lectura ROM.

El SIS Z8 puede trabajar con 2 esquemas (ver página 20). En el primero solamente es necesaria una terminal conectada al puerto serial #1. En este esquema el usuario se comunica unicamente con el SIS Z8 mediante la terminal y trabaja con los comandos del programa monitor del SIS Z8, que llamaremos SCAN Z8.

El segundo esquema requiere de una computadora anfitriona, el lenguaje ensamblador del Z8 y una terminal.Las ventajas de esta configuración son grandes. La computadora anfitriona complementa al SIS Z8 y permite que los programas sean escritos en nemónicos con un editor de linea, almacenados en discos flexibles, ensamblados con el ensamblador del Z8 y transferidos al SIS Z8 donde pueden depurarse. El canal de comunicación entre la terminal y la computadora maestra es el SIS Z8. El SCAN Z8 puede interpretar cuales comandos son destinados a la computadora anfitriona y cuales son dirigidos al SIS Z8. De esta

manera el usuario puede pasar el control de la computadora al SIS Z8 y viceversa con solo una instrucción.

El software necesario para esta configuración es mínimo, y a excepción del ensamblador del Z8, es perfectamente estandard en cualquier microcomputadora.
El SCAN Z8 consiste de 16 comandos de los cuales 12, son estandard en los programas depuradores comerciales, tales como sustitución y despliegue de memoria, despliegue de registros, etc. Además, contiene dos comandos que transfieren el control de la computadora anfitriona al SIS Z8 y 2 comandos de autoprueba.

Cuando se utiliza el segundo esquema, los programas para el Z8 se introducen a un archivo en forma de nemónicos en un editor de línea. El ensamblador del Z8 lee este archivo y genera el código objeto en otro archivo con formato INTEL. El comando TRM del SCAN Z8, con la ayuda de un programa codificado en BASIC, transfiere este archivo a la memoria del SIS Z8, donde el usuario lo pone en prueba y corrige errores. Todo esto elimina la necesidad de grabado de EPROMS hasta el punto en que el programa ha sido totalmente depurado. Si la computadora anfitriona dispone de elementos de desarrollo tales como grabadores de EPROMS, toda la operación de desarrollo se hace con el mismo conjunto y en consecuencia el tiempo y costo de elaboración de nuevos proyectos debe abatirse notoriamente.

Esta es la finalidad del SIS Z8, y para comprender mejor la forma de operarlo y sacar provecho de sus cualidades recomendamos el capítulo 4 "DESCRIPCION OPERACIONAL DEL SIS Z8". Los capítulos 2 y 3 son una introducción a la arquitectura del Z8, para aquellos que no conozcan este microcomputador de un solo chip . Los capítulos titulados "DESCRIPCION FUNCIONAL DEL SIS Z8", 5 Y 6 son para quien quiera modificar ó mejorar al SIS Z8 y contienen una descripción detallada del Hardware y Software del mismo, incluido el listado completo del SCAN Z8 y los diagramas del alambrado.

Finalmente el capítulo 7 es un resumen de las experiencias más sobresalientes obtenidas en el lapso de año y medio, que fué necesario en el desarrollo de este proyecto.

CAPITULO II

ARQUITECTURA DEL Z8

En este capitulo se trataran los aspectos de arquitectura de la microcomputadora de un solo chip Z8.

Inicialmente se hablará de las caracteristicas generales del Z8, asi como, de las representaciones en diagramas de bloques de la lógica funcional y de la arquitectura del mismo.

En el resto del capitulo se hablará detalladamente de los componentes y caracteristicas principales del Z8.

1.- INTRODUCCION

El Z8 introduce un nuevo nivel de sofisticación en arquitecturas de un solo chip. Comparado con microcomputadoras de un solo chip anteriores, el Z8 ofrece mayor rapidez de ejecución; uso de memoria más eficiente; mayor utilidad en interrupciones; entrada/salida con capacidad para manipular bits; y facilidades para expansión.

La caracteristica más notoria del Z8, es que sus multiples opciones pueden configurarse y modificarse dinamicamente bajo control de programa.

Algunas de las características generales del Z8 son las siguientes:

- Trabaja con solamente una fuente de +5V.
- Usando un Reloj de 8 MHz., las instrucciones se ejecutan en tiempos de 1.5 a 2.5 microsegundos.
- Cuenta con un sistema contra fallas de energia que protege el archivo de registros mientras dura el problema.

- Tres espacios de direcciones bàsicos: Memoria de Programa (Interna y Externa); Memoria de Datos (Externa); Archivo de Registros(Interno).

- Todas las señales son compatibles con niveles TTL.

- Las instrucciones del Z8 pueden operar sobre datos con distintas estructuras, tales como: datos de un solo bit, digitos codificados en BCD, bytes y palabras de 16 bits.

El diagrama de la Figura (2.1) nos muestra la estructura general del Z8, que a lo largo de este capítulo será explicada ampliamente.

Como se puede observar en este diagrama , el Z8 cuenta con los siguientes elementos:

- Un circuito Receptor/Transmisor Asincrono (UART).

- 2048 bytes de Memoria de Programa en una Memoria de Solo Lectura (ROM), la cual puede extenderse con 63488 bytes de memoria de programa externa.

- Dos Contadores/Temporizadores con un gran número de aplicaciones.

- Una Organización de Registros Internos que cuenta con 144 bytes de acceso aleatorio. Esta organización está compuesta por tres grupos de Registros.

  a) 124 Registros de Proposito General.
  b) 16 Registros de Control.
  c) 4 Registros de Puertos de Entrada/Salida.

- 32 lineas dedicadas a Entrada/Salida, agrupadas en 4 Puertos de 8 lineas cada uno, y que pueden configurarse como Entrada/Salida o Buses Dirección/Datos. Los Puertos también pueden programarse para proveer: Salida de Direcciones; Sincronización; Señales de Estado; y Caracteristicas de E/S Serie o Paralela, Sencilla o con Señales de Enlace.

El resto del capitulo discute los detalles de la arquitectura del Z8.

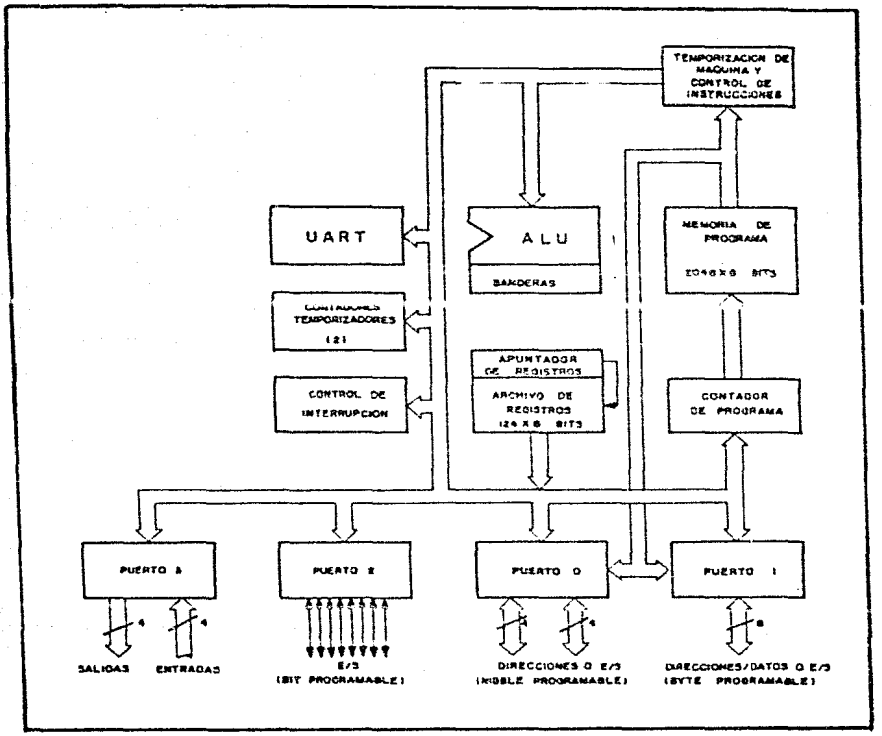
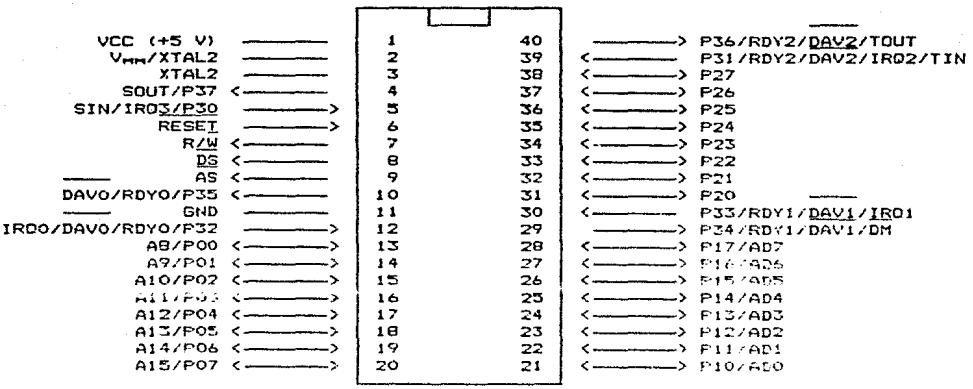


FIGURA (2.1)

2.- SEÑALES Y PINS DEL Z8.

Existen dos versiones del Z8: Una de 40 pins que es usada principalmente en productos comerciales; y otra de 64 pins para el desarrollo de productos. En esta sección se hablará exclusivamente de la versión de 40 pins, cuya asignación de señales se muestra en la siguiente página.



NOMBRE DEL PIN	DESCRIPCION	TIPO
P00-P07	E/S PUERTO 0	BIDIR.,TRISTATE
P10-P17	E/S PUERTO 1	BIDIR.,TRISTATE
P20-P27	E/S PUERTO 2	BIDIR.,OPCION DE COLECTOR ABIERTO
P30-P33	E/S PUERTO 3	SALIDA
P34-P37	E/S PUERTO 3	ENTRADA
AD0-AD7	BUS DATOS/DIRECCIONES	BIDIRECCIONAL
A8-A15	BUS DIRECCIONES	SALIDA
RDY0/DAV0	SEÑALES DE ENLACE PUERTO 0	ENTRADA O SALIDA
RDY1/DAV1	SEÑALES DE ENLACE PUERTO 1	ENTRADA O SALIDA
RDY2/DAV2	SEÑALES DE ENLACE PUERTO 2	ENTRADA O SALIDA
R/W	CONTROL DE ESCRITURA/LECTURA	SALIDA,TRISTATE
DS	DATA STROBE	SALIDA,TRISTATE
AS	ADDRESS STROBE	SALIDA,TRISTATE
DM	SELECCION MEMORIA DATOS	SALIDA
IRQ0-IRQ3	PETICIONES DE INTERRUPCION	ENTRADA
SIN	ENTRADA SERIAL DE DATOS	ENTRADA
SOUT	SALIDA SERIAL DE DATOS	SALIDA
TIN	ENTRADA DE TEMPORIZADORES	ENTRADA
TOUT	SALIDA DE TEMPORIZADORES	SALIDA
RESET	INICIALIZADOR	ENTRADA
XTAL1,XTAL2	CONECCIONES A CRISTAL	
VMM	FUENTE ADICIONAL STANDBY	
VCC,GND	FUENTE DE PODER,TIERRA	

El Z8 está compuesto por 32 pins dedicados a líneas de Puertos de Entrada/Salida, y 8 pins a Señales de Control, que serán descritas a continuación.

P00-P07, P10-P17, P20-P27, P30-P37.

Estas 32 líneas están divididas en 4 Puertos de E/S de 8 bits cada uno. Pueden ser configurados bajo control de programa en una variedad de formas. Además de sus funciones de E/S, el Puerto 0 puede programarse junto con el Puerto 1 para interfazar Memoria Externa, trabajando el primero como byte más significativo del Bus de Direcciones, mientras que el Bus de Datos y el byte menos significativo se presentan multiplexados en las terminales del Puerto 1. Ambos puertos opcionalmente pueden ser puestos en el estado de alta impedancia. El Puerto 2 puede configurarse con salidas tipo drenaje abierto (Open-Drain). Las líneas del Puerto 3 cumplen con una gran variedad de funciones además de la de E/S, que serán explicadas más adelante.

Por convención el primer dígito del número indica el puerto al que corresponde dicha línea, mientras que el segundo denota el orden individual. Por ejemplo, P30 corresponde al bit menos significativo del Puerto 3.

AS (Address Strobe).

Esta señal es activa durante la mitad del primer periodo de reloj del Ciclo de Máquina de Búsqueda (Fetch) a memoria externa o interna y durante los ciclos de transferencia de datos de memoria externa.

Cuando se usa memoria externa esta señal es útil para demultiplexar el byte menos significativo del bus de direcciones.

Por programación, AS junto con los Puertos 0 y 1, DS y R/W, pueden ser puestos en estado de alta impedancia.

DS (Data Strobe).

Es activa cada vez que se hace una transferencia de datos a memoria externa. Durante el ciclo de escritura el Z8 coloca el dato en el Puerto 1 cuando DS es baja. Durante un ciclo de lectura el dato debe ser puesto en el Puerto 1 cuando DS es bajo. Esta señal también tiene la opción de alta impedancia.

Cuando el Z8 no es configurado para transferencias en memoria externa, DS actua como una señal de sincronización de instrucciones y es pulsada baja durante el periodo de reloj inmediato precedente al Ciclo de Búsqueda de Código.

R/W (Read/Write).

Es baja cuando el Z8 escribe a memoria de datos o programa externa. Permanece alta el resto del tiempo y tiene la opción de alta impedancia por programación.

XTAL1, XTAL2.

Estas entradas sirven para conectar un cristal resonante serie de máximo 8 MHz (Con opción a 12 MHz) al reloj oscilador interno.

El reloj del Z8 debe ser generado externamente y conectado a XTAL1 cuando la opción de falla de energía es usada conectando XTAL2 a una fuente adicional (Vmm), la cual energiza el archivo de registros y la lógica de RESET durante una falla en la fuente Vcc.

El siguiente circuito es recomendado por la literatura de ZILOG para el uso de esta opción.

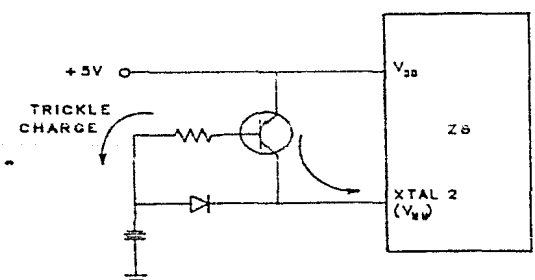


FIGURA (2.3)

Después de una falla general de energía, el circuito anterior provee la fuente de protección para energizar los registros 04H a 7FH y la lógica de RESET. Sin embargo, para que esta opción tenga sentido la lógica externa debe muestrear la fuente desde el transformador y detectar en forma incipiente una falla de energía para permitir que una rutina de interrupción dedicada a fallas de energía sea ejecutada.

RESET.

Esta señal inicializa al Z8 cuando se mantiene baja por lo menos 18 períodos de reloj. La lógica externa debe mantener bajo a RESET por lo menos 50 ms, después de que Vcc se ha estabilizado. El siguiente circuito es recomendable para un RESET automático.

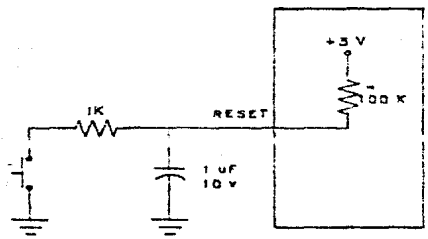


FIGURA (2.4)

Cuando RESET cambia al nivel alto, la ejecución de programa comienza con la instrucción grabada en la memoria interna en la localidad 000CH. Si el nivel de RESET es elevado a un voltaje mayor que Vcc, el Z8 es forzado a entrar al modo de prueba.

3.- SINCRONIZACION Y EJECUCION DE INSTRUCCIONES.

Los períodos de tiempo básicos del Z8 son: Ciclos de Máquina (Mn); Estados de Sincronización (Tn); y Períodos de Reloj. Todas las referencias de sincronización del Z8 son hechas con respecto a las señales AS Y DS.

El Z8 ejecuta las instruccions en una serie de Ciclos de Máquina, cada uno de los cuales consta de 3 períodos de Reloj Interno. La frecuencia de reloj interno es la mitad de la frecuencia en XTAL1. En los diagramas de tiempo el reloj mostrado corresponde al reloj externo, esto se ha hecho solamente por claridad.

Los diagramas de tiempo importantes del Z8 son los de: Pipeline de Instruccines; Ciclo de Búsqueda; Referencias a

Memoria Externa; Ciclo de Interrupción; y Ciclo de Reset. A continuación explicaremos estos ciclos y sus diagramas de tiempo.

PIPELINE DE INSTRUCCIONES.

La rapidez del Z8 es debida en parte al uso del Pipeline de las instrucciones, donde los Ciclos de Búsqueda de instrucción y los Ciclos de Ejecución de la instrucción anterior se traslapan. Esto es, cuando la ejecución de una instrucción se está realizando, el ciclo de búsqueda de la siguiente instrucción ya ha comenzado.

El traslape de Búsqueda/Ejecución hace que el tiempo efectivo de ejecución sea más corto que el tiempo de finalización de una instrucción, por una cantidad igual al traslape.

CICLOS DE BUSQUEDA Y REFERENCIAS A MEMORIA EXTERNA.

Para ejecutar cualquiera de sus instrucciones el Z8 combina 3 tipos distintos de ciclos de máquina: Lectura de Memoria; Escritura de Memoria; y Operación Interna. Solamente se muestran los diagramas para lectura y escritura de memoria, ya que en la operación interna no hay actividad externa en los buses.

Ciclo Extendido de Máquina. Opcionalmente y por programación, puede agregarse un cuarto periodo de reloj interno a cada ciclo de lectura/escritura a memoria externa. Este periodo adicional ocurre entre T2 y T3 y los niveles de las señales mostradas en T2 se mantienen hasta T3.

CICLO DE LECTURA DE MEMORIA. Cuando se utiliza memoria externa los Puertos 0 y 1 previamente deben ser programados como buses Datos/Direcciones, aunque como se verá más adelante opcionalmente se puede prescindir de P00 a P07, por lo que estos se muestran entre paréntesis en los diagramas. Como ya se mencionó anteriormente el byte menos significativo del bus de direcciones se presenta multiplexado con el bus de datos en P10-P17 y es estable solamente en el primer periodo del reloj del ciclo de máquina. Durante el segundo periodo, el nivel de estas líneas cambia para que durante el tercer ciclo el dato pueda entrar por las mismas.

Cualquier dirección presente en el Puerto 0 permanece estable durante todo el ciclo de máquina.

DIAGRAMA DE PIPELINE DE INSTRUCCIONES

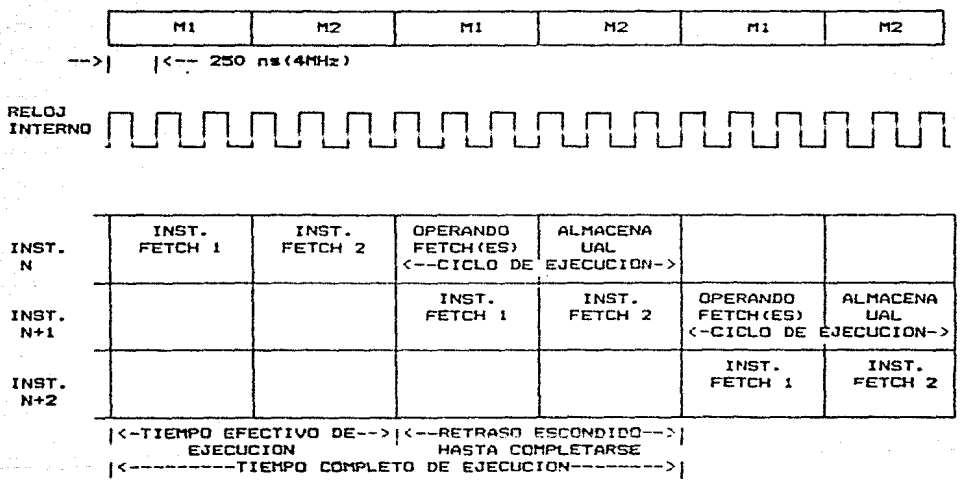


FIGURA (2.5)

AS es pulsada baja durante la mitad del primer periodo de reloj del primer ciclo de máquina. La lógica externa de selección de memoria debe aprovechar el flanco positivo de AS para amarrar las direcciones, ya que el byte menos significativo desaparece hacia el final del primer periodo de reloj.

DS cambia a un nivel bajo al final del primer periodo de reloj. La lógica externa debe colocar el dato correspondiente a la localidad de memoria direccionada en P10-P17 antes del flanco positivo de DS.

R/W permanece alta a lo largo del ciclo de máquina.

Si P34 es programado para funcionar como DM, este será bajo, si el ciclo de memoria de lectura ejecutado corresponde a una instrucción que explícitamente selecciona memoria de datos.

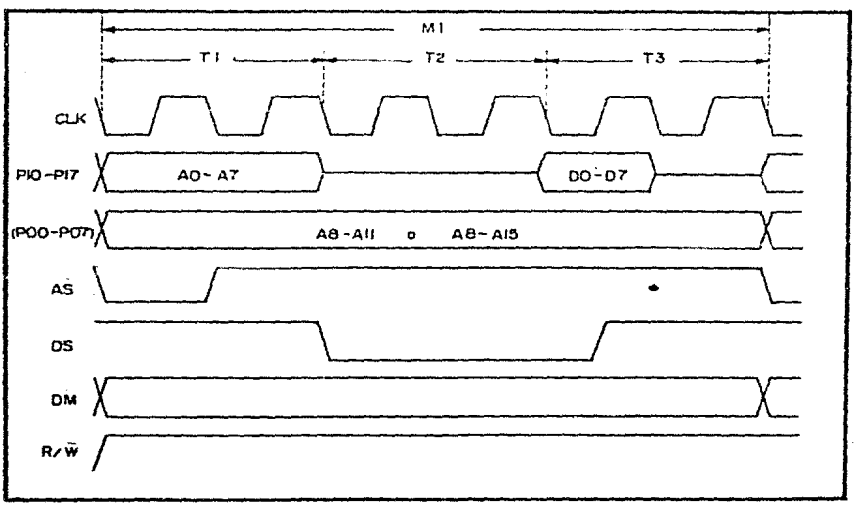


FIGURA (2.6)

CICLO DE ESCRITURA A MEMORIA. El diagrama de tiempo correspondiente a este ciclo no difiere significativamente del anterior, a excepción de que el dato aparece en P10-P17, y son válidos todo el tiempo que DS permanezca baja.

R/W en este caso es baja todo el ciclo de máquina.

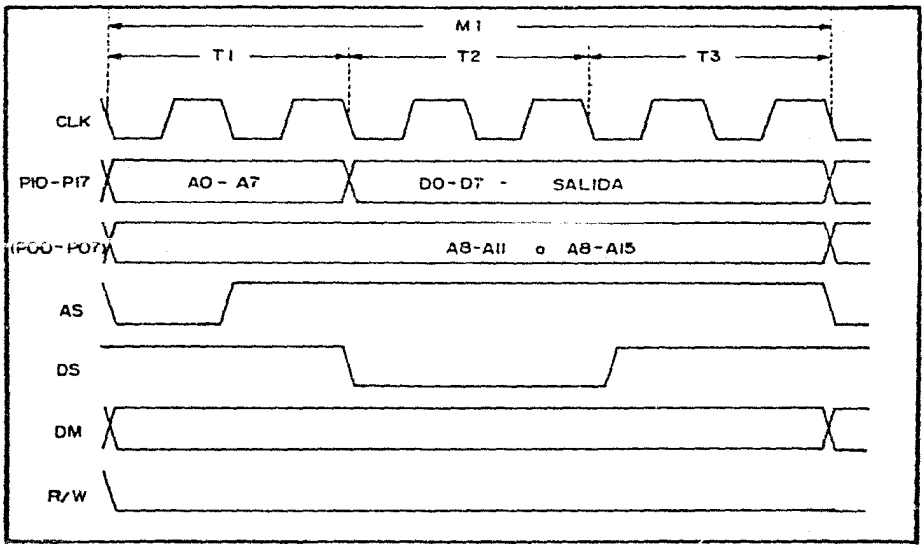


FIGURA (2.7)

CICLO DE INTERRUPCION. Las peticiones de interrupción se muestrean antes de cada ciclo de búsqueda de instrucción. Comenzando por las peticiones de interrupciones externas 4 períodos de reloj antes del pulso AS correspondiente a un ciclo de busqueda de instrucción.

Las peticiones de búsqueda de interrupciones internas son muestreadas en el período de reloj precedente a AS.

Si existe petición de interrupción, el Z8 necesita 7 ciclos de máquina (48 períodos de reloj) para resolver la prioridad de

las interrupciones, seleccionar el vector de interrupciones apropiado y salvar el contador de programa junto con las banderas en el stack.

NOTA: DURANTE EL DESARROLLO DEL SIS Z8, SE PUDO APRECIAR QUE LOS MANUALES DE ZILOG TIENEN UN ERROR AL RESPECTO DEL ORDEN EN QUE EN EL CICLO DE INTERRUPCION, LOS REGISTROS DE STATUS Y CONTADOR DE PROGRAMA SON SALVADOS EN EL STACK. SEGUN ESTOS, EL ORDEN ES EL SIGUIENTE: BYTE BAJO DE CONTADOR DE PROGRAMA, BYTE SUPERIOR DE CONTADOR DE PROGRAMA Y FINALMENTE REGISTRO DE STATUS. SIN EMBARGO, EL ORDEN CORRECTO DEBE SER EL SIGUIENTE : REGISTRO DE STATUS, BYTE BAJO DE CONTADOR DE PROGRAMA Y AL ULTIMO EL BYTE SUPERIOR DEL CONTADOR DE PROGRAMA.

En la figura de la siguiente página, se ilustra el diagrama de tiempo. El tiempo total de respuesta para una interrupción externa es 48 periodos de reloj, tiempo en el cual la primera instrucción de la rutina de servicio de interrupción es buscada.

CICLO DE RESET. La lógica interna es inicializada con un ciclo de RESET. Durante el tiempo que RESET es bajo, AS presenta una onda cuadrada cuya frecuencia es igual a la del reloj interno, DS es baja, R/W es inactiva y los Puertos 0, 1 y 2 son programados al modo de entrada.

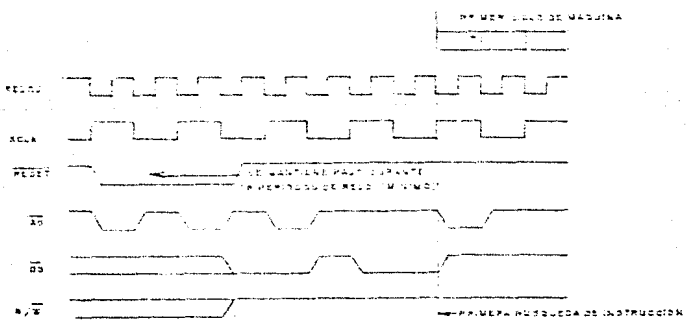
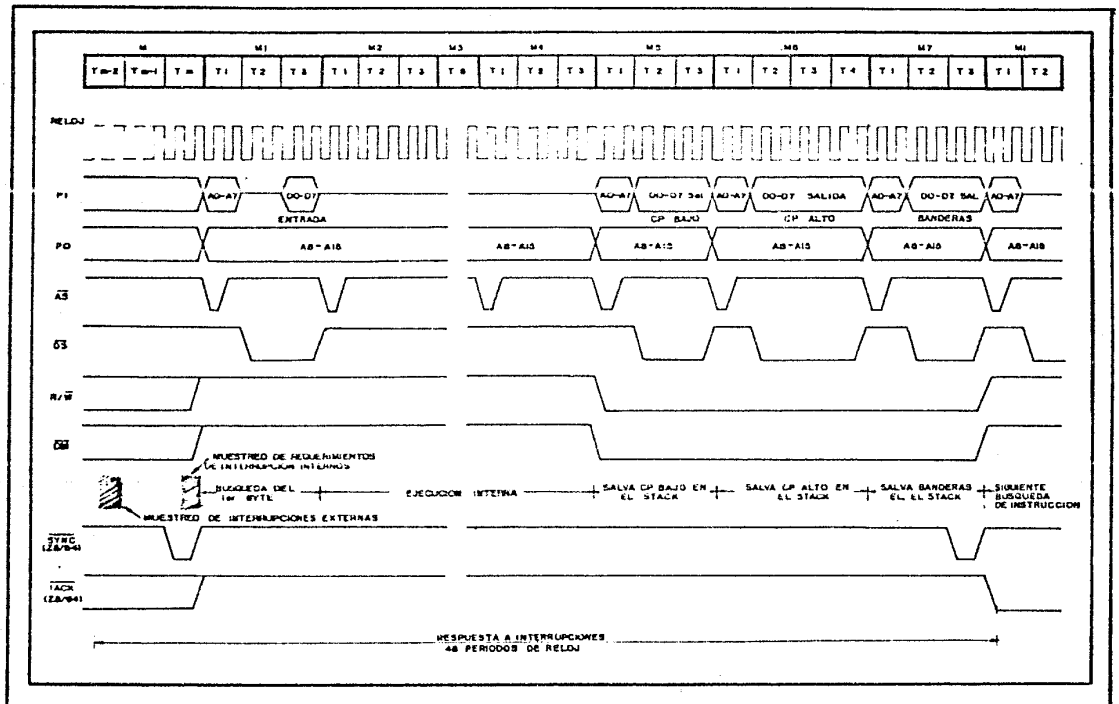


FIGURA (2.8)


M
M1
M2
M3
M4
M5
M6
M7
M1
RELOJ
P1
AD-A7
DO-D7
ENTRADA
DO-D7 Sal
DO-D7 SALIDA
CP BAJO
CP ALTO
DO-D7 SAL.
BANDERAS
P0
A8-A15
S3
S3
R/W
DBI
MUESTREO DE REQUERIMIENTOS DE INTERRUPCION INTERNOS
BUSQUEDA DEL 1er BYTE
EJECUCION INTERNA
SALVA CP BAJO EN EL STACK
SALVA CP ALTO EN EL STACK
SALVA BANDERAS EN EL EL STACK
SIGUIENTE BUSQUEDA DE INSTRUCCION
MUESTREO DE INTERRUPCIONES EXTERNAS
SYNC (Z8/84)
IACK (Z8/84)
RESPUESTA A INTERRUPCIONES
48 PERIODOS DE RELOJ

4.- ESPACIO DE DIRECCIONES.

El Z8 cuenta con tres espacios de direcciones básicos, mostrados en la siguiente figura.

- Memoria de Programa (Interna y Externa).
- Memoria de Datos (Externa).
- Archivo de Registros(Interno).

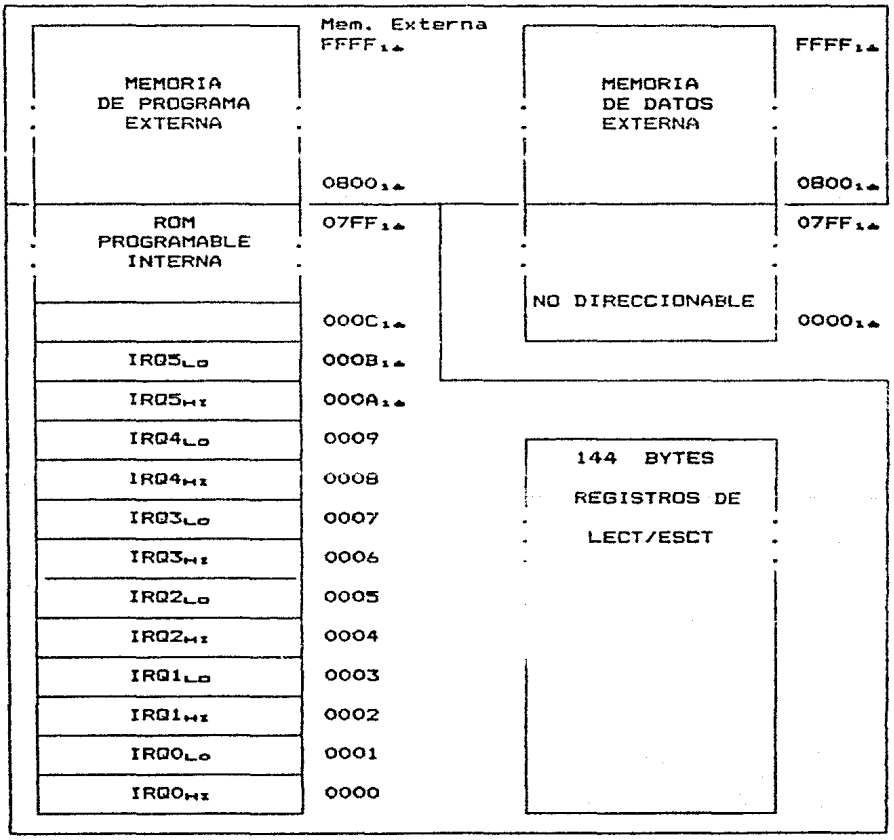


FIGURA (2.10)

1.- MEMORIA DE PROGRAMA.

El Contador de Programa de 16 bits direcciona 65,535 (64 KB) de espacio de memoria de programa.

La memoria de programa consiste de dos secciones: Una Interna formada con los primeros 2048 bytes contenidos en una Memoria ROM integrada en el Z8. A partir de la dirección 2048 en adelante el Z8 accesa memoria de programa externa.

La memoria de programa es continua, es decir, la memoria de programa externa es una extensión lógica de la Interna. Las instrucciones que accesan memoria de programa no hacen distinción entre una y otra.

Los primeros 12 bytes de memoria de programa están reservados para 6 vectores de interrupciones de 16 bits. Cuando ocurre una interrupción, el control del programa es pasado a una Rutina de Servicio, cuya dirección de inicio es apuntada por la localidad indicada en el vector de esa interrupción en particular. La primera localidad disponible para uso de programa es la 12 (OCH) e inmediatamente después de un RESET, el contador de programa apunta a esta dirección.

2.- MEMORIA DE DATOS.

La memoria de datos puede incluirse en el espacio de 62KB destinado a memoria de programa externa, o bien, separarse y aumentar hasta 62KB de memoria exclusiva de datos. En este último caso, la salida DM se encarga de distinguir entre la memoria de datos y la de programa.

Si la memoria de programa comparte espacio de direcciones con la memoria de datos, entonces, todas las instrucciones que accesan memoria de datos externa, seleccionan localidades de memoria de programa. Pero si la primera tiene su propio espacio de direcciones, entonces, algunas instrucciones accesan memoria externa de datos, mientras otras instrucciones continúan los accesos de datos a la memoria de programa externa. La lógica externa de selección determina si la memoria de programa y la de datos ocupan un mismo espacio o si ocupan espacios separados.

Cuando se emplean espacios separados, no quiere decir con esto, que el espacio direccionable aumenta hasta 128KB, ya que el programa en código objeto siempre resida en memoria de programa; y no hay forma de accesar instrucciones en memoria de datos. El Z8 está diseñado asumiendo que en ésta solo se almacenan datos.

- Memoria Externa.

Antes de que las referencias a memoria externa puedan ser hechas por alguna instrucción. El usuario debe configurar los puertos 0 y 1 apropiadamente.

Los dos espacios de memoria externa (de Datos y de Programa) pueden ser usados como un espacio de 62KB, o como dos espacios separados de 62KB cada uno.

Una sutil característica del Z8, es que puede ser usado para minimizar el número de líneas de E/S asignando al rol de salidas de direcciones por medio de aplicaciones de memoria medium-sized. Esta característica permite al usuario salida de direcciones a 10K de memoria, con solo 12 líneas de direcciones, más las líneas de control DM, DS y R/W.

Ordinariamente, 12 líneas de direcciones más DM pueden direccionar solo 4KB en ambos espacios: de programa y de datos (En realidad un total de 6KB porque los primeros 2KB de memoria de datos no son direccionables. Ambos DS o R/W pueden proveer un treceavo bit de dirección, si la aplicación requiere entre 4 y 6KB de memoria de programa ó 2 a 4KB de memoria de datos). Solo el puerto 1 y el nibble bajo del puerto 0 necesita ser asignado al rol de direcciones de salida. Si esta característica no es utilizada, el nibble alto del puerto 0 debe ser usado para salida auxiliar de direcciones.

STACK.

El Z8 usa un Stack, el cual puede residir entre los Registros de Propósito General o en Memoria Externa. La localidad del Stack es seleccionada con un código de control que el usuario carga en el Registro de Control Modo Puerto 0 y 1 (R248).

Para el manejo del Stack externo, el cual puede residir en algún lugar en la Memoria de Datos entre las localidades 2048 y 65535, se usa un apuntador al stack de 16 bits (R254 y R255).

Para el manejo del Stack Interno, el cual reside entre los 124 Registros de Proposito General (R4 a R217), se usa un apuntador al Stack de 8 bits R255.

Durante una instrucción CALL, el contador de programa o durante un ciclo de interrupción, el contador de programa y los registros de banderas se salvan automáticamente en el Stack.

Las instrucciones PUSH y POP pueden salvar, en cualquier registro del archivo, con la excepción de los registros de solo escritura.

Las instrucciones RET e IRET recuperan el valor salvado, ya sea, del Contador de Programa o del Contador de Programa y el Registro de Banderas, según corresponda.

El Registro FF_{16} apunta el byte de orden bajo de la dirección del Stack, mientras el FE_{16} apunta al byte de orden alto de la dirección del Stack.

3.- ORGANIZACION DE REGISTROS INTERNA O ARCHIVO DE REGISTROS.

El archivo de registros cuenta con 144 bytes de acceso aleatorio, los cuales incluyen 124 Registros de Propósito General; 16 Registros de Control y 4 Puertos de E/S, como se puede observar en la figura (2.22).

a) Registros de Propósito General.

Existen 124 registros de propósito general cuyas direcciones van de la 04H a la 7FH. Cualquiera de estos registros puede funcionar como: acumuladores, apuntadores a direcciones, registro indice, o formar parte de un stack interno.

Las instrucciones del Z8 accesan a los registros en forma directa con un campo de direcciones de 8 bits, o con uno de 4 bits. Esto último se logra empleando un registro que sirve de apuntador a grupos de 16 registros, este método salva bytes y reduce el tiempo de ejecución de programa. En este modo de direccionamiento de 4 bits, el archivo de registros es dividido en 9 grupos de Registros de Trabajo, cada uno ocupa 16 localidades continuas. El apuntador a registros (que es uno de los registros de control), contiene el nibble más significativo. Este modo de direccionamiento será explicado ampliamente en el Capítulo 3.

b) Registros de Control.

Los 16 Registros de Control con que cuenta el Z8 van del R240(F0H) al R255(FFH) (Las direcciones 80H-EFH no están implementadas), y sus funciones son diversas. Basta decir por el momento que algunos de ellos funcionan como STATUS, mientras que otros indican la configuración que debe adoptar el Z8. Posteriormente analizaremos cada uno de estos registros y su programación.

c) Registros de Puertos de E/S.

Los 4 primeros registros tienen la función de trabajar como recepción o salida de datos hacia los puertos respectivos, sin introducirnos más en ellos aprovechamos la ocasión para comenzar el estudio de los puertos.

PUERTOS

- El Z8 tiene 4 puertos de E/S de 8 bits, representados por P00-P07, P10-P17, P20-P27, y P30-P37.

- Los Puertos 0, 1 y 2 son bidireccionales, pero tienen capacidad para ser asignados en distintas formas.

- Los 4 bits de orden bajo y los 4 bits de orden alto del puerto 0 de E/S son asignados a entrada o salida como grupo.

- Los 8 pins del Puerto 1 de E/S deben ser simultaneamente asignados a entrada o salida.

- Los pins del Puerto 2 de E/S pueden ser asignados individualmente a entrada o salida.

- Los pins del Puerto 3 de E/S están permanentemente asignados. Los 4 pins de orden bajo quedan asignados siempre a entrada, mientras que los 4 de orden alto son asignados siempre a salida.

En resumen las opciones de direccionamiento Entrada/Salida de los puertos del Z8 son las mostradas en la figura 2.12.

Como se muestra en esta figura, el Puerto 2 opcionalmente puede tener salidas tipo Open Drain. Esta cualidad es única y se puede seleccionar por programación cargando el código apropiado en el Registro de Control asignado al Puerto 2.

Los Puertos 0, 1 y 2 pueden operar por si solos o con ayuda de señales de enlace (Handshake), estas señales son opciones de operación del Puerto 3. En la figura (2.2) son mostradas como RDY0 RDY2 y DAV0 a DAV2.

Los Puertos 0 y 1 opcionalmente trabajan como buses de Datos/Direcciones conectando al Z8 con Memoria Externa, además de la opción de Alta-Impedancia.

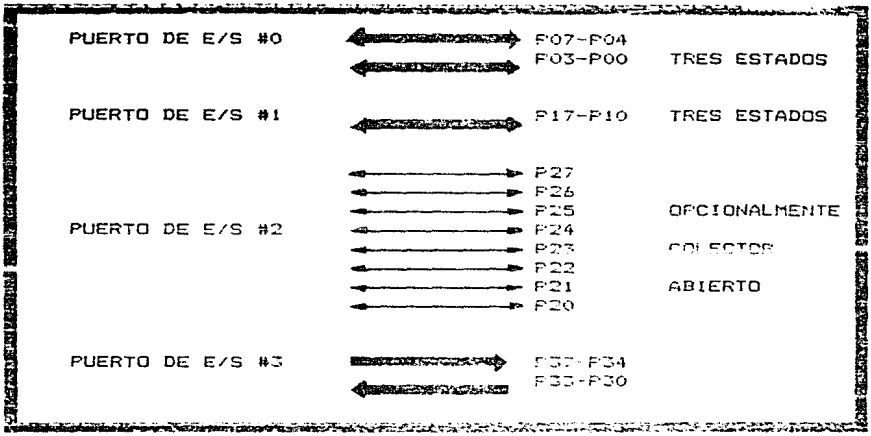


FIGURA (2.12)

Estructura General de los Puertos 0, 1 y 2.

Cada bit de los Puertos 0,1 y 2 tienen un registro de entrada, uno de salida, un buffer asociado y una lógica de control.

Cuando un bit de los Puertos 0,1 y 2 es configurado como salida, la escritura de ese bit en el registro de puerto correspondiente (00H-03H), hace que el dato sea guardado en el registro de salida. Si un bit de salida es leído, se obtiene el dato presente en el pin externo. Si la salida tiene sus resistencias de pullup activas, esto es equivalente a leer el registro de salida, sin embargo, si un bit del Puerto 2 es definido con salida Open-Drain, el dato que se obtiene puede no ser idéntico al registro de salida, ya que este valor puede ser forzado en el pin de entrada por la lógica externa.

Cuando un bit de los Puertos 0, 1 y 2 es definido como entrada, al leer ese bit se obtiene el valor presente en el pin externo. La única excepción es para los bits controlados por señales de enlace (Handshake). En este caso se obtiene el valor del dato amarrado en el registro de entrada. Se puede también escribir en los bits de entrada; pero en este caso el dato es almacenado en el registro de salida y no puede ser recuperado.

DIAGRAMA DE BLOQUES DE LOS PUERTOS 0, 1 Y 2

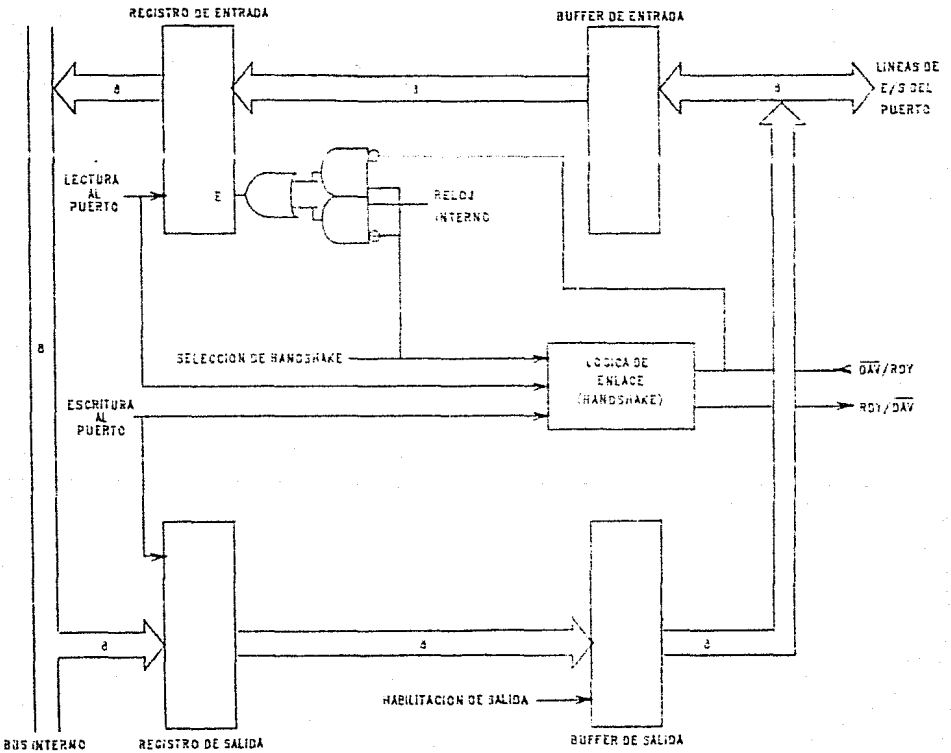


FIGURA (2.13)

Sin embargo, si este bit es reprogramado como salida, el dato guardado en el registro de salida se refleja en el pin externo. Este mecanismo permite que el usuario inicialice las salidas antes de conectarlas a sus respectivas cargas.

PUERTO 2.

Cada bit del Puerto 2 puede ser configurado individualmente como salida o como entrada programando R246, que es el registro de control asociado a este puerto y que se identifica por P2M.

Este puerto es accesado a través del registro R2 y se escribe un dato en el cuando en una instrucción se especifica a R2 como destinatario.

Para leer el Puerto es necesario especificar a R2 como la fuente de una instrucción.

El Puerto 2 puede ser controlado por señales de enlace (Handshake) programando el registro de control R247, cuya función es asignar los modos de operación del Puerto 3. Este registro se conoce por P3M. Cuando este es el caso los pins P31 y P36 son configurados para líneas de enlace de entrada DAV2 y RDY2, o bien RDY2 y DAV2 para líneas de enlace de salida. La dirección de estas señales se determina por la asignación E/S del bit 7 del Puerto 2.

La opción de salidas Open-Drain de este puerto, también es controlada por la programación de P3M.

En cualquier configuración del Z8 las líneas del Puerto 2 siempre están disponibles para operaciones de E/S.

PUERTO 0

Los 8 bits de este puerto se dividen en 2 nibbles que pueden ser configurados independientemente como E/S con o sin señales de enlace (Handshake), o como complementos del Bus de Direcciones.

La selección se programa con el registro de modos del Puerto 0 y 1, R248, o P01M.

Cuando un nibble de este puerto es usado en el modo E/S, es accesado por el nibble correspondiente del registro R0, el registro de salida de este puerto es almacenado cuando se escribe un dato especificando a R0 como el registro destino de una instrucción.

Mediante el registro P3M, cuando el Puerto 0 es usado como E/S, se puede programar su opción de líneas de enlace. En esta

opción los pins P32 y P35 del Puerto 3 trabajan como líneas de control de enlace DAV0 y RDY0 para entrada o RDY0 y DAV0 para enlace de salida. La asignación de las señales de enlace de P32 y P35 son determinadas por la opción E/S definida por el nibble superior del Puerto 0.

Cuando se necesita memoria externa dependiendo del espacio de direcciones requerido, el Puerto 0 puede programarse para suministrar los bits A8-A11 (nibble inferior) o A8-A15 (ambos nibbles) del bus de direcciones, esto es, si el espacio a direccionarse requiere 12 bits o menos, el nibble superior del Puerto 0 puede ser programado independientemente como E/S, mientras el nibble inferior es usado para direccionamiento. Cuando la aplicación requiere más de 12 bits, es necesario programar ambos nibbles para proveer el byte más significativo de un bus de direcciones de 16 bits. Cuando cualquiera de los nibbles es definido como bits de direccionamiento, programando adecuadamente P01M se obtiene la opción de alta impedancia que también afecta al puerto 1 y las señales de control AS, DS y R/W.

PUERTO 1.

El Puerto 1 puede ser programado como E/S de un byte en conjunto con opción de señales de enlace, o como bus Dirección/Datos para interface de memoria externa. La configuración es determinada programando el registro de modos de Puertos 0, 1 , P01M o R248.

Cuando es usado el modo byte de entrada o byte de salida, el puerto es accesado a través del registro R1. Se escribe en el puerto cuando se especifica a R1 como el registro destinatario de una isntrucción y el dato es guardado en el registro de salida del puerto. El puerto es leido por especificacion de R1 como fuente de una instrucción.

La opción de señal de enlace de este puerto se obtiene programando el registro de modos del Puerto 3. En este caso los pins del Puerto 3: P33 y P34 trabajan como DAV1 y RDY1 en modo entrada y viceversa en modo salida.

Para accesar memoria externa, el Puerto 1 debe ser programado en su modalidad de bus Dirección/Datos (AD0-AD7).

En esta configuración los 8 bits de direcciones menos significativos (A0-A7) se presentan multiplexados con los 8 bits de datos (D0-D7).

Las localidades de memoria externa son accesadas a través del Puerto 1. Si la aplicación requiere más de 256 localidades externas, el Puerto 0 debe aplicarse para obtener las líneas de dirección adicionales.

NOTA: Cuando el Puerto 1 es configurado como bus multiplexado Datos/Direcciones, R1 no puede ser accesado como registro.

Similarmente al Puerto 0, el Puerto 1 puede trabajar en estado de alta impedancia, permitiendo que el Z8 comparta recursos en una configuración multiprocesador y aplicaciones DMA (Acceso Directo a Memoria). En este caso las transferencias de datos pueden ser controladas asignando P33 como entrada de reconocimiento de Bus, BAK (IRQ1) y P34 como salida petitoria de Bus, BRQ.

SENALES DE ENLACE (HANDSHAKE) PARA LOS PUERTOS 0, 1 Y 2

La tabla (1) resume las opciones de E/S de los Puertos 0, 1 y 2. Como ya se mencionó anteriormente estos puertos pueden funcionar como señales de enlace: En el caso general de entrada se puede ilustrar la función de las señales de enlace como sigue:

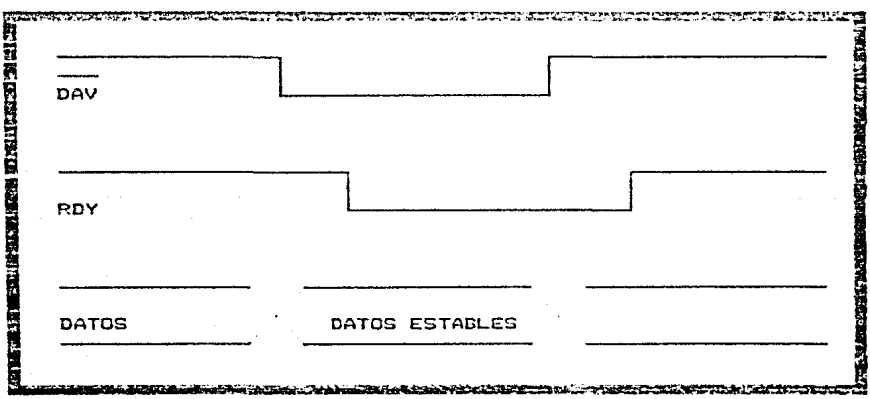


FIGURA (2.14)

PUERTOS	BITS	DIRECCION	SEÑALES DE CONTROL	COMENTARIOS
0	4 (P04-P07)	IN	DAV0 entrada a P32 RDY0 salida a P35	Las direcciones de datos pueden especificarse separadamente por P00-P03 y por P04-P07 (este selecciona las señales IN o OUT).
	o 8 (P00-P07)	OUT	RDY0 entrada a P32 DAV0 salida a P35	
1	8 (P10-P17)	IN	DAV1 entrada a P33 RDY1 salida a P34	La dirección de datos asignada pude ser la misma para todas las lineas del Puerto 1.
		OUT	RDY1 entrada a P33 DAV1 salida a P34	
2	8 (P20-P27)	IN	DAV2 entrada a P31 RDY2 salida a P36	Las direcciones de los pins son asignadas individualmente. La especificación de P27 selecciona las señales de control IN/OUT. Open Drain OUT puede funcionar como bidireccional.
		OUT	RDY2 entrada a P31 DAV2 salida a P36	
		Seudobidireccional		

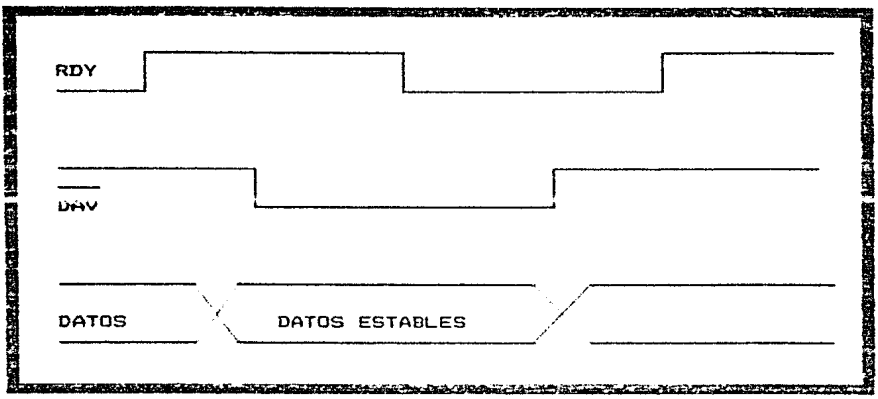
TABLA (1)

Inicialmente RDY tendrá un valor alto y DAV debe tener un valor alto en el pin apropiado del Puerto 3. Cuando la lógica externa introduce nuevos datos, simultaneamente debe poner a DAV en un nivel bajo.

El nivel bajo de DAV hace que la lógica de E/S del puerto amarre los datos en el registro de entrada. El flanco negativo de DAV puede producir una petición de interrupción, ya que ésta ocurre en los pins 1, 2 o 3 del Puerto 3. Esto es asumiendo que las interrupciones han sido apropiadamente habilitadas. En caso contrario el Z8 puede usar un programa de muestreo para detectar la entrada de datos. Esto último se puede lograr leyendo el estado en el registro de peticiones de interrupciones IRQ, o bien, leyendo el registro 3 para determinar el momento en que se está pidiendo una entrada de datos.

La lógica externa no puede escribir un dato en el registro de entrada si el dato anterior todavia está esperando ser leído, ya que debe esperar a que la señal RDY sea forzada baja, lo cual ocurre cuando el CPU ha leído el contenido del registro de entrada. El nivel bajo de RDY debe indicarle a la lógica externa que debe levantar el nivel de DAV, para que la lógica del puerto regrese a RDY a su valor inicial o indicarle a la lógica externa que el dato previamente transmitido ha sido leído por el CPU, y que el puerto está listo para aceptar un nuevo dato.

El diagrama siguiente ilustra el caso de salidas con enlace.



FIGURA(2.15)

Inicialmente la lógica externa debe enviar a RDY baja mientras el Z8 manda DAV alta. Cuando la primera está lista para recibir datos, debe indicarlo así con RDY alta. El programa ejecutado por el Z8 podría transmitir datos al puerto de salida en cualquier momento, sin embargo, la lógica del puerto no permitirá que DAV sea baja hasta que los datos han sido escritos en el registro del puerto y la entrada de RDY haya sido llevada a su nivel alto por la lógica externa.

Una vez que RDY es alta, la lógica externa debe esperar el nivel bajo de DAV, que indica que nuevos datos de salida estan disponibles. Después de leer los datos, la entrada RDY debe volver a su nivel bajo y consecuentemente la lógica del puerto regresa a DAV a su nivel alto. El flanco positivo de RDY puede producir una petición de interrupción siempre y cuando estas sean habilitadas. El programa de control de datos de salida del Z8 debe esperar esta petición de interrupción, o bien, muestrear el bit asociado del registro IRQ, antes de enviar más datos al puerto de salida.

Tan pronto como la lógica del puerto presenta a DAV alta, la lógica externa está libre para volver a RDY a su nivel alto e indicar que está lista para recibir nuevos datos.

PUERTO 3.

Este puerto tiene una estructura diferente a los puertos anteriores, ya que tiene un Registro de Salida de solamente 4 bits, asociado al nibble permanentemente asignado a salida. Cuando se escribe en el Puerto 3, el dato es almacenado en este registro, cuando es leído, el dato que se recupera está compuesto de los 4 bits en los pins de entrada y el dato almacenado en el registro de salida. Si las salidas del Puerto 3 son usadas para

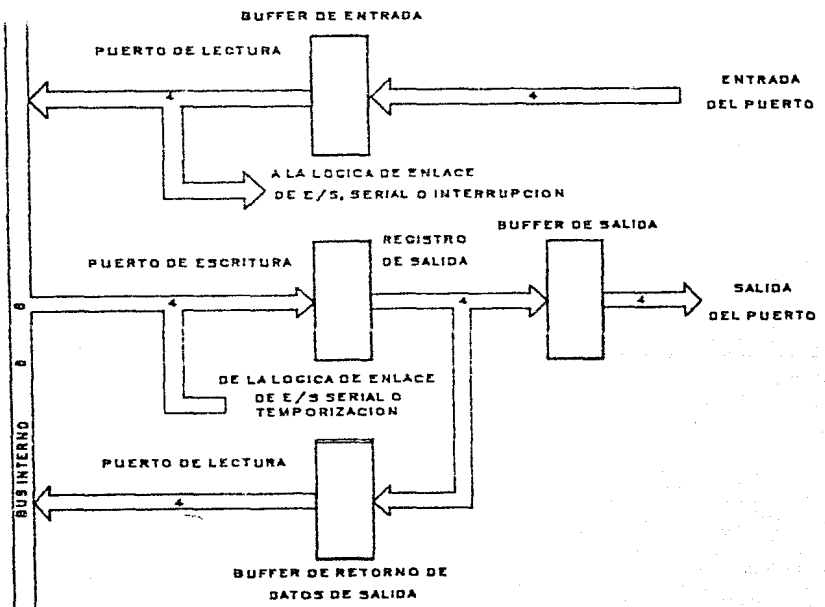


FIGURA (2.16)

funciones tales como: Salida Serial; Control de Señales de Enlace o Salida del Temporizador; entonces no se puede escribir en el registro de salida.

Ya hemos introducido algunas de las funciones del Puerto 3 y debe sernos familiar que las líneas de este puerto deban ser configuradas como E/S o líneas de control. Las múltiples opciones de este puerto se programan en el Registro de Modos del Puerto 3: P3M. En cualquier caso la dirección de las 8 líneas es fija y P30 a P33 son entradas, mientras P34 a P37 serán salidas.

El Puerto 3 es accesado mediante el Registro R3, cuando se lee éste se obtiene el dato presente en los 4 pins de entrada P30-P33 y el dato almacenado en el registro de salida.

Los 4 bits del Puerto de Salida pueden ser leidos solamente si son asignados como salidas de datos.

Para E/S Serial, las líneas P30 y P37 deben ser programadas como entrada serial (Sin) y salida serial (Sout) respectivamente, para más detalles ver la sección destinada a Puerto Serie.

El resto de las funciones de control de este puerto se programan asimismo por P3M en cualquiera de las siguientes opciones: Señales de Enlace para los Puertos 0, 1 y 2 (DAV y RDY); 4 Señales de Petición de Interrupciones Externas (IRQ0-IRQ3); Salidas y Entradas para Temporización (Tin y Tout); y Selección de Memoria Externa de Datos (DM).

FUNCION	LINEA	SEÑAL
Handshake	P31 P32 P33 P34 P35 P36	DAV2/RDY2 DAV0/RDY0 DAV1/RDY1 RDY1/DAV1 RDY0/DAV0 RDY2/DAV2
Petición de Interrupción	P30 P31 P32 P33	IRQ3 IRQ2 IRQ0 IRQ1
Contador/ Temporizador	P31 P32	Tin Tout
Salida Status	P34	DM

TABLA (2)

Debe hacerse notar que las 4 lineas de entrada, sin importar la configuración elegida, pueden ser siempre usadas para entrada de petición de interrupciones, sin embargo, la interrupción será generada solamente si el registro máscara de interrupciones (IMR) R251 es programado apropiadamente.

BUSES DE DIRECCIONES/DATOS.

Ya hemos comenzado a hablar de la manera en que el Z8 accesa memoria externa, sin embargo, existen algunas consecuencias no evidentes de esta estructura, que estudiaremos a continuación.

Se pueden crear buses de Dirección/Datos en 3 incrementos. Para aplicaciones que requieran una pequeña expansión de memoria se puede configurar solamente el Puerto 1 para accesarla, dejando al Puerto 0 en su opción E/S. Esta pequeña expansión permite que el Z8 accese hasta 256 bytes de Memoria Externa. Si la memoria de datos y la de programa tienen espacios direccionables separados, el número de localidades de memoria aumenta hasta 512 bytes.

Si los 256 bytes de memoria externa son insuficientes en el primer nivel de expansión, los bits 0 al 3 del Puerto 0, se suman al bus de direcciones para obtener un total de 12 lineas de direccionamiento, donde los 8 primeros bits son multiplexados con el bus de datos. En esta configuración se pueden tener un total de 10 kb de memoria, utilizando solamente 12 bits de direcciones con 6 KB de Memoria de Programa y 4 KB de Memoria Externa de Datos (Esto es la versión del Z8 con 2 KB de Memoria de Programa Interna).

Mem. de Programa Dirección	Mem. de Datos Dirección	Dirección Física en los Puertos 0 y 1	A11	DS y R/W
0000-07FF	....	000-7FF	0	Inactiva
0080-0FFF	0800-0FFF	800-FFF	1	Activa
1000-17FF	1000-17FF	000-7FF	0	Activa

TABLA (3)

Para aplicaciones con uso intensivo de memoria, en el siguiente nivel se puede generar un bus de direcciones de 16 líneas, programando la totalidad del Puerto 0 para obtener los 8 bits más significativos de este bus.

En cualquiera de estos niveles cada vez que un ciclo de máquina accese memoria sin importar si es: Memoria de Programa Interna, Memoria de Programa Externa o Memoria de Datos Externa, en el bus de direcciones aparecerá una dirección de memoria. Esto se debe a que los direccionamientos a memoria del Z8 son siempre de 16 bits, pero la lógica del Z8 que controla los Buses ignora los bits superiores de direcciones cuando el bus externo es configurado con menos de 16 bits. Así, un bus de direcciones de 8 bits solo permitirá la salida de los primeros 8 bits de la dirección de 16 líneas de selección, y asi sucesivamente para buses de 12 y 16 bits como se muestra a continuación.

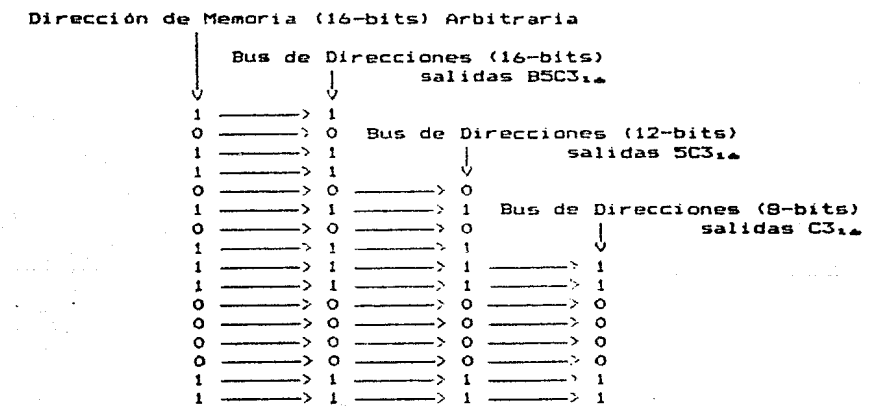


FIGURA (2.17)

El Z8 usa las señales de control AS, DS, R/W y DM para diferenciar entre los direccionamientos de Memoria Interna y los de Memoria Externa. Estas 4 señales son activas solamente cuando el Z8 es configurado con un bus de direcciones externas. Cuando este es el caso AS será pulsada baja cada vez que una dirección este presente en el bus externo a menos que haya sido programada en estado de Alta-Impedancia.

DS será baja cada vez que el Z8 envíe o espere recibir datos.

R/W es baja para diferenciar a los ciclos de máquina de escritura a memoria.

DM cuando es necesaria debe ser programada, ya que comparte el pin con el bit 4 de E/S del Puerto 3. Cuando es seleccionada esta señal es baja durante las instrucciones que específicamente accesan memoria de datos externa.

Las señales de control AS, R/W y DM no son activadas cuando cualquier dirección entre la 0000H y la 07FFH es seleccionada, aunque la dirección aparece físicamente en el bus externo.

De lo anterior se desprende que la lógica externa de selección de memoria debe interpretar el estado de DS y R/W para evitar que una dirección de memoria interna seleccione a una de memoria externa.

El bus externo de direcciones de 16 bits puede seleccionar 63468 bytes de memoria externa y doblar esta cantidad si se programa DM para diferenciar entre localidades de memoria de programa y datos con la misma dirección.

INTERRUPCIONES.

El Z8 puede responder a 6 peticiones de interrupción de 6 fuentes diferentes: Las 4 líneas del Puerto 3 (P30-P33), Entrada Serie, Salida Serie y los dos Contadores/Temporizadores. Las interrupciones son mascarables y su prioridad programable, usando el Registro de Prioridad de Máscaras (IMR) R251 y el Registro de Prioridad de Interrupción (IPR) R249. Las 6 interrupciones pueden ser globalmente inhibidas apagando el bit maestro habilitador de interrupciones en el Registro de Máscaras de Interrupciones (IMR) R251.

Cada una de las 6 interrupciones tiene asociado un vector de direcciones. La dirección de la primera localidad de la rutina de servicio correspondiente es almacenada en los primeros 12 bits de memoria de programa interna.

Es importante anotar que un flanco negativo en cualquiera de los pins P30 a P33 pueden provocar una petición de interrupción, si la interrupción correspondiente ha sido habilitada. Como las señales de entrada a estos pins pueden tener distintas interpretaciones, el programa es responsable del manejo apropiado de las interrupciones que puedan ocurrir. Un caso especial es IRQ3 cuando el Puerto P30 es usado como entrada serial de datos, entonces, las transiciones alto-bajo en este pin

Interrup-ción	Vector de Direcciones	Tipo	Condición de Petición
IRQ0	0.1	Externa	Una entrada con flanco positivo en P32. Sirve para peticiones de interrupción generales o entrada para señales de enlace DAV0/RDY0 para el puerto 0
IRQ1	2.3	Externa	Flanco positivo a la entrada de P33. Sirve para interrupciónes generales o entrada para señales de enlace DAV1/RDY1 para el puerto 1
IRQ2	4.5	Externa	Flanco positivo a la entrada de P31. Sirve para interrupciones generales o entrada para seña - les de enlace DAV2/RDY2 para el puerto 2.También sierve para una interrupción externa de los temporizadores
IRQ3	6.7	Externa Interna	Una transición de un flanco positivo a la entrada P30. IRQ0 sirve para peticiones generales de interrupción. Petición de interrupción para datos seriales de entrada.
IRQ4	8.9	Interna	Interrupciones del temporizador T0 o salida de datos seriales.
IRQ5	10.11	Interna	Interrupciones del temporizador T1.

TABLA (4)

no generan peticiones de interrupción; más bien, la lógica de entrada de datos serie genera una petición de interrupción cuando un caracter ha sido ensamblado y deberá ser leido.

Las interrupciones IRQ4 e IRQ5 son generadas solamente en respuesta a condiciones internas. IRQ4 está compartida por el Contador/Temporizador 0 y la salida de datos seriales, ya que estas 2 son mutuamente exclusivas.

Cuando ocurre una petición de interrupción y se encuentra habilitada la interrupción correspondiente, el Z8 entra en un ciclo de interrupción de máquina, el cual deshabilita globalmente todas las interrupciones subsecuentes, salva el contador de programa junto con el registro de banderas y salta a la dirección contenida dentro de la localidad del vector de interrupción correspondiente. En este punto el control pasa a la rutina de servicio de interrupción.

La figura (2.8) ilustra el ciclo de interrupción, cuando una petición ocurre.

Las interrupciones pueden ser rehabilitadas por la rutina de servicio con la instrucción EI, permitiendo el anidamiento de interrupciones. Las interrupciones también pueden ser rehabilitadas automáticamente por ejecución de una instrucción de retorno de interrupción (IRET) como la última instrucción de la rutina de servicio. IRET también restaura el contador de programa y el registro de banderas previos a la petición de interrupción.

CONTADORES/TEMPORIZADORES.

El Z8 cuenta con 2 Contadores/Temporizadores programables de 8 bits (T0 y T1) complementados por sus Prescaladores programables de 6 bits. T0 tiene menos opciones programables que T1 y funciona también como generador de Baud/Rate para las operaciones de E/S Serie del Z8. Por lo tanto, es siempre preferible usar T1 y recordar que no puede disponerse de T0 cuando se emplea la opción E/S Serie.

Ambos Contadores/Temporizadores están basados en una lógica muy elemental. Cada Contador/Temporizador recibe una señal de reloj que es dividida por un prescalador. El valor del prescalador, el cual es almacenado en los registros prescaladores R243 y R245 (PRE0 y PRE1 respectivamente), puede variar entre 1 y 64, ya que en estos registros solo 6 de sus 8 bits son empleados para esta función.

La frecuencia de reloj de entrada, dividida por el valor del prescalador, da el intervalo de tiempo en el cual el contador decrementa su contenido. El valor inicial que puede ser guardado

en el contador varía entre 1 y 256 (00 equivale a 256), programando el contenido de los registros R244 para T0 y R242 para T1.

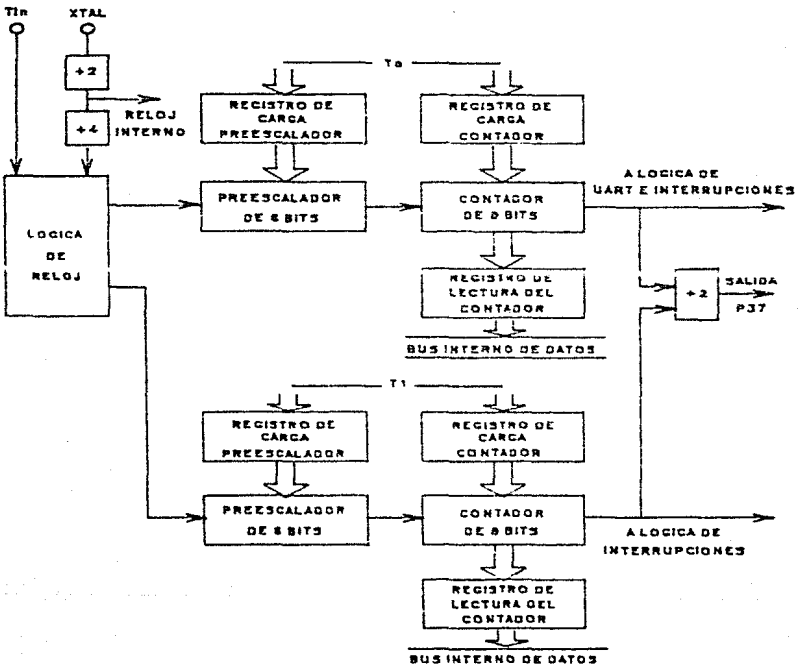


FIGURA (2.18)

Cuando el valor del contador llega al final de la cuenta, una petición de interrupción (IRQ4 para T0 o IRQ5 para T1) es generada.

Los contadores pueden activarse, detenerse, reactivarse desde su valor inicial o desde su último valor, todo esto por programación del Registro Modo de Temporizadores (TRM) R241. Así también, puede programarse la operación en alguno de los siguientes modos:

- El contador se detiene al llegar a 0 (Modo de 1 Pasada).

- Su valor inicial es automáticamente recargado y continúa la cuenta (Modo Continuo Modulo-N).

El contenido del contador, más no el del prescalador, puede ser leído en cualquier tiempo sin alterar su valor o modo de conteo.

Relojes de los Temporizadores.

La frecuencia de la señal de reloj del Contador/Temporizador T0 es la frecuencia del cristal externo dividida entre 8. Para el Contador/Temporizador T1 se obtienen las siguientes opciones programables.

1.- La frecuencia de cristal externo dividida entre 8.

2.- Una señal de reloj externa, introducida a través de la entrada P31 (Tin) a una frecuencia de máximo 1 MHz.

3.- Introduciendo una señal de condición externa en compañía de la señal de reloj interno.

Tin puede ser programada para proveer la señal de reloj del Contador/Temporizador T1, en este caso la frecuencia de ésta no es dividida antes de entrar al prescalador, por lo tanto, puede tener un valor máximo de 1 MHz, que equivale a la frecuencia máxima que se puede derivar del reloj interno. No existe frecuencia mínima para la señal de reloj introducida a través de Tin. Esto permite que T1 trabaje como un contador de eventos, ya que la señal en Tin puede tener transiciones aleatorias. Para que esto sea posible el valor del prescalador debe ser 1, de otra manera T1 dividirá la señal en Tin por el valor del prescalador y reportará solo una fracción del total de transiciones.

Cuando T1 trabaja con el reloj interno, Tin puede ser programada para recibir una señal de condición, capaz de activar o detener el reloj interno de los Contadores/Temporizadores. Controlando el reloj de esta manera Tin detiene a T1 cuando es baja y permite su operación cuando es alta, como se muestra en la figura (2.19) en la siguiente página.

Existen otras 2 opciones de programación de Tin, que son: Como Entrada Activadora y Entrada Disparadora no Reactivable del Contador/Temporizador T1 trabajando con señal de reloj interno. En el primer caso una transición alta-baja en Tin produce que el contenido del Registro Contador T1 sea transferida al Contador/Decrementador y comience a decrementarse.

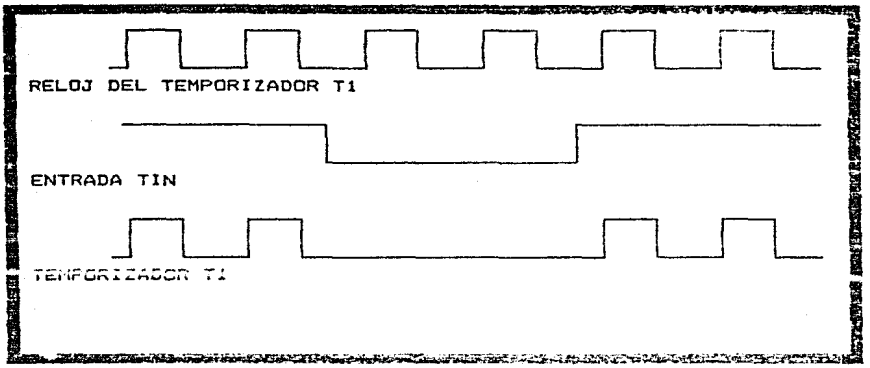


FIGURA (2.19)

Cada vez que Tin presente el flanco negativo, la lógica de T1 será reinicializada sin respetar el estado anterior. El segundo caso es idéntico, solo que después de la primera transición de Tin, las subsecuentes son ignoradas.

El pin P37 Tout puede ser programado como la salida de T0 o T1. En este caso Tout se modifica al final de cada cuenta. Si el temporizador es programado en modo de cuenta continua, la salida por Tout será una onda cuadrada con ciclo de trabajo al 50%. Cada vez que nuevos valores son cargados en T0 o T1, Tout es modificada a su nivel bajo.

T0 Como Generador de Baud/Rate.

El Baud/Rate de la E/S serie de datos es generada por los intervalos de tiempo del decrementador de T0, el cual trabaja a una velocidad 16 veces mayor que el Bit/Rate. Para poder generar Baud-Rates comunmente usados, se debe usar un cristal de 7.3728 MHz entre XTAL1 y XTAL2, cargar los valores iniciales indicados en la tabla (5) en el Contador/Decrementador 0. Esta tabla también muestra como los intervalos de tiempo subsecuentes son generados.

A Cristal Frecuencia (MHz)	B Contador O Reloj =A/8 (KHz)	C Intervalo del Prescalador [1] =3/B (µsec)	D Decremento Presente	E Intervalo Total =CxD (µsec)	F Bit Rate =10^6/(16E) (Bits/Sec)
7.3728	921.6	3.2552	1	3.255	19200
			2	6.510	9600
			4	13.021	4800
			8	26.042	2400
			16	52.083	1200
			32	104.167	600
			64	208.333	300
			128	416.667	150
			175 =	569.661	110 =
1 El Registro Prescalador contiene 3. 2 Error = 0.26%					

TABLA (5)

PUERTO SERIE

Las lineas del Puerto 3 P30 y P37 pueden ser programadas como lineas de E/S Serie para operación Receptor/Transmisor Asincrono Serial Full-Duplex. La estructura de este Receptor/Transmisor es la mostrada en figura (2.20)

La velocidad es controlada por el Contador/Temporizador T0. Los datos a transmitirse son cargados en el Registro R240 y deslizados a través de P37. El dato serie se recibe a través de P30, ensamblado en un caracter de 8 bits y transferido al buffer receptor. El registro R240 es en realidad 2 registros en 1, cuando se escribe en èl se comporta como transmisor, cuando se lee su contenido es el receptor.

La frecuencia obtenida del Contador/Temporizador T0 es dividida entre 16. En la tabla (5) se resumen los valores que es necesario cargar en el contador y el prescalador de T0 para generar los Bauds-Rates más comunes.

En el modo de transmisión, el Z8 automáticamente agrega 1 bit de inicio y 2 de parada, al dato transmitido. La paridad puede ser seleccionada por programación del registro R247. De cualquier forma, siempre serán transmitidos 8 bits, con o sin paridad. Si la praridad es seleccionada, el octavo bit es la paridad par. Entre caracteres, la salida de P37 es alta, para mantener la condición de marca.

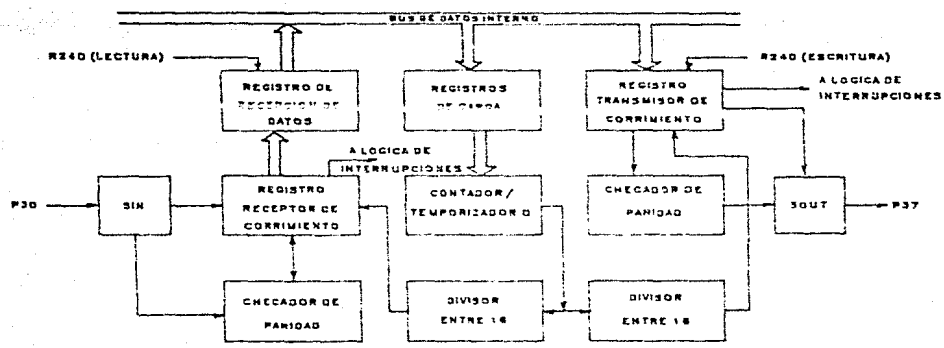


FIGURA (2.20)

En el modo de recepción el formato del dato debe tener 1 bit de parada. Si la paridad es seleccionada, el bit 7 del dato recibido (Bit de paridad) es reemplazado por una bandera de error de paridad.Un error debe poner a la bandera en un 1 lógico. Lo anterior se puede resumir en la siguiente figura, que muestra los formatos de datos seriales.

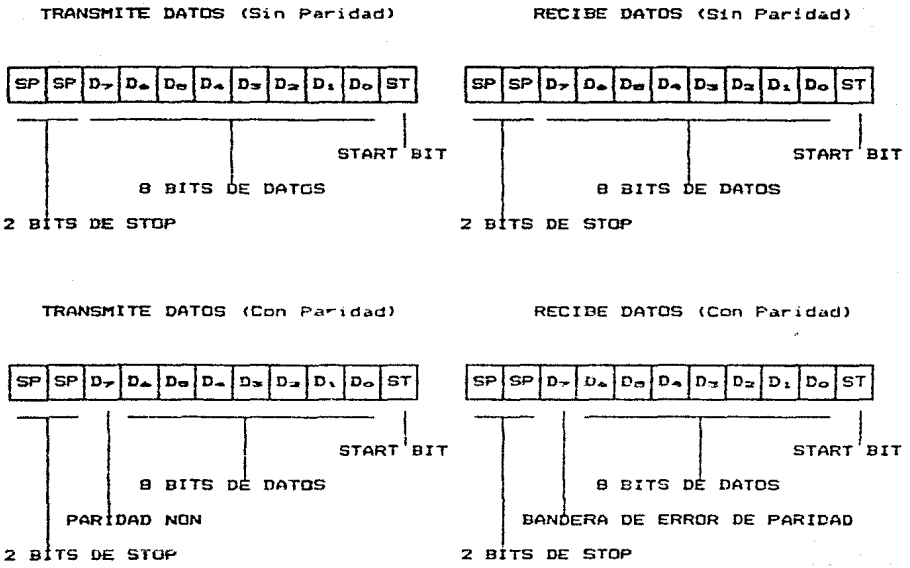


FIGURA (2.21)

Una petición de interrupción IRQ3 es generada cuando un caracter es transferido en el bus receptor. Deben tomarse precauciones de atender esta interrupción, ya que este registro no está protegido contra sobrescrituras. Un caracter transmitido también genera su petición de interrupción (IRQ4) y como receptor el buffer transmisor también puede ser reescrito.

3.- Registros de Control.

Hasta el momento hemos analizado todos los elementos constituyentes del Z8 sin introducirnos en la forma en que se programa el conjunto y la manera en que se pueden configurar las diversas opciones del Z8.

La programación de los diversos modos de operación de los módulos del Z8 se realiza introduciendo códigos de control apropiados en algunos de los registros de control. El resto de los registros de control tienen la función de indicar el STATUS del Z8. A continuación se explicará la programación y la función en forma individual de cada uno de los 16 Registros de Control.

DIAGRAMA DEL ARCHIVO DE REGISTROS

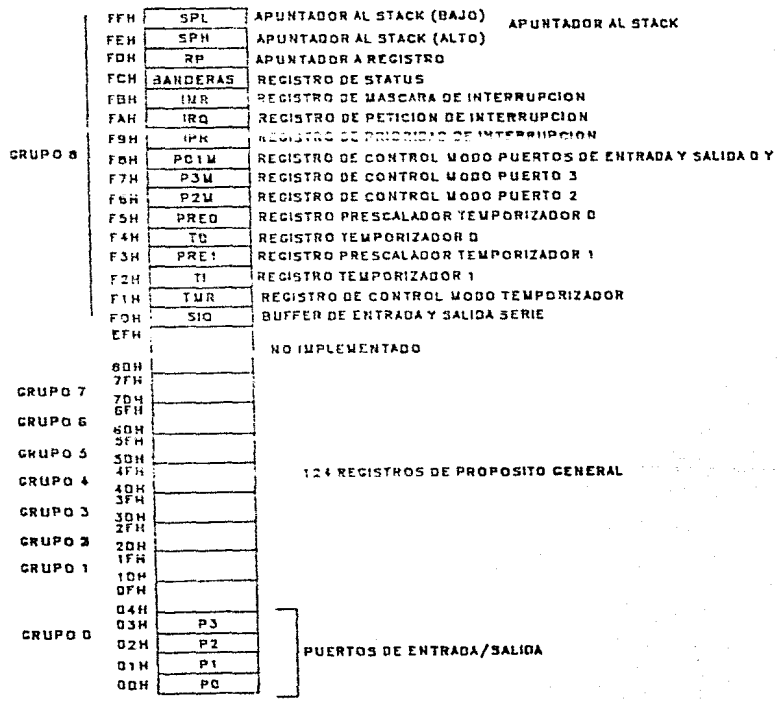
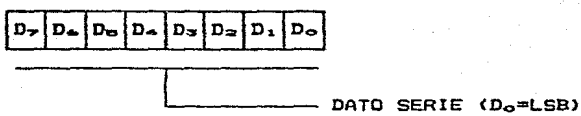


FIGURA (2.22)

SIO R240 (F0)

REGISTRO DE ENTRADA/SALIDA SERIE

(Lectura/Escritura)



Lectura de R240 = Recepción de Datos

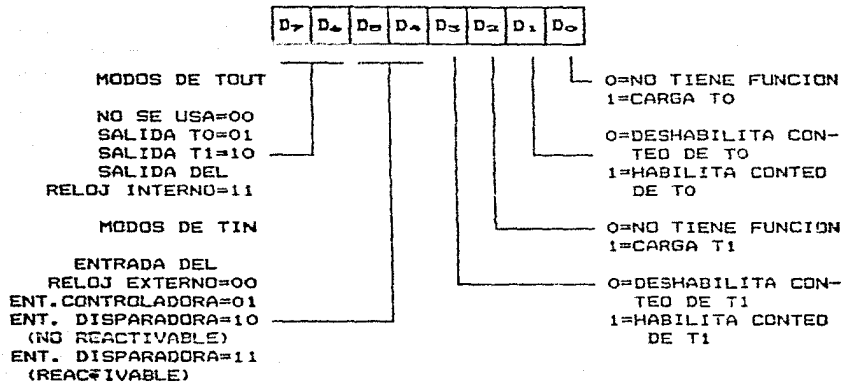
Escritura de R240 = Transmisión de Datos

Cuando el bit D6 de registro R247 tiene un 1 lógico, se configura a P30 y P37 como líneas de E/S serie respectivamente, entonces el registro R240 es usado como el registro receptor/transmisor de los datos. Cuando este registro es leido se recupera el caracter en el buffer receptor; Cuando se escribe, el caracter es introducido en el transmisor. Si la paridad no es seleccionada, los 8 bits de R240 forman el dato. En caso contrario D7 contiene la paridad par durante la transmisión serie y una bandera de error de paridad en la recepción.

TMR R241 (F1)

REGISTRO DE MODO TEMPORIZADOR

(Lectura/Escritura)



Este registro selecciona los modos de reloj de los Contadores/Temporizadores y controla la operación de TO y T1.

(D0) Carga el Contador TO. El contenido de los registros del Prescalador y de Carga TO, son transferidos al contador y al prescalador un período de reloj después de que este bit ha sido prendido (1). D0 adquiere automáticamente un valor de 0 después de que se realiza la carga del contador o después de un reset maestro.

(D1) Habilita Contador T0. D1 habilita o deshabilita las operaciones de conteo de T0. Cuando es prendido, los valores de T0 y de su prescalador empiezan la cuenta decreciente usando el reloj interno. Cuando se apaga, el conteo se detiene. Este bit se apaga después de un reset maestro. Es permitido prender el bit de carga D0 y el de habilitación de conteo simultaneamente.

(D2) Carga T1. Este bit tiene las mismas funciones para T1 como D0 las tiene para T0.

(D3) Habilita Contador T1. Este bit tiene las mismas funciones para T1 como D1 las tiene para T0.

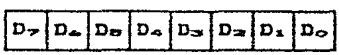
(D4,D5) Modos de Entrada de Temporización externa. D4 y D5 son codificados para definir 4 modos de operación de la Entrada de Temporización (Tin). P34 debe ser previamente definido como Entrada de Temporización Externa (R243,D1). La tabla siguiente ilustra la codificación de D4 y D5.

D5	D4	Usos de Tin	Comentarios
0	0	Entrada de Reloj Externo	Tin es usado como reloj externo para T1. En este modo, el reloj externo maneja directamente al prescalador, saltandose al divisor entre 4 interno.
0	1	Entrada Habilitadora del Reloj Interno (Activa alta)	T1 trabaja con la frecuencia del reloj interno (XTAL dividida por 8) si el nivel de Tin es alto, si es bajo T1 se detiene. El flanco negativo de Tin genera una interrupción.
1	0	Entrada Activadora no Reactivale.	T1 es cargado y sincronizado el reloj interno, solo después de que una transición Alto-Bajo ocurre en esta entrada. Una nueva transición no afecta la operación T1 hasta después del final de la cuenta.
1	1	Entrada Activadora Reactivable.	T1 es cargado y sincronizado el reloj interno solo después de que una trasición Alto-Bajo ocurre en Tin. Cuando pulsos subsecuentes ocurren en Tin, se vuelve a cargar el valor inicial de T1 y el conteo es restaurado.

T1 R242 (F2)

REGISTRO DEL CONTADOR TEMPORIZADOR T1

(Lectura/Escritura)



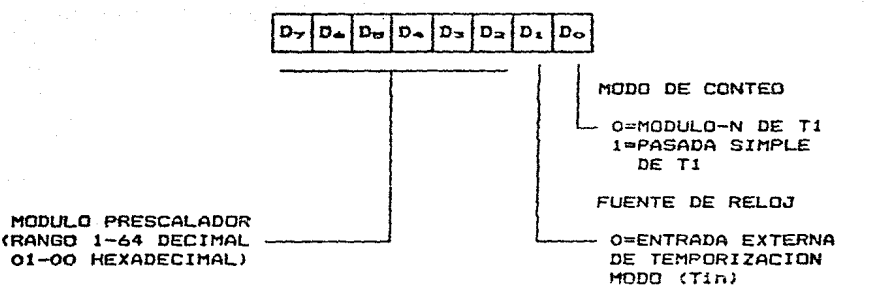
VALOR INICIAL DE T1 (CUANDO SE ESCRIBE)
(RANGO 1-256 DECIMAL 01-00 HEXADECIMAL)
VALOR ACTUAL DE T1 (CUANDO SE LEE)

Este registro funciona de dos formas: Como un registro de escritura, guarda el valor inicial de T1; como un registro de lectura, contiene el valor de la cuenta corriente de T1. R242 puede ser cargado con valores en el rango de 1 (01H) a 256 (00H). Este valor es transferido a T1 cuando el bit de carga de T1 (R241,D2), es puesto en 1 y subsecuentemente el conteo decreciente comienza. Una petición de interrupción es generada cuando el fin de la cuenta es alcanzado.

PRE1 R243 (F3)

REGISTRO DE CARGA DEL PREESCALADOR DE T1

(Lectura/Escritura)



Este registro guarda el valor inicial del prescalador T1, y define la fuente para el reloj T1 y el modo de conteo.

(D0) Modo de Selección. Cuando este bit es prendido, T1 opera en modo de conteo de una sola pasada, en el cual el valor en T1 es contado decrecientemente hasta cero una vez, por cada comamdo de carga de T1(R241,D2). Una petición de interrupción es generada cuando la cuenta termina.

Cuando este bit es apagado, T1 opera en modo continuo de cuenta Modulo-n. Al recibir el comamdo de Carga T1, los valores iniciales de T1 son cargados y contados decrecientemente hasta que se llega al fin de la cuenta, esto ocurre todo el tiempo que el bit de habilitación de conteo de T1 (R241,D3) esté prendido. Los valores iniciales pueden ser alterados sin afectar la operación de conteo, al final de la cuenta, los nuevos valores iniciales son cargados en el contad3or para los subsecuentes ciclos de conteo.

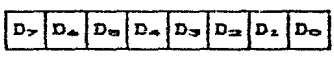
(D1) Selección de la Fuente de Reloj T1. Cuando D1 es prendido, Tin suple el reloj T1. Cuando es apagado, el reloj interno (frecuencia de reloj interno dividida por 4) suple al reloj T1. Cuando Tin es usado pora reloj externo, los modos apropiados deben ser codificados en el registro TMR (R241).

(D2-D7) Valor del Prescalador T1. El resto de los bits en R243 contiene un valor binario de 6-bits que determinan el módulo del prescalador. D2 es el bit menos significativo.

T0 R244 (F4)

REGISTRO DEL CONTADOR TEMPORIZADOR T0

(Lectura/Escritura)



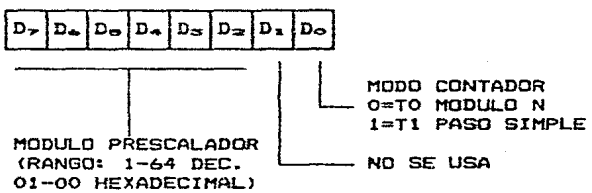
VALOR INICIAL DE T0 (CUANDO SE ESCRIBE)
(RANGO 1-256 DECIMAL 01-00 HEXADECIMAL)
VALOR ACTUAL DE T0 (CUANDO SE LEE)

Este registro provee las mismas funciones para T0 como R242 lo hace para T1.

PREO R245 (F5)

REGISTRO DE CARGA DEL PREESCALADOR DE TO

(Lectura/Escritura)

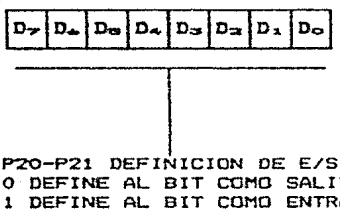


Este registro tiene las mismas funciones como su contraparte (R243), excepto en D1, el cual no tiene función alguna.

P2M R246 (F6)

REGISTRO DE MODOS DEL PUERTO 2

(Solo Escritura)

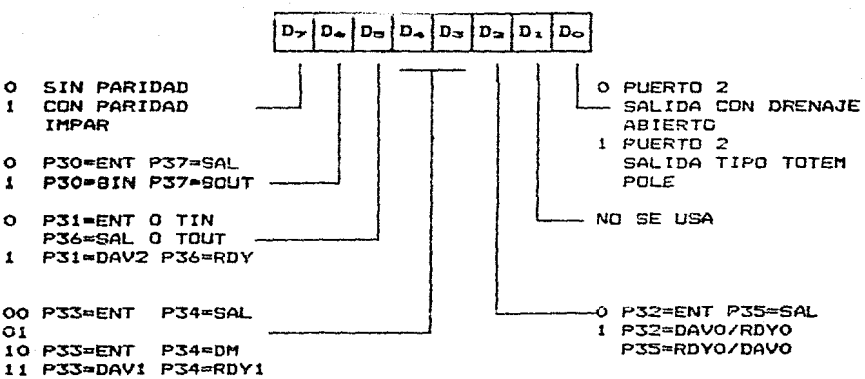


El Registro Modos del Puerto 2 programa cada bit del Puerto 2 como una línea de entrada o de salida. Cuando un bit de R246 es prendido, la línea correspondiente del Puerto 2 es definida como entrada. Cuando es cero, la línea correspondiente es definida como salida. Después de un reset maestro, R246 contiene un FF_{16} y por lo tanto, todas las líneas del Puerto 2 son definidas como entradas. La opción de Drenaje Abierto del Puerto 2 es definido por el bit D0 de R247.

P3M R247 (F7)

REGISTRO DE MODOS DEL PUERTO 3

(Solo Escritura)



El Registro Modos del Puerto 3 especifica cuales lineas del Puerto 3 son usadas para E/S paralela, E/S serie, E/S de temporizadores, Handshake o salidas de STATUS.

(D0) Pullups Puerto 2. Si este bit es cero, indica que las salidas del Puerto 2 trabajan en modo Drenaje Abierto, lo cual permite que estas lineas formen un OR alambrado con otras señales externas.

(D2) Modo P32 y P35. El bit D2 especifica si P32 y P34 son usados para E/S del Puerto 3, o como líneas de enlace del Puerto 0.

D2	Función	
0	P32=Entrada	P35=Salida
1	P32=DAV0/RDY0	P35=RDY0/DAV0

(D3,D4) Modo P33 y P34. Los bits D3 y D4 especifican si P33 y P34 son usados para E/S, o para señales de enlace del Puerto 1, para lo cual son codificados del siguiente manera:

D4	D3	Función	
0	0	P33=Entrada	P34=Salida
0	1		
1	0	P33=Entrada	P34=$\overline{DM}$
1	1	P33=$\overline{DAV1}$/RDY1	P34=RDY1/$\overline{DAV1}$

D5 Modo P31 y P36. El bit D5 determina si las líneas P31 y P36 del Puerto 3 son configuradas como E/S (D5=0), o como señales de enlace (D5=1) del Puerto 2. Cuando P31 y P36 son programados para Entreda/Salida, normalmente reflejan el contenido de los bits 1 y 6 respectivamente, del registro R3. Sin embargo, otras fuentes para estos pins pueden ser seleccionadas como veremos a continuación.

Si el bit D1 del Registro Prescalador T1 (R243) es prendido, P31 se comporta como la entrada de control del Temporizador T1 (Tin) y su modo de operación depende de los bits D4 y D5 del Registro de Modo Temporizador R241 (TMR), de acuerdo a la tabla de verdad mostrada en la sección destinada a este último registro. Para que P31 actúe como línea de entrada del registro R3, el bit D1 de R243 debe ser cero.

D5	Función	
0	P31=Entrada(Tin)	P36=Salida(Tout)
1	P31=$\overline{DAV2}$/RDY2	P36=RDY2/$\overline{DAV2}$

La salida de P36 viene siendo Tout (De T0 o T1), dependiendo de los valores del Registro Modo Temporizador (R241), bits D6 y D7. Para que P36 funcione como un dato de salida de R3, los bits D6 y D7 de R241 deben ser puestos en cero.

Si P31 y P36 son seleccionados como señales de handshake, R241 (D4-D7) y R245 (D1) no tienen efecto.

D6 Modo P30 y P37. D6 determina si las lineas P30 y P37 del Puerto 3 funcionan como E/S o como interfase Receptor/Transmisor Serie.

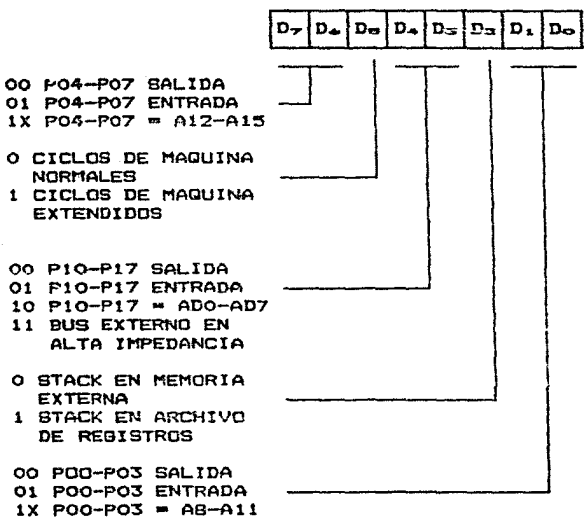
D6	Función	
0	P30=Entrada	P37=Salida
1	P30=Entrada Serie	P37=Salida Serie

D7 Paridad. Si la entrada serie es seleccionada por D6, prendiendo D7 se establece paridad non para transmision de datos y chequeo de paridad non para recepción de datos.

P01M R248 (F8)

REGISTRO DE MODOS DE LOS PUERTOS 0 Y 1

(Solo Escritura)



Este registro configura los puertos 0 y 1, selecciona un stack interno o externo, y selecciona una sincronización de memoria normal o extendida.

(D0,D1) Modo Puerto 0. Estos dos bits codifican el modo de las líneas P00-P03 del Puerto 0. Después de un reset, el nibble inferior del Puerto 0 queda configurado en modo de entrada. Notese que si el bit D7 de R248 es 1, las líneas P00-P03 se configuran como A8-A11, sin importar la configuración de los bits D0 y D1.

D7	D1	D0	P00-P03	Configuración
0	0	0	4-bits	Salida
0	0	1	4-bits	Entrada
0	1	X	4-bits	Dirección (A8-A11)
1	X	X	4-bits	Dirección (A8-A11)

(D2) Localización del Stack. Cuando D2 es prendido, el stack es interno en el archivo de registros y es indicado por el apuntador al stack SPL (R255). Cuando es cero, el stack se localiza en la memoria de datos externa y es señalado por el apuntador al stack SP(254) y (255). Cuando el stack interno es seleccionado, R254 puede ser usado como un registro de datos. Sin embargo, debe tenerse cuidado con el manejo de los bits de acarreo provenientes de R254 ya que pueden modificar a R255.

(D3,D4) Modo Puerto 1. Los bits D3 y D4 codifican el modo de las líneas P10-P17 del Puerto 1. Después de un reset, el Puerto 1 es configurado como un puerto de entrada de 8 bits.

D4	D3	P10-P17	Configuración
0	0	8-bits	Salida
0	1	8-bits	Entrada
1	0	8-bits	Multiplexado Dirección/Datos
1	1		Estado de alta impedancia

El estado de alta impedancia y los modos de direccionamiento seleccionados por R248 son independientes como se muestra en la tabla siguiente:

D7	D4	D3	D1	Lineas 3-Estados
0	1	1	0	Puerto 1, $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$
0	1	1	1	Puerto 1, $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ y P00-P03
1	1	1	X	Puerto 1, $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ y P00-P07

(D5) Temporización de Memoria Extendida. Cuando es prendido este bit, extiende el ciclo de memoria por un período de reloj para permitir un enlace con memorias más lentas. Cuando es apagado la temporización de memoria externa normal es seleccionada.

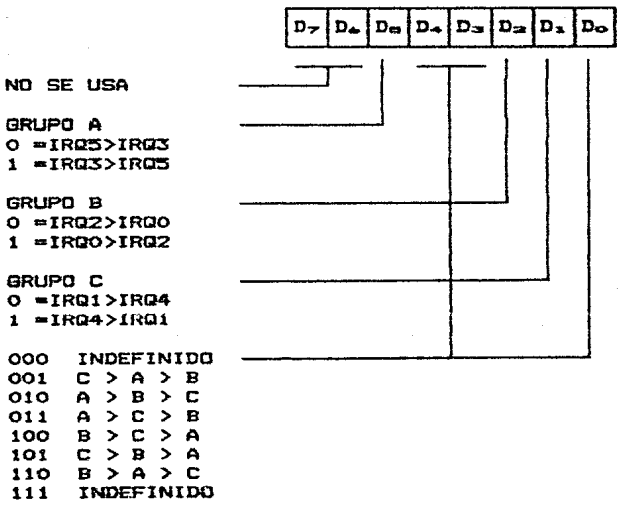
(D6 y D7) Modo Puerto 1. Los bits D6 y D7 controlan las líneas P04-P07 del Puerto 0. Después de un reset, el nibble del Puerto 0 es configurado en el modo de entrada. Cuando D7 es prendido, P00-P03 son A8-A11 sin tomar en cuenta la configuración determinada por D0 y D1.

D7	D6	P04-P07	Configuración
0	0	4-bits	Salida
0	1	4-bits	Entrada
1	X	4-bits	Dirección (A12-A15)

IPR R249 (F9)

REGISTRO DE PRIORIDAD DE INTERRUPCIONES

(Solo Escritura)



Este registro determina el nivel de prioridad de las 6 interrupciones, codificando 48 diferentes ordenes para resolver casos de petición de interrupcion simultáneas. Los 6 niveles de interrupción para IRQ0-IRQ5, son divididos en 3 grupos, arbitrariamente llamados A, B y C, cada uno de los cuales contiene 2 peticiones de interrupción. El grupo de prioridad A contiene: IRQ3(SI/PO3) e IRQ5(T1); el grupo B contiene: IRQ0(P32) e IRQ2(P31), y el grupo C IRQ1(P33) e IRQ4(SO/TO).

La prioridad de una interrupción se determina definiendola primero en forma individual en su grupo correspondiente y después indicando la prioridad de dicho grupo. Los bits D1, D2 y D5 definen la prioridad de los miembros individuales dentro de sus grupos.

GRUPO	BIT	PRIORIDAD MAYOR	PRIORIDAD MENOR
C	D1=0	IRQ1	IRQ4
	1	IRQ4	IRQ1
B	D2=0	IRQ2	IRQ0
	1	IRQ0	IRQ2
A	D5=0	IRQ5	IRQ3
	1	IRQ3	IRQ5

Las prioridades entre los grupos A, B y C, son codificadas por los bits D0, D3 y D4 de de la siguiente manera.

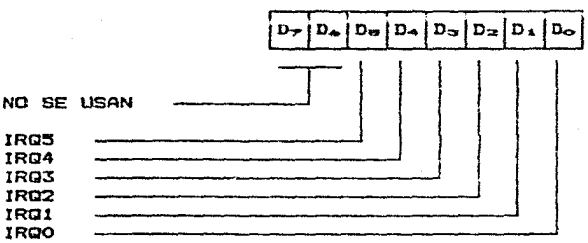
PATRON DE BITS			PRIORIDAD DEL GRUPO DE MAYOR A MENOR
D4	D3	D0	
0	0	0	indefinido
0	0	1	C A B
0	1	0	A B C
0	1	1	A C B
1	0	0	B C A
1	0	1	C B A
1	1	0	B A C
1	1	1	indefinido

Los bits D6 y D7 no se usan.

IRQ R250 (FA)

REGISTRO DE PETICIONES DE INTERRUPCION

(Lectura/Escritura)

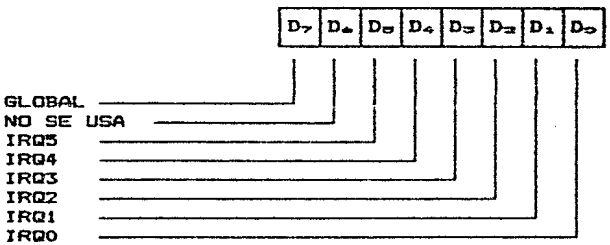


Este registro guarda las peticiones de interrupción. Puesto que éste es un registro de Lectura/Escritura, puede también usarse para muestreo (Polling) de interrupciones. Cuando ocurre una interrupción en alguno de los 6 niveles, un 1 es escrito en la posición del bit correspondiente de este registro. Los bits D0-D5 son asignados a las peticiones de interrupción IRQ0-IRQ5 respectivamente. D6 y D7 no tienen función.

IMR R251 (FB)

REGISTRO DE MASCARAS DE INTERRUPCION

(Lectura/Escritura)



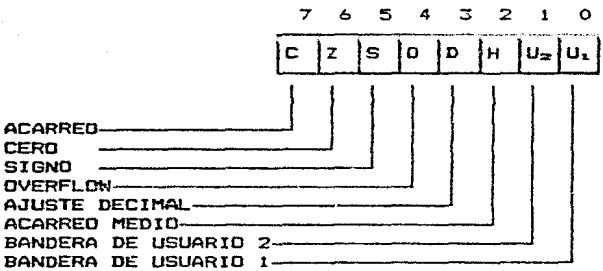
Este registro habilita o deshabilita individual o globalmente las peticiones de las 6 interrupciones. Cuando los bits D0-D5 son prendidos, las peticiones correspondientes son habilitadas. D7 es el habilitador maestro y debe prenderse después de que alguna de las interrupciones individuales sea reconocida. Apagando D7, se deshabilitan todos las peticiones de interrupción. D7 puede ser prendido o apagado por las instrucciones de habilitación de interrupciones (EI) e inhibición de interrupciones (DI). Se apaga automáticamente durante un ciclo de máquina de interrupción y es prendido después de la ejecución de la instrucción de retorno de interrupción (IRET).

El contenido de este registro puede modificarse solamente después de una instrucción DI.

FLAG R252 (FC)

REGISTRO DE BANDERAS

(Lectura/Escritura)



Este registro contiene las banderas del Z8. El significado asignado a cada bit se explica a continuación:

(D7) Bandera de Acarreo (Carry). Reporta un acarreo como resultado de operaciones aritmèticas. Puede ser alterado tambièn, por algunas operaciones de rotación y corrimiento.

(D6) Bandera de Cero. Su valor es uno cuando el resultado de una operación aritmètica o lógica es cero. El valor de esta es cero en cualquier otro caso.

(D5) Bandera de Signo. Su valor es igual al bit de mayor orden del resultado de una operación aritmètica o lógica.

(D4) Bandera de Overflow. Reporta acarreos de los bits de magnitud en operaciones aritmèticas con magnitud signada.

(D2) Bandera de Acarreo Medio (Half Carry). Representa un acarreo del bit 3 al bit 4, despuès de una operción aritmètica o lógica.

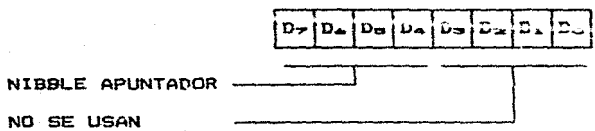
(D3) Bandera de Ajuste Decimal. Esta bandera es usada por el Z8 en una manera algo distinta a otros microprocesadores. Las operaciones que el microprocesador debe ejecutar para un ajuste decimal dependen de si la última operación aritmètica fuè una resta o una suma binarias. Si la última operación aritmètica fuè una resta binaria, esta bandera toma un valor de 1 y el valor de cero si fuè una adición binaria. La instrucción de ajuste decimal, posteriormente, interroga el estado de esta bandera para ejecutar correctamente la operación.

(D1, D0) Banderas de Usuario. El usuario tiene posibilidad de alterar a voluntad estas banderas.

RP R253 (FD)

REGISTRO APUNTADOR

(Lectura/Escritura)



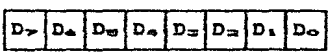
Los 4 bits más significativos de este registro forman un apuntador a registro, los cuales apuntan al grupo de registros de trabajo en curso. Los 4 bits menos significativos (designador de registro) que especifican el registro individual del grupo de registros de trabajo, son indicados por la instrucción. En este registro, los 4 bits de orden bajo (D0-D3) son siempre cero cuando se lee y no tienen importancia cuando se escribe.

SPL SPH R254-5 (FE-FF)

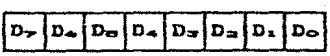
REGISTRO APUNTADORES AL STACK

(Lectura/Escritura)

SPH (FE)



SPL (FF)



El apuntador al stack está compuesto de dos registros de 8 bits: SPL(254) contiene el byte de dirección baja; SPH(255) contiene el byte alto.

El bit D2 de R248 especifica en que parte del stack es localizado en el archivo de registros o en memoria externa. Si el Stack es interno, R254 no se usa como apuntador al stack y es disponible como un registro de datos. Siempre y cuando el manejo de bits de acarreo hacia R255 sea manejado adecuadamente.

Finalmente en la tabla siguiente se muestra un resumen del estado de los registros de control inmediatamente después de un reset maestro.

NUMERO	NOMBRE DEL REGISTRO	CONTENIDO							
		7	6	5	4	3	2	1	0
FF	APUNTADOR DE STACK, BYTE DE MAYOR ORDEN	X	X	X	X	X	X	X	X
FE	APUNTADOR DE STACK, BYTE DE MENOR ORDEN	X	X	X	X	X	X	X	X
FD	APUNTADOR DE REGISTROS	X	X	X	X	0	0	0	0
FC	BANDERAS	X	X	X	X	X	X	X	X
FB	MASCARA DE INTERRUPCIONES. INHIBICION TOTAL.	0	X	X	X	X	X	X	X
FA	PETICIONES DE INTERRUPCION. NINGUNA PENDIENTE	X	X	0	0	0	0	0	0
F9	PRIORIDAD DE INTERRUPCIONES	X	X	X	X	X	X	X	X
F8	PUERTOS 0/1. ENTRADAS. STACK INTERNO, CICLO NORMAL	0	1	0	0	1	1	0	1
F7	PUERTOS 2/3. DRENAJE ABIERTO EN 2, 3 COMO E/S SIMPLE	0	0	0	0	0	0	X	0
F6	PUERTO 2. TODAS LAS LINEAS COMO ENTRADAS	1	1	1	1	1	1	1	1
F5	PRESCALADOR T0. MODO UNA PASADA, RELOJ INTERNO	X	X	X	X	X	X	X	0
F4	TEMPORIZADOR T0	X	X	X	X	X	X	X	X
F3	PRESCALADOR T1. MODO UNA PASADA, RELOJ INTERNO	X	X	X	X	X	X	0	0
F2	TEMPORIZADOR T1	X	X	X	X	X	X	X	X
F1	CONTROL DE TEMPORIZADORES. DETENIDOS AMBOS	0	0	0	0	0	0	0	0
F0	E/S SERIAL	X	X	X	X	X	X	X	X

CAPITULO III

PROGRAMACION DEL Z8

En el capitulo anterior se habló detalladamente de la arquitectura de la microcomputadora de un solo chip, Z8. Se hizo un estudio de los bloques funcionales de los que dispone el usuario del Z8, así como de otros conceptos relacionados con el funcionamiento de este microcomputador.

Se habló de componentes del Z8 como son:

- Buses de Direcciones.
- Puertos de Entrada/Salida.
- Contadores.
- Temporizadores.
- Osciladores.

Así como de conceptos relacionados con su funcionamiento, como son:

- Señales de Control.
- Espacio de Direccionamiento
  a) Memoria de Programa.
  b) Memoria de Datos.
  c) Archivo de Registros.
- Registros de Control etc.

En este capitulo se estudiará en forma detallada la programación del Z8. Desde este punto de vista se explicará lo que es la memoria de programa y la memoria de datos, anotando la diferencia entre una y otra, y la forma en que son utilizadas para la ejecución de programas desarrollados en el ensamblador del Z8.

INTRODUCCION

En todo procesador, una de las partes más importantes de su estudio son las diferentes formas en que éste maneja la información, es decir, las reglas para localizar los operandos de una instrucción. En esta parte del capitulo se realizará una descripción completa de los modos de direcionamiento del Z8, de esta forma, se tendrán los antecedentes para describir el conjunto de instrucciones en ensamblador con que cuenta el Z8.

Es claro que para que un procesador realice una función dada, debe de contar con un conjunto de instrucciones para poder ejecutarla, las cuales se combinan con los modos de direccionamiento. Para resumir estos últimos, y ver como afectan al conjunto de instrucciones, se presentarán dos tablas que describen estos dos conceptos de la programación del Z8.

En el presente capitulo se describe el conjunto de instrucciones del Z8. También se hará una comparación de los elementos de programación del Z8 (como por ejemplo: instrucciones de tipo RESTART), en relación a otros procesadores, analizando el porqué de estas diferencias.

Finalmente para reforzar los conocimientos relacionados con este capitulo, se dispondrá de un ápendice destinado a programas ensambladores, en donde se explicarán el manejo de èstos, los diferentes tipos que existen, etc.

MEMORIA DE DATOS Y MEMORIA DE PROGRAMA

Para comprender la manera de programar el Z8, es conveniente repasar algunos conceptos ya estudiados en el capitulo 2, y que nos serán utiles en el presente.

Una de las caracteristicas principales del Z8, es la forma eficiente en que maneja su memoria, ya que ésta puede ser utilizada en formas diferentes dependiendo de las necesidades especificas que se quieran resolver.

El Z8 puede ser configurado como una microcomputadora con 2K de memoria ROM y con capacidad de manejo de hasta 124K de memoria externa, además, cuenta con un Archivo de 144 Registros internos de lectura y escritura.

El espacio direccionable del Z8 puede clasificarse de la siguiente manera:

MEMORIA INTERNA	MEMORIA EXTERNA
PROGRAMA	PROGRAMA DATOS

- MEMORIA DE PROGRAMA.

El Z8 posee un contador de programa (PROGRAM COUNTER) de 16 bits, por lo que puede direccionar 65,535 bytes de memoria. Este espacio de memoria puede ser dividido en dos áreas llamadas MEMORIA INTERNA Y MEMORIA EXTERNA.

La memoria interna está formada por 2048 bytes de ROM. Si fuera necesario mayor capacidad, el Z8 debe realizar ciclos de lectura a memoria de programa externa, lo cuál es posible si ésta se expande externamente con hasta 63408 bytes y el Z8 es configurado correctamente.

La memoria externa de programa puede considerarse como una continuación lógica de la memoria interna, pues las instrucciones que hacen referencia a este tipo de memoria, no distinguen entre una y otra.

- MEMORIA DE DATOS

El Z8 tiene la capacidad de accesar hasta 62 Kbytes de memoria externa de datos, la cuál comienza en la localidad 2048, como se aprecia en la Figura (3.1):

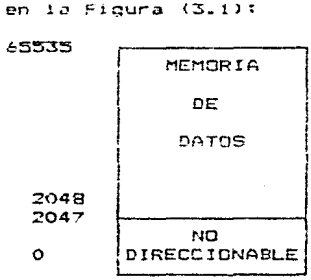


FIGURA (3.1)

La memoria de datos puede o no ser incluida en el espacio destinado a la memoria de programa externa.

Hasta aqui, parece no existir diferencia entre la memoria de programa y la de datos.

El Z8 asume que la memoria de datos externa es exclusiva para almacenamiento de datos, y por lo tanto, el código de operación no puede ser accesado desde ella. La memoria de datos solo puede accesarse con un número limitado de instrucciones que expresamente hacen referencia a ella.

De hecho, cuando se quiere separar el espacio de memoria de datos se tiene que habilitar la señal DM , programando adecuadamente el registro P3M.

En el caso de que se separe el espacio de memoria de datos activando DM, existe una diferencia en su manejo, ya que solo un número limitado de instrucciones ejecutan movimientos de datos entre registros internos y la memoria de datos externa.

Durante el acceso a estos 2 tipos de memoria (de programa y de datos), la memoria de programa externa y la de datos pueden c no compartir el mismo espacio de direcciones. Cuando lo comparten, todas las instrucciones que accesan memoria de datos externa accesarán localidades de memoria de programa.

ARCHIVO DE REGISTROS DEL Z8

Una de las ventajas que tiene el Z8 es el número de registros internos con que cuenta y la versatilidad con que pueden ser manejados.

Cualquier microprocesador tiene un conjunto de registros de un tamaño especifico de bits, los cuales son utilizados para almacenar información para ser utilizada en el control de algún proceso.

Por ejemplo, en el caso del microprocesador Z80 de ZILOG, éste tiene 14 registros de próposito general de 8 bits y 5 registros de próposito especifico, como se aprecia en la Figura (3.2).

De los registros de próposito general, solo el registro A puede usarse como acumulador, es decir, un registro especial cuya función es la de almacenar el resultado de una o varias operaciones.

El Z8 cuenta con un conjunto de registros que pueden ser utilizados por el usuario como registros de control o de próposito general, para el almacenamiento de información.

A	BANDERAS	A'	BANDERAS
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'
I	R		
IX			
IY			
SP			

REGISTROS DE PROPOSITO GENERAL

REGISTROS DE PROPOSITO ESPECIFICO

FIGURA (3.2)

Estos registros son agrupados en un espacio interno llamado ARCHIVO DE REGISTROS, el cuál cuenta con un total de 144 registros de 8 bits cada uno, clasificados de la siguiente manera:

- 4 registros utilizados como puertos de E/S (R0 - R3).
- 124 registros de proposito general (R4 - R127).
- 16 registros de control y estado.

En la Figura (3.3) se muestra como está organizado el Archivo de Registros.

A excepción de los registros de control, cualquiera de los restantes puede ser utilizado como acumulador, o simplemente para el almacenamiento de datos.

Para un manejo dinámico de los registros, se dividen en 9 grupos de 16 registros cada uno, y cuyas direcciones son mostradas en la Figura (3.4).

DIAGRAMA DEL ARCHIVO DE REGISTROS

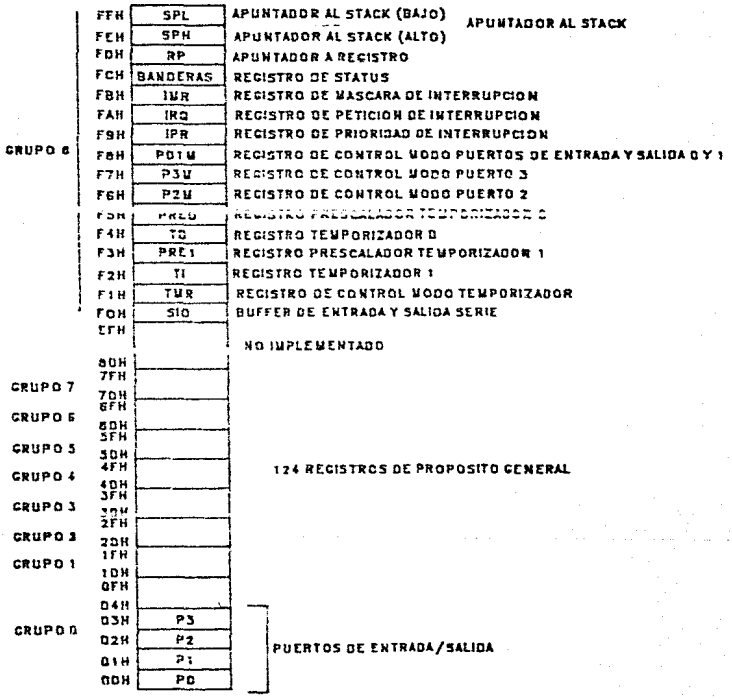


FIGURA (3.3)

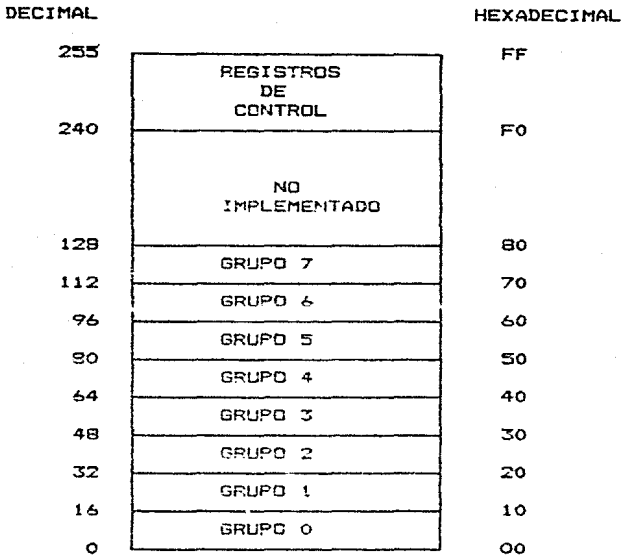


FIGURA (3.4)

MODOS DE DIRECCIONAMIENTO DEL Z8

Un operando es el dato sobre el cuál la instrucción se ejecuta. La combinación de las instrucciones y la forma como se indican sus operandos, es lo que se conoce como modos de direccionamiento.

Un operando puede ser indicado de varias maneras, por ejemplo, como un dato inmediato o un dato que se encuentra contenido en un registro.

Existen varias formas de direccionamiento del Z8 y pueden clasificarse en dos grupos:

- MODOS DE DIRECCIONAMIENTO A REGISTROS

FORMA INMEDIATA.
FORMA DIRECTA.
FORMA INDIRECTA.
FORMA INDEXADA.

- MODOS DE DIRECCIONAMIENTO A LOCALIDADES DE MEMORIA

FORMA DIRECTA.
FORMA INDIRECTA.

NOTACION

Los operandos y las banderas de status son representados por una notación que es utilizada al momento de describir cada una de las instrucciones del Z8, así como sus modos de direccionamiento.

Es importante señalar que esta notación es compatible con la que será utilizada al explicar el conjunto de instrucciones del lenguaje ensamblador del Z8. La notación para operandos, códigos de condición y modos de dirección, se muestra en la siguiente página.

Para especificar un modo de direccionamiento en particular, son utilizados en los operandos ciertos caracteres especiales, los cuales son :

" R "	precede a un número de registro de trabajo.
" RR "	precede a un número de registro par.
" @ "	precede a un número de registro para direccionamiento indirecto.
" # "	precede a un dato inmediato.
" ( ) "	contiene la parte del registro índice, cuando se utiliza direccionamiento indexado.
" $ "	en el modo de direccionamiento relativo es utilizado para representar la localidad actual de memoria en que se localiza el contador de programa (program counter).

Una manera apropiada de comenzar a describir los modos de direccionamiento del Z8, es estableciendo que de los registros descritos en la Figura (3.3), ninguno de los de propósito general tiene un modo de direccionamiento exclusivo o preferente, ya que

NOTACION	MODO DE DIRECCION	RANGO DEL OPERANDO
cc	Código de Condición	Códigos de condición
r	Registro de Trabajo	Rn donde n 0 - 15
R	Registro o Registro de Trabajo	reg = 0 -127,240-255 o Rn definido arriba.
RR	Registro Par o Registro de Trabajo Par	reg , donde reg es un número par en el rango arriba descrito o una variable con dirección par o bien RRp donde p = 0,2,4..14.
Ir	Registro de Trabajo Par	@Rn donde n = 0 - 15.
IR	Registro Indirecto o Registro de Trabajo	@reg , donde reg es definido antes o @Rn definido antes .
Irr	Registro Indirecto de Trabajo	@RRp , donde p = 0,2, 4,6....14.
IRR	Registro Indirecto Par o Registro de Trabajo Par	@reg donde reg es un número par en el rango definido antes, o una variable cuya dirección sea par o @RRp definida antes.
X	Indexado	reg(Rn), donde reg y Rn son definidos antes.
DA	Acceso Directo	Etiqueta del programa o expresión.
RA	Dirección Relativa	Etiqueta del programa o un desplazamiento, donde la localidad designada debe estar en elrango +127 128 bytes desde el inicio de la siguiente instrucción.
IM	Inmediato	#dato , donde dato es una expresion.

cada registro puede ser utilizadc como un acumulador, éstos pueden ser accesados con la misma facilidad, independientemente si se utilizan como fuente, destino o en forma indexada en una localidad de memoria.

MODO DE DIRECCIONAMIENTO A REGISTROS.

- FORMA DIRECTA:

Dentro de este tipo de direccionamiento, se pueden emplear dos formas diferentes que serán descritas a continuación:

1.- Mediante un código objeto que hace referencia a uno de los 144 registros del Z8 como se muestra en la Figura (3.5).

En esta forma directa, una instrucción dada hace referencia a un dato que se encuentra almacenado en un registro de proposito general.

2.- La segunda forma de utilizar direccionamiento a registros internos de manera directa, es estableciendo un código objeto de 4 bits a través del Apuntador a Registros (REGISTER POINTER), para seleccionar cualquiera de los registros de un grupo que es seleccionado con el número hexadecimal contenido en este registro. Hay que notar que el mismo apuntador a registros es uno de los 144 con que cuenta el Z8, en particular pertenece al grupo de registros de control.

Los registros direccionados de esta manera, se llaman **Registros de Trabajo.**

La forma en que las instrucciones utilizan este tipo de direccionamiento mediante un registro de trabajo se puede observar en la figura 3.6

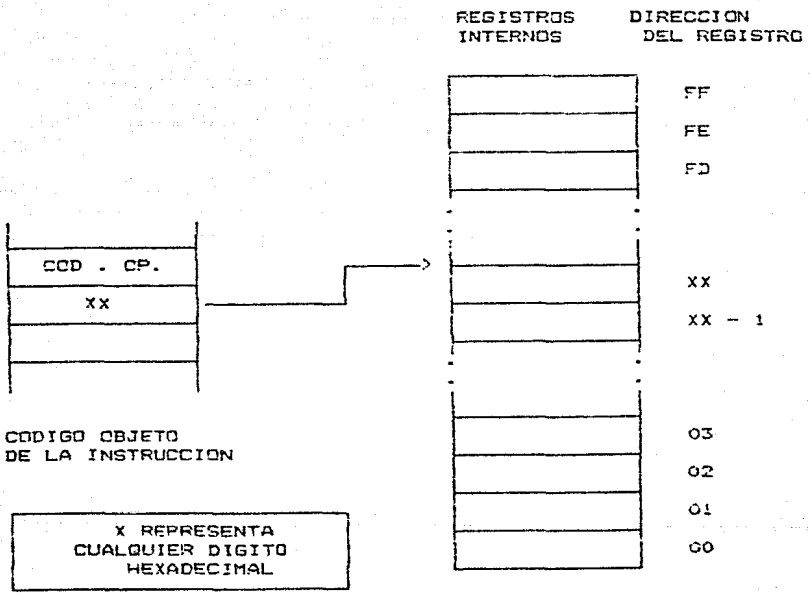

REGISTROS
INTERNOS
DIRECCION
DEL REGISTRO
FF
FE
FD
COD . OP.
XX
XX
XX - 1
03
02
01
00
CODIGO OBJETO
DE LA INSTRUCCION
X REPRESENTA
CUALQUIER DIGITO
HEXADECIMAL

FIGURA (3.5)

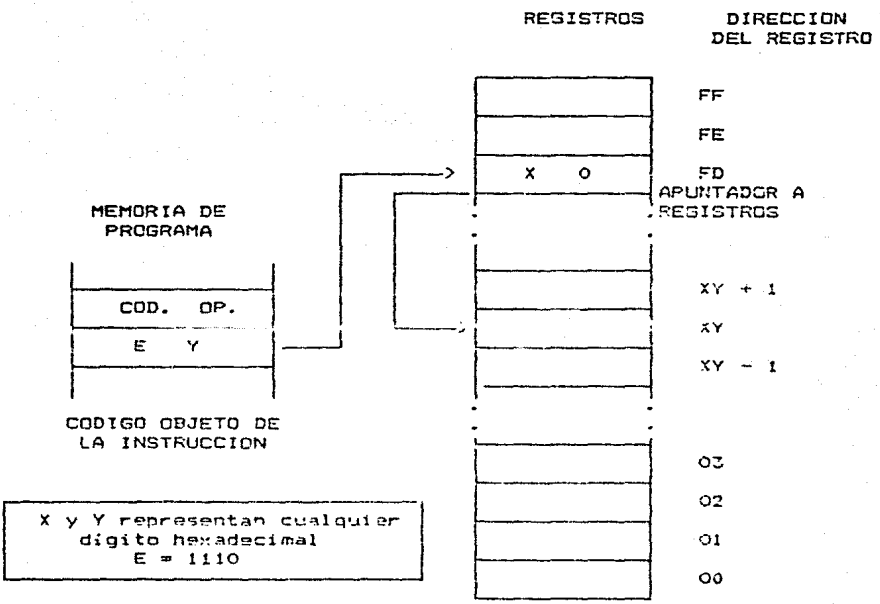


FIGURA (3.6)

Este tipo de direccionamiento en realidad no hace referencia a un número de registro en forma directa, sino que el registro a utilizarse dentro de una instrucción, es seleccionado concatenando el valor contenido en el Apuntador a Registros, con los 4 bits proporcionados por el código objeto. El RP proporciona los 4 bits mas significativos, y aunque puede grabarse cualquier valor hexadecimal en él, los 4 bits bajos no se usan y son forzados a valer 0.

Con este tipo de direccionamiento, se pueden direccionar hasta 256 registros, aunque solo existen 144. Hay que destacar el hecho de que los registros con direcciones desde 20_{16} hasta la EF_{16} no están implementados en el Z8.

Utilizando esta forma de direccionamiento, el manejo de registros se hace más dinámico, ya que con solo cambiar el valor del RP podemos contutar varios registros totalmente diferentes y, con esto, agilizar el proceso de programación, la rapidez de ejecución de una aplicación del Z8, además de que en esta forma de direccionamiento se ahorran bytes en el ensamblado del programa.

- DIRECCIONAMIENTO INDEXADO

En las instrucciones de carga puede utilizarse el direccionamiento indexado. Este direccionamiento consiste en seleccionar un registro que puede ser cualquiera de los 16 de un grupo designado de trabajo, el cual contendrá una cantidad hexadecimal que sumada al código objeto proporcionado por la instruccion, dara como resultado la dirección del registro que se quiere accesar.

Este tipo de direccionamiento es útil en programas en los que los registros deban ser accesados dinámicamente dentro del mismo programa, como podria ser la transferencia de bloques de memoria, en la cual el registro destino y el fuente varian constantemente.

En la Figura (3.7) se muestra esquemáticamente este modo de direccionamiento.

Este modo utilizado exclusivamente por las instrucciones de carga (LD).

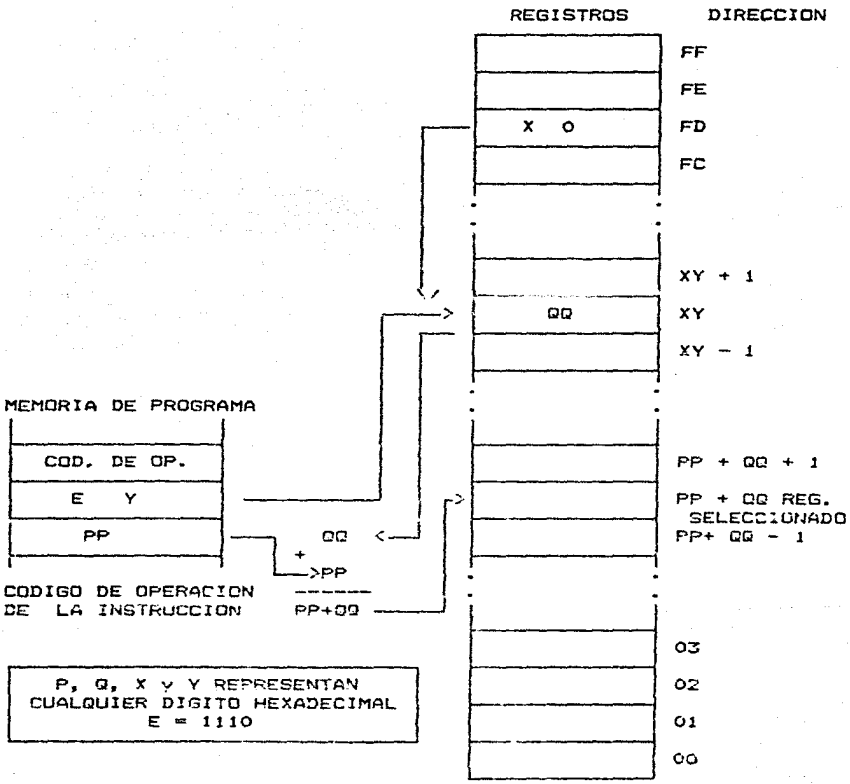

REGISTROS
DIRECCION
FF
FE
X O
FD
FC
XY + 1
QQ
XY
XY - 1
MEMORIA DE PROGRAMA
COD. DE OP.
E Y
PP
QQ
+
PP
PP + QQ + 1
PP + QQ REG.
SELECCIONADO
PP+ QQ - 1
CODIGO DE OPERACION
DE LA INSTRUCCION
PP+QQ
03
02
01
00
P, Q, X y Y REPRESENTAN
CUALQUIER DIGITO HEXADECIMAL
E = 1110

FIGURA (3.7)

- DIRECCIONAMIENTO INDIRECTO

Los registros del Z8, también pueden direccionarse indirectamente. De esta forma, la información no está contenida directamente en el registro accesado, sino que éste almacena la verdadera dirección del registro que debe operarse. Existen 2 variantes de este modo de direccionamiento. Uno a través de un registro cualquiera y la otra mediante el apuntador de registros y un registro de trabajo. En la siguiente figura se muestra la operación realizada vía un registro cualquiera:

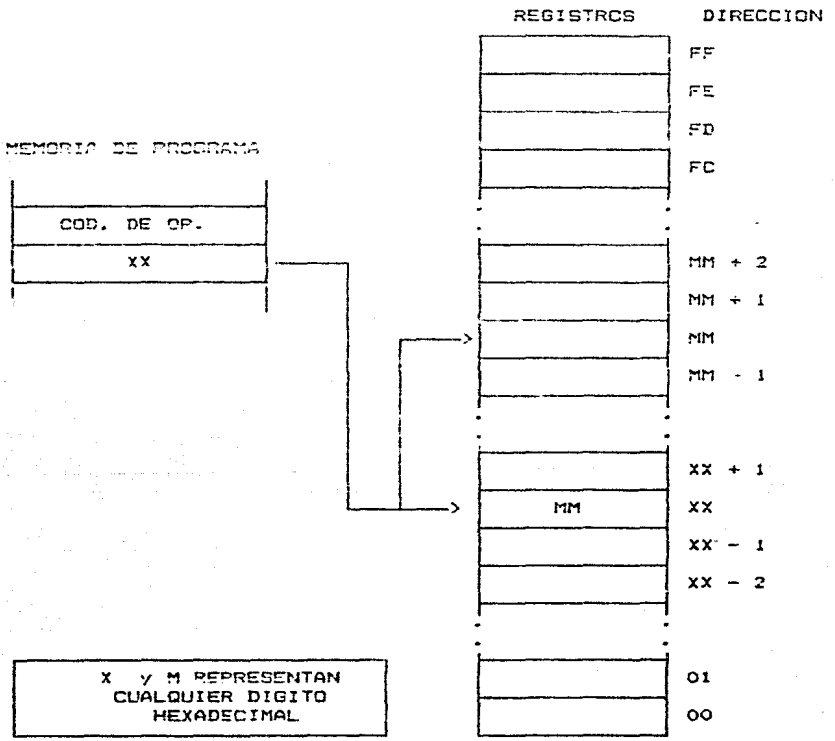


FIGURA (3.8)

Este mismo tipo de direccionamiento utilizando un registro de trabajo, para almacenar la dirección verdadera puede observarse en la figura 3.9:

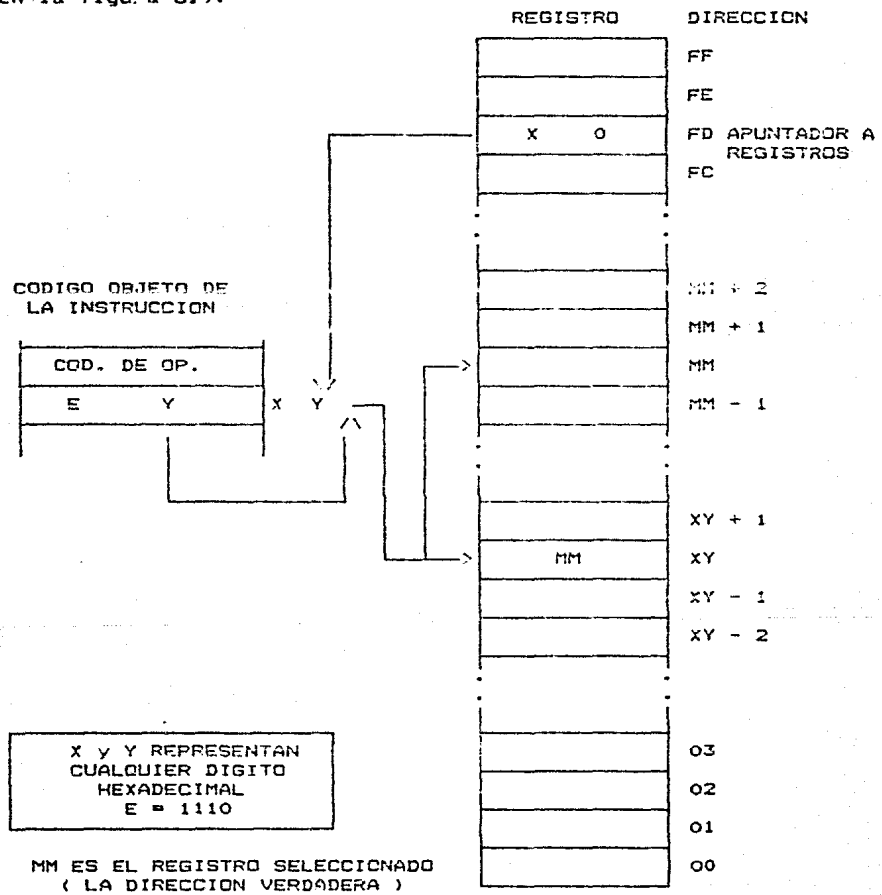


FIGURA (3.9)

MODOS DE DIRECCIONAMIENTO A LOCALIDADES DE MEMORIA

Toda localidad de memoria es accesada indirectamente vía un registro par de trabajo. Este último consiste de dos registros de trabajo adyacentes que contienen una palabra de 16 bits. El registro par toma su nombre del registro con dirección par el cual debe contener el byte de mayor orden de la palabra o dirección de 16 bits, mientras que el siguiente registro, cuya dirección es non, contiene el byte de menor orden como se muestra en la figura 3.10:

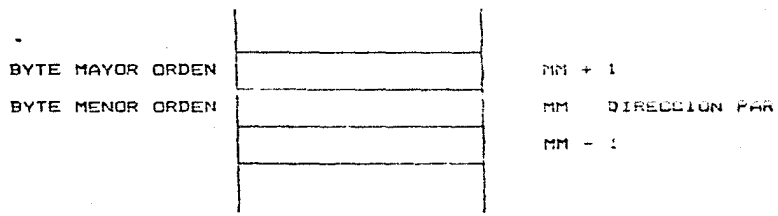


FIGURA (3.10)

La figura 3.11 muestra la forma en que se accesa indirectamente una localidad de memoria.

La dirección indirecta es almacenada en un par de registros ya que para accesar cualquier localidad de memoria se debe tener una dirección de 16 bits. En la figura anterior es representada por PPQQ. La dirección contenida en el código objeto de la instrucción debe ser par y es representada por XY. El registro XY debe contener el byte de mayor orden de la palabra PPQQ.

Si el valor de PPQQ es $07FF_{16}$ o menor, entonces una de las localidades de memoria de programa interna será seleccionada.

Si el valor de PPQQ es mayor a 0800_{16} entonces una localidad de memoria externa será accesada. Si la memoria de datos comparte el mismo espacio que la de programa, entonces PPQQ solo puede referirse a una localidad, pero si la memoria de datos y la de programa se han separado entonces PPQ puede accesar localidades distintas. Para distinguir entre ambas localidades la señal de control DM es baja durante la ejecución de instrucciones que especificamente accesan memoria de datos externa y se mantiene alta durante la ejecución de cualquiera otras instrucciones.

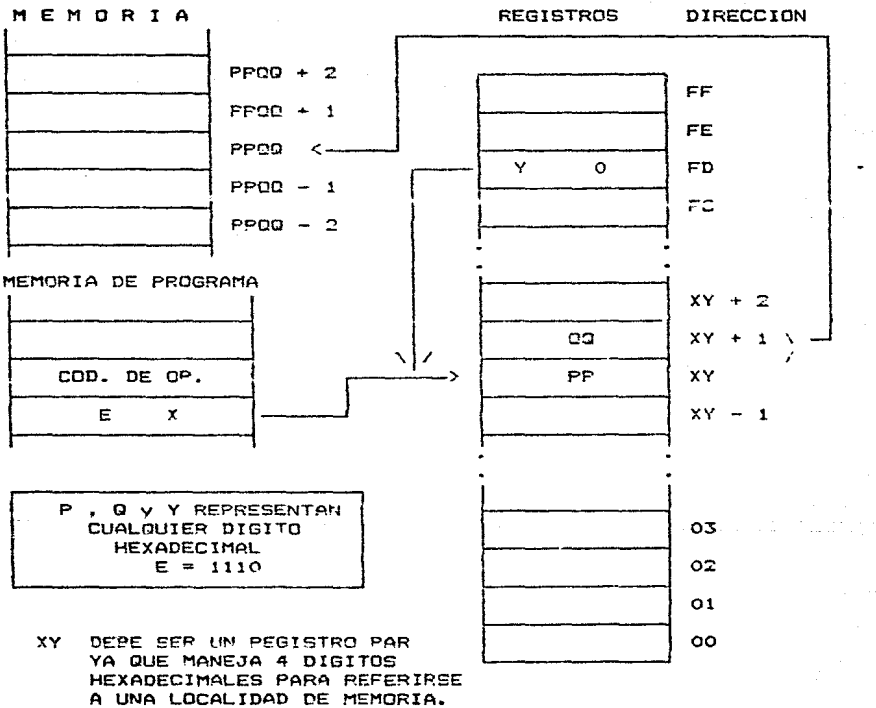

M E M O R I A
REGISTROS
DIRECCION
PPQQ + 2
PPQQ + 1
PPQQ <
PPQQ - 1
PPQQ - 2
Y 0
FF
FE
FD
FC
MEMORIA DE PROGRAMA
XY + 2
QQ
XY + 1
COD. DE OP.
PP
XY
E X
XY - 1
P , Q y Y REPRESENTAN
CUALQUIER DIGITO
HEXADECIMAL
E = 1110
03
02
01
00
XY DEBE SER UN REGISTRO PAR
YA QUE MANEJA 4 DIGITOS
HEXADECIMALES PARA REFERIRSE
A UNA LOCALIDAD DE MEMORIA.

FIGURA (3.11)

CONJUNTO DE INSTRUCCIONES DEL Z8

Para simplificar el estudio de instrucciones del Z8 se agrupan en alguna de las 8 clasificaciones siguientes:

1.- OPERACIONES DE CARGA.

2.- OPERACIONES ARITMETICAS.

3.- OPERACIONES LOGICAS.

4.- OPERACIONES DE CONTROL DE PROGRAMA .

5.- OPERACIONES DE MANIPULACION DE BITS .

6.- OPERACIONES DE TRANSFERENCIA DE BLOQUES DE INFORMACION.

7.- OPERACIONES DE ROTACION Y DESPLAZAMIENTO DE BITS.

8.- OPERACIONES SOBRE EL CONTROL DEL CPU.

Cada uno de estos grupos será descrito de manera general, haciendo énfasis en los detalles y características importantes de cada uno de ellos.

Adicionalmente para tener una referencia más precisa de cada instrucción se muestran dos tablas, que proporcionan los códigos de operación y modos de direccionamiento que puede utilizarse en cada instrucción.

Después de la ejecución de una instrucción, el registro de estado FC_H es modificado, puesto que cada uno de sus bits representa un estado que es alterado. En las tablas de resumen del conjunto de instrucciones del Z8 se presenta una columna donde se indican las banderas que pueden ser alteradas por el resultado de una operación.

Algunas de las instrucciones del ensamblador necesitan de expresiones que evalúen una condición y dependiendo del resultado de ésta, continuar con la ejecución del programa de una forma alterna. A éstas instrucciones las llamaremos del tipo condicionadas.

El registro de status contiene las siguientes banderas:

C	ACARREO
Z	CERO
S	SIGNO
V	DESBORDAMIENTO (OVERFLOW)
D	AJUSTE DECIMAL
H	MEDIO ACARREO

Para ejecutar las instrucciones condicionadas, los diseñadores del Z8 asumieron que solo podían haber 16 distintas condiciones y les asignaron un número a cada una de ellas. Cada condición agrupa un conjunto particular de estados del registro de banderas FC_{1-4}.

Los códigos de condición y asignación de estados para las BANDERAS, son los mostrados en la tabla de la siguiente página.

La notación utilizada para la explicación del conjunto de instrucciones es la siguiente:

NOMBRE DE LA INSTRUCCION — NUMERO DE OPERANDOS UTILIZADOS

En la descripción de operandos se utilizarán dos términos que son :

Operando fuente : src
Operando destino : dst

El operando fuente es generalmente un registro donde se encuentra almacenado el dato a utilizar en la instrucción, y el operando destino es donde se almacenará el resultado de la operación efectuada. También se incluye el código de condición (cc), que consiste en una prueba de selección para indicarle a la instrucción que operación especifica realice.

TABLA DE NEMONICOS , CODIGOS OBJETO E
INTERPRETACION DE LOS CODIGOS DE CONDICION

CODIGO DE CONDICION	MEMONICO	CODIGO OBJETO	ESTADO	COMENTARIOS
0		0000	NINGUNO	NUNCA VERDADERO
1	LT	0001	S + O	MENOR QUE
2	LE	0010	Z+(S + O)=1	MENOR QUE O IGUAL
3	ULE	0011	C+Z=1	MENOR QUE SIN SIG-NO
4	OV	0100	O=1	OVERFLOW
5	MI	0101	S=1	MENOS
6	Z	0110	Z=1	CERO
7	ULT	0111	C=1	MENOR QUE SIN SIG-NO
8		1000	NINGUNO	SIEMPRE VERDADERO
9	GE	1001	S + O = O	MAYOR QUE O IGUAL
10	GT	1010	Z+(S + O) = O	MAYOR QUE
11	UGT	1011	C + Z = O	MAYOR QUE SIN SIG-NO
12	NOV	1100	O = O	NO OVERFLOW
13	PL	11C1	S = O	POSITIVO
14	NZ	1110	Z = O	NO CERO
15	NC	1111	C = O	SIN ACARREO

TABLA 1

INSTRUCCIONES DE CARGA

Las instrucciones de carga son las encargadas de mover datos de un sitio a otro del Z8, como puede ser el movimiento de datos entre registros, entre memoria de datos o de programa a registros.

Se incluye también la instrucción de ajuste Decimal (DA) que permite operaciones de aritmética decimal codificada en binario que puede emplearse tanto para la suma como para la resta. gracias al bit de acarreo medio del registro de Status.

En forma general las instrucciones de CARGA son las siguientes:

INSTRUCCION	OPERANDO(S)	NOMBRE
CLR	dst	LIMPIA REGISTRO
LD	dst,src	CARGA DATO
LDC	dst,src	CARGA DATO DE MEMORIA DE PROGRAMA EXTERNA
LDE	dst,src	CARGA DATOS DE MEMORIA DE DATOS
POP	dst	REALIZA LA FUNCION POP
PUSH	src	REALIZA LA FUNCION PUSH

INSTRUCCIONES ARITMETICAS

Este tipo de instrucciones son para realizar operaciones de tipo aritmético, las cuáles son necesarias en la mayoría de las aplicaciones de un microprocesador.

Estas instrucciones son la siguientes:

INSTRUCCION	OPERANDO(S)	N O M B R E
ADC	dst,src	SUMA CON ACARREO
ADD	dst,src	SUMA
CP	dst,src	COMPARA
DA	dst	AJUSTE DECIMAL
DEC	dst	DECREMENTA UN REGISTRO DE TRABAJO
DECW	dst	DECREMENTA UN REGISTRO PAR O PALABRA
INC	dst	INCREMENTA UN REGISTRO DE TRABAJO
INCW	dst	INCREMENTA UN REGISTRO PAR O PALABRA
SBC	dst,src	RESTA CON ACARREO
SUB	dst,src	RESTA

INSTRUCCIONES LOGICAS

Las instrucciones de tipo lógico son utilizadas para pruebas de ciertos estados de uno ó dos registros. También permiten cambiar un registro de un estado a otro estado.

Estas instrucciones son las siguientes:

INSTRUCCION	OPERANDO(S)	N O M B R E
AND	dst,src	AND LOGICO
COM	dst	COMPLEMENTO A 2'S
OR	dst,src	OR LOGICO
XOR	dst,src	OR EXCLUSIVO LOGICO

INSTRUCCIONES DE CONTROL DE PROGRAMA

Para poder realizar ciertas funciones de programación, el Z8 cuenta con instrucciones muy importantes, ya que permiten alterar la ejecución normal de un programa, mediante el uso de saltos y bifurcaciones, así como de la realización de procesos repetitivos en base a subrutinas.

Los códigos de condición toman especial importancia para el uso de estas instrucciones. La condición a probarse depende del nibble indicado por (cc) cuyo valor indica la prueba según la tabla 1 de este mismo capítulo.

La interpretación de estas instrucciones es la siguiente :

INSTRUCCION	OPERANDO(S)	N O M B R E
CALL	dst	LLAMADA A UNA SUBRUTINA
DJNZ	dst	DECREMENTA Y SALTA SI LA CONDICION NO ES CERO
IRET		REGRESO DE UNA INTERRUPCION
JP	cc,dst	SALTO CONDICIONAL
JR	cc,dst	SALTO RELATIVO
RET		REGRESO DE UNA SUBRUTINA

INSTRUCCIONES DE CONTROL DEL CPU

Durante la ejecución de un proceso dentro de un microprocesador, generalmente es necesario variar las condiciones en que se encuentra el CPU (Unidad Central de Procesamiento), a fin de que éste realice otras funciones diferentes a las que estaba realizando al momento de alterar su estado. También para el manejo de INTERRUPCIONES es necesario controlar el estado del CPU.

Las instrucciones de Control del CPU son:

INSTRUCCION	OPERANDO(S)	N O M B R E
CCF		COMPLEMENTO DEL BIT DE ACARREO
DI		DESHABILITA INTERRUPCIONES
EI		HABILITA INTERRUPCIONES
NOP		NO OPERACION
RCF		PONE LA BANDERA DE ACCARREO EN CERO
SCF		PONE LA BANDERA DE ACARREO EN UNO
SRP	dst	GUARDA EL VALOR SELECCIONADO EN EL APUNTADOR A REGISTROS.

INSTRUCCIONES DE ROTACION Y DESPLAZAMIENTO

Las operaciones de desplazamiento y rotación permiten manipular bits a la izquierda o a la derecha sobre los registros del Z8.

Estas instrucciones son las siguientes:

INSTRUCCION	OPERANDO(S)	N O M B R E
RL	dst	ROTA A LA IZQUIERA UN BIT
RLC	dst	ROTA A LA IZQUIERDA UN BIT A TRAVES DEL DE ACARREO.
RR	dst	ROTA A LA DERECHA UN BIT
RRC	dst	ROTA A LA DERECHA UN BIT A TRAVES DEL DE ACARREO
SRA	dst	DESPLAZAMIENTO ARITMETICO A LA DERECHA
SWAP	dst	INTERCAMBIO DE INFORMACION ENTRE NIBBLES DE UN REGISTRO

INSTRUCCIONES DE TRANSFERENCIA DE BLOQUES

Permiten el movimiento de bloques de memoria de una localidad a otra a través de registros de trabajo.

INSTRUCCION	OPERANDO	NOMBRE
LDCI	@r , @rr	TRANSFERENCIA MEMORIA DE PROGRAMA A REGISTRO
LDCI	@rr, @r	TRANSFERENCIA REGISTRO A MEMORIA DE PROGRAMA
LDEI	@r , @rr	TRANSFERENCIA MEMORIA DE DATOS A REGISTRO
LDEI	@rr, @r	TRANSFERENCIA REGISTRO A MEMORIA DE DATOS

Las instrucciones LDC transfieren datos de memoria de programa a registros, mientras que las del tipo LDE hacen su función sobre memoria de datos.

INSTRUCCIONES DE MANIPULACION DE BITS

En el Z8 no existen bits especificos de un registro sobre los cuales operen directamente las instrucciones. Sin embargo las siguientes instrucciones pueden utilizarse para simular este tipo de operación:

INSTRUCCION	OPERANDO	NOMBRE
TCM	dst	PRUEBA COMPLEMENTO BAJO MASCARA
TM	dst	PRUEBA BAJO MASCARA
XOR	dst,src	OR EXCLUSIVO
AND	dst,src	AND LOGICO

Las dos primeras operan mediante máscaras para determinar el estado o valor de un bit específico, mientras que las últimas dos, posibilitan, mediante máscaras también, la modificación de bits individuales de cualquier registro.

CONJUNTO DE INSTRUCCIONES DEL Z8
LISTADAS POR CODIGO DE OPERACION

COD. DE OPERACION HEXADECIMAL	INSTRUCCION
00 R	DEC R
01 IR	DEC @R
02 $r_w r_v$	ADD r_w,r_v
03 rIr	ADD r,@r
04 R_w R_v	ADD R_w,R_v
05 R IR	ADD R,@R
06 R IM	ADD R,IM
07 IR IM	ADD @R,IM
08 R	LD r0,R
09 R	LD R,r0
0A RA	DJNZ r0,Direcc.
0B RA	JR cc0,Direc.
0C IM	LD r0,IM
0D DA_{Hi} DA_{Lo}	JP cc0,Direc.
0E	INC r0
0F =========	==========
10 R	RLC R
11 IR	RLC @R
12 $r_w r_v$	ADC r_w,r_v
13 rIr	ADC r,@r
14 R_w R_v	ADC R_w,R_v
15 R IR	ADC R,@R
16 R IM	ADC R,IM
17 IR IM	ADC @R,IM
18 R	LD r1,R
19 R	LD R,r1
1A RA	DJNZ r1,Direc.
1B RA	JR cc 1,Direc.
1C IM	LD r1,IM
1D DA_{Hi} DA_{Lo}	JP cc1,Direc.
1E	INC r1
1F =========	==========

COD.DE OPERACION HEXADECIMAL	INSTRUCCION
40 R	DA R
41 IR	DA @R
42 $r_w r_v$	OR r_w,r_v
43 rIr	OR r,@r
44 R_w R_v	OR R_w,R_v
45 R IR	OR R,@R
46 R IM	OR R,IM
47 IR IM	OR @R,IM
48 R	LD r4,R
49 R	LD R,r4
4A RA	DJNZ r4,Direc
4B RA	JR cc4,Direc
4C IM	LD r4,IM
4D DA_{Hi} DA_{Lo}	JP cc4,Direc
4E	INC r4
4F =========	==========
50 R	POP R
51 IR	POP @R
52 $r_w r_v$	AND r_w,r_v
53 rIr	AND r,@r
54 R_w R_v	AND R_w,R_v
55 R IR	AND R,@R
56 R IM	AND R,IM
57 IR IM	AND @R,IM
58 R	LD r5,R
59 R	LD R,r5
5A RA	DJNZ r5,Direc
5B RA	JR cc5,Direc
5C IM	LD r5,IM
5D DA_{Hi} DA_{Lo}	JP cc5,Direc
5E	INC r5
5F =========	==========

CONJUNTO DE INSTRUCCIONES DEL Z8
LISTADAS POR CODIGO DE OPERACION

COD. DE OPERACION HEXADECIMAL	INSTRUCCION
20 R	INC R
21 IR	INC @R
22 $r_d r_s$	SUB r_d, r_s
23 rIr	SUB r,@r
24 R_d R_s	SUB R_d, R_s
25 R IR	SUB R,@R
26 R IM	SUB R,IM
27 IR IM	SUB @R,IM
28 R	LD r2,R
29 R	LD R,r2
2A RA	DJNZ r2,Direc.
2B RA	JR cc2,Direc.
2C IM	LD r2,IM
2D DA_{Hi} DA_{Lo}	JP cc2,Direc.
2E	INC r2
2F ==========	==========
30 IRR	JP @RR
31 IM	SRP IM
32 $r_d r_s$	SBC r_d, r_s
33 rIr	SBC r,@r
34 R_d R_s	SBC R_d, R_s
35 R IR	SBC R,@R
36 R IM	SBC R,IM
37 IR IM	SBC @R,IM
38 R	LD r3,R
39 R	LD R,r3
3A RA	DJNZ r3,Direc.
3B RA	JR cc3,Direc.
3C IM	LD r3,IM
3D DA_{Hi} DA_{Lo}	JP cc3,Direc.
3E	INC r3
3F ==========	==========

COD.DE OPERACION HEXADECIMAL	INSTRUCCION
60 R	COM R
61 IR	COM @R
62 $r_d r_s$	TCM r_d, r_s
63 rIr	TCM r,@r
64 R_d R_s	TCM R_d, R_s
65 R IR	TCM R,@R
66 R IM	TCM R,IM
67 IR IM	TCM @R,IM
68 R	LD r6,R
69 R	LD R,r6
6A RA	DJNZ r6,Direc
6B RA	JR cc6,Direc
6C IM	LD r6,IM
6D DA_{Hi} DA_{Lo}	JP cc6,Direc
6E	INC r6
6F ==========	==========
70 R	PUSH R
71 IR	PUSH @R
72 $r_d r_s$	TM r_d, r_s
73 rIr	TM r,@r
74 R_d R_s	TM R_d, R_s
75 R IR	TM R,@R
76 R IM	TM R,IM
77 IR IM	TM @R,IM
78 R	LD r7,R
79 R	LD R,r7
7A RA	DJNZ r7,Dire
7B RA	JR cc7,Dire
7C IM	LD r7,IM
7D DA_{Hi} DA_{Lo}	JP cc7,Dire
7E	INC r7
7F ==========	==========

CONJUNTO DE INSTRUCCIONES DEL Z8
LISTADAS POR CODIGO DE OPERACION

COD. DE OPERACION HEXADECIMAL		INSTRUCCION
80	RR	DECW RR
81	IRR	DECW @RR
82	r1,r	LDE R,@rr
83	Ir1rr	LDEI @r,@rr
84		---
85		---
86		---
87		---
88	R	LD r8,R
89	R	LD R,r8
8A	RA	DJNZ r8,Direcc.
8B	RA	JR cc8,Direc.
8C	IM	LD r8,IM
8D	DA_{Hi} DA_{Lo}	JP cc8,Direc.
8E		INC r8
8F		DI
90	R	RL R
91	IR	RL @R
92	Ir rr	LDC @rr,r
93	IrrIr	LDEI @rr,@r
94		----
95		----
96		----
97		----
98	R	LD r9,R
99	R	LD R,r9
9A	RA	DJNZ r9,Direc.
9B	RA	JR cc9,Direc.
9C	IM	LD r9,IM
9D	DA_{Hi} DA_{Lo}	JP cc9,Direc.
9E		INC r9
9F		EI

COD.DE OPERACION HEXADECIMAL		INSTRUCCION
C0	R	RRC R
C1	IR	RRC @R
C2	rIrr	LDC r,@rr
C3	IrIrr	LDCI @r,@rr
C4		----
C5		----
C6		----
C7	rx R	LD r,x,R
C8	R	LD r12,R
C9	R	LD R,r12
CA	RA	DJNZ r12,Dire
CB	RA	JR cc12,Direc
CC	IM	LD r12,IM
CD	DA_{Hi} DA_{Lo}	JP cc12,Direc
CE		INC r12
CF		RCF
D0	R	SRA R
D1	IR	SRA @R
D2	rIrr	LDC r,@rr
D3	Irrr	LDCI @rr,r
D4	IRR	CALL @RR
D5		---
D6	DA_{Hi} DA_{Lo}	CALL Direc.
D7	rx r	LD R,x,r
D8	R	LD r13,R
D9	R	LD R,r13
DA	RA	DJRZ r13,Dire
DB	RA	JR cc13,Direc
DC	IM	LD r13,IM
DD	DA_{Hi} DA_{Lo}	JP cc13,Direc
DE		INC r13
DF		SCF

CONJUNTO DE INSTRUCCIONES DEL Z8
LISTADAS POR CODIGO DE OPERACION

COD. DE OPERACION HEXADECIMAL		INSTRUCCION
A0	RR	INCW RR
A1	IRR	INCW @RR
A2	$r_x r_y$	CP r_x, r_y
A3	rIr	CP r,@r
A4	R_x R_y	CP R_x, R_y
A5	R IR	CP R,@R
A6	R IM	CP R,IM
A7	IR IM	CP @R,IM
A8	R	LD r10,R
A9	R	LD R,r10
AA	RA	DJNZ r10,Direc.
AB	RA	JR CC10,Direc.
AC	IM	LD r10,IM
AD	DA_{Hi} DA_{Lo}	JP cc10,Direc
AE		INC r10
AF		RET
B0	R	CLR R
B1	IR	CLR @R
B2	$r_x r_y$	XOR r_x, r_y
B3	rIr	XOR r,@r
B4	R_x R_y	XOR R_x, R_y
B5	R IR	XOR R,@R
B6	R IM	XOR R,IM
B7	IR IM	XOR @R,IM
B8	R	LD r11,R
B9	R	LD R,r11
BA	RA	DJNZ r11,Direc.
BB	RA	JR cc11,Direcc.
BC	IM	LD r11,IM
BD	DA_{Hi} DA_{Lo}	JP cc11,Direc.
BE		INC r11
BF		IRET

COD.DE OPERACION HEXADECIMAL		INSTRUCCION
E0	R	RR R
E1	IR	RR @R
E2		----
E3	rIr	LD r,@r
E4	R_x R_y	LD R_x, R_y
E5	R_x IR_y	LD R^x,@R_y
E6	R IM	LD R,IM
E7	IR IM	LD @R,IM
E8	R	LD r14,R
E9	RA	LD R,r14
EA	RA	DJRZ r14,Dir.
EB	IM	LD r14,IM
EC	DA_{Hi} DA_{Lo}	JR cc14,Direc
ED		JP cc14,Direc
EE		INC r14
EF		CCF
F0	R	SWAP R
F1	IR	SWAP @R
F2		----
F3	$Ir^x r_y$	LD @r_x,r_y
F4		-----
F5	IR_x R_y	LD @R_x,R_y
F6		-----
F7		-----
F8	R	LD r15,R
F9	R	LD R,r15
FA	RA	DJNZ r15,Dir
FB	RA	JR cc15,Dire
FC	IM	LD R15,IM
FD	DA_{Hi} DA_{Lo}	JP cc15,Dire
FE		INC r15
FF		NOP

TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS						OPERACION DESARROLLADA
						C	Z	S	O	D	H	
REFERENCIA A MEMORIA PRIMARIA	LDC	@rr,r	D2 rIrr	2	12,0							[[rr]] <—— [r] Transfiere el contenido del registro de trabajo seleccionado a la localidad externa de memoria direccionada por el contenido del registro de trabajo par .
	LDC	r,@rr	C2 Irrr	2	12,0							[r] <—— [[rr]] Tranfiere el contenido de la localidad de memoria externa direccionada por el contenido del registro par al registro de trabajo seleccionado.
	LDCI	@rr,@r	D3 IrrIr	2	18,0							[[rr]] <—— [[r]] , [r] <—— [r] + 1 , [rr] <—— [rr] + 1 Transfiere el contenido del registro direccionado indirectamente a la localidad de memoria externa direccionada por el registro de trabajo par. Incrementa la dirección de memoria y la dirección del registro de trabajo.
	LDCI	@r,@rrr	C3 IrIrr	2	18,0							[[r]] <—— [[rr]], [r] <—— [r] 1, [rr] <—— [rr]+1 Transfiere el contenido de la localidad externa de memoria direccionada por el registro par al registro direccionado indirectamente. Incrementa la dirección de memoria y la dirección del registro de trabajo.
	LDE	@rr,r	92 rIrr	2	12,0							Lo mismo que LDC pero con DM activa baja para indicar memoria de datos.
	LDE	r,@rr	82 Irrr	2	12,0							
	LDEI	@rr,@r	93 IrrIr	2	18,0							Lo mismo que LDCI pero con DM activa baja para indicar memoria de datos.
	LDEI	@r,@rr	83 IrIrr	2	18,0							
INMEDIATO	LD	R, IM	E6 R IM	3	10,5							[R] <—— IM Transfiere el dato inmediato al registro seleccionado.
	LD	@R,IM	E7 IR IM	3	10,5							[[R]] <—— IM Transfiere el dato inmediato al registro direccionado indirectamente.
	LD	r,IM	rC IM	2	6,5							[r] <—— IM Transfiere el dato inmediato al registro de trabajo seleccionado.
	SRP	IM	31	2	6,1							[FD$_{16}$] <—— IM Transfiere el dato inmediato a SRP Registro APUNTADOR A REGISTROS.

TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
REFERENCIA A MEMORIA PRIMARIA	LDC	@rr,r	D2 rIrr	2	12,0							[[rr]] <—— [r] Transfiere el contenido del registro de trabajo seleccionado a la localidad externa de memoria direccionada por el contenido del registro de trabajo par .
	LDC	r,@rr	C2 Irrr	2	12,0							[r] <—— [[rr]] Transfiere el contenido de la localidad de memoria externa direccionada por el contenido del registro par al registro de trabajo seleccionado.
	LDCI	@rr,@r	D3 IrrIr	2	18,0							[[rr]] <—— [[r]] , [r] <—— [r] + 1 , [rr] <—— [rr] + 1 Transfiere el contenido del registro direccionado indirectamente a la localidad de memoria externa direccionada por el registro de trabajo par. Incrementa la dirección de memoria y la dirección del registro de trabajo.
	LDCI	@r,@rrr	C3 IrIrr	2	18,0							[[r]] <—— [[rr]], [r] <—— [r] 1, [rr] <—— [rr]+1 Transfiere el contenido de la localidad externa de memoria direccionada por el registro par al registro direccionado indirectamente. Incrementa la dirección de memoria y la dire - ción del registro de trabajo.
	LDE	@rr,r	92 rIrr	2	12,0							Lo mismo que LDC pero con DM activa baja para indicar memoria de datos.
	LDE	r,@rr	82 Irrr	2	12,0							
	LDEI	@rr,@r	93 IrrIr	2	18,0							Lo mismo que LDCI pero con DM activa baja para indicar memoria de datos.
	LDEI	@r,@rr	83 IrIrr	2	18,0							
INMEDIATO	LD	R, IM	E6 R IM	3	10,5							[R] <—— IM Tranfiere el dato inmediato al registro seleccionado.
	LD	@R,IM	E7 IR IM	3	10,5							[[R]] <—— IM Transfiere el dato inmediato al registro direccionado indirectamente.
	LD	r,IM	rC IM	2	6,5							[r] <—— IM Transfiere el dato inmediato al registro de trabajo seleccionado.
	SRP	IM	31	2	6,1							[FD$_{16}$] <—— IM Transfiere el dato inmediato a SRP Registro APUNTADOR A REGISTROS.

TIPO	MNEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
SALTO	DJNZ	r,RA	ra Desplazam.	2	12/10 2/5							[r] <—— [r] -1, si [r] <> 0 entonces [PC] <—— [PC] + Desplazam. Decrementa el contenido del registro de trabajo seleccionado. Si no es cero ejecuta el salto relativo de programa
	JP	cc,DA	ccD DA_{HB}...DA_{LB}	3	12/10,0							Si cc es 'verdadero' entonces [PC] <—— DA Salta a la localidad de memoria direccionada directamente si cc es verdadero.
	JP	@RR	30 RR	2	8,0							[PC] <—— [RR] Salta a la localidad de memoria cuya dirección es el contenido del registo par
	JR	cc,RA	ccB Desp.	2	12/10,3							Si cc es 'verdad' entonces [PC] <—— [PC]+Desp. ejecuta un salto relativo si cc es verdadero.
LLAMADO Y RETORNO DE SUBRUTINAS	CALL	DA	D6 DA_{HB}...DA_{LB}	3	20,0							[SP] <—— [SP]-2,[[SP]]<— [PC],[PC] <—— [DA] Salta a la dirección direccionada por la subrutina.
	CALL	@RR	D4 RR	2	20,0							[SP] <—— [SP]-2,[[SP]]<——[PC],[PC]<——[RR] Salta a la subrutina cuya dirección es contenida en el registro par.
	RET		AF	1	14,0							[PC] <—— [[SP]], [SP] <—— [SP] + 2 Regreso de una subrutina.

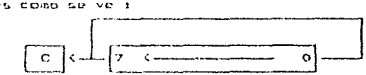
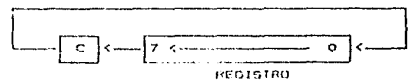
TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
OPERADOR INMEDIATO	ADC	R,IM	16 R IM	3	10,5	x	x	x	x	0	x	[R] <—— [R] + IM + C Suma el dato inmediato con el acarreo con el contenido del registro seleccionado.
	ADC	@R,IM	17 IR IM	3	10,5	x	x	x	x	0	x	[[R]] <—— [[R]] + IM + C Suma el dato inmediato con acarreo al contenido del registro direccionado indirectamente.
	ADD	R,IM	06 R IM	3	10,5	x	x	x	x	0	x	[R] <—— [R] + IM Lo mismo que ADC pero sin sumar el acarreo.
	ADD	@R,IM	07 IR IM	3	10,5	x	x	x	x	0	x	[[R]] <—— [[R]] + IM
	AND	R,IM	56 R IM	3	10,5		x	x	0			[R] <—— [R] AND IM Como ADC pero desarrolla una operación AND
	AND	@R,IM	57 IR IM	3	10,5		x	x	0			[[R]] <—— [[R]] AND IM lógica.
	CP	R,IM	6 R IM	3	10,5	x	x	x	x			[R] <—— IM Como ADC pero esta operación solo
	CP	@R,IM	A7 IR IM	3	10,5	x	x	x	x			[[R]] <—— IM afecta el estado de las banderas.
	OR	R,IM	46 R IM	3	10,5		x	x	0			[R] <—— [R] OR IM Como ADC pero desarrollando una operación OR lógica.
	OR	@R,IM	47 IR IM	3	10,5		x	x	0			[[R]]<——[[R]] OR IM
	SBC	R,IM	36 R IM	3	10,5	x	x	x	x	1	x	[R] <—— [R] - IM - C Como ADC pero ejecutando la operación de resta
	SBC	@R,IM	37 IR IM	3	10,5	x	x	x	x	1	x	[[R]]<—— [[R]] -IM -C con acarreo.
	SUB	R,IM	26 R IM	3	10,5	x	x	x	x	1	x	[R] <—— [R] - IM Como ADC pero ejecutando la operación de resta.
	SUB	@R,IM	27 IR IM	3	10,5	x	x	x	x	1	x	[[R]] <—— [[R]] - IM

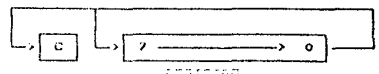
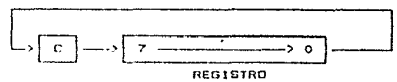
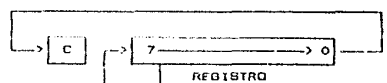
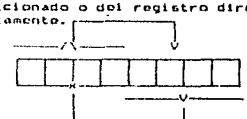
TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
OPERADOR INMEDIATO	TCM	R,IM	66 R IM	3	10,5		X	X	0			[R] AND (IM)' Como ADC pero esta opeación solo afecta el estado del registro de banderas.
	TCM	@R,IM	67 IR IM	3	10,5		X	X	0			[[R]] AND (IM)'
	TM	R,IM	76 R IM	3	10,5		X	X	0			[R] AND IM Como ADC pero esta operación solo afecta el estado del regis-
	TM	@R,IM	77 IR IM	3	10,5		X	X	0			[[R]] AND IM
	XOR	R,IM	B6 R IM	3	10,5		X	X	0			[R] <—— [R] O IM Como ADC pero ejecutando la
	XOR	@R,IM	B7 IR IM	3	10,5		X	X	0			[[R]] <—— [[R]] O IM operación lógica OR exclusiva.
MOVIMIENTO DE REGISTRO A REGISTRO Y E/S	LD	r,R	r8 R	2	6,5							[r] <—— [R] Transfiere el contenido del registro seleccionado al registro de trabajo seleccionado.
	LD	R,r	r9 R	2	6,5							[R] <—— [r] Transfiere el contenido del registro de trabajo seleccionado al registro seleccionado.
	LD	Rw,Rv	E4 Rw Rv	3	10,5							[Rw] <—— [Rv] transfiere el contenido del registro seleccionado (Rv) al registro seleccionado (Rw).
	LD	Rw,@Rv	E5 Rw IRv	3	10,5							[Rw] <—— [[Rv]]
	LD	@Rw,Rv	F5 IRw Rv	3	10,5							[[Rw]] <—— [Rv] Transfiere datos entre los registros seleccionados y los registros direccionados indirectamente.
	LD	r,IX,R	C7 rIX R									[r] <—— [R + (IX)] Transfiere datos entre los registros de trabajo seleccio
	LD	R,IX,r	D7 rIX R									[R + (IX)] <—— [r] nados y registros direcciona- dos usando direccionamiento indexado.

TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	C	D	II	OPERACION DESARROLLADA
OPERADOR REGISTRO A REGISTRO (CONTINUA)	ADC	r_w,r_v	12 $r_w r_v$	2	6,5	X	X	X	X	0	X	$[r_w] \leftarrow [r_w] + [r_v] + C$ Suma el contenido del registro de trabajo fuente seleccionado con acarreo al contenido del registro de trabajo destino seleccionado.
	ADC	$r_w,@r_v$	13 $r_w ir_v$	2	6,5	X	X	X	X	0	X	$[r_w] \leftarrow [r_w] + [[r_v]] + C$ Suma el contenido del registro de trabajo fuente direccionado indirectamente con acarreo al contenido del registro de trabajo destino seleccionado.
	ADC	R_w,R_v	14 $R_w R_v$	3	10,5	X	X	X	X	0	X	$[R_w] \leftarrow [R_w] + [R_v] + C$ Suma el contenido del registro fuente seleccionado con acarreo al registro destino seleccionado.
	ADC	$R_w,@R_v$	15 $R_w IR_v$	3	10,5	X	X	X	X	0	X	$[R_w] \leftarrow [R_w] + [[R_v]] + C$ Suma el contenido del registro fuente seleccionado indirectamente con acarreo al registro destino seleccionado.
	ADD ADD ADD ADD	r_w,r_v $r_w,@r_v$ R_w,R_v $R_w,@R_v$	02 $r_w r_v$ 03 $r_w ir_v$ 04 $R_w R_v$ 05 $R_w IR_v$	2 2 3 3	6,5 6,5 10,5 10,5	X X X X	X X X X	X X X X	X X X X	0 0 0 0	X X X X	$[r_w] \leftarrow [r_w] + [r_v]$ $[r_w] \leftarrow [r_w] + [[r_v]]$ $[R_w] \leftarrow [R_w] + [R_v]$ $[R_w] \leftarrow [R_w] + [[R_v]]$ Suma el contenido del registro fuente al registro destino usando una de las opciones descritas para ADC.
	AND AND AND AND	r_w,r_v $r_w,@r_v$ R_w,R_v $R_w,@R_v$	52 $r_w r_v$ 53 $r_w ir_v$ 54 $R_w R_v$ 55 $R_w IR_v$	2 2 3 3	6,5 6,5 10,5 10,5		X X X X	X X X X	X X X X			$[r_w] \leftarrow [r_w]$ AND $[r_v]$ $[r_w] \leftarrow [r_w]$ AND $[[r_v]]$ $[R_w] \leftarrow [R_w]$ AND $[R_v]$ $[R_w] \leftarrow [R_w]$ AND $[[R_v]]$ Realiza la operación lógica AND del registro fuente con el registro destino utilizando una de las opciones de direccionamiento de ADC.
	CP CP CP CP	r_w,r_v $r_w,@r_v$ R_w,R_v $R_w,@R_v$	A2 $r_w r_v$ A3 $r_w ir_v$ A4 $R_w R_v$ A5 $R_w IR_v$	2 2 3 3	6,5 6,5 10,5 10,5	X X X X	X X X X	X X X X	X X X X			$[r_w] - [r_v]$ $[r_w] - [[r_v]]$ $[R_w] - [R_v]$ $[R_w] - [[R_v]]$ Compara el registro fuente con el registro destino usando uno de los modos de direccionamiento descrito para ADC.

TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
OPERADOR REGISTRO A REGISTRO	OR	r_a,r_y	42 $r_a r_y$	2	6,5		X	X	X			$[r_a] \leftarrow [r_a]$ OR $[r_y]$
	OR	$r_a,@r_y$	43 $r_a ir_y$	2	6,5		X	X	X			$[r_a] \leftarrow [r_a]$ OR $[[r_y]]$
	OR	R_a,R_y	44 $R_a R_y$	3	10,5		X	X	X			$[R_a] \leftarrow [R_a]$ OR $[R_y]$
	OR	$R_a,@R_y$	45 $R_a IR_y$	3	10,5		X	X	X			$[R_a] \leftarrow [R_a]$ OR $[[R_y]]$ Realiza la operación lógica OR del registro fuente con el registro destino usando uno de los direccionamientos descritos para ADC.
	SBC	r_a,r_y	32 $r_a r_y$	2	6,5	X	X	X	X	1	X	$[r_a] \leftarrow [r_a] - [r_y] - C$
	SBC	$r_a,@r_y$	33 $r_a ir_y$	2	6,5	X	X	X	X	1	X	$[r_a] \leftarrow [r_a] - [[r_y]] - C$
	SBC	R_a,R_y	34 $R_a R_y$	3	10,5	X	X	X	X	1	X	$[R_a] \leftarrow [R_a] - [R_y] - C$
	SBC	$R_a,@R_y$	35 $R_a IR_y$	3	10,5	X	X	X	X	1	X	$[R_a] \leftarrow [R_a] - [[R_y]] - C$ Resta con acarreo el registro fuente del registro destino usando uno de los modos de direccionamiento descrito para ADC.
	SUB	r_a,r_y	22 $r_a r_y$	2	6,5	X	X	X	X	1	X	$[r_a] \leftarrow [r_a] - [r_y]$
	SUB	$r_a,@r_y$	23 $r_a ir_y$	2	6,5	X	X	X	X	1	X	$[r_a] \leftarrow [r_a] - [[r_y]]$
	SUB	R_a,R_y	24 $R_a R_y$	3	10,5	X	X	X	X	1	X	$[R_a] \leftarrow [R_a] - [R_y]$
	SUB	$R_a,@R_y$	25 $R_a IR_y$	3	10,5	X	X	X	X	1	X	$[R_a] \leftarrow [R_a] - [[R_y]]$ Resta el registro fuente del registro destino usando uno de los modos de direccionamiento descritos para ADC.
	TCM	r_a,r_y	62 $r_a r_y$	2	6,5		X	X	0			$[r_a]$ AND $([r_y])'$
	TCM	$r_a,@r_y$	63 $r_a ir_y$	2	6,5		X	X	0			$[r_a]$ AND $([[r_y]])'$
	TCM	R_a,R_y	64 $R_a R_y$	3	10,5		X	X	0			$[R_a]$ AND $([R_y])'$
	TCM	$R_a,@R_y$	65 $R_a IR_y$	3	10,5		X	X	0			$[R_a]$ AND $([[R_y]])'$ Realiza la operación lógica AND entre el primer registro y el complemento del segundo registro usando uno de los modos de direccionamiento descrito para ADC.
	TM	r_a,r_y	72 $r_a r_y$	2	6,5		X	X	0			$[r_a]$ AND $[r_y]$
	TM	$r_a,@r_y$	73 $r_a ir_y$	2	6,5		X	X	0			$[r_a]$ AND $[[r_y]]$
	TM	R_a,R_y	74 $R_a R_y$	3	10,5		X	X	0			$[R_a]$ AND $[R_y]$
	TM	$R_a,@R_y$	75 $R_a IR_y$	3	10,5		X	X	0			$[R_a]$ AND $[[R_y]]$ Realiza la operación lógica AND entre el contenido del primer registro y el contenido utilizando uno de los modos de direccionamiento descrito para ADC.

TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
OR A PE OR IA SO. DT OR OR	XOR	r_w,r_y	B2 $r_w r_y$	2	6,5		X	X	0			[r_w] <—— [r_w] + [r_y]
	XOR	r_w,@r_y	B3 r_w Ir$_y$	2	6,5		X	X	0			[r_w] <—— [r_w] + [[r_y]]
	XOR	R_w,R_y	B4 $R_w R_y$	3	10,5		X	X	0			[R_w] <—— [R_w] + [R_y]
	XOR	R_w,@R_y	B5 R_w IR$_y$	3	10,5		X	X	0			[R_w] <—— [R_w] + [[R_y]] Realiza la operación lógica OR exclusiva del registro fuente con el registro destino usando una de las opciones de direccionamiento descritas para ADC.
OPERADOR A REGISTRO	CLR	R	B0 R	2	6,5							[R] <—— 0 Borra el registro seleccionado.
	CLR	@R	B1 IR	2	6,5							[[R]] <—— 0 Borra el registro direccionado indirectamente.
	COM	R	60 R	2	6,5		X	X	0			[R] <—— ([R])' Complementa el contenido del registro seleccionado.
	COM	@R	61 IR	2	6,5		X	X	0			[[R]] <—— ([[R]])' Complementa el contenido del registro direccionado indirectamente.
	DA	R	40 R	2	8,5	X	X	X	U			Realiza el ajuste decimal del registro seleccionado.
	DA	@R	41 IR	2	8,5	X	X	X	U			Realiza el ajuste decimal del registro direccionado indirectamente.
	DEC	R	00 R	2	6,5		X	X	X			[R] <—— [R] -1 Decrementa el contenido del registro seleccionado.
	DEC	@R	01 IR	2	8,5		X	X	X			[[R]] <—— [[R]] - 1 Decrementa el contenido del registro direccionado indirectamente.
	DECW	RR	80 RR	2	10,5		X	X	X			[RR] <—— [RR] - 1 Decrementa el contenido del registro par seleccionado.
	DECW	@RR	81 IRR	2	10,5		X	X	X			[[RR]] <—— [[RR]] - 1 Decrementa el contenido del registro par direccionado indirectamente.

TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
OPERADOR A REGISTRO	INC	r	rE	1	6,5		X	X	X			[r] <—— [r] + 1 Incrementa el contenido del registro de trabajo seleccionado.
	INC	R	20 R	2	6,5		X	X	X			[R] <—— [R] + 1 Incrementa el contenido del registro seleccionado.
	INC	@R	21 IR	2	6,5		X	X	X			[[R]] <—— [[R]] +1 Incrementa el contenido del registro direccionado indirectamente.
	INCW	RR	A0 RR	2	10,5		X	X	X			[RR] <—— [RR] + 1 Incrementa el contenido del registro par seleccionado.
	INCW	@RR	A1 IRR	2	10,5		X	X	X			[[RR]] <—— [[RR]] + 1 Incrementa el contenido del registro par seleccionado, direccionado indirectamente.
	RL RL	R @R	90 R 91 IR	2 2	6,5 6,5	X X	X X	X X	X X			Ejecuta un rotamiento hacia la izquierda con acarreo sobre el registro seleccionado o sobre el registro direccionado indirectamente. Esto es como se ve : 
	RLC RLC	R @R	10 R 11 IR	2 2	6,5 6,5	X X	X X	X X	X X			Ejecuta una rotación a la izquierda a través del bit de acarreo sobre el registro seleccionado o sobre el registro direccionado indirectamente. 

TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
OPERADOR A REGISTRO	RR RR	R @R	E0 R E1 IR	2 2	6,5 6,5	X X	X X	X X	X X			Ejecuta una rotación a la derecha con acarreo del registro seleccionado o en el registro seleccionado indirectamente. 
	RRC RRC	R @R	10 R 11 R	2 2	6,5 6,5	X X	X X	X X	X X			Ejecuta una rotación a la derecha a través del bit de acarreo sobre el registro seleccionado o sobre el registro direccionado indirectamente. 
	SRA SRA	R @R	D0 R D1 IR	2 2	6,5 6,5	X X	X X	X X	0 0			Ejecuta un corrimiento aritmético a la derecha con acarreo sobre el registro seleccionado o sobre el registro direccionado indirectamente. 
	SWAP SWAP	R @R	F0 R F1 IR	2 2	6,5 6,5	X X	X X	X X	X X			Intercambia los nibbles del registro seleccionado o del registro direccionado indirectamente. 

TIPO	NEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS C	Z	S	O	D	H	OPERACION DESARROLLADA
STACK	POP	R	50 R	2	10,5							[R] <—— [[SP]], [SP] <—— [SP] + 1
	POP	@R	51 IR	2	10,5							[[R]] <—— [[SP]] , [SP] + 1 Realiza el POP al stack del registro direccionado directa o indirectamente.
	PUSH	R	70 R	2	10/12,1							[SP] <—— [SP] -1 , [[SP]] <—— [R]
	PUSH	@R	71 IR	2	12/14,1							[SP] <—— [SP] -1 , [[SP]] <—— [[R]] Realiza el PUSH hacia el stack desde el registro direccionado directa o indirectamente
INTERRUPCIONES	DI		8F	1	6,1							Deshabilita interrupciones: $[FB_{16}] \leftarrow [FB_{16}]$ AND $7F_{16}$
	EI		9F	1	6,1 16,0							Habilita interrupciones : $[FB_{16}] \leftarrow [FB_{16}]$ OR 80_{16}
	IRET		BF	1								[PC] <—— [[SP]], [SP] <—— [SP] +2, $[FB_{16}] \leftarrow [FB_{16}]$ OR 80_{16} Regreso de una rutina de servicio de interrupcion y habilita interrupciones.
ESTADO	CCF		EF	1	6,5	*						[C] <—— [[C]]' Complementa la bandera de acarreo.
	RCF		CF	1	6,5	0						[C] <—— 0 Pone a cero la bandera de acarreo.
	SCF		DF	1	6,5	1						[C] <—— 1 Pone a 1 la bandera de acarreo.
	NOP		FF	1	6,0							Sin operación.

CAPITULO IV

DESCRIPCION DEL SIS Z8

A partir de este capítulo entraremos en materia del SIS Z8, comenzando por una descripción general, en la que se hará un recuento de los bloques funcionales de que dispone el usuario del SIS Z8. Asimismo, también se aclarará su operación y en la última sección del capítulo se introducen los comandos del SCAN Z8.

En los dos siguientes capítulos, se explica ampliamente el funcionamiento del hardware y del software, para concluir en el Capítulo 7 con los pasos del diseño del SIS Z8.

Es evidente que hemos invertido el orden cronológico del desarrollo del SIS Z8. Esta organización obedece al objetivo de darle un carácter inductivo a la redacción para facilitar el entendimiento del proyecto.

ELEMENTOS DEL SIS Z8.

Cuando se revisaron los objetivos de este proyecto en el Capítulo 1, se presentaron superficialmente los principales elementos del SIS Z8. Aquí abundaremos sobre el tema comenzando por los diagramas de bloques mostrados en las siguientes páginas, en las que se hace una división de hardware y software respectivamente.

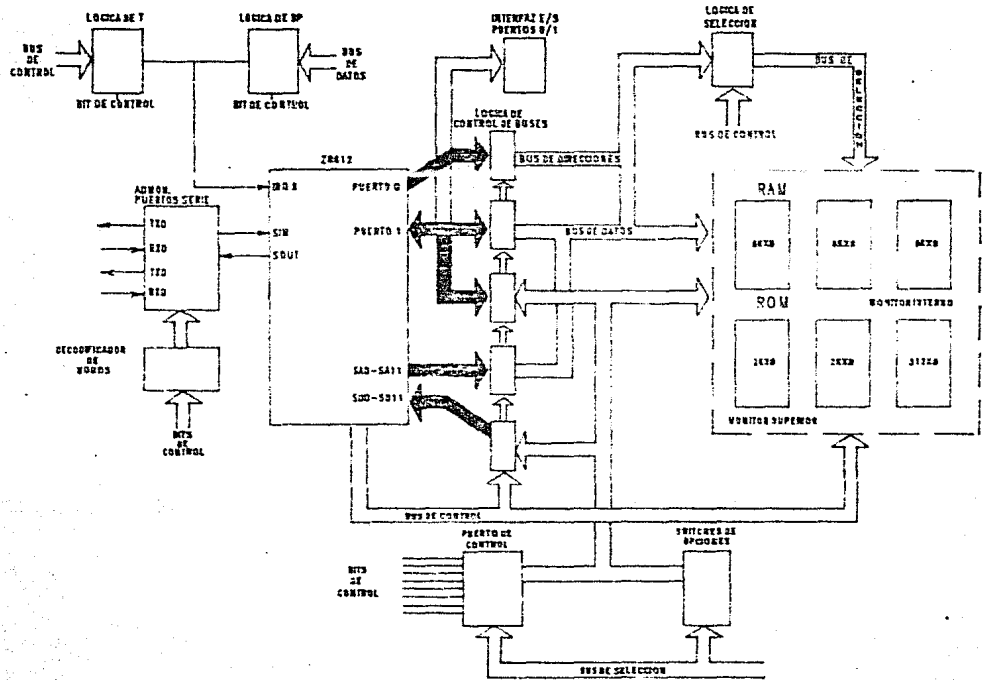

LOGICA DE T
LOGICA DE RP
BUS DE CONTROL
BIT DE CONTROL
BUS DE DATOS
INTERFAZ E/S PUERTOS 0/1
LOGICA DE SELECCION
BUS DE CONTROL
LOGICA DE CONTROL DE BUSES
Z8412
BUS DE DIRECCIONES
ADMON. PUERTOS SERIE
PUERTO 0
PUERTO 1
BUS DE DATOS
RAM
ROM
MONITOR/INTERRUPCION
MONITOR SUPERIOR
DECODIFICADOR DE WORDS
BITS DE CONTROL
PUERTO DE CONTROL
SWITCHES DE OPCIONES
BUS DE SELECCION

FIGURA (4.1)

De la Figura (4.1) se distinguen los siguientes bloques, cuya función se explica a continuación.

- Z8612.

Es el procesador central del SIS Z8 y el elemento con que el usuario trabaja en el desarrollo de sus programas. Todos los pins en sus distintas modalidades pueden ser empleados en el programa de prueba a excepción de P27 y P32.

- PUERTOS SERIE.

Cuenta con dos puertos asincrónos seriales compatibles con niveles RS232C, que pueden ser configurados en 5 modos distintos de operación.

- BUSES.

a) Direcciones y Datos.

Como el procesador es en realidad un Z8 en su versión de desarrollo, existen 5 buses: 3 de ellos se muestran en la Figura (4.1) sombreados y los llamaremos Buses Locales o Primarios, éstos son obtenidos de la configuración de los Puertos 0 y 1, como bus multiplexado Datos/Direcciones; y 2 buses: uno de Datos y otro de Direcciones que dispone el Z8612 para accesar a la memoria externa que sustituye a la ROM interna, de la cual carece esta versión. Los 2 buses restantes serán llamados Buses Principales o del Sistema, indistintamente, y son los que accesan directamente a los elementos restantes del SIS Z8.

b) Control y Selección.

Existen además otros buses: uno de Control y otro de Selección. El primero está formado por señales de temporización y control de buses. El segundo por señales de decodificación.

- LOGICA DE CONTROL DE BUSES.

Se encarga de formar los buses principales a partir de los buses locales.

- INTERFAZ ENTRADA/SALIDA DE LOS PUERTOS 0 y 1.

Como los Puertos 0 y 1 del Z8 son empleados por el SIS Z8 como buses Datos/Direcciones, esta interfaz se encarga de evitar interferencia entre las señales de direcciones y datos, y los

circuitos que el usuario pueda conectar a las terminales de la interfaz, cuando el SIS Z8 trabaja en el modo Monitor. En el modo Depurador, la interfaz es programada para comportarse como una conexión directa entre las terminales de los Puertos 0 y 1, y los circuitos conectados externamente.

- LOGICA DE SELECCION.

Se encarga de decodificar las direcciones presentes en los buses principales, para seleccionar las memorias y los puertos de control del SIS Z8.

- MEMORIAS DE SOLO LECTURA.

Contienen el código del SCAN Z8, y ocupan aproximadamente 4108 bytes del espacio direccionable de memoria de programa.

- MEMORIAS DE ACCESO ALEATORIO.

El SIS Z8 dispone de 24 KB de Memoria de Acceso Aleatorio, que es usada en parte por el SIS Z8. Sin embargo, un bloque de 8 KB de RAM puede ser accesada indistintamente en 2 localidades lógicas distintas, una en Memoria de Programa y otra en Memoria de Datos. De lo anterior la capacidad de Memoria de Acceso Aleatorio del SIS Z8 aumenta virtualmente hasta 32 KB. Similarmente un bloque de 3.5 KB de RAM es empleado para simular la Memoria Interna del Z8. Esto es, el programa de prueba puede ser depurado en las mismas localidades donde finalmente será accesado en el diseño final del usuario. El Mapa de Memoria del SIS Z8 puede ser consultado en la Figura (5.x) del Capítulo 5.

- LOGICA DE TRACE Y BREAK POINTS.

Se encargan de interrumpir el programa de prueba para facilitar el seguimiento y comprobar la secuencia del mismo, esta circuiteria se vale de la petición de interrupción IRQ0 para hacer el efecto de los comandos T(race) y B(reak) P(oints) del SCAN Z8.

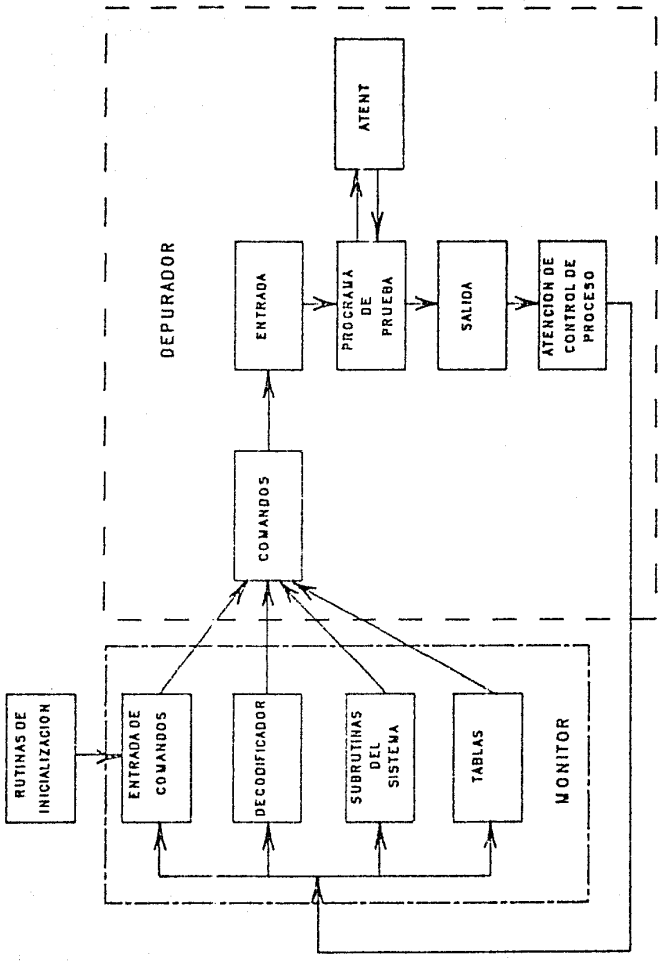

DEPURADOR
ATENT
ENTRADA
PROGRAMA DE PRUEBA
SALIDA
ATENCION DE CONTROL DE PROCESO
COMANDOS
MONITOR
RUTINAS DE INICIALIZACION
ENTRADA DE COMANDOS
DECODIFICADOR
SUBRUTINAS DEL SISTEMA
TABLAS

FIGURA (4.2)

En la Figura (4.2) se puede apreciar la organización de las rutinas que componen el software con que cuenta nuestro sistema.

A continuación se explica la función de dichas rutinas.

RUTINAS DE INICIALIZACION.

Estas rutinas tienen varias funciones, como son:

Configurar al Z8 programando los Registros de Control de los Puertos 0, 1, 2 y 3; el Stack Pointer para inicializar la transmisión; los registros relacionados con los Preescaladores y Temporizadores TMR, PREO.

También en estas rutinas se asigna el Grupo de Registros de Trabajo utilizados exclusivamente para el software del sistema, el cual es el Grupo 10.

MONITOR.

El monitor es el encargado de analizar la información proveniente de los Puertos Serie y decidir el destino de la misma. Su extructura es tal, que trabaja como una máquina secuencial, al reset, el monitor se encuentra en Estado 0 y programa al Puerto Serie de manera que el SIS Z8 sea transparente a la comunicación Terminal-Computadora Anfitriona. En este estado analiza los caracteres enviados por la terminal en busca de la secuencia de acceso al SIS Z8. Para interpretar esta secuencia, el monitor se vale de los Estados 1 y 2. Si la secuencia es aceptada, el monitor prosigue al Estado 3 y el usuario está accesando exclusivamente al SIS Z8. Ahora el monitor espera la introducción de algún comando. El primer caracter recibido forza al monitor al Estado 4, donde espera el caracter (Carriage Return) que le indica al monitor que toda una linea de un comando ha sido introducida. Cada caracter recibido durante el Estado 4 es introducido a un Buffer que recide en el Archivo de Registros del Z8. La recepción del caracter de retorno CR envia al monitor, del Estado 4 al Estado Decodificador, que es el encargado de interpretar los caracteres en el Buffer, para determinar el comando requerido y pasar el control a la rutina de ejecución particular de dicho comando. Todos los comandos concluyen a través de una subrutina del SCAN Z8, que regresa el control al monitor en su Estado 3.

Después de la secuencia de acceso, el monitor permanecerá en los Estados 3, 4 y Decodificador, hasta un reset maestro. En estos estados el SIS Z8, tiene sus puertos programados de manera que solo pueda haber comunicación con la terminal, excepto cuando se ejecutan los comandos MST y TRM.

El usuario puede determinar que el monitor se encuentra en su Estado 3, con la aparición del (*), que es el prompt del SCAN Z8.

SUBRUTINAS DEL SCAN Z8.

Estas subrutinas son empleadas tanto por el monitor como por los comandos. Podría decirse que son programas de utileria para realizar diversas funciones comunes a los comandos y al monitor. Algunos ejemplos interesantes de ellos son los siguientes:

NOMBRE DE LA SUBRUTINA	FUNCION
ENVSINT	Envia una cadena de caracteres por un Puerto Serie. No emplea interrupciones, para entrar a ella es necesario indicarle la localidad de memoria donde comienza la cadena, y el último caracter de ésta debe ser un FFH.
ESPDAT	Espera la recepción de una secuencia generalmente numérica, de longitud máxima de 2 caracteres. Tiene incorporadas las rutinas que interpretan los caracteres DELETE y ESC.
HEXASC	Convierte un Código Hexadecimal a un Código ASCII de 2 digitos.
ASCHEX	Conversión inversa de HEXASC.
RECDIR	Hace la conversión de una dirección de memoria en Código ASCII, guardada en el Buffer de recepción del Estado 4 del monitor.

COMANDOS.

Son rutinas que ejecutan la función asociada al nombre del comando, que el usuario especificamente desea utilizar. El conjunto de comandos incluye en total 17 rutinas, entre las cuales se incluyen las más comunmente usadas en paquetes comerciales de depuración, tales como: despliegue y sustitución de memoria, etc. La explicación detallada de éstos se reserva para el Capítulo 6.

INSTALACION DEL SIS Z8.

Antes de poderse operar el SIS Z8, es necesario que el usuario lo instale correctamente, para lo cual debe seguir los 4 pasos siguientes:

1.- Debe energizarlo conectando 3 fuentes externas de corriente directa a las terminales del conector CN1 del SIS Z8, el cual consiste de 3 puntas de tierra etiquetadas con GND (internamente unidas) y 3 puntas etiquetadas con sus valores respectivos de voltaje, que deben cumplir con las siguientes especificaciones:

- Fuente Regulada de +5 V, 1 A.
- Fuente Regulada de +12 V, 250 A.
- Fuente Regulada de -12 V, 250 A.

Estas 3 fuentes pueden ser independientes, o bien, tener un nodo común. En el caso de ser independientes se deben conectar los 3 conectores de tierra a sus repectivas terminales de referencia. No es necesario unir los nodos de referencia de las fuertes, puesto que esto se ha implementado ya en el SIS Z8.

2.- Hacer las conexiones pertinentes para la comunicación con la terminal y con la computadora anfitriona, en caso de usarse. Los Puertos Serie del SIS Z8 tienen sus salidas alambradas a conectores tipo DB25 y sus niveles son compatibles con los del protocolo RS232C. De este protocolo, el SIS Z8 maneja solamente 2 señales: RDX y TXD. La asignación de pins de estos conectores es la mostrada en la Figura (4.3):

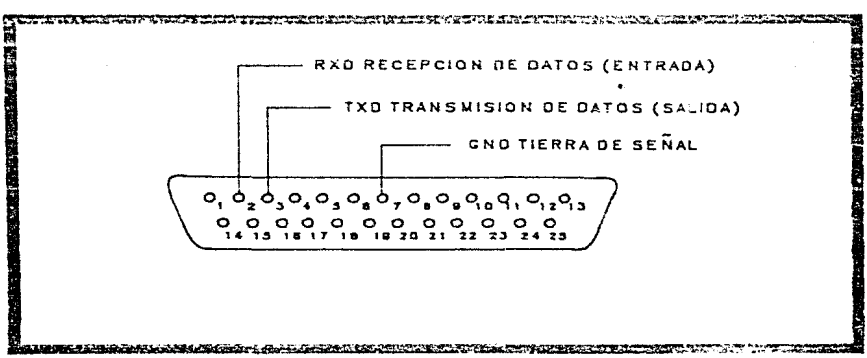
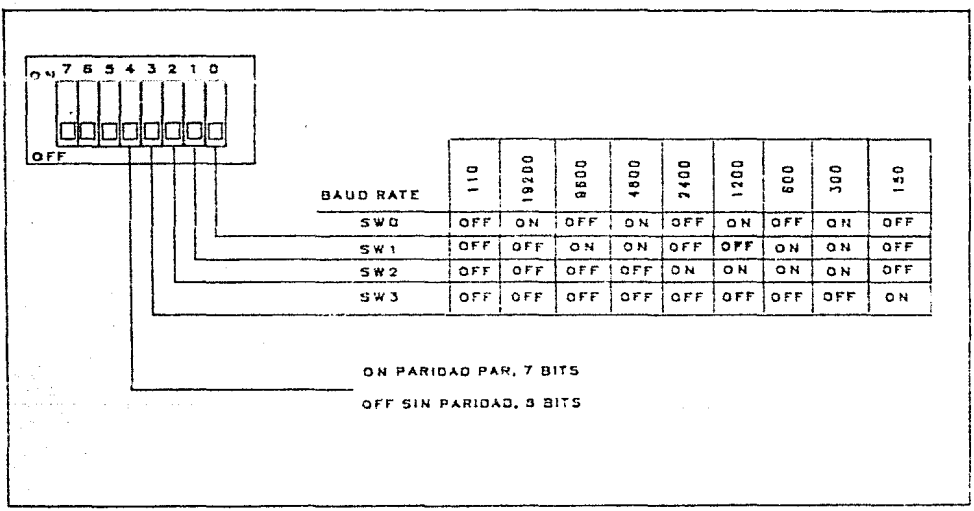


FIGURA (4.3)

3.- Configurar los switches de opción del SIS Z8 para determinar la velocidad de transmisión y la opción de paridad. Los 3 elementos interconectados: SIS Z8, Terminal y Computadora Anfitriona, deben trabajar acordemente, esto es, misma velocidad de transmisión, misma longitud e igual paridad, si esta es seleccionada. La Figura (4.4) muestra como las opciones son codificadas en el Banco de Switches.



BAUD RATE	110	19200	9600	4800	2400	1200	600	300	150
SW0	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF
SW1	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF
SW2	OFF	OFF	OFF	OFF	ON	ON	ON	ON	OFF
SW3	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

FIGURA (4.4)

4.- Encender las fuentes externas del SIS Z8 y el resto del equipo empleado. No es necesario oprimir el botón de RESET del SIS Z8, ya que este tiene incorporado un RESET automático al encender la Fuente de 5 V.

COMANDOS DEL SCAN Z8

En esta sección se hará una descripción funcional de los 17 comandos del Z8. Esta descripción es a nivel usuario y no contempla la explicación del firmware del SIS Z8.

- CONVENCIONES

Cada comando es tratado por separado. Las letras remarcadas en doble ancho seguidas de un CR, es lo único que es necesario introducir para iniciar la ejecución del comando.

[ ]	Lo que queda encerrado entre paréntesis cuadrados es opcional.
d_4	Indica un dato de extensión máxima de 4 dígitos hexadecimales.
n_2	Indica un dato de extensión máxima de 2 dígitos hexadecimales.
,	Cuando existe una coma entre 2 datos debe ser introducida literalmente.
	El nombre dentro de este símbolo indica una tecla o secuencia de teclas de la terminal que deben ser oprimidas.

Todos los datos numéricos deben ser introducidos en hexadecimal, por lo que el SCAN Z8 no tiene rutinas de conversión de decimal a hexadecimal.

Los datos alfanuméricos pueden ser introducidos en minúsculas o mayúsculas indistintamente.

La descripción supone una conexión, incluida en una computadora anfitriona, aunque a excepción de los comandos MST y TRM, esto no tiene ningún efecto sobre los demás comandos.

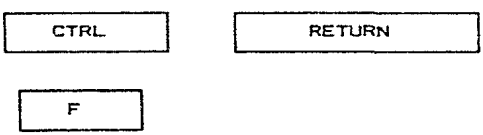
- OPERACION DEL SIS Z8

Asumiendo que la conexión ha sido realizada correctamente y que la Terminal, el SIS Z8 y la Computadora Anfitriona han sido encendidas, la operación sería la siguiente.

Al encender el SIS Z8, éste es completamente transparente a la operación normal Terminal Computadora. El monitor del SCAN Z8 intercepta cualquier caracter enviado por la terminal, en espera de la clave de acceso. La clave de acceso le indica, cuando el usuario seleciona el inicio de la sesión al SCAN Z8. Mientras esta clave no sea enviada, el SCAN Z8 repetirá la información y la envía a la computadora.

El usuario puede trabajar con la computadora como si el SIS Z8 no estuviera ahí.

Para accesar al SIS Z8, el usuario introduce la clave de acceso que consiste de la siguiente secuencia de caracteres :



Las teclas CTRL y F deben de ser oprimidas simultáneamente. Esta secuencia no será enviada a la computadora anfitriona. Al recibirla, el SIS Z8 cambia a su modo de operación por donde la transmisión es exclusivamente con la terminal. En la terminal debe aparecer el siguiente mensaje:

```
SIS Z8 LISTO PARA RECIBIR INSTRUCCIONES
*
```

A partir de este momento empieza la sesión con el SIS Z8. Usted trabajará exclusivamente con éste y la terminal, excepto con los comandos TMR y MST que involucran a la computadora anfitriona.

El (*) es el apuntador (PROMPT) indicador de que el SCAN Z8 puede aceptar nuevas instrucciones, inmediatamente a éste puede introducirse cualquier comando. El conjunto de instrucciones del SCAN Z8 consta de 16 comandos que pueden clasificarse de la siguiente forma:

CONTROL Y TRANSFERENCIA DE MEMORIA	CONTROL DEL ARCHIVO DE REGISTROS	AUTOPRUEBAS	CONTROL DEL SIS Z8	DEPURADOR
DM	MR	MA	MST	GO
SM	DR	PM		CALL
MV				T
TMR				AT
				DB
				BP
				BX

El uso de los comandos se reserva para el final de este capítulo.

- REGLAS DE OPERACION

Existen varias reglas que el usuario debe observar para el correcto funcionamiento del SIS Z8:

1.- No debe de tratar de desplegarse localidades de memoria en direcciones superiores a la F000H, ya que ésto involucra la lectura de una área que está destinada al puerto de control interno del SIS Z8. Si se intenta, el SIS Z8 quedará bloqueado y solamente mediante un RESET maestro regresará a su funcionamiento normal.

2.- A los pins del Z8: P27 y P32, no debe alambrarse ningún circuito externo, a menos que el usuario comprenda perfectamente la función que estos pins desempeñan en el SIS Z8.

Cuando el usuario desee depurar un programa es necesario cumplir con las siguientes reglas:

3.- Si el programa de prueba va a ser depurado empleando algunas de las opciones del SCAN Z8 para control de secuencia (Trace o Break Points), debe cerciorarse de no deshabilitar en su programa, ya sea individual o globalmente, la petición de interrupción IRQ0. El efecto de deshabilitar IRQ0, es que sencillamente no habrá rompimiento de la ejecución del programa.

4.- Debido a que varios de los registros de control del Z8 no pueden ser leídos y otros aunque pueden leerse, el valor obtenido difiere del que fué programado, es por lo tanto necesario, que al depurar un programa empleando la instrucción GO, él mismo se encargue de respaldar los registros de control del F1 al FB en el grupo de registros 1on. Por ejemplo, si se programa el registro F1 debe inmediatamente respaldarse en el registro 11H. Si no se respalda, al ocurrir un rompimiento debido a un Break Point o a Trace, la programación original se perderá.

En cuanto a la depuración de programas, existen ciertas reglas que dependen del comando empleado para iniciar la ejecución del programa de prueba, ya sea GO o CALL. Estas serán tratadas detenidamente en su oportunidad en sus comandos respectivos.

5.- Comprender el mapa de memoria de la Figura(5.8) del Capítulo 5, para saber que áreas están disponibles para el usuario, para poder hacer un uso adecuado de ellas y así obtener un máximo provecho. En particular existen 3 puntos importantes:

a) El área disponible para el usuario queda limitada a las siguientes localidades :

	MEMORIA DE PROGRAMA	MEMORIA DE DATOS
1	0200H - 0FFFH	3200H - 3FFFH 4000H - 4FFFH
2	1000H - 2FFFH	
3	5000H - 7FFFH	1000H - 2FFFH

b) De la tabla anterior la RAM #1 simula a la Memoria Interna, donde quedará localizado el programa final del usuario. El Depurador simulará la dirección 000CH en la 020CH. Las localidades entre la 0200H y la 020BH, quedan reservadas para el uso de Vectores de Interrupción y Programa de Prueba, descartando al vector correspondiente a la atención de interrupción IRQ0.
La rutina del SCAN Z8 busca en estas localidades los vectores de interrupción cuando es necesario. Esta RAM en Memoria de Programa sólo puede ser leida entre las localidades 0200H y 0FFFH.

Es posible hacer un despliegue de memoria de estas localidades, pero no puede hacerse una sustitución directamente en ellas. Para modificarlas es necesario accesarlas en sus equivalentes en Memoria de Datos, en las direcciones 3200H a 3FFFH. Si se hace un despliegue de memoria de cualquiera de estas 2 áreas usando el comando DM y DMD, se observará que el contenido es idéntico. Por lo anterior, debe tenerse cuidado al modificar alguna localidad del área de datos asignada a esta RAM, ya que esto resulta en una modificación en la localidad equivalente en Memoria de Programa.

Las localidades en Memoria de Datos después de la dirección 4000H hasta la 4FFFH, pueden ser empleadas libremente por el usuario.

c) Como en el caso anterior, la RAM #3 tiene asignadas 2 áreas que pueden ser accesadas indistintamente. El usuario puede hacer el uso que desee de las zonas delimitadas por las direcciones indicadas en la tabla anterior. Sin embargo, debe recordarse que existe un mapeo 1 a 1 entre las áreas de Memoria de Programa y su equivalente en Memoria de Datos. Al modificar, por ejemplo, la dirección 214CH de Memoria de Datos, simultaméamente se está alterando el contenido de la dirección 614CH de Memoria de Programa.

Lo anterior, aunque algo complicado, de comprenderse bien le da una gran flexibilidad al SIS Z8, y usándose correctamente, el usuario dispone de un total de 12288 bytes de Memoria de Datos y 19968 bytes para Memoria de Programa, que pueden adecuarse a las necesidades específicas de una aplicación determinada.

- MENSAJES DE ERROR

El SCAN Z8 cuenta con una serie de mensajes para indicarle al usuario que se ha cometido un error en su manejo.

Los mensajes de error son 7 y se explican a continuación:

Mensaje de Error: MESS

"Dato no Numérico"

Este mensaje es utilizado por la rutina ASCHEX, que es la que convierte 2 dígitos en Código ASCII a Código Hexadecimal. Indica que se ha recibido un dato no numérico.

Mensaje de Error: LET

"Dirección Inválida"

Este mensaje es utilizado por la rutina RECDIR, que es la que reconoce una dirección determinada. Indica que se ha tecleado una dirección fuera del rango de memoria permitido para el usuario.

Mensaje de Error: SOLOESC

"Registro de Solo Escritura"

Este mensaje es utilizado por las rutina del comando DR, que es la que despliega los registros del Z8. Indica que se está pidiendo desplegar un registro de control de solo escritura, y por lo tanto, no puede ser desplegado.

Mensaje de Error: REGINV

"Número de Registro Inválido"

Este mensaje es utilizado por la rutina NUMOK, la cual

verifica que un número de registro dado esté dentro del rango correcto, que son los registros del 00H-7EH y los registros F0H-FFH. Es usado cuando el número de registro dado no queda dentro de los grupos mencionados.

Mensaje de Error: OPINVA

"Opción Inválida de Movimiento de Memoria"

Este mensaje es utilizado por la Rutina MV, que es la encargada de mover bloques de memoria, con las opciones siguientes:

- De Memoria de Datos a Memoria de Datos.
- De Memoria de Programa a Memoria de Programa.
- De Memoria de Programa a Memoria de Datos.
- De Memoria de Datos a Memoria de Programa.

Este mensaje es utilizado cuando se trata de usar una opción diferente a las cuatro mencionadas anteriormente.

Mensaje de Error: BXEROR

"No existe un Break Point en esa localidad"

Este mensaje es utilizado en la rutina BP, que se encarga de colocar Break Points en una localidad de memoria determinada. Este ocurre cuando el usuario quiere borrar un Break Point de una localidad de memoria que no ha sido previamente definida como punto prueba.

Mensaje de Error: TRERR

"Errores"

Este mensaje es utilizado en la rutina TRM. Indica un error en la transferencia de datos.

- DESCRIPCION DE LOS COMANDOS:

A continuación se presentaran en las siguientes páginas, una descripción detallada de la función y operación de los comandos.

COMANDO DM

DESPLIEGUE DE MEMORIA.

FORMATO: DM[d] d_1,d_2

d_1.- Dirección Inicial
d_2.- Dirección Final

FUNCION:

Despliega la memoria del SIS Z8 contenida entre las direcciones d_1 y d_2. Si la opción es empleada, las direcciones inicial y final corresponden a Memoria de Datos; d_1 es la dirección inicial y d_2 la final, ambas deben separarse entre sí por una coma (,). La dirección inicial puede suprimirse, en cuyo caso se toma como dirección inicial la 0000H.

La logitud máxima para expresar una dirección, es de 4 dígitos. Si una dirección es mayor a esta longitud, el SCAN Z8 envia un mensaje de error.

Observación: Nunca desplegar memoria entre las direcciones F000H a la FFFFH, en caso contrario el SIS Z8 se bloqueará y será necesario un RESET maestro.

Ejemplo: DM 00,1000

Esta instrucción despliega memoria de programa a partir de la dirección 0000H hasta la dirección 1000H. El despliegue de memoria en la terminal consiste de 3 campos:

```
*DM 00,1000

0000   44 4F 44 4F 00 01 02 03 04 05 06 07 08 09 0A 0B   DODO________
0010   08 10 1A 53 59 53 5A 38 20 1F 1E 1D 1C 1B 1A 19   ...SYSZ8....
0020   01 02 03 55 04 4E 05 41 06 4D 18 1A 1C 1B 01 02   ...U.N.A.M.....
```

En el primer campo se indica la dirección del primer digito que será desplegado en el siguiente campo. El segundo campo consta de 16 datos en hexadecimal que corresponden a la primera y las 15 subsecuentes direcciones, a partir de la indicada en el primer campo. El tercer campo consta de 16 simbolos, que son los equivalentes ASCII de los 16 datos desplegados en el segundo campo. En el caso de que un dato no tenga un simbolo ASCII desplegable, será sustituido por un punto. El despliegue de memoria puede ser suspendido temporalmente oprimiendo en la terminal las teclas CTRL S simultáneamente. El desplieque continúa oprimiendo cualquier tecla. También es posible abortar el despliegue oprimiendo la tecla ESC.

COMANDO SM

SUSTITUCION DE MEMORIA.

FORMATO: SM[D] d_1

d_1.- Dirección de memoria a sustituirse.

FUNCION:

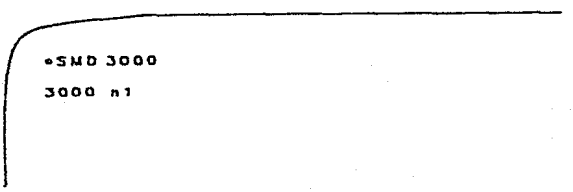
Sustituye memoria. Si la opción [D] es empleada, d_1 indica una dirección de Memoria de Datos. La opción [D] debe introducirse sin espacio inmediatamente después de SM.

d_1 debe ser de extensión máxima de 4 digitos. si esta longitud es excedida, el SCAN Z8 envia un mensaje de error.

Observación: Nunca tratar de sustituir memoria entre las direcciones F000H a la FFFFH para evitar el bloqueo del SIS Z8.

Ejemplo: SMD 3000

Esta instrucción le indica al SCAN Z8, que debe sustituir el contenido de la dirección 3000H de la memoria de datos. El SCAN Z8 responderá de la siguiente manera:



Donde n_1 es el contenido actual de la localidad 3000H de Memoria de Datos. A continuación se debe introducir el nuevo contenido de esa localidad seguido de un RETURN. El nuevo dato

debe ser un número en hexadecimal de extensión máxima de 2 digitos. Si el dato introducido es aceptado, será sustituido y se desplegará a continuación la siguiente localidad, junto con su contenido actual. Si se desea sustituir también esta localidad se debe repetir el paso anterior. Si se envía un RETURN sin ningún número previo, el contenido de la localidad no será alterado y se desplegará la siguiente localidad. Para abortar el comando, se debe enviar un ESC.

COMANDO MV

MOVIMIENTO DE BLOQUES DE MEMORIA

FORMATO: MV[O] d_1,d_2,d_3

P: Transferencia Programa a Programa.

D: Transferencia Datos a Datos.

E: Transferencia Programa a Datos.

I: Transferencia Datos a Programa.

d_1: Dirección Inicial del Bloque a Transferir.

d_2: Dirección Final del Bloque a Transferir.

d_3: Dirección Inicial del Destino.

FUNCION:

Transfiere un bloque de memoria, de una posición a otra, en memoria. El bloque queda comprendido entre las direcciones d_1 y d_2. La nueva localización comienza a partir de d_3. Los bloques pueden transferirse entre el espacio de Memoria de Programa y Memoria de Datos, en las 4 combinaciones posibles, eligiendo la opción adecuadamente de la tabla anterior. Si el movimiento se ha hecho correctamente, el SCAN Z8 responde simplemente con un PROMPT.

Ejemplo: MVI 2000,2500,1000

La instrucción anterior mueve el bloque en memoria de datos, comprendido entre la dirección 2000H hasta la 2500H. El destino de este bloque comienza a partir de la dirección 1000H de Memoria de Programa.

COMANDO MST

CAMBIA CONTROL A COMPUTADORA MAESTRA

FORMATO: MST

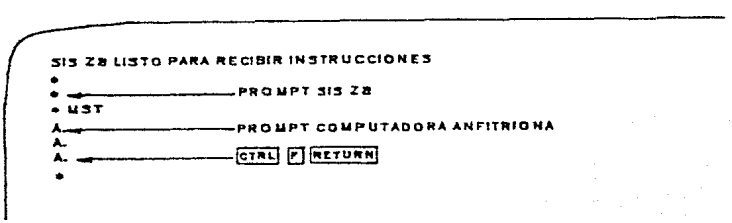
FUNCION:

Configura los puertos serie del SIS Z8 en el Modo 3 de operación, e introduce al SCAN Z8 en un ciclo de espera de la clave de retorno al monitor. La función de éste, es que el usuario pueda operar a la computadora anfitriona anulando temporalmente las funciones del SIS Z8. La clave de retorno es idéntica a la clave de acceso:

CTRL	F	RETURN

Donde las teclas CTRL y F deben oprimirse simultáneamente durante el intervalo del envio del comando MST y la clave de retorno, el SIS Z8 será transparente a la operación normal de la computadora maestra, y todo lo que se despliegue en la terminal dependerá de esta última. Después de la clave de retorno, el SIS Z8 se interpone a la transmisión Computadora Anfitriona a Terminal, y regresa la operación normal del SCAN Z8.

Ejemplo:



COMANDO TRM

TRANSFERENCIA DE MEMORIA

FORMATO: TRM

FUNCION:

Este comando transfiere un archivo en formato INTEL, generado por el ensamblador del Z9, de la Computadora Anfitriona a la Memoria del SIS Z8.

DESCRIPCION:

El ensamblador del Z8 deposita el código objeto en un archivo con formato INTEL, en un disco. Para transferir este código a la memoria del SIS Z8 se usa el comando TRM y un programa en la computadora anfitriona que puede ser codificado en BASIC. El programa que presentaremos más adelante, está hecho para el lenguaje BASIC estructurado de CROMEMCO. Sin embargo, si el usuario desea emplear otro programa o la computadora anfitriona no es una CROMEMCO y este programa no es trasportable, es necesario que el usuario comprenda el formato INTEL, así como la operación del comando TRM, por lo cual se presenta a continuación una breve descripción de ambos.

FORMATO INTEL:

En esta sección se supone que el usuario tiene ya conocimientos elementales sobre ensambladores. En caso contrario puede leerse previamente el apéndice dedicado a estos.

En general los ensambladores tienen 2 formas de presentar el código objeto del programa ensamblado:

La primera de ellas presenta el Código Máquina en forma absoluta, directamente almacenado en memoria y preparado para ejecutarse. En forma absoluta, puede guardarse en un archivo en varios formatos, uno de ellos es el formato INTEL.

La segunda forma es el Código Reubicable. El cual para poder ser ejecutado debe ser procesado previamente por un Editor de Enlaces. Como esta forma no es empleada por el SIS Z8, no será estudiada.

Las principales características de la codificación en formato INTEL son las siguientes:

1.- Utiliza el código ASCII para codificar la información.

2.- Incluye direcciones de carga de los datos, es decir, las localidades de memoria donde se espera que el código sea ubicado.

3.- Incluye bytes de ubicación para la detección y corrección de errores.

Si un archivo en formato INTEL es desplegado en pantalla puede entenderse fácilmente (ver Figura (4.5)), ya que está completamente codificado en ASCII.

```
:080000 00 23 43 54 6F 7C 42 56 32  93
:080008 00 32 43 55 00 00 32 1C 23  B5
:080010 00 FF FF 3C 4C 01 00 01 00  60
```

FIGURA (4.5)

Analizando la figura anterior, se desprende que este tipo de archivos se divide en registros. Cada línea representa un registro, el cual comienza con dos puntos (:), el número hexadecimal de 2 dígitos a continuación, especifica el número de bytes de datos que contiene el registro. Los 4 dígitos siguientes indican la dirección de carga, seguidoS por un campo de 2 dígitos que indica el tipo de archivo, el cual casi nunca es usuado. El siguiente campo contiene los datos codificados en hexadecimal. Este campo está formado con tantos datos como indica el primer número de registro después de los dos puntos. El último campo es un byte destinado a la detección y correción de errores.

La suma del valor de todos los bytes del registro mas el valor del byte de detección de errores, eliminando acarreos, debe ser igual a cero.

En cuanto al comando TRM, éste al ser llamado, su primer acción es modificar los puertos serie del SIS Z8, de tal manera que el Z8612 recibe datos e instrucciones de la computadora y los retransmite a la terminal, mientras que los datos provenientes de la terminal pasan directamente a la computadora. Inmediatamente el SIS Z8 parece invisible como al encender, por lo cual el usuario puede trabajar con la computadora ignorando al SIS Z8. El programa en BASIC propuesto abre el archivo en formato INTEL, lo lee y lo envia a la terminal, tal como se desplegaria con un editor de línea.

TRM intercepta el código, siempre y cuando el programa en BASIC le indique el momento en que comienza la lectura del archivo. Esto se hace enviando el código ASCII ACK (06H o CTRL-F), a partir del cual TRM se encarga de que el SIS Z8 tenga una comunicación exclusivamente con la computadora.

Al finalizar la lectura, el programa en BASIC cierra el archivo y envia nuevamente el código ACK indicándole a TRM la finalización de la transmisión. Ahora TRM calcula los errores, si es que hubo y la siguiente dirección que queda libre después de colocar el código en las localidades indicadas. Finalmente TRM se encarga de reprogramar los puertos serie y reestablecer la comunicación con la terminal, de manera que el estado del SIS Z8 queda como estaba antes del llamado de transferencia de memoria.

El programa en BASIC que se recomienda es el siguiente:

```
 10 ON ESC GOTO 140
 20 DIM CODIGO$(27)
 30 INPUT "DAME EL NOMBRE DEL ARCHIVO [ NOMBRE.HEX ].",NOM$
 40 OPEN\1\NOM$
 50 OUT 1,6
 60 INPUT \1\CODIGO$
 70     FOR BYTE=1 TO 26
 80     BYT$=CODIGO$(BYTE,BYTE)
 90     PRINT BYTE$;
 00     IF BYTE$=" " THEN 140
110     NEXT BYTE
120 PRINT
130 GOTO 60
140 CLOSE
150 OUT 1,6
160 END
```

Ejemplo:

```
*TRM
A.SBASIC

CROMEMCO 32K STRUCTURED BASIC

>>LOAD "TRM.SAV"
>>RUN
DAME EL NOMBRE DEL ARCHIVO .... PRUEBA.HEX
SIGUIENTE LOCALIDAD
0240
ERRORES
00
*
```

Este es un ejemplo en el cual la transferencia de memoria fué correcta y sin errores.

NOTA: El archivo debe tener siempre un extensión .HEX.

En caso de existir algún error es recomendable cambiar de disco c reensamblar el programa prueba .ASM para generar nuevamente el archivo Prueba.HEX.

COMANDOS DE AUTOPRUEBA MA Y PM.

Estos comandos son útiles cuando se sospecha de un mal funcionamiento del SIS Z8. El comando MA comprueba el buen funcionamiento de toda la lógica asociada a los puertos serie, mientras que PM es una rutina que hace una prueba de memoria de acceso aleatorio. Esta última consiste de 2 partes: la prueba de lectura de verificación después de escritura, y la prueba de lectura.

COMANDO MA

ENVIO DE CARACTERES ASCII IMPRIMIBLES

FORMATO: MA

FUNCION:

Envia todos los caracteres ASCII imprimibles a la terminal. MA es lo único que es necesario introducir para ejecutar esta prueba.

Ejemplo:

```
-MA

!"#$%&'()*+,-./0123456789
:;<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_`abcde
fghijklmnopqrstuvwxyz{|
}~
```

COMANDO PM

PRUEBA DE MEMORIA DE ACCESO ALEATORIO

FORMATO: PM

FUNCION:

PM es lo único que es necesario introducir para ejecutar la prueba de memoria. Este comando inicialmente ejecuta la prueba de lectura para verificación después de escritura, que se divide en 4 sectores:

SECTOR 1: Memoria RAM de Programa a partir de la dirección 1000H hasta la 2FFH.

SECTOR 2: Memoria RAM de Programa a partir de la dirección 5000H hasta la 6FFH.

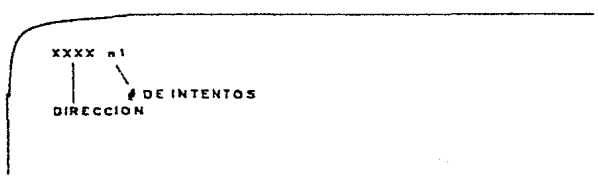
SECTOR 3: Memoria RAM de Datos a partir de la dirección 1000H hasta la 2FFH.

SECTOR 4: Memoria RAM de Datos a partir de la dirección 3000H hasta la 4FFH.

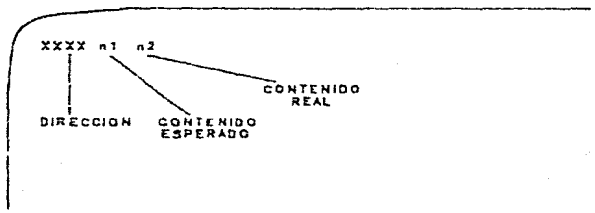
El comando PM envia un mensaje cada vez que comienza la prueba en determinado sector. Al finalizar esta primera prueba, inicia la prueba de lectura, la cual también se divide en los 4 sectores anteriores que son enunciados en forma similar a la anterior.

Si la memoria se encuentra en buen estado, solamente debe aparecer en la terminal los mensajes de inicio de pruebas y de inicio de sectores como se muestra en el ejemplo de esta misma sección.

Pueden ocurrir 2 tipos de errores. El primero es error en reintentos y se despliega de la manera siguiente:



Donde la dirección es la localidad donde ocurre la falla y el número de reintentos es el número de veces que tuvo que leerse esa localidad en particular hasta obtener el contenido esperado. PM ejecuta hasta 16 reintentos, que en caso de ser fallidos envia el segundo tipo de error que se despliega de la manera siguiente:



Donde la dirección indica la localidad donde ocurre la falla, n_1 es el dato esperado de esa localidad y n_2 es el dato real que fué leido.

Ejemplo:

```
*PM
PRUEBA DE VERIFICACION DESPUES DE ESCRITURA

SECTOR #1000H P
SECTOR #5000H P
SECTOR #1000H O
SECTOR #3000H O

PRUEBA DE LECTURA

SECTOR #1000H P
SECTOR #5000H P
SECTOR #1000H O
SECTOR #3000H O

*
```

COMANDOS DEL DEPURADOR.

Las funciones del depurador son las siguientes:

1.- Permitir la ejecución de programas de prueba.

2.- Control de secuencia y flujo del programa de prueba con comandos tales como: Inserción de Break Points y Trace.

3.- Despliegue y Movimiento de registros de control y de propósito general, del Z8.

Los comandos asignados al depurador son ejecutables desde el monitor. Sin embargo, es necesario una breve introducción sobre la estructura del depurador para emplear cualquiera de ellos.

Existen 2 comandos distintos para iniciar la ejecución de un programa de prueba: GO y CALL. Existen diferencias sustanciales entre ambos, y por lo tanto, están destinados para programas con estructuras distintas. Cualquiera de ellos puede emplearse en conjunto con los comandos Trace y Break Point.

El comando CALL está designado como una simple llamada a una subrutina. Este comando no hace una reprogramación de los registros del Z8 escenciales para el funcionamiento del SIS Z8. Estos registros de control y su programación por el SIS Z8 se explican a continuación.

Registro P3M.

De este registro las siguientes funciones son indispensables para el buen funcionamiento del Monitor del SIS Z8.

BIT	VALOR	FUNCION
0	0	Puerto 2 Modo Drenaje Abierto
4 y 3	1 , 0	P33=Salida P34=DM
6	1	P30=Sin P37=Sout
7	-	Paridad(En función de Dip Switch

Registro P01M.

Este registro es absolutamente necesario para el funcionamiento del monitor, y cualquier alteración a él puede producir el bloqueo del SIS Z8, su programación es la siguiente:

BIT	VALOR	FUNCION
1 y 0	1 , 0	P00-P03 funcionan como A8-A11
2	1	El stack está en los Regs. de Prop. Gral.
4 y 3	1 , 0	P10-P17 trabajando como A0-A7
5	0	Ciclos de máquina normales
7 y 6	1 , 1	P04-P07 trabajando como A11-A15

Registro P2M.

El único bit esencial para el SIS Z8 de este registro es el bit 0 y es programado como salida. Bajo ninguna circunstancia el programa del usuario puede alterar, ni la programación, ni el valor del bit 0 del Puerto 2, ya que ésto produciría el bloqueo del SIS Z8.

Registros IMR e IPR.

Estos registros no son empleados por el monitor del SIS Z8. Sin embargo, el usuario debe recordar que para la ejecución de programas empleando la opción 3 o Break Point, no debe deshabilitar global o individualmente la petición de interrupción IRQ0, y es aconsejable programar IPR de tal forma que IRQ0 siempre tenga la máxima prioridad.

Registro SPL.

El stack del SIS Z8 es interno y este registro es programado inicialmente con un valor de 40H.

Registro SRP.

Los registros de trabajo del monitor del SIS Z8 quedan asignados mediante este registro al grupo 10H.

Registro TO.

El SIS Z8 aprovecha las facilidades que da el Z8 en el manejo de entrada de datos en forma asincrona serial para el funcionamiento de sus puertos seriales. Por esta razón el Z8 trabaja con un cristal de 7.3728 MHz. para derivar las frecuencias de muestreo (Baud Rate) más usuales empleando el Temporizador TO, este registro es programado en función de la opción que el usuario asigne mediante los Dip Switches.

Registro PREO.

El Preescalador del Temporizador TO es programado con un 3 y el bit 0 del mismo tiene el valor de 1 para que el temporizador trabaje en Modo Free Ronning.

Registro TRM.

De este registro solamente los bits 0 y 1 son empleados para cargar, habilitar e iniciar la cuenta del Temporizador 0. El Temporizador 1 es deshabilitado y Tout no es usada.

Registro STATUS.

Este registro tiene 2 bits de STATUS llamados banderas de usuario, los cuales son aprovechados por el SIS Z8 para recordar la combinación GO/CALL y TRACE/BREAK POINT, que el usuario asigna para la ejecución de un programa. Posteriormente el comando K interpreta estos 2 bits para continuar un programa que haya sido interrumpido.

Aunque no es aconsejable que un programa de prueba inicializado por el comando CALL, altere cualquiera de estos registros, es permisible siempre y cuando éste concluya regresando la programación original de los mismos.

Como el comando CALL asume que el programa de prueba tiene la estructura propia de una subrutina, una instrucción RET al final del programa regresa el control al monitor. Este comando es ideal para registros de control, o bien, para prueba de programas de expansión del SIS Z8.

Cuando se emplea el comando GO se eliminan todas las limitaciones del comando CALL. El programa en prueba puede reconfigurar completamente al Z8, siempre y cuando observe las reglas establecidas al principio de esta sección, las cuales quedarán aclaradas después de entender como se ejecuta un programa de prueba inicializado por este comando.

En primer lugar se introducirá el concepto de archivo de registros alterno. La diferencia escencial entre los comandos GO y CALL, es que el primero permite reprogramar todos los registros de control.

El registro de archivos alterno entra en juego cuando un programa es inicializado por GO simultáneamente con TRACE o BP. Cuando ésto es interrumpido, la rutina de atención común a TRACE y BP regresa los registros de control la programación necesaria para el monitor, perdiéndose la del programa de prueba.

Las rutinas de TRACE y BP asumen que el usuario ha respaldado sus registros de control en el grupo 10 (registros de propósito general) y antes de alterar más registros transfieren el archivo de registros a un bloque de localidades de Memoria de Datos Externa, el cual es designado Archivo de Registros Alterno. Para continuar la prueba del programa ya estando en el monitor, el usuario puede llamar el comando K y éste se encargará automáticamente de reinvertir el proceso, regresando al archivo de registros interno del Z8 la programación que quedó respaldada en el archivo alterno, de tal manera que el programa de prueba continúe como si no hubiera existido la interrupción.

En cuanto a los comandos Trace y Break Point, el SIS Z8 hace uso de la petición de interrupción IRQ0 para hacer el efecto deseado, pero por esta razón el usuario debe tener cuidado con el manejo de los registros IMR, IPR e IRQ, si es que desea que los rompimientos de la ejecución de su programa en las localidades pertinentes se lleven a cabo, ya que el deshabilitar interrupciones elimina todos los Break Points y el efecto del Trace hasta la localidad en que nuevamente se habiliten mediante la instrucción EI.

El comando GO automáticamente habilita interrupciones antes de pasar el control al programa de prueba.

Finalmente los comandos GO, CALL y K hacen uso de los 2 bits menos significativos del registro de STATUS, los cuales son llamados Banderas de Usuario. Estos bits le indican a estos comandos cuando deben trabajar en conjunto con las opciones Trace y Break Point.

COMANDO DR

DESPLIEGUE DE REGISTROS

FORMATO: DR[U] n_1

n_1: Número del registro a desplegarse.

DESCRIPCION:

Este comando despliega uno o varios de los registros del Z8. Si n_1 corresponde al primer registro de un grupo (P. ej. 10, corresponde al primer registro del grupo 1), se despliegan los 16 registros pertenecientes a ese grupo. Sin la opción [U], los datos desplegados corresponden al valor actual del registro. La opción [U] indica el despliegue del contenido de los registros del Archivo Alterno, almacenado en Memoria de Datos Externa y que corresponden al estado de los registros del Archivo Interno del Z8 al momento de ocurrir una interrupción del programa de prueba, por un Punto Prueba o Trace. El contenido es desplegado en hexadecimal. Al desplegar registros correspondientes al grupo de registros de control, debe recordarse que en éste existen registros de solo lectura y registros dobles, de tal manera que el despliegue de ellos puede no ser consistente.

Ejemplo:

```
*DR 25
R25H=43H
```

COMANDO MR

MOVIMIENTO DE REGISTROS

FORMATO: MR[U] n_1

n_1: Número de Registros a Modificarse.

DESCRIPCION:

Este comando modifica el contenido del registro indicado por n_1. Sin la opción [U] el registro es modificado directamente en el Archivo Interno. La opción [U] hace la modificación sobre el registro equivalente en el Archivo Alterno en Memoria de Datos Externa.

Este comando despliega el valor actual y espera la introducción del nuevo dato. Si no se introduce ningún dato o se envia el código escape (ESC) no ocurrirá ninguna modificación.

Ejemplo:

```
•DRU 15
 R15H=40H
•URU 15
 R15H=40H FF
•DRU 15
 R15H=FFH
```

INTERRUPCIONES POR TRACE Y PUNTOS PRUEBA.

El procedimiento para interrumpir el flujo de un programa de prueba, consiste en definir inicialmente cual de estas opciones se ejecutará mediante los comandos BP o T. El programa de prueba puede iniciarse con los comandos GO o CALL. Si no se definieron opciones, el programa se ejecutará libremente. En caso contrario ocurrirá una interrupción cuando el contador de programa llegue a la localidad correspondiente a un punto de prueba o después de ejecutar una instrucción completa en el caso de Trace. En cualquiera de las opciones el efecto es similar: Se interrumpe el programa de prueba y regresa el control al monitor del SCAN Z8, desplegándose automáticamente todo el archivo de registros y la dirección donde ocurrió la interrupción.

En el caso de que el programa de prueba sea llamado por el comando CALL, el archivo de registros que se despliega es el interno. Con GO será desplegado el archivo alterno.

Cuando ocurre una interrupción por un punto prueba, la rutina de atención borra automáticamente el punto prueba. De esta manera, la siguiente vez que el contador del programa pase por esa localidad, no ocurrirá la interrupción a menos que el usuario restablezca el punto prueba previamente.

La rutina de servicio a las interrupciones culmina en el monitor del SCAN Z8, el usuario puede continuar con la ejecución del programa de prueba con el comando K. Este determina automáticamente si se inicia el programa con el efecto de GO o con el efecto de CALL. Sin embargo, para poder hacer uso de este comando no debe hacer despliegues de memoria. Si después de la rutina de interrupción, el usuario despliega memoria, para continuar con el programa de prueba debe hacerlo mediante los comandos GO o CALL, con la siguiente dirección ejecutable de su programa.

COMANDO TRACE

ROMPIMIENTO DE PROGRAMA PASO A PASO

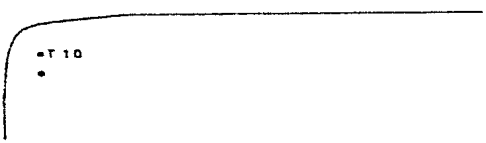
FORMATO: T n_1

DESCRIPCION:

Cuando se desee ejecutar el seguimiento de un programa de prueba, es necesario previamente a los comandos GO o CALL, indicarle al depurador que se desea esta opción.

n_1: Indica el número de instrucciones del programa de prueba que deben seguirse.

Ejemplo:



Con este comando, 16 instrucciones del programa de prueba serán ejecutadas, interrumpiéndose al inicio de la siguiente.

Nótese que Trace no inicia la ejecución del programa, solamente es indicativo para los comandos GO, CALL y K, de manera, que éstos programen la lógica externa para habilitar esta opción a partir de la primera instrucción de la rutina de prueba.

COMANDO AT

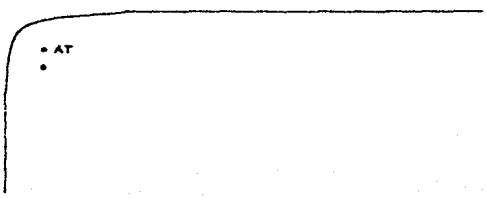
ANULA TRACE

FORMATO: AT

DESCRIPCION:

Elimina la opción AT borrando de las banderas de usuario del registro de STATUS, el bit correspondiente a esta opción.

Ejemplo:



COMANDO BP

BREAK POINTS

FORMATO: BP d_1

d_1: Dirección donde se desea colocar el Break Point.

FUNCION:

Este comando es usado para colocar un punto de ruptura en alguna localidad de memoria, el cual será permanente mientras no sea removido por la instrucción BX (Borrado de Break Point).

La rutina de Break Points utiliza una doble tabla de memoria de datos para almacenar la dirección (y su contenido) donde se ha colocado un Break Point.

Dentro de un programa se pueden colocar hasta ocho Break Points.

Este comando se utiliza tecleando BP y enseguida la dirección d_1, donde se desea colocar el Break Point, inmediatamente después se oprimirá la tecla CR y aparecerá en seguida el asterisco (prompt) del SIS Z8, indicando que el comando fué ejecutado correctamente.

COMANDO DB

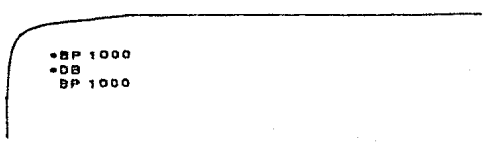
DESPLIEGUE DE LOCALIDADES CON BREAK POINTS

FORMATO: DB

DESCRIPCION:

Este comando despliega la tabla de Puntos Prueba (Break Points) introducidos por el comando BP. Indica solamente las direcciones en donde han sido colocados los puntos prueba.

Ejemplo:



COMANDO BX

BORRADO DE BREAK POINTS

FORMATO: BX[d_1]

d_1: Es la dirección especifica de donde se quiere eliminar un punto prueba.

DESCRIPCION:

La función de este comando es eliminar puntos prueba y limpiar la tabla de los mismos. Si no se indica la dirección especifica, borra todos los Puntos Prueba introducidos por BP. En caso contrario solo borra el indicado. En caso de no existir un punto prueba en esa dirección se genera un mensaje de error.

Ejemplo:

```
*BP 1000
*DB
 BP 1000
*BX 1000
*DB
*
```

COMANDO CALL

LLAMADA SIMPLE A UN PROGRAMA DE PRUEBA

FORMATO: CALL d_1

d_1: Es la dirección donde comienza la rutina de prueba.

DESCRIPCION:

Esta rutina salta a la dirección de Memoria de Programa donde comienza la rutina de prueba. Esta es la forma màs sencilla de ejecutar pruebas a programas, sin embargo, tiene sus limitaciones: en primer lugar no introduce el archivo de registros alterno, ni modifica paràmetros del SIS Z8. Està diseñada para probar subrutinas y, por lo tanto, es posible regresar a ROOT mediante una instrucción RET. Es excelente para expansiones del SIS Z8. Y sin embargo, es posible ejecutar Puntos Prueba o la opción Trace definièndola previamente.

Ejemplo:

En este ejemplo se habilita previamente el salto a la rutina de prueba con la opción de Trace. Después de la instrucción CALL, iniciarà la ejecución de la rutina que comienza en la dirección 20CH, interrumpièndose 16 veces la secuencia debido al comando Trace.

NOTA: No existe limitación en cuanto a la localidad donde comience la rutina de prueba, cuando èsta es llamada mediante el comando CALL, siempre y cuando se encuentre en àreas de memoria disponibles para el usuario.

COMANDO GO

EJECUCION DE UN PROGRAMA DE PRUEBA

FORMATO: GO[I] d_1

d_1: Dirección de inicio del programa de prueba.

DESCRIPCION:

Inicia la ejecución de un programa de prueba: introduce los registros del archivo alterno; deshabilita los buses y habilita interfaz de los Puertos 0 y 1; cambia el modo de los puertos seriales al modo 5 y deshabilita DM.

Si la opción [I] es empleada, el archivo de registros alterno es inicializado. Al entrar al programa de prueba, automàticamente este archivo sustituye al archivo de registros original del SIS Z8, los registros de control quedan programados en forma idèntica a un RESET maestro.

Al ser introducido el archivo alterno con el comando GO, la configuración del Z8 es modificada completamente. Podemos enumerar las funciones del comando GO de la manera siguiente:

1.-Determina si la opcion I es indicada. Si lo es, transfiere los registros del Archivo de Inicialización al Archivo Alterno.

2.- Entra a la rutina común, donde se salva la dirección de GO y se reconoce la dirección.

3.- Determina si la opción Trace ha sido habilitada. Habilita Trace o Break Point según sea el caso.

4.- Altera los registros de control del archivo alterno para evitar bloqueo del SIS Z8.

5.- Modifica el Stack Pointer.

6.- Modifica el Stack.

7.- Transfiere registros del Archivo Alterno al Archivo del Z8 a excepción de los registros de Control que quedan guardados en el Grupo 1.

8.- Deshabilita Interrupciones.

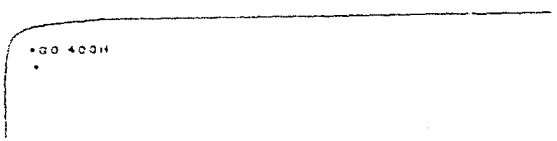
9.- Habilita Memoria de Datos.

10.- Salta a la rutina de entrada.

OBSERVACION:

El programa de Prueba debe comenzar en la Memoria Interna, por lo tanto, d_1 debe ser menor a 1000H y mayor que 200H. Si se van a utilizar interrupciones, la dirección inicial después de un RESET del Z8(000CH), se mapea como la dirección 02CH, y entre la dirección 0200H-020BH, se deben introducir los vectores de interrupción.

Ejemplo:



Esta instrucción inicia la ejecución del Programa de Prueba a partir de la dirección 400H.

COMANDO K

CONTINUA

FORMATO: K

DESCRIPCION:

Este comando tiene la función de continuar con la ejecución de cualquiera de los comandos siguientes: DM, CALL y GO. La acción del comando K es sobre el último de cualquiera de estos 3 comandos que haya sido ejecutado.

Si el último coamando en ejecutarse es DM, se desplegarán 256 localidades de memoria, a partir de la última que haya sido desplegada por DM.

En el caso de GO y CALL, el comando K reinicia la ejecución del programa de prueba, a partir de la última localidad previa a la interrupción por un Punto Prueba por Trace, es decir, que es equivalente a repetir el comando GO/CALL a la última dirección ejecutada por el programa de prueba.

CAPITULO V

HARDWARE DEL SIS Z8

En el capitulo anterior se presentaron, un poco a manera de manual de usuario, las reglas de operación y los comandos que puede ejecutar el SIS Z8. Hasta el momento el lector puede responder a las preguntas ¿ cómo se debe trabajar con el SIS Z8 ? y ¿ Que puede hacer el SIS Z8 ? En el presente capitulo y el en siguiente trataremos de dar respuesta a la pregunta ¿ Cómo funciona el SIS Z8 ? En este capitulo explicaremos el funcionamiento de la circuiteria (hardware), mientras que el siguiente está destinado al programa monitor (software) del SIS Z8.

Es recomendable que el lector consulte los primeros capitulos previos, relacionados con la arquitectura y conjunto de instrucciones del Z8, antes de comenzar a leer el presente.

GENERALIDADES DEL SIS Z8

Con el fin de facilitar la comprensión de la circuiteria del SIS Z8, haremos una división del mismo en cinco bloques funcionales cuyos diagramas respectivos se indican a continuación:

1.- Z8612, interface Puertos 0 y 1, buses y lógica de selección. Diagrama #1
2.- Memorias ROM. Diagrama #2
3.- Memorias RAM. Diagrama #3
4.- Puertos serie, lógica de control. Diagrama #4
5.- Lógica de trace y puntos prueba (break points). Diagrama #5

Antes de iniciar la revisión de los diagramas, es necesario explicar que han sido elaborados en función del prototipo de

laboratorio, consistente en dos tarjetas unidas por cuatro cables de 16 vias cada uno. La primera tarjeta es un circuito impreso (ver fig 5.1), que contiene únicamente al Z8612 y los cuatro conectores de los cables de unión . La segunda tarjeta, que de ahora en adelante llamaremos tarjeta principal, contiene al resto de los circuitos integrados. Los Diagramas 1 a 5 representan exactamente el alambrado de la tarjeta principal, por ésto en el diagrama 1 no se encuentra representado el Z8612, en su lugar se muestra el otro extremo de los cables de unión. Los cables de unión tienen la función de enviar señales entre el Z8612 y la tarjeta principal. La distribución de los circuitos en ambas tarjetas se muestra en la Figura (5.2). Los conectores de los cables de unión del lado de la tarjeta principal serán identificados por Cn2-1 a Cn2-4, mientras que los del extremo opuesto se reconocen por Jn-1 a Jn-4. Cada cable tiene en un extremo un conector Cn2-i y en el otro el correspondiente Jn-i.

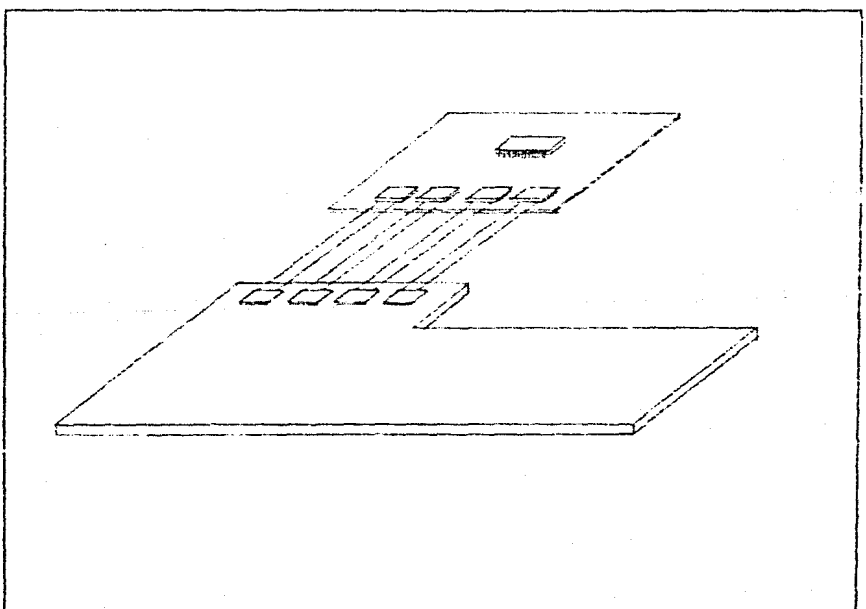
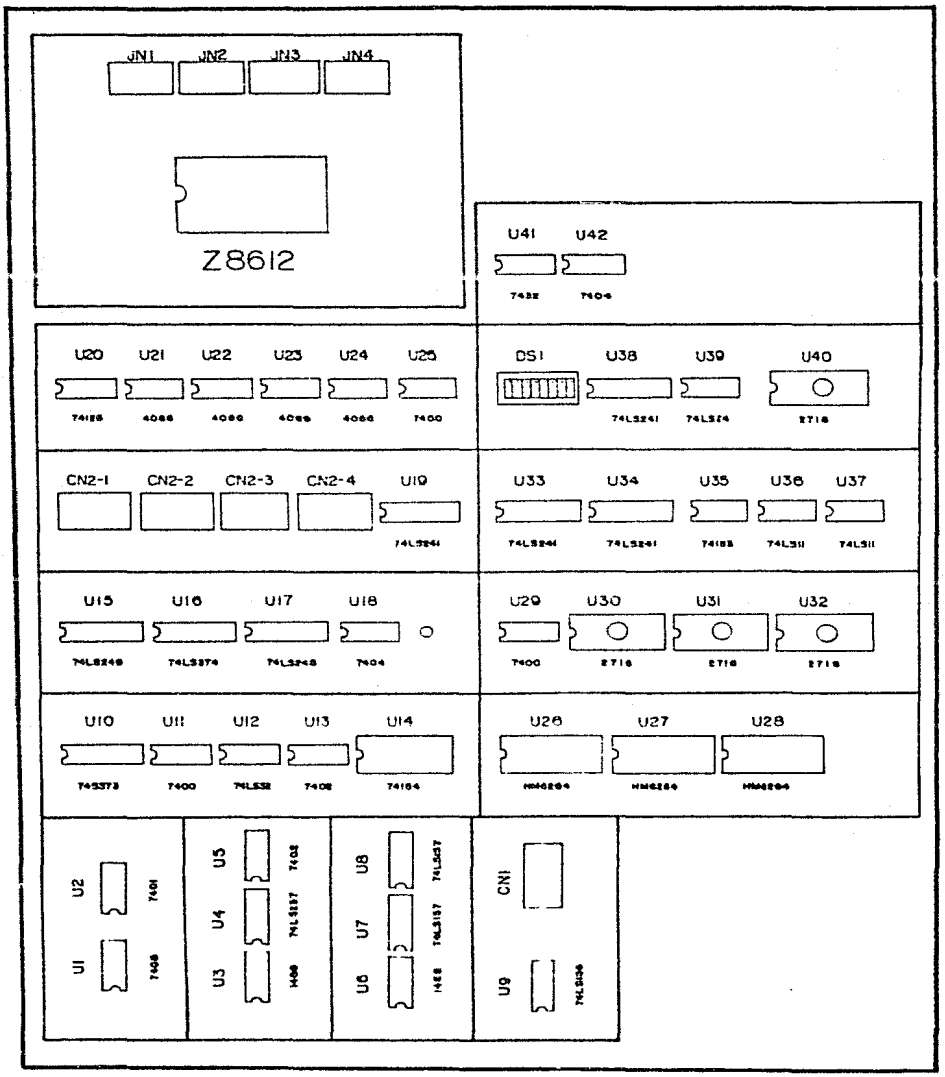


FIGURA (5.1)


JN1
JN2
JN3
JN4
Z8612
U41
U42
7432
7404
U20
U21
U22
U23
U24
U25
74155
4066
4066
4066
4066
7400
DS1
U38
U39
U40
74LS241
74LS74
2716
CN2-1
CN2-2
CN2-3
CN2-4
U19
74LS241
U33
U34
U35
U36
U37
74LS241
74LS241
74155
74LS11
74LS11
U15
U16
U17
U18
74LS240
74LS374
74LS245
7404
U29
U30
U31
U32
7400
2716
2716
2716
U10
U11
U12
U13
U14
74S373
7400
74LS32
7408
74164
U26
U27
U28
HM6264
HM6264
HM6264
U2
U1
7401
7408
U5
U4
U3
7402
74LS257
1488
U8
U7
U6
74LS257
74LS157
1488
CN1
U9
74LS240

El circuito integrado en la tarjeta impresa, el Z8612, como lo hemos llamado hasta ahora, es el corazón del SIS Z8. Este circuito integrado es la versión de 64 pins del Z8 y fué diseñado como un dispositivo para desarrollo de nuevos sistemas.

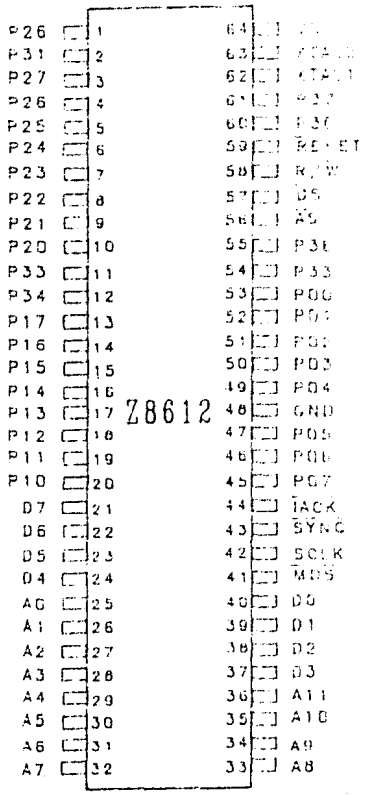
Existen algunas diferencias entre la versión normal del Z8 y el Z8612; la principal es que este último carece de ROM interna y necesita de memoria externa para simularla. Por èsto, el Z8612 tiene un bus de datos y direcciones adicionales para sustituir la memoria interna. La Figura (5.3) muestra la asignación de señales a sus respectivos pins del Z8612. Las siguientes señales difieren de las ya conocidas:

SA0-SA11	Es un bus de direcciones para seleccionar los primeros 4096 bytes de memoria de programa, esto es, la memoria de programa que en la versión normal del Z8 es interna.
SD0-SD7	Sirve como bus de datos para leer los primeros 4096 bytes de memoria de programa.

Es necesario hacer notar, que las señales SA0-SA11 y SD0-SD7 no pueden substituir a los buses externos de datos y direcciones, que se configuran de los Puertos 0 y 1. El Z8612 puede generar los buses direcciones/datos, de los Puertos 0 y 1, en forma idéntica al Z8. Las señales SA0-SA11 y SD0-SD7 estàn presentes solamente para accesar los primeros 4096 bytes de memoria de programa, además, SD0-SD7 no es un bus bidireccional, puesto que solamente trabaja como entrada de datos, y por lo tanto, no es posible escribir en memoria externa mediante este bus.

MDS	Es una señal equivalente a DATA STROBE y es válida (0 V) solamente en las instrucciones de los primeros 4096 bytes de memoria de programa.
SYNC	Es una señal de sincronización que es verdadera (0 V) solo durante el periodo de reloj inmediato precedente al siguiente ciclo de busqueda de instrucción.
SCLK	Es una salida del reloj interno, su frecuencia es la mitad de la frecuencia del cristal externo.
IACK	Es una señal de reconocimiento de interrupción, la cual es verdadera (+5 V) durante la secuencia de reconocimiento de interrupciones.

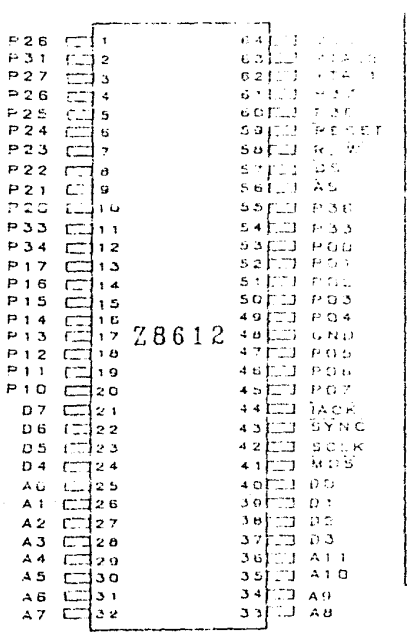
DESCRIPCION DEL Z8612



NOMBRE	DESCRIPCION	TIPO
LOS SIGUIENTES SON IDENTICOS AL Z8 VERSION NORMAL		
P00-P07	PUERTO 0 E/S	BIDIR,TRISTATE
P10-P17	PUERTO 1 E/S	BIDIR,TRISTATE
P20-P27	PUERTO 2 E/S	BIDIR,COLEC.ABIERTO
P30-P33	PUERTO 3 E/S	ENTRADA
P34-P37	PUERTO 3 E/S	SALIDA
AD0-AD7	BUS DATOS/DIRECCIONES	BIDIRECCIONAL
A8-A15	BUS DIRECCIONES	SALIDA
RDY0,DAV0	HANDSHAKE PUERTO 0 E/S	ENTRADA O SALIDA
RDY1,DAV1	HANDSHAKE PUERTO 1 E/S	ENTRADA O SALIDA
RDY2,DAV2	HANDSHAKE PUERTO 2 E/S	ENTRADA O SALIDA
R/W	CONTROL LECTURA/ESCRITURA	SALIDA,TRISTATE
DS	DATA STROBE	SALIDA,TRISTATE
AS	ADDRESS STROBE	SALIDA,TRISTATE
DM	SELECCION MEMORIA DATOS	SALIDA
IRQ0-IRQ3	LINEAS DE INTERRUPCION	ENTRADA
SIN	ENTRADA SERIAL	ENTRADA
SOUT	SALIDA SERIAL	SALIDA
TIN	ENTRADA TIMER	ENTRADA
TOUT	SALIDA TIMER	SALIDA
RESET	RESET DEL Z8612	ENTRADA
XTAL1-XTAL2	CONECCIONES DEL CRISTAL	
VDD,GND	FUENTE Y TIERRA	
LAS SIGUIENTES CORRESPONDEN ESPECIFICAMENTE AL Z8612		
SA0-SA11	BUS DIR.SIMULACION MEM.	SALIDA
SD0-SD7	BUS DATOS SIMULACION MEM.	ENTRADA
MDS	SELECCION DE MEM.INTERNA	SALIDA
SYNC	PULSO DE SINCRONIZACION	SALIDA
SCLK	RELOJ INTERNO Z8612	SALIDA
IACK	RECONOCIMIENTO DE INTERRUPCIONES	SALIDA

FIGURA (5.3)

DESCRIPCION DEL Z8612



NOMBRE	DESCRIPCION	TIPO
LOS SIGUIENTES SON IDENTICOS AL Z8 VERSION NORMAL		
P00-P07	PUERTO 0 E/S	BIDIR, TRISTATE
P10-P17	PUERTO 1 E/S	BIDIR, TRISTATE
P20-P27	PUERTO 2 E/S	BIDIR, COLEC. ABIERTO
P30-P33	PUERTO 3 E/S	ENTRADA
P34-P37	PUERTO 3 E/S	SALIDA
AD0-AD7	BUS DATOS/DIRECCIONES	BIDIRECCIONAL
A8-A15	BUS DIRECCIONES	SALIDA
RDY0, DAV0	HANDSHAKE PUERTO 0 E/S	ENTRADA O SALIDA
RDY1, DAV1	HANDSHAKE PUERTO 1 E/S	ENTRADA O SALIDA
RDY2, DAV2	HANDSHAKE PUERTO 2 E/S	ENTRADA O SALIDA
R/W	CONTROL LECTURA/ESCRITURA	SALIDA, TRISTATE
DS	DATA STROBE	SALIDA, TRISTATE
AS	ADDRESS STROBE	SALIDA, TRISTATE
DM	SELECCION MEMORIA DATOS	SALIDA
IRQ0-IRQ3	LINEAS DE INTERRUPCION	ENTRADA
SIN	ENTRADA SERIAL	ENTRADA
SOUT	SALIDA SERIAL	SALIDA
TIN	ENTRADA TIMER	ENTRADA
TOUT	SALIDA TIMER	SALIDA
RESET	RESET DEL Z8612	ENTRADA
XTAL1-XTAL2	CONECCIONES DEL CRISTAL	
VDD, GND	FUENTE Y TIERRA	
LAS SIGUIENTES CORRESPONDEN ESPECIFICAMENTE AL Z8612		
SA0-SA11	BUS DIR. SIMULACION MEM.	SALIDA
SD0-SD7	BUS DATOS SIMULACION MEM.	ENTRADA
MDS	SELECCION DE MEM. INTERNA	SALIDA
SYNC	PULSO DE SINCRONIZACION	SALIDA
SCLK	RELOJ INTERNO Z8612	SALIDA
IACK	RECONOCIMIENTO DE INTERRUPCIONES	SALIDA

FIGURA (5.3)

Los diagramas de tiempo de estas señales se muestran en las Figuras 4 y 5.

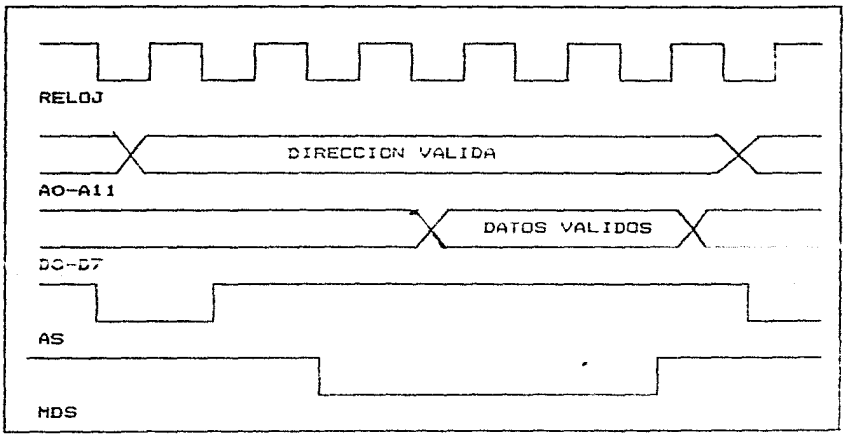


FIGURA (5.4)

El SIS Z8 emplea las siguientes señales del Z8612 en forma exclusiva, es decir, el usuario tiene que prescindir de ellas para sus desarrollos.

IRQO En las rutinas de GO y CALL, cuando se utilizan conjuntamente con alguno de los comandos de rompimiento de programa (TRACE ó BREAK POINTS), emplean la petición de interrupción IRQO, generada por lógica externa, para regresar el control al programa monitor.

P27 La señal que se obtiene de este puerto deshabilita la lógica externa asociada a los buses de dirección/datos del Z8612, al mismo tiempo que habilita la interfaz de los Puertos 0 y 1. Para permitir que el usuario disponga de los Puertos 0 y 1 en cualquiera de sus modalidades, este puerto se ha sacrificado en el SIS Z8, por razones que serán evidentes más adelante.

Así también, todas las señales del Z8612 que no corresponden a la versión normal del Z8, son empleadas en forma exclusiva, a excepción de SYNC, SCLK, IACK y MDS.

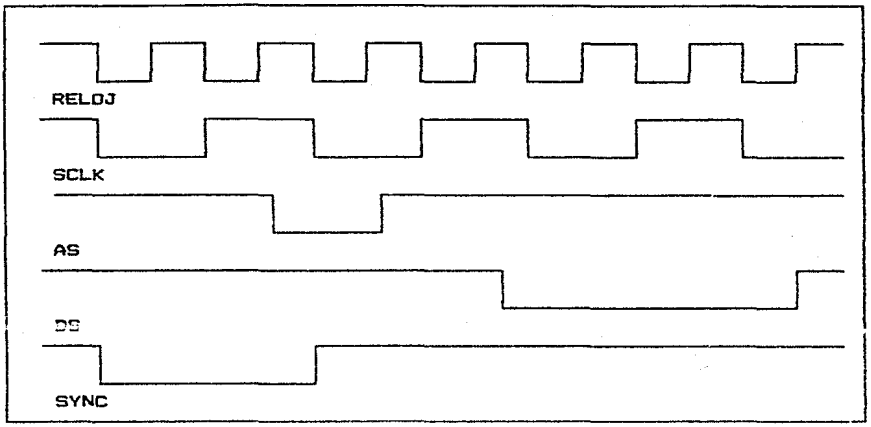
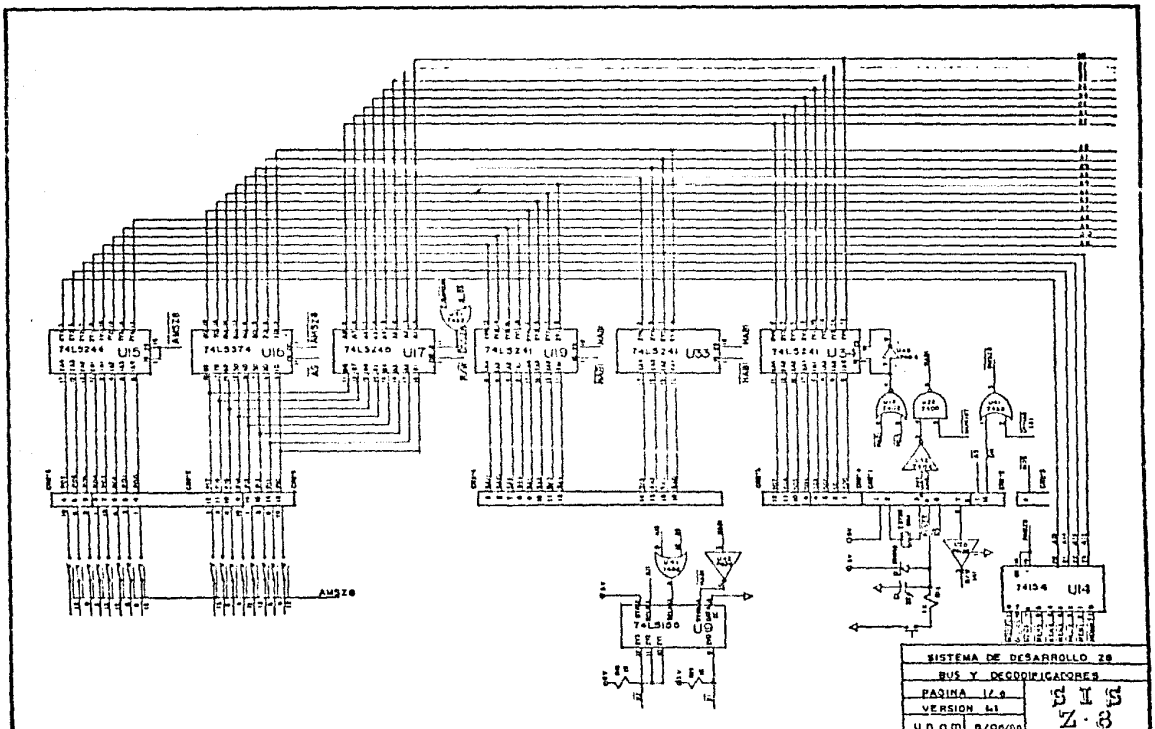


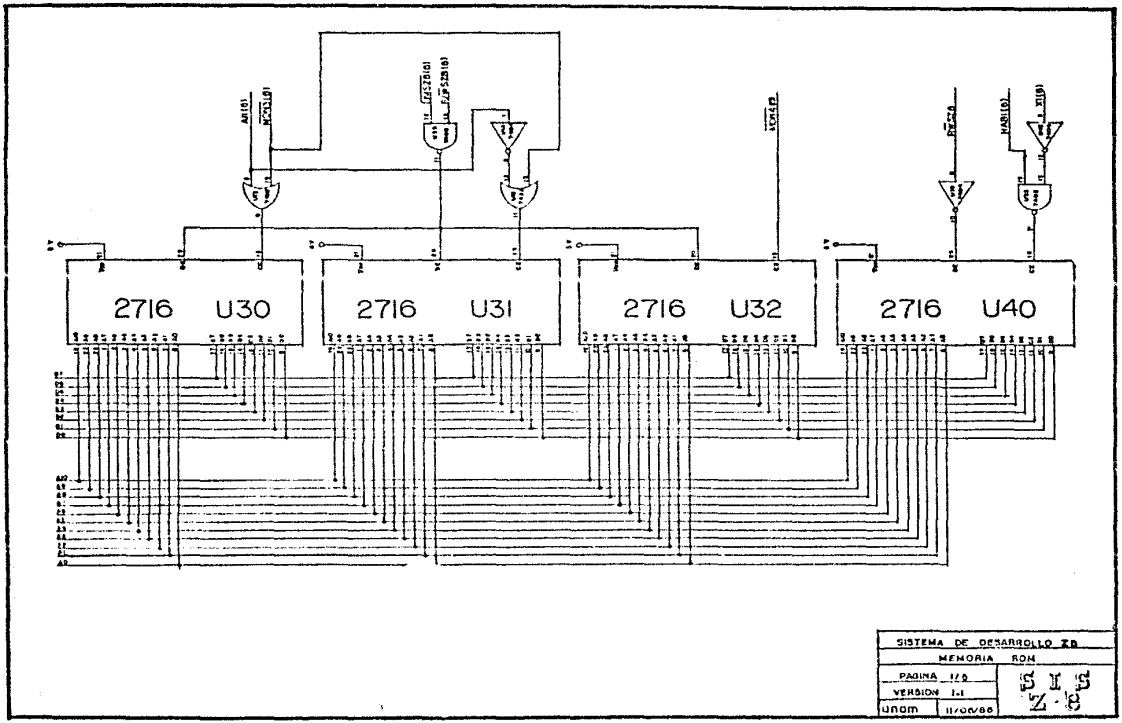
FIGURA (5.5)

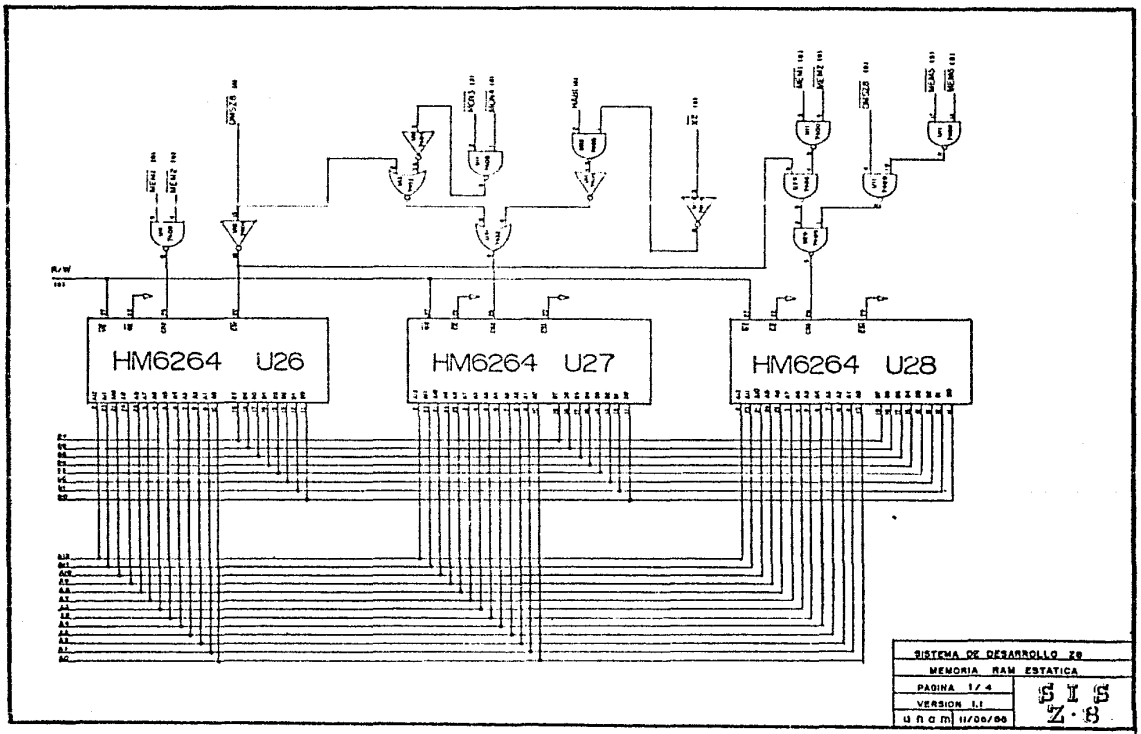
Las señales SIN, SOUT y DM, aunque son empleadas por el SIS Z8, pueden ser reprogramadas por el usuario en cualquiera de sus modalidades.

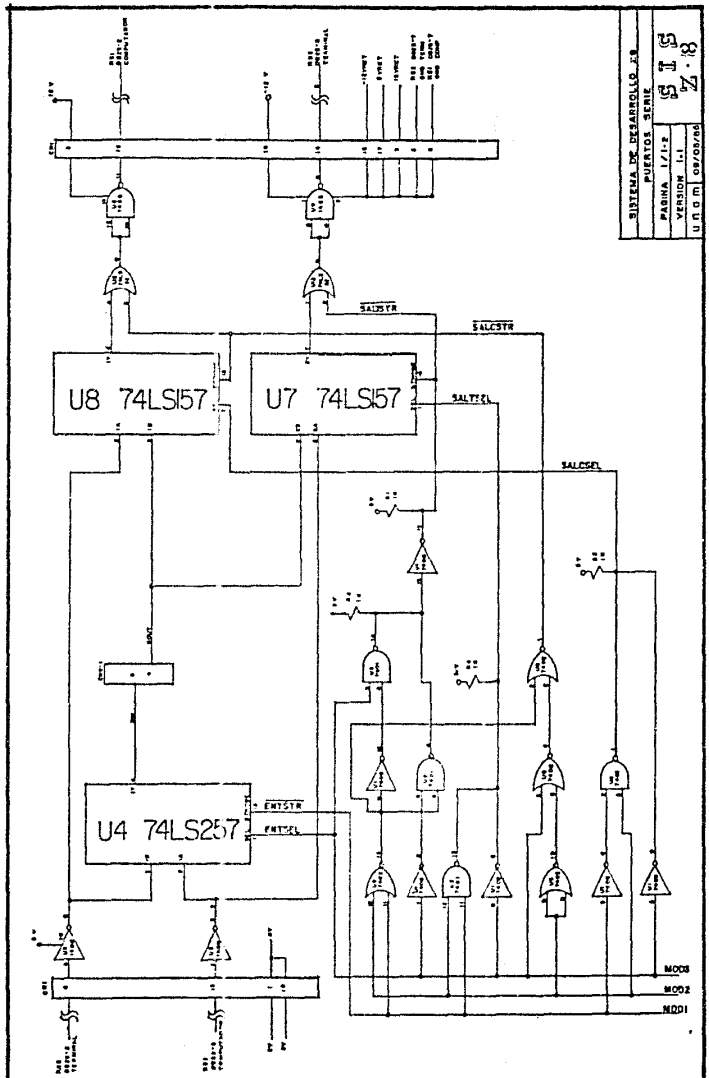
Puesto que el SIS Z8 se comunica con una terminal y una computadora anfitriona vía su puerto serial, el Z8612 trabaja con un cristal de cuarzo de 7.3728 MHZ, por la facilidad que esta frecuencia ofrece para generar internamente, algunos de los baudajes (Baud Rates) mas usados en el protocolo de comunicaciones asincronas RS232-C

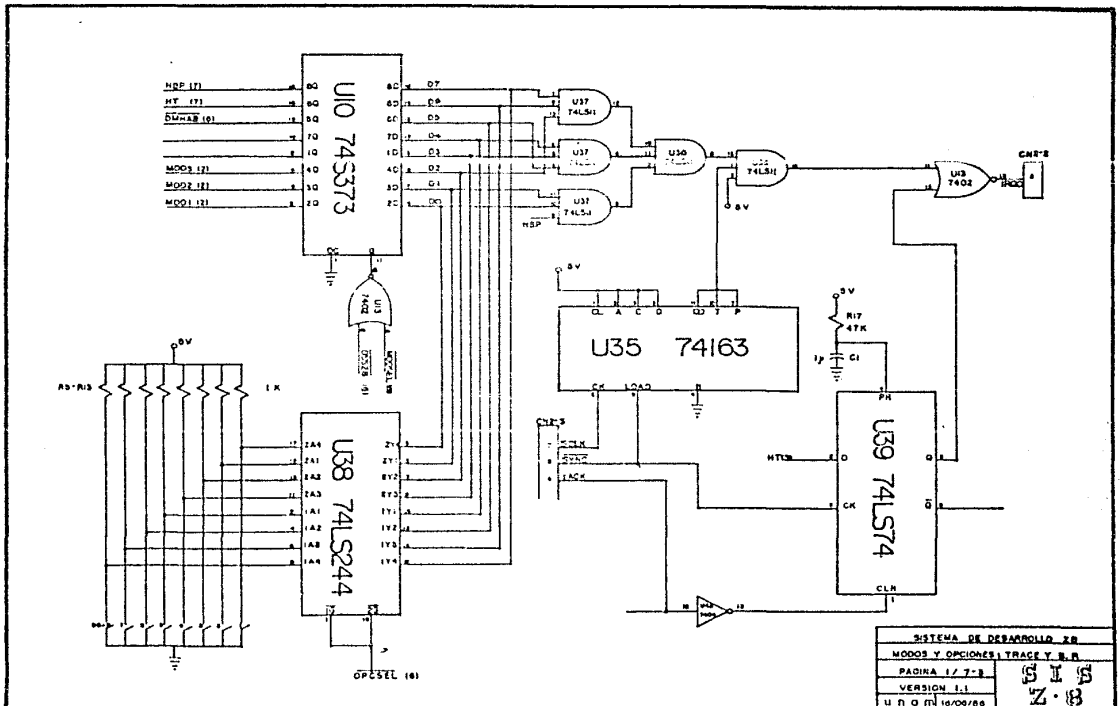
Ahora que ya se han visto algunas de las generalidades del SIS Z8 podemos continuar con la revisión de los diagramas.


74LS244 U15
74LS374 U16
74LS240 U17
74LS241 U19
74LS241 U33
74LS241 U34
74LS00 U9
74134 U14
AMS28
SISTEMA DE DESARROLLO Z8
BUS Y DECODIFICADORES
PAGINA 1/1
VERSION 1.1
unam 8/08/88
SIS Z8


2716 U30
2716 U31
2716 U32
2716 U40
SISTEMA DE DESARROLLO ZB
MEMORIA ROM
PAGINA 1/2
VERSION 1.1
unam
11/oct/88


HM6264 U26
HM6264 U27
HM6264 U28
SISTEMA DE DESARROLLO Z8
MEMORIA RAM ESTATICA
PAGINA 1/4
VERSION 1.1
unam 11/06/86
SIS Z-8


SISTEMA DE DESARROLLO J.B.
PUERTOS SERIE
PAGINA 1/1-2
VERSION 1.1
U n a m 08/08/96
U8 74LS157
U7 74LS157
U4 74LS257
SALSTR
SALCSTR
SALTSEL
SALCSEL
ENTSTR
ENTSEL
MOD0
MOD2
MOD1



SISTEMA DE DESARROLLO Z8	
MODOS Y OPCIONES ; TRACE Y B.B.	
PAGINA 1/ 7-8	
VERSION 1.1	
U N A M 10/06/86	

BUSES E INTERFAZ PUERTOS 0 Y 1.

El SIS Z8 tiene dos buses: El bus de direcciones que consta de 16 líneas y el bus de datos de 8 líneas. Estos buses pueden generarse de 2 maneras:

a) A través de los Puertos 0 y 1, cuando la señal AMSZ8 (obtenida del Puerto 27) habilita la lógica asociada a estos buses. En este caso se puede accesar cualquier byte de memoria a excepción de los primeros 4096 bytes de memoria de programa .

b) A través de las señales SA0-SA11 y SD0-SD7 independientemente del estado de la señal AMSZ8. En este caso solo se pueden accesar los primeros 4096 bytes de memoria de programa.

La pregunta obligada : ¿Por que tanta complicación para generar los buses ?

Durante la etapa de diseño del SIS Z8 se estudió la posibilidad de no incluir al Z8612 en el prototipo final, puesto que este circuito integrado es difícil de conseguir y de precio elevado. Se esperaba lograr un diseño funcional con la versión simple del Z8, pero se presentó un problema insoluble: EL Z8 EN SU VERSION SIMPLE TIENE ROM INTERNA. Este hecho obligaba a configurar al SIS Z8 con el SCAN Z8 en la ROM interna y el programa del usuario en RAM externa. En esta forma, el programa del usuario debe ser accesado por los Puertos 0 y 1 configurados como buses de datos/direcciones, con lo cual se limita las posibilidades del usuario al desperdiciarse un total de 16 puertos, sin otra opción que la de trabajar como buses.

Con el Z8612 en el diseño final, el usuario tiene disponibles ambos puertos en cualquiera de sus modalidades, ya que permite que el programa del usuario quede localizado en memoria de programa interna, aunque ciertamente èsto complicó el alambrado.

Para poder introducirnos en el funcionamiento de los buses del SIS Z8 examinemos algunas consecuencias no evidentes de su estructura:

1.- Si el usuario desea emplear los Puertos 0 y 1, en cualquiera de sus modalidades de puertos entrada/salida, entonces, debe colocar su programa dentro de los primeros 4096 bytes de memoria de programa, para liberar a los puertos de su función de buses. El SCAN Z8 se encarga de generar una señal que deshabilite la lógica asociada, de tal manera que las señales de entrada o salida a estos puertos no sean interpretadas como datos ó direcciones, y

visceversa. Cuando el control regresa al programa monitor y el SIS Z8 necesita de los Puertos 0 y 1, la misma señal habilita nuevamente la lógica externa de los buses y desconecta la interfaz a los Puertos 0 y 1. Esta es la función de la señal AMSZ8.

2.- En los primeros 4096 bytes de memoria de programa, hay lugar suficiente para el programa del usuario, ya que el programa monitor del SIS Z8 solo ocupa los primeros 512 bytes. A esta parte del programa monitor lo llamaremos MONITOR INTERNO, y queda localizado entre las direcciones 0000_H y $01FF_H$. El resto del programa se encuentra entre las localidades 3000_H y $3FFF_H$ de memoria de programa externa. A esta otra la llamaremos MONITOR EXTERNO ó PRINCIPAL.

3.- El programa del usuario normalmente se transfiere de la computadora anfitriona al SIS Z8 a través del puerto serial del Z8612. Este recibe el código y lo acomoda en la localidad de memoria correspondiente. Cuando el programa del usuario queda localizado en memoria de programa externa, la transferencia se hace directamente a través de los buses generados por los Puertos 0 y 1. Pero cuando se desea que el programa sea instalado en memoria interna, la escritura a memoria quedaría a cargo de SD0-SD7, lo cual no es posible, ya que este bus es unidireccional. Este problema se resolvió haciendo posible que la RAM que simula la memoria interna pudiera accesarse en dos localidades lógicas distintas, una como memoria de programa interna entre las localidades 0200_H y $0FFF_H$, y la otra como memoria de datos a partir de la dirección 3200_H hasta la $3FFF_H$. De esta manera cuando se desea modificar una localidad en memoria interna, lo que en realidad se hace es accesarla en memoria de datos externa, donde si son posibles los ciclos de escritura a memoria.

El SIS Z8 emplea dos señales para contolar el origen de los buses: AMSZ8 y MEMINT. Al inicializar (RESET), la señal AMSZ8 tiene un nivel alto deshabilitando la lógica externa asociada a los buses primarios de los Puertos 0 y 1, y por lo tanto, el origen de los buses de datos/direcciones es en las lineas SA0-SA11 y SD0-SD7 del Z8612. El programa monitor interno configura los Puertos 0 y 1 como buses, modifica el estado de la señal AMSZ8 e inmediatamente pasa el control al programa monitor principal. Cuando esto ocurre, la memoria es accesada mediante los Puertos 0 y 1. Cuando el monitor principal tiene que hacer una búsqueda a memoria de programa interna, el estado del bus de direcciones del SIS Z8 es $0XXX_H$. El cero en el nibble más significativo produce la señal MEM INT en el decodificador U14 (Ver diagrama #1). Esta señal la interpreta la lógica de control de buses y permite que el origen de los buses sea nuevamente en las señales SA0-11 y SD0-7. Este es le caso del comando GO, cuya

ejecución comienza en el monitor principal, salta al monitor interno y antes de ejecutar la rutina del usuario reconfigura estos puertos como entradas o salidas.

Veamos ahora como estas señales controlan los buses a nivel del alambrado. Cuando AMSZ8 tiene un nivel bajo habilita las salidas de U15 y U16, que en caso contrario se encuentran en estado de alta impedancia. La señal AS funciona como reloj para los flip flops de U16 que "amarran" el byte menos significativo del bus de direcciones, ya que éste se presenta multiplexado con el bus de datos en las terminales del Puerto 1.

Por otro lado cuando AMSZ8 y DS tienen un valor BAJO, la salida de la compuerta OR U41 es baja, permitiendo la entrada o salida de datos del transceptor bidireccional U17, y, donde la dirección del mismo la establece la señal R/W. La señal R/W original del Z8612 pasa primeramente por el seguidor amplificador U20, debido a que el fan out al que está sometida, sobrepasa las especificaciones eléctricas máximas.

AMSZ8 después del inversor U42, habilita a U14, cuya funcion es seccionar el espacio direccionable en 16 bloques, al decodificar las líneas de dirección A12-A15. Cuando estas últimas cuatro se encuentran en un nivel de cero lógico, forzan a MEM INT a cero volts.

Cuando MEM INT es cierta ó AMSZ8 es falsa (+5 V), la señal HABI de la NAND U25 es igual a cero volts; con lo cual quedan habilitados los seguidores amplificadores U19 y U33, conectados a las terminales SA0-SA11 del Z8612.

Esta misma señal se combina con MDS en la compuerta NOR U13, para que cuando ambas tengan un nivel bajo permitan el paso de un dato del bus del SIS Z8 al bus secundario SD0-SD7 a través de U34.

Debido a la estructura de los buses, el SIS Z8 trabaja con 2 decodificadores para seleccionar los distintos elementos funcionales con que cuenta: Uno es U9, para seleccionar aquellos cuyas direcciones correspondan al espacio destinado a memoria interna, y el otro, U14, del cual ya habíamos comenzado a hablar. Este último tiene la doble función de seleccionar a los elementos que se encuentren en el espacio destinado a memoria externa, además de servir de herramienta para distinguir entre memoria interna y memoria externa, ya que proporciona la señal MEM INT que ya estudiamos.

U14 divide los 64 KB de espacio direccionable en 16 bloques de 4 KB cada uno, sin hacer distinción entre memoria de programa y memoria de datos. Además de MEM INT, U14 produce 6 señales destinadas a seleccionar memoria y 2 mas para control del SIS Z8. De las 6 primeras, cuatro direccionan RAM y las otras dos al

programa monitor principal, asi como a la RAM de simulación de memoria interna. A estas señales regresaremos más adelante cuando se estudien los diagramas 2 y 3. En cuanto a las señales de control del SIS Z8, OPCSEL y MODSEL nos ocuparemos cuando veamos los puertos serie y la lógica de control.

La señal DMSZ8 se encarga de distinguir entre memoria de datos y de programa. Para lo cual se vale de DM en combinación con DMHAB. Esta última es una señal que se modifica por programación y que es producida por el puerto de control interno, el cual será analizado más adelante. La combinación de estas dos señales se realiza en la OR U41. DMHAB fué agregada para permitir que el Puerto 34, empleado por el SIS Z8 como DM, pueda ser programado por el usuario de la manera que sea, sin afectar el desempeño del SIS Z8. Si DMHAB es alta, DMSZ8 lo será también, independientemente del estado de DM. El usuario puede programar a P34 inclusive como DM, y emplear la memoria de datos del SIS Z8, si así lo desea, sin [illegible] y cuando modifique también la señal DMHAB. La señal DMHAB es baja casi siempre, excepto cuando se ejecutan rutinas del usuario a través del comando GO

El decodificador secundario U? es habilitado por HABI invertida por U42, y divide a la memoria interna en 8 bloques de 1/2 KB cada uno, aunque en realidad solo produce 2 señales: SEL1 y SEL2.

Cuando A9-A11 son todas bajas se produce SEL1, que selecciona al monitor interno. La resistencia de pullup R16 es necesaria ya que las salidas del 74LS156 son del tipo colector abierto. Cualquier otra combinación de las direcciones A9-A11 produce la señal SEL2, ya que las otras tres salidas están formando una OR alambrada con la resistencia R15. La señal SEL2 selecciona la RAM de simulación de memoria interna.

Finalmente los switches analógicos U21-U24, conforman lo que hemos denominado hasta ahora interfaz de los Puertos 0 y 1. La función de estos switches no resulta evidente a simple vista. Por lo que explicaremos su función con un ejemplo:

El usuario desea poner a prueba un programa que va a localizar en memoria interna y ejecutarlo a través del comando GO con puntos prueba (BREAK POINTS) en varias localidades. Este programa emplea la totalidad del Puerto 1 como entrada , mientras que el Puerto 0 es configurado como salida, y por lo tanto, tiene conectados externamente, algunos circuitos que van a enviar señales al Puerto 1 y recibir señales del Puerto 0, como se aprecia en la Figura (5.6).

El usuario transfiere su programa de la computadora anfitriona al SIS Z8 mediante el comando TRM. Desplegando la memoria en su terminal, verifica que su programa ha sido instalado correctamente, y después, con el comando BP, inserta

puntos prueba en las direcciones deseadas. Finalmente pasa a ejecutar su programa tecleando desde la terminal la instrucción GOI para que al inicio de su programa, el Z8612 quede con todos sus registros programados como inmediatos a un RESET.

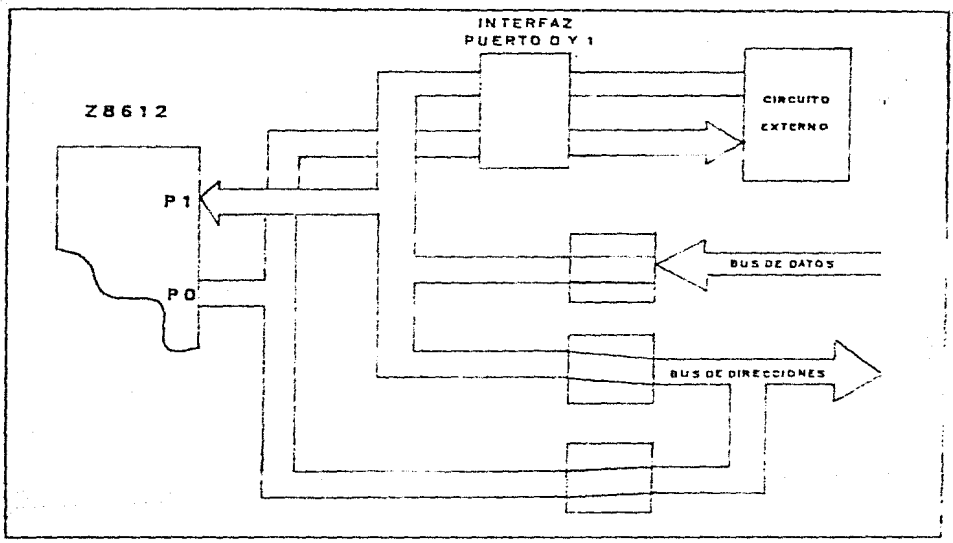


FIGURA (5.6)

Aunque durante todo este tiempo los circuitos externos han estado polarizados, la actividad en las terminales del Puerto 1 no se ha visto interferida, ni las señales presentes en el Puerto 0 han sido interpretadas como salidas, debido a que los switches analógicos interponen una alta impedancia entre estos puertos y el circuito externo.

Antes de pasar el control al programa en prueba, el SCAN Z8 reprograma el registro P01M para reconfigurar ambos puertos como entradas. Esto lo puede hacer el SCAN Z8, ya que esta parte del

programa se ejecuta en el monitor interno donde ya no es necesario el uso de los buses principales de los Puertos 0 y 1. La penúltima instrucción coloca un 1 en el bit menos significativo del registro R2, lo cual produce que AMSZ8 cierre los switches analógicos y deshabilite los drivers U15, U16 y U17 conectados a las terminales P00-P07 y P10-P17 .

El programa en prueba reprograma P01M y configura los Puertos 0 y 1 en la forma deseada (ver figura siguiente). La actividad entre los circuitos externos y estos Puertos no interfiere con los buses del SIS Z8 porque U15 a U17 han sido aislados por el estado de AMSZ8.

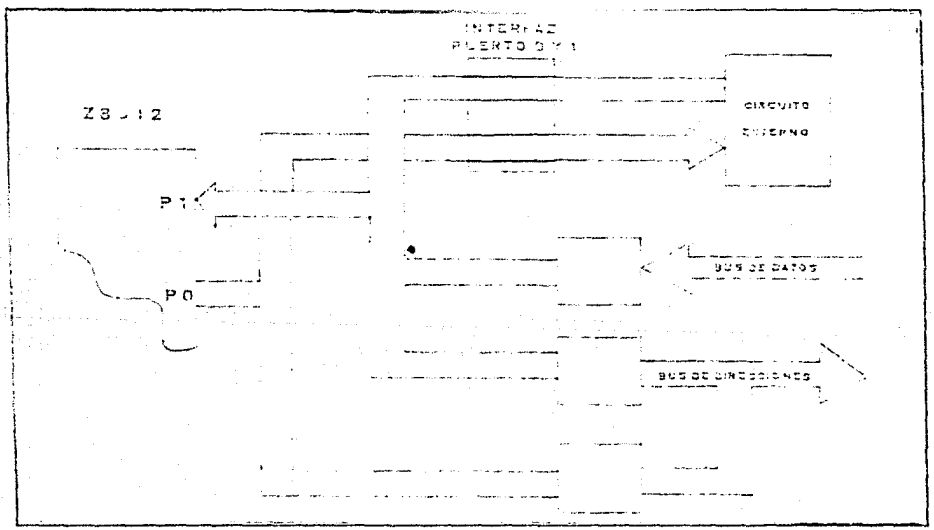


FIGURA (5.7)

Cuando el contador de programa llega a una localidad indicada como punto prueba, la lógica de Break Points produce una petición de interrupción IRQ0, que regresa el control al SCAN Z8. La rutina de servicio de esta interrupción comienza en el monitor interno y repite los pasos anteriores a la ejecución del programa en prueba, pero esta vez en sentido inverso, para culminar nuevamente en el esquema de la Figura (5.6).

MEMORIAS RAM Y ROM

La Figura (5.8) muestra el mapa de memoria del SIS Z8. El SCAN Z8 ocupa tres memorias EPROM de 2K8 cada una. El monitor interno se encuentra en los primeros 512 bytes de una de estas memorias (entre las direcciones 0000H y 01FFH), y contiene los programas del SCAN Z8 que no son ejecutables en memoria externa.

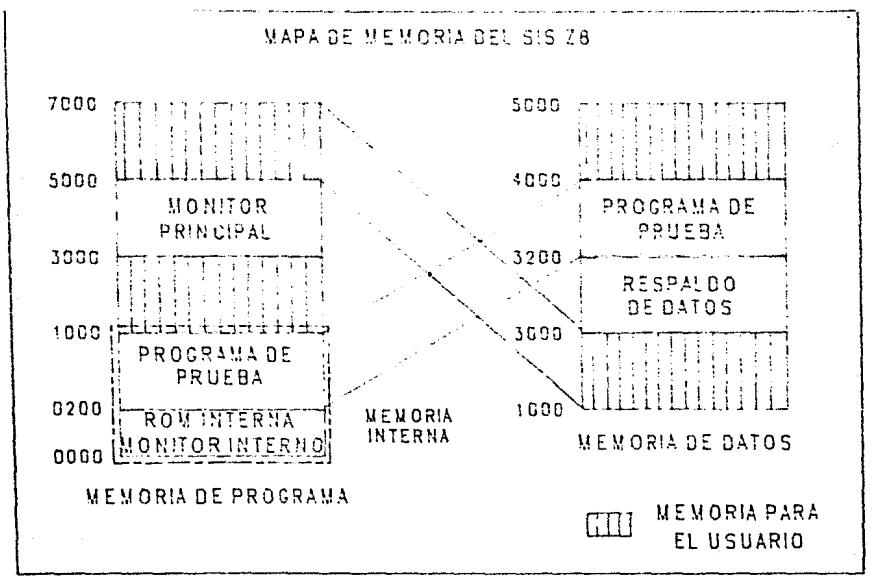


FIGURA (5.8)

El resto del programa se localiza entre las direcciones 3000_H y $3FFF_H$ de memoria de programa externa. En el prototipo se dispuso el alambrado necesario para seleccionar una EPROM 2716, a partir de la dirección 4000_H, con el propósito de que el usuario pueda insertar una EPROM de este tipo previamente programada para fines de expansión. Las direcciones 4800_H a $4FFF_H$ no han sido implementadas. Las localidades 1000_H a $2FFF_H$ de memoria de programa, son ocupadas por una RAM estática de 8Kbx8. Las 8192 localidades de memoria de programa, a partir de la dirección 5000_H a la $6FFF_H$, junto con las localidades de memoria de datos 1000_H a $2FFF_H$, están implementadas por una sola memoria RAM de 8KB, que puede ser accesada indistintamente en cualquiera de estas posiciones lógicas. Compartiendo el mismo espacio que el monitor superior, pero en memoria de datos, se localiza la RAM de simulación de memoria interna, por lo cual las direcciones 3200_H a $3FFF_H$ pueden ser escritas o leídas en memoria de datos o leídas entre las 0200_H y la $0F00_H$ de memoria de programa. De esta memoria, las localidades ocupadas entre la 3000_H hasta la $31FF_H$ y entre la 4000_H y la $41FF_H$, son ocupadas para operaciones del SCAN Z8. A partir de la 4100_H hasta la $4FFF_H$, quedan disponibles para el usuario.

En el diagrama #2, U30, U31 y U40, son las memorias EPROM 2716 que contienen al SCAN Z8.

En el diagrama #3, las memorias RAM estáticas HM6264, se identifican de la siguiente manera: U26 es la memoria fija de las direcciones 1000_H a $2FFF_H$ de memoria de programa. U27 es la memoria de simulación de programa interno y U28 es la memoria móvil direccionable en programa entre las direcciones 5000_H y la $6FFF_H$, y en datos desde la 1000_H a la $2FFF_H$.

El HM6264, es un circuito relativamente poco conocido, por lo cual se ha agregado en el apéndice 1 su hoja de datos y especificaciones.

A excepción de las áreas en memoria de datos marcadas como RAM Sistema, los 24576 Bytes obtenidos en este arreglo son disponibles para el usuario.

La RAM que ocupa 2 posiciones lógicas, hace que la capacidad de memoria del SIS Z8 aumente virtualmente hasta 32768 bytes, sin embargo, es necesario tener ciertas precauciones en el manejo de esta última, ya que los bytes escritos en una dirección en memoria de programa, pueden leerse o cambiarse en la dirección equivalente en memoria de datos.

Los circuitos de selección de estas memorias se explican a continuación:

U30 y U31: Las entradas OE (pin 20) de U30, U31 y U32, han sido alambradas en un nodo común, cuya salida es el pin 11 de la NAND U29 y que es baja cuando DMSZ8 y R/W son ambas altas, esto es, cuando se trata de un ciclo de lectura a una localidad de memoria exclusiva de programa. La señal MON3 es baja cuando U14 ha decodificado un 03_H de las direcciones A12-A15. Como esto último comprende 4KB de memoria, A11 indica cual de los circuitos U31 y U30 debe ser seleccionado a través de las OR's U12 (pin 8 y 11) y del inversor U18, de tal manera que cuando A11 es baja U30 es habilitada por un nivel bajo en su entrada CE (pin 18). En forma equivalente U31 es seleccionada cuando A11 tiene un nivel alto.

U40: En esta memoria se tiene programada el código del monitor interno y se encuentra protegida por la señal R/W. Es seleccionada cuando HABI es alta y X1 es baja, lo cual indica que el origen de los buses es a través de SA0 y SA11, y que del bus de direcciones resultante U9 ha decodificado un valor inferior a la localidad 8000...

U26: Este es el caso más simple de direccionamiento. El HM6264 tiene 2 entradas habilitadoras complementarias, ambas deben ser ciertas para acceder una localidad de esta memoria. U14 decodifica una dirección dentro del rango 1000_H al $2FFF_H$, mediante MEM1 o MEM2, y cualquiera de èstas selecciona a U26, a través del pin 6 de U11 en compañía de un nivel alto de DMSZ8, que indique direccionamiento a memoria de programa.

U28 : Esta es la memoria direccionable en espacio de programa y datos externos. La primera condición para seleccionarla, es cuando se accesa memoria de datos (DMSZ8 baja), a partir de la dirección 1000_H a la $2FFF_H$ (MEM1 o MEM2 de U14). La segunda es con memoria de programa (DMSZ8 alta), entre la 5000_H y la $6FFF_H$ (MEM5 o MEM6 de U14). El pin 3 de la NAND 29 solo hace la conjunción (OR) lógica de estas condiciones.

U27: Tiene asociado el circuito más complejo de selección. Al igual que U28, este circuito integrado se selecciona en 2 formas distintas, determinadas por las siguientes condiciones:

*Puede leerse o escribirse en este circuito integrado, en memoria de datos entre las localidades 3000_H a $4FFF_H$, cuando DMSZ8 es baja junto con MON3 o MON4.

*Simula a la memoria de programa interna del Z8 y solamente puede leerse cuando las señales HABI y X2 de U9 son simultáneamente bajas.

PUERTOS SERIE

Para comunicarse con la computadora anfitriona o con el usuario, a través de una terminal, el SIS Z8 dispone de dos puertos asincronos seriales compatibles con niveles RS232C. Aunque externamente los puertos se encuentran bien diferenciados, para fines del análisis que de ellos haremos posteriormente, es conveniente no tratarlos como dos entidades separadas, sino más bien, como un solo puerto serial programable, con dos canales de entrada y dos de salida, que pueden combinarse para operar en varios modos.

Esta interpretación es necesaria, ya que el UART interno del Z8612 ha sido aprovechado para operar ambos puertos y para la introducción y envio de información del SIS Z8. Así, el Z8612, solo puede atender una entrada y una salida a la vez, mientras que la otra paraja puede servir para transmisión de data o quedar deshabilitada. Como ya vimos desde el capitulo 4, cuando el SIS Z8 se usa con una computadora anfitriona, sirve de enlace entre esta última y una terminal. Para evitar interferencia entre ambas entradas, el SCAN Z8, selectivamente programa la lógica externa para dar paso a los datos provenientes de uno u otro puertos hacia el Z8612.

La estructura en bloques de estos puertos se muestra en la Figura 5.9.

En este diagrama es más comprensible la idea de un solo puerto serie. La separación en dos puertos, es aparente solo en los conectores DB25 externos. El bloque denominado Administrador de Entrada/Salida, se encarga de hacer posibles las combinaciones entre las entradas y las salidas, que en primera instancia son determinadas por el SCAN Z8. Cuando el SCAN Z8 debe cambiar el modo de operación, se vale del bloque llamado Puertos Interno de Control del SIS Z8.

El puerto interno de control, es el circuito integrado U10 del diagrama #5 y que consta de 8 Flip Flops tipo D, que al ser seleccionados congelan un byte de datos, del cual los 3 bits menos significativos codifican el modo de operación de los puertos. Los bits restantes cumplen funciones diversas, una de ellas es la señal DMAB que ya analizamos en la sección anterior.

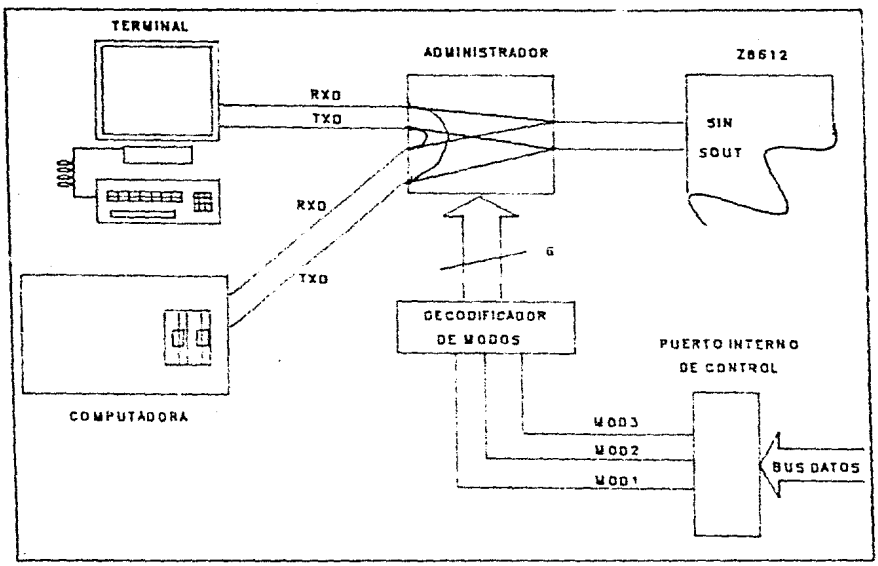


FIGURA (5.9)

El decodificador de modos de operación, es un circuito combinacional, que determina el estado de las señales de control del Administrador en función de los 3 bits del puerto interno MOD1-MOD3. En las figuras 5.10 se representan los modos de operación del administrador y la manera en que son codificados.

En cualquier caso la terminal debe trabajar en modo full duplex. Al inicializar, el SCAN Z8 programa al administrador en el modo 1. En este modo el SIS Z8 interpreta los códigos enviados por la terminal, en espera de la clave de acceso ($060D_H$). Mientras el usuario no introduzca la clave de acceso al SIS Z8, èste repite el código y lo transmite a la computadora.

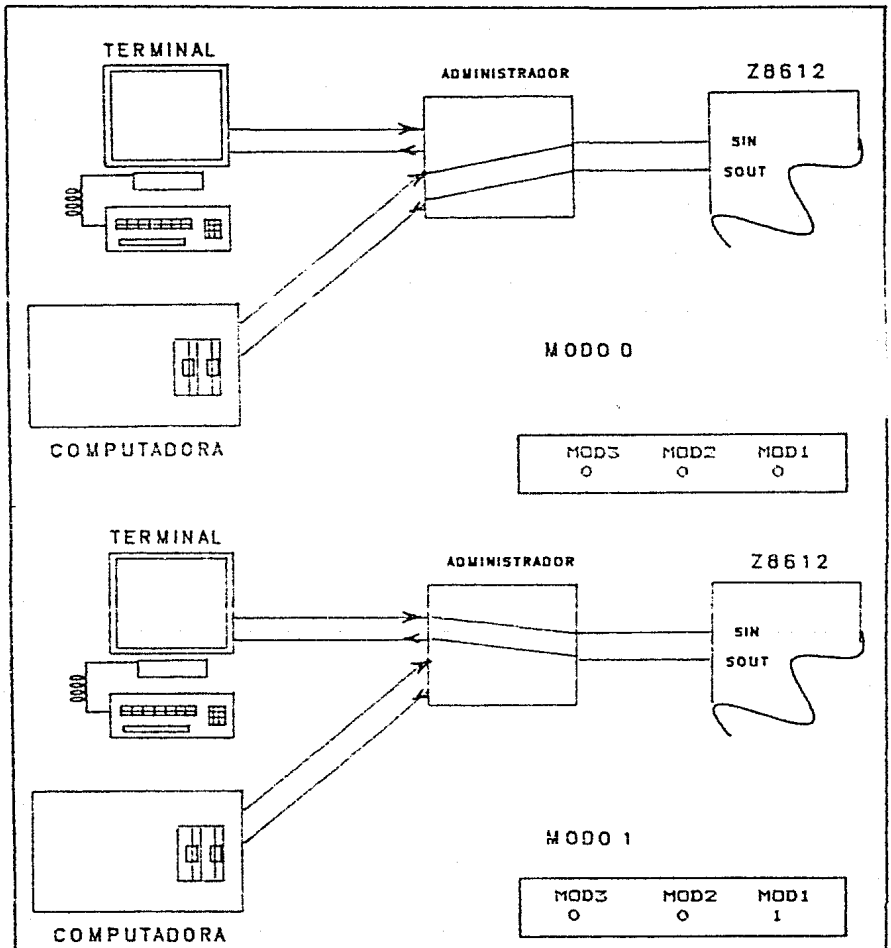


FIGURA (5.10 a y b)

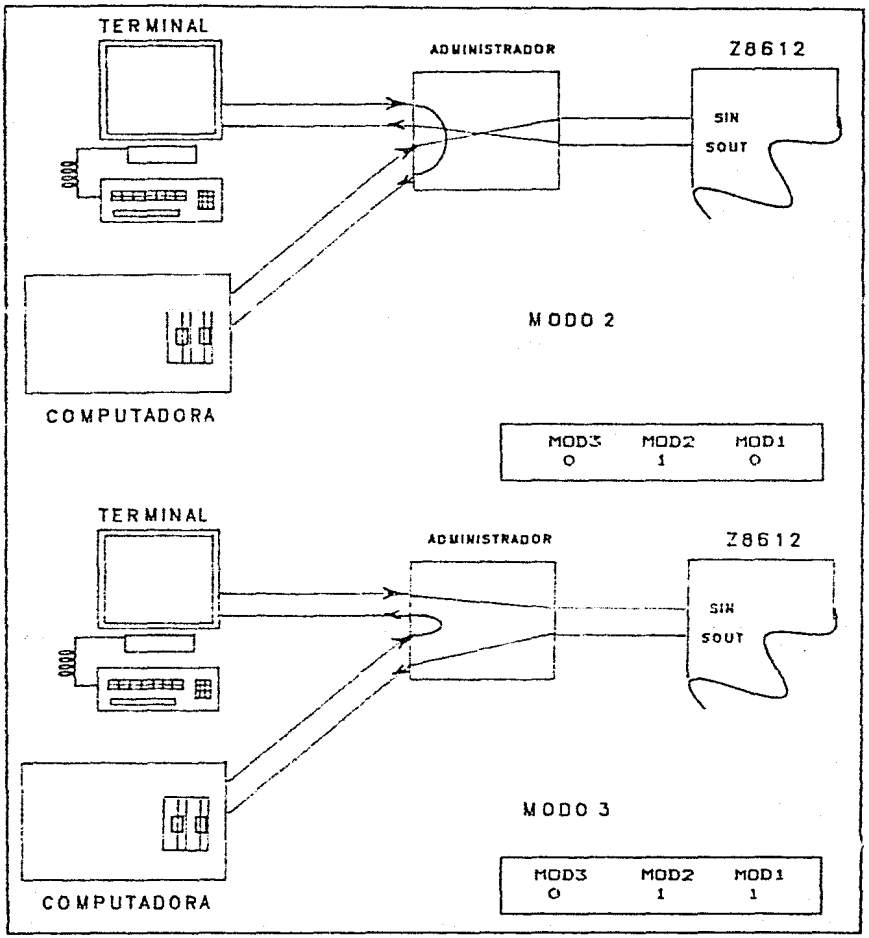


FIGURA (5.10 c y d)

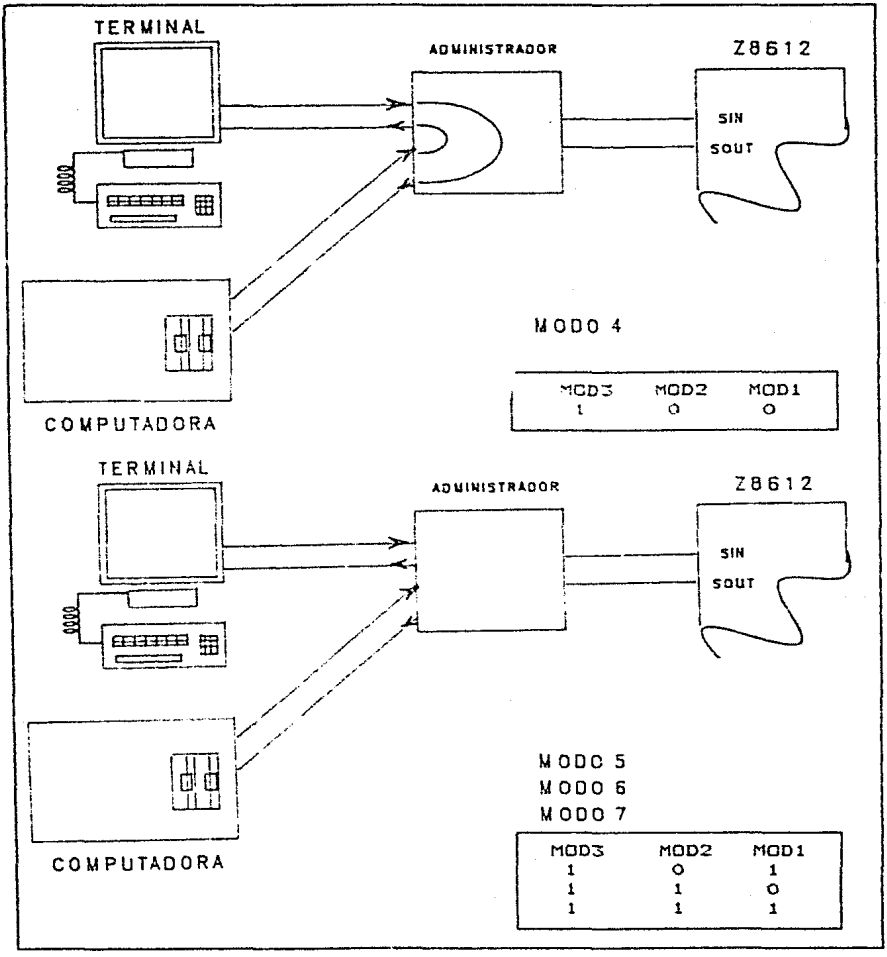


FIGURA (5.10 e y f)

Cualquier información enviada desde esta última, pasa por el SIS Z8 sin retraso hacia la terminal. En este modo el SIS Z8 es transparente y no interfiere con la operación de la computadora anfitriona.

Para accesar al SIS Z8, el usuario envía la secuencia de teclas CONTROL F seguida de un CR, que al ser interpretada por el SCAN Z8, inmediatamente cambia al modo 2 del Administrador, y el SIS Z8 se comunica únicamente con la terminal. Esta secuencia no pasa a la computadora anfitriona.

El modo 3, de comunicación exclusiva con la computadora anfitriona, es empleado durante la ejecución del comando TRM de transferencia de un archivo con formato INTEL, hacia la memoria del SIS Z8.

El modo 5, es programado antes de ejecutar un programa de prueba. Este modo enlaza la terminal y la computadora directamente y deja libres al usuario los terminales P30 (Sin) y P31 (Sout) del Z8612, ya que la salida 4 del circuito integrado U4, es puesta en estado de alta impedancia.

El Administrador de Entrada/Salida, se compone de los circuitos U3, U4, U6, U7, U12 y U6. Los circuitos integrados U3 y U6 (1488 y 1489), son solamente acondicionadores de las señales de niveles RS232-C a TTL y viceversa. El multiplexor de entrada U4 (74LS257), es programado por el nivel de ENSEL para determinar cual de las entradas será enviada al Z8612. Si ENSTR tiene un valor alto, la salida 4Y presenta alta impedancia, independientemente del nivel de ENSEL, esto permite que el Puerto P30 sea programado en cualquiera de sus opciones por el usuario.

Los multiplexores de salida U7 y U8 (74LS157), son programados por SALTSEL y SALCSEL respectivamente, para determinar cual canal es enviado a la computadora y a la terminal.

En los modos 5, 6 y 7, SALCSTR y SALTSTR tienen un nivel alto para interrupir la comunicación hacia la terminal, la computadora anfitriona o ambas. Las OR U12 son necesarias para mantener la condición de marca, durante las interrupciones de transmisión.

El decodificador de modos del administrador, está integrado por los C.I. U1, U2 y U5. Este bloque en conjunto tiene como entradas a MOD1-MOD3, y como salidas las señales que controlan a los multiplexores U4, U7 y U8: ENTSTR, SALTSTR, SALCSTR, ENTSEL, SALTSEL y SALCEL.

Para ilustrar claramente el funcionamiento global de los puertos, veremos como el modo 1 es programado y como ocurre el flujo de las señales transmitidas por la terminal y la computadora en este modo.

Cuando el SCAN Z8 debe trabajar en este modo, escribe en uno de los registros de trabajo un 001 en los 3 bits menos significativos. Después hace una escritura a memoria de programa a la localidad FXXX$_H$, siendo la fuente de la instrucción el registro recién programado. U14 decodifica la dirección y selecciona al Puerto Interno de control mediante un nivel bajo de la señal PINTSEL.

En el bus de datos aparece el contenido del registro de trabajo y es amarrado por los Flips Flops de U10. Ahora los 8 bits de salida del puerto interno de control tienen un valor idéntico al registro de trabajo. Los tres bits MOD1-MOD3 son decodificados por el decodificador de modos, produciendo que las señales de control de U4, U7 y U8 tengan los siguientes niveles:

ENTSTR.....	0 V
SALTSTR.....	0 V
SALCSTR.....	0 V
ENTSEL.....	0 V
SALTSEL.....	5 V
SALCSEL.....	0 V

El flujo de las señales de comunicación es el mostrado en la siguiente figura:

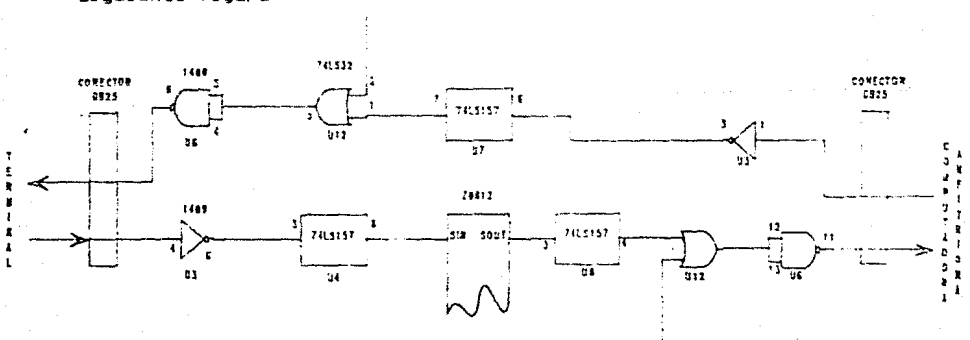


FIGURA (5.11)

Los caracteres enviados por la terminal son acondicionados al nivel TTL a la salida de U3 pin 6. Pasan a través de U4 y son enviados al Z8612. Este último interpreta la información y hace un eco que es enviado a la computadora, pasando por U8 y U2. A la salida de U6 (pin 11), la señal es acondicionada nuevamente al nivel RS232-C. La respuesta de la computadora pasa a través de U3, U7, U12 y U6 sin alterarse su contenido.

PUERTO INTERNO DE CONTROL

Unas palabras más acerca de este circuito integrado (U10), mostrado en el diagrama #5. Las señales de salida de este puerto, son de vital importancia para el correcto funcionamiento del SIS Z8. El acceso a él es directo, haciendo escrituras a memoria de programa, a cualquier localidad entre las direcciones $E000_H$ y $FFFF_H$. Esto tal vez se pueda interpretar como un desperdicio, si no se toma en cuenta que con este puerto se controla buena parte de la lógica del SIS Z8, sin necesidad de sacrificar alguno de los puertos del Z8612, además de que 2 pins de este puerto quedan disponibles para futuras expansiones. Una observación importante es que se deben evitar lecturas de memoria a las localidades $F000_H$-$FFFF_H$ (tales como despliegue de memoria, etc.), ya que este puerto no está protegido contra lectura, y esta acción resulta en el bloqueo del SIS Z8. Las 2 señales que este puerto controla y que no han sido estudiadas, HT y HBP, serán explicadas en su oportunidad, en la sección de puntos prueba y lógica de Trace.

OPCIONES.

El baud rate y la opción de paridad con la que debe trabajar el SIS Z8, es determinada por el SCAN Z8 en su rutina de inicialización, leyendo los Dip switches a través de U38, localizado en la dirección $E000_H$ de memoria de programa. La Figura (5.12), indica la posición correcta de los switches para determinar las opciones iniciales del SIS Z8.

PUNTOS DE PRUEBA Y LOGICA DE TRACE.

Tal vez uno de los problemas más difíciles en el diseño del SIS Z8, fué la solución al control y rompimiento de la ejecución del programa de prueba. La solución vía programación no fué posible ya que el conjunto de instrucciones del Z8 no contiene instrucciones de un solo byte del tipo Restart (RST), indispensables para ejecución de BP por el camino del Software.

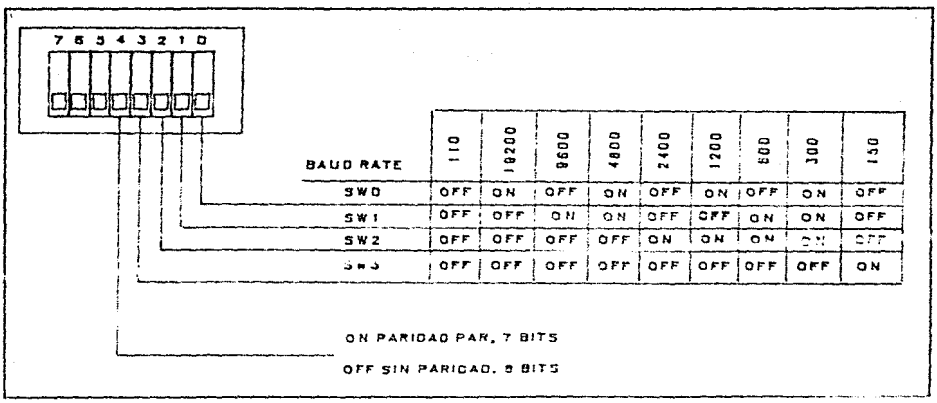


FIGURA (5.12)

La alternativa escogida es por medio de circuitería externa, en conjunto con un programa muy elaborado para identificación de Break Points y respaldo de registros del programa de prueba.

La esencia de un Break Point es reemplazar el código de una sola localidad de memoria, indicada por el usuario del sistema de desarrollo, por otro código que al ser interpretado regrese el control al programa monitor. Los microprocesadores con instrucciones del tipo restart de solo un byte, ofrecen la posibilidad de ejecución de puntos prueba, comodamente por el método de programación y por consiguiente no requieren de lógica externa para detener el flujo de un programa de prueba. De cualquier manera el código que reemplaza a la instrucción en un punto prueba, debe tener longitud máxima de un byte, en caso contrario, aunque la instrucción a reemplazarse ocupe solo una localidad de memoria, por lo menos el primer byte de la siguiente instrucción sería modificado.

El comando BP del SCAN Z8, reemplaza la localidad indicada por el usuario, por una instrucción NOP (FFH) y almacena el

código encontrado en esa localidad en memoria de datos, en una tabla de Puntos Prueba. El circuito secuencial formado por las AND U37, U36 y el contador U35, decodifica un NOP del bus de datos cuando el primer ciclo de búsqueda de instrucción ocurre. Al cumplirse que el primer byte de una instrucción sea un FFH, el pin 12 de U36, cambia de nivel y produce una petición de Interrupción IRQ0, a través de la NOR U13. El resto se ejecuta por programación y puede verse en el siguiente capitulo, en la rutina de servicio de IRQ0.

Para entender el funcionamiento de este circuito, es conveniente observar la siguiente figura, que muestra el diagrama de tiempo del ciclo de búsqueda de instrucción, similar al de la Figura (2.6) del capitulo 2, donde se ha añadido la señal SYNC y la operación del contador U35.

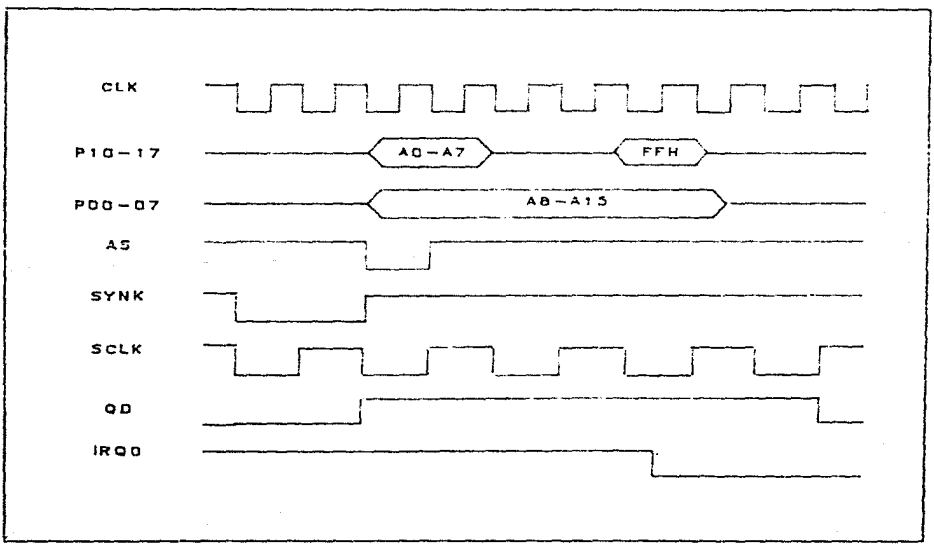


FIGURA (5.13)

La señal SYNC, controla la entrada load del contador, que es programado por sus entradas A, B, C y D, cuando SYNC es baja. El contador es temporizado por la señal ICLK, propia del Z8612. En el flanco positivo de SYNC, las salidas QA, QB, QC y QD del contador tienen valores de 1, 0, 1 y 1 respectivamente. El nivel alto de QD habilita las entradas T y P del contador, con lo cual comienza la cuenta con el siguiente flanco positivo de ICLK. QD también habilita a la AND U36. Por otro lado la señal HBP, del puerto interno de control debe estar en alto para permitir la interrupción.

Al aparecer el código FF_H en el Bus de datos, todas las entradas de las AND, están en alto y por consiguiente sus salidas, inclusive U36 pin 12. El 1 lógico en esta salida produce un cero en U13 pin 13, que al ser muestreado produce la petición de interrupción IRQ0.

Si el byte del bus de datos es distinto a FF_H, por lo menos una de las salidas (8, 12 ó 6) de U37, debe ser baja y la salida de la NOR U13 no cambia. Si la instrucción es mayor de 1 byte, al comenzar el siguiente ciclo de máquina, QD tiene un nivel bajo y deshabilita al contador y a la AND a la que se encuentra alambrada, por consiguiente en los subsecuentes ciclos no puede producirse la interrupción. Esto evita que un dato FF_H, de una instrucción cualquiera, produzca el rompimiento de la ejecución del programa. QD se vuelve a restablecer después del flanco positivo de SYNC y solo puede producir la interrupción en instrucciones de un solo byte.

Por otro lado la lógica de Trace, que es la opción de rompimiento continuo (instrucción por instrucción) de la ejecución del programa, consiste del FLIP-FLOP U39. Este es habilitado por la señal HT del puerto de control interno. La entrada PRESET de U39 está conectada a un circuito retardador, que inicializa al FLIP-FLOP.

SYNC maneja al reloj del FLIP-FLOP e IACK lo borra. El funcionamiento es el siguiente: Si HT es baja, cada flanco positivo de SYNC reafirma el nivel bajo de Q. Cuando HT es puesto en 1, el flanco positivo de SYNC cambia el nivel de Q a 1, cambiando al pin 13 de U13 a cero y produciendo la petición de interrupción. El nivel alto de Q permanece hasta que la petición de interrupción es aceptada y la señal IACK presenta un flanco positivo, con lo cual el FLIP-FLOP se reestablece automáticamente. Cuando se emplea la opción de TRACE en la ejecución de programa de prueba, el SCAN Z8 programa al puerto de control interno con HT alto. Sin embargo, esto no produce rompimiento de la ejecución del SCAN Z8, ya que éste deshabilita previamente las interrupciones.

Como la última instrucción del SCAN Z8, antes de la ejecución de un programa de prueba, es un IRET, las interrupciones son habilitadas automáticamente. El usuario debe tener cuidado de no deshabilitar globalmente las interrupciones, ni a IRQO individualmente, en el registro IMR, esto es, si es que desea trabajar cualquiera de las opciones TRACE o PUNTOS PRUEBA.

Hasta aquí con el Hardware del Sis Z8. Falta ahora la otra mitad de su funcionamiento: El Software del SCAN Z8, del cual nos ocuparemos en el siguiente capitulo.

CAPITULO VI

EL SCAN Z8

Este capitulo está dirigido a la documentación y descripción detallada del software del SIS Z8; el enfoque otorgado a esta parte es técnico y tiene como finalidad respaldar el desarrollo y diseño de las rutinas del sistema, para su posterior depuración.

Es indiscutible que en todo trabajo que involucre a la ingeniería de software, es necesaria la manifestación de todo documento que relacione diseño y programación, mas aún si contiene una poderosa interacción con hardware, por ello, en el presente capítulo se provee una explicación de la estructura del programa que controla al SIS Z8, la descripción de los comandos y rutinas más importantes, los diagramas de flujo generales y el listado de lo que hemos llamado SCAN Z8.

La utilidad operativa de las rutinas, instrucciones y comandos, concierne al capítulo cuatro. Un breve repaso a dicho capítulo, y otro más al conjunto de instrucciones, permitirá una mejor comprensión del programa, dado que, a pesar de la documentación "in situ", el lenguaje ensamblador del Z8 implica relativa dificultad interpretativa.

ESTRUCTURA Y DISEÑO DEL SCAN Z8.

Las características propias del ensamblador Z8, obligan a llevar cierto orden lógico de bloques de instrucciones, este orden es complementado con el diseño y necesidades de programación del monitor-depurador y rutinas de auxilio del SCAN Z8. Aunado a lo anterior, la dificultad de mantener el programa en un solo bloque de memoria, obliga a establecer una separación que nos llevará a nombrar un monitor interno, cuya localización lógica es la memoria que corresponde a la ROM interna del Z8, y uno mas llamado monitor externo en localidades de programa externa.

El diseño obtenido es esquematizado en la Figura (6.1).

ESTRUCTURA DEL SCAN Z8

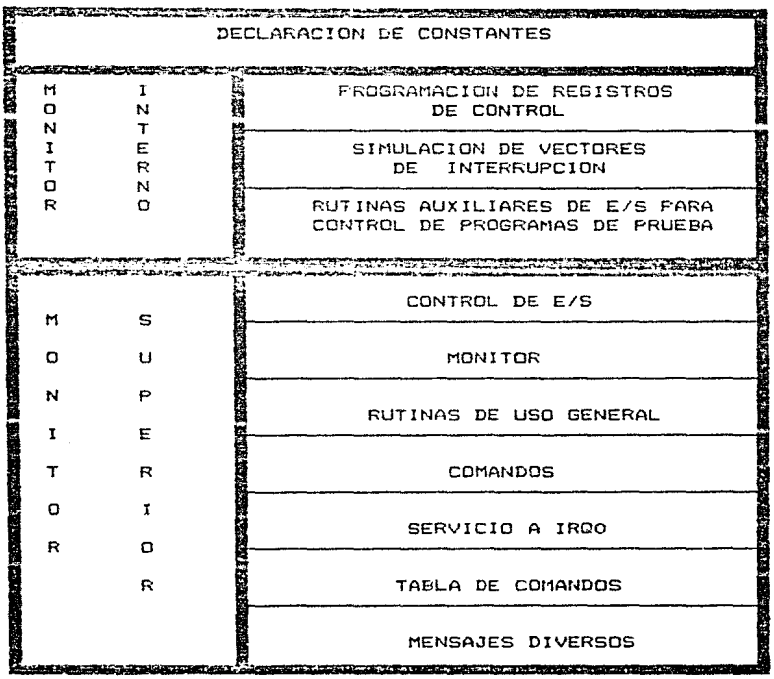


FIGURA (6.1)

La declaración de constantes permite una mejor interpretación del código fuente, ya que los datos numèricos son sustituidos con nombres genèricos a manera de nemónicos.

El monitor interno realiza, entre otras funciones, la programación de registros de control necesarios para la operación y control propios de la inicialización del Z8. La localización del mismo, es entre las direcciones 0000_{16} y 0200_{16}.

Las rutinas auxiliares de control sirven como apoyo para atender los procesos del usuario.

El monitor superior comienza en la localidad 3000_{16} y posee el mayor control del SCAN Z8. En él se incluyen el monitor, las rutinas de uso generalizado, las rutinas que ejecutan los comandos de SIS Z8 y los comandos del depurador. En muchas de estas rutinas se aplican controles de E/S, control de interrupciones y control de status. Los últimos rectàngulos son definidos mediante reserva de memoria, siendo el primero de apoyo para el monitor (decodificación de instrucciones) y el segundo para definición de mensajes informativos y de error.

DESCRIPCION DEL MONITOR.

Esta rutina es la encargada de establecer la interacción con el usuario, ya que tiene la capacidad de reconocer comandos y pasar el control a éstos, en función de los deseos del usuario.

Al inicializar o encender el SIS Z8, la lectura del programa comienza por la localidad 0000_H. A partir de èsta hasta la 0026_H se ejecuta una rutina, cuya función es inicializar los registros de control pertinentes, para permitir básicamente, el uso de memoria externa y realizar funciones preoperativas del monitor.

Cabe destacar que el registro apuntador de registros, es programado con el grupo 1, el cual será empleado en todo el SCAN Z8 como el grupo de trabajo, mientras que el stack queda localizado en registros y a partir de la dirección 40_H.

Estas operaciones continúan aún en el monitor superior con las rutinas INIT y IS, cuyas funciones son inicializar el archivo de registros alterno y reconocer de microinterruptores (DIP SWITCHES) la velocidad de transmisión y paridad con que el SIS Z8 debe operar para una adecuada transmisión. Al respecto de èsta última se hablará un poco más adelante.

Al concluir las funciones preoperativas, comienza el monitor, cuyos diagramas de flujo se muestran en las Figuras (6.2).

Este consiste de las rutinas ROOT, AGUANTA y RECEPCION.

RUTINA ROOT.

Una vez establecido el medio ambiente de control, es la rutina que espera hasta recibir un caracter por medio del puerto serial, esta recepción iniciará el reconocimiento de comandos del monitor.

ROOT es simplemente un proceso cíclico, que solo es concluido cuando la rutina RECEPCION reconoce un comando completo de la consola y pasa el control a èste.

RUTINA AGUANTA.

Esta rutina se encarga, mediante el muestreo del registro de interrupciones IRQ, de detectar y procesar un caracter cualquiera, en el momento en que ha sido ensamblado en el buffer de recepción del SIO. Esta rutina no permite que un caracter produzca interrupciones, ya que en el registro IMR se han deshabilitado individualmente todas las peticiones de interrupción. El caracter ensamblado es depositado en R2.

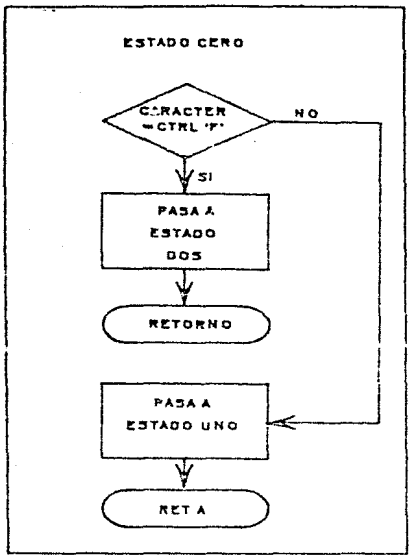
RUTINA RECEPCCION.

Es la rutina màs importante del monitor, ya que es la que permite interactuar con la terminal; controla y verifica las secuencias de caracteres recibidos para darles la debida interpretación. Funciona como el SCANNER de un compilador , y se auxilia de un decodificador para la ejecución de los distintos comandos y rutinas con que cuenta el sistema.

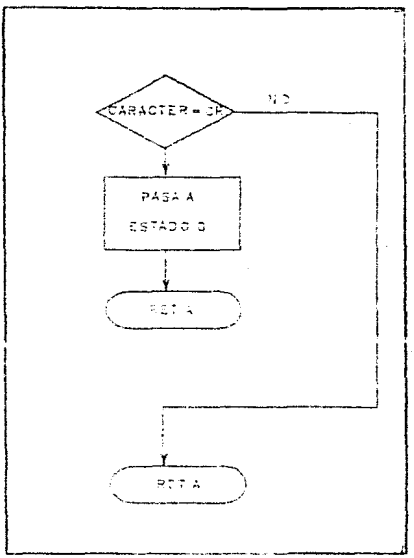
Una vez que la rutina AGUANTA ha detectado un caracter, se manifiesta la intervención de esta rutina por el eco, que èsta efectúa del caracter recibido. En el estado inicial (CERO) de RECEPCION se interpreta el caracter introducido; èste determinará el estado siguiente, siendo el Estado Dos si se tratara de un código ASCII ACK (06_H ó CTRLF), y el Estado Uno en caso de cualquier otro.

En el Estado Uno se seguirán recibiendo caracteres indefinidamente hasta recibir un código de CR para regresar al Estado Cero, contando en todo momento con la operación de retransmisión (eco) de los caracteres recibidos.

En el Estado Dos verifica la clave de acceso al recibir un CR después de CTRLF. En este caso procede al Estado Tres y cambia al modo 1 de operación del puerto serie del SIS Z8, interactuando con la terminal e ignorando la operación de la computadora anfitriona. De esta manera el SIS Z8 esta listo para recibir instrucciones y este modo es el que permitirá depurar, probar y desarrollar programas con el MCUZ8. En el caso de no recibir un CR a continuación del CTRLF, la rutina regresará al Estado Uno.



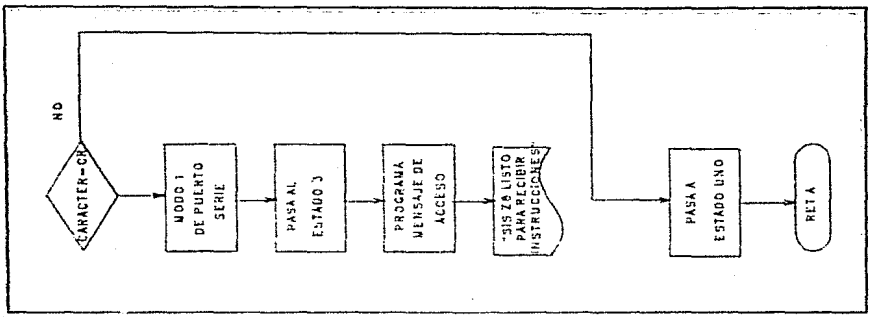
ESTADO CERO



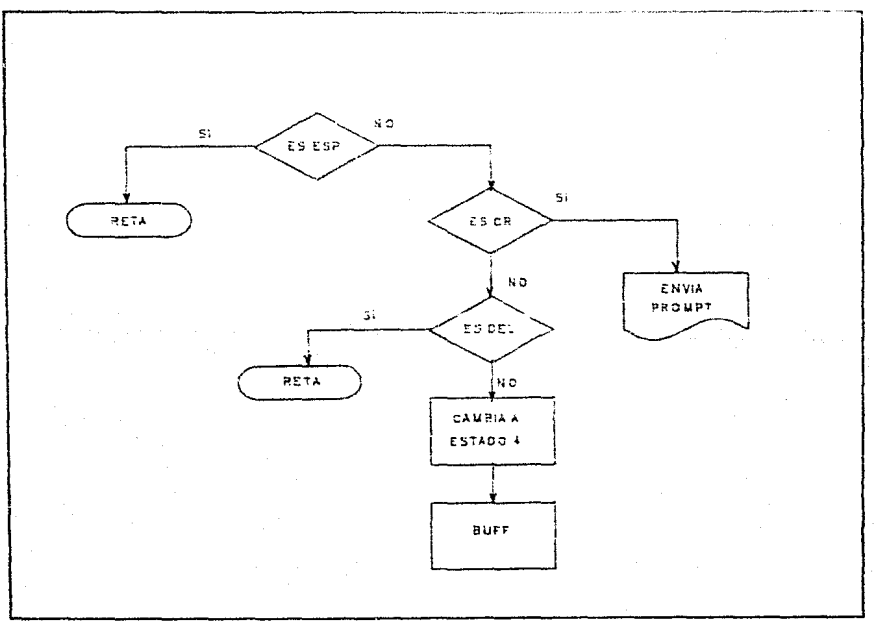
ESTADO UNO

El Estado Tres de la rutina, espera por codigos distintos a CR y ESP (20_H, ASCII) para cambiar de estado. Una vez que recibe un caracter distinto a los mencionados, la rutina cambia a su Estado Cuatro, en el cual se reciben caracteres hasta recibir un CR, indicativo del término de la instrucción. Cabe mencionar la posibilidad de corrección a la secuencia recibida, dicha corrección se realiza con el código DEL ($7F_H$) a manera de espacio en retroceso. Todas las letras minúsculas recibidas son cambiadas internamente a mayúsculas y toda secuencia de caracteres es introducida en un buffer de lectura apuntado por el registro R0.

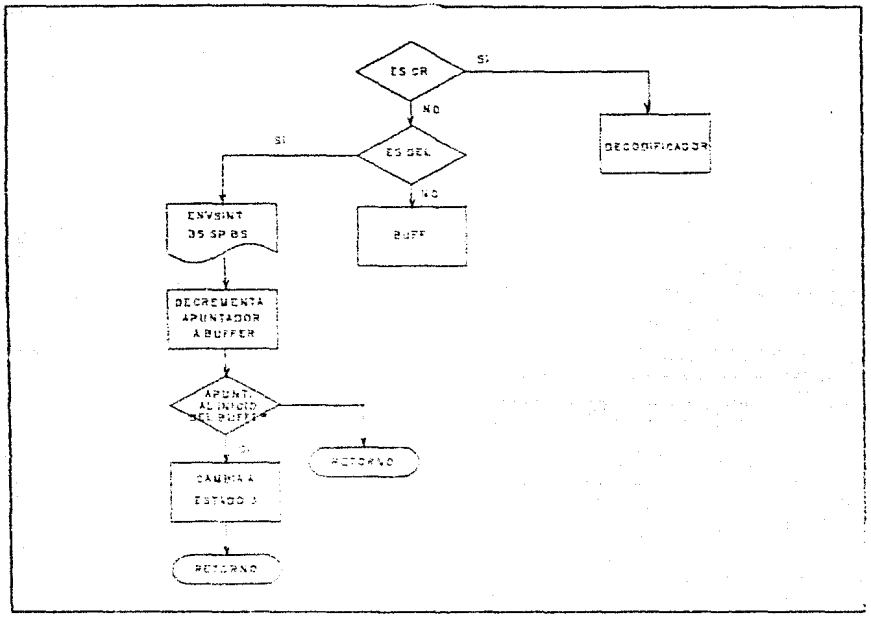
Cuando se recibe una instrucción completa se comprueba su validez a través de un decodificador de instrucciones. El decodificador hará el papel de parser, auxiliado por una tabla predefinida de direcciones y comandos. El decodificador verifica la existencia en dicha tabla de la secuencia anterior a un espacio en blanco, ya que esta secuencia se interpreta como comando. Si la mencionada secuencia corresponde a alguno de los comandos predefinidos, únicamente restará saltar a la dirección



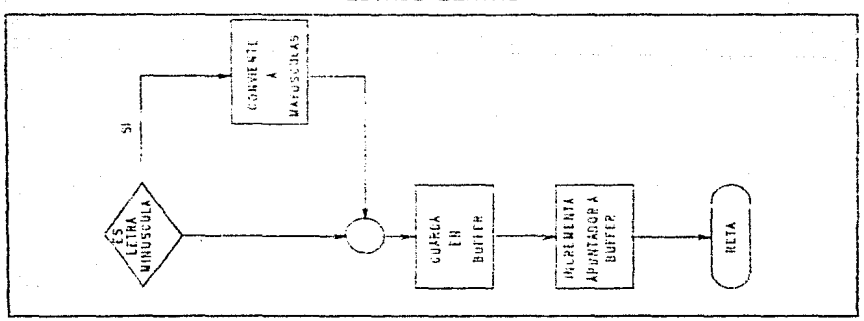
ESTADO DOS



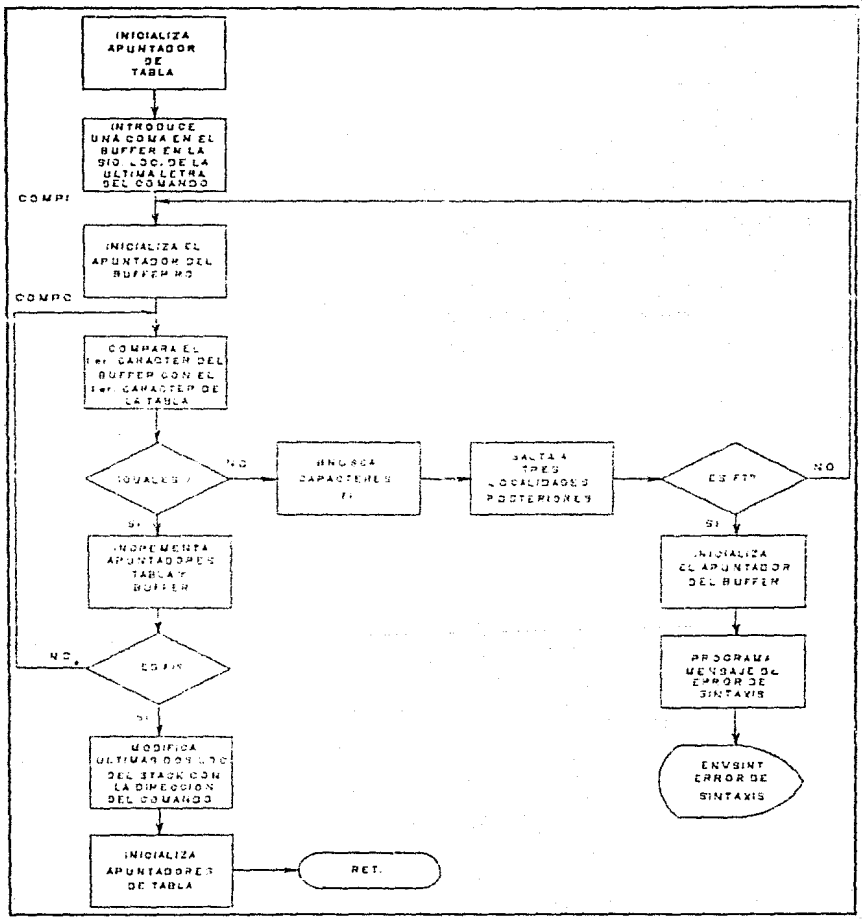
ESTADO TRES



ESTADO CUATRO



BUFF



DECODIFICADOR

contenida en dicha tabla, ya que ella indica el inicio de la rutina que controla el comando en tràmite. En caso de no encontrar la multicitada secuencia de caracteres, el monitor indicarà un error de sintaxis.

La tabla de comandos està diseñada de manera que su barrido sea fàcil de ejecutar, contiene por cada instrucción:

A	B	C	A	.....	C	D

A.- Secuencia de caracteres que definen un comando.
B.- Marca de fin de instrucción.
C.- Dirección donde comienza dicho comando.
D.- Marca de terminación de la tabla.

Al término de la ejecución de cualquier instrucción, retornarà al Estado Tres de esta rutina.

RUTINAS DEL SISTEMA.

En total existen 20 rutinas del sistema, pero solo seràn tratadas las màs importantes.

Estas rutinas cumplen funciones diversas y en general sirven de apoyo para los comandos y el monitor, a excepción de ATENT, la cual soporta el uso de interrupciones en programas de prueba.

RUTINA ATENT.

El objetivo de esta rutina es permitir el uso de interrupciones por programas de prueba. El monitor interno ocupa inclusive los primeras 12 localidades destinadas a guardar los vectores de interrupción del Z8. Al observar estas localidades en el listado, se puede notar que el primer vector (destinado IRQ0) es ocupado por una rutina que se encarga del manejo de interrupciones por BP o T.

El resto de las interrupciones disponibles son destinadas al usuario. Hay que recordar, en este momento, que el usuario debe colocar sus vectores de interrupción en memoria con un offset de 200_H, de esta manera el vector correspondiente a IRQ1, ocupará la localidad 0202_H en lugar de la 0002_H y asi sucesivamente. Al ocurrir una interrupción permitida por el usuario en su programa de prueba, por ejemplo IRQ1, el Z8 buscará automàticamente en la localidad 0002_H el vector correspondiente. Este apuntara a la rutina ATENT1, la cual se encargará de colocar en el stack del

usuario la dirección encontrada en las localidades 0202_H y 0203_H y después ejecutar una instrucción RET, con lo cual se pasará a la rutina de servicio real. De esta manera se simula sin afectar registro alguno, la operación de atención de interrupciones.

RUTINA PREROOT (Dir $32CA_H$).

Esta subrutina de apoyo permite reestablecer al sistema en un estado de espera para lectura de datos de pantalla, es la encargada de colocar el prompt del sistema, limpiar el buffer de lectura, predisponer el modo de comunicación, borrar el stack y saltar a la rutina ROOT.

Una vez concluido todo comando, esta es la rutina a la que normalmente se accesa.

RUTINA SALVAC (Salva Código Dir $334F_H$).

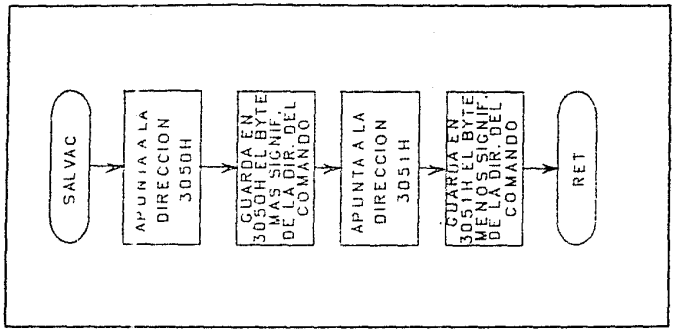
Esta subrutina auxiliar tiene la función de preservar el último comando ejecutado, es útil para el comando K (Continúa), ya que éste último comando es aplicable a tres distintos comandos (GO, CALL y DM).

La rutina salvaguarda el último comando en la dirección 3050_H con el formato mostrado en la tabla de la siguiente página.

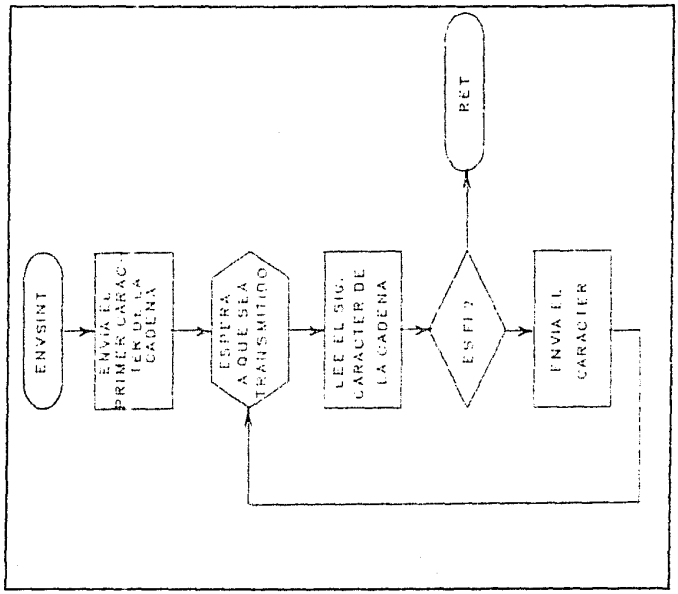
Formato:	
3050-3051	Dirección del comando correspondiente.
3052	Determina el caso de opción (D)
3053-3054	Ultima dirección desplegada por DM o ejecutada por GO o CALL antes de una interrupción por BP o T.

RUTINA ENVSINT (Dir $32A3_H$).

Esta rutina se encarga de enviar cadenas de caracteres al puerto serial. La transmisión se efectúa sin uso de



SALVAC



ENVSINT

interrupciones y termina cuando el caracter a enviar es un FF_H.

La rutina se apoya en el registro par RR6, el cual funciona como apuntador al caracter actual de la cadena en cuestión. Se auxilia, también, de la subrutina ESPERA, la cual coloca el caracter en el puerto y hace un muestreo del registro IR0 para determinar cuando el envio del caracter ha concluido.

El programa que llame a esta subrutina, debe colocar en el registro par RR6, la dirección inicial de la cadena a enviar, y ésta debe concluir con un caracter FF_H.

RUTINA HEXASC (Dir 3214_H).

Esta es una más de las rutinas de auxilio del monitor y tiene como función la conversión de código hexadecimal a caracteres imprimibles ASCII. Se auxilia del registro de trabajo R2, como parámetro de entrada y el resultado es almacenado en los registros $5E_H$ y $5F_H$.

La conversión es efectuada por medio de la separación de nibbles del registro de entrada y sumándole a cada nibble el offset correspondiente para convertir un caracter hexadecimal a un caracter ASCII 30_H para números y 37_H para caracteres.

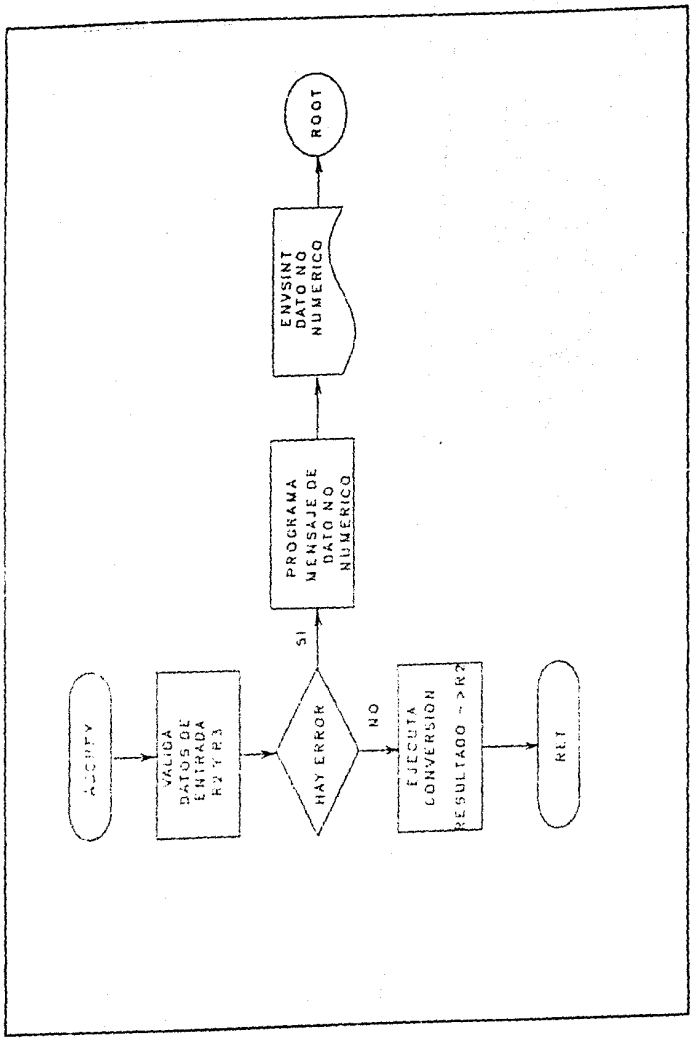
RUTINA ASCHEX (Dir $31D3_H$).

Es la rutina de auxilio que convierte el código ASCII a código hexadecimal; la operación se ejecuta sobre 2 digitos apoyándose en los registros R2 y R3 del Grupo 1 de trabajo, el resultado de la operacion es almacenado en el registro 2 del Grupo 1.

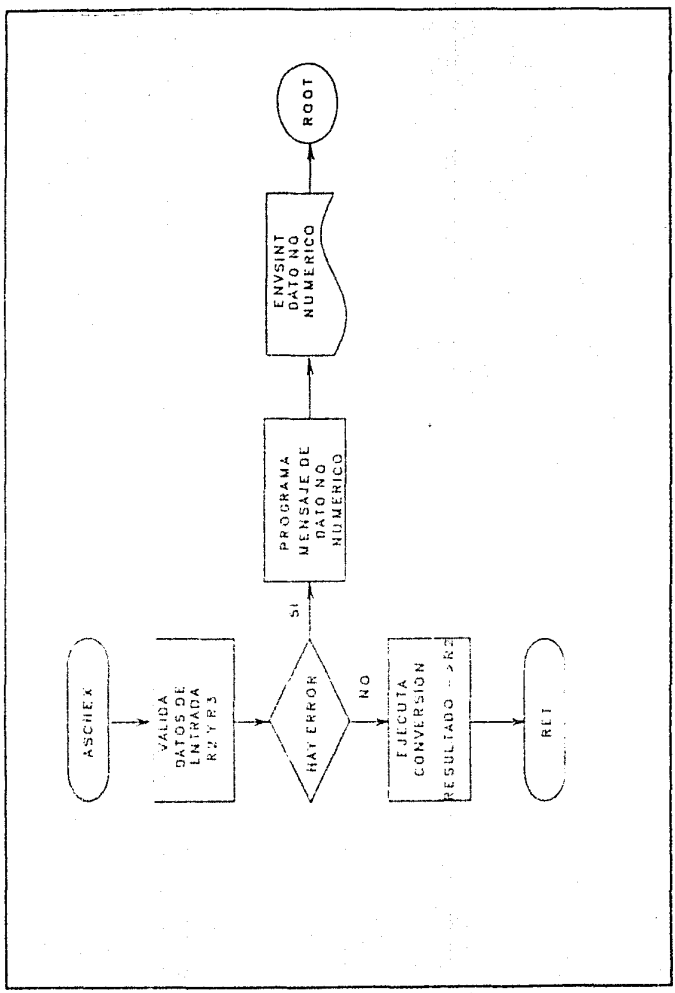
Las principales operaciones que realiza esta rutina son la verificación de los digitos leidos, es decir, los digitos a convertir siempre deben estar en un rango del 0 al 9 y de A a F; a estos digitos en ASCII se les resta un offset (30_H para números y 37_H para caracteres); posteriormente el contenido de los registros mencionados es unido en uno solo, conteniendo éste el valor hexadecimal de los digitos introducidos.

RUTINA RECDIR (Reconoce Dirección, Dir $32F1_H$).

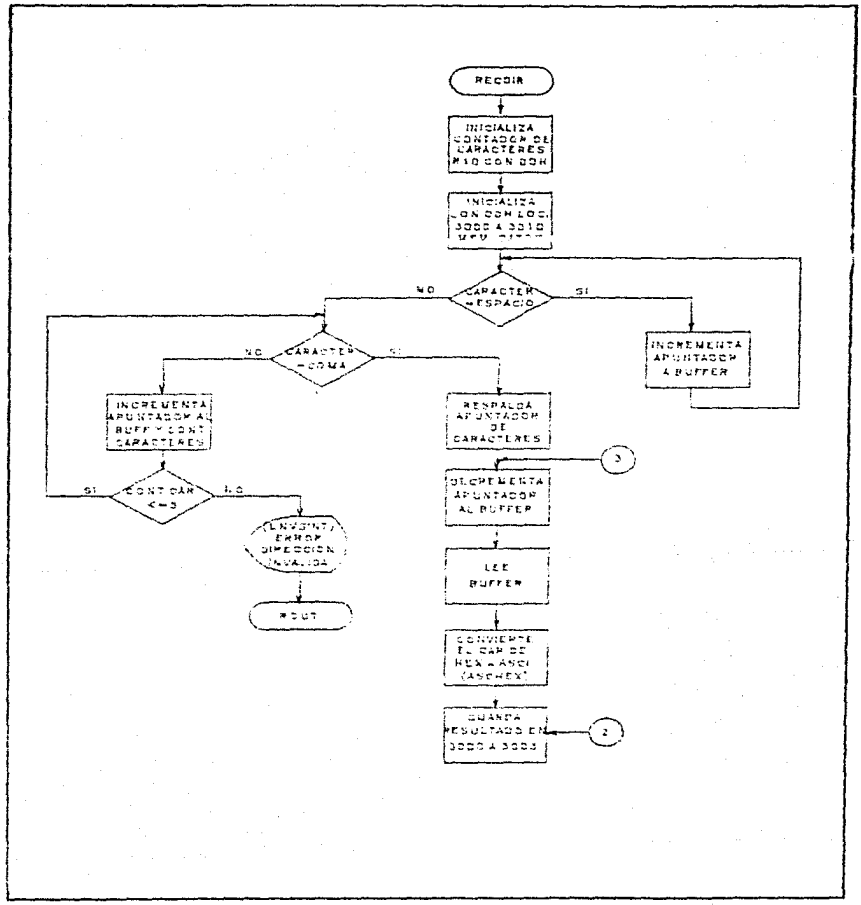
Esta rutina realiza el reconocimiento de las direcciones sobre las que actúan los diferentes comandos del SCAN Z8. Dichas direcciones son guardadas en un buffer (donde R0 es el apuntador) y recorridas, caracter a caracter, para ser reconocidas y teniendo cada dirección, como delimitador, una coma (,) .


ROOT
ENVSINT
DATO NO
NUMERICO
PROGRAMA
MENSAJE DE
DATO NO
NUMERICO
SI
HAY ERROR
NO
VALIDA
DATOS DE
ENTRADA
R1 Y R3
EJECUTA
CONVERSION
RESULTADO -> R2
RET

ASCHEX


ASCHEX
VALIDA DATOS DE ENTRADA R2 Y R3
HAY ERROR
SI
NO
PROGRAMA MENSAJE DE DATO NO NUMERICO
ENVSINT DATO NO NUMERICO
ROOT
EJECUTA CONVERSION
RET

ASCHEX


RECDIR
INICIALIZA CONTADOR DE CARACTERES #10 CON 00H
INICIALIZA CON CON LOS 3000 A 3010
NO
CARACTER = ESPACIO
SI
INCREMENTA APUNTADOR A BUFFER
NO
CARACTER = COMA
SI
INCREMENTA APUNTADOR AL BUFFER Y CONT CARACTERES
RESPALDA APUNTADOR DE CARACTERES
3
SI
CONT CAR <= 3
NO
INCREMENTA APUNTADOR AL BUFFER
(LLAMADA) ERROR DIRECCION INVALIDA
REGRESA
LEE BUFFER
CONVIERTE EL CAR DE HEX A ASCII (ASCHEX)
GUARDA RESULTADO EN 3000 A 3003
2

RECDIR

RECDIR reconoce direcciones hasta de 4 dígitos ASCII. En caso de que el usuario utilice una dirección mayor, aparecerá un mensaje de error indicando que la dirección solicitada es inválida.

Esta rutina se apoya en otras rutinas como son: ASCHEX, GUARDA, ENVSINT y PREROOT. Utiliza principalmente los siguientes registros:

- R10 como contador del número de dígitos que forman la dirección.
- RR4 como apuntador a las localidades de memoria de la 3000_H-3010_H (en donde se guardan los dígitos en ASCII que se están reconociendo).
- Los registros R2 y R3 sirven como buffer para guardar cada dígito en ASCII de la dirección y son utilizados por la rutina ASCHEX para realizar la conversión a dígitos hexadecimales, quedando el resultado de dicha conversión en R2. Posteriormente R2 también es utilizado por la rutina GUARDA, la cual deja en él, un byte de la dirección.
- RR12 como apuntador a la memoria donde quedará guardada la dirección en hexadecimal, el byte menos significativo se guarda en la primera localidad de memoria apuntada por RR12 y el byte más significativo es guardado en la localidad siguiente.

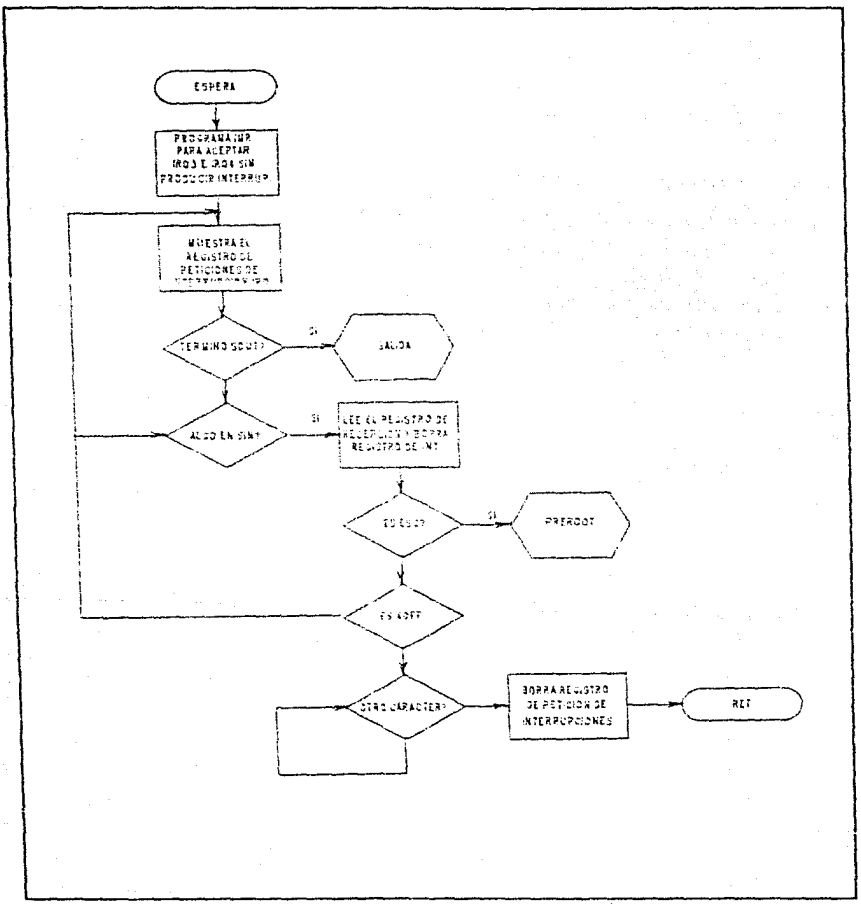
RUTINA ESPERA (Dir. 3272_H).

Es una rutina auxiliar, cuya función, mediante el muestreo del registro IR00, es deteminar el momento en que ha concluido el envío de un caracter por el SIO, a la vez que determina si otro ha sido recibido, en cuyo caso verifica si éste ha sido ESC (27_H) o CTRLS (XOFF). En el primer caso concluye la ejecución del comando, sea cual sea, y en el segundo, entra en un ciclo de espera hasta recibir un nuevo caracter. El único registro afectado es R2 en caso de recepción.

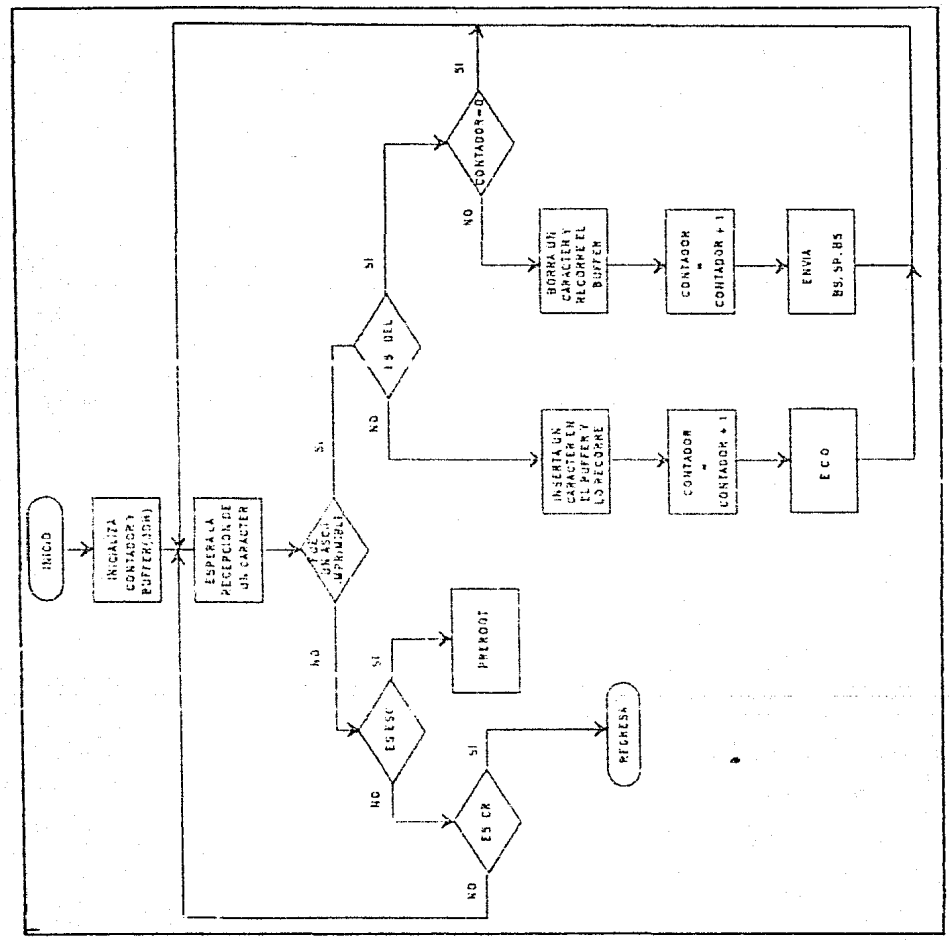
RUTINA ESPDAT (Dir. 3187_H).

Esta rutina auxiliar es empleada por los comandos MR y SM. Se trata de un pequeño monitor, cuya función es recibir un dato hexadecimal codificado en ASCII, de la consola.

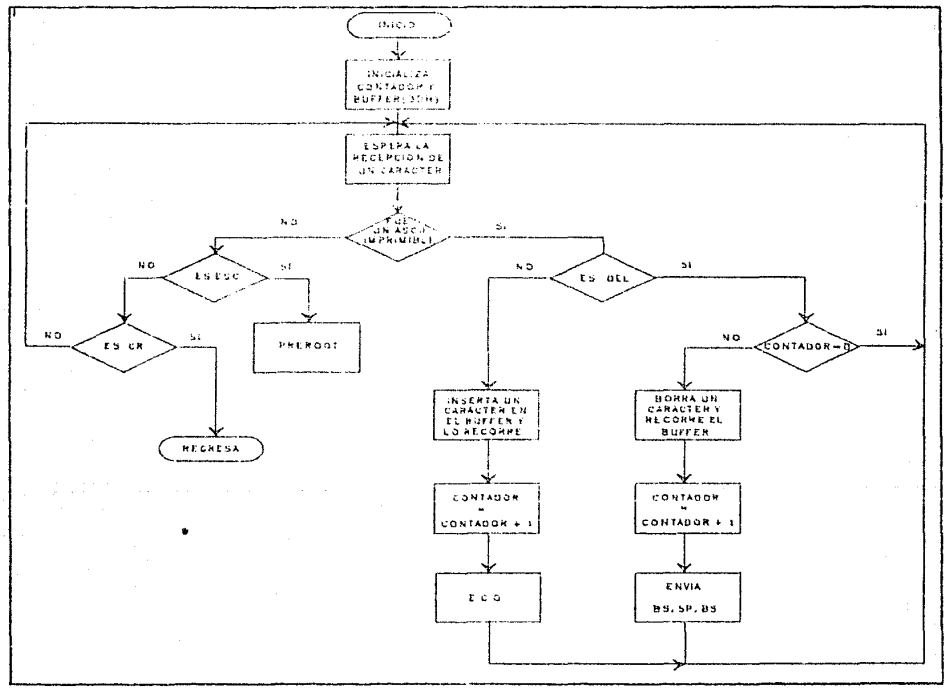
Este dato es almacenado en un pequeño buffer formado por R1,


ESPERA
PROGRAMAR IMR PARA ACEPTAR IRQ3 E IRQ4 SIN PRODUCIR INTERRUP
MUESTRA EL REGISTRO DE PETICIONES DE INTERRUPCION
TERMINO SCOT?
SI
SALIDA
ALGO EN SIN?
SI
LEE EL REGISTRO DE RECEPCION Y BORRA REGISTRO DE INT
TU ES?
SI
PREROOT
FS ASFF
OTRO CARACTER?
BORRA REGISTRO DE PETICION DE INTERRUPCIONES
RET

ESPERA


INICIO
INICIALIZA CONTADOR Y BUFFER(CONT)
ESPERA LA RECEPCION DE UN CARACTER
ES UN ASCII IMPRIMIBLE
SI
NO
ES ESC
SI
NO
PRINCIPAL
ES CR
SI
NO
REGRESA
ES DEL
SI
NO
CONTADOR=0
SI
NO
BORRA UN CARACTER Y RECORRE EL BUFFER
CONTADOR = CONTADOR - 1
ENVIA BS, SP, BS
INSERTA UN CARACTER EN EL BUFFER Y LO RECORRE
CONTADOR = CONTADOR + 1
ECO

ESPDAT


INICIO
INICIALIZA CONTADOR Y BUFFER (STR)
ESPERA LA RECEPCION DE UN CARACTER
¿QUE UN ASCII IMPRIMIBLE?
NO
SI
ES ESC
NO
SI
ES CR
NO
SI
PREPOST
REGRESA
ES DEL
NO
SI
CONTADOR=0
NO
SI
INSERTA UN CARACTER EN EL BUFFER Y LO RECORRE
CONTADOR = CONTADOR + 1
ECO
BORRA UN CARACTER Y RECORRE EL BUFFER
CONTADOR = CONTADOR + 1
ENVIA BS, SP, BS

ESPDAT

R2, R3 y el registro auxiliar R9. El registro R0 funciona como contador de caracteres.

Incorporada a esta rutina estan las funciones de detección de caracteres ESC (27_H), DEL ($7F_H$) y CR (12_H). Este último concluye la rutina y regresa al comando. ESC concluye en ROOT. Si se recibió algún dato, èste regresa en R2 y R3, En caso contrario, estos regresan con un cero ASCII en ambos.

RUTINA CRLF (Dir. $325F_H$).

Esta rutina envía los caracteres de retorno y avance de línea. Se apoya en la rutina ESPERA.

RUTINA ESPACIO (Dir. $325A_H$).

Al igual que CRLF, trabaja con la rutina ESPERA y su función es enviar un espacio en blanco (20_H ASCII).

RUTINA MODOS (Dir. $325A_H$).

Esta rutina cambia el valor del puerto interno de control. El valor deseado de los bits de control debe ser colocado en el registro R2. La rutina altera el valor de R4 para direccionar al puerto.

RUTINA DESP (Dir. $329C_H$).

Su función es enviar el contenido del registro R2 (en hexadecimal) por el puerto serie. Se apoya en la rutina HEXASC para convertir el valor de R2 a dos dígitos ASCII, en los registros $5E_H$ y $5F_H$, los cuales son introducidos en el puerto serie en la rutina OTRA.

RUTINA POD (Dir. $336F_H$).

Esta rutina sirve para los comandos con opción [D], tales como DM y SM. Reconoce del buffer de recepción de caracteres el código ASCII (D), y realiza operaciones que serán empleadas en ambos comandos, para poder escribir o leer en forma automàtica en memoria de datos o programa.

PARAK (Dir. $32BF_H$).

Esta rutina, al igual que SALVAC, sirve para guardar información útil para el comando K. En especial èsta es empleada por DM, para guardar en memoria de datos la última dirección desplegada.

XS.

Es la rutina que se encarga de la programación de la lógica externa.

La programación de la lógica externa se inicia con la lectura de datos de cualquier dirección que inicie con E ($E000_H$ hasta $EFFF_H$), ya que ésta selecciona el puerto de entrada de los microinterruptores.

El sistema cuenta con la decodificación de esta dirección para habilitar los switches programables del mismo. Los switches mantendrán el control elemental de velocidad de transmisión y la paridad manejada en la comunicación con el puerto serial.

La rutina observa el nibble alto para controlar la velocidad, obteniendo ésta a través de la lectura de los microinterruptores; la operación interna se ejecuta con n-1 rotaciones, donde "n" es el dato binario leído en los switches y colocando el resultado de las rotaciones en el registro T0 o 244.

El otro parámetro programable es la paridad; tomada del bit adyacente al nibble alto, es leído y programado en el registro 247, siendo par y non para 0 y 1 respectivamente.

Además de la programación de los dos registros de control anteriores, la rutina también actúa en la programación de los registros PREO y TMR.

La gráfica mostrada en la Figura (5.12), contiene la disposición de los microinterruptores y su interrelación con la programación.

COMANDOS.

A continuación se hará una descripción de cada uno de los comandos. Se anexan los diagramas de flujo pertinentes. La rutina de servicio a IR00, se explica al final de esta sección, pero debe recordarse que no se trata de un comando, puesto que no hay manera de accesarla por medio del monitor.

MV (Movimiento de Bloques de Memoria).

Esta rutina permite el traspaso de bloques de memoria de una dirección a otra y de un tipo a otro, hay que recordar que el Z8 manipula dos tipos de memoria: de Datos y de Programa.

Es necesario, por lo tanto, reconocer las opciones del tipo de traspaso, en otras palabras, el tipo de memoria origen y el tipo de memoria destino; para ello la rutina preestablece cuatro letras distintas, siendo

P: Transferencia de memoria de programa a memoria de programa.

D: Transferencia de memoria de datos a memoria de datos.

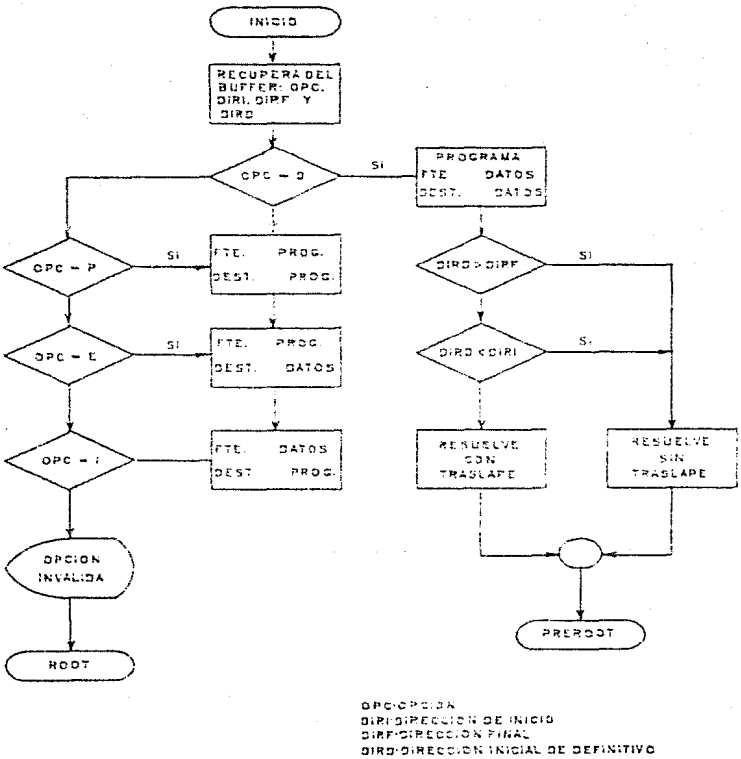
E: Transferencia de memoria de programa a memoria de datos.

I: Transferencia de memoria de datos a memoria de programa.

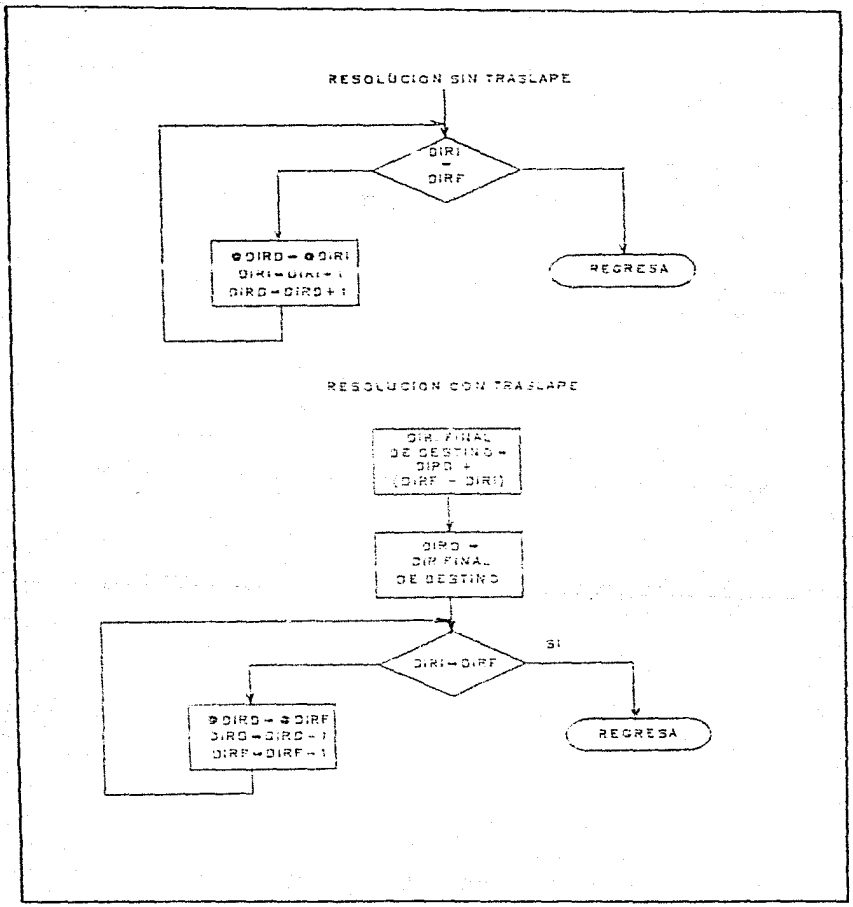
Tambien es indispensable reconocer tres direcciones: el inicio del bloque, el final y la nueva dirección destino.

Se auxilia en la rutina de reconocimiento y validación de direcciones y utiliza apuntadores a cada una de las direcciones mencionadas, interpreta las direcciones y detrmina si existe traslape para iniciar con el movimiento de contenidos.

La rutina no envia letreros, a menos que existan errores, simplemente regresa un prompt una vez que ha ejecutado la transferencia.


INICIO
RECUPERA DEL BUFFER: OPC, DIRI, DIRF Y DIRO
OPC = O
SI
PROGRAMA FTE DATOS DEST. DATOS
OPC = P
SI
FTE. PROG. DEST. PROG.
OPC = E
SI
FTE. PROG. DEST. DATOS
OPC = I
FTE. DATOS DEST. PROG.
OPCION INVALIDA
ROOT
DIRO > DIRF
SI
DIRO < DIRI
SI
RESUELVE CON TRASLAPE
RESUELVE SIN TRASLAPE
PREROOT
OPC:OPCION
DIRI:DIRECCION DE INICIO
DIRF:DIRECCION FINAL
DIRO:DIRECCION INICIAL DE DEFINITIVO

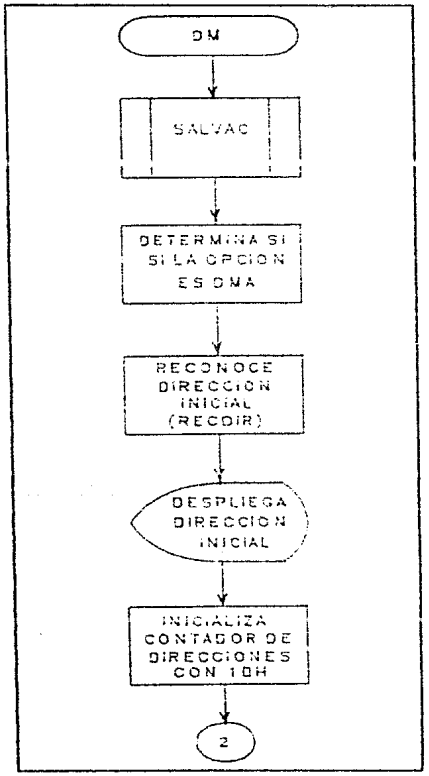
MV -I


RESOLUCION SIN TRASLAPE
DIRI = DIRF
REGRESA
RESOLUCION CON TRASLAPE
DIRI = DIRF
SI
REGRESA

MV -II

DM (Despliegue de Memoria)

Esta es la rutina que permite la observación del contenido de la memoria del sistema; sabiendo de antemano que se cuenta con dos tipos de ella, la variante DMD desplegarà memoria de datos y el comando original (DM) actuarà sobre memoria de programa. La opción es determinada por la rutina POD.



DM - I

La rutina està habilitada para reconocer dos direcciones, una inicial y otra final, reconociendo de esta manera el rango de direcciones cuyo contenido estara bajo visualización. Las direcciones no deben omitirse y deberàn estar separadas por una coma. Se apoya en la subrutina de reconocimiento de direcciones (RECDIR), la cual actua sobre datos en hexadecimal.

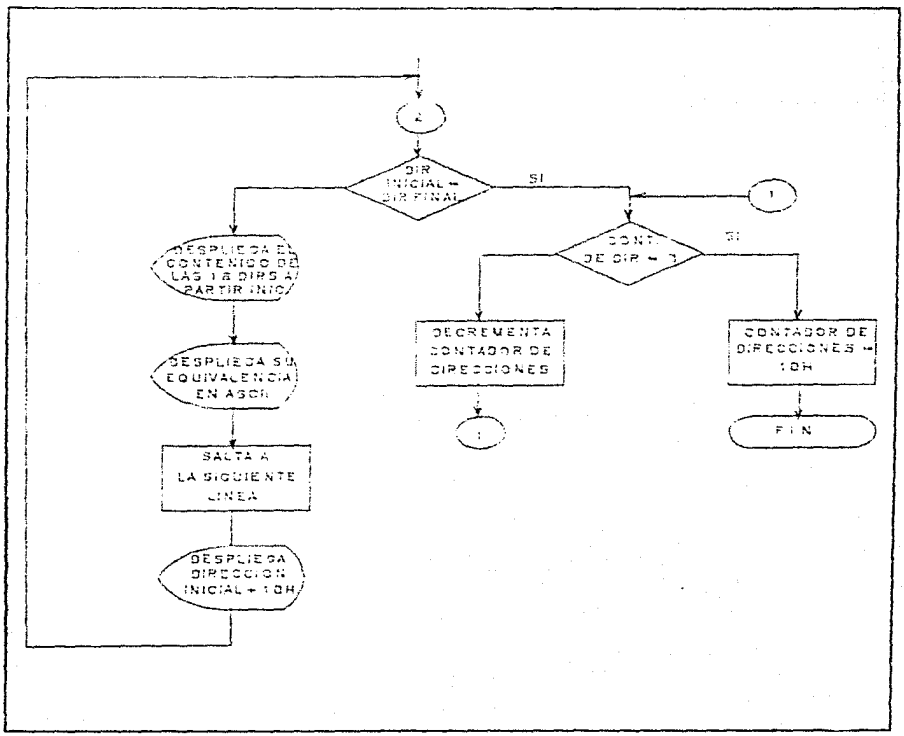
Una vez que se determina el rango de direcciones (recordar tambièn que el despliegue de direcciones entre $F000_H$ y $FFFF_H$ bloqueará al sistema), se procede al envio de información a pantalla. La rutina tiene necesidad de utilizar la mayoria de los registros del grupo uno, debido a la complejidad en la organización del despliegue de contenidos y direcciones ya que se requieren contadores y apuntadores entre otros elementos.

Se despliega la dirección inicial seguida del contenido en hexadecimal de esa dirección y 15 subsecuentes, ademàs se envia el contenido en caracteres ASCII, mandando a pantalla un punto representativo de los caracteres no imprimibles; esta operación se repite hasta cumplir con el envio de la dirección final indicada.

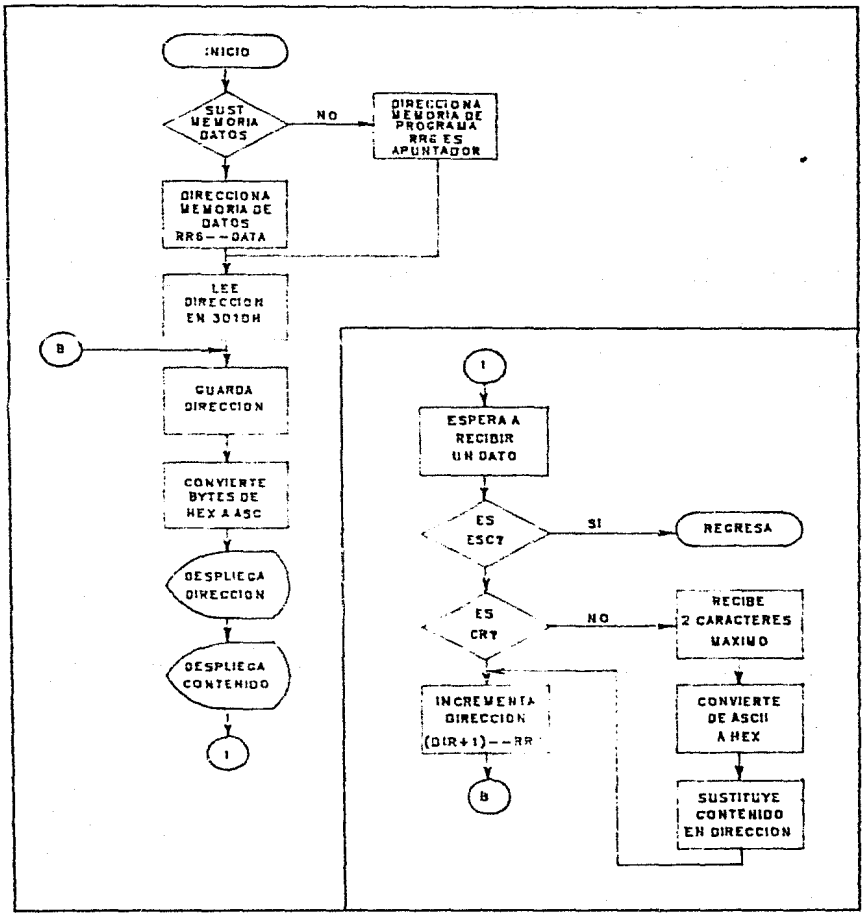
Al usar las rutinas ENVSINT

y ESPERA, este comando puede ser interferido por los caracrteres de control CTRLS y ESC, teniendo como consecuencia la suspensión momentánea y definitiva del comando respectivamente. La visualización de las siguientes 256 localidades puede ser conseguida con la rutina K (continuación).

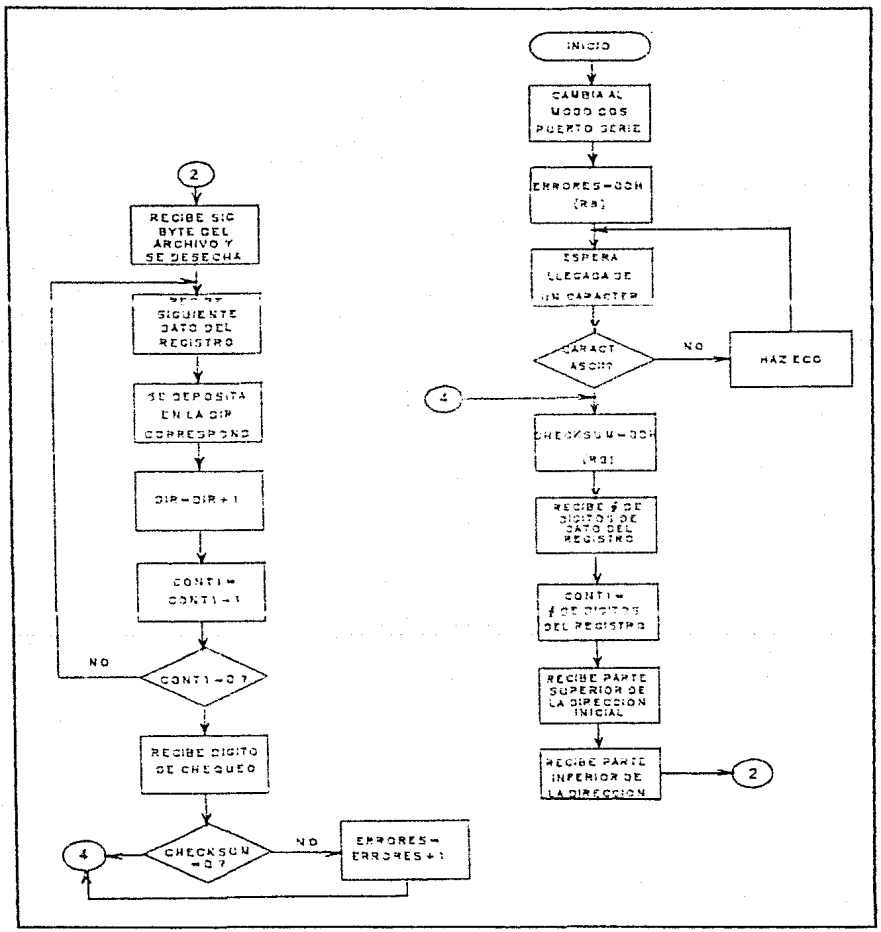
Otras subrutinas de apoyo para DM son las de envio de caracteres (ENVSINT), reconocimiento de direcciones (RECDIR) y respaldo de direcciones (SALVAC Y PARAK).



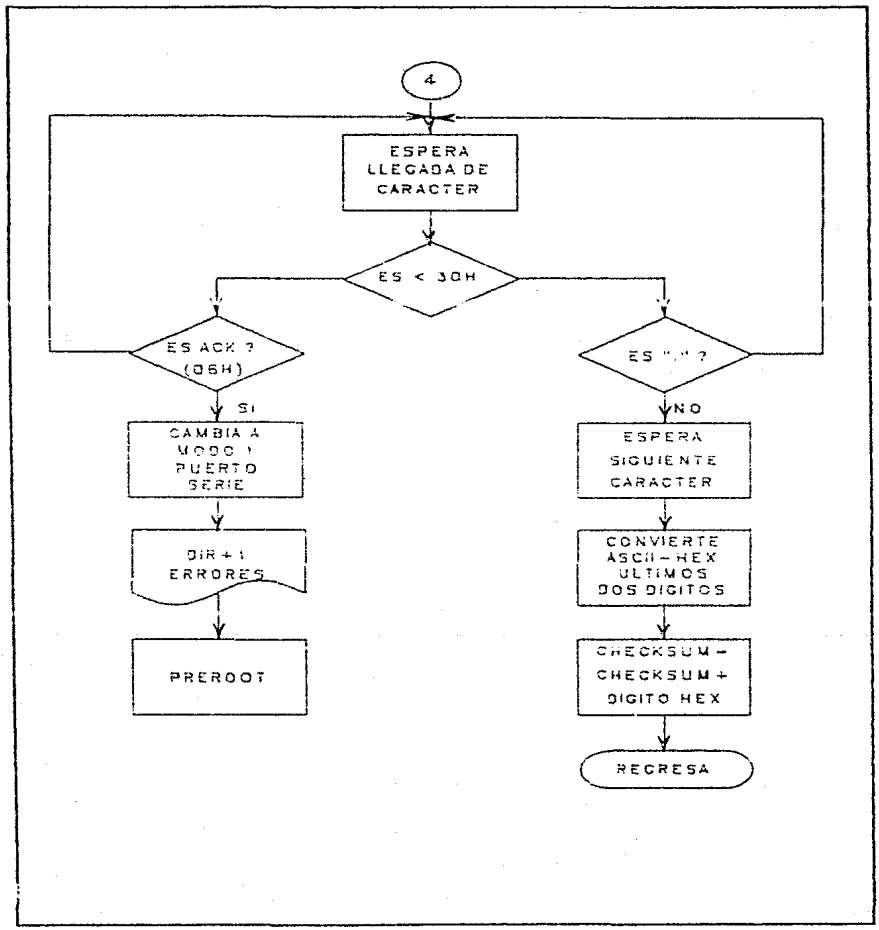
DM — II


INICIO
SUST MEMORIA DATOS
NO
DIRECCIONA MEMORIA DE PROGRAMA RR6 ES APUNTADOR
DIRECCIONA MEMORIA DE DATOS RR6--DATA
LEE DIRECCION EN 3010H
B
GUARDA DIRECCION
CONVIERTE BYTES DE HEX A ASC
DESPLIEGA DIRECCION
DESPLIEGA CONTENIDO
1
1
ESPERA A RECIBIR UN DATO
ES ESC?
SI
REGRESA
ES CR?
NO
RECIBE 2 CARACTERES MAXIMO
INCREMENTA DIRECCION (DIR+1)--RR
CONVIERTE DE ASCII A HEX
B
SUSTITUYE CONTENIDO EN DIRECCION

SM


INICIO
CAMBIA AL MODO DOS PUERTO SERIE
ERRORES=00H (R8)
ESPERA LLEGADA DE UN CARACTER
CARACT ASCII?
NO
HAZ ECO
4
CHECKSUM=00H (R9)
RECIBE # DE DIGITOS DE DATO DEL REGISTRO
CONT1= # DE DIGITOS DEL REGISTRO
RECIBE PARTE SUPERIOR DE LA DIRECCION INICIAL
RECIBE PARTE INFERIOR DE LA DIRECCION
2
2
RECIBE SIG BYTE DEL ARCHIVO Y SE DESECHA
SIGUIENTE DATO DEL REGISTRO
SE DEPOSITA EN LA DIR CORRESPOND
DIR=DIR+1
CONT1= CONT1-1
NO
CONT1=0 ?
RECIBE DIGITO DE CHEQUEO
4
CHECKSUM =0 ?
NO
ERRORES= ERRORES+1

TRM


4
ESPERA LLEGADA DE CARACTER
ES < 30H
ES ACK ? (06H)
SI
CAMBIA A MODO 1 PUERTO SERIE
DIR + 1 ERRORES
PREROOT
ES ":" ?
NO
ESPERA SIGUIENTE CARACTER
CONVIERTE ASCII - HEX ULTIMOS DOS DIGITOS
CHECKSUM = CHECKSUM + DIGITO HEX
REGRESA

TRM-I

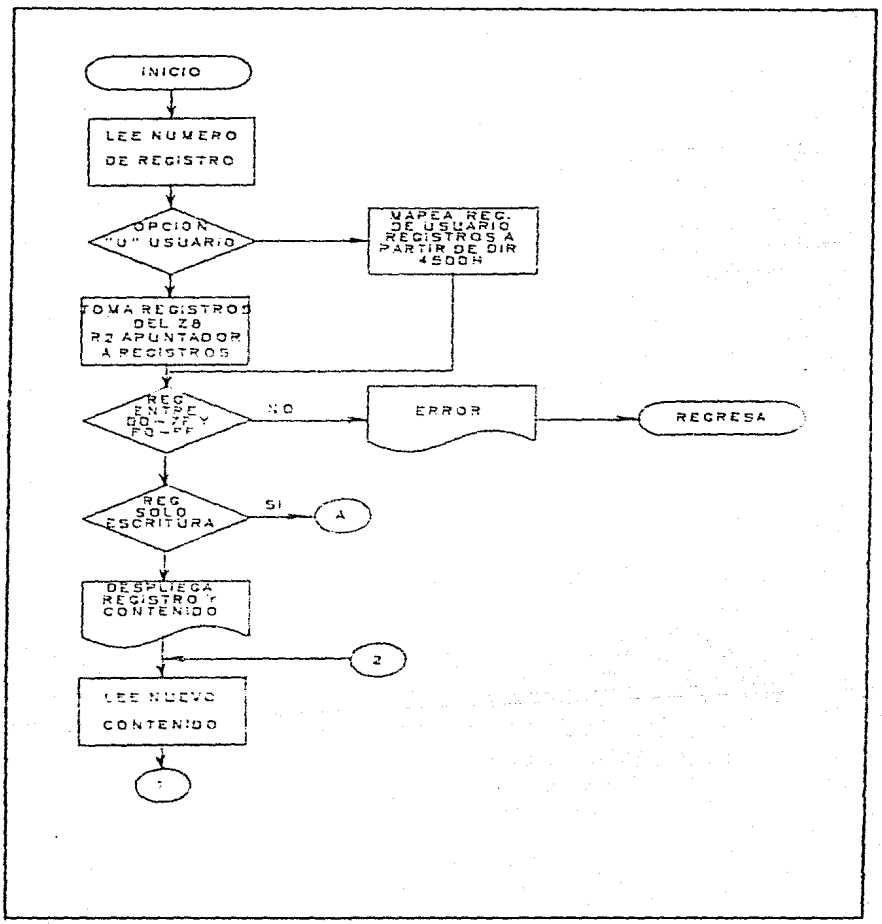
MR y MRU (Movimiento de Registros).

Esta rutina hace posible la modificación del contenido de los registros del Z8, incluyendo los de control. Considera dos alternativas: una se refiere a la modificación de registros de usuario, para lo cual hace referencia al archivo alterno de registros en memoria de datos externa, la otra opción concierne al archivo de registros interno.

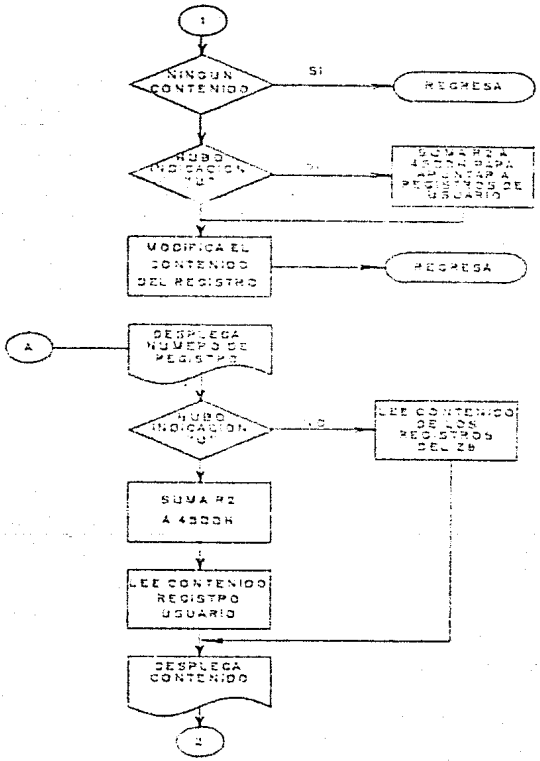
La rutina reconoce el registro solicitado junto con el comando para poder leerlo, enviar el contenido a pantalla y solicitar un dato de entrada que sustituirá al valor actual desplegado, tomando en cuenta la variación de si es registro de usuario (MRU), se auxilia por lo tanto de subrutinas tales como la de reconocimiento de registros (RECREG), rutinas de apoyo de envio de caracteres y errores, rutinas de validación de registros correctos (hay que recordar que no todo el rango de registros entre 00 y FF_H es válido) y rutinas de conversión de codigos.

Para ejecutar la modificación se requiere un apuntador al registro y al contenido que este tendrá convertido en hexadecimal en el registro R5 y R2 respectivamente.

Ninguna modificación será realizada si no se introduce ningún dato o se envia un código de escape.


INICIO
LEE NUMERO DE REGISTRO
OPCION "U" USUARIO
MAPEA REG. DE USUARIO REGISTROS A PARTIR DE DIR 4500H
TOMA REGISTROS DEL Z8 R2 APUNTADOR A REGISTROS
REG ENTRE 00-7F Y F0-FF
NO
ERROR
REGRESA
REG SOLO ESCRITURA
SI
A
DESPLIEGA REGISTRO Y CONTENIDO
2
LEE NUEVO CONTENIDO
1

MR-I


1
NINGUN CONTENIDO
SI
REGRESA
HUBO INDICACION "U"
SUMA P2 A 4300H APUNTA A REGISTROS DE USUARIO
MODIFICA EL CONTENIDO DEL REGISTRO
REGRESA
A
DESPLEGA NUMERO DE REGISTRO
HUBO INDICACION "U"
LEE CONTENIDO DE LOS REGISTROS DEL ZB
SUMA P2 A 4300H
LEE CONTENIDO REGISTRO USUARIO
DESPLEGA CONTENIDO
2

MR-II

DR (Despliegue de Registros).

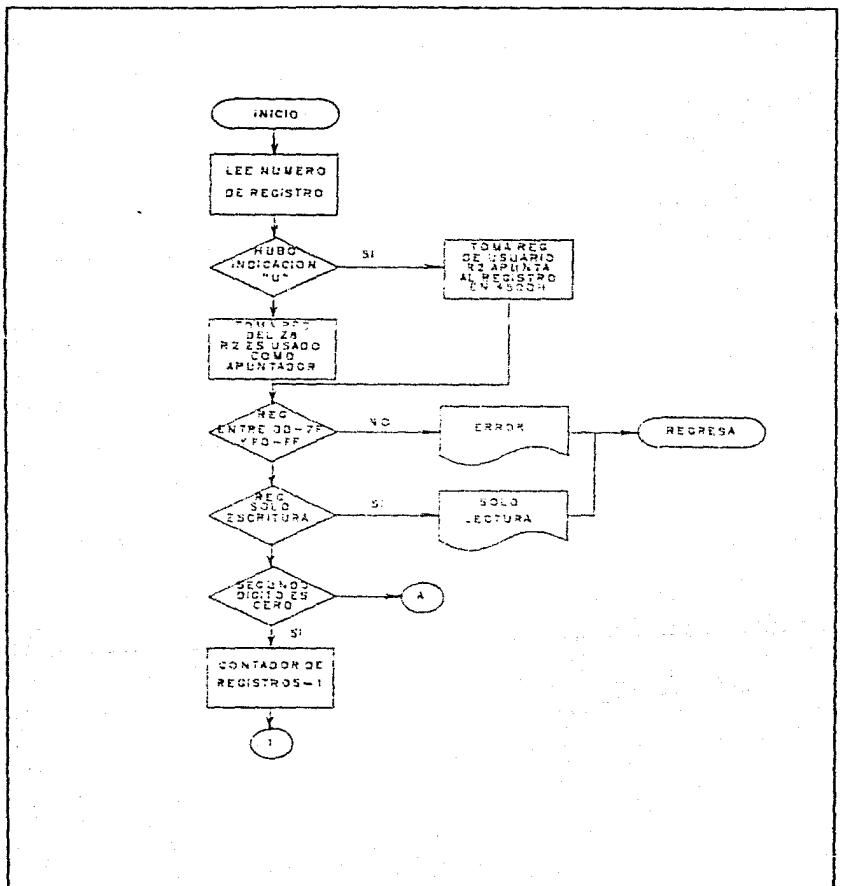
Esta rutina permite la visualización de uno o varios registros del Z8, es decir, la rutina puede enviar a pantalla un solo registro o un grupo (16), dependiendo de la indicación realizada después de las letras DR o DRU.

Maneja dos posibles opciones; la primera de ellas es la visualización de registros de usuario, manipulados en el archivo de registros alterno residente en memoria de datos externa y que corresponden al estado de los registros del archivo interno del Z8 al ocurrir una interrupción provocada por la ruptura de un programa de prueba; la segunda opción concierne al despliegue del contenido de registros del archivo interno del Z8.

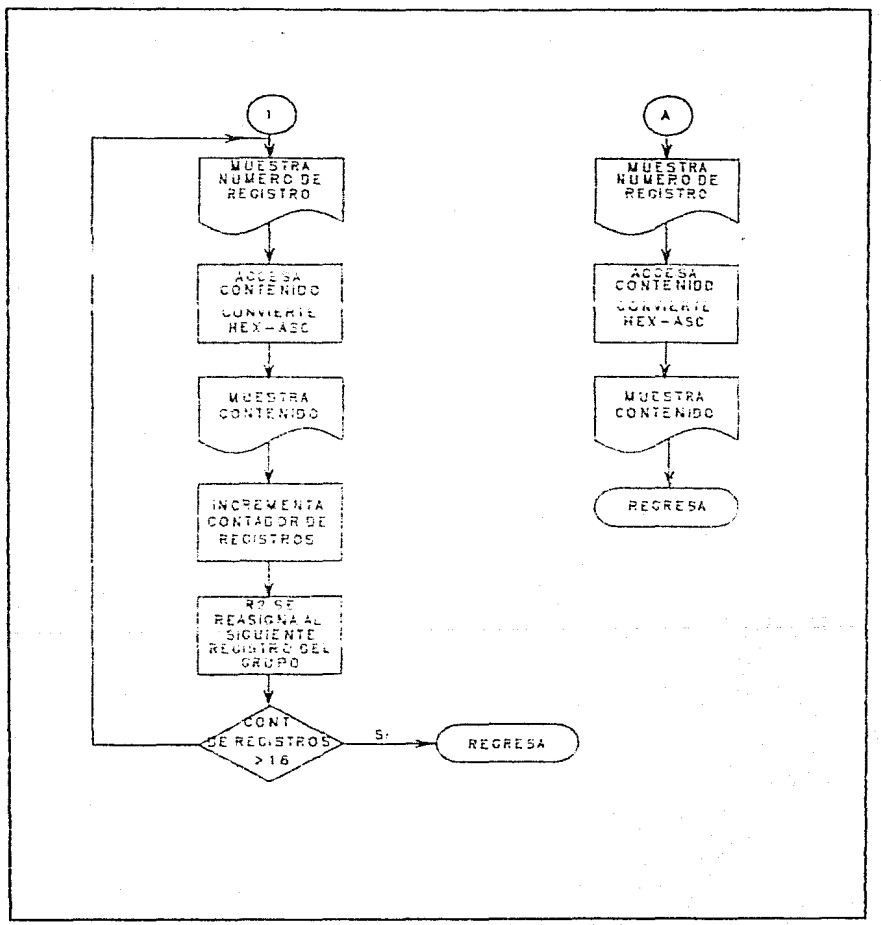
La rutina se auxilia en subrutinas como las siguientes: reconocimiento y lectura de un registro dado en forma hexadecimal, cuando el último digito de este numero sea cero desplegará todo el grupo correspondiente al primer digito, cuando no sea así se indicará únicamente el contenido del registro referenciado en forma hexadecimal; también utiliza una subrutina de validación de registros. Es necesario recordar que solo se utilizan los primeros 8 grupos y el último (F) grupo de registros, además de esta, también está previsto la utilización de rutinas preestablecidas para el envio de errores y caracteres, en este caso la de registros y su contenido.

Cuando se manipula todo un grupo, se apoya en un contador de registros (R8) para enviar todos estos.

Puede existir inconsistencia en el desplegado de registros de contol (grupo F) debido a que algunos de ellos son registros de solo escritura, mientras que otros son dobles.


INICIO
LEE NUMERO DE REGISTRO
HUBO INDICACION "U"
SI
TOMA REG DE USUARIO
DEL ZA
RZ ES USADO COMO APUNTADOR
REG ENTRE 00-7F Y F0-FF
NO
ERROR
REGRESA
REG SOLO ESCRITURA
SI
SOLO LECTURA
SEGUNDO DIGITO ES CERO
A
SI
CONTADOR DE REGISTROS-1
1

DR-I


1
MUESTRA NUMERO DE REGISTRO
ACCESA CONTENIDO
CONVIERTE HEX-ASC
MUESTRA CONTENIDO
INCREMENTA CONTADOR DE REGISTROS
R2 SE REASIGNA AL SIGUIENTE REGISTRO DEL GRUPO
CONT DE REGISTROS >16
SI
REGRESA
A
MUESTRA NUMERO DE REGISTRO
ACCESA CONTENIDO
CONVIERTE HEX-ASC
MUESTRA CONTENIDO
REGRESA

DR-II

MA (Manda)

Es la rutina que se encarga de probar el correcto funcionamiento de transmisiones a través del puerto serial.

Ejecuta el envio de todos los caracteres ASCII imprimibles, desde el espacio (20_H) hasta el "~" ($7E_H$).

Se auxilia de los registros 14 y 15 para el envio de caracteres, siendo estos desplegados hasta 30 por línea.

PM (Prueba de Memoria).

Esta otra rutina de autoprueba verifica el buen funcionamiento de Lecturas y Escrituras a memoria. Es particularmente útil para cuando se sospecha mal funcionamiento de la memoria principal del SIS Z8.

La rutina ejecuta escrituras sobre todas y cada una de las localidades de memoria comprendidas en los cuatro siguientes sectores:

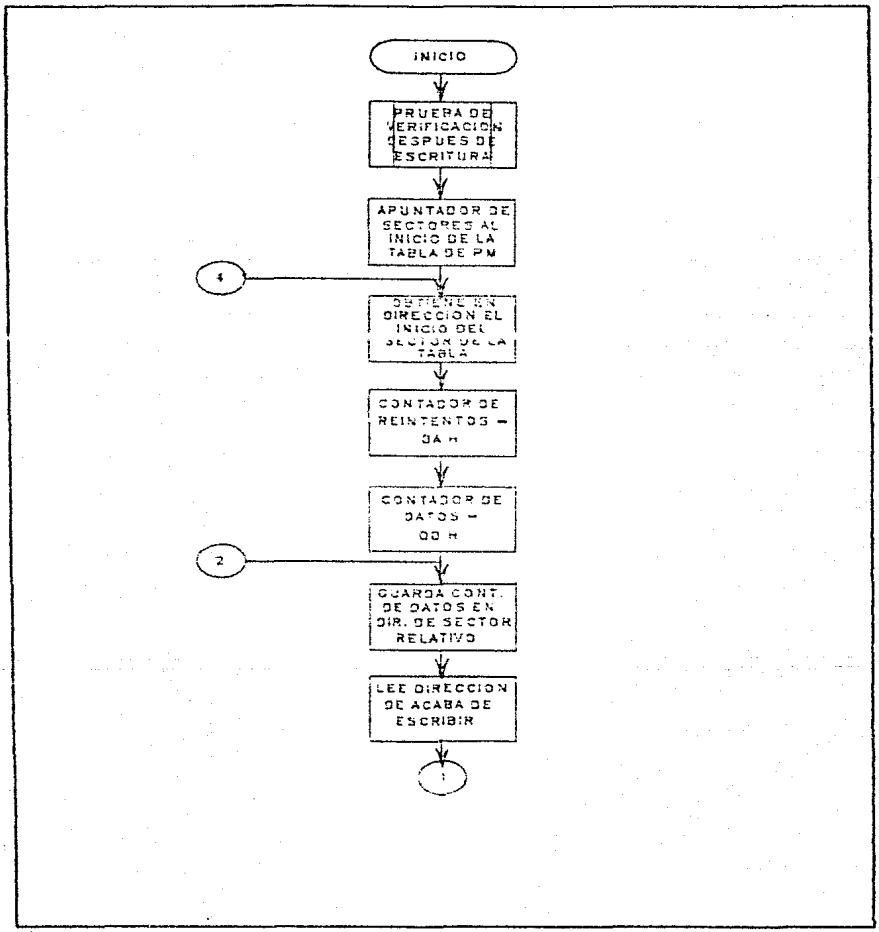
Sector 1: Memoria de Programa de 1000_H a $2FFF_H$
Sector 2: Memoria de Programa de 5000_H a $6FFF_H$
Sector 3: Memoria de Datos de 1000_H a $2FFF_H$
Sector 4: Memoria de Datos de 3000_H a $4FFF_H$

Como se escribe un solo patrón a todas las localidades mencionadas anteriormente, la verificación de tales escrituras se hace mas sencilla, después de las escrituras se procede a la lectura del contenido en los anteriores sectores, haciendo inicaciones en las localidades cuyo contenido no es el patrón: se manifiesta por lo tanto un error de acceso a tales localidades.

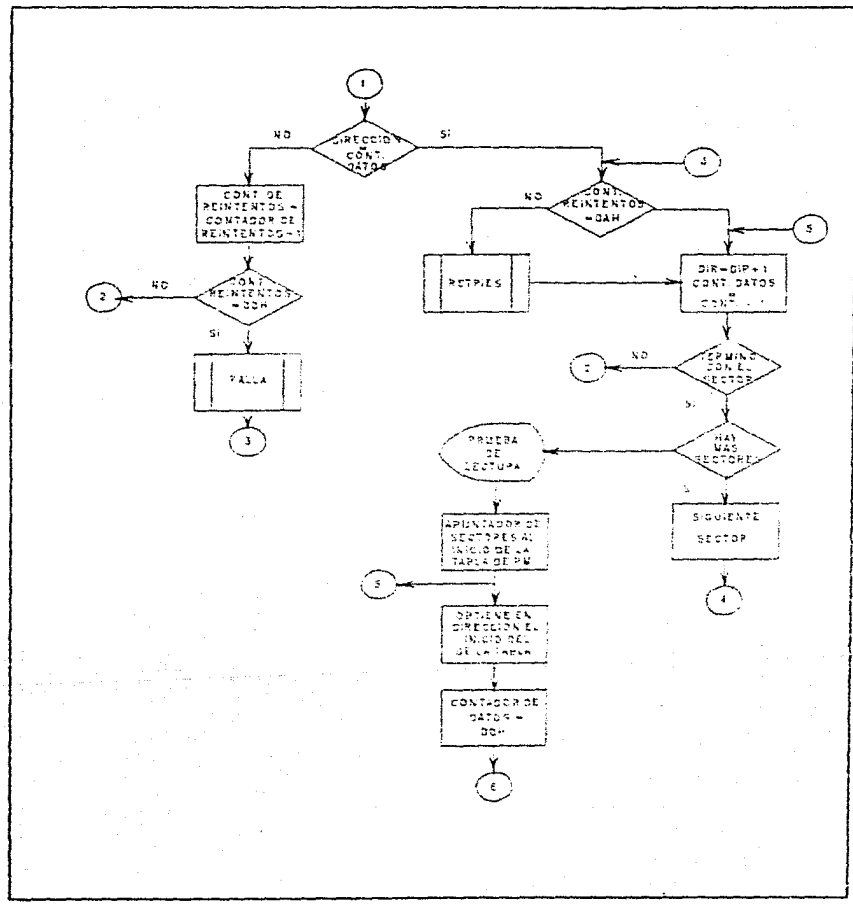
Con fines informativos, la rutina incluye el envio de mensajes a fin de indicar el progreso de la prueba, indicando el inicio de cada sector revisado.

la rutina se apoya en casi todos los registros del grupo 1 a fin de utilizarlos como contadores y apuntadoresa a memoria, control de mensajes y control de errores.

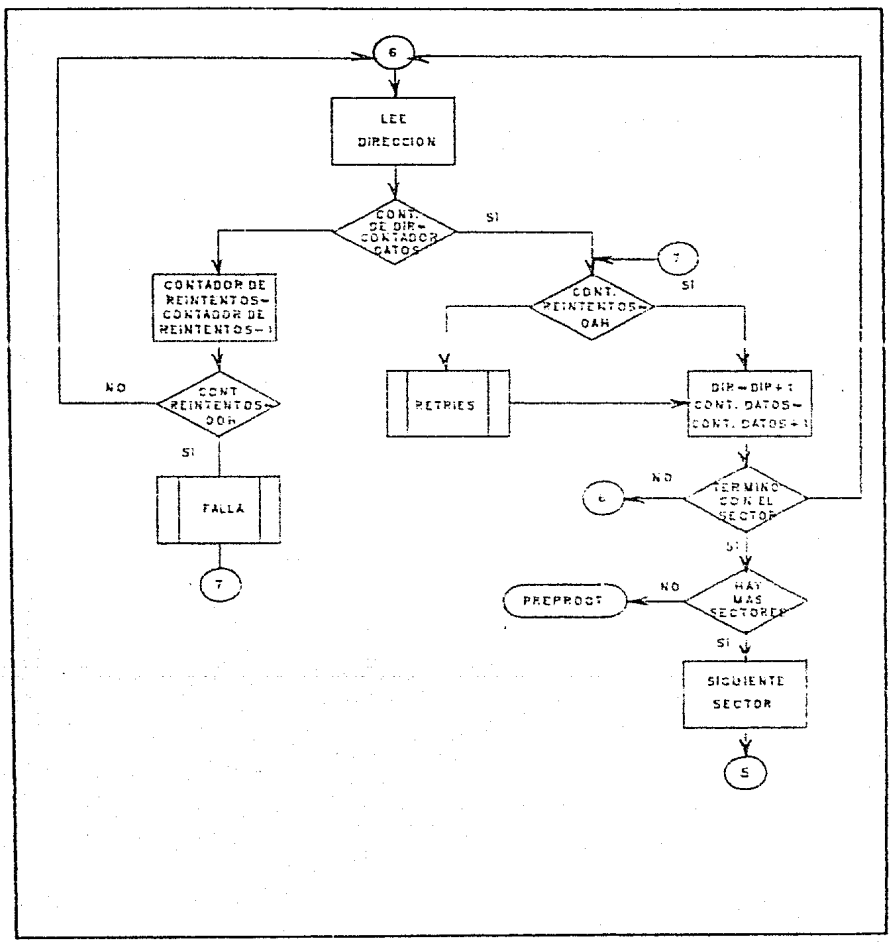
Un detalle importante de mencionar es el de que un error es manifestado después de ejecutar 10 intentos de lectura a una localidad de la que se espera un determinado contenido.


INICIO
PRUEBA DE VERIFICACION DESPUES DE ESCRITURA
APUNTADOR DE SECTORES AL INICIO DE LA TABLA DE PM
4
OBTIENE EN DIRECCION EL INICIO DEL SECTOR DE LA TABLA
CONTADOR DE REINTENTOS = 0A H
CONTADOR DE DATOS = 00 H
2
GUARDA CONT. DE DATOS EN DIR. DE SECTOR RELATIVO
LEE DIRECCION DE ACABA DE ESCRIBIR
1

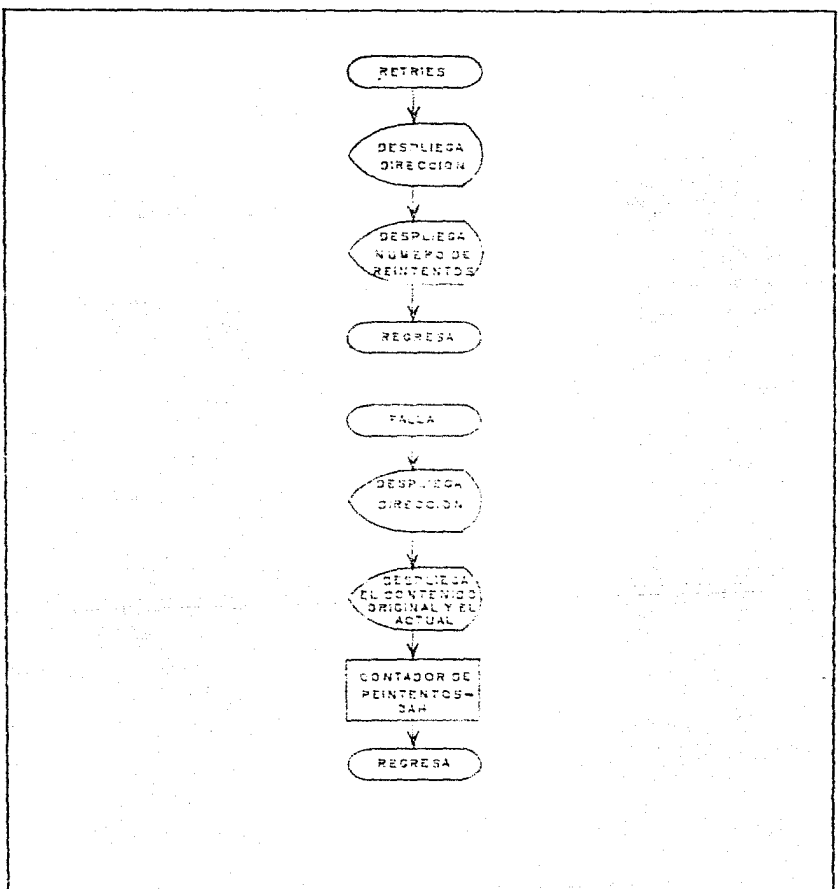
PM-I


1
NO
SI
CONT. DE REINTENTOS = CONTADOR DE REINTENTOS + 1
CONT. REINTENTOS = 00H
2
NO
SI
FALLA
3
4
NO
CONT. REINTENTOS = 00H
5
RETRIES
DIR = DIR + 1
CONT. DATOS =
CONT. - 1
6
NO
TERMINO CON EL SECTOR
SI
HAY MAS SECTORES
PRUEBA DE LECTURA
SIGUIENTE SECTOR
APUNTADOR DE SECTORES AL INICIO DE LA TABLA DE PM
7
OBTIENE EN DIRECCION EL INICIO DE LA TABLA
CONTADOR DE DATOS = 00H
8

PM-II


6
LEE DIRECCION
CONT. DE DIR = CONTADOR DATOS
SI
7
SI
CONTADOR DE REINTENTOS = CONTADOR DE REINTENTOS + 1
CONT. REINTENTOS = 0AH
NO
CONT REINTENTOS = 0DH
RETRIES
DIR = DIR + 1
CONT. DATOS = CONT. DATOS + 1
SI
FALLA
6
NO
TERMINO CON EL SECTOR
7
SI
PREPROOT
NO
HAY MAS SECTORES
SI
SIGUIENTE SECTOR
5

PM-III


RETRIES
DESPLIEGA DIRECCION
DESPLIEGA NUMERO DE REINTENTOS
REGRESA
FALLA
DESPLIEGA DIRECCION
DESPLIEGA EL CONTENIDO ORIGINAL Y EL ACTUAL
CONTADOR DE REINTENTOS=0AH
REGRESA

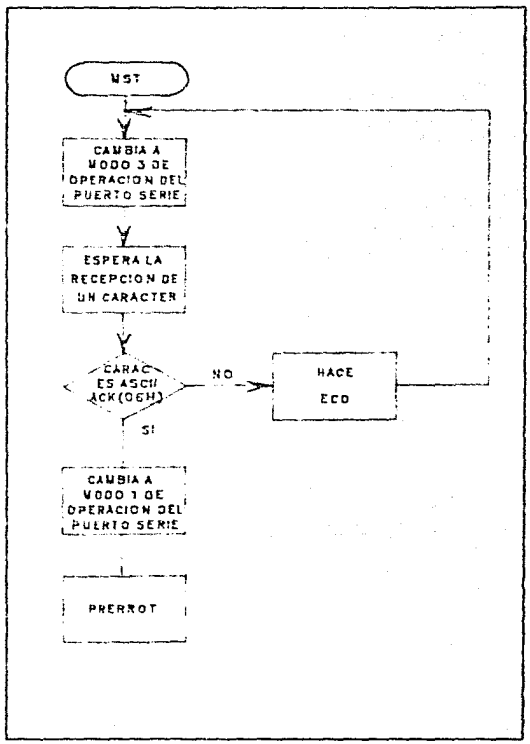
PM-IV

MST (Maestro).

Esta subrutina se encarga de configurar el puerto serie del SIS Z8, de manera que este último no interfiera en la transmisión entre computadora anfitrión y terminal, a este modo de operación se le denomina modo 3.

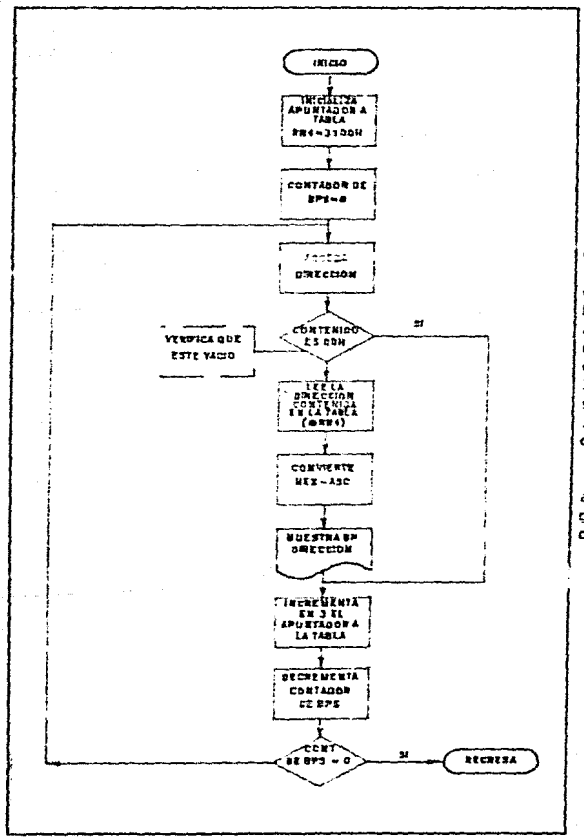
Es obvio que este modo de operación deshabilita las funciones del SIS-Z8, maniobrando éste únicamente en espera de la clave de retorno al monitor.

Durante este ciclo de espera, la rutina lee los caracteres enviados por la terminal retransmitiendolos si no se trata de un CTRL F, cuando ha leido este caracter, reconfigura los puertos al modo 1, con lo cual se reinician las funciones del monitor, los mensajes dados por la terminal se verán interferidos entonces por el SIS-Z8.



MST

DB (Despliegue de Break Points).



La rutina de despliegue de Break Points, se encarga de mostrar los Puntos de Prueba o Ruptura que se encuentren almacenados en la tabla de Break Points, que inicia en la dirección 3100_H de datos.

La rutina revisa cada uno de los ocho lugares posibles para guardar los Puntos de Ruptura, desechando aquellos que se encuentren vacios y enviando a la pantalla las direcciones donde se encuentren localizados los Puntos de Prueba.

Esta rutina se apoya en las rutinas generales de envio de caracteres.

DB

BP (Break Point).

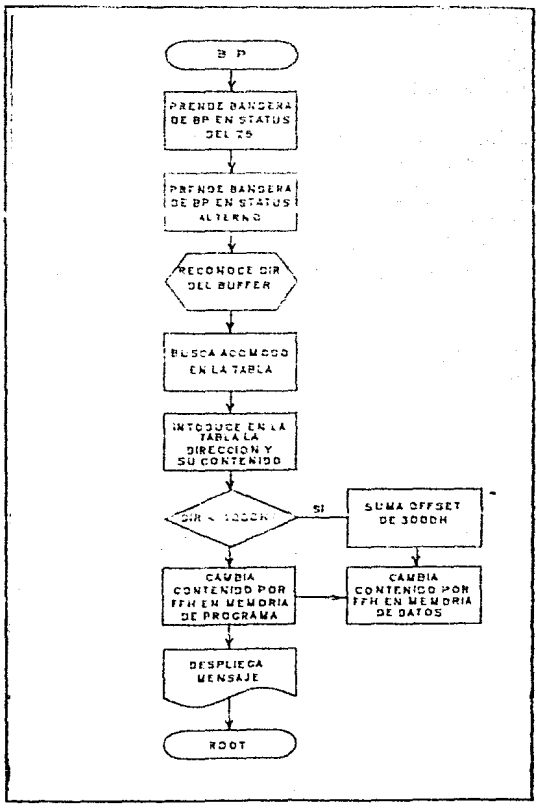
Este comando es usado para colocar puntos de ruptura en localidades incluidas en el rango de direcciones ocupadas con los programas de prueba. Los puntos de ruptura tienen como objetivo el detener la ejecución del programa y mostrar el estado de los registros.

La rutina permite colocar hasta ocho puntos de ruptura a la vez, utilizando para ello una tabla de 24 localidades de memoria que almacenan la dirección donde se colocó el punto de ruptura (dos palabras) y el contenido original de la dirección.

El código que sustituye al original, y que provoca la interrupción de la ejecución, es un FF_H.

La rutina incluye una subrutina de búsqueda de lugares vacíos en tabla.

La tabla de Break Points es modificada por el mismo comando BP y el comando BX, el cual permite borrar uno, varios, o todos los puntos de ruptura.



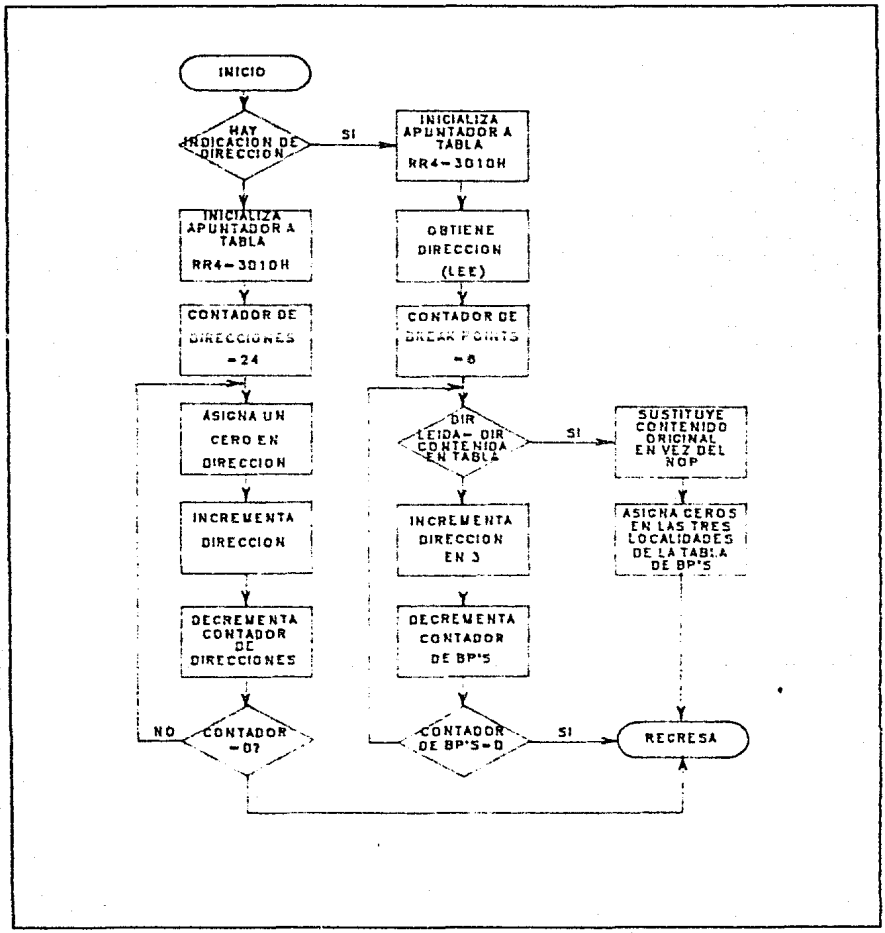
BP

BX (Borrado de Break Points).

Esta rutina es la encargada de eliminar los puntos de ruptura o prueba, opera bajo dos modos diferentes, en el primero se verifica si existe la indicación de borrar un break point en una dirección especifica y en el segundo, en caso de no existir tal indicación, ejecuta la limpieza de toda la tabla.

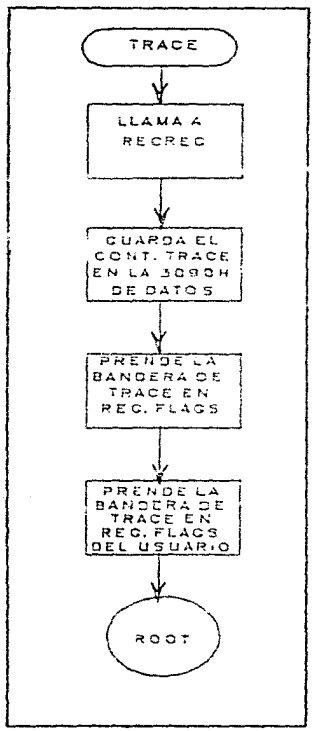
Como se ha mencionado, una vez indicada una dirección especifica, verifica que exista en la tabla, y si este es el caso reintegra el código original del programa en lugar del colocado por la rutina de BP (FF_H). Obviamente genera un error en caso de no existir un punto de ruptura en la dirección indicada.

Cuando no se le indica una dirección especifica coloca 00_H en todas las direcciones que abarca la tabla de Break Points (3100_H en adelante).


INICIO
HAY INDICACION DE DIRECCION
SI
INICIALIZA APUNTADOR A TABLA RR4-3010H
INICIALIZA APUNTADOR A TABLA RR4-3010H
OBTIENE DIRECCION (LEE)
CONTADOR DE DIRECCIONES =24
CONTADOR DE BREAK POINTS =8
ASIGNA UN CERO EN DIRECCION
DIR LEIDA= DIR CONTENIDA EN TABLA
SI
SUSTITUYE CONTENIDO ORIGINAL EN VEZ DEL NOP
INCREMENTA DIRECCION
INCREMENTA DIRECCION EN 3
ASIGNA CEROS EN LAS TRES LOCALIDADES DE LA TABLA DE BP'S
DECREMENTA CONTADOR DE DIRECCIONES
DECREMENTA CONTADOR DE BP'S
NO
CONTADOR =0?
CONTADOR DE BP'S=0
SI
REGRESA

BX

T (Trace).



TRACE

Es la rutina utilizada para la predisposición de ejecución de programas paso a paso o intstrucción por instrucción. La rutina por si sola no ejecuta el programa, tiene que ser auxiliada por los comandos GO o CALL y K (Continúa) (en caso de haberse aplicado estos últimos).

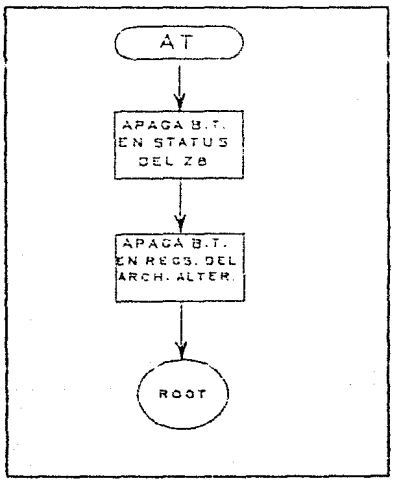
La rutina reconoce el número de instrucciones a ejecutarse paso a paso, a través de la rutina de reconocimiento de dos digitos hexadecimales (RECREG), utiliza el último bit del registro de estado o banderas para indicar que la opción de Trace está activa.

El número de instrucciones que ejecuta paso a paso es respaldado en la dirección 3090_H de datos y se ve decrementada cada vez que se ejecuta una interrupción por IRQO.

AT (Apaga Trace).

Esta es la rutina que desactiva la opción de ejecución de programas paso por paso, su única función es desactivar el último bit de los registros de banderas interno y alterno (bandera T/BP).

Los últimos bits del registro de status son conocidos como Banderas de Usuario; el último es ocupado para establecer el control sobre ejecuciones instrucción por instrucción.



CALL (Llamada a programa de prueba)

Este comando consta de dos etapas denominadas COMUN y TAMCAK. La primera ejecuta funciones compatibles al comando GO.

La subrutina COMUN se encarga de salvaguardar la dirección de la rutina CALL, mediante la rutina SALVAC, para después ser reconocida por el comando K. Además, esta rutina recupera del buffer la dirección de arranque del programa de prueba, la cual queda depositada en el registro par RR12.

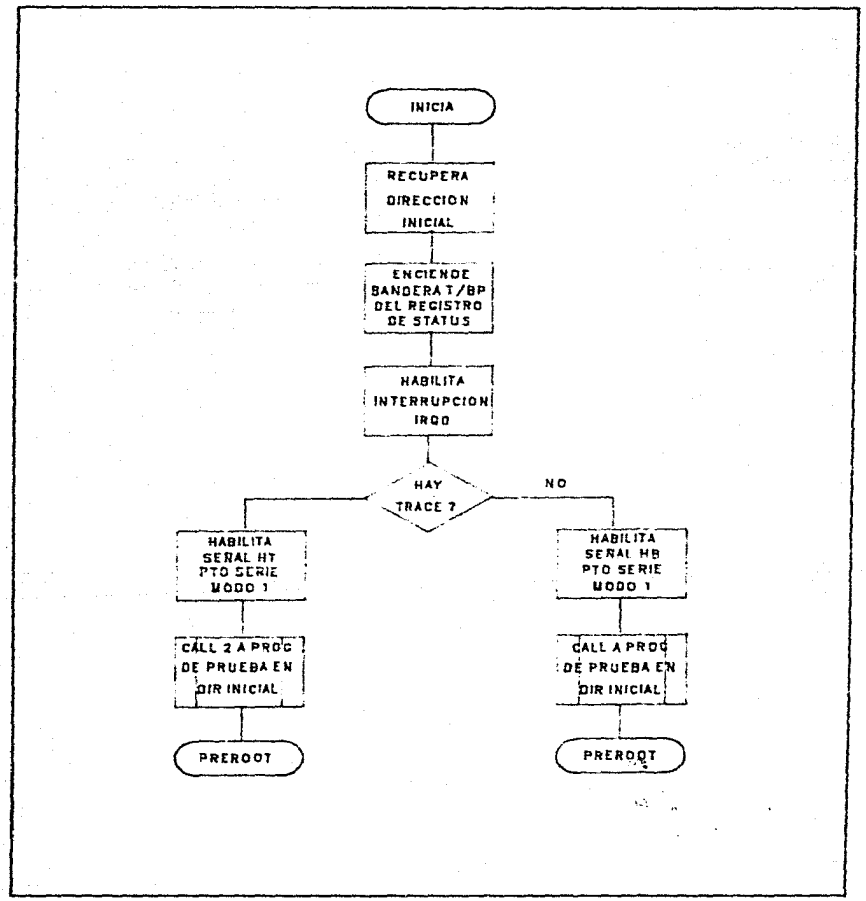
La segunda etapa o TAMCAK prende la bandera de CALL/GO en el bit 2 del registro de status y programa los registros R6 , R7 como sus equivalentes $F8_H$ y $F7_H$ respectivamente. Esto último se hace con el objeto de auxiliar a la rutina de servicio a IRQ0, ya que ésta es inicialmente "ciega" en cuanto que no reconoce si el programa de prueba ha modificado registros de control o no. Como los registros de control $F8_H$ y $F7_H$ son esenciales para la operación de la rutina IRQ0, ésta asume que los primeros se encuentran respaldados en R8 y R7 respectivamente, por lo que toma su valor para reprogramar a $F8_H$ y $F7_H$.

Finalmente, revisando el valor del bit 1 del registro de status (bandera T/BP), determina si Trace fué seleccionado previamente, en cuyo caso habilita la señal HT del puerto interno de control y hace una llamada (CALL) a la dirección de arranque del programa de prueba. En caso contrario hace operaciones idénticas al caso anterior, pero habilitando HBP.

Debe notarse que dada la forma de accesar al programa de prueba, este puede concluir con una instrucción RET, y el comando CALL, automáticamente se encargará de regresar al monitor.

Las interrupciones por IRQ0 se habilitan en dos fases. La primera en el bit 1 del registro IMR y la segunda , previamente al salto a la dirección de arranque, hay una instrucción EI.

La rutina TAMCAK es empleada también en el comando K, cuando éste determina que debe continuar con la ejecución de un comando CALL.


INICIA
RECUPERA DIRECCION INICIAL
ENCIENDE BANDERA T/BP DEL REGISTRO DE STATUS
HABILITA INTERRUPCION IRQ0
HAY TRACE ?
NO
HABILITA SEÑAL HT PTO SERIE MODO 1
HABILITA SEÑAL HB PTO SERIE MODO 1
CALL 2 A PROC DE PRUEBA EN DIR INICIAL
CALL A PROC DE PRUEBA EN DIR INICIAL
PREROOT
PREROOT

CALL

GO (Arranque de un programa de prueba).

La primera operación de este comando es reconocer la opción [I]. Si esta es vàlida, programa al archivo alterno con los valores que el archivo de registros debe tener después de un reset maestro. Para la operación se vale de un subarchivo adyacente al archivo alterno en memoria de datos, y que previamente fuè grabado por la rutina INIT.

En la subrutina COMUN, discutida en la sección del comando CALL, recupera la dirección de arranque, para proceder a la rutina TAMGOK, que al igual que su contraparte TAMCAK, es utilizada por el comando K al continuar con la operación del comando GO, cuando así procede.

La dificultad de esta rutina se debe a que debe recuperar el archivo alterno y colocarlo en el interno. Para lograr èsto se vale de la rutina ENTRADA que se encuentra en el Monitor Interno.

Después de apagar la bandera CALL/GO del registro de banderas (bit 2), reconoce si T debe ser habilitada en el puerto interno de control y procede a la transferencia de registros del archivo alterno.

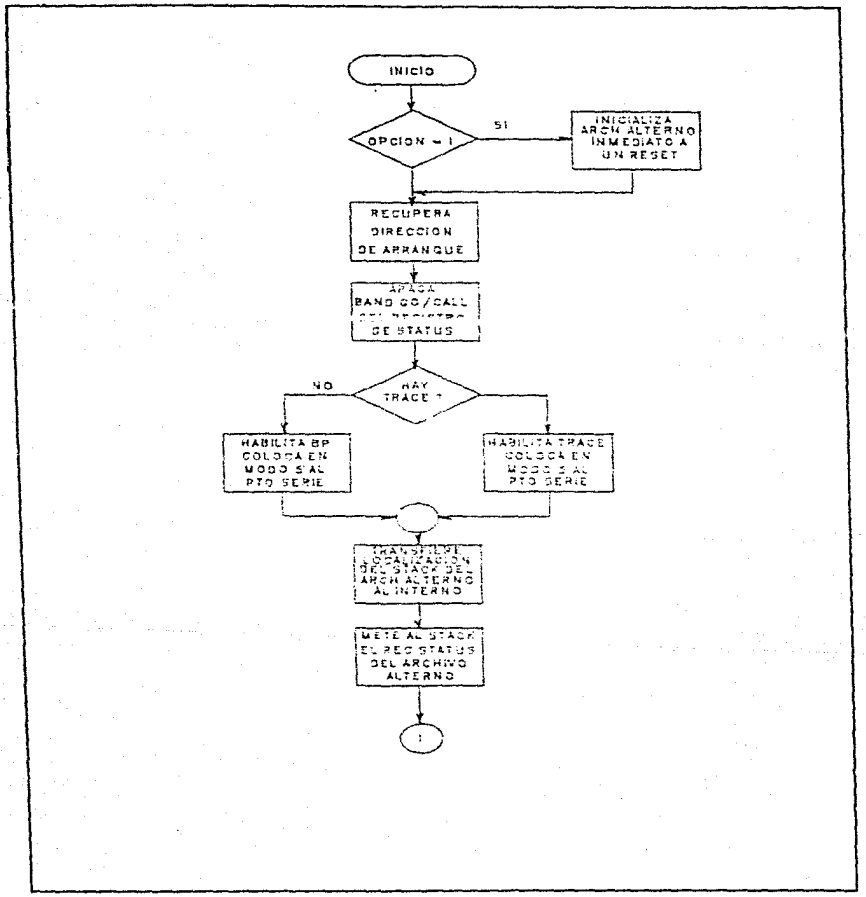
Inicialmente pasa el apuntador al stack de los registros alternos correspondientes, así también determina del registro $F8_H$ alterno la localización del stack.

Con el apuntador del stack en la última localidad antes de una interrupción, introduce la nueva dirección, el registro de status, además del registro RP del archivo alterno y posteriormente el registro 02_H previo a la interrupción.

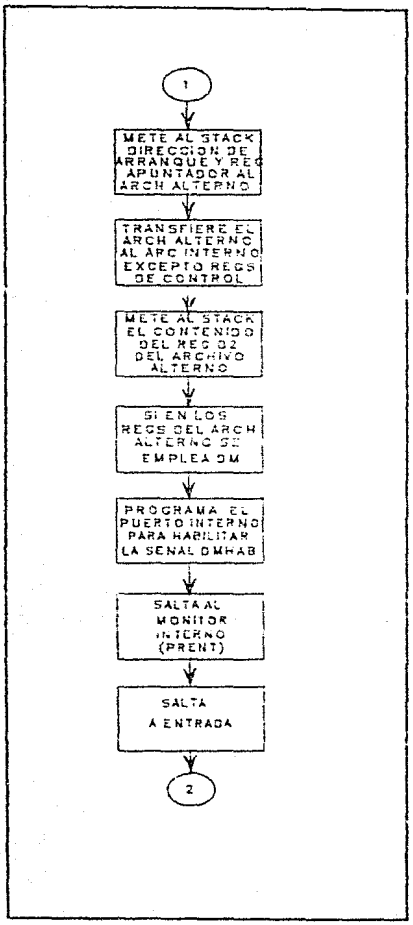
Procede a la transferencia del archivo alterno a los registros del Z8, a excepción de los registros de control, los cuales quedan en el grupo 1, dejando libres para operación solo a los registros R12, R13, R14 y R15.

Finalmente si la opción de memoria de datos se encuentra habilitada en registros alternos, entonces habilita la señal DMHAB del puerto interno de control y salta a la rutina ENTRADA.

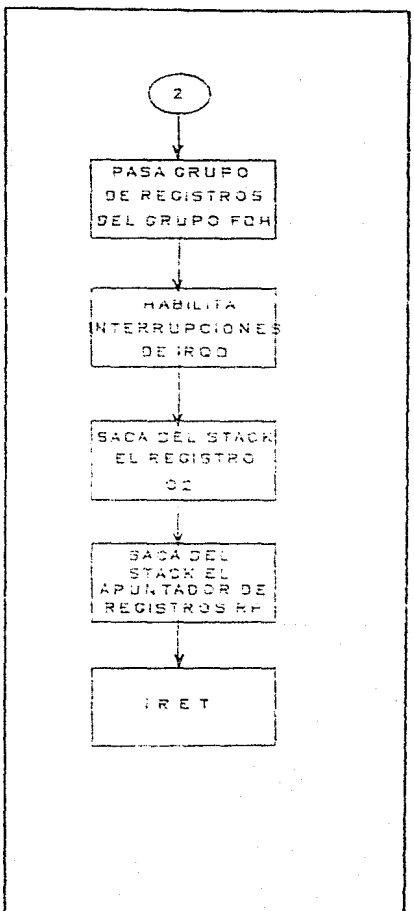
En esta última rutina, se transfieren a los registros de control los registros del grupo 1, así como el apuntador de registros y el registro 02_H, los cuales se encontraban dentro del stack. Por último habilita interrupciones por IR00 y ejecuta una instrucción IRET (recordar que la dirección de arranque fuè introducida al stack pasos atrás), con lo cual automáticamente habilita interrupciones.


INICIO
OPCION = 1
SI
INICIALIZA ARCH ALTERNO INMEDIATO A UN RESET
RECUPERA DIRECCION DE ARRANQUE
APAGA BAND GO/CALL
DE STATUS
NO
HAY TRACE ?
HABILITA BP COLOCA EN MODO S AL PTO SERIE
HABILITA TRACE COLOCA EN MODO S AL PTO SERIE
METE AL STACK EL REG STATUS DEL ARCHIVO ALTERNO

GO-I



GO-II



ENTRADA

K (Continúa)

Los comandos DM, GO y CALL pueden ser reejecutados por este comando. En caso de DM este despliega 256 localidades a partir de la última dirección mostrada.

Para los comandos GO y CALL, éste tiene efecto si fuerón interrumpidos por BP o T.

El comando K tiene como respaldo para su operación a las subrutinas SALVAC y PARAK, que son empleadas por los comandos DM, GO, CALL y servicio a IRQ0. La función de estas es respaldar en memoria de datos, información pertinente para la ejecución de K:

DIRECCION	MEMORIA DE DATOS	
3050		DIRECCION DEL ULTIMO COMANDO EJECUTADO
3051		
3052	(D) O (ESP) ASCII	RECONOCIMIENTO [D]
3053		
3054		ULTIMA DIRECCION DE DESPLIEGE O DE RETORNO
3055		

Inspeccionando en esta tabla las localidades 3050_H y 3051_H, determina cual fué el último comando ejecutado.

Cuando encuentra que este fue DM, restaura el buffer de recepción como si se hubiera obtenido directamente de la consola y salta a DM.

En caso de GO o CALL recupera la dirección de retorno de las localidades 3054_H y 3055_H y pasa a ejecutar estos comandos desde las rutinas TAMGOK y TAMCAK respectivamente.

RUTINA DE SERVICIO A IRQ0.

Este programa esta formado por muchas pequeñas rutinas, comenzando por SALPRO, cuya dirección està contenida en el vector de interupción correspondiente.

SALPRO. Esta rutina debe modificar los registros RP, 02, P01M, P3M, SPH y SPL, para permitir al resto de las rutinas su correcta operación. Esto lo hace salvando su contenido en el stack, o como en el caso de P3M, supone que èste se encuentra respaldado en el registro 18_H. La modificación al registro P01M se hace evitando alterar el valor del bit correspondiente a la localización del stack. Finalmente determina de las banderas CALL/GO y T/BP del registro de status, cuales de estas operaciones fueron seleccionadas, ya que la rutina debe proceder en forma distinta segùn cualquiera de los cuatro casos siguientes:

RUTINA CTOATO	CALL CON TRACE
RUTINA CBPOATO	CALL CON BP
RUTINA GOTOATO	GO CON TRACE
RUTINA GOBPOATO	GO CON BP

XXXOATO. Estas rutinas estàn formadas por combinaciones de otras cuatro:

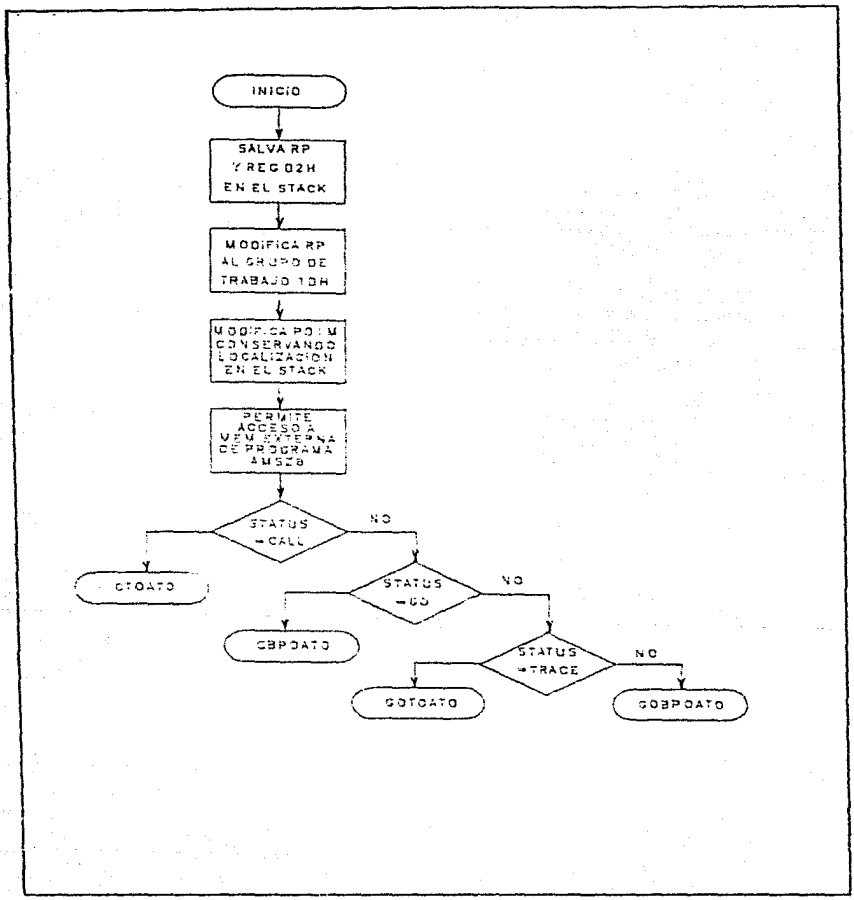
ATRACE Atención a interrupciones por TRACE.

ATBP Atención a interrupciones por BP.

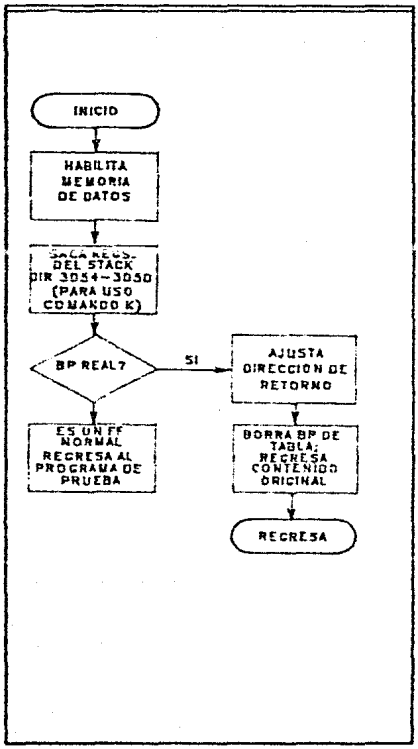
RESPALDAU Rutina que debe ejecutarse en caso de GO.

DTRCALL Rutina que debe ejecutarse en caso de CALL.

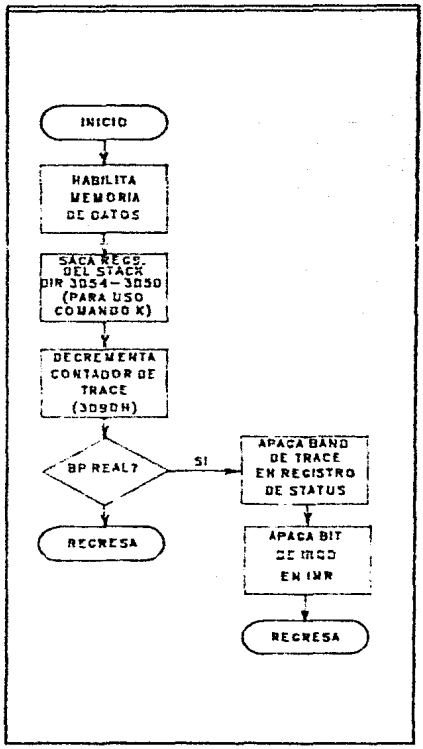
ATRACE. Su función es decrementar el contador de eventos TRACE que se encuentra en la localidad 3090_H de memoria de datos. Si èste es cero apaga la bandera de T/BP del registro de status y elimina posteriores interrupciones apagando el bit 1 del registro IMR.


INICIO
SALVA RP
Y REG 02H
EN EL STACK
MODIFICA RP
AL GRUPO DE
TRABAJO 10H
MODIFICA PO:M
CONSERVANDO
LOCALIZACION
EN EL STACK
PERMITE
ACCESO A
UNA EXTERNA
DE PROGRAMA
AMSZ8
STATUS
=CALL
NO
GTOATO
STATUS
=GO
NO
GBPOATO
STATUS
=TRACE
NO
GOTOATO
GOBPOATO

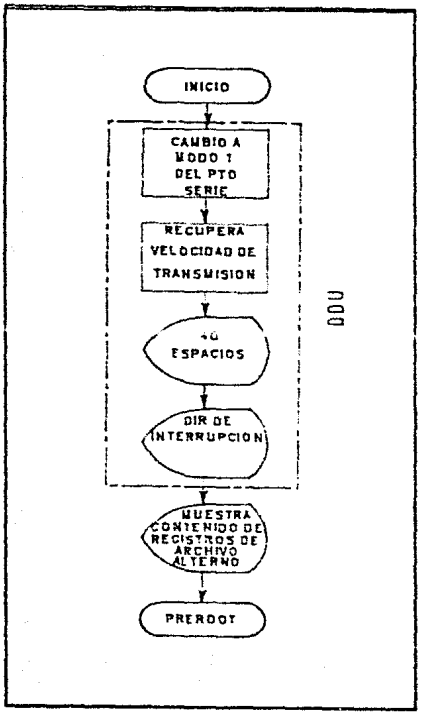
SALPRO


INICIO
HABILITA MEMORIA DE DATOS
SACA REGS. DEL STACK DIR 3054-3050 (PARA USO COMANDO K)
BP REAL?
SI
AJUSTA DIRECCION DE RETORNO
ES UN FF NORMAL REGRESA AL PROGRAMA DE PRUEBA
BORRA BP DE TABLA; REGRESA CONTENIDO ORIGINAL
REGRESA

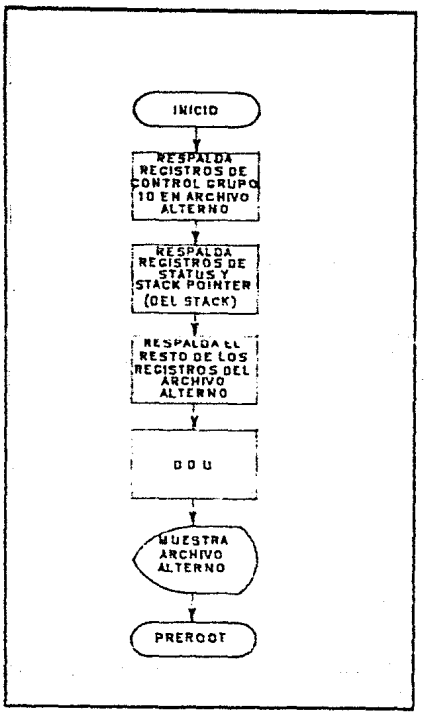
AT BP


INICIO
HABILITA MEMORIA DE DATOS
SACA REGS. DEL STACK DIR 3054-3050 (PARA USO COMANDO K)
DECREMENTA CONTADOR DE TRACE (3090H)
BP REAL?
SI
APAGA BAND DE TRACE EN REGISTRO DE STATUS
APAGA BIT DE MOD EN IMR
REGRESA
REGRESA

ATRACE

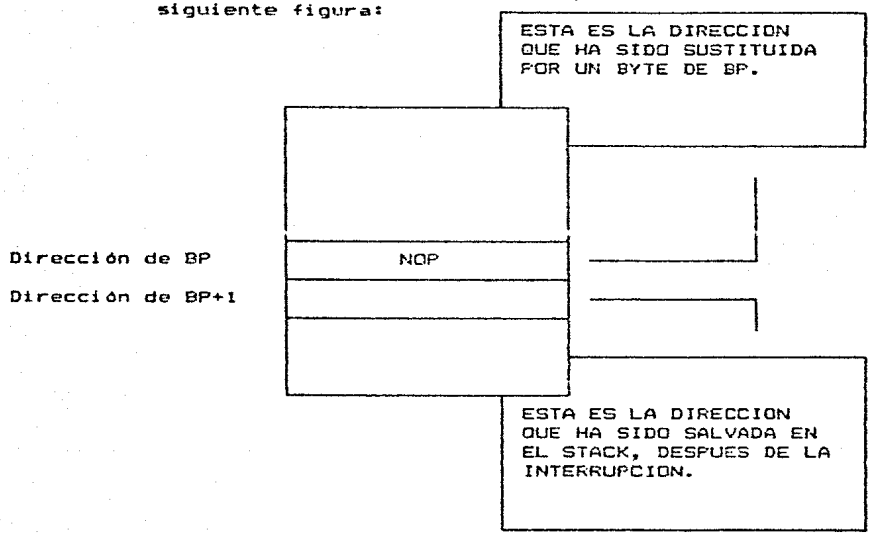


DTRCALL



RESPALDAU

ATBP. Cuando ocurre una interrupción debida a BP, la dirección de retorno está alterada, èsto se debe a que la original debe ser la ocupada por el byte de BP y no la siguiente, como se puede observar en la siguiente figura:



Es por èsto que es necesario hacer una corrección a esta dirección, para que en caso de ser un BP y no un NOP, después de sustituir el contenido original de la localidad, al retorno no comience por la siguiente.

La rutina confirma que el BP es verdadero, buscando la dirección de rompimiento en la tabla de BP. En caso de serlo, toma el contenido original de la dirección y lo regresa a su localidad. Después elimina el BP de la tabla.

Si el la dirección de la localidad no fuè encontrada en la tabla de BP, regresa al programa de prueba.

RESPALDAU. Su función es respaldar los registros del Z8 en el archivo alterno en memoria de datos. Esto se hace en varias etapas, respaldando primero los registros de control, que se asumen estar en el grupo 10_H, seguidos por los registros apuntadores de stack y de status. Finalmente el resto de los registros de propósito general.

La función DTRGO despliega el contenido del archivo alterno y la dirección donde ocurrió la interrupción.

DTRCALL. Despliega el contenido de los registros del Z8 y la dirección donde ocurrió la interrupción. Es una rutina muy parecida a DTRGO.

A continuación en las siguientes páginas se muestra el listado completo del SCAN Z8.

```
0000            ;**********************************************************
0000            ;**********************************************************
0000            ;**                                                      **
0000            ;**                                                      **
0000            ;**                      S C A N   Z 8                   **
0000            ;**                                                      **
0000            ;**                                                      **
0000            ;**                                                      **
0000            ;**********************************************************
0000            ;**********************************************************
0000            ;**********************************************************
0000            ;*                                                      *
0000            ;*                    DECLARACIONES                     *
0000            ;*                                                      *
0000            ;**********************************************************
00FC            FLAGS       EQU     252
0004            EOF         EQU     04
003D            IGUAL       EQU     3DH
0048            MACHE       EQU     48H
0012            AP          EQU     12H
0013            XOFF        EQU     13H
0013            CONTAR      EQU     13H
000A            LF          EQU     0AH
002C            COMA        EQU     2CH
0000            NUL         EQU     00
0008            BS          EQU     08
007F            DEL         EQU     7FH
00FE            FI          EQU     0FEH
00FD            FT          EQU     0FDH
00FF            FF          EQU     0FFH
0006            CTRLF       EQU     06H
0020            ESP         EQU     20H
000D            CR          EQU     13
001A            REG         EQU     1AH
001E            REG1        EQU     1EH
001F            REG2        EQU     1FH
```

```
0000                    FORM
0000            ;*****************************************************************
0000            ;*                                                               *
0000            ;*              M O N I T O R     I N T E R N O                  *
0000            ;*                                                               *
0000            ;*****************************************************************
0000                    ORG     000H
0000  0073               DW      SALPRO
0002  002C               DW      ATENT1              ;***************************
0004  0032               DW      ATENT2              ;*                         *
0006  0038               DW      ATENT3    ;SERIAL IN  *VECTORES DE INTERRUPCION *
0008  003E               DW      ATENT4    ;SERIAL OUT *                         *
000A  0044               DW      ATENT5              ;***************************
000C  8F                 DI
000D  E6F750             LD      P3M,#01010000B      ;***************************
0010  E6F8D6             LD      P01M,#11010110B     ;*PROGRAMACION DE REGISTROS*
0013  E6F600             LD      P2M,#00000000B      ;*       DE CONTROL        *
0016  E6FD00             LD      SMR,#00000000B      ;*   MODIFICA OSM70        *
0019  E6F914             LD      IPR,#00010100B      ;*   R8 EN GRUPO 10        *
001C  E6FF40             LD      SPL,#40H            ;*STACK INTERNO A PARTIR DE*
001F  E6FA00             LD      IRQ,#00H            ;* 40H,SALTA MONITOR EXTRNO*
0022  3110               SRP     10H                 ;***************************
0024  E60200             LD      02,#00000000B
0027  8D3008             JP      MONSUP
002A  FF                 NOP
002B  FF                 NOP
002C            ;*****************************************************************
002C            ;*                                                               *
002C            ;*          SIMULACION DE VECTORES DE INTERRUPCION               *
002C            ;*                                                               *
002C            ;*****************************************************************
002C  E61F02    ATENT1   LD      IFH,#02H    ;***************************
002F  8D004A             JP      GENERAL     ;* ATENT1 - ATENT5 PARA SIMULAR *
0032  E61F04    ATENT2   LD      IFH,#04H    ;* LOS VECTORES DE INTERRUPCION *
0035  8D004A             JP      GENERAL     ;* DE PROGRAMA DE PRUEBA DE     *
0038  E61F06    ATENT3   LD      IFH,#06H    ;* IRQ1 - IRQ5 RESPECTIVAMENTE  *
003B  8D004A             JP      GENERAL     ;***************************
003E  E61F08    ATENT4   LD      IFH,#08H
0041  8D004A             JP      GENERAL
0044  E61F0A    ATENT5   LD      IFH,#0AH
0047  8D004A             JP      GENERAL
004A  70FD      GENERAL  PUSH    RP                  ; CAMBIA DE RO DE TRABAJO
004C  3110               SRP     10H
004E  EC02               LD      R14,#02H
0050  C2CE               LDC     R12,@RR14           ; TOMA EL CONTENIDO DE LA LOCALIDAD
0052  A0EE               INCW    RR14                ; 02XXH Y 02XX + 1 H
0054  C2DE               LDC     R13,@RR14
0056  50FD               POP     RP                  ; LO INTRODUCE AL STACK Y RECUPERA RP
0058  701D               PUSH    1DH
005A  701C               PUSH    1CH
005C  AF                 RET                         ;SALTA A LA VERDADERA RUTINA DE SERVICIO
005D            ;
```

```
SIS-Z8 SISTEMA DE DESARROLLO
085D                    FORM
085D            ;*******************************************************************
085D            ;*          RUTINA DE ENTRADA AL PROGRAMA DEL USUARIO              *
085D            ;*                SE EMPLEA PARA GO Y CONTINUE                     *
085D            ;*******************************************************************
085D  CCF1      ENTRADA   LD        R12,#0F1H       ;*********************************
085F  DC11                LD        R13,#11H        ;* PASA REGISTROS DEL GRUPO 1@ A   *
0861  EC0B                LD        R14,#0BH        ;* SUS SIMILARES EN EL GRUPO DE    *
0863  E3FD      MUEVE     LD        R15,@R13        ;* REGISTROS DE CONTROL            *
0865  F3CF                LD        @R12,R15        ;*********************************
0867  CE                  INC       R12
0868  DE                  INC       R13
0869  EAF8                DJNZ      R14,MUEVE
086B  46FB01              OR        IMR,#00000001B  ; HABILITA IRQ0
086E  5002      VAMONOS   POP       02              ; SACA DEL STACK EL REGISTRO 02
0870  50FD                POP       RP              ; Y EL REGISTRO RP
0872  BF                  IRET
0873            ;*******************************************************************
0873            ;*          RUTINA DE SALIDA DE CUALQUIER PROGRAMA DE USUARIO      *
0873            ;*              SE ACCESA A ELLA MEDIANTE UNA INTERRUPCION         *
0873            ;*******************************************************************
0873  70FD      SALPR0    PUSH      RP              ; INTRODUCE AL STACK A RP Y 02
0875  7002                PUSH      02
0877  3110                SRP       10H             ; MODIFICA RP AL GRUPO DE TRABAJO
0879  0CFB                LD        R0,#11111011B
087B  44F8E0              OR        R0,P01M         ; MODIFICA EL REGISTRO P01M SIN
087E  56E0D6              AND       R0,#11010110B   ; ALTERAR LA LOCALIZACION DE STACK
0881  09F8                LD        P01M,R0
0883  E6F750              LD        P3M,#01010000B  ; MODIFICA P3M
0886  C8FE                LD        R12,SPH
0888  D8FF                LD        R13,SPL
088A  70EC                PUSH      R12
088C  70ED                PUSH      R13
088E  56027F              AND       02,#01111111B   ; HABILITA AMSZB
0891  76FC03              TM        FLAGS,#00000011B ; DETERMINA LAS COMBINACIONES
0894  6D3B1D              JP        EQ,CTOAT0
0897  76FC02              TM        FLAGS,#00000010B
089A  6D3B23              JP        EQ,CBPOAT0      ; T, GO , CALL Y
089D  76FC01              TM        FLAGS,#00000001B ; BP CON QUE FUE EJECUTADO EL
08A0  6D3B11              JP        EQ,GOTOAT0      ; PROGRAMA DE PRUEBA
08A3  8D3B17              JP        GOBPOAT0
```

```
SIS-Z8 SISTEMA DE DESARROLLO
00A6                    FORM
00A6            ;*********************************************************************
00A6            ;*********************************************************************
00A6            ;*                                                                   *
00A6            ;*          AQUI COMIENZA EL MONITOR SUPERIOR                        *
00A6            ;*                                                                   *
00A6            ;*********************************************************************
00A6            ;*********************************************************************
00A6                    ORG     3000H
3000            ;*********************************************************************
3000            MONSUP
3000  2C03              LD      R2,#03H
3002  D63242            CALL    MODOS           ; MODO 3 DEL PUERTO SERIE
3005  4C44      INIT    LD      R4,#44H
3007  5C00              LD      R5,#00H         ;*****************************
3009  0C00              LD      R0,#00H         ;*   PROGRAMA ARCHIVO DE     *
300B  1C7F              LD      R1,#7FH         ;*  REGISTROS ALTERNE A      *
300D  9204      TODOS   LDE     @RR4,R0         ;*  ALTERNO CON LOS VALORES  *
300F  A0E4              INCW    RR4             ;*  QUE EL ARCHIVO INTERNO   *
3011  1AFA              DJNZ    R1,TODOS        ;*  TIENE DESPUES DE UN      *
3013  5C02              LD      R5,#02H         ;*  RESET MAESTRO.           *
3015  0C80              LD      R0,#80H         ;*  ESTO ES UTILIZADO        *
3017  9204              LDE     @RR4,R0         ;*  POSTERIORMENTE POR       *
3019  0C40              LD      R0,#40H         ;*  EL COMANDO GO CON LA     *
301B  5C1E              LD      R5,#1EH         ;*  OPCION 1                 *
301D  9204              LDE     @RR4,R0         ;*****************************
301F  A0E4              INCW    RR4
3021  9204              LDE     @RR4,R0
3023  0CFE              LD      R0,#0FEH
3025  5C16              LD      R5,#16H
3027  9204              LDE     @RR4,R0
3029  0C4D              LD      R0,#4DH
302B  5C19              LD      R5,#19H
302D  9204              LDE     @RR4,R0
302F  4C45              LD      R4,#45H
3031  5C1C              LD      R5,#1CH
3033  3214              LDE     R1,@RR4
3035  56E1FC            AND     R1,#11111100B
3038  9214              LDE     @RR4,R1
303A  56FCFC            AND     FLAGS,#11111100B
303D  D63136            CALL    XS
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3040                    FORM
3040            ;***************************************************************
3040            ;*****                                                   *******
3040            ;*****  M     O     N     I     T     O     R            *******
3040            ;*****                                                   *******
3040            ;***************************************************************
3040  2C03              LD        R2,#03H
3042  D63242            CALL      MODOS
3045  4C30              LD        R4,#CERO>8    ; ESTADO CERO INICIAL
3047  5C58              LD        R5,#CERO\256
3049  0C50              LD        R0,#50H       ; R0 APUNTA A BUFFER DE CARACTERES
304B  D63127    ROOT    CALL      AGUANTA
304E  D63053            CALL      RECEPCION
3051  88F8              JR        ROOT
3053  30E4      RECEPCION JP      @R4           ; SALTA A LA RUTINA CORRESPONDIENTE
3055  29F0      RETA    LD        SIO,R2        ; AL ESTADO EN QUE SE ENCUENTRE
3057  AF                RET                     ; HACE UN ECO Y TERMINA RECEPCION
3058  A61206    CERO    CP        12H,#CTRLF
305B  6806              JR        EQ,CAM2       ; SI EL CARACTER ES CTRL F CAMBIA AL
305D  4C30      CAMUNO  LD        R4,#UNO>8     ; ESTADO 2 DE LO CONTRARIO CAMBIA AL
305F  5C68              LD        R5,#UNO\256   ; ESTADO 1
3061  88F2              JR        ,RETA
3063  4C30      CAM2    LD        R4,#DOS>8     ; PROCEDIMIENTO PARA CAMBIAR AL ESTADO 2
3065  5C73              LD        R5,#DOS\256
3067  AF                RET
3068  A61200    UNO     CP        12H,#CR       ; SI EL CARACTER ES CARRIAGE RETURN
306B  EBE8              JR        NE,RETA
306D  4C30              LD        R4,#CERO>8    ; CAMBIA AL ESTADO CERO
306F  5C58              LD        R5,#CERO\256
3071  88E2              JR        ,RETA
3073  A61200    DOS     CP        12H,#CR       ; COMPRUEBA QUE EL SIGUIENTE CARACTER
3076  EBE5              JR        NE,CAMUNO     ; DESPUES DE CTRLF SEA CR
3078  2C01              LD        R2,#01H
307A  D63242            CALL      MODOS         ; SI LO ES PROGRAMA EL PUERTO SERIE
307D  4C30              LD        R4,#TRES>8
307F  5C38              LD        R5,#TRES\256  ; CAMBIA AL ESTADO 3
3081  6C3E              LD        R6,#MENACC>8  ; Y MANDA EL MENSAJE DE
3083  7C8C              LD        R7,#MENACC\256; "SIS Z8 LISTO PARA RECIBIR INSTRUCCIONES"
3085  8D3287            JP        ,ENVSINT
3088  A61220    TRES    CP        12H,#ESP
308B  68C8              JR        EQ,RETA       ; SI LOS CARACTERES SOH CR O ESP
308D  A61200            CP        12H,#CR       ; SE MANTIENE EL ESTADO 3
3090  ED309A            JP        NE,CAM4
3093  6C3D              LD        R6,#PROMPT>8  ; MANDA EL PROMPT DEL SIS Z8
3095  7CF3              LD        R7,#PROMPT\256
3097  8D3287            JP        ,ENVSINT
309A  A6127F    CAM4    CP        12H,#DEL
309D  6D3055            JP        EQ,RETA
30A0  4C30              LD        R4,#CUATRO>8
30A2  5CD9              LD        R5,#CUATRO\256; CAMBIA AL ESTADO 4
```

```
SIS-Z8 SISTEMA DE DESARROLLO
30A4                        FORM
30A4  A6E261      BUFF      CP        R2,#61H
30A7  1B11                  JR        LT,SAL
30A9  A6E27A                CP        R2,#7AH       ; CAMBIO DE MINUSCULAS A MAYUSCULAS
30AC  980C                  JR        GE,SAL
30AE  3812                  LD        R3,12H
30B0  26E320                SUB       R3,#20H
30B3  F51310                LD        @10H,13H      ; GUARDA EL CARACTER EN BUFFER E
30B6  0E                    INC       R0            ; INCREMENTA APUNTADOR
30B7  8D3055                JP        ,RETA
30BA  F51210      SAL       LD        @10H,12H
30BD  0E                    INC       R0
30BE  8D3055                JP        ,RETA
30C1  A6127F      PDEL      CP        12H,$DEL      ; SI EL CARACTER RECIBIDO ES DEL
30C4  EB0E                  JR        NE,BUFF       ; MANDA CARACTER BS(BACK SPACE)
30C6  6C3D                  LD        R6,$DELETE)8   ; ESP Y BS
30C8  7CF8                  LD        R7,#DELETE\256
30CA  00E0                  DEC       R0
30CC  A61050                CP        10H,#50H      ; ELIMINA CARACTER DEL BUFFER
30CF  ED3287                JP        NE,ENVSINT    ; SI SE HA BORRADO TODA LA LINEA
30D2  4C30                  LD        R4,#TRES)8    ; REGRESA AL ESTADO 3
30D4  5C88                  LD        R5,#TRES\256
30D6  8D3287                JP        ,ENVSINT
30D9  A6120D      CUATRO    CP        12H,#CR       ; BUSCA EL CARACTER DE CR
30DC  EBE3                  JR        NE,PDEL
30DE  4C38                  LD        R4,#TRES)8    ; SI LO ENCONTRO
30E0  5C88                  LD        R5,#TRES\256  ; COMIENZA EL DECODIFICADOR
30E2  8C3C                  LD        R8,#TABL)8    ; TABL CONTINE LA TABLA DE COMANDOS
30E4  9CE1                  LD        R9,#TABL\256
30E6  E7102C                LD        @10H,#COMA
30E9  C218                  LDC       R1,@RR8       ; ESTA EL PRIMER CARACTER DEL BUFFER?
30EB  0C50        COMPI     LD        R0,#50H
30ED  A310        COMPC     CP        R1,@R0        ; SI NO LO ESTA BUSACA EL SIGUIENTE
30EF  681F                  JR        EQ,CONT       ; COMANDO EN TABL
30F1  A0E8        BUSCAFI   INCW      RR8
30F3  C218                  LDC       R1,@RR8
30F5  A611FE                CP        11H,#FI
30F8  EBF7                  JR        NE,BUSCAFI
30FA  061903                ADD       19H,#03H
30FD  161800                ADC       18H,#00H
3100  C218                  LDC       R1,@RR8
3102  A611FD                CP        11H,#FT
3105  EBE4                  JR        NE,COMPI        ; SI LLEGO AL FIN DE TABL
3107  6C3F        SINTAX    LD        R6,#MENESIN)8
3109  7C0A                  LD        R7,#MENESIN\256 ; ENVIA MENSAJE DE ERROR DE SINTAXIS
310B  0C50                  LD        R0,#50H
310D  8D3287                JP        ,ENVSINT
```

```
3110                    FORM
3110  A0E8      CONT    INCW    RR8
3112  0E                INC     R0              ; SI EL RESTO DE LOS CARACTERES DEL
3113  C218              LDC     R1,@RR8         ; COMANDO COINCIDEN CON LOS DEL
3115  A611FE            CP      11H,#FI         ; BUFFER
3118  EBD3              JR      NE,COMPC
311A  1C3F              LD      R1,#3FH
311C  A0E8              INCW    RR8             ; TOMA LA DIRECCION DE LA TABLA DE
311E  C318              LDCI    @R1,@RR8        ; COMANDOS
3120  C318              LDCI    @R1,@RR8        ; Y LO INTRODUCE AL STACK
3122  8C3C              LD      R8,#TABL>8
3124  9CE1              LD      R9,#TABL\256    ; AL EJECUTAR ESTA INSTRUCCION YA
3126  AF                RET                     ; NO REGRESA A ROOT SINO AL COMANDO
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3127                    FORM
3127            ;*******************************************************************
3127            ;*                      A G U A N T A                              *
3127            ;*                                                                 *
3127            ;*******************************************************************
3127  8F        AGUANTA   DI
3128  E6FB08              LD          IMR,#00001000B      ; ELIMINA INTERRUPCIONES
312B  76FA08    TOUNO     TM          IRQ,#08H            ; MUESTREO DE IRQ3
312E  68FB                JR          Z,TOUNO
3130  E6FA00              LD          IRQ,#00H
3133  28F0                LD          R2,SIO              ; REGRESA CARACTER DEL SIO
3135  AF                  RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3135                    FORM
3136                ;*************************************************************
3136                ;*                                                           *
3136                ;*          RUTINA DE PROGRAMACION DE LOGICA EXTERNA         *
3136                ;*                                                           *
3136                ;*************************************************************
3136  4CEE          XS        LD        R4,@3EEH        ;RR4 APUNTARA A LA DIRECCION EXXX
3138  C224                    LDC       R2,@RR4         ;ACTIVA LOGICA CON DATO DE
313A                                                    ;SWITCHES
313A  38E2          XDEC3D    LD        R3,R2           ;RESPALDA DATO EN R3
313C  4C0F                    LD        R4,%00001111B   ;MASCARA PARA VELOCIDAD
313E  5224                    AND       R2,R4
3140  A6E200                 CP        R2,#00
3143  EB06                    JR        NE,XMASUEL
3145  E6F4AF                 LD        244,#175          ;VELOCIDAD=110
3148  8D3154                 JP        PARIDAD
314B  0000          XMASUEL   LD        R0,#00H
314D  DF                      SCF                       ;LAS VELOCIDADES SE CALCULAN CON
314E                                                    ;N-1 ROTACIONES DONDE N ES EL
314E  10E0          XCALC     RLC       R0              ;VALOR OBTENIDO DEL REGISTRO R2
3150  2AFC                    DJNZ      R2,XCALC
3152  09F4                    LD        T0,R0
3154  28E3          PARIDAD   LD        R2,R3
3156  98F7                    LD        R9,247
3158  4C10                    LD        R4,%00010000B
315A  5224                    AND       R2,R4
315C  7224                    TM        R2,R4
315E  6B08                    JR        EQ,XSON
3160  8C7F                    LD        R8,%01111111B
3162  5298                    AND       R9,R8
3164  99F7                    LD        247,R9
3166  8B06                    JR        XEND
3168  0C00          XSON:     LD        R8,%10000000B
316A  4298                    OR        R9,R8
316C  99F7                    LD        247,R9
316E  E6F50D        XEND      LD        PRE0,%0000110IB
3171  E6F103                  LD        TMR,%00000011B
3174  AF                     RET
```

```
3175                    FORM
3175                ;***************************************************************
3175                ;***************************************************************
3175  0C00      ESPDAT    LD        R0,#00H
3177  1C30                LD        R1,#30H
3179  9C30                LD        R9,#30H
317B  3C30                LD        R3,#30H
317D  D6325A      NDAT      CALL      ESPERA          ; ESPERA UN CARACTER DE CONSOLA
3180  A61230              CP        12H,#30H
3183  1B29                JR        LT,CRDEL        ; DETERMINA SI EL CARACTER ES
3185  A6127F              CP        12H,#DEL        ; NO IMPRIMIBLE
3188  EB16                JR        NE,ACOMODA
318A  A6E000              CP        R0,#00H
318D  6BEE                JR        EQ,NDAT
318F  3219                LD        R3,19H          ; SI ES DEL ACOMODA EL BUFFER
3191  9811                LD        R9,11H
3193  1C30                LD        R1,#30H
3195  00E0                DEC       R0
3197  6C3D                LD        R6,#DELETE)8
3199  7CF8                LD        R7,#DELETE\256
319B  D63287              CALL      ENVSINT
319E  8BDD                JR        NDAT
31A0  29F0      ACOMODA   LD        SIO,R2          ; HACE UN ECO DEL CARACTER Y
31A2  D6325A              CALL      ESPERA          ;
31A5  1819                LD        R1,19H          ; LO INTRODUCE AL BUFFER
31A7  9813                LD        R9,13H
31A9  3812                LD        R3,12H
31AB  0E                  INC       R0
31AC  8BCF                JR        NDAT
31AE  A6120D      CRDEL     CP        12H,#CR
31B1  EBCA                JR        NE,NDAT
31B3  A6E000              CP        R0,#00H
3186  6602                JR        EQ,BYE
31B8  2819                LD        R2,19H
31BA  AF          BYE       RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3188                    FORM
3188            ;*****************************************************************
3188            ;*                                                               *
3188            ;*              A   S   C   -   H   E   X                        *
3188            ;*                                                               *
3188            ;*****************************************************************
3188            ;
3188            ;
3188            ;    ***********************************************
3188            ;    *  CONVERSION ASCII- HEX DE DOS DIGITOS       *
3188            ;    *  NECESITA COMO PARAMETROS LOS REGISTROS     *
3188            ;    *  R2 Y R3 DEL GRUPO1. EL RESULTADO LO DEJA   *
3188            ;    *  EN EL REGISTRO R2.                         *
3188            ;    ***********************************************
3188  A6E247    ASCHEX      CP      R2,#47H
318E  9031F2                JP      GE,HERROR
31C1  A6E22F                CP      R2,#2FH
31C4  2D31F2                JP      LE,HERROR       ;VERIFICA DIGITOS
31C7  A6E347                CP      R3,#47H
31CA  9031F2                JP      GE,HERROR
31CD  A6E32F                CP      R3,#2FH
31D0  2D31F2                JP      LE,HERROR       ;DIGITO MENOS SIGNIFICATIVO
31D3                                                ;   COMIENZA CONVERSION
31D3  A6E240                CP      R2,#40H
31D6  8B05                  JR      GT,UNO1
31D8  26E230                SUB     R2,#30H
31DB  8B03                  JR      CONTIN1
31DD  26E237    UNO1        SUB     R2,#37H
31E0  A6E340    CONTIN1     CP      R3,#40H
31E3  8B05                  JR      GT,UNO2
31E5  26E330                SUB     R3,#30H
31E8  8B03                  JR      CONTIN2
31EA  26E337    UNO2        SUB     R3,#37H
31ED  F8E2      CONTIN2     SWAP    R2
31EF  4223                  OR      R2,R3
31F1  AF                    RET
31F2                                        ;
31F2                                        ; MENSAJE DE ERROR
31F2                                        ;
31F2  6C30      HERROR      LD      R6,#MESS > 8
31F4  7CF0                  LD      R7,#MESS \ 256
31F6  063287                CALL    ENVSINT
31F9  8932AE                JP      PREPOOT
31FC            ;
```

```
SIS-Z8 SISTEMA DE DESARROLLO
31FC                    FORM
31FC            ;*******************************************************************
31FC            ;*                                                                 *
31FC            ;*          H   E   X   -   A   S   C                              *
31FC            ;*                                                                 *
31FC            ;*******************************************************************
31FC            ;
31FC            ;     ***************************************************
31FC            ;     * RUTINA DE CONVERSION DE CODIGO HEXADECIMAL     *
31FC            ;     * A ASCCI. NECESITA COMO PARAMETROS A R2 Y        *
31FC            ;     * ( R3 EN EL CASO DE UNA CONVERSION DE 4         *
31FC            ;     * DIGITOS) .                                     *
31FC            ;     ***************************************************
31FC  70EA      HEXASC      PUSH        R10
31FE  E61A5E                LD          REG,#5EH
3201  F5E21A      INICIO      LD          @REG,R2
3204  571A0F                AND         @REG,#00001111B
3207  201A                  INC         REG
3209  F5E21A                LD          @REG,R2
320C  571AF0                AND         @REG,#11110000B
320F  F05F                  SWAP        5FH
3211  E61A5E                LD          REG,#5EH
3214  A71A09      CIGUE       CP          @REG,#09H
3217  AB05                  JR          GT,LETRAS
3219  071A30                ADD         @REG,#30H
321C  8B03                  JR          CONTI
321E  071A37      LETRAS      ADD         @REG,#37H
3221  201A        CONTI       INC         REG
3223  A61A60                CP          REG,#60H
3226  EBEC                  JR          NE,CIGUE
3228  50EA                  POP         R10
322A  AF                    RET
3228            ;
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3228                    FORM
3228            ;*******************************************************************
3228            ;*                                                                 *
3228            ;*                    RUTINA  AUXILIAR  A  DESP                    *
3228            ;*                                                                 *
3228            ;*******************************************************************
3228  8F        OTRA        DI
322C  E6F810              LD          IMR,#00010000B
322F  E6F000              LD          SIO,#NUL
3232  D6325A              CALL        ESPEPA
3235  E45FF0              LD          SIO,5FH
3238  D6325A              CALL        ESPERA
323B  E45EF0              LD          SIO,5EH
323E  D6325A              CALL        ESPERA
3241  AF                  RET
3242            ;*******************************************************************
3242            ;*                                                                 *
3242            ;*                M     O     D     O     S                      *
3242            ;*                                                                 *
3242            ;*******************************************************************
3242            ;
3242            ;     ***************************************************
3242            ;     * RUTINA PARA CAMBIAR BYTE DE CONTROL              *
3242            ;     * DEL PUERTO INTERNO. R2 CONTIENE EL BYTE DE       *
3242            ;     * CONTROL .                                        *
3242            ;     ***************************************************
3242  4CF0      MODOS       LD          R4,#0F0H
3244  D224                  LDC         @RR4,R2
3246  AF                    RET
3247            ;*******************************************************************
3247            ;*                                                                 *
3247            ;*                    RUTINA PARA CAMBIAR DE LINEA                 *
3247            ;*                                                                 *
3247            ;*******************************************************************
3247  E6F00B    CRLF        LD          SIO,#NUL
324A  D6325A              CALL        ESPERA
324D  E6F00D              LD          SIO,#CR
3250  D6325A              CALL        ESPERA
3253  E6F00A              LD          SIO,#LF
3256  D6325A              CALL        ESPERA
3259  AF                  RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
325A                    FORM
325A                ;***************************************************************************
325A                ;*                                                                         *
325A                ;*                E     S     P     E     R     A                      *
325A                ;*                                                                         *
325A                ;***************************************************************************
325A                ;
325A                ;     ***************************************************
325A                ;     * DETERMINADO CUANDO SE HA TERMINADO DE ENVIAR   *
325A                ;     * UN CARACTER Y SI EN EL INTER SE HA RECIBIDO    *
325A                ;     * UNO MAS.                                       *
325A                ;     ***************************************************
325A  8F            ESPERA      DI
325B  E6FB18                    LD          IMR,#00011000B
325E  76FA10        BUSCAINT    TM          IRQ,#10H                ; MUESTREA IRQ4
3261  EB19                     JR          NZ,SALIDA
3263  76FA08                    TM          IRQ,#08H                ; MUESTREA IRQ3
3266  6BF6                     JR          Z,BUSCAINT
3268  28F0                     LD          R2,SIO                  ; SE RECIBE UN CARACTER
326A  A6121B                   CP          12H,#27                 ; SI ES ESC
326D  603F                     JR          EQ,PREBOOT              ; ABORTA
326F  A61213                   CP          12H,#XOFF               ; SI ES XOFF
3272  EBEA                     JR          NE,BUSCAINT
3274  E6FA00                   LD          IRQ,#00H
3277  76FA03        XON         TM          IRQ,#08H                ; ENTRA A ESTE LOOP
327A  6BFB                     JR          Z,XON                   ; DEL CUAL SALE HASTA RECIBIR
327C  E6FA00        SALIDA      LD          IRQ,#00H                ; NUEVO CARACTER
327F  AF                       RET
3280                ;***************************************************************************
3280                ;*                                                                         *
3280                ;*     RUTINA PARA DESPLEGAR UN NUMERO HEXADECIMAL EN LA CONSOLA           *
3280                ;*                                                                         *
3280                ;***************************************************************************
3280  D631FC        DESP        CALL        HEXASC
3283  D6322B                    CALL        OTRA
3286  AF                        RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3287                    FORM
3287            ;*****************************************************************
3287            ;*                                                               *
3287            ;*          E   H   V   S   I   H   T                            *
3287            ;*                                                               *
3287            ;*****************************************************************
3287            ;     ***************************************************
3287            ;     * SE ENCARGA DE ENVIAR CADENAS DE MENSAJES       *
3287            ;     * SIN USO DE INTERRUPCIONES. LA RUTINA REGRESA    *
3287            ;     * CUANDO EL CARACTER A ENVIAR ES UN FF HEXA-      *
3287            ;     * DECIMAL                                         *
3287            ;     ***************************************************
3287  D6329C    ENVSINT     CALL        ZIGUE
328A  D6325A    ENVSININT   CALL        ESPERA          ;ENVIO SIN INTERRUPCION
328D  C226                  LDC         R2,@RR6         ;APUNTADOR AL LETRERO DE STATUS
328F  A612FF                CP          12H,@0FFH       ; SI ES FINAL DE LA CADENA REGRESA
3292  6D3298                JP          EQ,PREREG       ; DESPLIEGA NUMERO DE SECTOR
3295  D6329C                CALL        ZIGUE
3298  8D328A                JP          ENVSININT
329B  AF          PREREG    RET
329C  C226        ZIGUE     LDC         R2,@RR6         ; ENVIO DE CARACTERES
329E  29F0                  LD          SIO,R2
32A0  A0E6                  INCW        RR6
32A2  AF                    RET
32A3            ;*****************************************************************
32A3  CC30        PARAK     LD          R12,#30H
32A5  DC54                  LD          R13,#54H
32A7  924C                  LDE         @RR12,R4
32A9  A0EC                  INCW        RR12
32AB  925C                  LDE         @RR12,R5
32AD  AF                    RET
32AE            ;*****************************************************************
32AE            ;*                                                               *
32AE            ;*              RUTINA  DE  REGRESO  AL  MONITOR                 *
32AE            ;*                                                               *
32AE            ;*****************************************************************
32AE  E6FA00    PREROOT     LD          IRQ,#00H
32B1  6C3D                  LD          R6,@PPROMPT>8
32B3  7CF3                  LD          R7,@PPROMPT\256         ; ENVIA EL PROMPT A CONSOLA
32B5  D63287                CALL        ENVSINT
32B8  D6333D                CALL        CAM3                    ; REGRESA AL ESTADO 3
32BB  D63375                CALL        LIMPIAB                 ; PONE EN CEROS LAS LOCALIDADES
32BE  0C50                  LD          R0,#50H                 ; 3000 - 300F
32C0  E6FF40                LD          SPL,#40H
32C3  E6FE40                LD          SPH,#40H
32C6  8D304B                JP          ROOT
32C9            ;*****************************************************************
32C9  E6F020      ESPACIO   LD          SIO,#ESP
32CC  D6325A                CALL        ESPERA
32CF  AF                    RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
32D0                    FORM
32D0            ;**********************************************************
32D0            ;***              R E C O I R                           ***
32D0            ;**********************************************************
32D0            ;**********************************************************
32D0            ;*  SIRVE PARA RECONOCER DIRECCIONES EN CODIGO ASCII, ESTE *
32D0            ;*  RECONOCIMIENTO LO REALIZA DE 2 EN 2 DIGITOS           *
32D0            ;**********************************************************
32D0  AC00      RECDIR  LD      R10,#00H    ; INICIALIZA EL CONTADOR R10 CON 00H
32D2  4C30              LD      R4,#30H     ; APUNTA CON RR4 A LA DIRECCION 3000H
32D4  5C00              LD      R5,#00H
32D6  3C10              LD      R3,#10H     ; INICIALIZA EL CONTADOR R3 CON 10H
32D8  92A4      LIMPIA  LDE     @RR4,R10    ; LLENA DE CEROS DESDE LA DIRECCION
32DA  A0E4              INCW    RR4         ; 3000H HASTA LA 3010H
32DC  3AFA              DJNZ    R3,LIMPIA
32DE  4C30              LD      R4,#30H     ; APUNTA NUEVAMENTE A LA DIRECCION 3000H
32E0  5C00              LD      R5,#00H
32E2  A710Z0    INI     CP      R10H,#ESP   ; COMPARA EL CONTENIDO DEL BUFFER R0 CON
32E5  E803              JR      NE,CCOMA    ; 0 SI NO ES 0 SE VA A CCOMA
32E7  0E                INC     R0          ; INCREMENT EL APUNTADOR AL BUFFER R0
32E8  8BF8              JR      INI         ; REGRESA A INI A HACER NUEVAMENTE LA
32EA  A7102C    CCOMA   CP      R10H,#COMA  ; COMPARACION
32ED  EB2E              JR      NE,ERR      ; SI NO ES COMA SE VA A ERR
32EF  B81A              LD      R11,1AH     ; GUARDA EN R11 EL VALOR DE R10
32F1  00E0      DIR1    DEC     R0          ; DECREMENTA EL APUNTADOR AL BUFFER R0
32F3  E330              LD      R3,@R0      ; CARGA R3 CON EL DIGITO CONTENIDO EN EL BUFFER
32F5  2C30              LD      R2,#30H     ; CARGA R2 CON UN 30H
32F7  D631B8            CALL    ASCHEX      ; CONVIERTE EL DIGITO DEL CARACTER ASCII A
32FA  9224              LDE     @RR4,R2     ; HEXADECIMAL
32FC  A0E4              INCW    RR4         ; GUARDA EN LA DIRECCION APUNTADA POR RR4 EL
32FE  AAF1              DJNZ    R10,DIR1    ; DIGITO EN HEXADECIMAL
3300  4C30      GDAREG  LD      R4,#30H     ; SI R10 ES DIFERENTE DE CERO LO DECREMENTA Y SE
3302  5C00              LD      R5,#00H     ; VA A DIR1
3304  D63697            CALL    GDA         ; GUARDA EL CONTENIDO DE LA DIRECCION APUNTADA EN R2
3307  9812              LD      R9,12H
3309  A0E4              INCW    RR4
330B  D63697            CALL    GDA
330E  8812              LD      R8,12H      ; GUARDA EL DIGITO CONTENIDO EN R2 EN EL REG R8
3310  928C              LDE     @RR12,R8    ; GUARDA LOS 2 DIGITOS RECONOCIDOS EN LA DIRECCION
3312  A0EC              INCW    RR12        ; APUNTADA POR RR12
3314  929C              LDE     @RR12,R9
3316  A0EC              INCW    RR12
3318  8E                INC     R11
3319  0E      DEC     INC     R0          ; INCREMENTA EL APUNTADOR AL BUFFER
331A  BAFD              DJNZ    R11,DEC     ;  HASTA QUE R11 VALGA 0
331C  AF                RET                 ; REGRESA
```

```
SIS-Z8 SISTEMA DE DESARROLLO
331D                    FORM
331D  0E        ERR     INC     R0          ;  INCREMENTA APUNTADOR AL BUFFER
331E  AE                INC     R10         ;  INCREMENTA EL CONTADOR R10
331F  A6EA05            CP      R10,#05H    ;  COMPARA EL CONTDOR CON 5H
3322  1BC6              JR      LT,CCOMA    ;  SI ES MENOR SE VA A CCOMA
3324  6C3E      ERR1    LD      R6,#LET>8   ;  MANDA EL MENSAJE DE "CARACTER INVALIDO"
3326  7C34              LD      R7,#LET\256
3328  D63287            CALL    ENVSIHT     ;  LO ENVIA A LA CONSOLA
332B  8D32AE            JP      PRERDOT     ;  REGRESA A PRERDT
332E                                        ;  FIN DE LA RUTINA QUE RECONOCE UNA DIRECCION;
332E            ;***************************************************************************
332E  4C30      SALVAC  LD      R4,#30H
3330  5C50              LD      R5,#50H
3332  D63348            CALL    DRATA
3335  2813              LD      R2,13H
3337  A0E4              INCW    RR4
3339  D63348            CALL    DRATA
333C  AF                RET
333D            ;***************************************************************************
333D  4C30      CAM3    LD      R4,#TRES>8
333F  5C80              LD      R5,#TRES\256
3341  AF                RET
3342  82c4    DATA    LDE     R2,@RR4
3344  AF                RET
3345  C224    PROG    LDC     R2,@RR4
3347  AF                RET
3348  9224    DRATA   LDE     @RR4,R2
334A  AF                RET
334B  D224      PRAG    LDC     @RR4,R2
334D  AF                RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
334E                    FORM
334E                ;***************************************************************************
334E  4C30          POD         LD          R4,#30H
3350  5C52                      LD          R5,#52H
3352  A71044                    CP          @10H,#44H
3355  680F                      JR          EQ,DAT
3357  6C33                      LD          R6,#PROG)8
3359  7C45                      LD          R7,#PROG\256
335B  AC33                      LD          R10,#PRAG)8
335D  BC4B                      LD          R11,#PRAG\256
335F  2C20                      LD          R2,#ESP
3361  063348                    CALL        DRATA
3364  880E                      JR          FINPOD
3366  6C33          DAT         LD          R6,#DATA)8
3368  7C42                      LD          R7,#DATA\256
336A  AC33                      LD          R10,#DRATA)8
336C  BC48                      LD          R11,#DRATA\256
336E  2C44                      LD          R2,#44H
3370  063348                    CALL        DRATA
3373  0E                      INC         R0
3374  AF            FINPOD    RET
3375                ;***************************************************************************
3375  0C50          LIMPIAB     LD          R0,#50H
3377  1C20                      LD          R1,#20H
3379  E7E020        GHUY        LD          @R0,#ESP
337C  9E                      INC         R0
337D  1AFA                    DJNZ        R1,GHUY
337F  AF                      RET
3380                ;
3380                ;
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3380                    FORM
3380                ;**********************************************************************
3380                ;**********************************************************************
3380                ;**********                                              **********
3380                ;**********          C  O  M  A  N  D  O  S              **********
3380                ;**********                                              **********
3380                ;**********************************************************************
3380                ;**********************************************************************
3380                ;
3380                ;
3380                ;     ********************************************************
3380                ;     *          T     R     M                               *
3380                ;     ********************************************************
3380  2C02  TRM         LD        R2,#02
3382  D63242            CALL      MODOS       ; CAMBIA A MODO 2 DEL PUERTO SERIE
3385  8C80              LD        R8,#80H
3387  D63127  MOH       CALL      AGUANTA
338A  A61D0D              CP        12H,#0DH    ;  ESPERA HASTA RECIBIR CTRLF
338D  6B04              JR        EQ,RECIBE   ;    PARA EMPEZAR A
338F  29F0              LD        SIO,R2      ;    ALMACENAR DATO
3391  8BF4              JR        MOH
3393  9C80  RECIBE     LD        R9,#80H
3395  D633D7            CALL      DDCP
3398  C812              LD        R12,12H
339A  D633D7            CALL      DDCP
339D  A6E210            CP        R2,#10H
33A0  9B0F              JR        GE,PREPRO
33A2  06E230            ADD       R2,#30H
33A5  F812              LD        R15,12H
33A7  E71044            LD        @10H,#44H
33AA  D6334E            CALL      POD
33AD  481F              LD        R4,1FH
33AF  8B0A              JR        PBAJA
33B1  E71000  PREPRO    LD        @10H,#00H
33B4  F812              LD        R15,12H
33B6  D6334E            CALL      POD
33B9  481F              LD        R4,1FH      ;   ALMACENA DIRECCION
33BB  D633D7  PBAJA     CALL      DDCP        ;   EN R4 Y R5
33BE  5812              LD        R5,12H
33C0  D633D7            CALL      DDCP        ;  COMIENZA LECTURA Y ALMACENAMIENTO
33C3  D633D7  TRFR      CALL      DDCP
33C6  04EA              CALL      @RR10       ;  DE BYTES DEL REGISTRO
33C8  A0E4              INCW      RR4
33CA  CAF7              DJNZ      R12,TRFR
33CC  D633D7            CALL      DDCP
33CF  A6E900            CP        R9,#00H     ;  R9 ES EL REGISTRO DE CHECKSUM
33D2  6B8F              JR        EQ,RECIBE
33D4  8E                INC       R8          ;  R8 RECIBE CONTADOR DE ERRORES
33D5  8BBC              JR        RECIBE
```

```
SIS-Z8 SISTEMA DE DESARROLLO
33D7                    FORM
33D7            ;*****************************************************************************
33D7            ;*                                                                           *
33D7            ;*                        D     D     C     P                              *
33D7            ;*                                                                           *
33D7            ;*****************************************************************************
33D7            ;
33D7            ;     ***************************************************************
33D7            ;     *  EN ESTA RUTINA SE RECIBEN CARACTERES Y SE INTERPRETAN      *
33D7            ;     *  DIRECTAMENTE DE LA COMPUTADORA ANFITRIONA                  *
33D7            ;     *  EL RESULTADO DE ESTA RUTINA QUEDA EN R2                    *
33D7            ;     ***************************************************************
33D7 D63127     DDCP    CALL    AGUANTA
33DA A61230             CP      12H,#30H
33DD 1D33F4             JP      LT,PRCF
33E0 A6123A             CP      12H,#3AH
33E3 68F2               JP      EQ,DDCP
33E5 1812               LD      R1,12H
33E7 D63127             CALL    AGUANTA
33EA 3812               LD      R3,12H
33EC 2811               LD      R2,11H
33EE D63188             CALL    ASCHEX
33F1 0292               ADD     R9,R2           ; AQUI SE HACE LA SUMA PARA CHECKSUM
33F3 AF                 RET
33F4 A61206     PRCF    CP      12H,#CTRLF      ; AQUI SE PRUEBA SI SE RECIBE EL SEGUNDO
33F7 EBDE               JR      NE,DDCP         ; SEGUNDO CTRL F
33F9 F815               LD      R15,15H
33FB E814               LD      R14,14H
33FD 2C01               LD      R2,#01H         ; DE SER CTRL F REGRESA AL MODO I PUERTO
33FF D63242             CALL    MODOS           ; SERIE
3402 D63247             CALL    CRLF
3405 6C3F               LD      R6,#SIGM>8
3407 7C22               LD      R7,#SIGM\256
3409 D63287             CALL    ENVSINT
340C A0EE               INCW    RR14
340E 281E               LD      R2,1EH
3410 D63280             CALL    DESP
3413 281F               LD      R2,1FH
3415 D63280             CALL    DESP
3418 D63247             CALL    CRLF
341B 6C3F               LD      R6,#TRERR>8     ; MENSAJE DEL NUMERO DE ERRORES
341D 7C9E               LD      R7,#TRERR\256
341F D63287             CALL    ENVSINT
3422 D632C9             CALL    ESPACIO
3425 2818               LD      R2,18H
3427 D63280             CALL    DESP
342A 2C01               LD      R2,#01H         ; RETORNO A ROOT
342C D63242             CALL    MODOS
342F 8D32AE             JP      PREROOT
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3432                    FORM
3432            ;*******************************************************************
3432            ;*                      M     S     T                              *
3432            ;*******************************************************************
3432  2C03      MST       LD        R2,#03H
3434  D63242              CALL      MODOS           ; CAMBIA AL MODO 3 DEL PUERTO SERIE
3437  D63127    STAMD     CALL      AGUANTA
343A  A61206              CP        12H,#CTRLF      ;  Y ESPERA A RECIBIR CTRLF
343D  EB03                JR        NE,ESCL         ;  PARA REGRESAR AL MODO1
343F  2C01                LD        R2,#01H
3441  D63242              CALL      MODOS
3444  8D32AE              JP        PREROOT
3447  29F0      ESCL      LD        S10,R2
3449  8BEC                JR        STAMD
```

```
SIS-Z8 SISTEMA DE DESARROLLO
344B                    FORM
344B              ;*****************************************************************
344B              ;*     COMANDO   DE SUSTITUCION DE MEMORIA        S   M          *
344B              ;* NECESITA RECONOCER UNA DIRECCION ENCONTRADA EN LA LOCALIDA 3010 *
344B              ;* ESA LOCALIDAD SERA ALTERADA O NO , DEPENDIENDO DE LA LECTURA HECHA *
344B              ;* A TRAVES DE LA RUTINA ESPDAT; PARA SALIR DE SM ES NECESARIO TECLEAR *
344B              ;* ( ESC ) EN ESPDAT.                                            *
344B              ;*                                                               *
344B              ;*                                                               *
344B              ;*****************************************************************
344B  D6334E      SM          CALL        POD         ; RECONOCE LA OPCION O
344E  E81A                    LD          R14,1AH
3450  F81B                    LD          R15,1BH
3452  CC30                    LD          R12,#30H
3454  DC10                    LD          R13,#10H    ; RECUPERA DIRECCION; ESTA QUEDA ALMACENADA
3456  D6320D                  CALL        RECDIR      ; EN 3010 Y 3011H DE DATOS
3459  A81E                    LD          R10,1EH
345B  B81F                    LD          R11,1FH
345D  E816                    LD          R14,16H
345F  F817                    LD          R15,17H
3461  0C30                    LD          R0,#30H
3463  1C10                    LD          R1,#10H
3465  8240                    LDE         R4,@RR0
3467  A0E0                    INCW        RR0
3469  8250                    LDE         R5,@RR0
346B  D4EE        MANDAR      CALL        @RR14
346D  8812                    LD          R8,12H
346F  D63247                  CALL        CRLF
3472  D632C9                  CALL        ESPACIO
3475  2814                    LD          R2,14H
3477  D63280                  CALL        DESP
347A  2815                    LD          R2,15H
347C  D63280                  CALL        DESP        ; DESPLIEGA DIRECCION EN R4 Y R5
347F  D632C9                  CALL        ESPACIO
3482  D632C9                  CALL        ESPACIO
3485  2818                    LD          R2,18H
3487  D63280                  CALL        DESP        ; ENVIA EL CONTENIDO DE LA DIRECCION
348A  D632C9                  CALL        ESPACIO
348D  D63175                  CALL        ESPDAT      ; ESPERA DATO A SUSTITUIR
3490  A6E20D                  CP          R2,#CR
3493  6805                    JR          EQ,VEA1NC   ; SI ES CR INCREMENTA LOCALIDAD Y REGRESA A
3495  D6318B                  CALL        ASCHEX      ; MANDAR
3498  D4EA                    CALL        @RR10       ; SI ES UN DATO VALIDO LO MODIFICA
```

```
349A                    FORM
349A  A0E4      VERINC  INCW    RR4     ; INCREMENTA APUNTADOR Y LO REGRESA A MANDAR
349C  8BCD              JR      MANDAR
349E            ;*****************************************************************
349E            ;**                      D   R                                  *
349E            ;*****************************************************************
349E            ;
349E            ;*****************************************************************
349E            ;*    RUTINA QUE DESPLIEGA LOS REGISTROS DEL Z8                  *
349E            ;*    LOS REGISTROS SE DEBEN DE DAR EN FORMA HEXADECIMAL         *
349E            ;*****************************************************************
349E            DR
349E  DC00              LD      R13,#00H        ; R13 DETERMINA SI ES UN REGISTRO DEL
34A0  D634D0            CALL    RECREG          ; USUARIO O DEL SIS Z8
34A3  D634F8            CALL    NUMOK           ; LEE EL REGISTRO A DESPLEGAR Y LO VALIDA
34A6  A612F4    PRUEWO  CP      12H,#0F4H       ;  \
34A9  6B14              JR      EQ,INIREG       ;   ; VERIFICA QUE LOS REGISTROS A DES-
34AB  A612F0            CP      12H,#0F0H       ;   ; PLEGAR SEAN DE LECTURA/ESCRITURA
34AE  AB0F              JR      GT,INIREG       ;   ; . SI SON DE SOLO ESCRITURA MANDA
34B0  A612F3            CP      12H,#0F3H       ;   ; EL MENSAJE DE "REGISTRO DE SOLO
34B3  1B0A              JR      LT,INIREG       ;   ; ESCRITURA"
34B5  6C3E              LD      R6,#SOLOESC)8   ;   ;
34B7  7C4B              LD      R7,#SOLOESC\256 ;  /
34B9  D63287            CALL    ENUSINT
34BC  3D32AE            JP      PREPOOT
34BF  76E20F    INIREG  TM      R2,#0FH         ;  SI SE DESEA DESPLEGAR EL REGISTRO R0
34C2  6B06              JR      Z,DESGRUPO      ;  DE UN GRUPO DADO, SE DESPLIEGAN LOS
34C4  D63500            CALL    UNREG           ;  16 REGISTROS DE ESE GRUPO R0..R15
34C7  3D32AE            JP      PREPOOT         ;  CON UNREG DESPLIEGA UN SOLO REGISTRO
34CA  D63558    DESGRUPO CALL   GRUPO           ; DESPLIEGA TODO UN GRUPO DE REGISTROS
34CD  3D32AE            JP      PREPOOT
34D0            ;*****************************************************************
34D0            ;*    RUTINA QUE RECONOCE UN NUMERO DE REGISTRO Y MANDA A LA RUTINA  *
34D0            ;*    DE ASCHEX DEJANDO EL NUMERO DE REGISTRO A DESPLEGAR EN R2      *
34D0            ;*****************************************************************
34D0            ;
34D0            ;
34D0  A71020    RECREG  CP      #10H,#ESP       ; EN ESTA RUTINA SE LEE EL REGISTRO A DES-
34D3  EB03              JR      NE,USUARIO      ; PLEGAR SE UTILIZA R2 Y R3 PARA LA CONVER-
34D5  0E                INC     R0              ; SION ASCII-HEXADECIMAL
34D6  88F8              JR      RECREG
34D8  A71055    USUARIO CP      #10H,#55H
34DB  ED34E6            JP      NE,HUMUNO
34DE  DC01              LD      R13,#01H        ; SI EL REGISTRO A DESPLEGAR ES DE
34E0  0E        ESPO    INC     R0              ; USUARIO CARGA R13 CON 1
```

```
34E1                    FORM
34E1  A71020            CP          @10H,@ESP
34E4  6BFA              JR          EQ,ESP0
34E6  E320      NUMUNO  LD          R2,@R0          ; \
34E8  0E                INC         R0              ;  :
34E9  A71020    ESPACIO2 CP         @10H,@ESP       ;  :
34EC  EB03              JR          ME,NUMDOS       ;  : LEE EL NUMERO DE REGISTRO Y EL
34EE  0E                INC         R0              ;  : PRIMER DIGITO ASCII LO GUARDA EN R2
34EF  8BF8              JR          ESPACIO2        ;  : EL SEGUNDO DIGITO ASCII LO CARGA
34F1  E330      NUMDOS  LD          R3,@R0          ;  : R3
34F3  0E                INC         R0              ;  :
34F4  D631BB            CALL        ASCHEX          ;  : CONVIERTE EL NUMERO DADO EN ASCII A
34F7  AF                RET                         ;  / HEXADECIMAL
34F8            ;
34F8            ;*****************************************************************
34F8            ;*   RUTINA QUE VERIFICA QUE EL NUMERO DE REGISTRO DADO ESTE DENTRO   *
34F8            ;*   DE UN GRUPO CORRECTO QUE SON DE 00H-7EH , F0H-FFH               *
34F8            ;*****************************************************************
34F8            ;
34F8            ;
34F8  A61280    NUMOK   CP          12H,@80H
34FB  180F              JR          LT,FINUMOK
34FD  A612F9            CP          12H,@0F9H
3500  980A              JR          GE,FINUMOK
3502  6C3E              LD          R6,@REGINUV2B   ; SI NO ES UN NUMERO DE REGISTRO VALIDO
3504  7C68              LD          R7,@REGINUV256  ; MANDA EL MENSAGE
3506  D63287            CALL        ENVSINT         ; "NUMERO DE REGISTRO INVALIDO"
3509  8D32AE            JP          PREROOT
350C  AF        FINUMOK RET
350D            ;*****************************************************************
350D            ;*   RUTINA QUE DESPLIEGA UN SOLO REGISTRO . EL REGISTRO A DESPLEGAR  *
350D            ;*   ESTA EN R2 . SE DESPLEGA EN LA FORMA  RXXH=XXH                   *
350D            ;*****************************************************************
350D            ;*
350D  D635DB    UNREG   CALL        RXX
3510  A61D01              CP        1DH,@01H
3513  ED3549              JP        NE,REG28
3516  EC45                LD        R14,@45H
3518  FC00                LD        R15,@00H
351A  A6127F              CP        12H,@7FH
351D  AD3533              JP        GT,REST
3520  02F2                ADD       R15,R2
3522  822E                LDE       R2,@RR14
3524  D63280              CALL      DESP
3527  E6F048              LD        SIO,@HACHE
352A  D6325A              CALL      ESPERA
352D  D632C9              CALL      ESPACIO
3530  8D3557              JP        FINUH
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3533                    FORM
3533  26E2E0    REST    SUB     R2,#0E0H
3536  02F2              ADD     R15,R2
3538  822E              LDE     R2,@RR14
353A  D63280            CALL    DESP
353D  E6F048            LD      SIO,#HACHE
3540  D6325A            CALL    ESPERA
3543  D632C9            CALL    ESPACIO
3546  8D3557            JP      FINUN
3549  E325      REGZ8   LD      R2,@R5
354B  D63280            CALL    DESP
354E  E6F048            LD      SIO,#HACHE
3551  D6325A            CALL    ESPERA
3554  D632C9            CALL    ESPACIO
3557  AF        FINUN   RET
3558                    ;*************************************************
3558                    ;*  RUTINA QUE DESPLIEGA UN GRUPO DE REGISTROS  R2 ES UN CONTADOR DE *
3558                    ;*  16 REGISTROS                                   *
3558                    ;*************************************************
3558                    ;
3558  8C00      GRUPO   LD      R8,#00H
355A  D6350D      GRUP1   CALL    UNREG
355D  8E                  INC     R8
355E  5E                  INC     R5
355F  2815                LD      R2,15H
3561  A6130F              CP      13H,#0FH
3564  AB02                JR      GT,FINGRUPO
3566  88F2                JR      GRUP1
3568  AF        FINGRUPO  RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3569                    FORM
3569            ;
3569            ;*****************************************************************
3569            ;**                      M    A                                  **
3569            ;*****************************************************************
3569            ;
3569            MA
3569            ;*************************************************************
3569            ;*    MANDA EN CODIGO ASCII DESDE 20H(ESPACIO) HASTA 7EH(~) *
3569            ;*    EN R15 SE CARGA EL CARACTER A MANDAR, A TRAVES DEL SIO*
3569            ;*    R14 ES UN CONTADOR DE 30 CARACTERES POR LINEA SEPARA- *
3569            ;*    DOS POR UN ESPACIO                                    *
3569            ;*************************************************************
3569  FC20      ENVASC      LD          R15,#20H
356B  EC00      LINEA0X     LD          R14,#00H
356D  D63247                CALL        CRLF
3570  F9F0      LINEA       LD          SIO,R15
3572  D6325A                CALL        ESPERA
3575  E6F020                LD          SIO,#ESP
3578  D6325A                CALL        ESPERA
357B  FE                    INC         R15
357C  EE                    INC         R14
357D  A61F80                CP          1FH,#80H
3580  6D32AE                JP          EQ,PREFOOT
3583  A61E10                CP          1EH,#10H
3586  6BE3                  JR          EQ,LINEA0X
3588  9BE6                  JR          LINEA
```

```
SIS-Z8 SISTEMA DE DESARROLLO
358A                    FORM
358A                ;*************************************************************
358A                ;***                    M    R                              ***
358A                ;*************************************************************
358A                ;
358A                ;*************************************************************
358A                ;*        RUTINA DE INTERCAMBIO DE REGISTROS                 *
358A                ;*        LOS REGISTROS SE DAN DE LA FORMA MR 30H            *
358A                ;*************************************************************
358A                ;
358A                ;
358A  DC00          MR          LD          R13,#00H    ; EN R13 SE DETERMINA SI ES REGISTRO DE USUARIO
358C  D634D9                    CALL        RECREG      ; O DEL SIS Z8
358F  D634F8                    CALL        NUMOK       ; LEE Y VALIDA EL NUMERO DE REGISTRO
3592  A612F4                    CP          12H,#0F4H   ; \
3595  ABAF                      JR          EQ,CONTMR   ; |   MODIFICA REGISTROS DE LECTURA/ESCRITURA
3597  A612F3                    CP          12H,#0F3H   ; |   Y SI EL REGISTRO ES DE SOLO ESCRITURA
359A  189A                      JR          LT,CONTMR   ; |   DESPLIEGA SU NUMERO Y SOLO LEE SU NUEVO
359C  A612F9                    CP          12H,#0F9H   ; |   CONTENIDO
359F  AB05                      JR          GT,CONTMR   ; |
35A1  0635D8                    CALL        RXX         ; /
35A4  8B03                      JR          NOWO
35A6  0635DD        CONTMR      CALL        UNREG
35A9  063175        NOWO        CALL        ESPDAT
35AC  A6120D                    CP          12H,#CR
35AF  6D32AE                    JP          EQ,PREROOT
35B2  063188                    CALL        ASCHEX      ; \
35B5  A61D01                    CP          1DH,#01H    ; |
35B8  ED35D6                    JP          NE,MRUSUA   ; |
35BB  EC45                      LD          R14,#45H    ; |
35BD  FC00                      LD          R15,#00H    ; |  SI SON REGISTROS DEL USUARIO LOS ACCESA
35BF  A6157F                    CP          15H,#07FH   ; |  A PARTIR DE LA LOALIDAD 4500H Y LOS
35C2  8D35CC                    JP          GT,REST1    ; |  MODIFICA
35C5  02F5                      ADD         R15,R5      ; |
35C7  922E                      LDE         @RR14,R2    ; |
35C9  8D32AE                    JP          PREROOT     ; |
35CC  26E5E0        REST1       SUB         R5,#0E0H    ; |
35CF  02F5                      ADD         R15,R5      ; |
35D1  922E                      LDE         @RR14,R2    ; |
35D3  8D32AE                    JP          PREROOT     ; |
35D6  F352          MRUSUA      LD          @R5,R2      ; |
35D8  8D32AE                    JP          PREROOT     ; /
35DB                ;
35DB                ;*************************************************************
35DB                ;
```

```
SIS-Z8 SISTEMA DE DESARROLLO
35DB                        FORM
35DB              RXX
35DB  D63247                CALL        CRLF            ; RUTINA PARA IMPRIMIR EL NUMERO DE
35DE  5812                  LD          R5,12H          ; REGISTRO A DESPLEGARSE EN LA FORMA
35E0  E6F052                LD          SIO,#52H        ; RXX H
35E3  D6325A                CALL        ESPERA
35E6  D63280                CALL        DESP
35E9  E6F048                LD          SIO,#HACHE
35EC  D6325A                CALL        ESPERA
35EF  E6F03D                LD          SIO,#IGUAL
35F2  D6325A                CALL        ESPERA
35F5  D632C9                CALL        ESPACIO
35F8  AF                    RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
35F9  .                    FORM
35F9                ;*****************************************************************************
35F9                ;*****************************************************************************
35F9                ;***                        D     M                                        ***
35F9                ;*****************************************************************************
35F9                ;*****************************************************************************
35F9                ;
35F9                ;*************************************************************
35F9                ;*  DM SIRVE PARA HACER EL DESPLIEGUE DE MEMORIA DE UN BLOQUE DE  *
35F9                ;*  DIRECCIONES. PRIMERO DESPLIEGA LA DIRECCION INICIAL DEL BLOQUE*
35F9                ;*  Y POSTERIORMENTE EL CONTENIDO DE LAS 16 LOCALIDADES SIGUIENTES*
35F9                ;*  ASI COMO SU EQUIVALENCIA ASCII, ESTO LO HACE HASTA TERMINAR CON*
35F9                ;*  EL BLOQUE DE DIRECCIONES                                     *
35F9                ;*************************************************************
35F9                ;*************************************************************
35F9  2C35          DM          LD          R2,@DM>8
35FB  3CF9                      LD          R3,@DM\256
35FD  D63332                    CALL        SOLOC
3600  D6334E                    CALL        POD
3603  CC30                      LD          R12,#30H
3605  DC10                      LD          R13,#10H
3607  D632D0                    CALL        RECDIR
360A  D632D0                    CALL        RECDIR
360D  D63247                    CALL        CRLF
3610  D63684      DESDIR        CALL        DUNO
3613  3C02                      LD          R3,#02H
3615  CC30                      LD          R12,#30H
3617  DC10                      LD          R13,#10H
3619  822C        DESDIR1       LDE         R2,@RR12
361B  D63280                    CALL        DESP
361E  A0EC                      INCW        RR12
3620  3AF7                      DJNZ        R3,DESDIR1
3622  D632C9                    CALL        ESPACIO
3625  8C10                      LD          R8,#10H
3627  A24E        DESPL2        CP          R4,R14
3629  EB04                      JR          NE,SIG
362B  A25F                      CP          R5,R15
362D  6B0F                      JR          EQ,DESPL3
362F  D4E6        SIG           CALL        @RR6
3631  D63280                    CALL        DESP
3634  D632C9                    CALL        ESPACIO
3637  A0E4                      INCW        RR4
3639  D632A3                    CALL        PARAK
363C  8AE9                      DJNZ        R8,DESPL2
363E  A6E800      DESPL3        CP          R8,#00H
3641  6B05                      JR          EQ,DESPL4
3643  D632C9                    CALL        ESPACIO
3646  8AF6                      DJNZ        R8,DESPL3
3648  8C10        DESPL4        LD          R8,#10H
364A  D63684                    CALL        DUNO
364D  A24E        COMP          CP          R4,R14
364F  EB04                      JR          NE,SIG1
3651  A25F                      CP          R5,R15
3653  6B2C                      JR          EQ,FIN
```

```
3655                        FORM
3655  D4E6      SIG1        CALL    @RR6
3657  A61220                CP      12H,#20H
365A  AB02                  JR      GT,EV0
365C  2C2E                  LD      R2,#2EH
365E  29F0      EV0         LD      SIO,R2
3660  D6325A                CALL    ESPERA
3663  A0E4                  INCW    RR4
3665  D632A3                CALL    PARAK
3668  8AE3                  DJNZ    R8,COMP
366A  A24E      SALTO       CP      R4,R14
366C  EB04                  JR      NE,SIG2
366E  A25F                  CP      R5,R15
3670  6B0F                  JR      EQ,FIN
3672  D63247    SIG2        CALL    CRLF
3675  CC30                  LD      R12,#30H
3677  CC10                  LD      R13,#10H
3679  924C                  LDE     @RR12,R4
367B  A0EC                  INCW    RR12
367D  925C                  LDE     @RR12,R5
367F  8B8F                  JR      DESDIR
3681  8D32AE      FIN       JP      PREROOT
3684                DUHO
3684  CC30                  LD      P12,#30H
3686  DC10                  LD      R13,#10H
3688  824C                  LDE     R4,@RR12
368A  A0EC                  INCW    RR12
368C  825C                  LDE     R5,@RR12
368E  A0EC                  INCW    RR12
3690  82EC                  LDE     R14,@RR12
3692  A0EC                  INCW    RR12
3694  82FC                  LDE     R15,@RR12
3696  AF                    RET
3697  8224      GDA         LDE     R2,@RR4
3699  A0E4                  INCW    RR4
369B  8234                  LDE     R3,@RR4
369D  F0E3                  SWAP    R3
369F  4223                  OR      R2,R3
36A1  AF                    RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
36A2                    FORM
36A2            ;*****************************************************************
36A2            ;*                                                               *
36A2            ;*                          G      O                             *
36A2            ;*                                                               *
36A2            ;*****************************************************************
36A2  E3F8      GO          LD          R15,@R0
36A4  A6EF49                CP          R15,#'I'        ; DETERMINA LA OPCION I
36A7  ED368F                JP          NE,NORMAL
36AA  0E                    INC         R0
36AB  4C44                  LD          R4,#44H
36AD  5C00                  LD          R5,#00H         ; EN CASO DE I INICIALIZA
36AF  6C45                  LD          R6,#45H         ; ARCHIVO ALTERNO DE REGISTROS
36B1  7C00                  LD          R7,#00H
36B3  9C7F                  LD          R9,#7FH
36B5  82B4      TRANSFER    LDE         R8,@RR4
36B7  9286                  LDE         @RR6,R8
36B9  A0E4                  INCW        RR4
36BB  A0E6                  INCW        RR6
36BD  9AF6                  DJNZ        R9,TRANSFER
36BF  2C36      NORMAL      LD          R2,#GO>8        ; ESTA PARTE ES UTIL  PARA EJECUCION
36C1  3CA2                  LD          R3,#GO\256      ; POSTERIOR DE K
36C3  D6378C                CALL        COMUN
36C6  56FCFD    TAMGOK      AND         FLAGS,#11111101B ; APAGA BANDERA CALL / GO
36C9  76FC01                TM          FLAGS,#00000001B ; Y DETERMINA OPCION T
36CC  6D36D6                JP          EQ,ESCONT
36CF  2C85                  LD          R2,#85H
36D1  D63242                CALL        MODOS
36D4  8805                  JR          ALTERA
36D6  2C45      ESCONT      LD          R2,#45H
36D8  D63242                CALL        MODOS
36DB  2909      ALTERA      LD          09,R2
36DD  4C45                  LD          R4,#45H
36DF  5C1C                  LD          R5,#1CH
36E1  8214                  LDE         R1,@RR4
36E3  28FC                  LD          R2,FLAGS
36E5  56E203                AND         R2,#00000011B
36E8  4212                  OR          R1,R2
36EA  9214                  LDE         @RR4,R1         ; \
36EC  5C18                  LD          R5,#18H         ;  :
36EE  8214                  LDE         R1,@RR4         ;  :
36F0  56E104                AND         R1,#00000100B   ;  :   TRANSFIERE INFORMACION DEL
36F3  44E1F8                OR          P01M,R1         ;  :   STACK DEL ARCHIVO EXTERNO
36F6  5C1F                  LD          R5,#1FH         ;  :   A LOS REGISTROS DEL Z8
36F8  8214                  LDE         R1,@RR4         ;  :
36FA  19FF                  LD          SPL,R1          ;  :
36FC  80E4                  DECW        RR4             ;  :
36FE  8204                  LDE         R0,@RR4         ;  :
3700  89FE                  LD          SPH,R0          ; /
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3702                    FORM
3702  80E4              DECW    RR4             ; \
3704  8204              LDE     R0,@RR4         ;  ;
3706  80E4              DECW    RR4             ;  ;
3708  8214              LDE     R1,@RR4         ;  ;
370A  50E9              POP     R9              ;  ;  METE AL STACK RP Y STATUS DEL
370C  50E9              POP     R9              ;  ;  ARCHIVO ALTERNO ASI COMO
370E  50E9              POP     R9              ;  ;  DIRECCION DE ARRANQUE
3710  50E9              POP     R9              ;  ;
3712  70E1              PUSH    R1              ;  ;
3714  70ED              PUSH    R13             ;  ;
3716  70EC              PUSH    R12             ;  ;
3718  70E0              PUSH    R0              ;  /
371A  0809              LD      R0,09H          ; \
371C  EC45              LD      R14,#45H        ;  .
371E  FC10              LD      R15,#10H        ;  ;
3720  920E              LDE     @RR14,R0        ;  ;
3722  DC7B              LD      R13,#6FH+12     ;  ;
3724  EC45              LD      R14,#45H        ;  ;
3726  FC00              LD      R15,#00H        ;  ;  TRANSFIERE ARCHIVO ALTERNO AL
3728  CC00              LD      R12,#00H        ;  ;  ARCHIVO INTERNO EXCEPTO
372A  83CE        G027    LDEI    @R12,@RR14      ;  ;  REGISTROS DE CONTROL. ESTOS
372C  A6EC1C              CP      R12,#1CH        ;  ;  QUEDAN EN EL GRUPO 10.
372F  ED3736              JP      NE,ESTOSI       ;  ;
3732  CC20        ESTONO  LD      R12,#20H        ;  ;
3734  FC20              LD      R15,#20H        ;  ;
3736  A6EC02    ESTOSI  CP      R12,#02H        ;  ;
3739  EB06              JR      NE,NOES2        ;  ;
373B  820E              LDE     R0,@RR14        ;  ;
373D  70E0              PUSH    R0              ;  ;
373F  A6EE              INCW    RR14            ;  ;
3741  DAE7        NOES2   DJNZ    R13,G027        ;  /
3743  F817              LD      R15,17H         ; \
3745  56EF18            AND     R15,#00011000B  ;  ;
3748  680A              JR      Z,DMHAB         ;  ;
374A  76EF08              TM      R15,#00001000B  ;  ;  DETERMINA SI EN LOS REGISTROS DEL
374D  680E              JR      Z,PRENT         ;  ;  ARCHIVO ALTERNO ESTA HABILITADA
374F  76FF10              TM      R15,#00010000B  ;  ;  DM PARA HABILITAR LA SENAL DMHAB
3752  6809              JR      Z,PRENT         ;  ;  ANTES DE SALIAR AL MONITOR
3754  C810        DMHAB   LD      R12,19H         ;  ;  INTERNO.
3756  46EC20            OR      R12,#00100000B  ;  ;
3759  ECF0              LD      R14,#0F0H       ;  ;
375B  02CE              LDC     @RR14,R12       ;  ;
375D  8F          PRENT   DI                      ;  /
375E  80005D              JP      ENTRADA
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3761                    FORM
3761                ;*******************************************************************
3761                ;**                                                               **
3761                ;**              ESTO ES UTIL PARA POSTERIORES USOS DE CALL       **
3761                ;**                                                               **
3761                ;*******************************************************************
3761  2C37      CALL      LD        R2,#CALL>8
3763  3C61                LD        R3,#CALL\256
3765  D6378C              CALL      COMUN
3768  46FC02    TAMCAK    OR        FLAGS,#00000010B   ; PRENDE BANDERA CALL / GO
376B  76FC01              TM        FLAGS,#00000001B
376E  6D3779              JP        EQ,FUETRACE
3771  2C81                LD        R2,#81H
3773  D63242              CALL      MODOS
3776  ED377E              JP        INTERNA
3779  2C41      FUETRACE  LD        R2,#41H
377B  D63242              CALL      MODOS
377E  8F        INTERNA   DI
377F  E6FB01              LD        IMR,#00000001B
3782  8CD6                LD        R8,#11010110B
3784  7C50                LD        R7,#01010000B
3786  9F                  EI
3787  D4EC                CALL      @RR12              ; LLAMADA A LA DIRECCION DE ARRANQUE DEL
3789  8D32AE              JP        PREROOT            ; PROGRAMA DE PRUEBA
378C                ;*******************************************************************
378C                ;**                                                               **
378C                ;**              ESTA RUTINA ES COMUN A CALL Y GO                 **
378C                ;**                                                               **
378C                ;*******************************************************************
378C  D6332E  COMUN     CALL      SALVAC
378F  CC28              LD        R12,#28H
3791  DC10              LD        R13,#10H
3793  D6320D              CALL      RECDIR
3796  2C30              LD        R2,#30H
3798  3C10              LD        R3,#10H
379A  82C2              LDE       R12,@RR2
379C  A0E2              INCW      RR2
379E  82D2              LDE       R13,@RR2
37A0  AF                RET
```

```
37A1                    FORM
37A1                ;*******************************************************************
37A1                ;**                                                               **
37A1                ;**                  T  R  A  C  E                                **
37A1                ;**                                                               **
37A1                ;*******************************************************************
37A1  D634D0        T           CALL        RECREG
37A4  4C30                      LD          R4,#30H
37A6  5C90                      LD          R5,#90H
37A8  9224                      LDE         @RR4,R2         ; RECONOCE NUMERO DE INSTRUCCIONES A
37AA  46FC01                    OR          FLAGS,#00000019 ; SEGUIR Y QUEDAN DEPOSITADAS EN 3090
37AD  4C45                      LD          R4,#45H         ; PRENDE BANDERAS T / BP EN ARCHIVOS
37AF  5C1C                      LD          R5,#1CH         ; INTERNO Y ALTERNO
37B1  8214                      LDE         R1,@RR4
37B3  46E101                    OR          R1,#00000019
37B6  9214                      LDE         @RR4,R1
37B8  8D32AE                    JP          PREROOT
37BB                ;*******************************************************************
37BB                ;*******************************************************************
37BB                ;**                                                               **
37BB                ;**                  A P A G A    T R A C E                       **
37BB                ;**                                                               **
37BB                ;*******************************************************************
37BB                ;*******************************************************************
37BB  56FCFE        AT          AND         FLAGS,#11111110B
37BE  4C45                      LD          R4,#45H
37C0  5C1C                      LD          R5,#1CH         ; APAGA BANDERAS DE T / BP EN
37C2  8214                      LDE         R1,@RR4         ; ARCHIVOS INTERNO Y ALTERNO
37C4  56E1FE                    AND         R1,#11111110B
37C7  9214                      LDE         @RR4,R1
37C9  8D32AE                    JP          PREROOT
```

```
SIS-7 SISTEMA DE DESARROLLO
37CC                    FORM
37CC            ;***************************************************************
37CC            ;***                                                         ***
37CC            ;***      P    M       RUTINA QUE PRUEBA LA MEMORIA DE PROGRAMA     ***
37CC            ;***                    Y LA MEMORIA DE DATOS                ***
37CC            ;***                                                         ***
37CC            ;***************************************************************
37CC  EC3E      PM          LD          R14,$TAPM>8
37CE  FC14                  LD          R15,$TAPM\256
37D0  6C3F                  LD          R6,$VDE>8       ; MENSAJE DE "PRUEBA
37D2  7CB8                  LD          R7,$VDE\256     ; DE VERIFICACION DESPUES DE
37D4  D63287                CALL        ENVSINT         ; ESCRITURA "
37D7  D6383B    PRINC       CALL        C4KB
37DA  0C04      CUNKB       LD          R0,$04H         ; COMIENZA CICLO PRINCIPAL
37DC  3C00      C1D4KB      LD          R3,$00H         ; CONTADOR DE CARACTERES R3
37DE  CC0A      CONRET      LD          R12,$0AH        ; CONTADOR DE INTENTOS
37E0  2812      SIEMPRE     LD          R2,$12
37E2  D4E8                  CALL        @FR3            ; GUARDA CONTENIDO
37E4  D4EA                  CALL        @ER10           ; LEE CONTENIDO
37E6  A223                  CP          R2,R3
37E8  6805                  JR          EQ,PASA         ; VERIFICA IGUALDAD
37EA  CAF4                  DJNZ        R12,SIEMPRE     ; SI HAY ERROR HUECO INTENTO
37EC  D63849                CALL        FALLA           ; SI DESPUES DE 13 INTENTOS
37EF  A61C0A    PASA        CP          1CH,$0AH        ; MARCA ERROR
37F2  6803                  JR          EQ,NORETRIES    ; SI HUBO REINTENTO CONTINUA
37F4  D63865                CALL        RETRIES
37F7  A0E4      NORETRIES   INCW        RR4
37F9  3AE3                  DJNZ        R3,CONRET
37FB  0A0F                  DJNZ        R0,C1D4KB
37FD  1ADB                  DJNZ        R1,CUNKB
37FF  A6EF34                CP          R15,$TAPM\256+32 ; SI NO HAY MAS SECTORES CONTINUA
3802  EBD3                  JR          NE,PRINC
3804  6C3F                  LD          R6,$LDM>8
3806  7CE7                  LD          R7,$LDM\256     ; MENSAJE DE PRUEBA DE LECTURA
3808  D63287                CALL        ENVSINT
380B  EC3E                  LD          R14,$TAPM>8
380D  FC14                  LD          R15,$TAPM\256
```

```
380F                    FORM
380F  D6383B    CONLEC  CALL    C4KB            ; CICLO PRINCIPAL DE PRUEBA DE
3812  8C04      LUNK    LD      R8,#04H         ; LECTURA
3814  3C00      LC1D4   LD      R3,#00H         ; CONTADOR DE CARACTERES R3
3816  CC0A      LECR    LD      R12,#0AH
3818  D4EA      LSIEM   CALL    @RR10           ; LEE DIRECCION
381A  A223              CP      R2,R3           ; COMPARA CON CONTADOR DE CARACTERES
381C  6B05              JR      EQ,BIEN
381E  CAF8              DJNZ    R12,LSIEM       ; SI NO ES IGUAL NUEVO INTENTO
3820  D63849            CALL    FALLA           ; SI DESPUES DE 10 INTENTOS ERROR
3823  A61C0A    BIEN    CP      1CH,#0AH
3826  6B03              JR      EQ,NOINT
3828  D63865            CALL    RETRIES         ; SI HUBO INTENTOS DESPLIEGA ERROR
382B  A0E4      NOINT   INCW    RR4             ; DE REINTENTOS
382D  3AE7              DJNZ    R3,LECR
382F  0AE3              DJNZ    R9,LC1D4
3831  1ADF              DJNZ    R1,LUNK
3833  A6EF34            CP      R15,#TAPM\256+32 ; SI EXISTEN MAS SECTORES REGRESA
3836  EBD7              JR      NZ,CONLEC       ; AL CICLO PRINCIPAL
3838  8D32AE            JP      PREPOOT
383B            ;***************************************************************
383B  CC14      C4KB    LD      R12,#14H
383D  DC03              LD      R13,#03H
383F  C3CE      CAREG   LDCI    @R12,@RR14      ; OBTIENE INFORMACION REFERENTE
3841  DAFC              DJNZ    R13,CAREG       ; A CADA SECTOR DE LA TABLA DE
3843  1C08              LD      R1,#08H         ; PRUEBA DE MEMORIA
3845  D63287            CALL    ENVSINT
3848  AF                RET
3849  C812      FALLA   LD      R12,12H
384B  D813              LD      R13,13H
384D  D63075            CALL    DIRECCION       ; DESPLIEGUE DE ERRORES
3850  281D              LD      R2,1DH
3852  D63280            CALL    DESP
3855  D632C9            CALL    ESPACIO
3858  281C              LD      R2,1CH
385A  D63280            CALL    DESP
385D  D63247            CALL    CRLF
3860  CC0A              LD      R12,#0AH
3862  381D              LD      R3,1DH
3864  AF                RET
3865  DC0A      RETRIES LD      R13,#0AH
3867  22DC              SUB     R13,R12
3869  D63875            CALL    DIRECCION
386C  281D              LD      R2,1DH
386E  D63280            CALL    DESP
3871  D63247            CALL    CRLF
3874  AF                RET
3875  2814      DIRECCION LD    R2,14H          ; DESPLIEGUE DE ERRORES POR
3877  D63280            CALL    DESP            ; REINTENTOS
387A  2815              LD      R2,15H
387C  D63280            CALL    DESP
387F  D632C9            CALL    ESPACIO
3882  AF                RET
```

```
3883                    FORM
3883            ;*****************************************************************
3883            ;***                                                           ***
3883            ;***                      M     U                              ***
3883            ;***                                                           ***
3883            ;*****************************************************************
3883            ;
      *
3883            ;* RUTINA DE MOVIMIENTO DE BLOQUES DE MEMORIA. SE CONTEMPLAN 4          *
3883            ;* OPCIONES :                                                           *
3883            ;*   MVD :  MEMORIA DE DATOS A MEMORIA DE DATOS                         *
3883            ;*   MVP :  MEMORIA DE PROGRAMA A MEMORIA DE PROGRAMA                   *
3883            ;*   MVE :  MEMORIA DE PROGRAMA A MEMORIA DE DATOS                      *
3883            ;*   MVI :  MEMORIA DE DATOS A MEMORIA DE PROGRAMA                      *
3883            ;*****************************************************************
3883            ;
3883            MV
3883  CC40              LD        R12,#40H
3885  DC10              LD        R13,#10H
3887  930C              LDEI      @RR12,@R0                ; EL TIPO PUEDE SER MVX  DONDE X=D,P,E O I
3889                                                       ; DIRECCIONES DE MEMORIA A MOVER
3889  D6320D            CALL      RECDIR
388C  D6320D            CALL      RECDIR
388F  D6320D            CALL      RECDIR
3892                                                       ; RECUPERA DIRECCIONES Y TIPO DE MOVIMIENTO
3892  2C40              LD        R2,#40H
3894  3C10              LD        R3,#10H
3896  8212              LDE       R1,@RR2
3898  A0E2              INCW      RR2
389A  82A2              LDE       R10,@RR2
389C  A0E2              INCW      RR2
389E  82B2              LDE       R11,@RR2
38A0  A0E2              INCW      RR2
38A2  82C2              LDE       R12,@RR2
38A4  A0E2              INCW      RR2
38A6  82D2              LDE       R13,@RR2
38A8  A0E2              INCW      RR2
38AA  82E2              LDE       R14,@RR2
38AC  A0E2              INCW      RR2
38AE  82F2              LDE       R15,@RR2
3880            ;
3880            ;
3880                                                       ;  MANDA AL TIPO DE MOVIMIENTO ADECUADO
3880  A61144            CP        11H,#44H                 ; ES MVD ?
3883  6D38D2            JP        EQ,DATDAT
3886  A61150            CP        11H,#50H                 ; ES MVP
3889  6D38DD            JP        EQ,PROGPROG
388C  A61149            CP        11H,#49H                 ; ES MVI ?
388F  6D38E8            JP        EQ,DATPROG
38C2  A61145            CP        11H,#45H                 ; ES MVE ?
38C5  6D38F3            JP        EQ,PROGDAT
38C8  6C3F              LD        R6,@OPINVA)8
38CA  7C3D              LD        R7,@OPINVA\256
38CC  D63287            CALL      ENVSINT
38CF  8D390B            JP        FINMV
```

```
38D2                    FORM
38D2                    ;*****************************************************************
38D2                    ;***                 RUTINAS DE MOVIMIENTO DE BLOQUES DE MEMORIA    ****
38D2                    ;***                                                                ****
38D2                    ;*****************************************************************
38D2  6C33      DATDAT  LD      R6,#DATA>8
38D4  7C42              LD      R7,#DATA\256
38D6  8C33              LD      R8,#DRATA>8
38D8  9C48              LD      R9,#DRATA\256
38DA  8D38FB            JP      VETRAS                  ; VA A LA RUTINA DE TRASLAPE
38DD  6C33      PROGPROG LD     R6,#PROG>8
38DF  7C45              LD      R7,#PROG\256
38E1  8C33              LD      R8,#PRAG>8
38E3  9C4B              LD      R9,#PRAG\256
38E5  8D38FB            JP      VETRAS
38E8  6C33      DATPROG LD      R6,#DATA>8
38EA  7C42              LD      R7,#DATA\256
38EC  8C33              LD      R8,#PRAG>8
38EE  9C4B              LD      R9,#PRAG\256
38F0  8D38FB            JP      VETRAS
38F3  6C33      PROGDAT LD      R6,#PROG>8
38F5  7C45              LD      R7,#PROG\256
38F7  8C33              LD      R8,#DRATA>8
38F9  9C48              LD      R9,#DRATA\256
38FB  A2EC      VETRAS  CP      R14,R12
38FD  9B04              JR      GE,TRASNO
38FF  A2EA              CP      R14,R10
3901  AB05              JR      GT,TRASSI
3903  D63956    TRASNO  CALL    NOTRAS
3906  8B03              JR      FINMV
3908  D6390E    TRASSI  CALL    TRAS
390B  8D32AE    FINMV   JP      PREROOT
390E                    ;*****************************************************************
390E                    ;*   RUTINA QUE MUEVE BLOQUES DE MEMORIA EN EL CASO DE QUE       *
390E                    ;*   EXISTA TRASLAPE                                             *
390E                    ;*****************************************************************
390E                    ;
390E  A2D0      TRAS    CP      R13,R11
3910  1B0F              JR      LT,MEM1
3912  3B1D              LD      R3,1DH
3914  281C              LD      R2,1CH
3916  223B              SUB     R3,R11
3918  222A              SUB     R2,R10
391A  02F3              ADD     R15,R3
391C  12E2              ADC     R14,R2
391E  8D3935            JP      PROC            ; SE VA A PROCESO DE MOVIMIENTO
3921  281C      MEM1    LD      R2,1CH
3923  381B              LD      R3,1BH
3925  223D              SUB     R3,R13
3927  60E3              COM     R3
3929  06E301            ADD     R3,#01H
392C  222A              SUB     R2,R10
392E  26E201            SUB     R2,#01H
3931  02F3              ADD     R15,R3
3933  12E2              ADC     R14,R2
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3935                        FORM
3935  A2DB      PROC        CP          R13,R11
3937  ED3942                JP          NE,MUE1
393A  A2CA                  CP          R12,R10
393C  ED3942                JP          NE,MUE1
393F  8D3955                JP          FINTRAS
3942  581D      MUE1        LD          R5,1DH
3944  481C                  LD          R4,1CH
3946  D4E6                  CALL        @RR6            ; LEE
3948  581F                  LD          R5,1FH
394A  481E                  LD          R4,1EH
394C  D4E8                  CALL        @RR8            ; GRABA
394E  80EC                  DECW        RR12
3950  80EE                  DECW        RR14
3952  8D3935                JP          PROC
3955  AF        FINTRAS    RET
3956                        ;
3956                        ;*************************************************************
3956                        ;*  RUTINA QUE MUEVE BLOQUES DE MEMORIA EN EL CASO DE QUE NO EXISTA  *
3956                        ;*  TRASLAPE                                                 *
3956                        ;*************************************************************
3956                        ;
3956                        ;
3956                        NOTRAS
3956  A2BD      TRAS1       CP          R11,R13
3958  ED3963                JP          NE,MUE
395B  A2AC                  CP          R10,R12
395D  ED3963                JP          NE,MUE
3960  8D3976                JP          FINOTRAS
3963  481A      MUE         LD          R4,1AH
3965  581B                  LD          R5,1BH
3967  D4E6                  CALL        @RR6
3969  481E                  LD          R4,1EH
396B  581F                  LD          R5,1FH
396D  D4E8                  CALL        @RR8
396F  A0EA                  INCW        RR10
3971  A0EE                  INCW        RR14
3973  8D3956                JP          TRAS1
3976  AF        FINOTRAS    RET
```

```
3977                ;**********************************************************
3977                ;**********************************************************
3977                ;**                                                      **
3977                ;**      RUTINA DE CONTINUE PARA LOS COMANDOS GO Y CONTINUE **
3977                ;**                          K                           **
3977                ;**                                                      **
3977                ;**********************************************************
3977 4C30       K       LD      R4,#30H     ; \
3979 5C50               LD      R5,#50H     ;  :
397B 6C36               LD      R6,#GO03    ;  :
397D 7CA2               LD      R7,#GO\256  ;  :
397F 8234               LDE     R3,@RR4     ;  :
3981 A0E4               INCW    RR4         ;  :
3983 8294               LDE     R9,@RR4     ;  :    DETERMINA DE LAS LOCALIDADES
3985 A286               CP      R8,R6       ;  :    3050 - 3051 SI LA ULTIMA
3987 EB05               JR      NE,PSC      ;  :    DIRECCION FUE GO, CALL O DM
3989 A297               CP      R9,R7       ;  :
398B 6D39E3             JP      EQ,FG       ;  :
398E 6C37       PSC     LD      R6,#CALL+8  ;  :
3990 7C61               LD      R7,#CALL\256;  :
3992 A286               CP      R8,R6       ;  :
3994 EB05               JR      NE,COEME    ;  :
3996 A297               CP      R9,R7       ;  :
3998 6D39EE             JP      EQ,STC      ; /
399B 0C52       COEME   LD      R0,#52H     ;SI NO FUE GO DESPLIEGA MEMORIA
399D 5C52               LD      R5,#52H
399F 8304               LDEI    @R0,@RR4    ;COLOCA SI ES O O M
39A1 E7E020             LD      @R0,#ESP
39A4 0E                 INC     R0
39A5 8234               LDE     R3,@RR4
39A7 A0E4               INCW    RR4
39A9 8294               LDE     R9,@RR4
39AB 80E4               DECW    RR4
39AD D639D3             CALL    CLLENAB     ; SI FUE DM RESTAURA EL BUFFER DE
3980 D639D3             CALL    CLLENAB     ; RECEPCION, CON DIRECCION INICIAL
3983 E7E02C             LD      @R0,#COMA   ; IGUAL A LA ULTIMA DESPLEGADA
3986 0E                 INC     R0          ; POR DM + 1 Y LA ULTIMA COMO
3987 1CFF               LD      R1,#0FFH    ; DIR DM + 257 Y SALTA A DM
3989 A0E8       CINC    INCW    RR8
398B 1AFC               DJNZ    R1,CINC     ;DIRECCION + 256 LOCALIDADES
398D 9234               LDE     @RR4,R8
398F A0E4               INCW    RR4
39C1 9294               LDE     @RR4,R9
39C3 80E4               DECW    RR4
39C5 D639D3             CALL    CLLENAB
39C8 D639D3             CALL    CLLENAB     ;LLENA LA SEGUNDA DIRECCION
39CB E7E02C             LD      @R0,#COMA
39CE 0C52               LD      R0,#52H
3900 8035F9             JP      DM
```

```
SIS-Z8 SISTEMA DE DESARROLLO
39D3                    FORM
39D3  8224      CLLENAB LDE     R2,@RR4
39D5  D631FC            CALL    HEXASC
39D8  F55FE0            LD      @R0,5FH
39DB  0E                INC     R0
39DC  F55EE0            LD      @R0,5EH
39DF  A0E4              INCW    RR4
39E1  0E                INC     R0
39E2  AF                RET
39E3  5C54      FG      LD      R5,#54H
39E5  82D4              LDE     R13,@RR4        ; RECUPERA LA DIRECCION DE RETORNO
39E7  A0E4              INCW    RR4             ; Y SALTA A TAMOOK
39E9  82C4              LDE     R12,@RR4
39EB  8D36C6            JP      TAMOOK
39EE            ;***************************************************************
39EE  5C54      STC     LD      R5,#54H
39F0  82D4              LDE     R13,@RR4        ; RECUPERA LA DIRECCION DE RETORNO Y
39F2  A0E4              INCW    RR4             ; SALTA A TAMCAK
39F4  82C4              LDE     R12,@RR4
39F6  8D3768            JP      TAMCAK
39F9            ;***************************************************************
39F9            ;***                     D     B                             ***
39F9            ;***          RUTINA DE DESPLIEGUE DE BREAKPOINTS.             ***
39F9            ;***       REVISA LAS LOCALIDADES DE MEMORIA DE DATOS 3100 EN   ***
39F9            ;***       ADELANTE PARA OCHO BRAKPOINT                        ***
39F9            ;***************************************************************
39F9  4C31      DB      LD      R4,#31H
39FB  5C00              LD      R5,#00H
39FD  8C08              LD      R8,#08H         ;# DE BP'S A BARRER
39FF  2C00      DBMAX   LD      R2,#00H
3A01  8234              LDE     R3,@RR4
3A03  A223              CP      R2,R3
3A05  EB0A              JR      NE,DBSIES       ;VERIFICA QUE LA TABLA NO TENGA CEROS
3A07  A0E4              INCW    RR4
3A09  8234              LDE     R3,@RR4
3A0B  80E4              DECW    RR4
3A0D  A223              CP      R2,R3
3A0F  3B0D              JR      DBNDES
3A11  D63247    DBSIES  CALL    CRLF            ;PONE EN PANTALLA LA DIRECCION DE BP'S
3A14  D63A26            CALL    DOSBY
3A17  A0E4              INCW    RR4
3A19  8AE4      DBDEC   DJNZ    R3,DBMAX
3A1B  8D32AE            JP      PREBOOT
3A1E  A0E4      DBNDES  INCW    RR4             ; DE NO HABER BREAK POINTS RECORRE
3A20  A0E4              INCW    RR4             ; TABLA
3A22  A0E4              INCW    RR4
3A24  8BF3              JR      DBDEC
3A26  8224      DOSBY   LDE     R2,@RR4         ; DESPLIEGA DIRECCION
3A28  D63280            CALL    DESP
3A2B  A0E4              INCW    RR4
3A2D  8224              LDE     R2,@RR4
3A2F  D63280            CALL    DESP
3A32  A0E4              INCW    RR4
3A34  AF                RET
```

```
3A35                    FORM
3A35                ;*********************************************************
3A35                ;**                          B     P                      **
3A35                ;**                  RUTINA DE COLOCACION DE BREAK POINT    **
3A35                ;**                                                         **
3A35                ;*********************************************************
3A35  56FCFE    BP      AND     FLAGS,#11111110B
3A38  4C45              LD      R4,#45H
3A3A  5C1C              LD      R5,#1CH
3A3C  8214              LDE     R1,@RR4
3A3E  56E1FE            AND     R1,#11111110B
3A41  9214              LDE     @RR4,R1
3A43  CC30      BUNO    LD      R12,#30H
3A45  DC10              LD      R13,#10H
3A47  D6320D            CALL    RECDIR
3A4A  4C31              LD      R4,#31H
3A4C  5C00              LD      R5,#00H
3A4E  8C30              LD      R8,#30H
3A50  9C10              LD      R9,#10H
3A52  8268              LDE     R6,@RR8
3A54  A0E8              INCW    RR8
3A56  8278              LDE     R7,@RR8         ;  RR6 CONTINE LA DIRECCION DEL BP
3A58  D63A93            CALL    BUSVAC
3A5B  9264              LDE     @RR4,R6
3A5D  A0E4              INCW    RR4
3A5F  9274              LDE     @RR4,R7         ; CARGA EN LA TABLA
3A61  A0E4              INCW    RR4
3A63  C206              LDC     R0,@RR6
3A65  9204              LDE     @RR4,R0         ;  CARGA EL CONTENIDO EN LAS TABLAS
3A67  2CFF              LD      R2,#0FFH
3A69  A6E610            CP      R6,#10H
3A6C  AB07              JR      GT,COMOSIEM
3A6E  06E630            ADD     R6,#30H
3A71  9226              LDE     @RR6,R2
3A73  8802              JR      GUION
3A75  D226      COMOSIEM LDC    @RR6,R2         ; COLOCO UN NOP EN LA DIR
3A77  E6FC2D    GUION   LD      SIO,#2DH        ; ENVIAR UN GUION
3A7A  D6325A            CALL    ESPERA
3A7D  E6F041            LD      SIO,#41H        ;  ENVIAR UN B'P
3A80  D6325A            CALL    ESPERA
3A83  D632C9            CALL    ESPACIO
3A86  2816              LD      R2,16H
3A88  D63280            CALL    DESP
3A8B  2817              LD      R2,17H
3A8D  D63280            CALL    DESP
3A90  8D32AE            JP      PRERCOT
3A93  0C00      BUSVAC  LD      R0,#00H
3A95  8214      BUMAS   LDE     R1,@RR4
3A97  A210              CP      R1,R0
3A99  6B08              JR      EQ,YA
3A9B  A0E4              INCW    RR4
3A9D  A0E4              INCW    RR4
3A9F  A0E4              INCW    RR4
3AA1  8BF2              JR      BUMAS
3AA3  AF        YA      RET
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3AA4                    FORM
3AA4                ;*****************************************************************
3AA4                ;**                                                             **
3AA4                ;**                      B    X                                 **
3AA4                ;**                                                             **
3AA4                ;*****************************************************************
3AA4  0E            BX          INC         R0
3AA5  E330                      LD          R3,@R0
3AA7  A6E32C                    CP          R3,#COMA        ;SI NO SE REGISTRO NINGUNA DIRECCION
3AAA                                                        ;LLENA LA TABLA DE CEROS
3AAA  EB11                      JR          NE,BX3
3AAC  4C31          BBP         LD          R4,#31H
3AAE  5C00                      LD          R5,#00H         ;RR4 CONTIENE EL APUNTADOR A LA TABLA
3AB0  2C18          BX21        LD          R2,#24
3AB2  3C00          BX2         LD          R3,#00H         ;LLENA LA TABLA DE CEROS
3AB4  9234                      LDE         @RR4,R3
3AB6  A0E4                      INCW        RR4
3AB8  2AF8                      DJNZ        R2,BX2
3ABA  8D32AE                    JP          PREROOT
3ABD  CC1E          BA3         LD          R12,#30
3ABF  DC0A                      LD          R13,#10
3AC1  D632D0                    CALL        RECDIR
3AC4  6C30                      LD          R6,#30H
3AC6  7C10                      LD          R7,#10H
3AC8  C226                      LDC         R2,@RR6
3ACA  A0E6                      INCW        RR6
3ACC  C236                      LDC         R3,@RR6
3ACE  8C18                      LD          R8,#24
3AD0  8204          BX4         LDE         R0,@RR4
3AD2  A228                      CP          R2,R8
3AD4  ED3AE2                    JP          NE,BXMAS
3AD7  A0E4                      INCW        RR4
3AD9  8214                      LDE         R1,@RR4
3ADB  A231                      CP          R3,R1
3ADD  6D3AEC                    JP          EQ,YABX
3AE0  80E4                      DECW        RR4
3AE2  A0E4          BXMAS       INCW        RR4
3AE4  A0E4                      INCW        RR4
3AE6  A0E4                      INCW        RR4
3AE8  8AE6                      DJNZ        R8,BX4
3AEA  8B15                      JR          BXERR
3AEC  8294          YABX        LDE         R9,@RR4
3AEE  D292                      LDC         @RR2,R9
3AF0  80E4                      DECW        RR4
3AF2  80E4                      DECW        RR4
3AF4  80E4                      DECW        RR4
3AF6  8C00                      LD          R8,#00H
3AF8  9284                      LDE         @RR4,R8
3AFA  A0E4                      INCW        RR4
3AFC  9284                      LDE         @RR4,R8
3AFE  8D32AE                    JP          PREROOT
3B01  D63247        BXERR       CALL        CRLF
3B04  6C3F                      LD          R6,#BXERROR>8
3B06  7C6C                      LD          R7,#BXERROR\256
3B08  D63287                    CALL        ENVSINT
3B0B  D63247                    CALL        CRLF
3B0E  8D32AE                    JP          PREROOT
```

```
3B11                    FORM
3B11            ;*****************************************************************
3B11            ;***             RUTINA DE ATENCION A IRQ0                     ***
3B11            ;*****************************************************************
3B11  D63B29    GOTOAT0     CALL    ATRACE
3B14  8D3C5F                JP      RESPALDAU
3B17  D63B4F    GOBPOAT0    CALL    ATBP
3B1A  8D3C5F                JP      RESPALDAU
3B1D  D63B29    CTOAT0      CALL    ATRACE
3B20  8D3BEF                JP      DIRCALL
3B23  D63B4F    CBPOAT0     CALL    ATBP
3B26  8D3BEF                JP      DIRCALL
3B29            ;*****************************************************************
3B29  0C45      ATRACE      LD      R0,#45H
3B2B  ECF0                  LD      R14,#0F0H
3B2D  D20E                  LDC     @RR14,R0
3B2F  D63B02                CALL    IRQSALVA
3B32  EC30                  LD      R14,#30H
3B34  FC90                  LD      R15,#90H        ; TRAE EL CONTADOR DE TRACE Y LO DECREMENTA
3B36  820E                  LDE     R0,@RR14
3B38  00E0                  DEC     R0
3B3A  6D3B40                JP      Z,APAGAT
3B3D  920E                  LDE     @RR14,R0
3B3F  AF                    RET
3B40  56FCFE    APAGAT      AND     FLAGS,#11111110B
3B43  920E                  LDE     @RR14,R0
3B45  0C01                  LD      R0,#01H         ;  SI EL CONTADOR ES CERO APAGA BANDERA
3B47  ECF0                  LD      R14,#0F8H       ;  DE TRACE Y SE PONE EN MODO 1 DEL PUERTO
3B49  D20E                  LDC     @RR14,R0        ;  SERIE APAGA EL BIT DE IRQ0 DE IMR
3B4B  56F8FE                AND     IMR,#11111110B
3B4E  AF                    RET
3B4F            ;*****************************************************************
3B4F  0C85      ATBP        LD      R0,#85H
3B51  ECF0                  LD      R14,#0F0H
3B53  D20E                  LDC     @RR14,R0
3B55  D63B02                CALL    IRQSALVA
3B58  EC30                  LD      R14,#30H        ; AL OCURRIR UN BP ES NECESARIO AJUSTAR
3B5A  FC54                  LD      R15,#54H        ; LA DIRECCION DE RETORNO YA QUE ESTA ES
3B5C  82DE                  LDE     R13,@RR14       ; GUARDADA COMO DIR BP+1
3B5E  A0EE                  INCW    RR14
3B60  82CE                  LDE     R12,@RR14
3B62  80EC                  DECW    RR12
3B64  92CE                  LDE     @RR14,R12
3B66  80EE                  DECW    RR14
3B68  92DE                  LDE     @RR14,R13
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3B6A                    FORM
3B6A  EC31              LD      R14,#31H        ; \
3B6C  FC00              LD      R15,#00H        ; :
3B6E  0C08              LD      R0,#08H         ; :
3B70  70E0      OAT02   PUSH    R0              ; :
3B72  820E              LDE     R0,@RR14        ; :
3B74  A20C              CP      R0,R12          ; :  DETERMINA SI ESTA DIRECCION CONTENIA
3B76  EB0A              JR      NE,OATO1        ; :  UN BP VERDADERO COMPARANDO CON LA
3B78  A0EE              INCW    RR14            ; :  TABLA DE BP
3B7A  820E              LDE     R0,@RR14        ; :
3B7C  A20D              CP      R0,R13          ; :
3B7E  6B34              JR      EQ,OATDES       ; :
3B80  80EE              DECW    RR14            ; :
3B82  A0EE      OATO1   INCW    RR14            ; :
3B84  A0EE              INCW    RR14            ; :
3B86  A0EE              INCW    RR14            ; :
3B88  50E0              POP     R0              ; :
3B8A  0AE4              DJNZ    R0,OAT02        ; /
3B8C  50EE              POP     R14             ; \
3B8E  50EE              POP     R14             ; :
3B90  50EE              POP     R14             ; :
3B92  50EE              POP     R14             ; :
3B94  F817              LD      R15,17H         ; :
3B96  56EF18            AND     R15,#00011000B  ; :  DE NO ENCONTRAR ESTA DIRECCION
3B99  6B0A              JR      Z,SINDM         ; :  EN LA TABLA REGRESA AL PROGRAMA
3B9B  76EF08            TM      R15,#00001000B  ; :  DE PRUEBA
3B9E  6B0B              JR      Z,CONDM         ; :
3BA0  76EF10            TM      R15,#00010000B  ; :
3BA3  6B06              JR      Z,CONDM         ; :
3BA5  CCF0      SINDM   LD      R12,#0F0H       ; :
3BA7  ECA5              LD      R14,#0A5H       ; :
3BA9  D2EC              LDC     @RR12,R14       ; :
3BAB  E418F8    CONDM   LD      P01M,18H        ; :
3BAE  E417F7            LD      P3M,17H         ; :
3BB1  3D006E            JP      COMANDO         ; :
3BB4                                            ; /
3BB4  A0EE      OATDES  INCW    RR14            ; \
3BB6  820E              LDE     R0,@RR14        ; :
3BB8  A6EC10            CP      R12,#10H        ; :
3BBB  AD3BC5            JP      GT,CAMBDAT      ; :
3BBE  06EC30            ADD     R12,#30H        ; :
3BC1  920C              LDE     @RR12,R0        ; :
3BC3  8802              JR      ELIMINA         ; :  ELIMINA EL BP DE LA TABLA DE BP
3BC5                                            ; :  Y LA DIPECCION PARTICULAR
3BC5  D20C      CAMBDAT LDC     @RR12,RC        ; :
3BC7  CC00      ELIMINA LD      R12,#00H        ; :
3BC9  0C03              LD      R0,#03H         ; :
3BCB  92CE      QUITA   LDE     @RR14,R12       ; :
3BCD  80EE              DECW    RR14            ; :
3BCF  0AFA              DJNZ    R0,QUITA        ; /
3BD1  AF                RET
```

```
3BD2                    FORM
3BD2            ;***************************************************************
3BD2  0C0B      IRQSALVA  LD      R0,#0BH       ; \
3BD4  CC30                LD      R12,#30H      ;  ;
3BD6  DC5D                LD      R13,#5DH      ;  ;
3BD8  50EF      NECESARIOS POP    R15           ;  ;
3BDA  92FC                LDE     @RR12,R15     ;  ;
3BDC  80EC                DECW    RR12          ;  ; RECUPERA LOS REGISTROS QUE ESTABAN
3BDE  0AF8                DJNZ    R0,NECESARIOS ;  ; EN EL STACK Y LOS GUARDA EN MEMORIA
3BE0  0C0B                LD      R0,#0BH       ;  ; DE DATOS A PARTIR DE LA DIRECCION
3BE2  C0FE                LD      R12,SPH       ;  ; 3054 HASTA LA 305D
3BE4  D8FF                LD      R13,SPL       ;  ;
3BE6  80EC      CORRIGE   DECW    RR12          ;  ;
3BE8  0AFC                DJNZ    R0,CORRIGE    ;  ;
3BEA  C9FE                LD      SPH,R12       ;  ;
3BEC  D9FF                LD      SPL,R13       ;  ;
3BEE  AF                  RET                   ;  /
3BEF            ;***************************************************************
3BEF  D63C20    DIRCALL   CALL    DDU
3BF2  3C10                LD      R3,#10H
3BF4  0C00                LD      R0,#00H               ;  CONTADOR DE REGISTROS R3 Y R0 APUNTADOR A REGISTROS
3BF6  D63247              CALL    CRLF
3BF9  7C03                LD      R7,#3CH               ; \
3BFB  9C00                LD      R9,#00H               ;  ;
3BFD  D63C0E              CALL    MUESTRAME             ;  ;  DESPLIEGA EL CONTENIDO DE LOS REGISTROS
3C00  3C10                LD      R3,#10H               ;  ;  DEL ARCHIVO INTERNO
3C02  0CF0                LD      R0,#0F0H              ;  ;
3C04  7C01                LD      R7,#01H               ;  ;
3C06  9CF0                LD      R9,#0F0H              ;  ;
3C08  D63C0E              CALL    MUESTRAME             ;  /
3C0B  8D32AE              JP      PREPROT
3C0E  2819      MUESTRAME LD      R2,#19H
3C10  D63280              CALL    DESP
3C13  D632C9              CALL    ESPACIO
3C16  D632C9              CALL    ESPACIO
3C19  E320      Z2        LD      R3,@R0
3C1B  D63280              CALL    DESP
3C1E  D632C9              CALL    ESPACIO
3C21  0E                  INC     R0            ; SIGUE REGISTRO
3C22  9E                  INC     R9
3C23  0AF4                DJNZ    R0,Z2         ; SIGUE DESPLEGANDO LOS REGISTROS DEL MISMO GRUPO
3C25  3C10                LD      R3,#10H       ; REINICIA CONTADOR
3C27  D63247              CALL    CRLF
3C2A  7AE2                DJNZ    R7,MUESTRAME
3C2C  AF                  RET
3C2D            ;***************************************************************
3C2D  2C01      DDU       LD      R2,#01H       ; \
3C2F  D63242              CALL    MODOS         ;  ;  CAMBIA MODO : DE PUERTO SERIE
3C32  D63136              CALL    XS            ;  ;  DETERMINA VELOCIDAD DE TRANSMISION
3C35  4C30                LD      R4,#30H       ;  ;
3C37  5C54                LD      R5,#54H       ;  /
3C39  8C28                LD      R8,#40
3C3B  D63247              CALL    CRLF
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3C3E                    FORM
3C3E  D632C9    FORMAT  CALL    ESPACIO         ; \
3C41  8AFB              DJNZ    R8,FORMAT       ;  :
3C43  6C3F              LD      R6,@DADIR)8     ;  :
3C45  7CAA              LD      R7,@DADIR\256   ;  :   DESPLIEGA LA DIRECCION DONDE
3C47  D63287            CALL    ENVSINT         ;  :   OCURRIO LA INTERRUPCION
3C4A  8234              LDE     R3,@RR4         ;  :
3C4C  A0E4              IHCW    RR4             ;  :
3C4E  8224              LDE     R2,@RR4         ;  :
3C50  D63230            CALL    DESP            ;  :
3C53  2813              LD      R2,13H          ;  :
3C55  D63230            CALL    DESP            ;  :
3C58  D63247            CALL    CRLF            ;  :
3C5B  D63247            CALL    CRLF            ;  /
3C5E  AF                RET
3C5F            ;***************************************************************
3C5F  EC09      RESPALDAU LD    R14,#09H        ; \
3C61  CC45              LD      R12,#45H        ;  :
3C63  CC11              LD      R13,#11H        ;  : RESPALDA LO REGISTROS DE CONTROL
3C65  0C11              LD      R0,#11H         ;  : ( SE ASUME QUE ESTOS ESTAN EN EL GRUPO
3C67  930C      PASALOS LDEI    @RR12,@R0       ;  : 10 ) HACIA EL ARCHIVO EXTERNO
3C69  EAFC              DJNZ    R14,PASALOS     ;  :
3C6B  0CFA              LD      R0,#0FAH        ;  :
3C6D  930C              LDEI    @RR12,@R0       ;  :
3C6F  930C              LDEI    @RR12,@R0       ;  /
3C71  DC1F              LD      R13,#1FH        ; \
3C73  2C03              LD      R2,#03H         ;  :   RESPLADA LOS REGISTROS SPH, SPL Y STATUS
3C75  50E1      SPAFLAG POP     R1              ;  :   QUE SE ENCUENTRAN EN EL STACK
3C77  921C              LDE     @RR12,R1        ;  :
3C79  80EC              DECW    RR12            ;  :
3C7B  2AF8              DJNZ    R2,SPAFLAG      ;  /
3C7D  EC30              LD      R14,#30H        ; \
3C7F  FC53              LD      R15,#53H        ;  :
3C81  821E              LDE     R1,@RR14        ;  :
3C83  921C              LDE     @RR12,R1        ;  :
3C85                                            ;  :   PASA EL RESTO DE LOS REGISTROS AL ARCHIVO
3C85  DC00              LD      R13,#00H        ;  :   ALTERNO
3C87  0C00              LD      R0,#00H         ;  :
3C89  1C6F              LD      R1,#6FH         ;  :
3C8B  930C      LOSDEMAS LDEI   @RR12,@R0       ;  :
3C8D  A6E010            CP      R0,#10H         ;  :
3C90  ED3C97            JP      NE,SINPRO       ;  :
3C93  0C20              LD      R0,#20H         ;  :
3C95  DC20              LD      R13,#20H        ;  :
3C97  1AF2      SINPRO  DJNZ    R1,LOSDEMAS     ;  /
3C99  CC45              LD      R12,#45H
3C9B  0C02              LD      R13,#02H
3C9D  EC30              LD      R14,#30H
3C9F  FC57              LD      R15,#57H
3CA1  821E              LDE     R1,@RR14
3CA3  921C              LDE     @RR12,R1
```

```
3CA5  D63C2D      DIRGO     CALL      DDU         ;   DESPLIEGA DIRECCION DE INTERRUPCION
3CA8  4C45                  LD        R4,#45H     ; \
3CAA  5C00                  LD        R5,#00H     ;  :
3CAC  7C08                  LD        R7,#08H     ;  :
3CAE  9C10                  LD        R8,#10H     ;  :
3CB0  9C00                  LD        R9,#00H     ;  :
3CB2  D63CC1                CALL      MUSTRG      ;  :
3CB5  7C01                  LD        R7,#01H     ;  :
3CB7  9CF0                  LD        R9,#0F0H    ;  :
3CB9  5C10                  LD        R5,#10H     ;  :
3CBB  D63CC1                CALL      MUSTRG      ;  :
3CBE  3D32AE                JP        PREROGT     ;  :  DESPLIEGA EL ARCHIVO DE REGISTROS ALTERNO
3CC1                                                ;  :  TAL COMO QUEDARON ANTES DE LA
3CC1  2819        MUSTRG    LD        R2,19H      ;  :  INTERRUPCION
3CC3  D63280                CALL      DESP        ;  :
3CC6  D632C9                CALL      ESPACIO     ;  :
3CC9  D632C9                CALL      ESPACIO     ;  :
3CCC  8224        OTREG     LDE       R2,@RR4     ;  :
3CCE  D63280                CALL      DESP        ;  :
3CD1  D632C9                CALL      ESPACIO     ;  :
3CD4  A0E4                  INCW      RR4         ;  :
3CD6  9E                    INC       R9          ;  :
3CD7  8AF3                  DJNZ      R8,OTREG    ;  :
3CD9  9C10                  LD        R8,#10H     ;  :
3CDB  D63247                CALL      CRLF        ;  :
3CDE  7AE1                  DJNZ      R7,MUSTRG   ;  :
3CE0  AF                    RET                   ; /
```

```
SIS-Z8 SISTEMA DE DESARROLLO
3CE1                    FORM
3CE1                ;
3CE1                ;*************************************************************
3CE1                ;**              M   E   N   S   A   J   E   S              **
3CE1                ;*************************************************************
3CE1                ;
3CE1                ;
3CE1                ;
3CE1                ;
3CE1                ;
3CE1  444DFE35      TABL        DB          'DM',FI,DM>8,DM\256,'PM',FI,PM>8,PM\256
      F9504DFE
      37CC
3CEB  54FE37A1                  DB          'T',FI,T>8,T\256,'DR',FI
      4452FE
3CF2  349E4D41                  DB          DR>8,DR\256,'MA',FI,MA>8,MA\256,'AT',FI,AT>8
      FE356941
      54FE37
3CFD  BB534DFE                  DB          AT\256,'SM',FI,SM>8,SM\256,'MR',FI,MR>8,MR\256
      344B4D52
      FE358A
3D08  4BFE3977                  DB          'K',FI,K>8,K\256,'DB',FI,DB>8,DB\256
      4442FE39
      F9
3D11  4258FE3A                  DB          'BX',FI,BX>8,BX\256,'MU',FI,MU>8,MU\256,'BP',FI,BP>8
      A44D56FE
      38834250
      FE3A
3D1F  3554524D                  DB          BP\256,'TRM',FI,TRM>8,TRM\256
      FE3388
3D26  4D5354FE                  DB          'MST',FI,MST>8,MST\256
      3432
3D2C  474FFE36                  DB          'GO',FI,GO>8,GO\256,'CALL',FI,CALL>8,CALL\256,FT
      A243414C
      4CFE3761
      FD
3D39                            ORG         3DF3H
3DF3  000D0A2A      PROMPT      DB          NUL,CR,LF,'*',FF
      FF
3DF8  00082008      DELETE      DB          NUL,BS,ESP,BS,FF
      FF
3DFD  000D0A44      MESS        DB          NUL,CR,LF,'DATO NO NUMERICO',CR,LF,'*',FF
      41544F20
      4E4F204E
      554D4552
      49434F0D
      0A2AFF
3E14  10003EBA      TAPM        DB          10H,00H,SECUNO>8,SECUNO\256,PRAG>8,PRAG\256,PROG>8,PROG\256
      33483345

3E1C  50003ECE                  DB          50H,00H,SECDOS>8,SECDOS\256,PRAG>8,PRAG\256,PROG>8,PROG\256
      33483345

3E24  10003EE2                  DB          10H,00H,SECTRES>8,SECTRES\256,DRATA>8,DRATA\256,DATA>8,DATA\256
      33483342

3E2C  30003EF6      TAP1        DB          30H,00H,SECCUAT>8,SECCUAT\256,DRATA>8,DRATA\256,DATA>8,DATA\256
      33483342
```

```
3E34 000D0A44   LET     DB      NUL,CR,LF,'DIRECCION INVALIDA',CR,FF
     49524543
     43494F4E
     20494E56
     414C4944
     410DFF
3E4B 000D0A52   SOLOESC DB      NUL,CR,LF,'REGISTRO DE SOLO ESCRITURA',CR,LF,FF
     45474953
     54524F20
     44452053
     4F4C4F20
     45534352
     49545552
     410D0AFF

3E6B 000D0A4E   REGINV  DB      NUL,CR,LF,'NUMERO DE REGISTRO INVALIDO',CR,LF,FF
     55404552
     4F204445
     20524547
     49535452
     4F20494E
     56414C49
     444F0D0A
     FF
3E8C 000D0A53   MENACC  DB      NUL,CR,LF,'SIS Z8 LISTO PARA RECIBIR INSTRUCCIONES',CR,LF,'*',FF
     4953205A
     38204C49
     53544F20
     50415241
     20524543
     49424952
     20494E53
     54525543
     43494F4E
     45530D0A
     2AFF
3EBA 000D0A53   SECUNO  DB      NUL,CR,LF,'SECTOR # 1000H P',FF
     4543544F
     52202320
     31303030
     482050FF

3ECE 000D0A53   SECDOS  DB      NUL,CR,LF,'SECTOR # 5000H P',FF
     4543544F
     52202320
     35303030
     482050FF

3EE2 000D0A53   SECTRES DB      NUL,CR,LF,'SECTOR # 1000H D',FF
     4543544F
     52202320
     31303030
     482044FF
```

```
3EF6  000D0A53  SECCUAT   DB        NUL,CR,LF,'SECTOR # 3080H D',FF
      4543544F
      52202320


      33303830
      482044FF

3F0A  000D0A45  MENESIN   DB        NUL,CR,LF,'ERROR DE SINTAXIS',CR,LF,'*',FF
      52524F52
      20444520
      53494E54
      41584953
      0D0A2AFF

3F22  000D0A53  SIGM      DB        NUL,CR,LF,'SIGUIENTE LOCALIDAD =',CR,LF,FF
      49475549
      454E5445
      204C4F43
      414C4944
      4144203D
      0D0AFF
3F3D  000D0A20  OPINVA    DB        NUL,CR,LF,ESP,'OPCION INVALIDA DE MOVIMIENTO DE MEMORIA'
      4F504349
      4F4E2049
      4E56414C
      49444120
      4445204D
      4F56494D
      49454E54
      4F204445
      20404540
      4F524941

3F69  0D0AFF              DB        CR,LF,FF
3F6C  00455252  BXEROR    DB        NUL,'ERROR: NO EXISTE UN BREAK POINT EN ESA LOCALIDAD',FF
      4F523A20
      4E4F2045
      58495354
      4520554E
      20425245
      41482050
      4F494E54
      20454E20
      45534120
      4C4F4341
      4C494441
      44FF
3F9E  000D0A45  TRERR     DB        NUL,CR,LF,'ERRORES ',FF
      52524F52
      455320FF

3FAA  00444952  DADIR     DB        NUL,'DIRECCION = ',FF
      45434349
      4F4E203D
      20FF
```

```
3FB8  0D0D0A50   VDE        DB         NUL,CR,LF,'PRUEBA DE VERIFICACION DESPUES DE ESCRITURA',FF
      52554542
      41204445
      20564552
      49464943
      4143494F
      4E204445
      53505545
      53204445
      20455343
      52495455
      5241FF
3FE7  0D0D0A50   LDM        DB         NUL,CR,LF,'PRUEBA DE LECTURA',CR,LF,FF
      52554542
      41204445
      204C4543
      54555241
      0D0AFF
0000 ERRORS
```

SIS-Z8 SISTEMA DE DESARROLLO

SYMBOL TABLE

ACOMODA	31A0	AGUANTA	3127	ALTERA	36DB	AP	0012
APAGAT	3B40	ASCHEX	310B	AT	37BB	ATBP	384F
ATENT1	002C	ATENT2	0032	ATENT3	0038	ATENT4	003E
ATENT5	0044	ATRACE	3B29	BBP	3AAC	BIEN	3823
BP	3A35	BS	0008	BUFF	30A4	BUMAS	3A95
BUNO	3A43	BUSCAFI	30F1	BUSCAINT	325E	BUSVAC	3A93
BX	3AA4	BX2	3AB2	BX21	3AB0	BX3	3ABD
BX4	3AD0	BXEROR	3F6C	BXERR	3B01	BXMAS	3AE2
BYE	31BA	C1D4KB	37DC	C4KB	383B	CALL	3761
CAM2	3063	CAM3	333D	CAM4	309A	CAMBDAT	38C5
CAMUNO	305D	CAREG	383F	CBPOATO	3B23	CCOMA	32EA
CDEME	399B	CERO	3053	CIGUE	3214	CINC	3989
CLLENAB	39D3	COMA	002C	COMOSIEM	3A75	COMP	364D
COMPC	30E0	COMP1	30E8	[illegible]	[illegible]	[illegible]	[illegible]
CONLEC	380F	CONRET	37DE	CONT	3110	CONTAR	0013
CONT1	3221	CONTIN1	31E0	CONTIN2	31ED	CONTMR	35A6
CORRIGE	3BE6	CR	000D	CRDEL	31AE	CRLF	3247
CTOATO	381D	CTRLF	0006	CUATRO	30D9	CUNKB	37DA
DADIR	3FAA	DAT	3366	DATA	3342	DATDAT	38D2
DATPROG	38E8	DB	39F9	DBDEC	3A19	DBMAX	39FF
DBNDES	3A1E	DBSIES	3A11	DDCP	33D7	DDU	3C2D
DEC	3319	DEL	007F	DELETE	30F8	DESDIR	3610
DESDIR1	3619	DESGRUPO	34CA	DESP	3293	DESPL2	3627
DESPL3	363E	DESPL4	3648	DIR1	32F1	DIRECCION	3375
DM	35F9	DMHAB	3754	DOS	3073	DOSBY	3A26
DR	349E	DRATA	3348	DTRCALL	38EF	DTRGO	3CA5
DUNO	3684	ELIMINA	3BC7	ENTRADA	005D	ENVASC	3569
ENVSININT	328A	ENVSINT	3297	EOF	0004	ERR	331D
ERR1	3324	ESCL	3447	ESCONT	36D6	ESP	002C
ESP0	34E0	ESPACIO	3209	ESPACIO2	34E9	ESPDAT	3175
ESPERA	325A	ESTONO	3732	ESTOSI	3736	EYO	365E
FALLA	3849	FF	00FF	FG	39E3	FI	00FE
FIN	3681	FINGRUPO	3568	FINMV	3906	FINOTRAS	3976
FINPOD	3374	FINTRAS	3955	FINUMOK	350C	FINUM	3557
FLAGS	00FC	FORMAT	3C3E	FT	00FD	FUETRACE	3779
G027	372A	GDA	3697	GDAREG	3390	GENERAL	004A
GHUY	3379	GO	36A2	GOBPOATO	3B17	GOTOATO	3B11
GRUP1	355A	GRUPO	3558	GUION	3A77	HACHE	0049
HERROR	31F2	HEXASC	31FC	IGUAL	003D	INI	32E2
INICIO	3201	INIREG	348F	INIT	3005	INTERNA	377E
IRBSALVA	3BD2	K	3977	LC1D4	3814	LDM	3FE7
LECR	3816	LET	3E34	LETRAS	321E	LF	000A
LIMPIA	32D8	LIMPIAB	3375	LINEA	3570	LINEAOK	356B
LOSDEMAS	3C88	LSIEM	3818	LUNK	3812	MA	3569
MANDAR	346B	MEN1	3921	MENACC	3E8C	MENESIN	3F0A

SIS-Z8 SISTEMA DE DESARROLLO

MESS	3DFD	MODOS	3242	MON	3387	MONSUP	3000
MR	358A	MRUSUA	35D6	MST	3432	MUE	3763
MUEI	3942	MUESTRAME	3C0E	MUEVE	0063	MUSTRG	3CC1
MV	3883	NDAT	317D	NECESARIOS	3B03	NOES2	3741
NOINT	382B	NORETRIES	37F7	NORMAL	36BF	NOTRAS	3956
NOWO	35A9	NUL	0000	NUMDOS	34F1	NUMOK	34F3
NUMUNO	34E6	OAT02	3B70	OATDES	3BB4	OATOI	3B82
OPINUA	3F3D	OTRA	322B	OTREG	3CCC	PARAX	32A3
PARIDAD	3154	PASA	37EF	PASALOS	3C67	PBAJA	33BB
PDEL	30C1	PM	37CC	POD	334E	PRAG	3348
PRCF	33F4	PRENT	375D	PREPRO	30B1	PREREG	3298
PREROOT	32AE	PRINC	37D7	PROC	3935	PROG	3345
PROGDAT	38F3	PROGPROG	38DD	PROMPT	3DF3	PRUEWO	34A6
PSC	398E	QUITA	38CB	RECDIR	3200	RECEPCION	3053
RECIBE	3393	RECREG	3400	REG	001A	REGI	001E
REG2	001F	REGINU	3E68	REGZ8	3549	RESPALDAU	3C5F
REST	3533	RESTI	35CC	RETA	3955	RETRIES	3965
ROOT	3448	RXX	35DB	SAL	38BA	SALIDA	327C
SALPRO	0073	SALTO	366A	SALVAC	332E	SECCUAT	3EF6
SECDOS	3ECE	SECTRES	3EE2	SECUNO	3E8A	SIEMPRE	37E3
SIG	362F	SIGI	3655	SIG2	3672	SIGM	3F22
SINDM	3BA5	SINPRO	3C97	SINTAX	3107	SM	3448
SOLDESC	3E4B	SPAFLAG	3C75	STAND	3437	STC	39EE
T	37A1	TABL	3CE1	TAMCAK	3769	TAMDOK	36C6
TAPI	3E2C	TAPM	3E14	TDUNO	312B	TDOOS	3C0D
TRANSFER	36B5	TRAS	399E	TRAS1	3956	TRASNO	3903
TRASSI	3908	TRERR	3F9E	TRES	3083	TRFR	33C3
TRM	3380	UNO	3063	UNO1	31DD	UNO2	31EA
UNREG	350D	USUARIO	3403	VAMONOS	006E	VDE	3FB8
VEAINC	349A	VETRAS	38FB	XCALC	314E	XDECCD	313A
XEND	316E	XMASUEL	3146	XOFF	0013	XON	3277
XS	3136	XSON	3168	YA	3AA3	YABX	3AEC
Z2	3C19	ZIGUE	329C				

CAPITULO VII

DISEÑO DEL SIS Z8

El objetivo de éste capitulo es presentar detalladamente los pasos del proceso de diseño, aplicados exclusivamente a proyectos de Ingeniería Electrónica e Ingeniería en Computación, enfocados a desarrollos microprocesador-firmware, hasta el punto de lograr un prototipo de laboratorio aplicable, tomando en cuenta a manera de ejemplo, la experiencia del SIS Z8.

Para muchos estudiantes de Ingeniería, la primera y última experiencia formal de diseño, es la que llevan a cabo en su trabajo de tesis. En el caso de la Facultad de Ingeniería, las tesis de este tipo son generalmente auspiciadas y en beneficio de la UNAM.

Esto no es extraño, ya que dentro de las paredes de la Universidad Nacional se concentra en un altísimo porcentaje la investigación y desarrollo en todas las áreas a nivel nacional. Tal vez por ésto sea, que en ella se han invertido fuertes sumas de dinero para la creación de la infraestructura necesaria para trabajos de investigación, diseño y desarrollo. Sin embargo, el estudiante de Ingeniería Electrónica y en Computación, con toda la disposición de crear y diseñar, al intentarlo por primera vez encuentra un obstáculo: no sabe como aplicar sus conocimientos, porque nunca cursó una materia especifica de diseño de proyectos electrónicos.

Es cierto que el diseño tiene mucho del arte en virtud del criterio y de la creatividad que requiere. Sin embargo, existen pasos fundamentales que conforman el proceso de diseño, cuya práctica adecuada facilita la creación, ahorra tiempo y permite obtener resultados más profesionales, que los que cualquier intento empirico podria lograr.

Es difícil explicar las estructuras mentales necesarias para que el individuo logre conjuntar los conocimientos adquiridos en

un todo, que resuelva de alguna manera un problema planteado, por ende, el seguir los pasos de diseño no garantizan que el practicante logre su objetivo, pero sí le permiten encauzar y organizar ideas ortodoxamente hacia el fin deseado.

Existe una amplia literatura sobre la ingeniería de diseño y en ella se esbozan los componentes del proceso de diseño en una forma general, aplicable a cualquier área ingenieril.

Al respecto, la mayoría de los autores coinciden en que el proceso de diseño consiste de los siguientes pasos:

1.- Formulación del problema.
2.- Análisis del problema.
3.- Búsqueda de alternativas.
4.- Evaluación de alternativas.
5.- Especificación de la solución preferida.
6.- Desarrollo de la solución.
7.- Prueba de los resultados.

Estos conceptos son una buena base para comenzar, pero la realidad es que el problema del diseño varía sustancialmente dependiendo del área específica a la cual se quiera aplicar el proceso de diseño, y el caso de los proyectos de ingeniería electrónica no son la excepción.

Así podríamos poner por ejemplo: una de las diferencias más notables que presentan los proyectos de ingeniería electrónica es que requieren la realización de un prototipo de laboratorio, como el caso del SIS Z8.

Volviendo un poco al SIS Z8, para realizar un prototipo de laboratorio se requiere de muchos Mapas de Karnaugh, del cálculo de resitencias de pullup y de un sin fin de elementos que sería largo de enumerar, y más aún, detallar.

Para lo anterior proponemos un procedimiento para la realización de proyectos electrónicos basados en microprocesadores el cual se muestra en el diagrama de la Figura (7.1) y se explica en la siguiente sección.

El diseño del SIS Z8 ha sido analizado a posteriori, basado en las reflexiones sobre los aciertos y errores que tuvimos a lo largo de esta experiencia.

Esperamos que el enfoque dado a este capítulo, pueda aprovecharse más que una bitácora de cálculos sobre un proyecto específico que probablemente no vuelva a repetirse en el futuro.

PROPUESTA DE PROCEDIMIENTO PARA PROYECTOS DE I.E.

En este procedimiento se supone la existencia de 3 participantes relacionados con el proceso:

1.- El proponente: es una persona o institución que demanda la solución de un problema. Frecuentemente, éste es también la fuente de financiamiento y su función es establecer las bases, costos, presupuestos y vigilar que el diseño sea encausado correctamente.

2.- El grupo de diseño: consiste en un grupo de ingenieros a los que se les encarga el diseño de la solución del problema y se encuentra regulado generalmente por el proponente, debiendo tomar en cuenta en su diseño al tercer participante : El usuario.

3.- El usuario es quien, finalmente, utilizará el resultado del diseño para la solución del problema.

En el procedimiento se distinguen 4 fases, que comienzan con el análisis del problema por parte del proponente y concluyen con la documentación y difusión del proyecto.

A lo largo del proyecto, el análisis de alternativas se presenta una y otra vez, ya que es inherente al proceso de diseño el encontrar varias soluciones a un mismo problema. En el análisis de las alternativas existen parámetros y restricciones que ayudan a determinar la solución óptima. Algunos de estos criterios son:

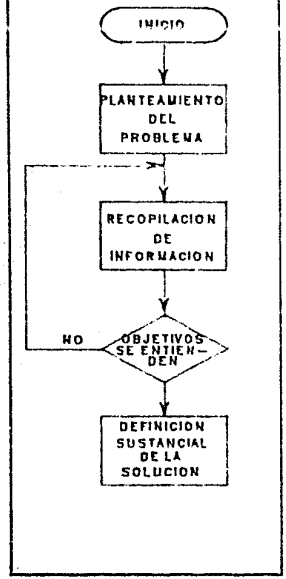
- El costo.
- La demanda del producto.
- El tiempo asignado al producto.
- La oferta en el mercado de componentes y elementos.
- El equipo disponible.

Los dos últimos en la práctica son de suma importancia para quién diseña en México. Esto es, donde la infraestructura es pobre. No es lo mismo querer diseñar cuando se dispone de un analizador lógico o de un analizador de espectros, que cuando solo se cuenta con un osciloscopio.

Asimismo, querer diseñar un sistema con circuitos integrados VLSI, resulta una pérdida de tiempo, ya que, por ejemplo, un 74134 es materialmente imposible adquirirlo en el mercado nacional.

Estos y otros aspectos, serán analizados a continuación en la descripción de las fases del procedimiento.

PROPUESTA Y DESCRIPCION PRELIMINAR



PLANTEAMIENTO DEL PROBLEMA

Definición del problema en forma objetiva. Primera evaluación de alternativas. El resultado de este análisis es una solución en forma de caja negra.

El proponente establece: se requiere un sistema de desarrollo para el MCSC Z8.

RECOPILACION DE INFORMACION

En esta etapa el grupo de diseño debe comprender el problema, la solución planteada en el análisis anterior y definir los objetivos.

El grupo de diseño comprende:
¿Que es el Z8?
¿Que es un sistema de desarrollo?
¿Que es lo que desea el proponente?
¿Que es lo que desearia el usuario?

DEFINICION DE LA SOLUCION

Se definen las características de la caja negra. Se puede hacer una evaluación del costo y el tiempo de desarrollo.

El grupo de diseño define:
Configuración del SIS Z8.
Comandos del SCAN Z8.
Elementos del SIS Z8: puertos, capacidad en RAM, etc.

FIGURA (7.2 a)

DISEÑO PRELIMINAR

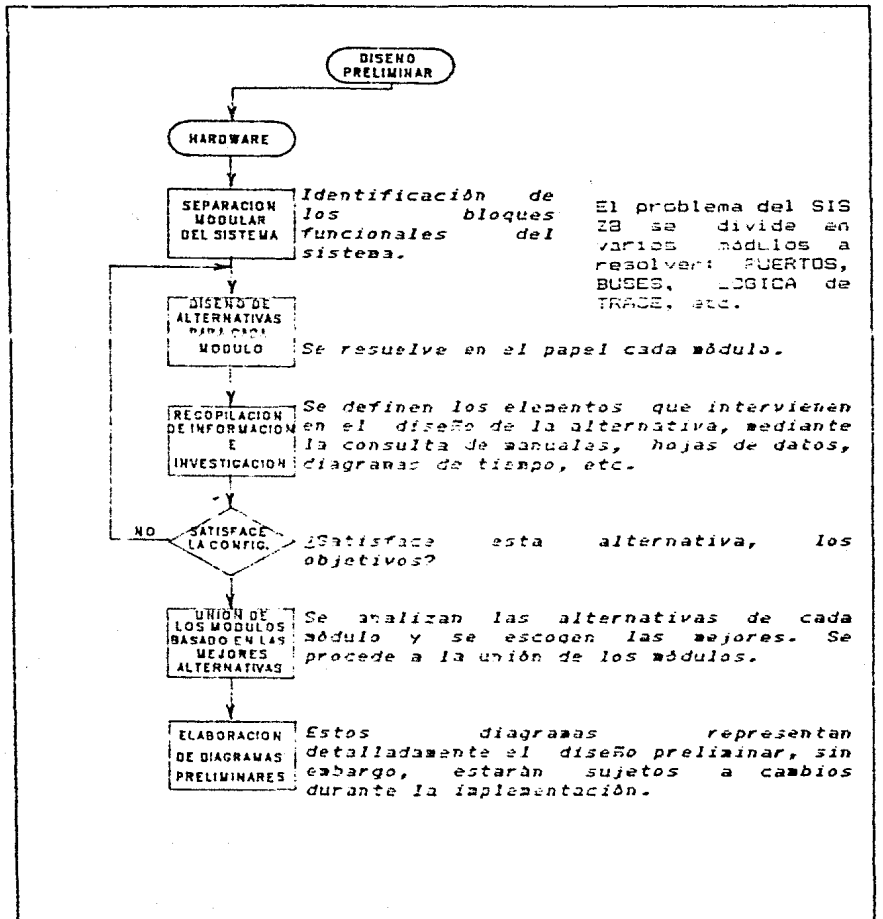
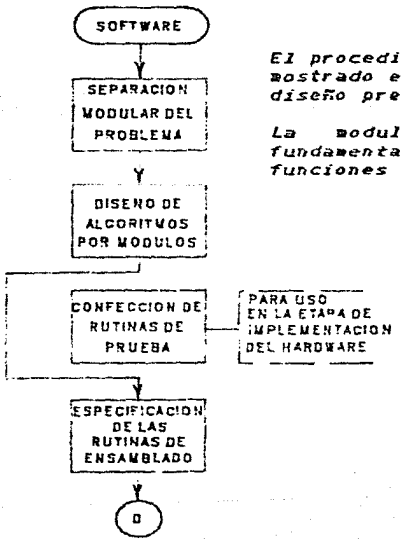


FIGURA (7.2 b)

DISEÑO PRELIMINAR (software)



El procedimiento a seguir es semejante al mostrado en la Figura (7.2 b), para el diseño preliminar de Hardware.

La modularidad en este caso es fundamental, para evitar duplicación de funciones y el agotamiento de la memoria.

FIGURA (7.2 c)

IMPLEMENTACION DEL PROTOTIPO Y DOCUMENTACION (a).

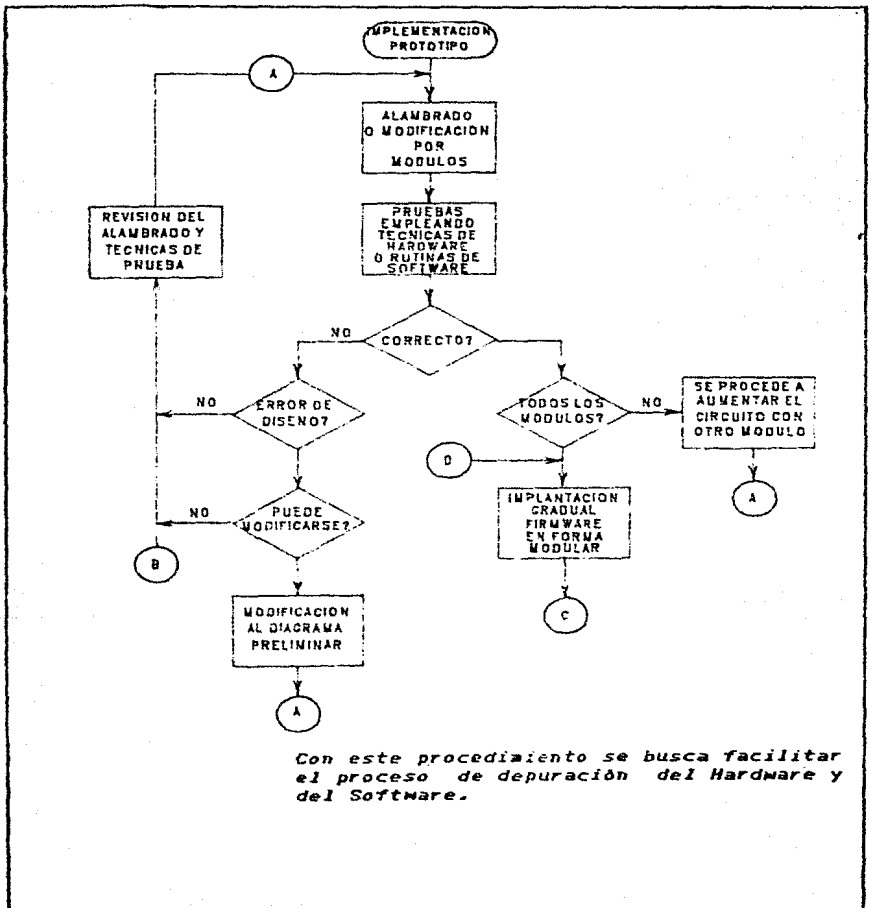


FIGURA (7.2 d)

IMPLEMENTACION DEL PROTOTIPO Y DOCUMENTACION (b).

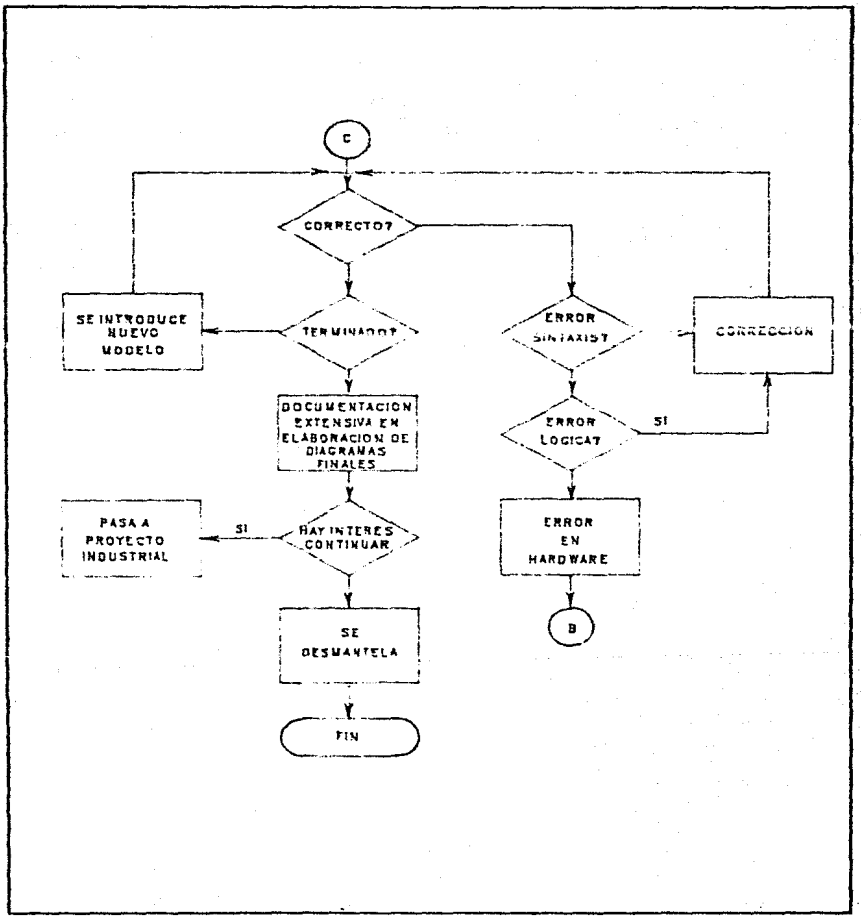


FIGURA (7.2 e)

1.- PROPUESTA Y DESCRIPCION PRELIMINAR

Esta fase comienza cuando el proponente identifica un problema o necesidad a satisfacer.

Hace dos años cuando estábamos en busca de un Seminario de Tesis, encontramos una compañía privada (IDET S.A.) dedicada, entre otras actividades al diseño de sistemas electrónicos enfocados a computación. En aquel entonces, IDET ya contaba con un circuito funcionando con el Z8 en vías de comercialización, además de otros proyectos en puerta, para lo cuál era necesario desarrollar herramientas y procedimientos más eficientes para diseñarlos.

El primer análisis de alternativas debe de hacerlo el proponente en esta etapa con el fin de determinar entre varias soluciones, en forma teórica, la más viable.

La identificación de esta necesidad fué el primer paso del proyecto y las alternativas variadas: Un emulador de Z8 en una PC; Comprar un sistema de desarrollo comercial; Crear un nuevo sistema de desarrollo, etc. Después de un análisis se llegó a la conclusión de crear este último, al cuál se le puso por nombre SIS Z8.

Se procede a la selección de un grupo de diseño y éste comienza su labor enterándose del problema y comprendiendo la idea del proponente. Para esta etapa el grupo de diseño debe acumular y revisar gran cantidad de información, así como consultar continuamente al proponente y al usuario final.

Al momento de formarse este grupo aún no conocíamos el microcomputador de un solo chip Z8, por lo que el trabajo comenzó con un acercamiento a este dispositivo mediante la recopilación y estudio proveniente de diversas fuentes: Manuales del Z8, revistas comerciales de computación, aplicaciones diversas del Z8, literatura sobre sistemas operativos y electrónica en general.

Al finalizar esta etapa el grupo de diseño debe de tener definidos los alcances y objetivos. Estos deben concordar con la idea del proponente y con lo que el usuario finalmente desea.

Los objetivos logrados por el grupo fueron un amplio conocimiento del Z8, sistemas de desarrollo y compartir la idea de nuestro proponente modificada y aumentada según las opiniones de posibles usuarios del sistema.

La etapa culminante de esta fase es una propuesta descriptiva por parte del grupo de diseño, donde se formalice la idea y se especifiquen los objetivos, elementos y características

de la caja negra, junto con una evaluación, tiempo de desarrollo e infraestructura necesaria.

La propuesta preliminar del SIS Z8 consistió en un documento muy breve, donde se explicaba la configuración del mismo (conexiones con terminal y computadora anfitriona) y se desarrollaban elementos tales como: 2 puertos serie, la capacidad de RAM en datos y en programa, los comandos deseables del SCAN Z8, etc. La propuesta original incluía un puerto paralelo para impresora, un grabador de EPROMS 2732 y una fuente de C.D. de +5, +12 y -12 V., las cuales aunque fueron diseñadas en el papel no fueron incluídas, por quedar fuera de presupuesto y tiempo. Por ésto, consideramos ampliamente recomendable agregar una evaluación de costos y tiempo, que en nuestro caso no fué realizado.

2.- DISEÑO PRELIMINAR.

Esta fase es la parte medular del trabajo de diseño, por eso debe de arrancar sobre bases muy sólidas en cuanto, a conocimiento de los objetivos y del problema, y ésto depende en gran medida de que la primera fase sea observada al detalle. Es natural que exista cierta desesperación en la primera fase porque durante ella el proyecto aparentemente no avanza. Sin embargo, ceder a esta presión hace de la segunda fase un fracaso rotundo. Esto mismo es cierto en la fase de implementación del prototipo, si es que la segunda no ha sido cabalmente terminada y revisada. Es por ésto que se recomienda que en cada fase, sobre todo en el diseño preliminar sea revisada de principio a fin y no dejar nada bajo la probabilidad de que puede llegar a funcionar.

Es característico de esta etapa la dificultad que representa el manejo de una gran cantidad de variables cuyo número esta íntimamente ligado al tamaño y complejidad del sistema.

Un método de diseño sistemático requiere desglosar al sistema en varios bloques cuya función pueda dicernirse de los restantes para después atacar, independientemente, cada uno hasta completar todos los módulos.

Asi pues, la primera etapa de esta fase consiste en identificar las funciones independientes que el sistema debe realizar y preparar un diagrama modular o de bloques.

Lo anterior es válido para ambas partes de un proyecto con microprocesadores: Hardware y Software. El principio del diseño debe ser un diagrama de bloques para cada parte, sin embargo, aunque ambas tienen la misma importancia el tratamiento debe ser distinto, por lo que a partir de este momento se abren 2 lineas de trabajo que, aunque deben complementarse en todo momento, no serán reunidas sino casi al final de la tercera fase.

En el SIS Z8 el hardware se dividió en los siguientes modulos:

- Z8612 y control de buses.
- Puerto serie.
- Decodificación.
- ROMS y RAMS.
- Puertos de control y lectura de opciones.
- Lógica de B P y Trace.

El diagrama de bloques ya fué estudiado en su oportunidad en el Capitulo 5.

En cuanto al SCAN Z8, se cometieron 2 errores: falta de modularidad y la creencia de que el desarrollo del Software no

puede empezar hasta no tener el Hardware siquiera en papel. El resultado de estos errores fué un desarrollo tardío de los programas, con la consiguiente extensión del tiempo previsto, además de la complejidad y duplicación de funciones, en algunas de las rutinas del sistema.

La programación debe comenzar en el momento en que se tenga a mano el diagrama de bloques, sin embargo, éstos suponen un dominio absoluto de las herramientas y capacidades del programa ensamblador a emplearse, debiendo quedar el programa perfectamente documentado y estructurado.

La modularidad en el SCAN Z8 es obra de la improvisación y dentro de esta modularidad se distinguen: un Programa Monitor, un Conjunto de Comandos y un Conjunto de Rutinas que soportan tanto al monitor como a los comandos.

Los programas del SCAN Z8 se fueron diseñando en el siguiente orden: Primero el monitor y a continuación los comandos. Las rutinas del sistema se fueron implementando conforme se requerían. Esto se debe a que teníamos una idea global de como debería trabajar el SCAN Z8 y una lista de comandos, pero nunca existió una lista de rutinas comunes. Si tuviéramos que repetir algún día este trabajo, definitivamente no seguiríamos el mismo camino. La experiencia nos ha enseñado que la forma óptima de diseñar un sistema operativo residente (como es el caso del SCAN Z8) debe comenzar por el monitor, seguido por las rutinas y finalmente los comandos. De esta manera cada nueva etapa de programación se apoya en los logros de la anterior.

El trabajo de programación de las rutinas, que finalmente quedarán implantadas en el firmware, puede ser interrumpido momentáneamente, a solicitud de quienes estén encargados en el diseño del Hardware, para confeccionar una serie de programas muy sencillos, cuyo objetivo es probar el funcionamiento correcto de determinado módulo. Otro beneficio de la realización de estos programas es que debido a su sencillez, los programadores pueden obtener rápidamente experiencia sobre el manejo eficaz y adecuado del microprocesador y el sistema en general.

Durante esta etapa se implementaron una serie de rutinas para probar puertos, memorias, lecturas de switches, etc. Algunas de ellas después de cumplir su función fueron desechadas, mientras que otras fueron modificadas y ampliadas para formar parte del firmware final, como es el caso de los comandos PM y MA.

En el diseño del Software, quizá los elementos con más peso son el ingenio y experiencia del Ingeniero en Computación en proyectos semejantes. Por otro lado el Hardware requiere de algo más de investigación, partiendo de diagramas modulares el diseñador debe producir dos, tres o más soluciones para cada

módulo y después decidir por la más adecuada según diversos criterios que dependen de los objetivos y alcances del proyecto.

Recordemos que este procedimiento culmina con un prototipo de laboratorio cuyo objetivo principal es completar un producto funcional que no necesariamente sea industrializable y menos aún comercializable. Esto último no debe decepcionar a nadie. Un producto profesional no se hace de la noche a la mañana y menos aún solamente con dinero, sino que además requiere de esfuerzo, ingenio, y conocimientos. Resultaría infantil suponer que al primer intento se pueda lograr un producto Industrial-Comercial (auque obviamente esto depende mucho de la complejidad del proyecto).

Al prototipo de laboratorio le debe seguir un nuevo proyecto para hacer posible su industrialización. Este segundo trabajo debe finalizar en un producto totalmente terminado capaz de ser introducido en el mercado. Más aún, el problema del diseño no termina ahí, puesto que todavía deben programarse revisiones periodicas para evitar su obsolescencia, y probablemente de ellas, se establezcan las bases para iniciar un nuevo proyecto a fin de obtener un nuevo producto o modelo mejorado.

De todo lo anterior se desprende que al diseñar y elegir de entre las alternativas la adecuada para cada módulo, los criterios que el diseñador debe observar no necesariamente son los costos o la cantidad mínima de circuitos, máxime si el proyecto tiene como objetivo final producir solamente un prototipo como en el caso del SIS Z8. Algunos de los criterios que se recomiendan seguir son los siguientes:

1.- El Hardware del prototipo debe ser sobrado para facilitar el trabajo de programación, el cual necesita más tiempo de desarrollo. Esto es, mientras más funciones del sistema puedan implementarse via Hardware, menos tiempo de programación será necesario.

2.- En la actualidad existen circuitos integrados VLSI que empacan una gran cantidad de funciones. Estos pueden resolver fácilmente muchos problemas, sin embargo, el mercado nacional de estos dispositivos no es vanguardista y el conseguir algunos C.I. en él, puede convertirse en una inútil pérdida de tiempo.

3.- El equipo de desarrollo disponible, debe inventariarse para tratar de ajustarse a él y evitar inversiones costosas en equipo que no sea absolutamente indispensable.

4.- Los circuitos impresos deben ser evitados, a menos que sean estrictamente necesarios.

5.- Si los 4 criterios anteriores han sido observados, entonces, en lo posible, debe buscarse optimizar los costos y la cantidad de circuitos.

Aprovechamos este momento para justificar algunos detalles sobre el Hardware del SIS Z8, que no fueron mencionados en el Capitulo 5 por quedar fuera de contexto.

En primer lugar, el SIS Z8 no tiene antecesor alguno. Es decir, es claro que existen Sistemas de desarrollo comerciales para varios microprocesadores, muchos de ellos autosuficientes, que consisten de un teclado, un juego de displays de 7 segmentos, una fuente de C. D., una tarjeta de lógica con el microprocesador correspondiente, etc. Sin embargo, dada la diferencia sustancial entre un microprocesador y un microcomputador de un solo chip, obligadamente los sistemas de desarrollo para uno y otro son enteramente distintos. El modo de operación del SIS Z8 (Terminal-SIS Z8-Computador Anfitrión) fué tomada de un Sistema de desarrollo creado por ZILOG, el cual trabaja con un microprocesador Z8000 conjuntamente con el Z8. Fuera de esto último, el SIS Z8 no fué creado apoyándose de alguna experiencia anterior. Es cierto que en algunos casos recurrimos a soluciones de problemas semejantes, por ejemplo, la forma en que se demultiplexa el Bus de Datos/Direcciones es prácticamente estandar y existen muchos sistemas comerciales que operan, por ejemplo, con el 8748 de INTEL, que emplean la misma configuración. A excepción de éste, el SIS Z8 fué diseñado empezando desde cero a partir de lo que conceptualmente se deseaba.

El resultado final es un dispositivo confiable que cumple con sus objetivos. Sin embargo, puede ser mejorado. Quien examine los diagramas con detenimiento, podrá encontrar formas para disminuir el número de elementos, a base de eliminar funciones innecesarias ó sustituir circuitos integrados. Algunos de estos errores, por llamarlos de alguna manera, son debidos a falta de experiencia, mientras que otros fueron cometidos a propósito a fin de cumplir con los criterios de diseño establecidos en párrafos anteriores. Los siguientes ejemplos son muestra de ello.

1.- Los puertos serie pueden operar en seis modos distintos, pero finalmente solo cuatro de ellos son utilizados en el SCAN Z8. Al momento de diseñarlos no sabíamos si 3 modos fueran necesarios. Actualmente consideramos un acierto incluirlos, puesto que tuvimos mayor libertad en la programación de las rutinas que necesitan la intervención de los puertos.

2.- La memoria de solo lectura adecuada para el monitor externo es una de 4Kx8 (Por ejemplo una 2732) en lugar de las 2 memorias de 2Kx8(2716) utilizadas. Esto se debe a que el decodificador principal, secciona el espacio direccionable en

bloques de 4KB. Además el tamaño del monitor externo es muy cercano a este valor. De igual manera, el monitor interno no ocupa ni una cuarta parte de la ROM 2716 empleada. La razón de lo anterior, es que solo se disponía de equipo para grabar EPROM del tipo 2716, y aunque pudo hacerse la inversión en un grabador para 2732, el primero tenía la ventaja de que su operación era ampliamente conocida por nosotros, aún antes de comenzar este trabajo.

3.- El detector de BP, está implementado con 4 compuertas AND de 3 entradas 74LS11, en lugar de una sola AND de 13 entradas 74LS134. La razón es que este último no se vende en México.

Sin duda el SIS Z8 puede aún ser mucho mejor, pero hacerlo implica una inversión muy superior a la ya hecha. El costo del SIS Z8, sin contar nuestro tiempo, consta del precio de sus componentes, incluyendo los cables y las tarjetas de experimentación sobre las cuales está armado. La inversión para el equipo necesario, ya había sido realizada para otros fines, y por lo tanto, algunos de ellos no se encontraban en buenas condiciones como se puede apreciar en las fotografías de esta sección. Incluso en ciertas sesiones de trabajo fué necesaria una labor previa de mantenimiento. Aún así, obsoleto o no, este equipo, gracias al pleno conocimiento que de él tuvimos, fué una estupenda herramienta.

Una mirada retrospectiva nos muestra que ni el SIS Z8, ni cualquier prototipo de laboratorio, justifica por sí solo la inversión de, por ejemplo, un grabador de EPROMS 2732.

Quizá la primera de nuestras conclusiones de esta experiencia, es que solo en caso de una producción industrial, es tecnológica y económicamente justificable, la inversión de fuertes sumas de dinero en equipo sofisticado o vanguardista.

Para lograr prototipos de laboratorio que solo sirven para presentar tesis profesionales, no es necesario un almacen de aparatos de laboratorio, que al final mueren en la obsolecencia de una vida útil subempleada.

Lo anterior es algo más serio de lo que parece y merece un tratamiento más profundo del que podemos dar aquí. Solo unas líneas más para concretar la idea: ¿Cuál es el recurso indispensable de una institución dedicada al diseño? Como siempre, la infraestructura material es necesaria, pero ésta debe ser adecuada a los alcances y objetivos de la institución (jamás lo inverso). Lo que resulta inevitable es el recurso humano; para ponerlo en forma clara: Con un equipo humano calificado, casi cualquier diseño puede hacerse, con o sin abundancia de infraestructura. Si el recurso humano es pobre, es inútil intentarlo, sea cual sea la magnitud del recurso monetario. Es claro que un ingeniero incapaz de aplicar sus conocimientos

teóricos, no concluirá nunca un proyecto en electrónica digital. Lo aprendido en la Universidad resulta clave en esta etapa, pero como hemos visto, con un amplio dominio de las materias universitarias pertinentes, es evidente que aún quedan muchos temas por resolverse en el amplio proceso del diseño; algunos de ellos requieren de tanto conocimiento que de ser omitido en la Universidad debe aprenderlo el ingeniero (si alguna vez tiene la oportunidad), en una costosa base de prueba y error, en su práctica profesional.

Al final de esta etapa el grupo de diseño tiene un conjunto de soluciones para cada módulo, informalmente presentadas en hojas sueltas de papel. Se procede a la unión de las mejores alternativas en un solo sistema al mismo tiempo que se realiza la primera revisión del funcionamiento del sistema, a base de pruebas de escritorio y comprobación de cálculos. Cuando se tiene plena confianza en el diseño preliminar, deben hacerse sus diagramas correspondientes. Estos diagramas serán muy útiles durante la siguiente fase y probablemente sobre ellos se harán modificaciones, correcciones y anotaciones, por lo cual se recomienda que éstos sean ampliamente detallados sin perder tiempo en su calidad artística.

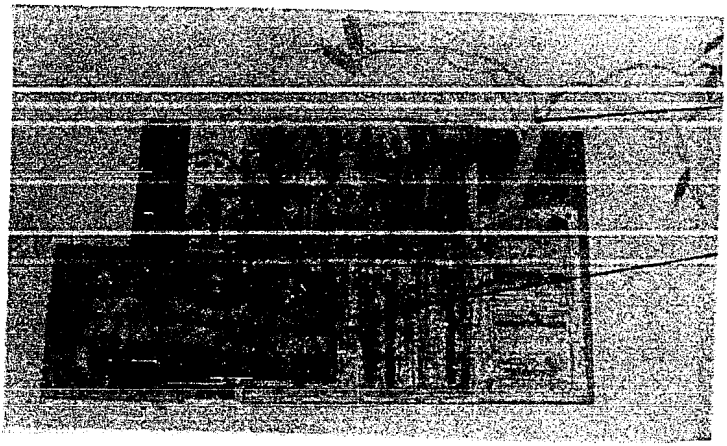
7.-IMPLEMENTACION DEL PROTOTIPO

Esta fase no consiste solamente de la realización del cableado que conforma el prototipo, sino que implica la primera depuración tanto del Hardware como del Software.

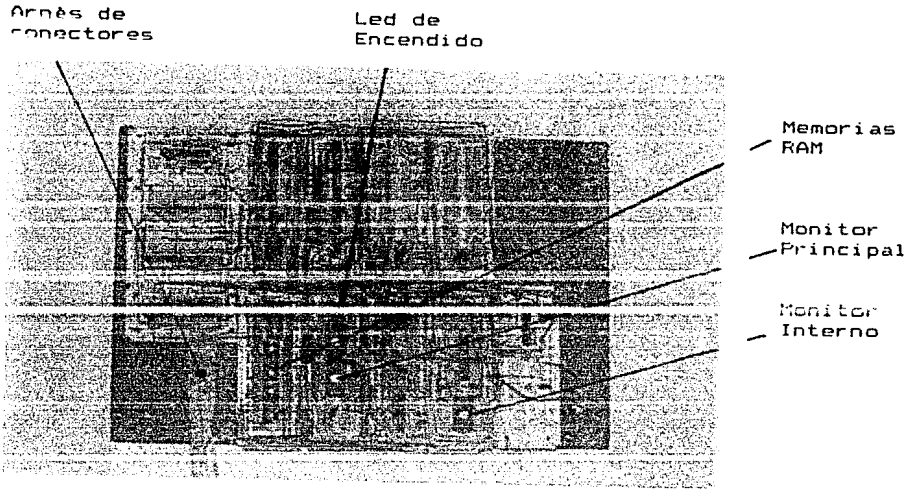
Los errores que en la segunda fase sean cometidos, se harán evidentes ahora. A éstos se sumarán los que ocurran durante el alambrado y los errores debido a la digitación, en la elaboración de los programas. Estos últimos serán abundantes, puesto que dependen de factores netamente humanos, tales como la habilidad manual. Cada error lleva implícito una pérdida de tiempo y a veces riesgos que pueden derivar en costosos daños físicos a algunos elementos del Hardware o del equipo de soporte. Para disminuir este problema debe observarse sistemáticamente una estricta limpieza del alambrado y rigurosas revisiones de los programas a fin de eliminar errores lógicos y de digitación, previos al ensamblado y grabado de EPROMS.

Las fotografías de esta sección son algunos ejemplos prácticos que fueron muy útiles en la implementación del SIS Z8.

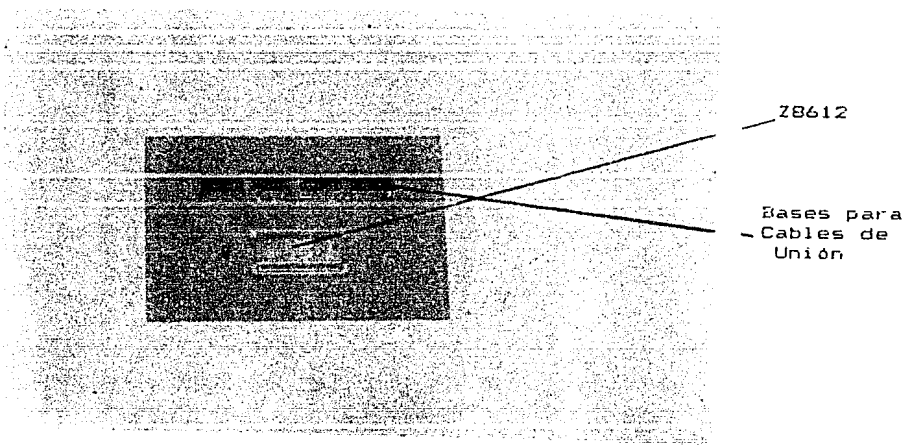
Botón de
Reset



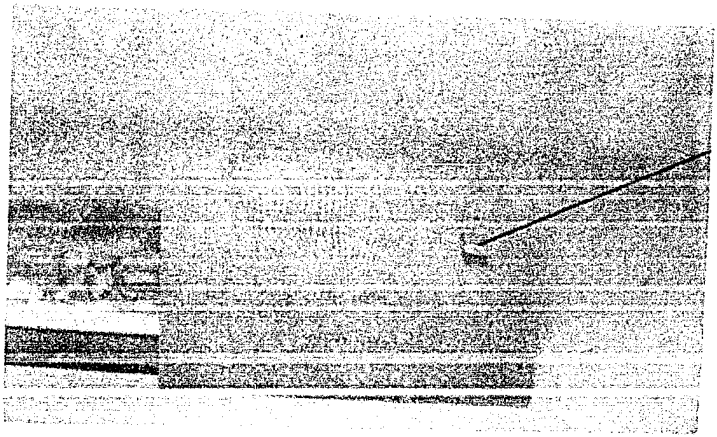
Fotografía # 1: El prototipo del SIS Z8, debajo de los cables de unión se encuentra la tarjeta impresa con el Z8612.



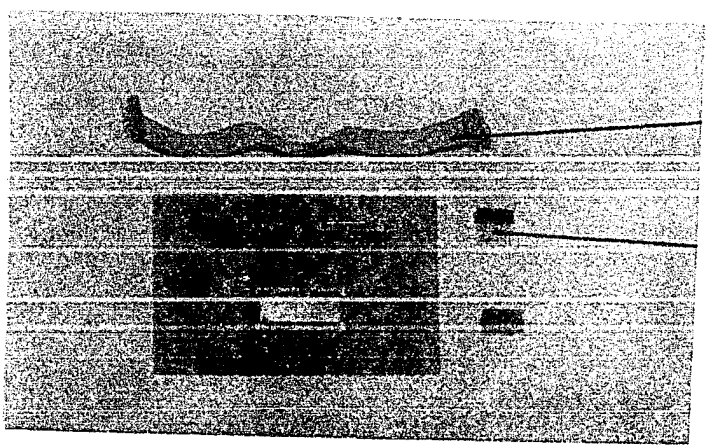
Fotografía # 2: Tarjeta principal del SIS Z8. Se muestra la disposición de los elementos.



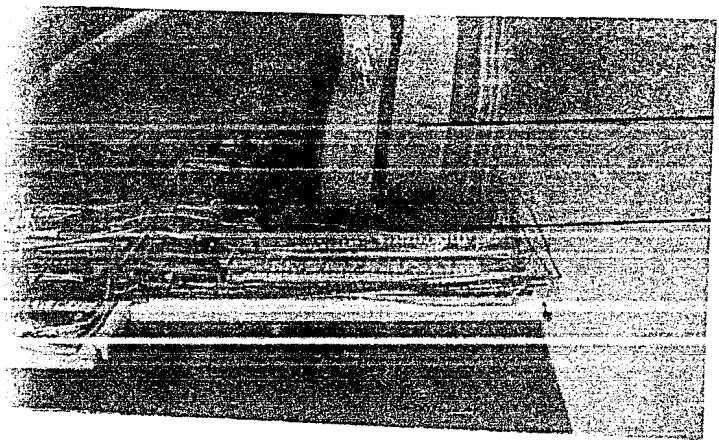
Fotografía # 3: Tarjeta secundaria (Z8612).



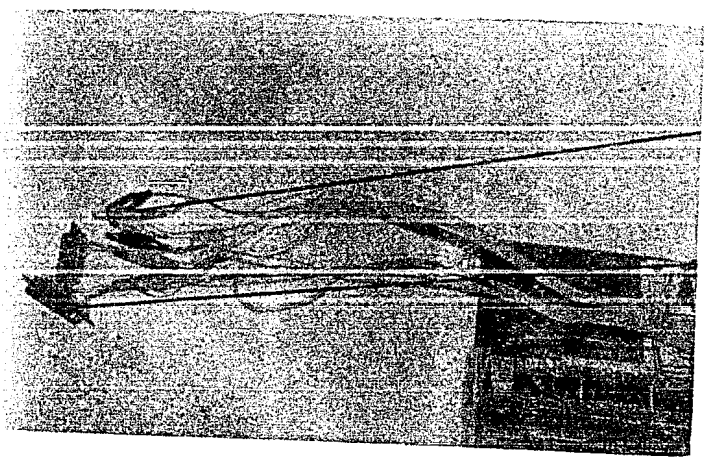
Fotografía # 4: Vista anterior de la tarjeta secundaria. El cristal de cuarzo [illegible] soldado a esta tarjeta debido a lo delgado de sus patillas.



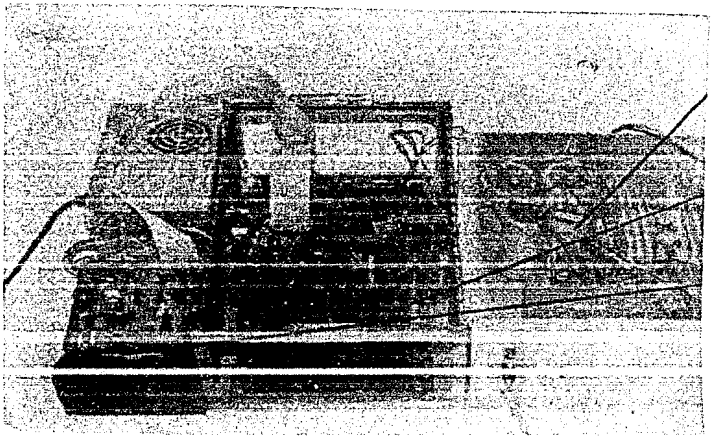
Fotografía # 5: Tarjeta Secundaria.



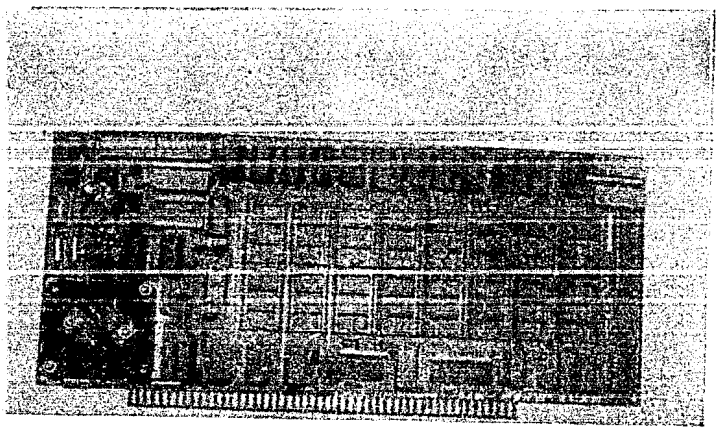
Fotografía # 6: Detalle de conectores de cables de unión.



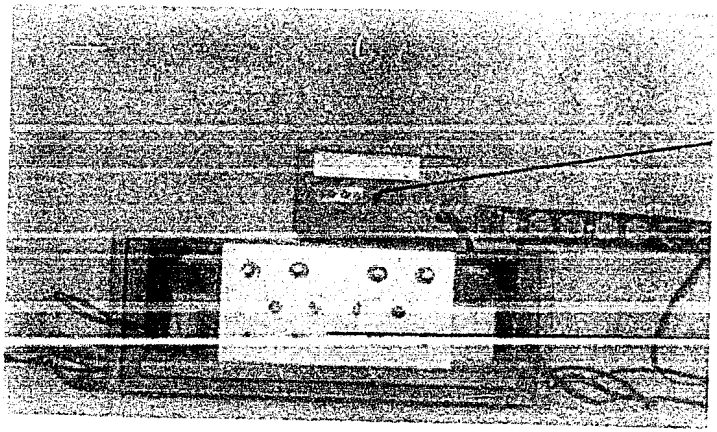
Fotografía # 7: Detalle de arnes de conectores.



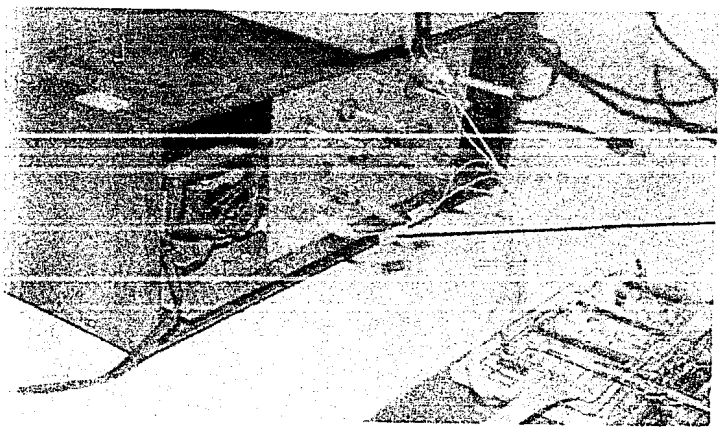
Fotografía # 8: Computadora empleada en el desarrollo del SIS Z8. Se trata de un viejo sistema CROMEMCO con 64KB de memoria, y una unidad de disco flexible de 5 ¼ ".



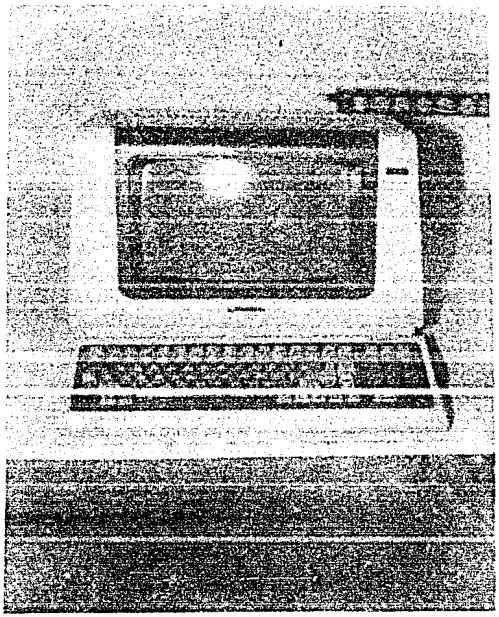
Fotografía # 9: Tarjeta 32KBYTESAVER empleada en el grabado de EPROMS 2716.



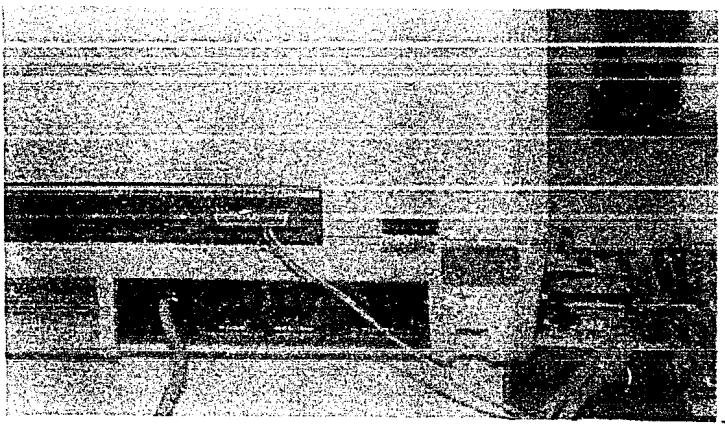
Fotografía # 10: Las fuentes de Poder empleadas para energizar al SIS Z8.

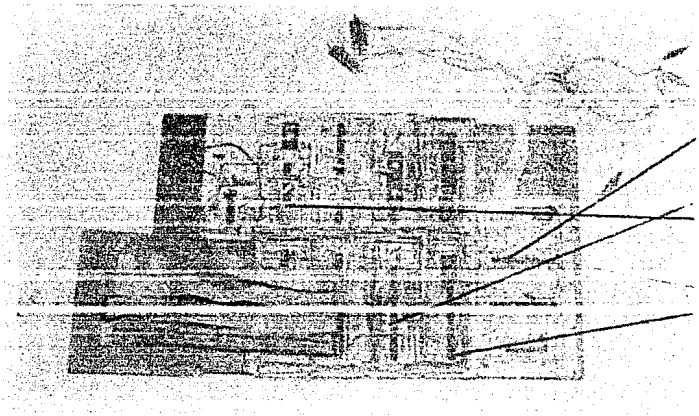


Fotografía # 11: Detalle de conección del SIS Z8 a las fuentes de poder Cada caimán tiene una etiqueta que indica su conección.

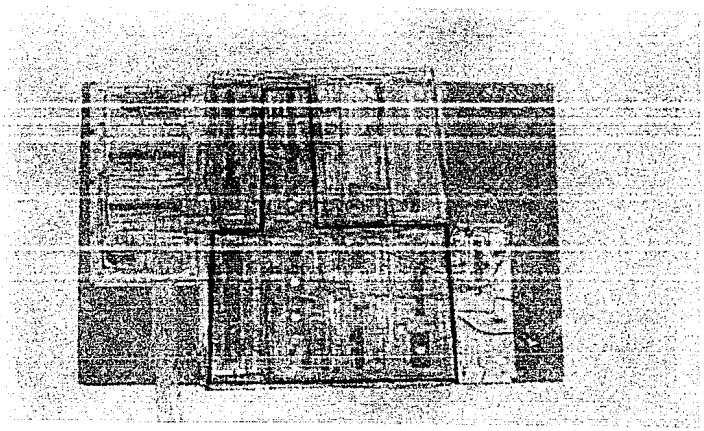


Fotografias # 12 y 13
Terminal empleada para
desarrollo del SIS Z8
y detalle de conección
del puerto serie del
SIS Z8 con la terminal.





Fotografía # 14: Distribución de los bloques del SIS Z8 sobre la Tarjeta Principal.



Fotografía # 15: El bus de datos/direcciones formaba un anillo, de tal manera que podía zafarse un cable sin afectar la operación del SIS Z8.

Sin contar el tiempo de importación del Z8612, la implementación del SIS Z8 consumió por si sola aproximadamente un año. Considerando que, aproximadamente, se disponia diario de 2 a 3 horas de laboratorio, 2 o 3 dias por semana y que este tiempo incluye uso de computadoras para introducir y ensamblar programas, además del grabado de EPROMS. Podemos afirmar que si bien la implementación no fué concluida en un tiempo record, por lo menos si fué adecuada. Uno de los aciertos que permitieron concluir el SIS Z8, fué la limpieza sistemática del alambrado. Solo en dos ocasiones fué necesario el mantenimiento correctivo del SIS Z8: Una vez por inversión de las fuentes de +12 y -12V , y otra por un falso contacto en la polarización de un C. I.

Las modificaciones que necesariamente ocurran durante la depuración del sistema deben ser sistemáticamente anotadas sobre los diagramas preliminares.

Es natural que durante esta fase ocurrirá una "lluvia de ideas" para mejorar el sistema. Estas ideas deben ser apuntadas, pero nunca ejecutadas ya que de lo contrario la tarea puede ser de duración infinita. Estas nuevas ideas y modificaciones serán útiles solo para el proyecto de industrialización.

4.-DOCUMENTACION

Esta última fase es tan importante como las demás. Si el prototipo llega a ser algún día un producto comercial, depende en buena medida del profesionalismo con que esta fase sea concluída, que por ser la última del proyecto puede no prestársele la debida atención. De la documentación del proyecto pueden obtenerse las bases de los que serán los manuales de operación y mantenimiento del producto final, además de que permitirá la continuidad del proyecto en su fase de industrialización, aunque no sea con los mismos elementos humanos.

CONCLUSIONES

A lo largo de este trabajo, hemos analizado extensamente al Z8 y ahora debe resultar evidente que este elemento pertenece a una familia de nuevos dispositivos, cuya arquitectura representa una concepción significativamente distinta a sus parientes mas cercanos, los microprocesadores.

Como consecuencia de lo anterior, existen aplicaciones para las que especificamente fueron creados los microcomputadores de un solo chip (MCSC), con vistas a disminuir el costo y aumentar la eficiencia en comparación con su realización con µP y circuitos periféricos de soporte.

El Z8 ha sido diseñado para emplearse en sistemas donde el número de elementos y el espacio físico son una limitante. Por lo tanto, algunas de las aplicaciones mas importantes de este dispositivo son en instrumentación, interfasic y processamiento distribuido, tomando tareas que tradicionalmente han sido ejecutadas por el µP anfitrión (o por Hardware adicional), por mencionar algunos ejemplos: control de dispositivos de E/S, transferencia y formateo de datos, etc.

Por otro lado es conveniente hacer notar, que debido a las carencia de instrucciones de E/S del Z8, el bus de control sumamente limitado y el reducido soporte de interrupciones, así como la ausencia de interrupciones no mascarables, hacen que este integrado no sea una buena elección para la sustitución de funciones comúnmente asignadas a los µP, tales como computación.

Al aparecer en el mercado el Z8 y analizando los proyectos que pueden desarrollarse con él, fué como surgio la idea de crear un sistema Hardware - Software para hacer más accesible y ágil el desarrollo de sistemas basados en el propio Z8.

En el mercado mundial existen dispositivos análogos, pero dadas las caracteristicas del desarrollo de sistemas en nuestro país, estos no están disponibles, ni son distribuidos comercialmente. Zilog, el fabricante del Z80 y el Z8, vende emuladores para diseños en laboratorio y pruebas en campo, a un precio bastante elevado.

El SIS Z8 es una poderosa herramienta que permite el diseño de software aplicable a nuevos sistemas basados en el Z8. La manipulación convencional de estos desarrollos es ejecutada al través de multiples grabaciones de programas en EPROMS, siendo las correcciones y depuración de los mismos una tarea tediosa, amèn de complicada. Dado que el SIS Z8 provee un monitor/depurador de programas, facilita estas tareas, coadyuvando así al desarrollo acelerado de tales sistemas.

El proceso mencionado anteriormente puede desarrollarse àgilmente en el SIS Z8, siempre y cuando se tomen en cuenta las siguientes condiciones, detalladas a continuación, y que son necesarias para un funcionamiento correcto del mismo:

a) Se debe conocer ampliamente al Z8, asi como disponer de suficiente información al respecto, por si se llegara a presentar algún problema en el desarrollo del proyecto.

b) Contar con software adecuado para realizar los programas de aplicación necesarios. El software que es indispensable para sistemas profesionales, es el que permite el ensamblado de los programas de aplicación y que puede ser un Macroensamblador.

c) Una vez teniendo los programas ensamblados y depurados, es necesario contar con un programador de memorias EPROM.

d) Al establecer la comunicación SIS Z8-Computadora Anfitriona, es necesario también establecer la comunicación SIS Z8-Terminal para poder tener un control del sistema , mediante un monitor.

e) Por lo anterior, los sistemas que pueden ser empleados como cpu anfitrion son tipicamente aquellos en que la consola es accesada por un puerto serie, como es el caso de los sistemas CROMEMCO, siendo este factor una de sus principales limitaciones.

Partiendo del desarrollo actual del SIS Z8, algunas se las caracteristicas que se le pueden adaptar, serian las siguientes:

1.- Un sistema, junto con el Firmware correspondiente, para el grabado de EPROMS.

2.- Adaptar su funcionamiento para acoplarlo con computadoras mas comerciales, como podria ser una microcomputadora personal.

3.- Generar màs rutinas en forma de utilerias, para que de esta manera el sistema fuese màs general y en caso

de su comercialización, lograr una aceptación adecuada por parte del usuario.

El SIS Z8 requiere aún de ciertas mejoras, sin embargo posee lo escencial para este tipo de dispositivos. Hasta la fecha no se ha analizado la posibilidad de su comercialización, pero en caso de presentarse la opción, aún resta bastante trabajo para mejorar su presentación y simplificar y extender las rutinas del depurador, lo cual sería escencial para este objetivo.

Finalmente, una de las experiencias más positivas de este trabajo, es que, a manera de un sencillo experimento, el desarrollo del SIS Z8 fué realizado, prácticamente en su totalidad, con recursos y equipos de pequeñas empresas privadas, cuyo rubro principal no es el diseño en electrónica, y que sin embargo la capacidad instalada en ellas, potencialmente permite la creación de tecnología propia en electrónica, como el éxito en el desarrollo del SIS Z8 lo ha confirmado. Quisiéramos, pues, concluir este trabajo, destacando la importancia de fomentar el diseño en electrónica en pequeñas empresas privadas, como algo, que a nuestro juicio, sería una alternativa para acelerar el proceso de desarrollo de la electrónica en este, nuestro país, que tanto requiere de independencia tecnológica.

Por mi raza hablará el espíritu.

México, D. F.,21 de Noviembre de 1987.

BIBLIOGRAFIA

AN ENGINEERING APPROACH TO DIGITAL DESING,
Willian Fletcher,
Prentice Hall.

AN INTRODUCTION TO MICROCOMPUTERS,
Volumen II
Some Real Microprocessors,
A. Osborne-Mc. Graw Hill.

AN INTRODUCTION TO COMPUTER LOGIC,
Nagle, Carroll, Irwin,
Prentice Hall.

DIGITAL INTEGRATED ELECTRONICS,
Taub, Schilling,
Mc. Graw Hill.

DIGITAL HARDWARE DESING,
John B. Peatman,
Mc. Graw Hill.

INTRODUCTION TO MICROPROCESSOR SYSTEM DESING,
Harry Garland,
Mc. Graw Hill.

LOGIC HANDBOOK,
Volumen II,
National.

TTL HANDBOOK,
Texas Instruments,

ZILOG COMPONENTS DATABOOK,
ZILOG 1980.

ZILOG Z8 HANDBOOK,
ZILOG 1980.

Z8 PLZ/ASM,
Assembly Language Programming,
December 1980,
ZILOG:

Z80 ASSEMBLER LANGUAGE PROGRAMMING,
Leventhal,
A. Osborne, Mc. Graw Hill.

Z80/8080 CROSS ASSEMBLER FOR THE ZILOG Z8 MICROPROCESSOR,
Allen Ashley.

32K BYTE SAVER TECHNICAL MANUAL,
CROMEMCO 1980.

APENDICE A

HOJAS DE DATOS DEL HM6264

PRELIMINARY

NEW DEVICE

(IE-1B)

July, 1982

HITACHI

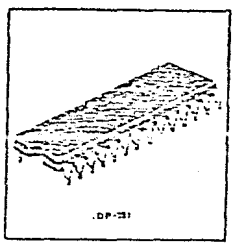
Semiconductor & Integrated Circuits Div.

HI-CMOS 64K STATIC RAM

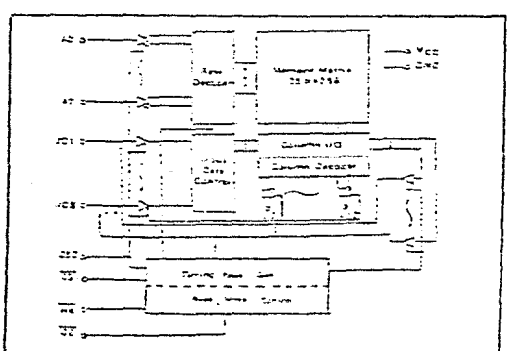
HM6264P-10/-12/-15, HM6264LP-10/-12/-15

8192 × 8 BIT HIGH SPEED STATIC HI-CMOS RAM

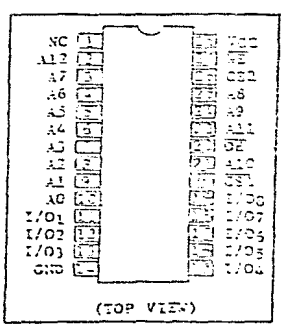
- .Single 5V Supply.
- .High speed: Fast Access Time 100ns/120ns/150ns max.
- .Equal Access and Cycle Time.
- .Completely Static RAM: No Clock or Timing Strobe Required.
- .Low Power Standby and Low Power Operation;
  - Standby: 10uW typ.
    - Two Chip Selection for Battery Back up.
  - Operation: 200mW typ.
- .Capability of Battery Back up Operation.(LP)
- .Common Data Input and Output, Three State Outputs.
- .Directly TTL Compatible: All Inputs and Outputs.
- .Standard 28pin Package Configuration.
- .Pin Out Compatible with 64K EPROM HN482764.



Functional Block Diagram



Pin Arrangement



* Truth Table

Mode	WE	CS1	CS2	OE	I/O	Current	Note
Not Selected (Power Down)	*	H	*	*	High-Z	ISB, ISB1	
	*	*	L	*	High-Z	ISB, ISB2	
Output Disabled	H	L	H	H	High-Z	ICC, ICC1	
Read	H	L	H	L	Dout	ICC, ICC1	
Write	L	L	H	*	Din	ICC, ICC1	Write Cycle(1)
	L	L	H	L	Din	ICC, ICC1	Write Cycle(2)

* Don't Care (H or L)

Notes: The specifications of this device are subject to change without notice.
Please contact your nearest Hitachi's Sales Dept. regarding specifications.

* Absolute Maximum Ratings

Item	Symbol	Rating	Unit
Voltage on Any Pin Relative to GND	Vin	-0.5*to 7	V
Operating Temperture	Topr	0 to 70	°C
Storage Temperature	Tstg	-55 to 125	°C
Temperature under Bias	Tbias	-10 to 85	°C
Power Dissipation	P_T	1.0	W

*-3.0V (pulse width 50ns)

* Recommened DC Operating Conditions
(Ta=0°C to 70°C)

Parameter	Symbol	min	typ	max	Unit	Note
Supply Voltage	Vcc	4.5	5.0	5.5	V	
	GND	0	0	0	V	
Input High (logic 1) Voltage	VIH	2.2		6.0	V	
Input Low (logic 0) Voltage	VIL	-0.3*		0.8	V	-3.0V (Pulse width 50ns)

* DC and Operating Characteristics
(Ta=0°C to 70°C, Vcc=5V±10%, GND=0V)

Parameter	Symbol		HM6264P(LP) -10/12/15 min	typ [1]	max	Unit	Test Conditions	Notes
Input Leakage Current	$\|I_{LI}\|$				2	μA	Vin=GND to V_{CC}	
Output Leakage Current	$\|I_{LO}\|$				2	μA	$\overline{CS1}$=VIH or CS2=VIL or $\overline{OE}$=VIH $V_{I/O}$=GND to V_{CC}	
Operating Power Supply Current: DC	I_{CC}			40	60	mA	$\overline{CS1}$=VIL CS2=VIH $I_{I/O}$=0mA	
Average Operating Current	I_{CC1}			60	110	mA	Minimum Cycle duty=100% $\overline{CS1}$=VIL, CS2=VIH	
Standby Power Supply Current (1)	I_{SB}			1	3	mA	$\overline{CS1}$=VIH or CS2=VIL $I_{I/O}$=0mA	
Standby Power Supply Current(2): DC	I_{SB1}	P		0.02	2	mA	$\overline{CS1}$ ≧ Vcc-0.2V, CS2 ≧ Vcc-0.2V or CS2≦0.2V	[2]
		LP		2	100	μA		
Standby Power Supply Current(3): DC	I_{SB2}	P		0.02	2	mA	CS2≦0.2V	[2]
		LP		2	100	μA		
Output Low Voltage	V_{OL}				0.4	V	I_{OL}=2.1mA	
Output High Voltage	V_{OH}		2.4			V	I_{OH}=-1.0mA	

Notes: 1) Typical limits are at Vcc=5.0V, Ta=+25°C and specified loading.
2) VILmin=-0.3V

* Capacitance (Ta=25°C, f=1.0 MHz) [1]

Parameter	Symbol	TYP	max	Unit	Conditions	Notes
Input Capacitance	C_{in}		6	pF	V_{in}=0V	
Input / Output Capacitance	$C_{I/O}$		8	pF	V_{IO}=0V	

Note: 1) This parameter is sampled and not 100% tested.

* AC Test Conditions
  - Input Pulse levels..... 0.8V to 2.4V
  - Input rise and fall times..... 10ns
  - Input and Output timing reference level... 1.5V
  - Output load..... 1TTL Gate and CL=100pF (Including scope & Jig)

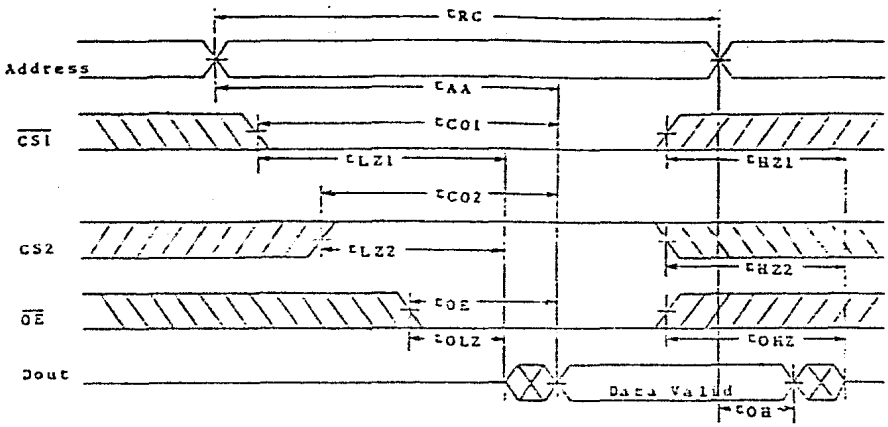
* AC Characteristics
  (Ta=0°C to 70°C, Vcc=5.0V±10% unless otherwise noted)

* Read Cycle

Parameter	Symbol	HM6264P(LP)-10		HM6264P(LP)-12		HM6264P(LP)-15		Unit
		min	max	min	max	min	max	
Read Cycle Time	t_{RC}	100		120		150		ns
Address Access Time	t_{AA}		100		120		150	ns
Chip Selection ($\overline{CS1}$) to Output	t_{CO1}		100		120		150	ns
Chip Selection (CS2) to Output	t_{CO2}		100		120		150	ns
Output Enable($\overline{OE}$) to Output Valid	t_{OE}		50		60		70	ns
Chip Selection ($\overline{CS1}$) to Output in Low Z	t_{LZ1}	10		10		15		ns
Chip Selection (CS2) to Output in Low Z	t_{LZ2}	10		10		15		ns
Output Enable($\overline{OE}$) to Output in Low Z	t_{OLZ}	5		5		5		ns
Chip Deselection ($\overline{CS1}$) to Output in High Z	t_{HZ1}	0	35	0	40	0	50	ns
Chip Deselection (CS2) to Output in High Z	t_{HZ2}	0	35	0	40	0	50	ns
Output Disable($\overline{OE}$) to Output in High Z	t_{OHZ}	0	35	0	40	0	50	ns
Output Hold from Address Change	t_{OH}	10		10		15		ns

Notes: 1) t_{HZ} and t_{OHZ} are defined as the time at which the outputs achieve the open circuit condition and are not referred to output voltage levels.
2) At any given temperature and voltage condition, t_{HZmax} is less than t_{LZmin} both for a given device and from device to device.

Timing Waveform of Read Cycle [1]



Note: 1) $\overline{WE}$ is High for Read Cycle.

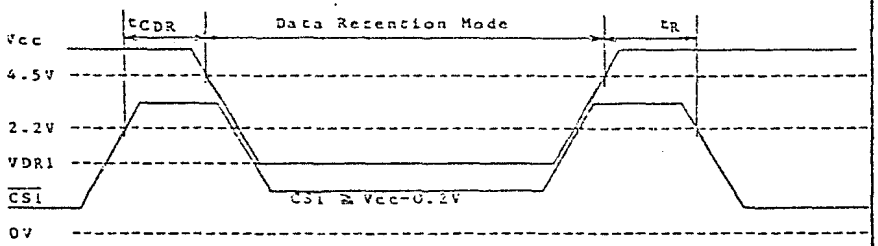
* Low Vcc Data Retention Characteristics (Only for LP versions)

[1] VILmin= −0.3V

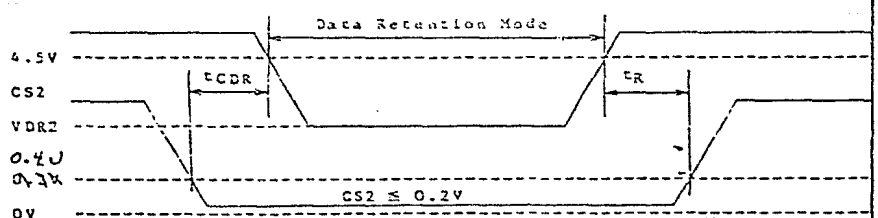
(Ta= 0°C to 70°C) [2] tRC=Read Cycle Time

Parameter	Symbol	Test Conditions	min	typ	max	unit
Vcc for Data Retention (1)	V_{DR1}	$\overline{CS1} \geq Vcc-0.2V$ $CS2 \geq Vcc-0.2V$	2.0			V
Vcc for Data Retention (2)	V_{DR2}	$CS2 \leq 0.2V$	2.0			V
Data Retention Current (1)	I_{CCDR1}	Vcc=3.0V $\overline{CS1} \geq Vcc-0.2V$, $CS2 \geq Vcc-0.2V$ or $CS2 \leq 0.2V$		1	[1] 50	μA
Data Retention Current (2)	I_{CCDR2}	Vcc=3.0V $CS2 \leq 0.2V$		1	[1] 50	uA
Chip Deselection to Data Retention Time	t_{CDR}	See Retention Waveform	0			ns
Operation Recovery Time	t_R		[2] t_{RC}			ns

Low Vcc Data Retention Waveform No.1 ($\overline{CS1}$ Controlled)



Low Vcc Data Retention Waveform No.2 (CS2 Controlled)



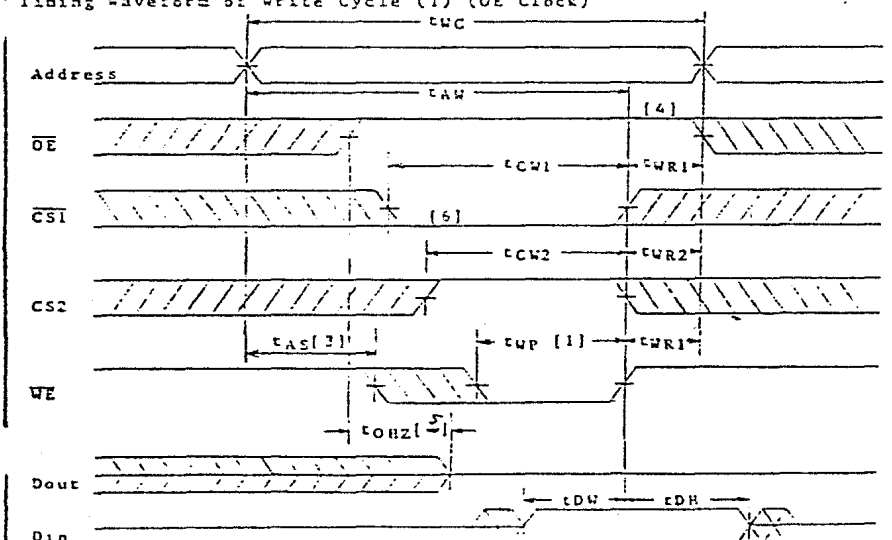
Note: 1) CS2 controls Address buffer, $\overline{WE}$ buffer, $\overline{CS1}$ buffer and Din buffer. If CS2 controls data retention mode, Vin levels (Address, $\overline{WE}$, $\overline{CS1}$, I/O) can be in the high impedance state. If $\overline{CS1}$ controls data retention mode, CS2 must be $CS2 \geq Vcc-0.2V$. The other inputs' level (address, $\overline{WE}$, I/O) can be in the high impedance state.

or $\leq 0.2V$

* Write Cycle

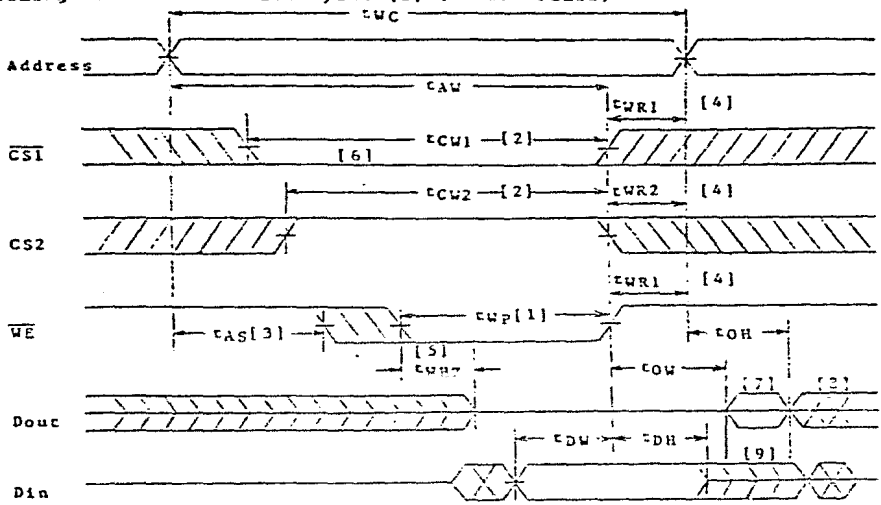
Parameter	Symbol	HM6264P(LP)-10		HM6264P(LP)-12		HM6264P(LP)-15		Unit
		min	max	min	max	min	max	
Write Cycle Time	t_{WC}	100		120		150		ns
Chip Selection to End of Write	t_{CW}	80		85		100		ns
Address Set Up Time	t_{AS}	0		0		0		ns
Address Valid to End of Write	t_{AW}	80		85		100		ns
Write Pulse Width	t_{WP}	60		70		90		ns
Write Recovery Time from $\overline{WE}$ or $\overline{CS1}$	t_{WR1}	5		5		10		ns
Write Recovery Time from CS2	t_{WR2}	15		15		15		ns
End of Write to Output in High Z	t_{WHZ}	0	35	0	40	0	50	ns
Data Valid to End of Write	t_{DW}	40		50		60		ns
Data Hold from End of Write	t_{DH}	0		0		0		ns
Output disable($\overline{OE}$) to Output in High Z	t_{OHZ}	0	35	0	40	0	50	ns
Output Active from End of Write	t_{OW}	5		5		10		ns

Timing Waveform of Write Cycle (1) ($\overline{OE}$ Clock)



HITACHI

Timing Waveform of Write Cycle (2) ($\overline{OE}$ Low Fixed)



Notes: 1) A write occurs during the overlap of a low $\overline{CS1}$, a high CS2 and a low $\overline{WE}$. A write begins at the latest transition among $\overline{CS1}$ going low, CS2 going high and $\overline{WE}$ going low. A write ends at the earliest transition among $\overline{CS1}$ going high, CS2 going low and $\overline{WE}$ going high. tWP is measured from the beginning of write to the end of write.

2) tCW is measured from the later of $\overline{CS1}$ going low or CS2 going high to the end of write.

3) tAS is measured from the address valid to the beginning of write.

4) tWR is measured from the end of write to the address change. tWR1 applies in case a write ends at $\overline{CS1}$ or $\overline{WE}$ going high. tWR2 applies in case a write ends at CS2 going low.

5) During this period, I/O pins are in the output state, therefore the input signals of opposite phase to the outputs must not be applied.

6) If $\overline{CS1}$ goes low simultaneously with $\overline{WE}$ going low or after $\overline{WE}$ going low, the outputs remain in high impedance state.

7) Dout is in the same phase of written data of this cycle.

8) Dout is the read data of the new address.

9) If $\overline{CS1}$ is low and CS2 is high during this period, I/O pins are in the output state. Therefore, the input signals of opposite phase to the outputs must not be applied to them.

APENDICE B

INTRODUCCION AL LENGUAJE ENSAMBLADOR

El conjunto de instrucciones de un microprocesador es un conjunto de entradas binarias las cuales producen acciones definidas durante un ciclo de instrucciòn. Esta conjunto de instruciones son fundamentales para poder establecer una acciòn bien definida que ejecutarà el microprocesador.

Cada instrucciòn de este conjunto, es un simple patròn binario, el cuàl debe de ser instroducido como informaciòn hacia el microprocesador para que sea interpretada como una instrucciòn.

QUE ES UN PROGRAMA?

Un programa escrito en cualquier lenguaje de alto nivel ò en lenguaje ensamblador, es una serie de instrucciones que causan que una computadora desarrolle una tarea particular. En forma màs precisa, se puede afirmar que un programa de computadora incluye algo màs que instrucciones, tambièn contiene los datos y las direcciones de memoria que el microprocesador (de la computadora) necesita para complementar y ejecutar adecuadamente cada instrucciòn del programa.

Después de tener el programa fuente, èste es traducido a un conjunto de nùmeros binarios, este conjunto es lo que se conoce como PROGRAMA OBJETO ò LENGUAJE DE MAQUINA. Toda vez que el programa objeto està creado, està listo para ser ejecutado.

Para que una computadora pueda ejecutar un programa, èste debe de estar en lenguaje de màquina, ya que el microprocesador entiende solo instrucciones binarias, por lo que se tiene que contar con otro programa (un programa del sistema), que convierta cada una de las instrucciones a su equivalente en lenguaje de mÀquina.

Si no se tuvieran este tipo de programas, para unaaplicaciòndada se tendria que programar en binario, lo que ocasionaria que los programas fueran dificiles de entender y por consiguiente de corregir.

INSTRUCCIONES ESCRITAS EN NEMONICOS

Una manera de escribir programas, sin tener que utilizar patrones binarios, pero sin tener que utilizar también algùn lenguaje de alto nivel, es utilizando instrucciones, a las cuales se les asigna un nombre especifico. Este nombre que se le da a cada còdigo de instrucciòn es llamado un ' NEMONICO '. Este nemònico deberà describir de alguna manera que funciòn realiza la instrucciòn.

Cada microprocesador tiene su propio conjunto de nemonicos para el conjunto de sus instrucciones.

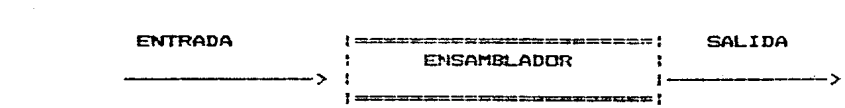
EL LENGUAJE ENSAMBLADOR

Una vez que se ha decidido programar utilizando un lenguaje de nemonicos, para que el programa pueda ser ejecutado deberà de ser traducido a lenguaje de màquina. Cada fabricante de microprocesadores aparte de proporcionar el conjunto de nemònicos asociados a sus instrucciones, proporciona su còdigo binario equivalente, por lo que un programa escrito en nemònicos puede ser traducido a lenguaje de màquina a manualmente.

Si el programa traducido es pequeño, la generaciòn del còdigo objeto puede no ocasionar problemas, pero si se trata de un programa de un nùmero considerable de instrucciones, su traducciòn manual puede ocasionar pequeños errores que tendràn como resultado una erronéa ejecuciòn del mismo.

Para evitar este problema, en lugar de realizar la traducciòn a lenguaje de màquina manualmente, se utiliza un programa llamado ENSAMBLADOR. El programa ensamblador traduce un programa de usuario o programa fuente escrito en nemonicos, a un programa escrito en lenguaje de màquina.

Esquemàticamente la funciòn de un ensamblador se puede representar de la siguiente manera:



Cada ensamblador tiene sus propias reglas de funcionamiento, las cuales incluyen el uso de ciertos delimitadores (como espacios, comas, puntos y comas y dos puntos) en lugares apropiados, una sintàxis especifica, un apropiado control de la informaciòn e inclusive el uso en el lugar apropiado, de nombres y nùmeros.

CARACTERISTICAS DE LOS ENSAMBLADORES

Los ensambladores en realidad pueden efectuar màs funciones, que el solo hecho de traducir nombres de nemònicos y registros en sus equivalentes binarios. Algunas de las caracteristicas adicionales de los ensambladores son :

1.- Permitir al usuario asignar nombres a localidades de memoria, a dispositivos de entrada y salida y aùn, a una secuencia de instrucciones (definiendo MACROS).

2.- Convertir datos o direcciones de varios sistemas numèricos (por ejemplo de hexadecimal ò decimal) a binario y convertir caracteres de còdigo ASCII ò EBCDIC a su còdigo binario equivalente.

3.- Informar al programa cargador (loader program), que partes de la memoria de programa ò de datos deberà ser procesado.

4.- Permitir al usuario asignar àreas de memoria para almacenamiento temporal de datos y colocar datos fijos en àreas de memoria de programa.

5.- Proporcionar la informaciòn necesaria para poder incluir otros programas externos, dentro del programa que se estè utilizando.

Es obvio que todas estas caracteristicas se reflejan en un mayor espacio en memoria que ocuparà el ensamblador para su ejecuciòn.

DESVENTAJAS DEL LENGUAJE ENSAMBLADOR

Una de las principales desventajas de programar en lenguaje ensamblador, es que se debe de tener un conocimiento detallado, de la computadora en la cual se està trabajando. Se debe de conocer con que registros e instrucciones cuenta el microprocesador de la computadora, como las instrucciones afectan los registros, que modos de direccionamiento se utilizan etc. Toda esta informaciòn puede ser irrelevante para la funciòn final

que ejecuta la computadora.

Por las razones anteriormente mencionadas se puede decir que los programas escritos en lenguaje ensamblador no son PORTABLES, ya que cada computadora tiene su propio lenguaje ensamblador el cual refleja su propia arquitectura.

El problema de PORTABILIDAD, reside en el hecho de que un programa escrito en ensamblador de una computadora dada, no puede ser ejecutado en otra màquina que tenga un microprocesador diferente.

ENSAMBLADORES

Esta parte del capitulo describe las funciones desarrolladas por los ensambladores, comenzando por las caracteristicas comunes de varios tipos de èstos, hasta capacidades màs elaboradas, tales como MACROS y condiciones de ensamblado.

CARACTERISTICAS DE LOS ENSAMBLADORES

Como se mencionò anteriormente, hoy en dia las funciones que pueden desarrollar los ensambladores, van màs allà que el simple hecho de traducir nemònicos en lenguaje ensamblador a còdigos binarios.

INSTRUCCIONES DEL ENSAMBLADOR

Las instrucciones o preposiciones del lenguaje ensamblador, estàn divididas en un nùmero de CAMPOS, como se muestra a continuaciòn :

CAMPO DE ETIQUETA	CODIGO DE OPERACION O CAMPO DE NEMONICO	OPERANDO O CAMPO DE LA DIRECCION	CAMPO DE COMENTARIO
ROOT:	CALL	AGUANTA	;LLAMA A LA RUTINA
	CALL	RECEPCION	
	JR	ROOT	; SALTA AL CONTROL
			; DEL PROGRAMA
RECEPCION:	JP	@RR4	; VA A LA RUTINA
			; DIRECCIONADA POR
			; RR4

El campo de código de operación. es el único campo que nunca puede estar vacio, ya que siempre contiene un nemónico de una instrucción o una directiva para el ensamblador la cual se conoce como una pseudo-operación.

El campo del operando puede contener una dirección, un dato ó puede estar en blanco.

Los campos de comentario y etiqueta son opcionales. Un programador asignará una etiqueta a una instrucción ó agregará un comentario si considera que es necesario, éste último principalmente. Esta desición de utilizar el campo de comentario, se debe de hacer tomando en cuenta de que su uso es importante para entender un programa escrito en ensamblador, ya que de hecho el utilizar nemónicos en un momento dado, puede causar problemas de comprención.

Los comentarios son utilizados para poder documentar un programa, explicando en forma general que función realiza una instrucción o un grupo de ellas.

El ensamblador tiene sus propias reglas de sintáxis para especificar donde termina un campo y comienza otro. Muchos ensambladores usan un símbolo o delimitador especial para especificar el fin o inicio de un campo. El más utilizado para este fin, es el espacio en blanco. También son usados como delimitadores comas, puntos, puntos y comas y dos puntos.

Aunque todos estos caracteres son utilizados como delimitadores, serán usados especificamente por el ensamblador que se este utilizando.

A continuación serán definidos los campos que constituyen una instrucción del lenguaje ensamblador.

ETIQUETAS

El campo de las etiquetas es el primero en una instrucción de lenguaje ensamblador. Este campo es opcional, por lo que puede aparecer en blanco. Si la etiqueta esta presente, el ensamblador le asigna el valor de la dirección para la localidad de memoria correspondiente al primer byte del programa objeto de la instrucción en que aparece la etiqueta.

Después de que la etiqueta ha sido utilizada, ésta puede ser utilizada subsecuentemente para representar un dato ò una dirección. Cuando el programa objeto es creado, el ensamblador substituirá el valor asignado a la etiqueta.

Las etiquetas son utilizadas generalmente en saltos (JUMP), llamadas a subrutinas (CALL) ò en instrucciones de ramificación.

Todas estas instrucciones colocan un nuevo valor en el contador de programa, y por lo tanto alteran la ejecución secuencial del programa. Por ejemplo, la instrucción JP @RR2, significa "colocar el valor de la dirección contenida en la localidad que se encuentra en el registro R2 y R3". JP MONITOR significa "colocar el valor asignado a la etiqueta llamada 'MONITOR ' en el contador de programa". La siguiente instrucción que será ejecutada será aquella que se encuentra en la localidad cuyo valor es el que fué sustituido al encontrarse la etiqueta.

Algunas de las razones por lo que es conveniente utilizar etiquetas, son las siguientes :

1.- Una etiqueta hace que una localidad del programa sea fàcil de encontrar y recordar.

2.- La etiqueta puede ser cambiada de lugar para cambiar o corregir un programa. En este caso no es necesario cambiar las instrucciones que siguen a la definicioòn de la etiqueta, el ensamblador se encarga de realizar todos los cambios necesarios.

3.- El ensamblador puede relocalizar un programa completo agregando una constante (una constante de relocalización) para cada dirección en la cuál una etiqueta fué utilizada. Por esta razòn, es sencillo modificar un programa mediante la inserciòn de otros programas ò simplemente mediante la reorganizaciòn de la memoria.

4.- Mediante el uso de etiquetas es fàcil utilizar programas separados y formar una biblioteca de programas. Por ejemplo, cada uno de estos programas se puede anexar para formar un programa totalmente diferente.

Una vez establecidas las razones de porquè es importante el uso de etiquetas dentro de un programa, la siguiente pregunta que nos podría surgir, es la de que características deben de tener las etiquetas seleccionadas. El ensamblador casi siempre tiene restricciones en el nùmero de caracteres utilizados en el nombre de etiquetas, el caracter inicial y los caracteres permitidos en dicho nombre. Una vez cumplidas estas especificaciones la desiciòn de que etiquetas se deben de utilizar es del programador.

El nombre de etiqueta seleccionado, debe ser tal que sugiera el propòsito de la misma. Por ejemplo, el nombre PRUEBAMEMO, debe de corresponder a una rutina en la cuàl se hace una prueba de memoria antes de empezar a trabajar con un sistema determinado.

Algunos programadores prefieren utilizar un formato estandar para el uso de etiquetas, por ejemplo empezando con la etiqueta ET000. Este tipo de etiquetas llevan una secuencia que permite su fàcil detecciòn, aunque no ayudan a la documentaciòn del programa.

Existen algunas reglas para la selecciòn de etiquetas para que en un momento dado nos ayuden a seleccionar el nombre de éstas. Recomendamos las siguientes:

1.- No utilizar etiquetas cuyo nombre pudiera parecerse a un nombre de nemònico o a algùn còdigo de instrucciòn, ya que puede crear confusiòn aùn al propio ensamblador.

2.- No utilizar etiquetas cuyo nombre exceda a la longitud permitida por el ensamblador, pués en la mayoría de las veces, son truncadas al nùmero permitido de caracteres, pudiendo perder su significado.

3.- Evitar escoger caracteres especiales (no alfabèticos o no numéricos), ò letras minùsculas, ya que algunos ensambladores no lo permiten. La regla màs sencilla a seguir a èste respecto, es seleccionar solo letras mayùsculas y nùmeros.

4.- Comenzar el nombre de cada etiqueta con una letra. Este tipo de etiquetas son siempre aceptadas.

5.- No utilizar dos o màs nombres de etiquetas que puedan confundirse. Evitar las letras I,O,Z y los numeros 0,1,2. También evitar nombres como XXXX y XXXXX ya que carecen de todo sentido.

6.- Cuando no se està seguro si una etiqueta es permitida no usarla.

. Los seis puntos anteriores también pueden ser considerados como recomendaciones que pueden servir en la selección de etiquetas.

CODIGOS DE OPERACION (NEMONICOS)

La principal tarea del ensamblador, es la traducción de los códigos de operación de los nemonicos a sus códigos binarios equivalentes. El ensamblador realiza esta función utilizando una tabla definida, similar a la utilizada en el ensamblado manual.

Sin embargo, el ensamblador debe de realizar más funciones aparte de traducir los códigos de operación, ya que debe de determinar cuántos campos requiere la instrucción y de que tipo deben de ser. Esta sola tarea es por demás compleja, ya que algunas instrucciones no necesitan de operandos para ejecutarse, otras necesitan de uno y algunas otras necesitan de dos operandos.

PSEUDO-OPERACIONES

Algunas instrucciones del lenguaje ensamblador no son traducidas directamente a instrucciones en lenguaje de máquina. Este tipo de instrucciones son llamadas DIRECTIVAS AL ENSAMBLADOR. Estas instrucciones pueden asignar al programa ciertas áreas de memoria, definir símbolos, designar áreas de RAM para almacenamiento temporal, definir tablas de datos en memoria, permitir utilizar otros programas y ejecutar otros procesos importantes.

Para utilizar las directivas al ensamblador o pseudo-operaciones se debe de definir el nemónico de la pseudo-operación en el campo de código de operación, y si así lo requiere ésta, definir una dirección ó dato en el campo de dirección.

Las pseudo-operaciones más comunes son las siguientes:

ORG
EQUATE O DEFINE
DB(DEFINE BYTE)
DW(DEFINE WORD)
RESERVE
ENTRY
EXTERNAL

CAMPO DE LA DIRECCION Y DATO

Muchos ensambladores permiten una gran libertad en la descripción del contenido del campo de la dirección y operando.

Algunas opciones para el campo del operando son :

1) Números Decimales :

Varios ensambladores asumen que todos los numeros son decimales a menos que se especifique algún otro tipo.

2) Otros sistemas numéricos :

Algunos ensambladores también aceptan numeros binarios, octales o hexadecimales. Estos tipos de numeros se deben de identificar de alguna manera, por ejemplo antes o después del número se pone un caracter especial. Algunos de estos identificadores que se utilizan son :

B ó	para números binarios.
O,Q ó C	para números octales.
H ó $	para números hexadecimales.
D	para números decimales.

El caracter D es oprional siendo el tipo por omisión.

Los ensambladores requieren que un número hexadecimal comience con un digito decimal (que en la mayoria de veces es el cero), para destinguir entre números y etiquetas.

3) Nombres simbólicos:

Nombres pueden aparecer en el campo del operando y son tratados como el dato que representa. Hay que recordar que existen diferencias entre datos y direcciones.

La siguiente secuencia :

```
TOTAL :  EQU 10
         ADD 10H,TOTAL
```

sumará el contenido de la localidad de memoria 10 al registro 10H.

4) El valor actual del contador de localidades(usualmente referido como * ó $).

Es útil principalmente en instrucciones de salto(JUMP), por ejemplo

JP $6

causa un salto a la localidad de memoria 6 palabras después de la palabra que contiene el primer byte de la instrucción JP.

5) Códigos de Caracteres:

Varios ensambladores permiten que partes de texto sean introducidas como cadenas ASCII. Tales cadenas pueden estar delimitadas por apostrofes o comillas, estas cadenas también pueden empezar o terminar con simbolos tales como A ó C. Pocos ensambladores también permiten cadenas EBCDIC.

6) Combinación de 1 a 5 con operadores logicos o aritméticos:

Casi todos los ensambladores permiten simples operaciones aritméticas como INICIO+2. Algunos ensambladores también permiten multiplicación , división, funciones lógicas, recorrimientos étc.
Estas operaciones son referenciadas como expresiones. Hay que notar que el ensamblador evalúa estas expresiones y las ensambla al mismo tiempo.

Los ensambladores tienen diferentes formas de aceptar las instrucciones y de como las interpretan. Expresiones complejas hacen que un programa sea difícil de leer y de entender.
Como en el caso del uso de etiquetas es conveniente seguir algunas recomendaciones para crear un programa que pueda ser fácil de entender, modificar y utilizar.

A este respecto y en general al diseñar y elaborar el programa en ensamblador se debe enfatizar en la claridad y simplicidad.

ENSAMBLADO CONDICIONAL

Algunos ensambladores permiten incluir o excluir partes del programa fuente, dependiendo de condiciones que son examinadas durante el tiempo del ensamblado. Este proceso es llamado ENSAMBLADO CONDICIONAL, y proporciona al ensamblador una flexibilidad parecida a la de un compilador. Aunque algunos ensambladores tienen limitada esta función, no deja de ser una gran ventaja el poder seleccionar una parte del programa y realizar una función específica dependiendo de una condición dada que depende de alguna característica de la aplicación que se esté ejecutando.

Por ejemplo, si en un proceso de transmisión de información a través de una terminal, y si se tuvieran varios modelos de éstas para realizar el proceso, se tendría que utilizar una condición de ensamblado dependiendo de la terminal utilizada, para ensamblar la parte del programa que contiene la aplicación utilizando los pàrametros de la terminal dada.

Una forma usual del ensamblado condicional :

```
IF CONDICION
   .
   .
.
PROGRAMA CONDICIONAL
.
.
   .
ENDIF
```

Si la expresión CONDICION es verdadera al momento del ensamblado, las instrucciones contenidas entre IF y ENDIF (2 pseudo-operaciones), son incluidas dentro del programa.

Algunos casos típicos del uso del ensamblado condicional son:

1.- Para incluir o excluir variables.
2.- Para establecer diagnósticos o pruebas especiales en pruebas de ejecución.
3.- Para permitir datos de varios bits de longitud.
4.- Para crear versiones especializadas de un programa común.

Desafortunadamente, el ensamblado condicional tiende a hacer confuso el programa y por consiguiente hacerlo difícil de leer. Por lo tanto hay que usar el ensamblado condicional solo si en realidad es necesario.

MACROS

Existen ocasiones dentro de un programa en que una secuencia de instrucciones tienen que aparecer repetidamente en el programa fuente. Este hecho se puede deber a la propia lógica del programa ò para compensar alguna deficiencia en el conjunto de instrucciones del microprocesador que se esté utilizando.

Para evitar escribir repetidamente la misma secuencia de instrucciones se puede utilizar lo que se conoce como un MACRO.

Las MACROS permiten asignar un nombre a una secuencia de instrucciones. El nombre de la macro se utiliza en el programa fuente en lugar de repetir la misma secuencia de instrucciones.

El ensamblador reemplazará el nombre de la macro con la apropiada secuencia de instrucciones, cada vez que la macro es invocada. Este hecho se puede ilustrar como sigue :

```
PROGRAMA FUENTE                                  PROGRAMA OBJETO

MACRO1      MACRO  (definición de la macro)

            instrucción 1
            instrucción 2
            instrucción 3

            ENDM  (fin de la definición de la macro)

            instrucción P1   (PROGRAMA FUENTE)      instrucción P1
            instrucción P2                          instrucción P2
            instrucción P3                           instrucción P3

                                                    instrucción 1
              MACRO1                                instrucción 2
                                                    instrucción 3
            instrucción P4                          instrucción P4
            instrucción P5                          instrucción P5
            instrucción P6                          instrucción P6
            instrucción P7                          instrucción P7

                                                    instrucción 1
              MACRO1                                instrucción 2
                                                    instrucción 3

            instrucción P8                          instrucción P8
            instrucción P9                          instrucción P9
                .                                       .
                .                                       .
```

Las macros no son lo mismo que las subrutinas. Una subrutina apararece una vez en el programa, y la ejecuciòn del programa translada a la subrutina cada vez que se llama. Una macro es EXPANDIDA a la actual secuencia de instrucciones cada vez que la macro ocurre, èsto quiere decir que una macro no causa que el contador del programa cambie de valor, no existe un salto a la localidad donde se encuentra definida la macro.

El uso de macros tiene las siguientes ventajas :

1) Hacen posible realizar un programa fuente màs corto.

2) Ofrecen una mejor documentaciòn de un programa.

3) Los programas que utilizan macros, son màs faciles de

   corregir puès si un error ocurre en la macro, se puede librar de errores gran cantidad del programa.

4) Los programas son faciles de modificar tan solo modificando la definiciòn de la macro.

5) Se pueden utilizar para aumentar la capacidad del conjunto de instrucciones que contenga el microprocesador usado.

Algunas ventajas del uso de macros son las siguientes:

1) Repeticiòn de las mismas instrucciones en lenguaje de màquina, ya que la macro es EXPANDIDA cada vez que es usada.

2) Una simple macro puede crear una gran cantidad de instrucciones.

3) La falta de estandarizaciòn en su uso puede hacer al programa dificil de leer y entender.

4) Posibles efectos sobre registros y banderas, las cuales no pueden estar claramente establecidas.

COMENTARIOS

Todos los ensambladores permiten colocar comentarios en el programa fuente. Estos comentarios no tienen efecto en el còdigo objeto, pero ayudan a leer, entender y documentar el programa.

El uso de buenos comentarios es una parte esencial en el desarrollo de programas escritos en lenguaje ensamblador. Sin comentarios, los programas son dificil de entender.

Algunas sugerencias para la documentación de un programa se describen a continuación:

1.- Utilizar comentarios para describir la función que realiza el programa no el efecto que tiene el uso de una instrución. Los comentarios deben de establecer cosas como "PROGRAMANDO PUERTOS","EXAMINANDO BANDERAS", "ALMACENANDO REGISTROS".
Los comentaris no deben decir cosas como : "SUMA 1 AL REGISTRO R10", "SALTA AL INICIO","PONE LA BANDERA DE ACARREO A 1". Se debe de describir como el programa está afectando al sistema. Los efectos internos dentro del CPU son de poco interés.

2.- Poner comentarios breves pero en el lugar adecuado. Los detalles pueden ser disponibles en cualquier lugar de la documentacion.

3.- Comentar TODOS los puntos clave.

4.- No comentar instrucciones estandard o secuencias que modifican contadores o apuntadores. Poner especial atención a instrucciones cuyo significado no sea obvio.

5.- No utilizar abreviaciones confusas.

6.- Hacer comentarios claros y legibles.

7.- Comentar todas las definiciones, describiendo su propósito. También demarcar las tablas y àreas de almacenamiento de datos.

8.- Comentar secciones del programa así como instrucciones individuales.

9.- Mantener una consistencia en la terminología. Se puede y se debe de ser repetitivo.

10.- No poner comentarios los cuales puedan confundir por no ser precisos, como por ejemplo "EN LA ULTIMA INSTRUCCION SE PUSO A CERO EL CARRY".

Un programa bien documentado, es fàcil trabajar con él. Se puede recobrar el tiempo gastado en su descripción varias veces.

TIPOS DE ENSAMBLADORES

Aunque todos los ensambladores realizan la misma tarea, sus implementaciones varian enormemente. No trataremos de describir todos los tipos de ensambladores existentes, solamente se definirán los terminos e indicaremos algunas caracteristicas de varios tipos de ensambladores.

CROSS-ASSEMBLER

Un cross-asambler es un ensamblador que "corre" en una computadora ajena a la que ensambla los programas.

La computadora en la cuál el croos-asambler se ejecuta, es tipicamente una computadora grande con gran soporte de software y periféricos veloces, tal como una IBM 370, UNIVAC 1108 o una BURROUGHS 6700. La computadora para la cuál el cross-asambler ensambla los programas es tipicamente una microcomputadora como la Z80. Varios cross-assambleres están escritos en FORTRAN por lo que son portables.

ENSAMBLADOR RESIDENTE

Un autoensamblador o ensamblador residente es un ensamblador que corre en la computadora para la cuál ensambla los programas.

El autoensamblador requiere espacio de memoria y periféricos, y puede correr mas lentamente.

MACROENSAMBLADOR

Un macroensamblador es un ensamblador que permite definir una secuencia de instrucciones como MACROS.

MICROENSAMBLADOR

Un microensamblador es un ensamblador usado para escribir los microprogramas que definen el conjunto de instrucción de una computadora.

METAENSAMBLADOR.

Un metaensamblador es un ensamblador que puede manejar varios conjuntos diferentes de instrucciones. El usuario debe definir el conjunto especifico de instrucciones a utilizar.

ENSAMBLADOR DE UNA PASADA.

Un ensamblador de un solo paso, es un ensamblador que reconoce el programa fuente una sola vez. Tal ensamblador debe de algún modo resolver el problema de referencias hacia adelante, tales como las instrucciones de salto las cuales utilizan etiquetas que aparecen despuis en el programa fuente y que aún no han sido definidas.

ENSAMBLADOR DE DOS PASADAS.

Un ensamblador de dos pasos es un ensamblador que reconoce el programa fuente en lenguaje ensamblador dos veces.En la primera pasada, el ensamblador reconoce y define todos los simbolos, en la segunda pasada el ensamblador reemplaza las referencias con las definiciones actuales. Un ensamblador de dos pasadas resuelve facilmente el problema de las referencias adelantadas. Sin embargo, una macro expandida y el ensamblado condicional pueden causar problemas.

Muchos de los ensambladores basados en microprocesadores requieren de dos pasos.

ERRORES

Los ensambladores normalmente proporcionan mensajes de errores, y frecuentemente consistan de una sola letra. Los errores, que con más frecuencia, aparecen al ensamblar un programa son :

1) Nombres indefinidos (frecuentemente mal escritos o no definidos).

2) Caracter ilegal (por ejemplo en un número binario un 2).

3) Formato ilegal (un delimitador incorrecto u operandos incorrectos).

4) Expresiones invalidas(por ejemplo 2 operandos en una sola linea).

5) Valor ilegal (usualmente un valor más grande de lo permitido).

6) Falta operando.

7) Definición doble(2 valores diferentes asignados a un mismo nombre).

8) Etiqueta legal.

9) Falta etiqueta.

10) Código de operación indefinido.

APENDICE C

EQUIPO UTILIZADO

Para la impresión y redacción del texto, se empleó el siguiente equipo:

Impresora ATI Z1050

Procesador de Textos WORD PERFECT 4.0.

Computadora PRINTAFORM 5203.

Para el diseño y depuración del SIS Z8, se emplearon los siguientes elementos:

Sistemas CROMEMCO (SII Y SIII).

Grabador de EPROMS 2716, 32 KBYTE SAVER (CROMEMCO).

Borrador de EPROMS.

Terminal TELEVIDEO 921.

Fuentes de alimentación (fabricación casera):

5 Volts, 2 Amperes.
+12 y -12 Volts, 1 Ampere.

Osciloscopio marca LEADER de 2 canales, 15 MHz.

Multimetro Digital marca FLUKE 74.

Ensamblador Cruzado de Z8 para sistemas Z80-8080.

Paquete de Depuración DEBUG para CDOS (CROMEMCO).

APENDICE D

HOJAS DE ESPECIFICAIONES DEL Z8

Electrical Parameters

1 Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3V to +7.0V

Operating Ambient Temperature . . . . . 0°C to +70°C

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

2 Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows: $+4.25V \le V_{CC} \le +5.25V$, GND = 0V, $0°C \le T_A \le +70°C$.

3 DC Characteristics

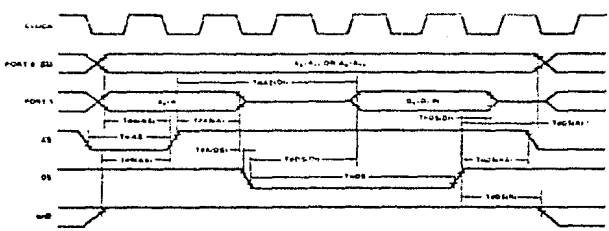
Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage			V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage			V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{RH}	Reset Input High Voltage			V		
V_{RL}	Reset Input Low Voltage			V		
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$	1
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0\ mA$	1
I_{IL}	Input Leakage			μA	$0 \le V_{IN} \le +5.25V$	
I_{OL}	Output Leakage			μA	$0 \le V_{IN} \le +5.25V$	
I_{IR}	Reset Input Current			μA	$V_{RL} = 0V$, $V_{CC} = +5.25V$	
I_{DD}	V_{DD} Supply Current			mA		
I_{HM}	V_{HM} Supply Current			mA		

1 For A_0-A_{11}, $\overline{RD}$, $\overline{STRB}$, SCLK and $\overline{TACK}$ on the 28-pin version $I_{OH} = -100\ \mu A$ and $I_{OL} = 1.0\ mA$

4 External Instruction Fetch, I/O or Memory Read Timing

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdAS(A)	Address Strobe to Address Float Delay Time	60		ns	Test Load 1	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		200	ns	Test Load 1	3
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdA(DS)	Address Float to Data Strobe Delay Time	0		ns	Test Load 1	
TwDS	Data Strobe Width	230		ns	Test Load 1	2
TdDS(DI)	Data Strobe to Data In Valid Delay Time		160	ns	Test Load 1	3
ThDS(DI)	Data In Hold Time	0		ns		
TdDS(A)	Data Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TdDS(AS)	Data Strobe to Address Strobe Delay Time	50		ns	Test Load 1	1
TdR(AS)	Read Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdDS(R)	Data Strobe to Read Change Delay	60		ns	Test Load 1	1

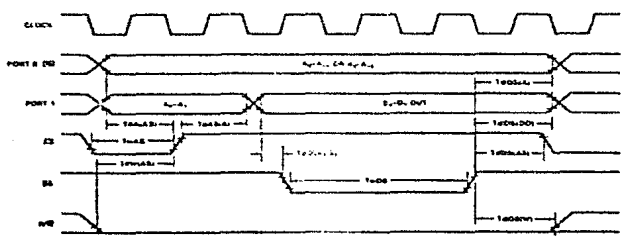
1. Delay times given are for an 8 MHz crystal input frequency. For lower frequencies, the change in clock periods must be added to the delay time.
2. Data Strobe Width is given for an 8 MHz crystal input frequency. For lower frequencies the change in three clock periods must be added to obtain the minimum width. The Data Strobe Width varies according to the instruction being executed. Refer to Figures 1.9 and 1.10.
3. Address Strobe and Data Strobe to Data In Valid delay times represent memory system access times and are given for an 8 MHz crystal input frequency. For lower frequencies, the change in four clock periods must be added to TdAS(DI) and the change in three clock periods added to TdDS(DI).
4. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."



5 External I/O or Memory Write Timing

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdAS(A)	Address Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdDO(DS)	Data Out Valid to Data Strobe Delay Time	30		ns	Test Load 1	1
TwDS	Data Strobe Width	150		ns	Test Load 1	2
TdDS(A)	Data Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TdDS(DO)	Data Strobe to Data Out Change Delay Time	60		ns	Test Load 1	1
[illegible]	Data Strobe to Address Strobe Delay Time	60		ns	Test Load 1	1
TdW(AS)	Write Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdDS(W)	Data Strobe to Write Change Delay Time	60		ns	Test Load 1	1

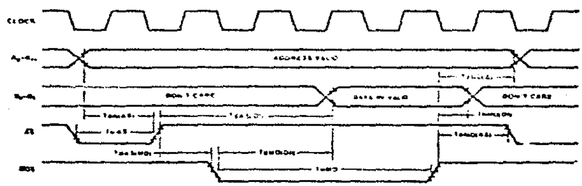
1. Delay times given are for an 8 MHz crystal input frequency. For lower frequencies, the change in clock period must be added to the delay time.
2. Data Strobe Width is given for an 8 MHz crystal input frequency. For lower frequencies the change in three clock periods must be added to obtain the minimum width. The Data Strobe Width varies according to the instruction being executed. Refer to Figures 1-B and 1-10.
3. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."



8
Memory Port (Z8/64) Timing

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 2	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		280	ns	Test Load 2	3
TwAS	Address Strobe Width	60		ns	Test Load 2	1
TdAS(MD)	Address Strobe to Memory Data Strobe Delay Time	60		ns	Test Load 2	1
TwMD	Memory Data Strobe Width	230		ns	Test Load 2	2
TdMD(DI)	Memory Data Strobe to Data In Valid Delay Time		160	ns	Test Load 2	1
ThMD(DI)	Data In Hold Time	0		ns		
TdMD(A)	Memory Data Strobe to Address Change Delay Time	60		ns	Test Load 2	1
TdMD(AS)	Memory Data Strobe to Address Strobe Delay Time	50		ns	Test Load 2	1

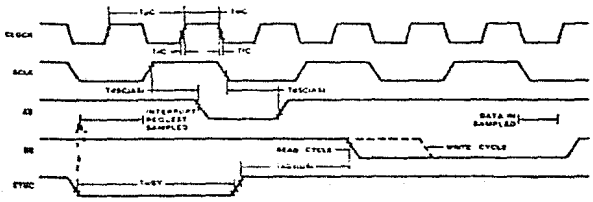
1. Delay times given are for an 8 MHz crystal input frequency. For lower frequencies, the change in clock periods must be added to the delay time.
2. Memory Data Strobe Width is given for an 8 MHz crystal input frequency. For lower frequencies the change in three clock periods must be added to obtain the minimum width. The Memory Data Strobe Width varies according to the instruction being executed. Refer to Figures 1-9 and 1-10.
3. Address Strobe and Memory Data Strobe to Data In Valid delay times represent memory system access times and are given for an 8 MHz crystal input frequency. For lower frequencies the change in four clock periods must be added to TdAS(DI) and the change in three clock periods added to TdMD(DI).
4. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".



7 Additional Timing

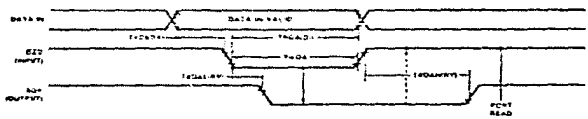
Symbol	Parameter	Min	Max	Unit	Condition	Notes
TpC	Input Clock Period	125		ns		
TrC, TfC	Input Clock Rise and Fall Times			ns	From External Clock Generator	
TwC	Input Clock Width			ns	From External Clock Generator	
TdSC(AS)	System Clock Out to Address Strobe Delay Time			ns		1
TdSY(DS)	Instruction Sync Out to Data Strobe Delay Time			ns		1, 2
TwSY	Instruction Sync Out Width			ns		1, 2

1. Test Conditions use Test Load 1 for SCLK and $\overline{SYNC}$ when output through their respective Port 3 pins and Test Load 2 on the SCLK and $\overline{SYNC}$ (direct) on pins on the 64 pin version.
2. Times given assume an 8 MHz crystal input frequency. For lower frequencies, the change in two clock periods must be added.
3. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."

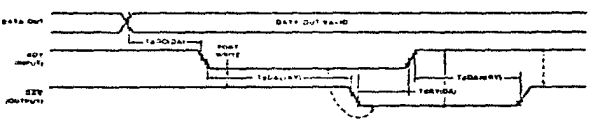


8 Handshake Timing

Symbol	Parameter	Min	Max	Unit	Condition
TsDI(DA)	Data In Setup Time	0		ns	
ThDA(DI)	Data In Hold Time	190		ns	
TwDA	Data Available Width			ns	Input Handshake Test Load 1
TdDAL(RY)	Data Available Low to Ready Delay Time			ns	Input Handshake Test Load 1
		0	-	ns	Output Handshake Test Load 1
TdDAH(RY)	Data Available High to Ready Delay Time			ns	Input Handshake Test Load 1
		0		ns	Output Handshake Test Load 1
[illegible]	Data Out Data Available Delay Time			ns	Test Load 1
TdRY(DA)	Ready to Data Available Delay Time			ns	Test Load 1



Input Handshake



Output Handshake

9
Test Load Circuits