



65
2ej

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
E INGENIERO MECANICO	ELECTRICISTA	
RODOLFO	HEREDIA	VAZQUEZ

DIRECTOR

ING. MANUEL CORREA SINTORA



México, D. F.

1987.



UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

INDICE

PROLOGO.....	15
CAPITULO I. INTRODUCCION.....	17
¿Que es el SIS Z8?.....	21
CAPITULO II. ARQUITECTURA DEL Z8.....	25
Intruducción.....	25
Señales y Pins del Z8.....	27
Sincronización y Ejecución de Instrucciones..	31
Espacio de Direcciones.....	38
Puertos.....	42
Interrupciones.....	54
Contadores/Temporizadores.....	56
Puerto Serie.....	60
Registros de Control.....	62
CAPITULO III. PROGRAMACION DEL Z8.....	89
Archivo de Registros del Z8.....	92
Modos de Direcccionamiento del Z8.....	95
Conjunto de Instrucciones del Z8.....	107

CAPITULO IV. DESCRIPCION DEL SIS Z8.....	133
Elementos del SIS Z8.....	133
Instalación del SIS Z8.....	140
Operación del SIS Z8.....	143
Reglas de Operación.....	144
Descripción de los Comandos.....	148
 CAPITULO V. HARDWARE DEL SIS Z8.....	 179
Generalidades del SIS Z8.....	179
Diagramas del SIS Z8.....	186
Bases e Interfaz, Puertos 0 y 1.....	191
Memoria RAM y ROM.....	197
Puertos Serie.....	200
Opciones.....	207
Puntos de Prueba y Lógica de Trase.....	207
 CAPITULO VI. EL SCAN Z8.....	 213
Estructura y Diseño del SCAN Z8.....	213
Descripción del Monitor.....	215
Rutinas del Sistema.....	221
Comandos.....	231
Listado del SCAN Z8.....	275
 CAPITULO VII. DISEÑO DEL SIS Z8.....	 335
Propuesta de Procedimientos para Proyectos de I. E.....	337
Propuesta y Descripción Preliminar.....	343

Diseño Preliminar.....	345
Implementación del Prototipo.....	351
Documentación.....	359
CONCLUSIONES.....	361
BIBLIOGRAFIA.....	365
APENDICE A.....	367
APENDICE B.	377
Ensambladores.....	377
APENDICE C.....	397
Equipo Empleado.....	397
APENDICE D.....	399
Hojas de Datos del ZB.....	399

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 esta 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. Quizá esto 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 28 (SIS 28), no adoleciera de una justificación, es decir que el elemento de trascendencia del proyecto no sea el SIS 28 por sí solo. El SIS 28 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 28.

Tenemos la certeza, que cualquier desarrollo tecnológico es útil, si 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 bien, sea capaz de crear tecnología propia, entonces todo el aparato productivo seguirá en el retraso permanente.

El SIS ZB 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 llevarán a cabo muchos proyectos (de tesis, por ejemplo) con este enfoque, seguramente algunos de ellos lograrían su objetivo. Y esto sí sería un avance...

En cuanto al texto, se ha buscado lograr una lectura ágil del mismo, para que las ideas y conocimientos que en él se encuentran sean fácilmente 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 ZB, para hacer comprensible el resto del texto, así también para proveer una fuente con información de primera mano, para aquellos que deseen aplicar este dispositivo.

En cuanto a la aplicación de los términos técnicos en inglés, 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 consenso de una institución o comunidad ampliamente reconocida. Además de que, tristemente, son escasos los textos técnicos en español, referentes a la microcomputación y por lo tanto, los ingenieros en electrónica y computación, aprenden de textos en inglés, por lo cual se encuentran más familiarizados con los términos técnicos en este idioma.

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

En resumen, es nuestra esperanza, que el SIS ZB y el presente trabajo sean útiles como semillas para nuevas y mejores ideas, y no solamente el efímero 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.

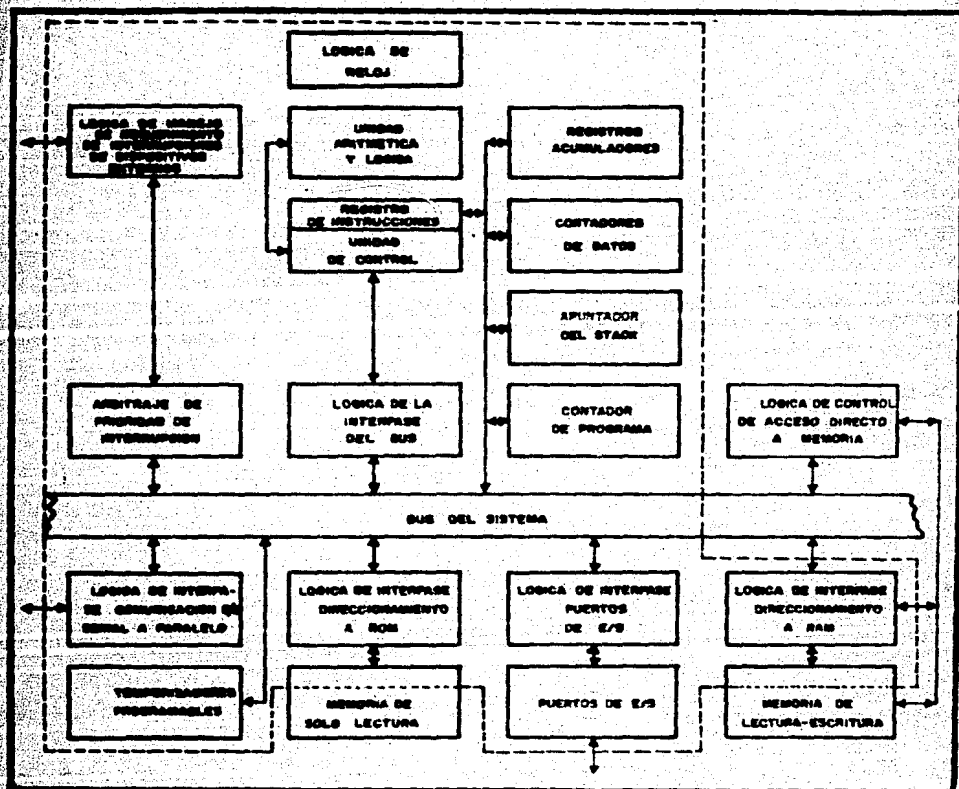
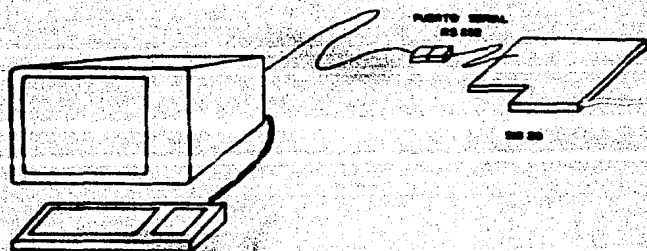


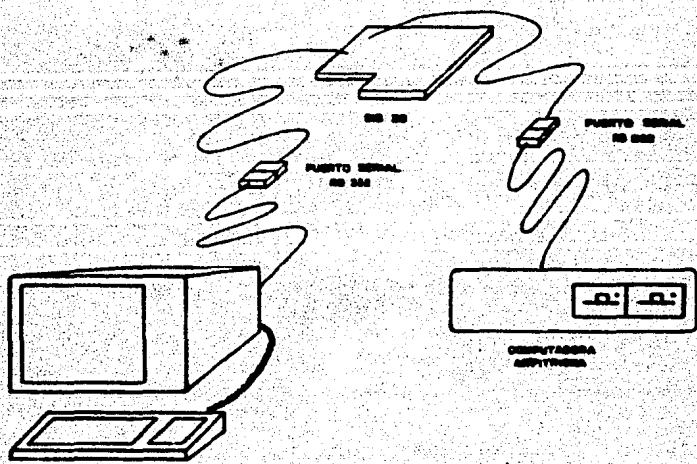
FIGURA (1.1)

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



TERMINAL

CONFIGURACION SENCILLA



TERMINAL

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 características

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 esto 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 únicamente 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 línea, 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 estándar en cualquier microcomputadora.

El SCAN Z8 consiste de 16 comandos de los cuales 12, son estándar 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 mnemónicos en un editor de líneas. El ensamblador del Z8 lee este archivo y genera el código objeto en otro archivo con formato INTEL. El comando TRN 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 alabrado.

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

CAPITULO II

ARQUITECTURA DEL Z8

En este capítulo se tratarán los aspectos de arquitectura de la microcomputadora de un solo chip Z8.

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

En el resto del capítulo se hablará detalladamente de los componentes y características 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 característica más notoria del Z8, es que sus múltiples opciones pueden configurarse y modificarse dinámicamente 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 energía que protege el archivo de registros mientras dura el problema.

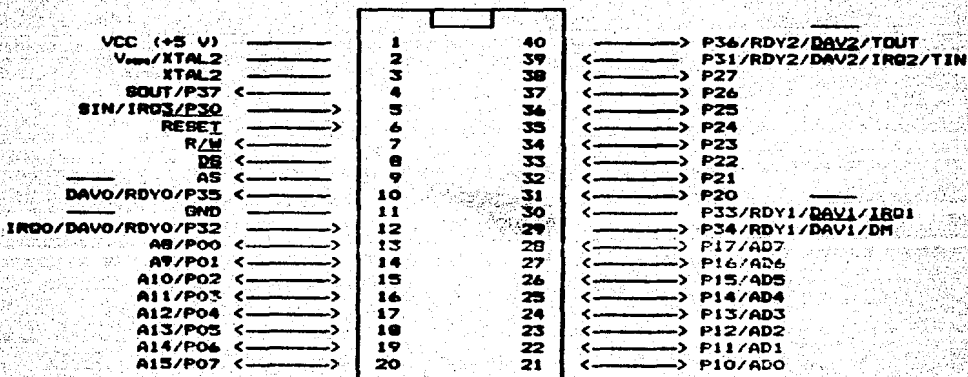
- Tres espacios de direcciones básicos: Memoria de Programa (Interna y Externa); Memoria de Datos (Externa); Archivo de Registros(Interno).
- Todas las señales son compatibles con niveles TTL.
- Las instrucciones del Z8 pueden operar sobre datos con distintas estructuras, tales como: datos de un solo bit, dígitos 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 Asíncrono (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 Propósito General.
 - b) 16 Registros de Control.
 - c) 4 Registros de Puertos de Entrada/Salida.
- 32 líneas dedicadas a Entrada/Salida, agrupadas en 4 Puertos de 8 líneas 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 Características de E/S Serie o Paralela, Sencilla o con Señales de Enlace.

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



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
AS-A15	BUS DIRECCIONES	SALIDA
RDY0/DAV0	SENALES DE ENLACE PUERTO 0	ENTRADA O SALIDA
RDY1/DAV1	SENALES DE ENLACE PUERTO 1	ENTRADA O SALIDA
RDY2/DAV2	SENALES 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
DH	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	CONEXIONES A CRISTAL	
V _{DD}	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 actúa 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 (V_{mm}), la cual energiza el archivo de registros y la lógica de RESET durante una falla en la fuente V_{cc}.

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

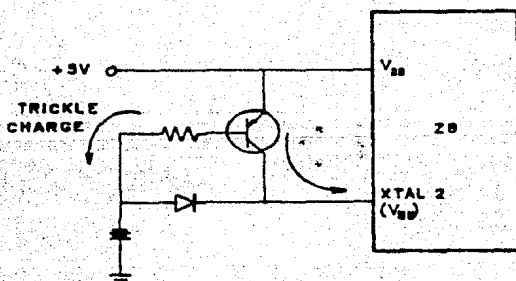


FIGURA (2.3)

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 10 periodos 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.

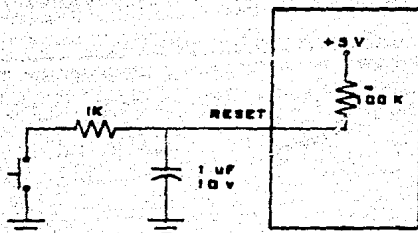


FIGURA (2.4)

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

El Z8 ejecuta las instrucciones en una serie de Ciclos de Máquina, cada uno de los cuales consta de 3 periodos 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 del Pipeline de Instrucciones; 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 período de reloj interno a cada ciclo de lectura/escritura a memoria externa. Este período 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 período del reloj del ciclo de máquina. Durante el segundo período, 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

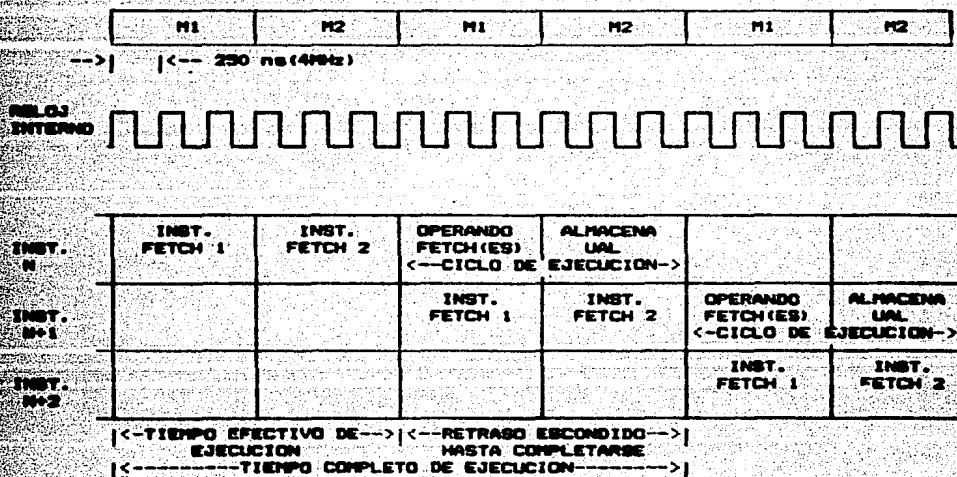


FIGURA (2.5)

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 arrancar 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.

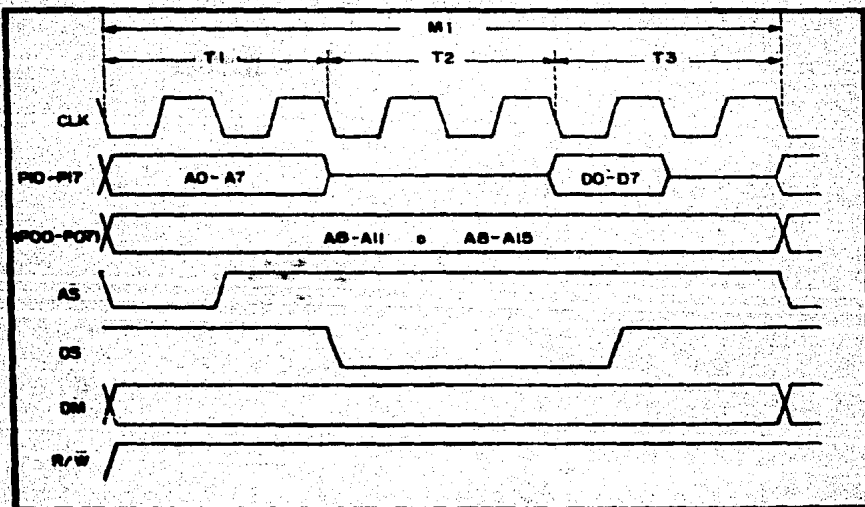


FIGURA (2.6)

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.

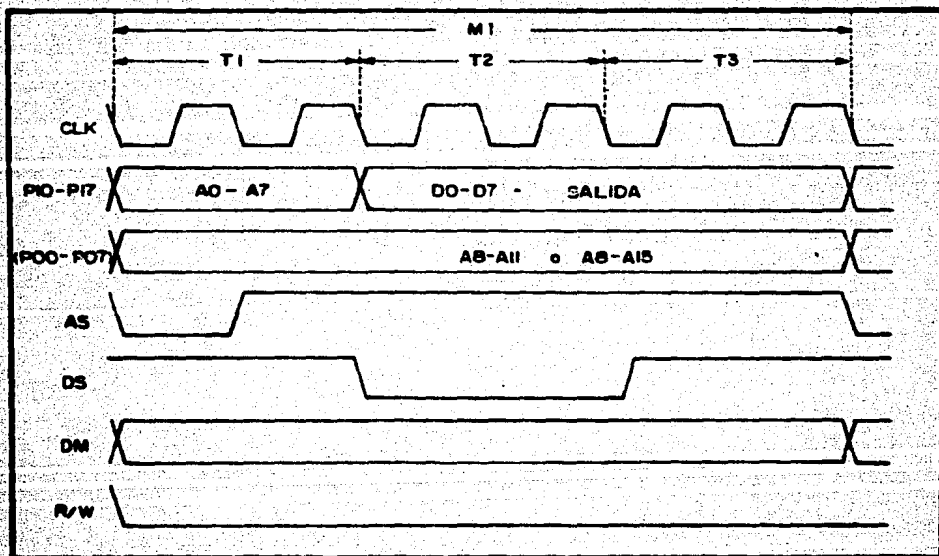


FIGURA (2.7)

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 periodos de reloj antes del pulso AS correspondiente a un ciclo de búsqueda de instrucción.

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

Si existe petición de interrupción, el Z8 necesita 7 ciclos de máquina (48 periodos 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.

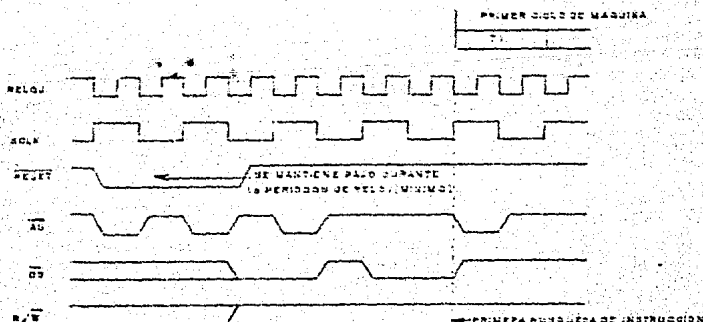
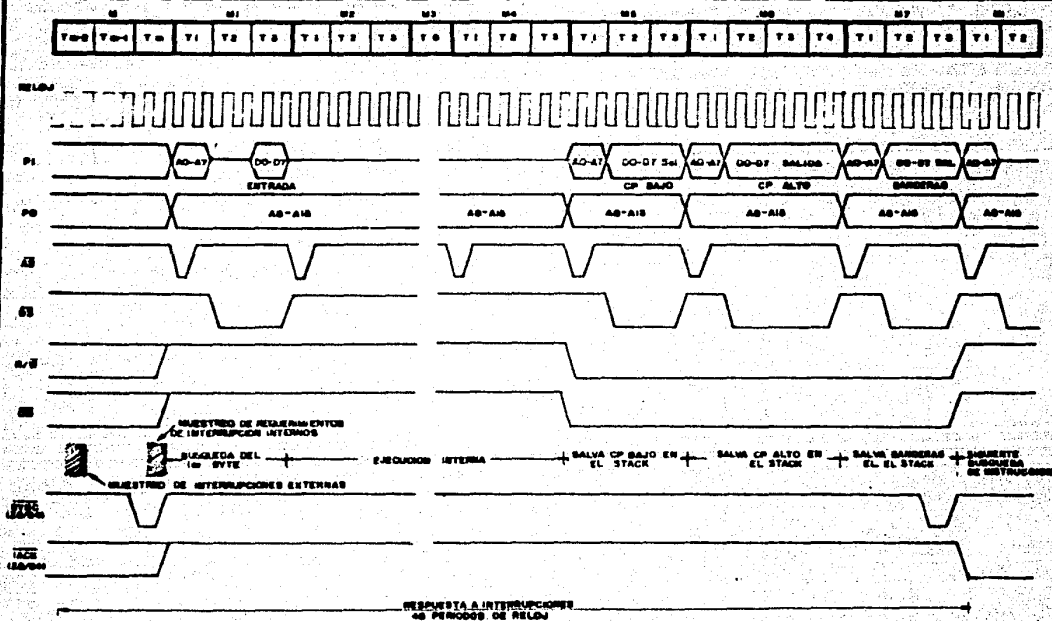


FIGURA (2.8)



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).

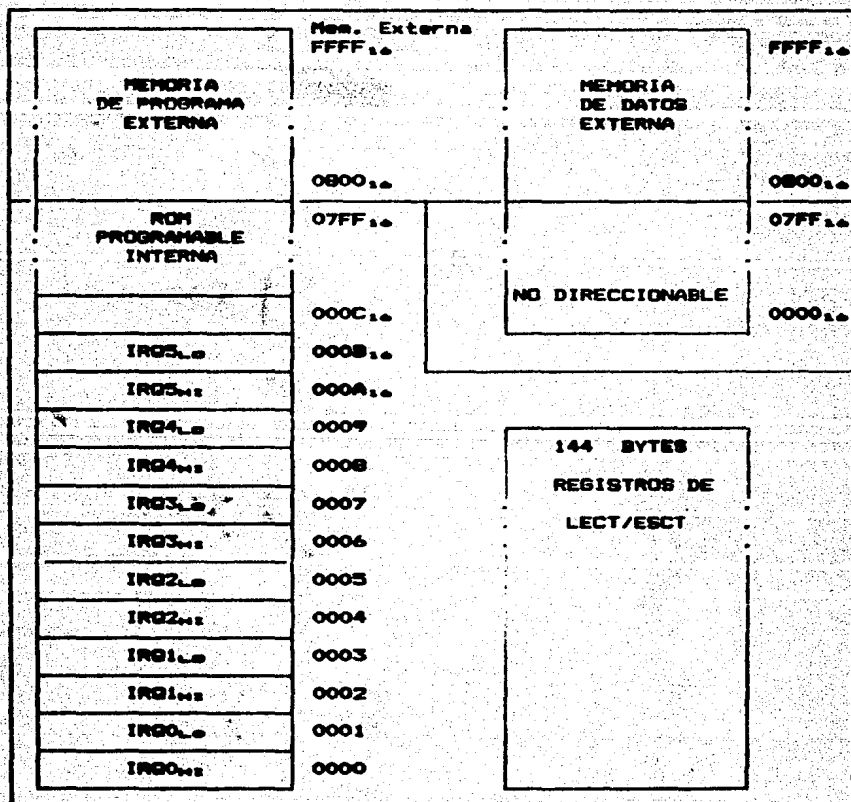


FIGURA (2.10)

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 (0CH) 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 reside en memoria de programa; y no hay forma de accesar instrucciones en memoria de datos. El Z8 está diseñado asumiendo que en ésta sólo 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 Propósito 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 Bandejas, según corresponda.

El Registro FF₁₆ apunta al byte de orden bajo de la dirección del Stack, mientras el FE₁₆ 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 índice, 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(FOH) 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 sí 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.

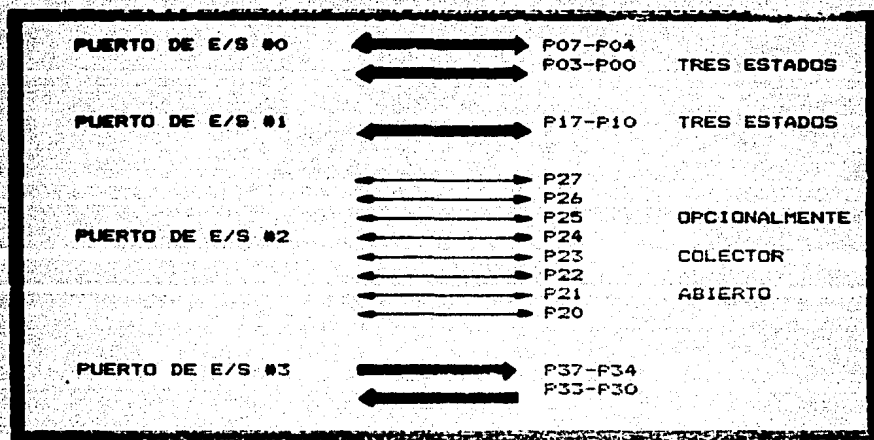


FIGURA (2.12)

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

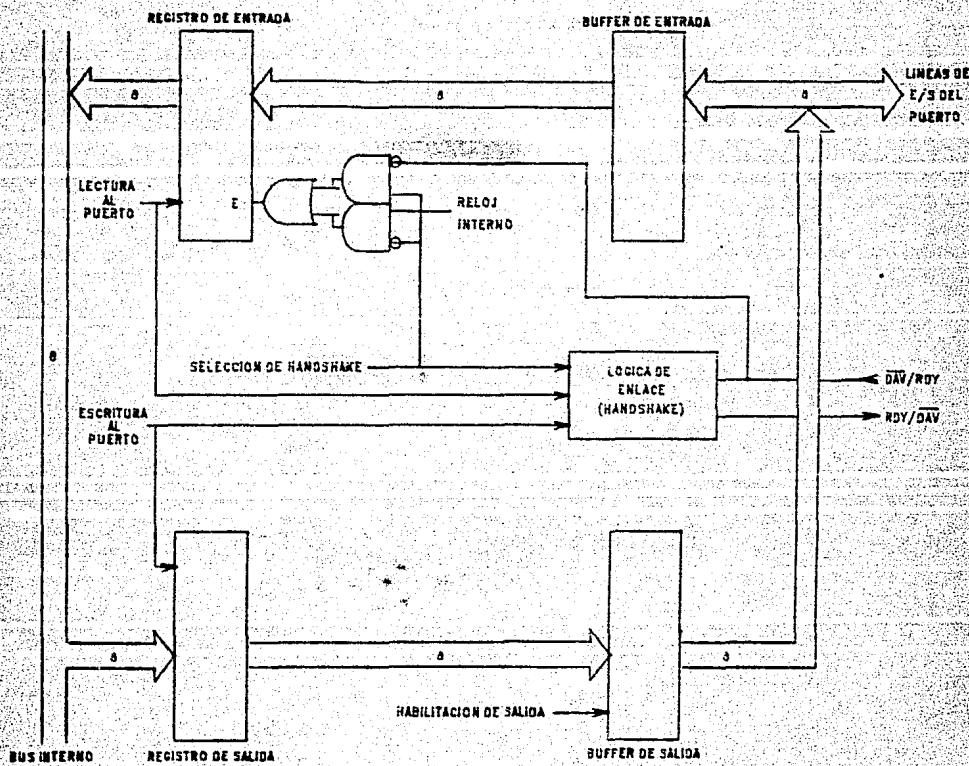


FIGURA (2.13)

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 accedido 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 IB 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 accedido 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 pines P32 y P35 del Puerto 3 trabajan como líneas de control de enlace DAVO y RDYO para entrada o RDYO y DAVO 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 POIM 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, POIM o R248.

Cuando es usado el modo byte de entrada o byte de salida, el puerto es accedido a través del registro R1. Se escribe en el puerto cuando se especifica a R1 como el registro destinatario de una instrucción y el dato es guardado en el registro de salida del puerto. El puerto es leído por especificación 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 pines del Puerto 3: P33 y P34, trabajan como DAV1 y RDY1 en modo entrada y viceversa en modo salida.

Para acceder 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 accedidas 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.

SEÑALES 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:

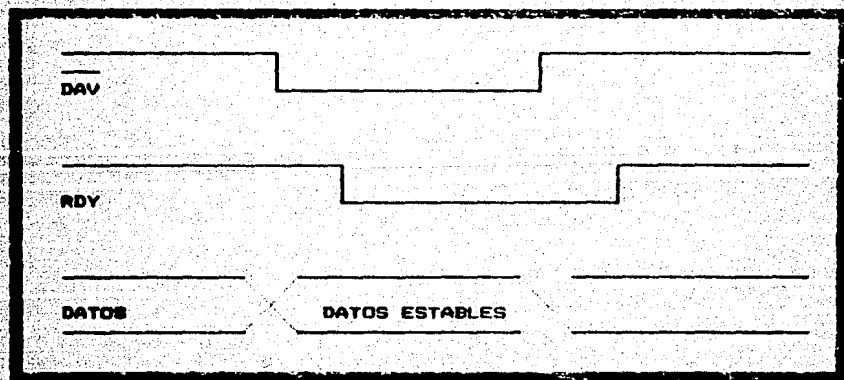


FIGURA (2.14)

PUERTOS	SITS	DIRECCION	SENALES 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).
	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 puede ser la misma para todas las líneas 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		

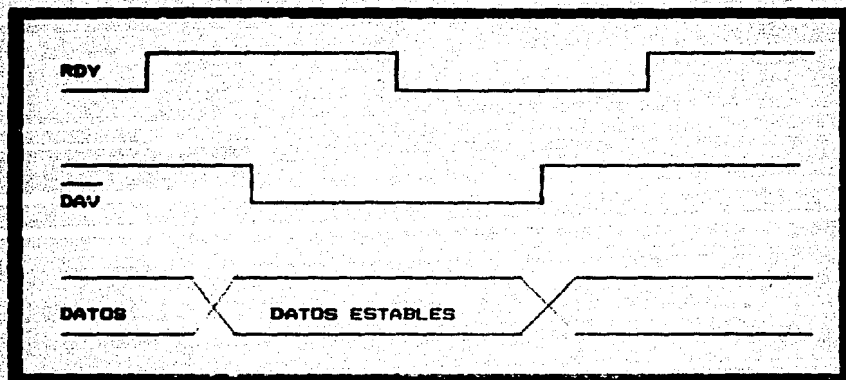
TABLA (1)

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, simultáneamente debe poner a DAV en un nivel bajo.

El nivel bajo de DAV hace que la lógica de E/S del puerto saque 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 todavía 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 e 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 están 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 salir. 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: el dato almacenado en el registro de salida. Si las salidas del Puerto 3 son usadas para

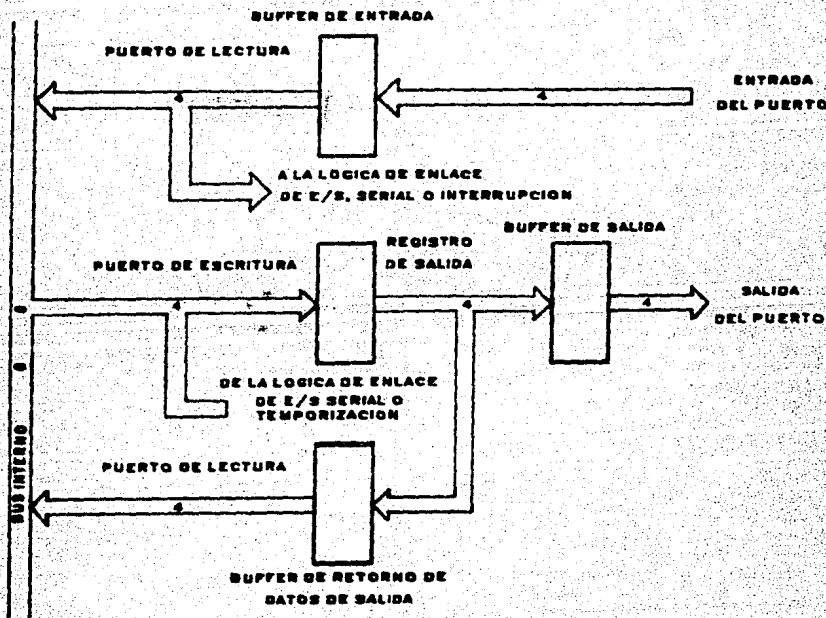


FIGURA (2.16)

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

TABLA (2)

Debe hacerse notar que las 4 líneas 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) R231 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 al Puerto 1 para accesaria, 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 líneas 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	Alt	DS y R/W
0000-07FF		000-7FF	0	Inactiva
0800-0FFF	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 así sucesivamente para buses de 12 y 16 bits como se muestra a continuación.

Dirección de Memoria (16-bits) Arbitraria

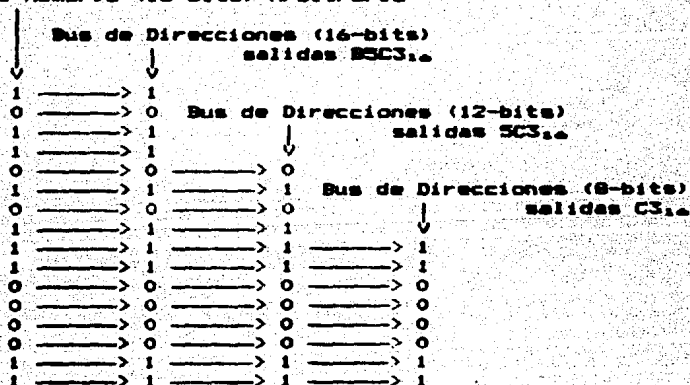


FIGURA (2.17)

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 65536 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 Mascaras (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 Mascaras 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

Interrupción	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 interrupciones 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 señales de enlace DAV2/RDY2 para el puerto 2. También sirve para una interrupción externa de los temporizadores
IRQ3	6.7	Externa	Una transición de un flanco positivo a la entrada P30. IRQ0 sirve para peticiones generales de interrupción.
		Interna	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 carácter ha sido ensamblado y deberá ser leído.

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

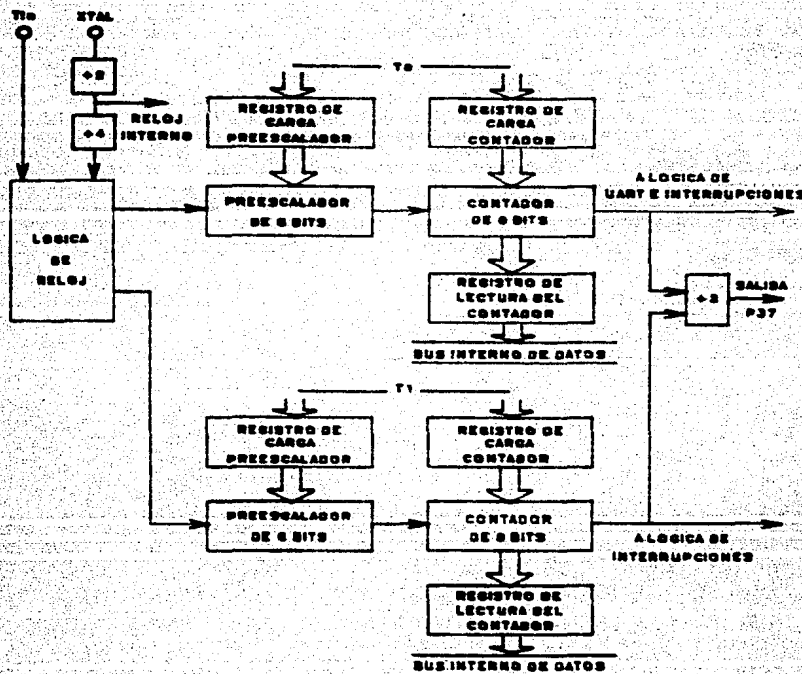


FIGURA (2.18)

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 exista 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.

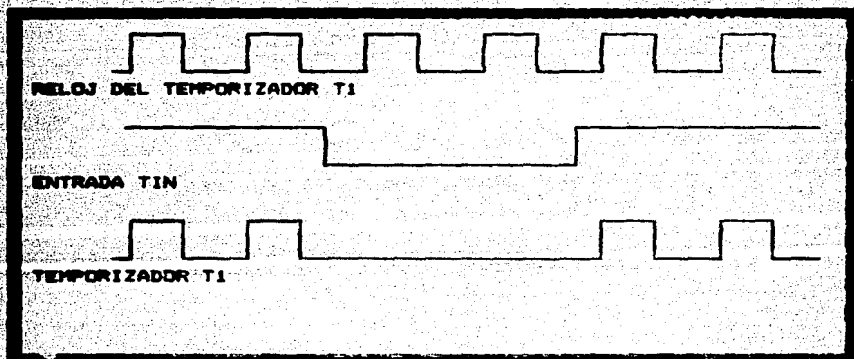


FIGURA (2.19)

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 0 Reloj =A/B (KHz)	C Intervalo del Prescalador = =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 líneas del Puerto 3 P30 y P37 pueden ser programadas como líneas de E/S Serie para operación Receptor/Transmisor Asíncrono 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 carácter de 8 bits y transferido al buffer receptor. El registro R240 es en realidad 2 registros en 1, cuando se escribe en él 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 paridad es seleccionada, el octavo bit es la paridad par. Entre caracteres, la salida de P37 es alta, para mantener la condición de marca.

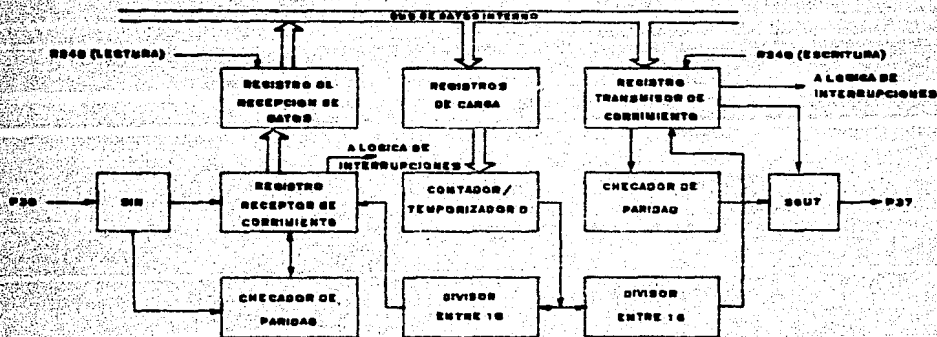
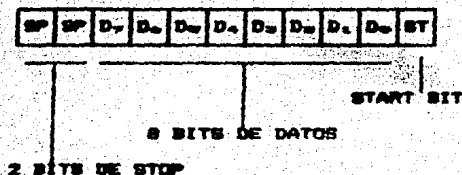


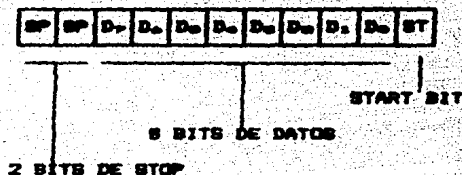
FIGURA (2.20)

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.

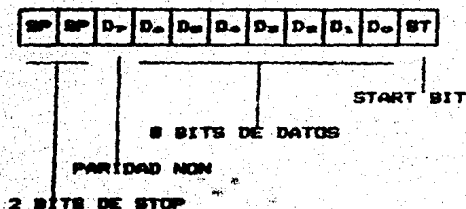
TRANSMITE DATOS (Sin Paridad)



RECIBE DATOS (Sin Paridad)



TRANSMITE DATOS (Con Paridad)



RECIBE DATOS (Con Paridad)

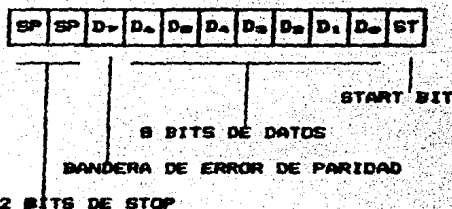


FIGURA (2.21)

Una petición de interrupción IRQ3 es generada cuando un carácter es transferido en el bus receptor. Deben tomarse precauciones de atender esta interrupción, ya que este registro no está protegido contra sobrescrituras. Un carácter 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

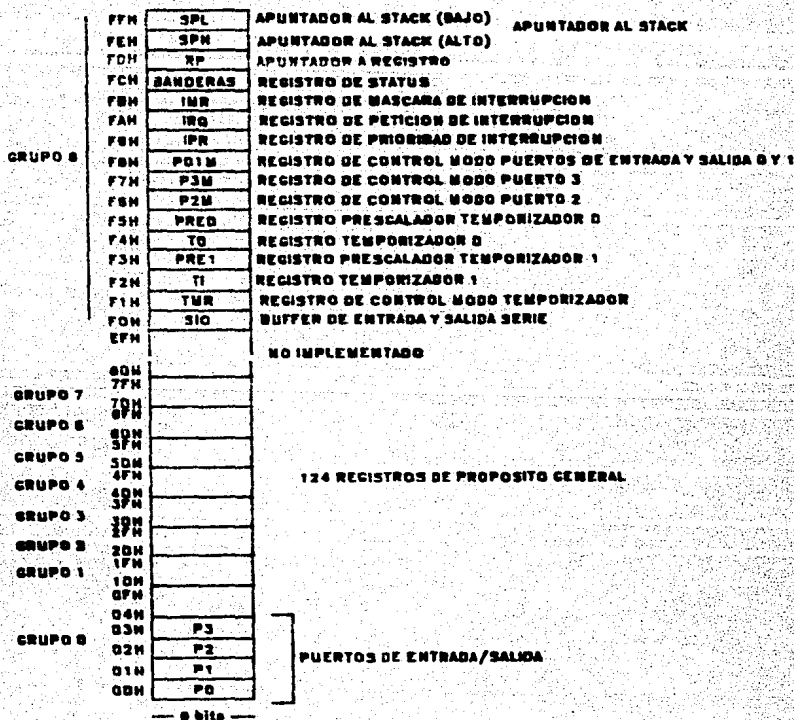


FIGURA (2.22)

SIO R240 (FO)

REGISTRO DE ENTRADA/SALIDA SERIE

(Lectura/Escritura)



Lectura de R240 = Recepción de Datos

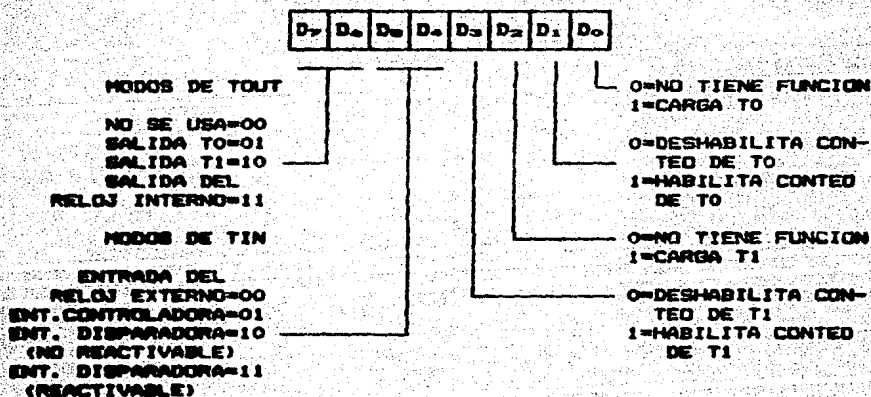
Escritura de R240 = Transmisión de Datos

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 leído se recupera el carácter en el buffer receptor. Cuando se escribe, el carácter 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.

THR 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 simultáneamente.

(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.

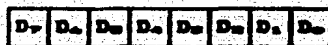
(D4,D5) Modos de Entrada de Temporización externa. D4 y D5 son codificados para definir 4 modos de operación de la Entrada de Temporización (Tin). P34 debe ser previamente definido como Entrada de Temporización Externa (R243,D1). La tabla siguiente ilustra la codificación de D4 y D5.

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, saltándose 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 Reactivable.	T1 es cargado y sincronizado al 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 al reloj interno solo después de que una transición Alto-Bajo ocurre en Tin. Cuando pulsos subsiguientes 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)

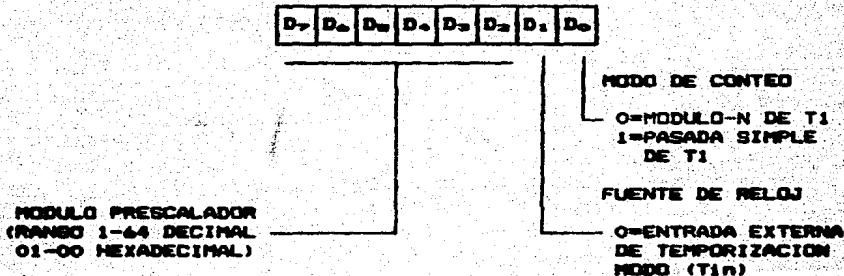


VALOR INICIAL DE T1 (CUANDO SE ESCRIBE)
(RANGO 1-256 DECIMAL 01-00 HEXADECIMAL)
VALOR ACTUAL DE T1 (CUANDO SE LEE)

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.

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 comando 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 comando 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) está 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 contador para los subsiguientes 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 para 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.

TO R244 (F4)

REGISTRO DEL CONTADOR TEMPORIZADOR TO

(Lectura/Escritura)



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

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

PREGO R245 (FS)

REGISTRO DE CARGA DEL PREESCALADOR DE TO

(Lectura/Escritura)



MODULO PRESCALADOR
(RANGO: 1-64 DEC.
01-00 HEXADECIMAL)

MODULO CONTADOR
0=TO MODULO N
1=TI PAGO SIMPLE

NO SE USA

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

P2M R246 (F4)

REGISTRO DE MODOS DEL PUERTO 2

(Solo Escritura)



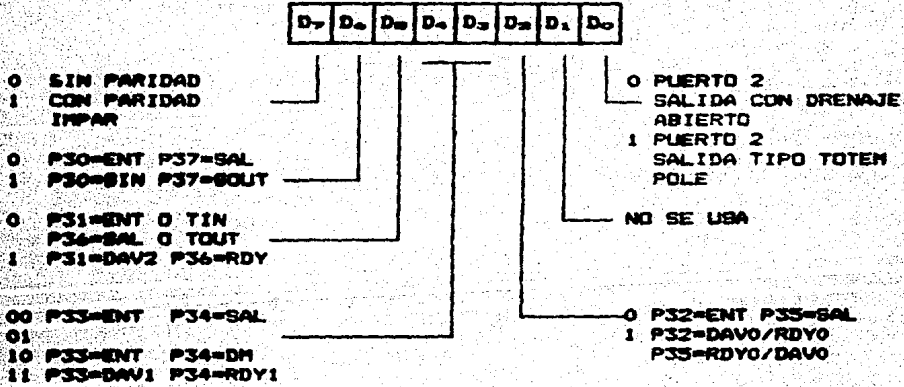
P20-P21 DEFINICION DE E/S
0 DEFINE AL BIT COMO SALIDA
1 DEFINE AL BIT COMO ENTRADA

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, 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.

PSM R247 (F7)

REGISTRO DE MODOS DEL PUERTO 3

(Solo Escritura)



El Registro Modos del Puerto 3 especifica cuales líneas 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 líneas 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=DM
1	1	P33=DAV1/RDY1	P34=RDY1/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 Entrada/Salida, normalmente reflejan el contenido de los bits 1 y 4 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=DAV2/RDY2	P36=RDY2/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 líneas P30 y P37 del Puerto 3 funcionan como E/S o como interfase Receptor/Transmisor Serie.

D6 Función

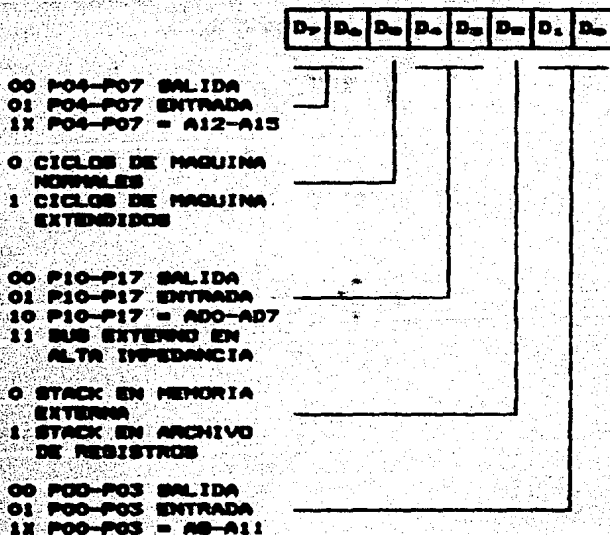
0	P30=Entrada	P37=Salida
1	P30=Entrada Serie	P37=Salida Serie

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

PO1M R248 (FB)

REGISTRO DE MODO DE LOS PUERTOS O Y I

(Solo Escritura)



Este registro configura los puertos O y I, 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. Nótese 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 (R253). 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 R253.

(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 siguientes:

D7	D4	D3	D1	Líneas 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

(D6) Temporización de Memoria Extendida. Cuando es prendido este bit, extiende el ciclo de memoria por un periodo 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 0. 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 AS-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)

REGISTRO DE PRIORIDAD DE INTERRUPCIONES

(Solo Escritura)

NO SE USA

GRUPO A

0 =IRQ5>IRQ3

1 =IRQ3>IRQ5

GRUPO B

0 =IRQ2>IRQ0

1 =IRQ0>IRQ2

GRUPO C

0 =IRQ1>IRQ4

1 =IRQ4>IRQ1

000 INDEFINIDO

001 C > A > B

010 A > B > C

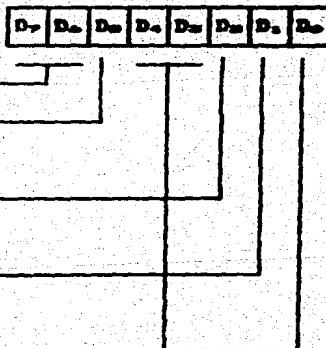
011 A > C > B

100 B > C > A

101 C > B > A

110 B > A > C

111 INDEFINIDO



Este registro determina el nivel de prioridad de las 4 interrupciones, codificando 48 diferentes ordenes para resolver casos de petición de interrupción simultáneas. Los 4 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(B1/P03) e IRQ5(T1); el grupo B contiene IRQ0(P32) e IRQ2(P31), y el grupo C IRQ1(P33) e IRQ4(B0/T0).

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 D3 definen la prioridad de los miembros individuales dentro de sus grupos.

GRUPO	BIT	PRIORIDAD	
		MAYOR	MEJOR
C	D1=0	IRQ1	IRQ4
	1	IRQ4	IRQ1
B	D2=0	IRQ2	IRQ0
	1	IRQ0	IRQ2
A	D3=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 la siguiente manera.

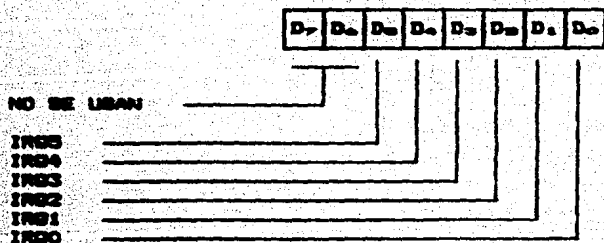
PATRON DE BITS			PRIORIDAD DEL GRUPO	
D4	D3	D0	DE MAYOR	A MENOR
0	0	0		indefinido
0	0	1		C A B
0	1	0		A B C
0	1	1		A C B
1	0	0		B C A
1	0	1		C B A
1	1	0		B A C
1	1	1		indefinido

Los bits D6 y D7 no se usan.

IRQ R250 (FA)

REGISTRO DE PETICIONES DE INTERRUPCION

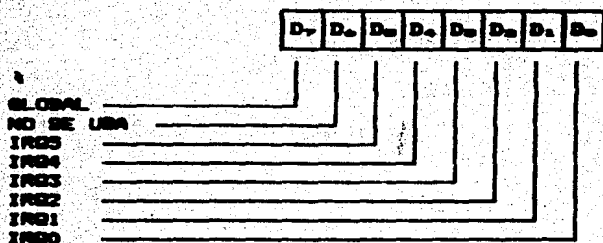
(Lectura/Escritura)



Este registro guarda las peticiones de interrupción. Puesto que este 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-D3 son asignados a las peticiones de interrupción IRQ0-IRQ3 respectivamente. D6 y D7 no tienen función.

REGISTRO DE MASCARAS DE INTERRUPCION

(Lectura/Escritura)



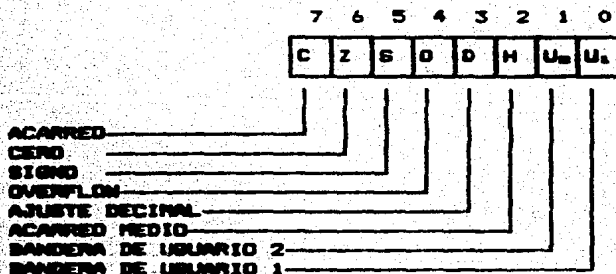
Este registro habilita o deshabilita individual o globalmente las peticiones de las 6 interrupciones. Cuando los bits D0-D5 son prendidos, las peticiones correspondientes son habilitadas. D7 es el habilitador maestro y debe prenderse después de que alguna de las interrupciones individuales sea reconocida. 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 ZB. El significado asignado a cada bit se explica a continuación:

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

(D6) Bandera de Cero. Su valor es uno cuando el resultado de una operación aritmética 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 aritmética o lógica.

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

(D2) Bandera de Acarreo Medio (Half Carry). Representa un acarreo del bit 3 al bit 4, después de una operación aritmética o lógica.

(B3) Bandera de Ajuste Decimal. Esta bandera es usada por el 28 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 aritmética fue una resta o una suma binarias. Si la última operación aritmética fue una resta binaria, esta bandera toma un valor de 1 y el valor de cero si fue una adición binaria. La instrucción de ajuste decimal, posteriormente, interrra 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.

REGISTRO APUNTADOR

(Lectura/Escritura)



NISBLE APUNTADOR

NO SE USAN

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 APUNTAORES 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 R24B 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/C 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	PREESCALADOR T0. MODO UNA PASADA, RELOJ INTERNO	X	X	X	X	X	X	X	0
F4	TEMPORIZADOR T0	X	X	X	X	A	A	A	A
F3	PREESCALADOR 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 capítulo 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 capítulo 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 capítulo se realizará una descripción completa de los modos de direccionamiento 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 capítulo 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 capítulo, se dispondrá de un Apéndice destinado a programas ensambladores, en donde se explicarán el manejo de éstos, 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 capítulo 2, y que nos serán útiles en el presente.

Una de las características 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 específicas 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

PROGRAMA

MEMORIA EXTERNA

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 cual es posible si ésta se expande externamente con hasta 63488 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 acceder hasta 62 Kbytes de memoria externa de datos, la cual comienza en la localidad 2048, como se aprecia en la Figura (3.1):

65535

2048

2047

0

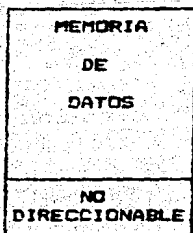


FIGURA (3.1)

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

Hasta aquí, 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 accederse 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 no compartir el mismo espacio de direcciones. Cuando lo comparten, todas las instrucciones que accedan memoria de datos externa accederán 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 específico 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 INTEL, este tiene 14 registros de propósito general de 8 bits y 3 registros de propósito específico, como se aprecia en la Figura (3.2).

De los registros de propósito 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 propósito general, para el almacenamiento de información.

A	BANDERAS	A'	BANDERAS
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'
I	R	REGISTROS DE PROPOS ESPECIFICO	
IX			
IV			
SP			

REGISTROS DE
PROPOSITO
GENERAL

FIGURA (3.2)

Estos registros son agrupados en un espacio interno llamado ARCHIVO DE REGISTROS, el cual 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 propósito 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

	7FH	SPL	APUNTAOR AL STACK (BAJO)	APUNTAOR AL STACK
	FEN	SPH	APUNTAOR AL STACK (ALTO)	
	FDN	BP	APUNTAOR A REGISTRO	
	FCN	BANDERAS	REGISTRO DE STATUS	
	FDN	IMR	REGISTRO DE MASCARA DE INTERRUPCION	
	FAN	IRS	REGISTRO DE PETICION DE INTERRUPCION	
	FDN	IPR	REGISTRO DE PRIORIDAD DE INTERRUPCION	
GRUPO 8	FDN	PCM	REGISTRO DE CONTROL MODO PUERTOS DE ENTRADA Y SALIDA 0 Y 1	
	F7N	PCM	REGISTRO DE CONTROL MODO PUERTO 3	
	F6N	PCM	REGISTRO DE CONTROL MODO PUERTO 2	
	F5N	PRE0	REGISTRO PRESCALADOR TEMPORIZADOR 0	
	F4N	TD	REGISTRO TEMPORIZADOR 0	
	F3N	PRE1	REGISTRO PRESCALADOR TEMPORIZADOR 1	
	F2N	TI	REGISTRO TEMPORIZADOR 1	
	F1N	TMR	REGISTRO DE CONTROL MODO TEMPORIZADOR	
	F0N	SIO	BUFFER DE ENTRADA Y SALIDA SERIE	
	EFN		NO IMPLEMENTADO	
	80N			
	7FN			
GRUPO 7	7DN			
	6FN			
GRUPO 6	6DN			
	5FN			
GRUPO 5	5DN			
	4FN			
GRUPO 4	4DN			
	3FN			
GRUPO 3	3DN			
	2FN			
GRUPO 2	2DN			
	1FN			
GRUPO 1	1DN			
	0FN			
	84N			
	83N	P3		
GRUPO 0	82N	P2		
	81N	P1		
	80N	P0		

124 REGISTROS DE PROPOSITO GENERAL

PUERTOS DE ENTRADA/SALIDA

— 8 bits —

FIGURA (3.3)

DECIMAL**HEXADECIMAL**

255	REGISTROS DE CONTROL	FF
240		F0
	NO IMPLEMENTADO	
128	GRUPO 7	80
112	GRUPO 6	70
96	GRUPO 5	60
80	GRUPO 4	50
64	GRUPO 3	40
48	GRUPO 2	30
32	GRUPO 1	20
16	GRUPO 0	10
0		00

FIGURA (3.4)**MODOS DE DIRECCIONAMIENTO DEL Z8**

Un operando es el dato sobre el cual 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 pueda 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 espaciales, 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
RR	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	RRn donde n = 0 - 15.
IR	Registro Indirecto o Registro de Trabajo	RRreg, donde reg es definido antes o RRn definido antes.
IRR	Registro Indirecto de Trabajo	RRRp, donde p = 0, 2, 4, 6...14.
IRR	Registro Indirecto Par o Registro de Trabajo Par	RRreg donde reg es un número par en el rango definido antes, o una variable cuya dirección sea par o RRRp 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 el rango +127 - 128 bytes desde el inicio de la siguiente instrucción.
IM	Inmediato	#dato, donde dato es una expresión.

cada registro puede ser utilizado como un acumulador, éstos pueden ser accedidos 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 propósito 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

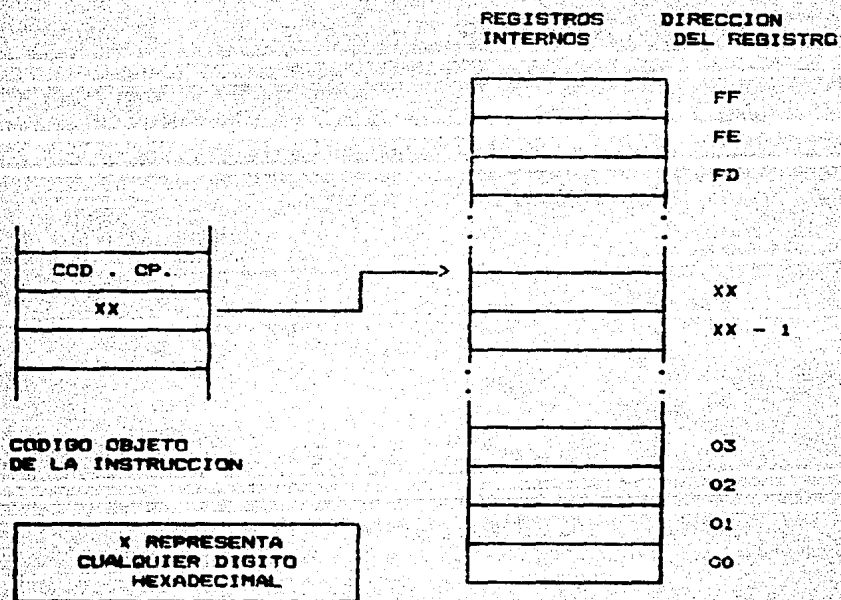
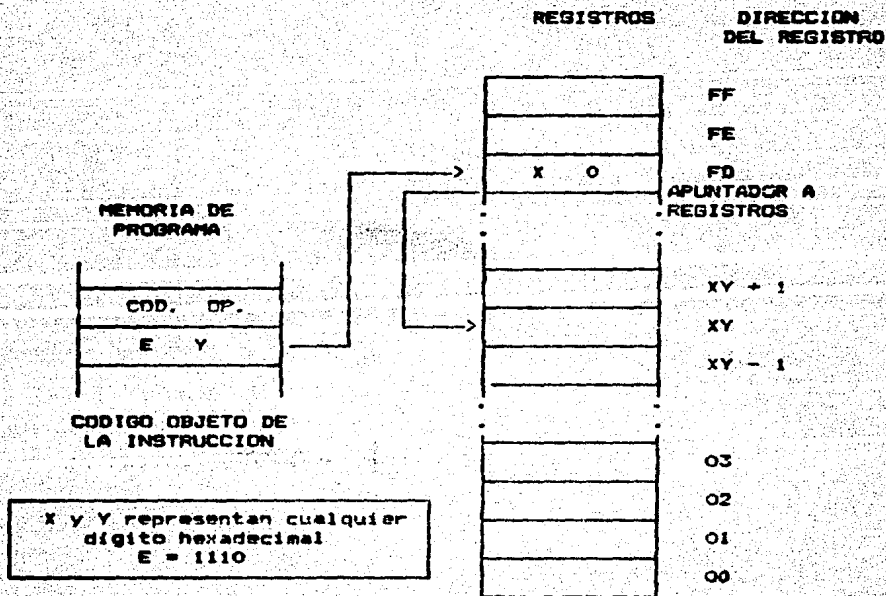


FIGURA (3.5)



-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 AP 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 20h hasta la FFh 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 contar varios registros totalmente diferentes y, con esto, agilizar el proceso de programación y 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 suada al código objeto proporcionado por la instrucción, dará como resultado la dirección del registro que se quiere acceder.

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

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

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

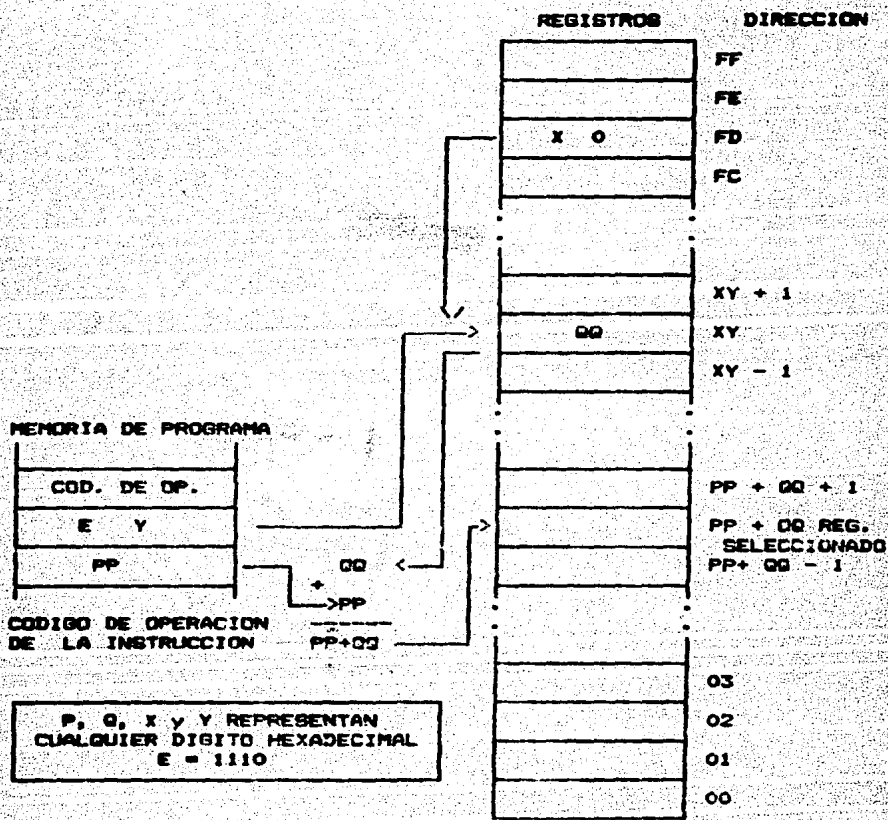


FIGURA (3.7)

- DIRECCIONAMIENTO INDIRECTO

Los registros del Z8, también pueden direccionarse indirectamente. De esta forma, la información no está contenida directamente en el registro accedido, 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:

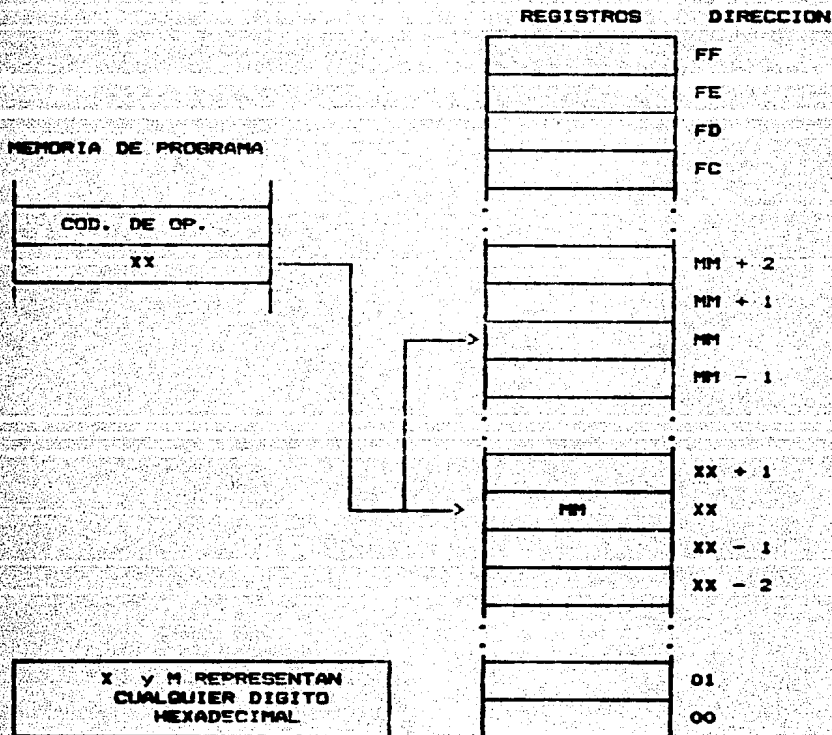


FIGURA (3.8)

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

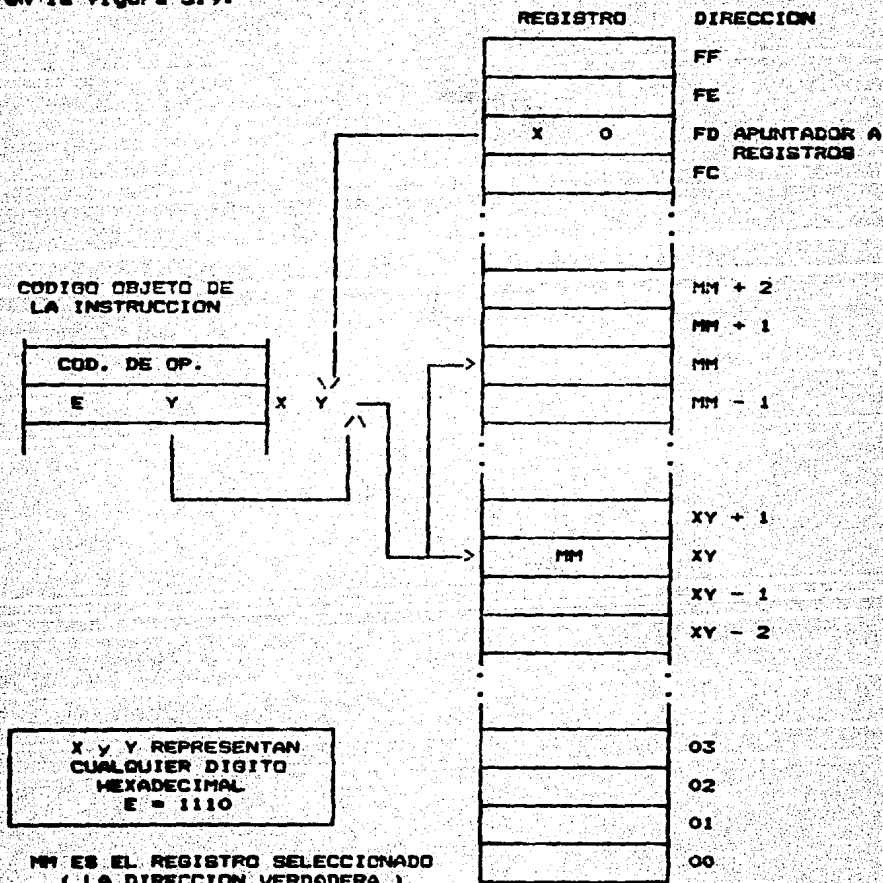


FIGURA (3.9)

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:

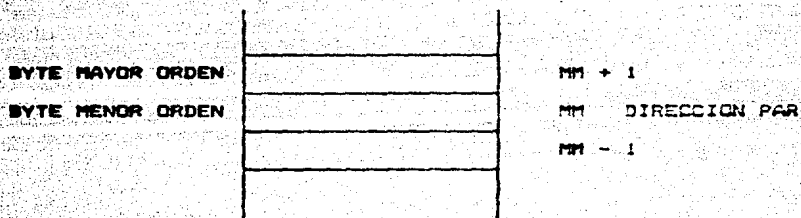


FIGURA (3.10)

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 acceder 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 07FF₁₆ o menor, entonces una de las localidades de memoria de programa interna será seleccionada.

Si el valor de PPQQ es mayor a 0800₁₆, 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 acceder localidades distintas. Para distinguir entre ambas localidades la señal de control DN es baja durante la ejecución de instrucciones que específicamente accedan memoria de datos externa y se mantiene alta durante la ejecución de cualquiera otras instrucciones.

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 endos de direccionamiento que puede utilizarse en cada instrucción.

Después de la ejecución de una instrucción, el registro de estado FCW es modificado, puesto que cada uno de sus bits represente 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 estas 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 FC₁₆.

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 específica realice.

**TABLA DE MEMONICOS , 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	$S + 0$	MEJOR QUE
2	LE	0010	$Z + (S + 0) = 1$	MEJOR QUE O IGUAL
3	ULE	0011	$C + Z = 1$	MEJOR QUE SIN SIG- NO
4	OV	0100	$O = 1$	OVERFLOW
5	NI	0101	$S = 1$	MEJOR
6	Z	0110	$Z = 1$	CERO
7	ULT	0111	$C = 1$	MEJOR QUE SIN SIG- NO
8		1000	NINGUNO	SIEMPRE VERDADERO
9	GE	1001	$S + 0 = 0$	MAJOR QUE O IGUAL
10	GT	1010	$Z + (S + 0) = 0$	MAJOR QUE
11	UGT	1011	$C + Z = 0$	MAJOR QUE SIN SIG- NO
12	NOV	1100	$O = 0$	NO OVERFLOW
13	PL	1101	$S = 0$	POSITIVO
14	NZ	1110	$Z = 0$	NO CERO
15	NC	1111	$C = 0$	SIN ACARRED

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 programas 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
DECN	dst	DECREMENTA UN REGISTRO PAR O PALABRA
INC	dst	INCREMENTA UN REGISTRO DE TRABAJO
INCN	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)	NOMBRE
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 SUBROUTINA
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 SUBROUTINA

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	dat	GUARDA EL VALOR SELECCIONADO EN EL APUNTADEOR 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 78.

Estas instrucciones son las siguientes:

INSTRUCCION	OPERANDO(S)	NOMBRE
RL	dst	ROTA A LA IZQUIERA UN BIT
RLC	dst	ROTA A LA IZQUIERA 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	ar , arr	TRANSFERENCIA MEMORIA DE PROGRAMA A REGISTRO
LDCI	arr, ar	TRANSFERENCIA REGISTRO A MEMORIA DE PROGRAMA
LDEI	ar , arr	TRANSFERENCIA MEMORIA DE DATOS A REGISTRO
LDEI	arr, ar	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 28 no existen bits específicos 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 LOGIC

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 3R
02 r ₁ ,r ₂	ADD r ₁ ,r ₂
03 r1r	ADD r,3r
04 R _n R _v	ADD R _n ,R _v
05 R IR	ADD R,3R
06 R IM	ADD R,IM
07 IR IM	ADD 3R,IM
08 R	LD r0,R
09 R	LD R,r0
0A RA	DJNZ r0,Dircc.
0B RA	JR cc0,Dircc.
0C IM	LD r0,IM
0D DA ₁ : DA ₀	JP cc0,Dircc.
0E	INC r0
0F	-----
10 R	RLC R
11 IR	RLC 3R
12 r ₁ ,r ₂	ADC r ₁ ,r ₂
13 r1r	ADC r,3r
14 R _n R _v	ADC R _n ,R _v
15 R IR	ADC R,3R
16 R IM	ADC R,IM
17 IR IM	ADC 3R,IM
18 R	LD r1,R
19 R	LD R,r1
1A RA	DJNZ r1,Dircc.
1B RA	JR cc1,Dircc.
1C IM	LD r1,IM
1D DA ₁ : DA ₀	JP cc1,Dircc.
1E	INC r1
1F	-----

COD. DE OPERACION HEXADECIMAL	INSTRUCCION
40 R	DA R
41 IR	DA 3R
42 r ₁ ,r ₂	OR r ₁ ,r ₂
43 r1r	OR r,3r
44 R _n R _v	OR R _n ,R _v
45 R IR	OR R,3R
46 R IM	OR R,IM
47 IR IM	OR 3R,IM
48 R	LD r4,R
49 R	LD R,r4
4A RA	DJNZ r4,Dircc
4B RA	JR cc4,Dircc
4C IM	LD r4,IM
4D DA ₁ : DA ₀	JP cc4,Dircc
4E	INC r4
4F	-----
50 R	POP R
51 IR	POP 3R
52 r ₁ ,r ₂	AND r ₁ ,r ₂
53 r1r	AND r,3r
54 R _n R _v	AND R _n ,R _v
55 R IR	AND R,3R
56 R IM	AND R,IM
57 IR IM	AND 3R,IM
58 R	LD r5,R
59 R	LD R,r5
5A RA	DJNZ r5,Dircc
5B RA	JR cc5,Dircc
5C IM	LD r5,IM
5D DA ₁ : DA ₀	JP cc5,Dircc
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,r _v	SUB r,r _v
23 r,r	SUB r,@r
24 R, R _v	SUB R,R _v
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,Dirac.
2B RA	JR cc2,Dirac.
2C IM	LD r2,IM
2D DA _h , DA _o	JP cc2,Dirac.
2E	INC r2
2F	=====
30 IRR	JP @RR
31 IM	BRP IM
32 r,r _v	SBC r,r _v
33 r,r	SBC r,@r
34 R, R _v	SBC R,R _v
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,Dirac.
3B RA	JR cc3,Dirac.
3C IM	LD r3,IM
3D DA _h , DA _o	JP cc3,Dirac.
3E	INC r3
3F	=====

COD. DE OPERACION HEXADECIMAL	INSTRUCCION
60 R	COM R
61 IR	COM @R
62 r,r _v	TCH r,r _v
63 r,r	TCH r,@r
64 R, R _v	TCH R,R _v
65 R IR	TCH R,@R
66 R IM	TCH R,IM
67 IR IM	TCH @R,IM
68 R	LD r6,R
69 R	LD R,r6
6A RA	DJNZ r6,Dirac
6B RA	JR cc6,Dirac
6C IM	LD r6,IM
6D DA _h , DA _o	JP cc6,Dirac
6E	INC r6
6F	=====
70 R	PUSH R
71 IR	PUSH @R
72 r,r _v	TM r,r _v
73 r,r	TM r,@r
74 R, R _v	TM R,R _v
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,Dirac
7B RA	JR cc7,Dirac
7C IM	LD r7,IM
7D DA _h , DA _o	JP cc7,Dirac
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 IrIrr	LDEI @r,@rr
84	---
85	---
86	---
87	---
88 R	LD r8,R
89 R	LD R,r8
8A RA	DJNZ r8,Dircc.
8B RA	JR cc8,Dircc.
8C IM	LD r8,IM
8D DAH: DALo	JP cc8,Dircc.
8E	INC r8
8F	DI
90 R	RL R
91 IR	RL @R
92 Ir rr	LDE @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,Dircc.
9B RA	JR cc9,Dircc.
9C IM	LD r9,IM
9D DAH: DALo	JP cc9,Dircc.
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,Dircc
CB RA	JR cc12,Dircc
CC IM	LD r12,IM
CD DAH: DALo	JP cc12,Dircc
CE	INC r12
CF	RCF
D0 R	SRA R
D1 IR	SRA @R
D2 rIrr	LDC r,@rr
D3 Irrrr	LDCI @rr,r
D4 IRR	CALL @RR
D5	---
D6 DAH: DALo	CALL Dircc.
D7 rx r	LD R,x,r
DB R	LD r13,R
D9 R	LD R,r13
DA RA	DJNZ r13,Dircc
DB RA	JR cc13,Dircc
DC IM	LD r13,IM
DD DAH: DALo	JP cc13,Dircc
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 IR	INCW $\overline{RR}$
A2 r_1, r_2	CP r_1, r_2
A3 $r, \overline{r}$	CP $r, \overline{r}$
A4 r, R, R_v	CP R, R_v
A5 R IR	CP $R, \overline{IR}$
A6 R IM	CP R, IM
A7 IR IM	CP $\overline{IR}, IM$
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 $DA_n: DA_{Lo}$	JP $cc10, Direc.$
AE	INC $r10$
AF	RET
B0 R	CLR R
B1 IR	CLR $\overline{IR}$
B2 r_1, r_2	XOR r_1, r_2
B3 $r, \overline{r}$	XOR $r, \overline{r}$
B4 r, R, R_v	XOR R, R_v
B5 R IR	XOR $R, \overline{IR}$
B6 R IM	XOR R, IM
B7 IR IM	XOR $\overline{IR}, IM$
B8 R	LD $r11, R$
B9 R	LD $R, r11$
BA RA	DJNZ $r11, Direc.$
BB RA	JR $cc11, Direc.$
BC IM	LD $r11, IM$
BD $DA_n: DA_{Lo}$	JP $cc11, Direc.$
BE	INC $r11$
BF	IRET

COD. DE OPERACION HEXADECIMAL	INSTRUCCION
E0 R	RR R
E1 IR	RR $\overline{IR}$
E2	-----
E3 $r, \overline{r}$	LD $r, \overline{r}$
E4 R, R_v	LD R, R_v
E5 $R, \overline{IR}_v$	LD $R, \overline{IR}_v$
E6 R IM	LD R, IM
E7 IR IM	LD $\overline{IR}, IM$
E8 R	LD $r14, R$
E9 RA	LD $R, r14$
EA RA	DJRZ $r14, Dir.$
EB IM	LD $r14, IM$
EC $DA_n: DA_{Lo}$	JR $cc14, Direc.$
ED	JP $cc14, Direc.$
EE	INC $r14$
EF	CCF
F0 R	SWAP R
F1 IR	SWAP $\overline{IR}$
F2	-----
F3 $r, \overline{r}$	LD $\overline{r}, r$
F4 R, R_v	LD $\overline{R}_v, R_v$
F5	-----
F6	-----
F7	-----
F8 R	LD $r15, R$
F9 R	LD $R, r15$
FA RA	DJNZ $r15, Dir.$
FB RA	JR $cc15, Direc.$
FC IM	LD $r15, IM$
FD $DA_n: DA_{Lo}$	JP $cc15, Direc.$
FE	INC $r15$
FF	NOP

TIPO	HEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS						OPERACION DESARROLLADA
						C	Z	S	O	D	H	
REFERENCIALA MEMORIA PRIMARIA	LDC	0rr,r	D2 r1rr	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,0rr	C2 lrrr	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	0rr,0r	D3 lrrlr	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	0r,0rrr	C3 lr1rr	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	0rr,r	92 r1rr	2	12,0							Lo mismo que LDC pero con DM activa baja para indicar memoria de datos.
	LDE	r,0rr	82 lrrr	2	12,0							
	LDEI	0rr,0r	93 lrrlr	2	18,0							Lo mismo que LDCI pero con DM activa baja para indicar memoria de datos.
	LDEI	0r,0rr	83 lr1rr	2	18,0							
INMEDIATO	LD	R, IM	E6 R IM	3	10,5							[R] ← IM Transfiere el dato inmediato al registro seleccionado.
	LD	0R, IM	E7 IR IM	3	10,5							[[R]] ← IM Transfiere el dato inmediato al registro direccionado indirectamente.
	LD	r, IM	rC IM	2	4,5							[r] ← IM Transfiere el dato inmediato al registro de trabajo seleccionado.
	SRP	IM	31	2	4,1							[FD,4] ← IM Transfiere el dato inmediato a SRP Registro APUNTAOR A REGISTROS.

TIPO	MEMONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS						OPERACION DESARROLLADA
						C	Z	S	O	D	H	
REFERENCIALA MEMORIA PRIMARIA	LDC	8rr,r	D2 r lrr	2	12,0							[Err] <— [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,8rr	C2 lrrr	2	12,0							[r] <— [Err] Transfiere el contenido de la localidad de memoria externa direccionada por el contenido del registro par al registro de trabajo seleccionado.
	LDCI	8rr,8r	D3 lrrir	2	18,0							[Err] <— [Err] , [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	8r,8rrr	C3 lrlrr	2	18,0							[Err] <— [Err], [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	8rr,r	92 r lrr	2	12,0							Lo mismo que LDC pero con DM activa baja para indicar memoria de datos.
	LDE	r,8rr	82 lrrr	2	12,0							
	LDEI	8rr,8r	93 lrrir	2	18,0							Lo mismo que LDCI pero con DM activa baja para indicar memoria de datos.
	LDEI	8r,8rr	83 lrlrr	2	18,0							
INMEDIATO	LD	R, IM	E6 r l IM	3	10,5							[R] <— IM Transfiere el dato inmediato al registro seleccionado.
	LD	8R, IM	E7 lR IM	3	10,5							[R] <— IM Transfiere el dato inmediato al registro direccionado indirectamente.
	LD	r, IM	FC IM	2	6,5							[r] <— IM Transfiere el dato inmediato al registro de trabajo seleccionado.
	SNP	IM	31	2	6,1							[FD, a] <— IM Transfiere el dato inmediato a SNP Registro APUNTADOR A REGISTROS.

TIPO	NEMONICO	OPERANDO	C O D I G O O B J E T O	BYTES	CICLOS DE RELOJ	STATUS					O P E R A C I O N D E S A R R O L L A D A
						C	Z	S	O	D	
B A L T O	DJNZ	r, f(n)	ra Desplazam.	2	12/10 3/5						[r] ← [r] - 1, si [r] > 0 entonces [PC] ← [PC] + Desplazam. Incrementa el conte- nido del registro de trabajo seleccionado. Si no es cero ejecuta el salto relativo de programa.
	JP	cc, DA	ccD DA _{cc} DA _L	3	12/10, 0						Si cc es "verdadero" entonces [PC] ← DA Salta a la localidad de memoria direccionada directamente si cc es verdadero.
	JP	DIR	30 IRR	2	6, 0						[PC] ← [RR] Salta a la localidad de memoria cuya dirección es el contenido del registro par.
	JR	cc, RA	ccB Desp.	2	12/10, 3						Si cc es "verdadero" entonces [PC] ← [PC] + Desp. ejecuta un salto relativo al cc es verdadero.
L B U R A U T O D I R N A D B	CALL	DA	D6 DA _{cc} DA _L	3	20, 0						[SP] ← [SP] - 2, [SP] ← [PC], [PC] ← [DA] Salta a la dirección direccionada por la subru- tina.
	CALL	DIR	D4 IRR	2	20, 0						[SP] ← [SP] - 2, [SP] ← [PC], [PC] ← [RR] Salta a la subrutina cuya dirección es conte- nida en el registro par.
	RET	AF		1	14, 0						[PC] ← [SP], [SP] ← [SP] + 2 Regreso de una subrutina.

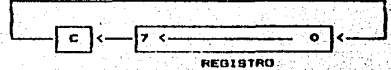
TIPO	MEMONICO	OPERANDO	C O D I G O O B J E T O	BYTES	CICLOS DE RELOJ	STATUS					O P E R A C I O N D E S A R R O L L A D A	
						C	Z	S	O	D		H
O P E R A D O R I M E D I A T O	ADC	R, IM	16 R IM	3	10,5	X	X	X	X	0	X	[R] $\leftarrow$ [R] + IM + C Suma el dato inmediato con acarreo con el contenido del registro seleccionado.
	ADC	DR, IM	17 IR IM	3	10,5	X	X	X	X	0	X	[R] $\leftarrow$ [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] $\leftarrow$ [R] + IM Lo mismo que ADC pero sin sumar el acarreo.
	ADD	DR, IM	07 IR IM	3	10,5	X	X	X	X	0	X	[R] $\leftarrow$ [R] + IM
	AND	R, IM	56 R IM	3	10,5		X	X	0			[R] $\leftarrow$ [R] AND IM Como ADC pero desarrolla una operación AND
	AND	DR, IM	57 IR IM	3	10,5		X	X	0			[R] $\leftarrow$ [R] AND IM lógica.
	CP	R, IM	6 R IM	3	10,5	X	X	X	X			[R] $\leftarrow$ IM Como ADC pero esta operación solo afecta el estado de las banderas.
	CP	DR, IM	A7 IR IM	3	10,5	X	X	X	X			[R] $\leftarrow$ IM
	OR	R, IM	46 R IM	3	10,5		X	X	0			[R] $\leftarrow$ [R] OR IM Como ADC pero desarrollando una operación OR lógica.
	OR	DR, IM	47 IR IM	3	10,5		X	X	0			[R] $\leftarrow$ [R] OR IM
	RBC	R, IM	36 R IM	3	10,5	X	X	X	X	1	X	[R] $\leftarrow$ [R] - IM - C Como ADC pero ejecutando la operación de resta con acarreo.
	RBC	DR, IM	37 IR IM	3	10,5	X	X	X	X	1	X	[R] $\leftarrow$ [R] - IM - C
	SUB	R, IM	26 R IM	3	10,5	X	X	X	X	1	X	[R] $\leftarrow$ [R] - IM Como ADC pero ejecutando la operación de resta.
	SUB	DR, IM	27 IR IM	3	10,5	X	X	X	X	1	X	[R] $\leftarrow$ [R] - IM

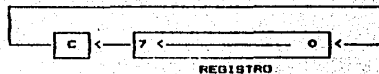
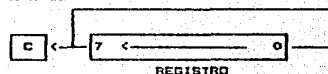
TIPO	NEMONICO	OPERANDO	C O D I G O D E O B J E T O	BYTES	CICLOS DE RELOJ	STATUS						O P E R A C I O N	D E S A R R O L L A D A
						C	Z	S	O	D	H		
C O M P A R A D O R	TCH	R, IM	66 R IM	3	10,5	x	x	0				[R] AND (IM)* Como AND para esta operaci3n solo afecta al estado del registro de banderas.	
	TCM	8R, IM	76 IR IM	3	10,5	x	x	0				[CR] AND (IM)* Como AND para esta operaci3n solo afecta al estado del registro de banderas.	
	TM	R, IM	76 R IM	3	10,5	x	x	0				[R] AND IM Como AND para esta operaci3n solo afecta al estado del registro de banderas.	
	TR	8R, IM	77 IR IM	3	10,5	x	x	0				[CR] AND IM Como AND para esta operaci3n solo afecta al estado del registro de banderas.	
	XOR	R, IM	D6 R IM	3	10,5	x	x	0				[R] ← [R] O IM Como AND para ejecutando la operaci3n l3gica OR exclusiva.	
	XOR	8R, IM	D7 IR IM	3	10,5	x	x	0				[CR] ← [CR] O IM Como AND para ejecutando la operaci3n l3gica OR exclusiva.	
M O V I M I E N T O	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	R _s , R _v	E4 R _s R _v	3	10,5							[R _s] ← [R _v] Transfiere el contenido del registro seleccionado (R _v) al registro seleccionado (R _s).	
R E C O R D A D O	LD	R _s , 8R _v	E5 R _s 1R _v	3	10,5							[R _s] ← [8R _v] Transfiere datos entre los registros seleccionados y los registros direccionados indirectamente.	
	LD	8R _s , 8R _v	F5 1R _s R _v	3	10,5							[8R _s] ← [8R _v] Transfiere datos entre los registros seleccionados y los registros direccionados indirectamente.	
A	LD	r, IX, R	C7 r IX R									[r] ← [R + (IX)] Transfiere datos entre los registros de trabajo seleccionados y los registros direccionados usando direccionamiento indexado.	
	LD	R, IX, r	D7 r IX R									[R + (IX)] ← [r] Transfiere datos entre los registros de trabajo seleccionados y los registros direccionados usando direccionamiento indexado.	

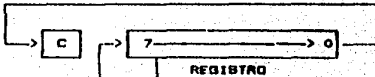
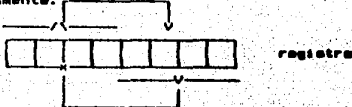
TIPO	HEXONICO	OPERANDO	CODIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS								OPERACION	DESARROLLADA
						C	Z	S	O	D	H				
OPERADOR	ADC	r_n, r_v	12 r_n, r_v	2	6,5	X	X	X	X	O	X		$[r_n] \leftarrow [r_n] + [r_v] + C$ Suma el contenido del registro de trabajo fuente seleccionado con acarreo al contenido del registro de trabajo destino seleccionado.		
	ADC	$r_n, 8r_v$	13 $r_n, 1r_v$	2	6,5	X	X	X	X	O	X		$[r_n] \leftarrow [r_n] + [[r_v]] + C$ Suma el contenido del registro de trabajo fuente direccionado indirectamente con acarreo al contenido del registro de trabajo destino seleccionado.		
	ADC	R_n, R_v	14 R_n, R_v	3	10,5	X	X	X	X	O	X		$[R_n] \leftarrow [R_n] + [R_v] + C$ Suma el contenido del registro fuente seleccionado con acarreo al registro destino seleccionado.		
MODIFICADOR	ADC	$R_n, 8R_v$	15 $R_n, 1R_v$	3	10,5	X	X	X	X	O	X		$[R_n] \leftarrow [R_n] + [[R_v]] + C$ Suma el contenido del registro fuente direccionado indirectamente con acarreo al registro destino seleccionado.		
	ADD	r_n, r_v	02 r_n, r_v	2	6,5	X	X	X	X	O	X		$[r_n] \leftarrow [r_n] + [r_v]$		
	ADD	$r_n, 8r_v$	03 $r_n, 1r_v$	2	6,5	X	X	X	X	O	X		$[r_n] \leftarrow [r_n] + [[r_v]]$		
HEG	ADD	R_n, R_v	04 R_n, R_v	3	10,5	X	X	X	X	O	X		$[R_n] \leftarrow [R_n] + [R_v]$		
	ADD	$R_n, 8R_v$	05 $R_n, 1R_v$	3	10,5	X	X	X	X	O	X		$[R_n] \leftarrow [R_n] + [[R_v]]$		
	ADD	r_n, r_v	02 r_n, r_v	2	6,5	X	X	X	X	O	X		Suma el contenido del registro fuente al registro destino usando una de las opciones descritas para ADC.		
STH	AND	r_n, r_v	52 r_n, r_v	2	6,5	X	X	X	X				$[r_n] \leftarrow [r_n] \text{ AND } [r_v]$		
	AND	$r_n, 8r_v$	53 $r_n, 1r_v$	2	6,5	X	X	X	X				$[r_n] \leftarrow [r_n] \text{ AND } [[r_v]]$		
	AND	R_n, R_v	54 R_n, R_v	3	10,5	X	X	X	X				$[R_n] \leftarrow [R_n] \text{ AND } [R_v]$		
CONTINUA	AND	$R_n, 8R_v$	55 $R_n, 1R_v$	3	10,5	X	X	X	X				$[R_n] \leftarrow [R_n] \text{ AND } [[R_v]]$		
	CP	r_n, r_v	A2 r_n, r_v	2	6,5	X	X	X	X				Realiza la operación lógica AND del registro fuente con el registro destino utilizando una de las opciones de direccionamiento de ADC.		
	CP	$r_n, 8r_v$	A3 $r_n, 1r_v$	2	6,5	X	X	X	X				$[r_n] - [r_v]$		
A	CP	R_n, R_v	A4 R_n, R_v	3	10,5	X	X	X	X				$[R_n] - [R_v]$		
	CP	$R_n, 8R_v$	A5 $R_n, 1R_v$	3	10,5	X	X	X	X				$[R_n] - [[R_v]]$		
	CP	r_n, r_v	A2 r_n, r_v	2	6,5	X	X	X	X				Compara el registro fuente con el registro destino usando uno de los modos de direccionamiento descrito para ADC.		

TIPO	ALFABÉTICO	OPERANDO	C O D I G O O B J E T O	L Y T E S	C I C L O S D E R E L O J	S T A T U S					O P E R A C I O N D E S A R R O L L A D A
						C	I	S	D	H	
O P E R A D O R E G I S T R O A R E Q U I S I T O	OR	r_n, r_v	42 r_n, r_v	2	6,5	x	x	x			$[r_n] \leftarrow [r_n] \text{ OR } [r_v]$
	OR	$r_n, 0r_v$	43 $r_n, 0r_v$	2	6,5	x	x	x			$[r_n] \leftarrow [r_n] \text{ OR } [r_v]$
	OR	R_n, r_v	44 R_n, r_v	3	10,5	x	x	x			$[R_n] \leftarrow [R_n] \text{ OR } [r_v]$
	OR	$R_n, 0r_v$	45 $R_n, 0r_v$	3	10,5	x	x	x			$[R_n] \leftarrow [R_n] \text{ OR } [R_v]$
											Realiza la operación lógica OR del registro fuente con el registro destino usando uno de los direccionamientos descritos para ADC.
	SBC	r_n, r_v	32 r_n, r_v	2	6,5	x	x	x	1	x	$[r_n] \leftarrow [r_n] - [r_v] - C$
	SBC	$r_n, 0r_v$	33 $r_n, 0r_v$	2	6,5	x	x	x	1	x	$[r_n] \leftarrow [r_n] - [r_v] - C$
	SBC	R_n, r_v	34 R_n, r_v	3	10,5	x	x	x	1	x	$[R_n] \leftarrow [R_n] - C$
	SBC	$R_n, 0r_v$	35 $R_n, 0r_v$	3	10,5	x	x	x	1	x	$[R_n] \leftarrow [R_n] - [R_v] - C$
											Resta con acarreo el registro fuente del registro destino usando uno de los modos de direccionamiento descrito para ADC.
	SUB	r_n, r_v	22 r_n, r_v	2	6,5	x	x	x	1	x	$[r_n] \leftarrow [r_n] - [r_v]$
	SUB	$r_n, 0r_v$	23 $r_n, 0r_v$	2	6,5	x	x	x	1	x	$[r_n] \leftarrow [r_n] - [r_v]$
	SUB	R_n, r_v	24 R_n, r_v	3	10,5	x	x	x	1	x	$[R_n] \leftarrow [R_n] - [r_v]$
	SUB	$R_n, 0r_v$	25 $R_n, 0r_v$	3	10,5	x	x	x	1	x	$[R_n] \leftarrow [R_n] - [R_v]$
											Resta el registro fuente del registro des- tino usando uno de los modos de direcciona- miento descritos para ADC.
	TCM	r_n, r_v	63 r_n, r_v	2	6,5	x	x	0			$[r_n] \text{ AND } [r_v]$
	TCM	$r_n, 0r_v$	64 $r_n, 0r_v$	2	6,5	x	x	0			$[r_n] \text{ AND } [r_v]$
	TCM	R_n, r_v	64 R_n, r_v	3	10,5	x	x	0			$[R_n] \text{ AND } [r_v]$
	TCM	$R_n, 0r_v$	65 $R_n, 0r_v$	3	10,5	x	x	0			$[R_n] \text{ AND } [R_v]$
											Realiza la operación lógica AND entre el primer registro y el complemento del segun- do registro usando uno de los modos de di- reccionamiento descrito para ADC.
	TH	r_n, r_v	72 r_n, r_v	2	6,5	x	x	0			$[r_n] \text{ AND } [r_v]$
	TH	$r_n, 0r_v$	73 $r_n, 0r_v$	2	6,5	x	x	0			$[r_n] \text{ AND } [r_v]$
	TH	R_n, r_v	74 R_n, r_v	3	10,5	x	x	0			$[R_n] \text{ AND } [r_v]$
	TH	$R_n, 0r_v$	75 $R_n, 0r_v$	3	10,5	x	x	0			$[R_n] \text{ AND } [R_v]$
											Realiza la operación lógica AND entre el contenido del primer registro y el conteni- do utilizando uno de los modos de direcciona- miento descrito para ADC.

TIPO	NEMÓNICO	OPERANDO	CÓDIGO OBJETO	BYTES	CICLOS DE RELOJ	STATUS				OPERACION DESARROLLADA
						C	Z	S	O	
OPERACIONES BÁSICAS	XOR	R_n, R_v	B2 R_n, R_v	2	6,5	X	X	O		$[R_n] \leftarrow [R_n] + [R_v]$
	XOR	R_n, DR_v	B3 R_n, DR_v	2	6,5	X	X	O		$[R_n] \leftarrow [R_n] + [DR_v]$
	XOR	R_n, R_v	B4 R_n, R_v	3	10,5	X	X	O		$[R_n] \leftarrow [R_n] + [R_v]$
	XOR	R_n, DR_v	B5 R_n, DR_v	3	10,5	X	X	O		$[R_n] \leftarrow [R_n] + [DR_v]$
OPERACIONES AVANZADAS										Realiza la operación lógica OR exclusiva del registro fuente con el registro destino usando una de las opciones de direccionamiento descritas para ADC.
	CLR	R	B0 R	2	6,5					$[R] \leftarrow 0$ Borra el registro seleccionado.
	CLR	DR	B1 IR	4	6,5					$[R] \leftarrow 0$ Borra el registro direccionado indirectamente.
	COM	R	B0 R	2	6,5	X	X	O		$[R] \leftarrow ([R])'$ Complementa el contenido del registro seleccionado.
	COM	DR	B1 IR	2	6,5	X	X	O		$[R] \leftarrow ([R])'$ Complementa el contenido del registro direccionado indirectamente.
	DA	R	B0 R	2	8,5	X	X	X	U	Realiza el ajuste decimal del registro seleccionado.
	DA	DR	B1 IR	2	8,5	X	X	X	U	Realiza el ajuste decimal del registro direccionado indirectamente.
	DEC	R	B0 R	2	6,5	X	X	X		$[R] \leftarrow [R] - 1$ Decrementa el contenido del registro seleccionado.
	DEC	DR	B1 IR	2	8,5	X	X	X		$[R] \leftarrow [R] - 1$ Decrementa el contenido del registro direccionado indirectamente.
	DECH	RR	B0 RR	2	10,5	X	X	X		$[RR] \leftarrow [RR] - 1$ Decrementa el contenido del registro por seleccionado.
	DECH	DRR	B1 RRR	2	10,5	X	X	X		$[RR] \leftarrow [RR] - 1$ Decrementa el contenido del registro por direccionado indirectamente.

TIPO	NOMINICO	OPERANDO	C O D I G O O B J E T O	BYTES	CICLOS DE RELOJ	STATUS					OPERACION DESARROLLADA
						C	Z	S	O	D	
O P E R A D O R A R E G I S T R O	INC	r	rE	1	6,5	X	X	X	X		[r] ← [r] + 1 Incrementa el contenido del registro de trabajo seleccionado.
	INC	R	20 R	2	6,5	X	X	X	X		[R] ← [R] + 1 Incrementa el contenido del registro seleccionado.
	INC	BR	21 IR	2	6,5	X	X	X	X		[RR] ← [RR] + 1 Incrementa el contenido del registro direccionado indirectamente.
	INCM	RR	A0 RR	2	10,5	X	X	X	X		[RR] ← [RR] + 1 Incrementa el contenido del registro par seleccionado.
	INCM	RRR	A1 IRR	2	10,5	X	X	X	X		[RRR] ← [RRR] + 1 Incrementa el contenido del registro par seleccionado, direccionado indirectamente.
	RL RL	R BR	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 BR	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	C O D I G O O B J E T O	BYTES	CICLOS DE RELOJ	STATUS								O P E R A C I O N D E S A R R O L L A D A
						C	Z	S	O	D	H			
O P E R A D O R A R E G I S T R A D O	RR RR	R RR	EO R E1 IR	2 2	4,5 4,5	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 RR	10 R 11 R	2 2	4,5 4,5	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 RR	DO R D1 IR	2 2	4,5 4,5	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. 
	SWP SWP	R RR	FO R FI IR	2 2	4,5 4,5	X X	X X	X X	X X					Interchangea los nibbles del registro seleccionado o del registro direccionado indirectamente. 

TIPO	Mnemónico	Operando	C O D I D O B J E T O	BYTES	CICLOS DE RELOJ	STATUS					OPERACION DESARROLLADA
						C	I	O	D	H	
S T A C K	POP	R	50 R	2	10,5						[R] ← ([SP]), [SP] ← ([SP] + 1) [R3] ← ([R3] + 1) Realiza el POP al stack del registro direccionado directo o indirectamente.
	PUSH	R	70 R 71 IR	2	10/12,1 12/14,1						
I N T E R R U P T I O N E S	DI		DF	1	4,1						Deshabilita interrupciones: [FD _{1,3}] ← ([FD _{1,3}] AND 7F ₁₆). Habilita interrupciones: [FD _{1,3}] ← ([FD _{1,3}] OR 80 ₁₆). [PC] ← ([SP]), [SP] ← [SP] + 2, [FD _{1,3}] ← ([FD _{1,3}] OR 80 ₁₆). Regreso de una rutina de servicio de interrupción y habilita interrupciones.
	SI		DF	1	4,0						
	INTE		DF	1							
E S T A D O	CCF		EF	1	4,5	X					[C] ← ([C]) [*] Complementa la bandera de acarreo. [C] ← 0 Pone a cero la bandera de acarreo. [C] ← 1 Pone a 1 la bandera de acarreo.
	RCF		CF	1	4,5	0					
	SCF		DF	1	4,5	1					
	NOP		FF	1	4,0						Sin operación.

CAPITULO IV

DESCRIPCION DEL SIS 28

A partir de este capítulo entraremos en materia del SIS 28, comenzando por una descripción general, en la que se hará un recuento de los bloques funcionales de que dispone el usuario del SIS 28. Asimismo, también se aclarará su operación y en la última sección del capítulo se introducen los comandos del SCAN 28.

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 28.

Es evidente que hemos invertido el orden cronológico del desarrollo del SIS 28. Esta organización obedece al objetivo de darle un carácter inductivo a la redacción para facilitar el entendimiento del proyecto.

ELEMENTOS DEL SIS 28.

Cuando se revisaron los objetivos de este proyecto en el Capítulo 1, se presentaron superficialmente los principales elementos del SIS 28. 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.

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 pines en sus distintas modalidades pueden ser empleados en el programa de prueba a excepción de P27 y P32.

- PUERTOS SERIE.

Cuenta con dos puertos asíncronos 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 acceder 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.

FIGURA (4.2)

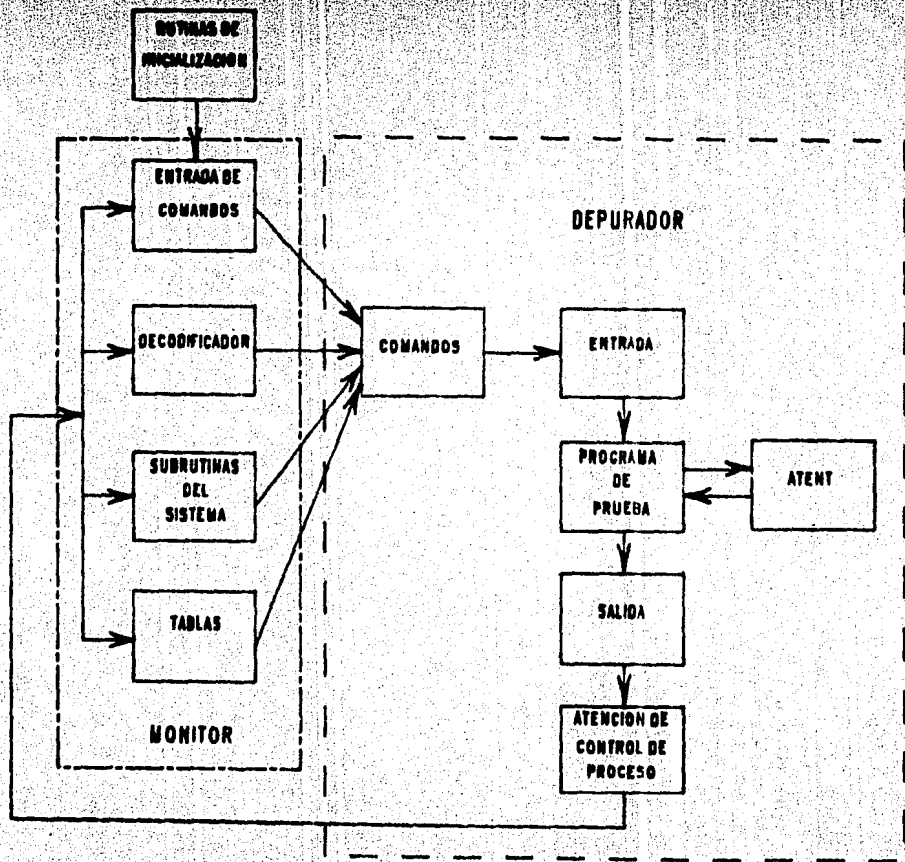
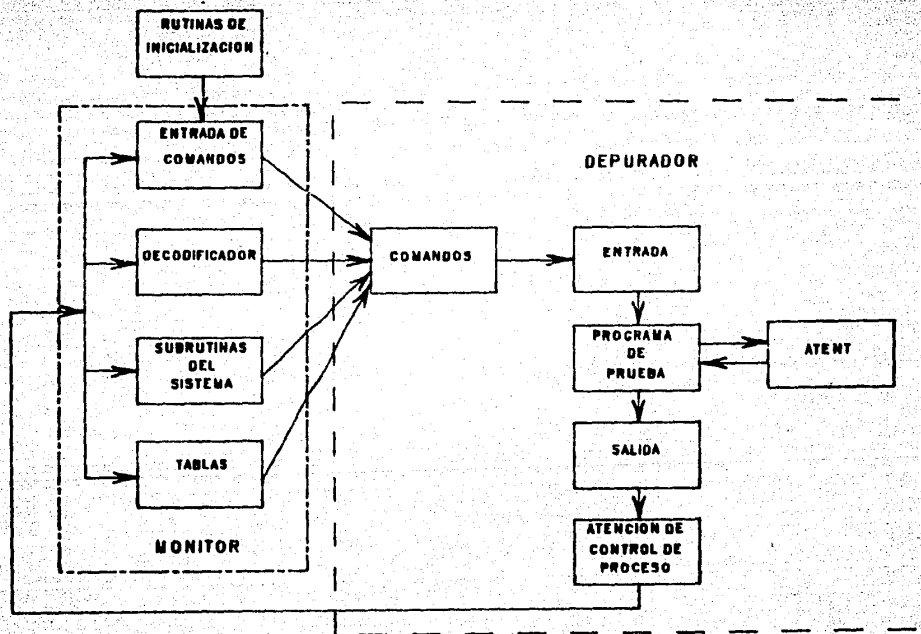


FIGURA (4.2)



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 Prescaladores 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 estructura 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 accediendo exclusivamente al SIS Z8. Ahora el monitor espera la introducción de algún comando. El primer carácter recibido forza al monitor al Estado 4, donde espera el carácter (Carriage Return) que le indica al monitor que toda una línea de un comando ha sido introducida. Cada carácter recibido durante el Estado 4 es introducido a un Buffer que reside en el Archivo de Registros del Z8. La recepción del carácter de retorno CR envía 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 (8), que es el prompt del SCAN 28.

SUBROUTINAS DEL SCAN 28.

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

NOMBRE DE LA SUBROUTINA	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 carácter 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 dígitos.
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 específicamente desea utilizar. El conjunto de comandos incluye en total 17 rutinas, entre las cuales se incluyen las más comúnmente 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 28.

Antes de poderse operar el SIS 28, 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 28, 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 respectivas terminales de referencia. No es necesario unir los nodos de referencia de las fuentes, puesto que esto se ha implementado ya en el SIS 28.

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 28 tienen sus salidas alambradas a conectores tipo DB25 y sus niveles son compatibles con los del protocolo RS232C. De este protocolo, el SIS 28 maneja solamente 2 señales: RDX y TXD. La asignación de pins de estos conectores es la mostrada en la Figura (4.3):

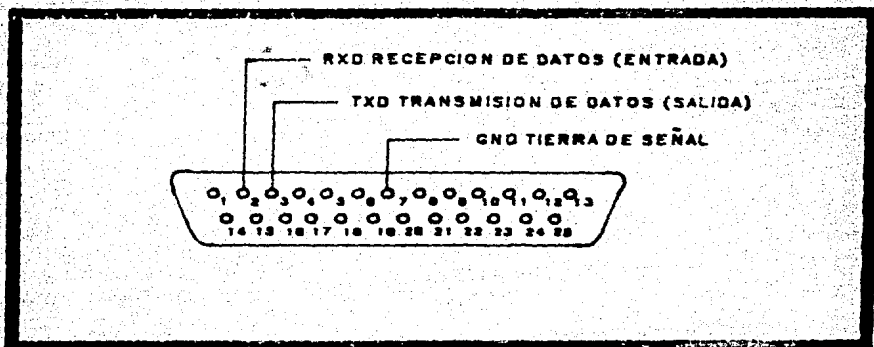


FIGURA (4.3)

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 acordeamente, 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.

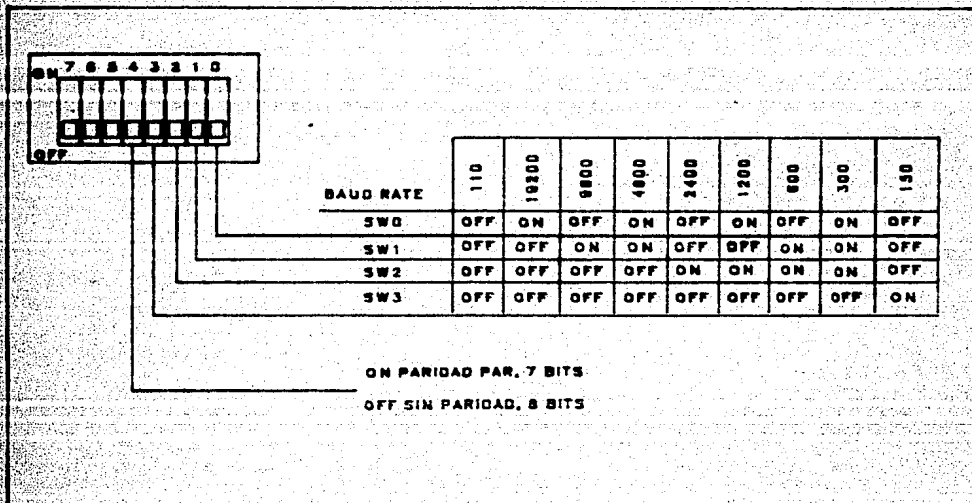


FIGURA (4.4)

4.- Encender las fuentes externas del SIS Z8 y el resto del equipo copiado. 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.

- CONVENIONES

Cada comando es tratado por separado. Las letras resacadas 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. Indica un dato de extensión máxima de 4 dígitos hexadecimales.
- n. 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

Assumiendo 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 carácter enviado por la terminal, en espera de la clave de acceso. La clave de acceso le indica, cuando el usuario selecciona 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 acceder al SIS Z8, el usuario introduce la clave de acceso que consiste de la siguiente secuencia de caracteres :

CTRL

RETURN

F

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 THR 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 SM MV THR	MR DR	MA PM	MST	GO CALL T 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 ésta 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, el mismo se encargue de respaldar los registros de control del F1 al FB en el grupo de registros 10H. 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 - 0FFFFH	3200H - 3FFFFH 4000H - 4FFFFH
2	1000H - 2FFFFH	
3	5000H - 7FFFFH	1000H - 2FFFFH

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 leída 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 accederlas 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 accedidas 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 2143H de Memoria de Datos, simultáneamente se está alterando el contenido de la dirección 6143H 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 ZB 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 rutinas del comando DR, que es la que despliega los registros del ZB. 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.

- DESCRIPCIÓN 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] d₁,d₂

d₁.- Dirección Inicial

d₂.- Dirección Final

FUNCION:

Despliega la memoria del SIS Z8 contenida entre las direcciones d₁ y d₂. Si la opción es empleada, las direcciones inicial y final corresponden a Memoria de Datos; d₁ es la dirección inicial y d₂ 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 envía 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:

•00 00,1000

0000	44 4F 44 4F 00 01 02 03 04 05 06 07 08 09 0A 0B	DOBO.....
0010	09 10 1A 33 39 33 3A 3B 2B 1F 1E 1B 1C 1D 1A 19	..SVSZ.....
0020	01 02 03 35 04 4E 03 41 06 49 1B 1A 1C 1B 01 02	..U.N.A.N.....

En el primer campo se indica la dirección del primer dígito 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 símbolos, que son los equivalentes ASCII de los 16 datos desplegados en el segundo campo. En el caso de que un dato no tenga un símbolo 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 despliegue continúa oprimiendo cualquier tecla. También es posible abortar el despliegue oprimiendo la tecla ESC.

COMANDO SM

SUSTITUCION DE MEMORIA.

FORMATO: SMID] d_i

d_i.- Dirección de memoria a sustituirse.

FUNCION:

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

d_i debe ser de extensión máxima de 4 dígitos, si esta longitud es excedida, el SCAN ZB envía un mensaje de error.

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

Ejemplos: SMD 3000

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

•SMD 3000

3000 n1

Donde n₁ 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 dígitos. 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] d₁,d₂,d₃

P: Transferencia Programa a Programa.

D: Transferencia Datos a Datos.

E: Transferencia Programa a Datos.

I: Transferencia Datos a Programa.

d₁: Dirección Inicial del Bloque a Transferir.

d₂: Dirección Final del Bloque a Transferir.

d₃: Dirección Inicial del Destino.

FUNCION:

Transfiere un bloque de memoria, de una posición a otra, en memoria. El bloque queda comprendido entre las direcciones d₁ y d₂. La nueva localización comienza a partir de d₃. 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 ZB 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 esto, 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 envío 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:

SIS Z8 LISTO PARA RECIBIR INSTRUCCIONES

•

PROMPT SIS Z8

•

MST

•

PROMPT COMPUTADORA ANFITRIONA

A

A

CTRL F RETURN

•

COMANDO TRM

TRANSFERENCIA DE MEMORIA

FORMATO: TRM

FUNCION:

Este comando transfiere un archivo en formato INTEL, generado por el ensamblador del Z8, 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 transportable, 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.

```
10B0000 00 23 43 54 6F 7C 42 56 32 93
10B0008 00 32 43 55 00 00 32 1C 23 B5
10B0010 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 usado. 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 corrección 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 envía a la terminal, tal como se desplegaría 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 CTRLF), 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 envía 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 CODIGOS(27)
30 INPUT "DAME EL NOMBRE DEL ARCHIVO [ NOMBRE.HEX ].",NOM$
40 OPEN "\\NOM$
50 OUT 1,6
60 INPUT "\\CODIGOS$
70   FOR BYTE=1 TO 26
80     BYTES=CODIGOS(BYTE,BYTE)
90     PRINT BYTES;
00     IF BYTES="" THEN 140
110   NEXT BYTE
120 PRINT
130 GOTO 60
140 CLOSE
150 OUT 1,6
160 END
```

Ejemplos:

```
STRM
ASBASIC
```

```
CROMEMCO 32K STRUCTURED BASIC
```

```
>>LOAD "TRM.SAV"
>>RUN
```

```
DAME EL NOMBRE DEL ARCHIVO .... PRUEBA.HEX
```

```
SIGUIENTE LOCALIDAD
```

```
6240
```

```
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 o 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 18. 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

```
!^@#&'()*+,-./01234567  
89:;<=>?@ABCDEFGHIJKLMN  
OPQRSTUVWXYZ[\]^_`abede  
fghijklmnopqrstuvwxyzi  
j~
```

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 3000H 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.

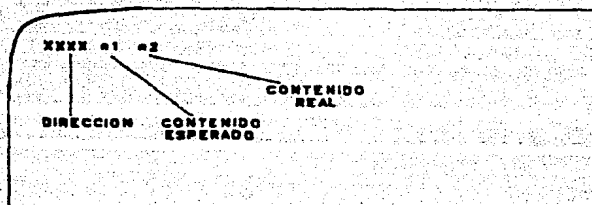
El comando PM envía un mensaje cada vez que comienza la prueba en determinado sector. Al finalizar esta primera prueba, inicia la prueba de lectura, la cual también se divide en los 4 sectores anteriores que son enunciados en forma similar a la anterior.

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:

```
XXXX n1
 |    \
 |     \
DIRECCION  DE INTENTOS
```

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 envía el segundo tipo de error que se despliega de la manera siguiente:



Donde la dirección indica la localidad donde ocurre la falla, n1 es el dato esperado de esa localidad y n2 es el dato real que fué leído.

Ejemplo:

• PM

PRUEBA DE VERIFICACION DESPUES DE ESCRITURA

SECTOR #1000H P
SECTOR #3000H P
SECTOR #1000H D
SECTOR #3000H D

PRUEBA DE LECTURA

SECTOR #1000H P
SECTOR #3000H P
SECTOR #1000H D
SECTOR #3000H D

•

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 esenciales 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

En primer lugar se introducirá el concepto de archivo de registros alterno. La diferencia esencial 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 esto es interrumpido, la rutina de atención común a TRACE y BP regresa los registros de control a 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 al 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_i

n_i: Número del registro a desplegarse.

DESCRIPCION:

Este comando despliega uno o varios de los registros del ZB. Si n_i 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 Alternativo, almacenado en Memoria de Datos Externa y que corresponden al estado de los registros del Archivo Interno del ZB 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 23
*R23H=43H

En primer lugar se introducirá el concepto de archivo de registros alterno. La diferencia esencial 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 éste es interrumpido, la rutina de atención común a TRACE y BP regresa los registros de control a 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 al 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] n₁

n₁: Número del registro a desplegarse.

DESCRIPCION:

Este comando despliega uno o varios de los registros del Z8. Si n₁ 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 Alternó, 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] n₁

n₁: Número de Registros a Modificarse.

DESCRIPCION:

Este comando modifica el contenido del registro indicado por n₁. 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 envía el código escape (ESC) no ocurrirá ninguna modificación.

Ejemplo:

```
•DRU 15  
  R15H=40H  
•MRU 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 ZB, 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 externo.

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 ZB, 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

ACOMPLIMIENTO DE PROGRAMA PASO A PASO

FORMATO: T n.

DESCRIPCION:

Cuando se desea 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.º Indica el número de instrucciones del programa de prueba que deben seguirse.

Ejemplos:

• T 10
•

Con este comando, 10 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:

• AT

•

COMANDO BP

BREAK POINTS

FORMATO:

BP d₁

d₁: 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 d₁, 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 fue ejecutado correctamente.

COMANDO DB

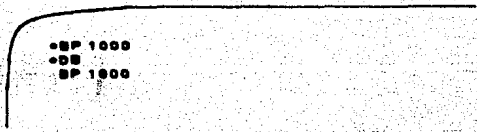
DESPLIEGUE DE LOCALIDADES CON BREAK POINTS

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:



•BP 1000
•BP 1000
•BP 1000

COMANDO BX

BORRADO DE BREAK POINTS

FORMATO: BX[d,1]

d,1 Es la dirección específica 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 específica, 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 d₁

d₁: 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 más sencilla de ejecutar pruebas a programas, sin embargo, tiene sus limitaciones: en primer lugar no introduce el archivo de registros alterno, ni modifica parámetros del SIS 28. Está diseñada para probar subrutinas y, por lo tanto, es posible regresar a ROOT mediante una instrucción RET. Es excelente para expansiones del SIS 28. Y sin embargo, es posible ejecutar Puntos Prueba o la opción Trace definiéndola previamente.

Ejemplo:

• CALL 20CH

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

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

COMANDO GO

EJECUCION DE UN PROGRAMA DE PRUEBA

FORMATO: GO[II] d₁

d₁: 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 [II] es empleada, el archivo de registros alterno es inicializado. Al entrar al programa de prueba, automáticamente este archivo sustituye al archivo de registros original del SIS Z8, los registros de control quedan programados en forma idéntica 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 opción I es indicia. 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.

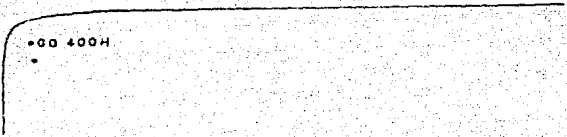
9.- Habilita Memoria de Datos.

10.- Salta a la rutina de entrada.

OBSERVACION:

El programa de Prueba debe comenzar en la Memoria Interna, por lo tanto, d, 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:



• 40 400H

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 comando 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 capítulo 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 ¿Qué puede hacer el SIS Z8 ? En el presente capítulo y el en siguiente trataremos de dar respuesta a la pregunta ¿Cómo funciona el SIS Z8 ? En este capítulo 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 capítulos 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 vías cada uno. La primera tarjeta es un circuito impreso (ver fig 5.1), que contiene únicamente el 28612 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 esto en el diagrama 1 no se encuentra representado el 28612, 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 28612 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.

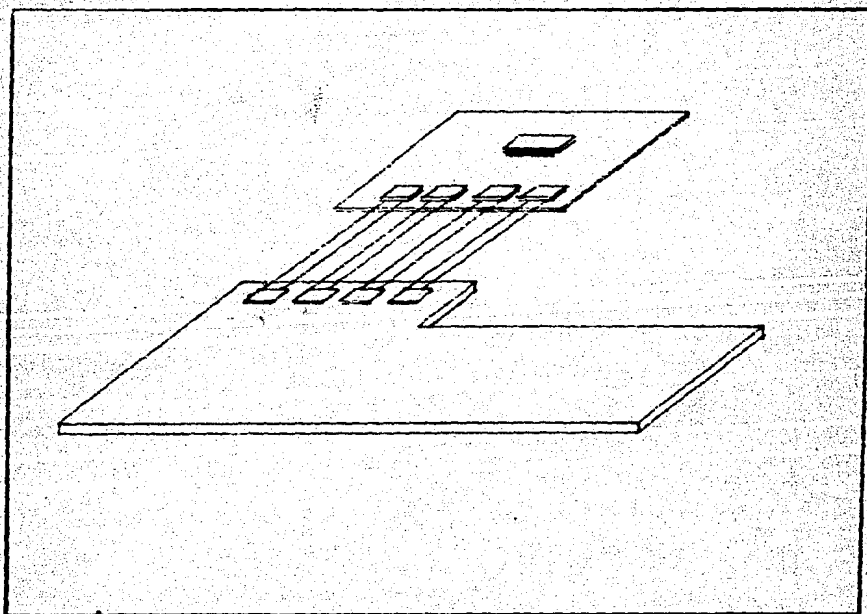
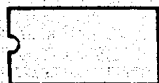


FIGURA (5.1)



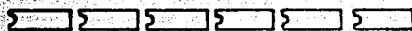
Z8612

U41 U42



T452 T454

U80 U81 U82 U83 U84 U85



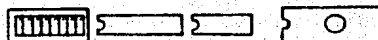
T453 T455 T456 T457 T458 T459

CN2-1 CN2-2 CN2-3 CN2-4 U10



T45201

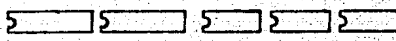
DS1 U38 U39 U40



T45201 T45204

8718

U33 U34 U35 U36 U37



T45201

T45201

T455

T4521

T4521

U15 U16 U17 U18



T45201

T45201

T45201

T455

U29 U30 U31 U32



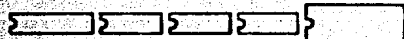
T455

8718

8718

8718

U10 U11 U12 U13 U14



T45201

T455

T4521

T4521

T4521

U26 U27 U28



T45201

T45201

T45201

U8



T455

U1



T455

U3



T452

U4



T452

U5



T452

U6



T452

U7



T452

U8



T452

CH1



U9



T45201

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 esto, 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 Sirva 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 están presentes solamente para acceder 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.

MDB 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 período de reloj inmediato precedente al siguiente ciclo de búsqueda 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.

P26	1	64	VCC
P31	2	63	XTAL2
P27	3	62	XTAL1
P26	4	61	P37
P25	5	60	P30
P24	6	59	RESET
P23	7	58	R/W
P22	8	57	DS
P21	9	56	AS
P20	10	55	P36
P33	11	54	P33
P34	12	53	P00
P17	13	52	P01
P16	14	51	P02
P15	15	50	P03
P14	16	49	P04
P13	17	48	GND
P12	18	47	P05
P11	19	46	P06
P10	20	45	P07
D7	21	44	IACK
D6	22	43	SYNC
D5	23	42	SCLK
D4	24	41	MDS
A0	25	40	D0
A1	26	39	D1
A2	27	38	D2
A3	28	37	D3
A4	29	36	A11
A5	30	35	A10
A6	31	34	A9
A7	32	33	A8

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-P37	PUERTO 3 E/S	ENTRADA
P34-P37	PUERTO 3 E/S	SALIDA
AD0-AD7	BUS DATOS/DIRECCIONES	BIDIRECCIONAL
AB-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	CONEXIONES 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.

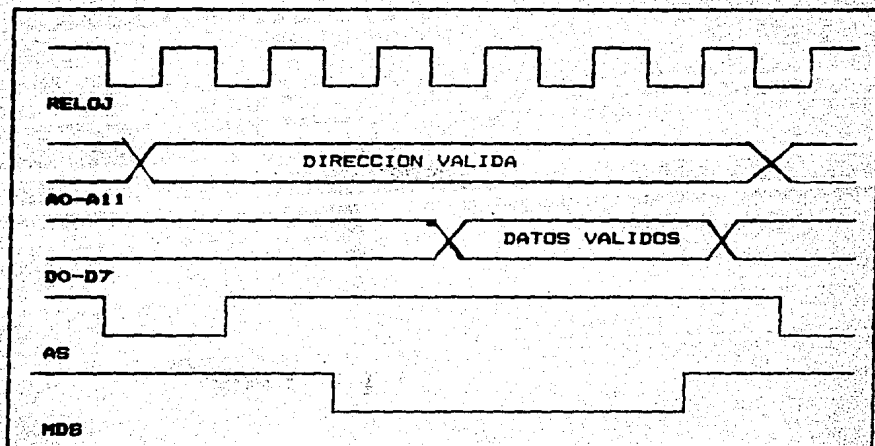


FIGURA (5.4)

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

IRQ0 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 IRQ0, 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.

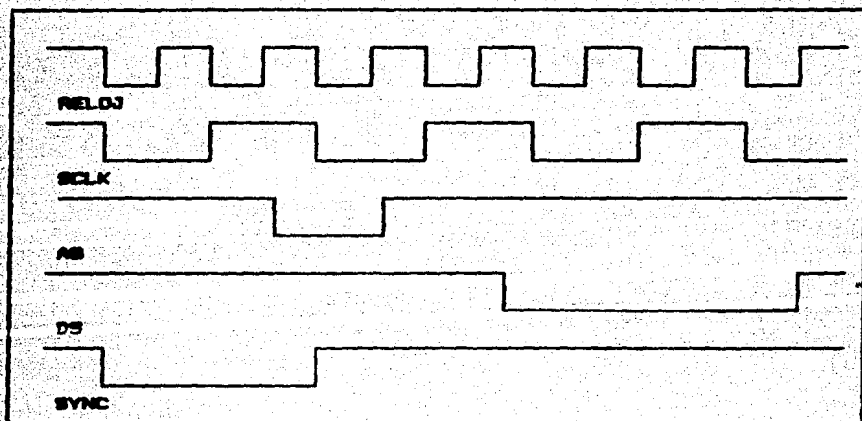
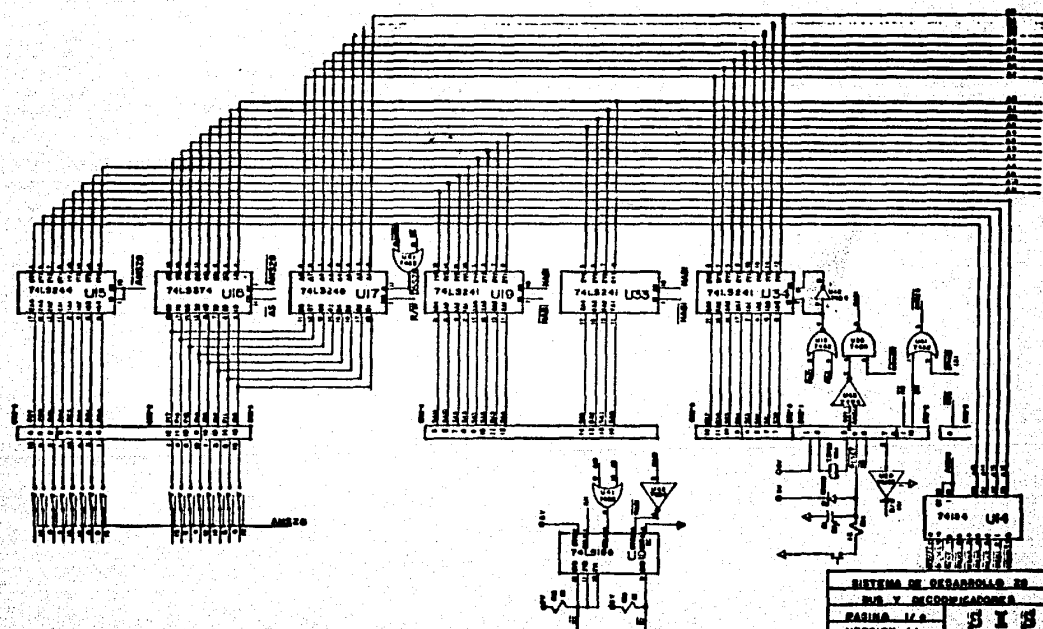


FIGURA (5.5)

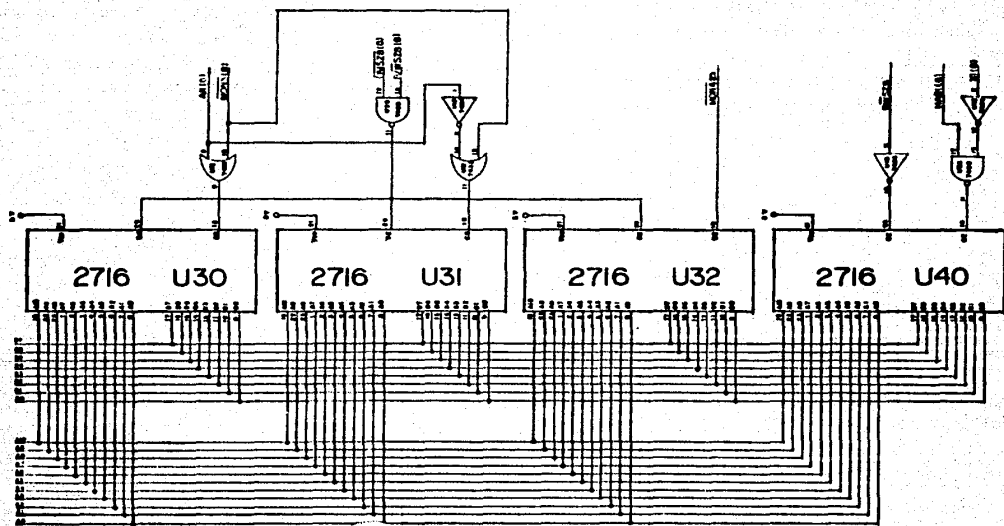
Las señales SIN, SOUT y DM, aunque son empleadas por el SIS 28, pueden ser reprogramadas por el usuario en cualquiera de sus modalidades.

Puesto que el SIS 28 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 asíncronas RS232-C

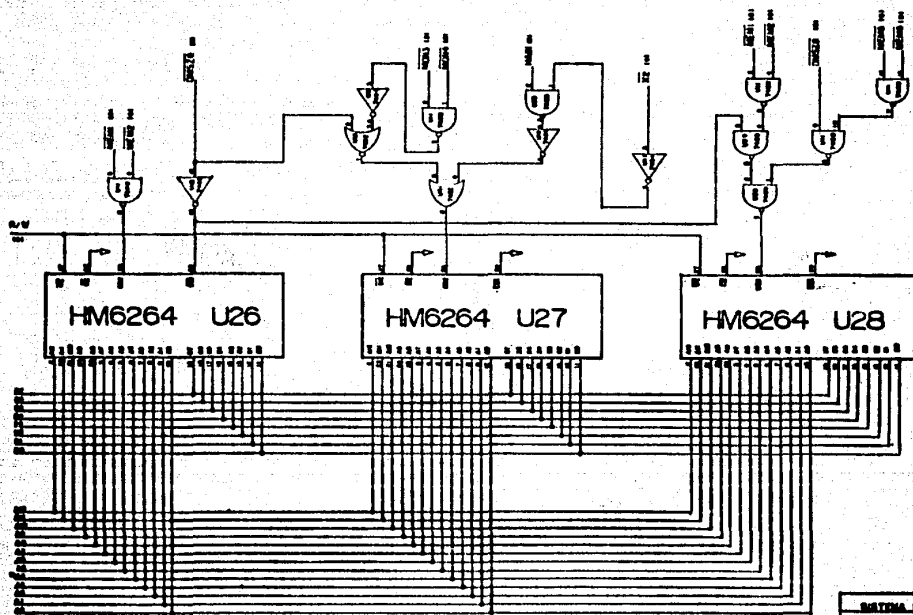
Ahora que ya se han visto algunas de las generalidades del SIS 28 podemos continuar con la revisión de los diagramas.



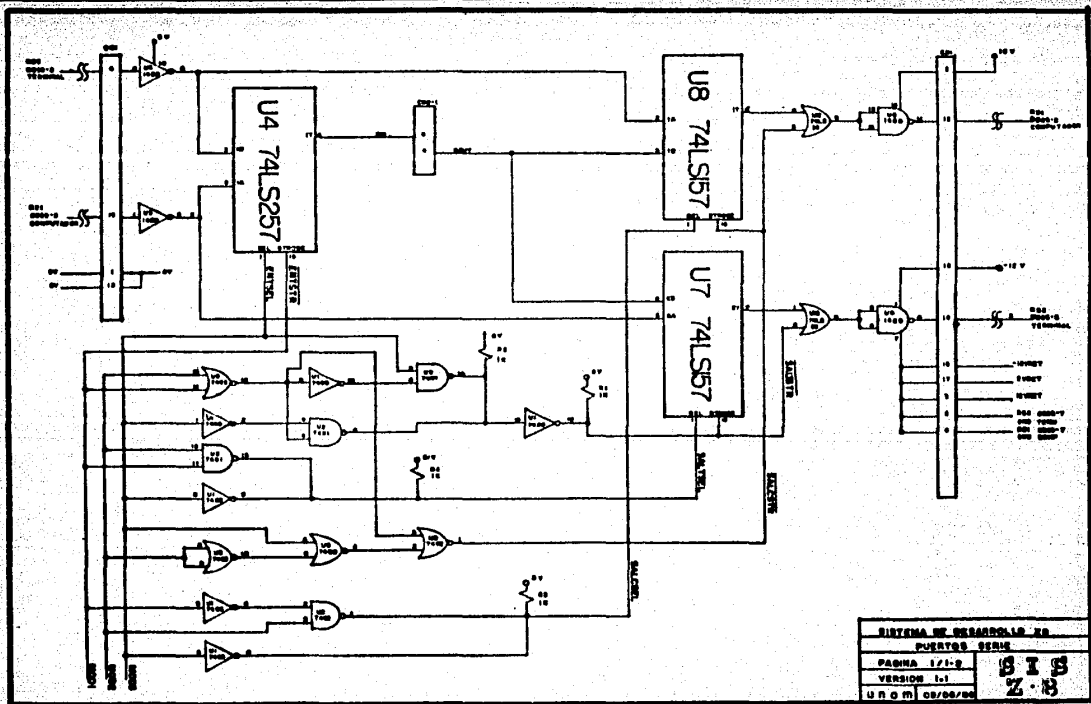
SISTEMA DE DESARROLLO EN	
BUS Y RECONOCEDORA	
PAGINA 1/2	SIS
VERSION 4.1	Z-8
UDGM 8/88/88	



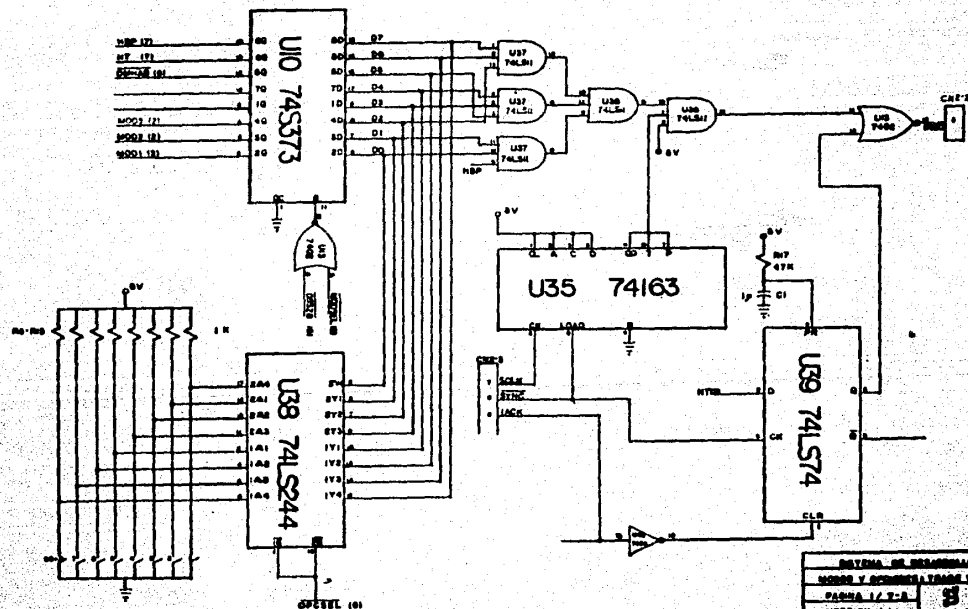
SISTEMA DE DESARROLLO RA	
MEMORIA	ROM
PAGINA 1/8	518
VERSION 1.0	2.8
UNOM	11/02/88



SISTEMA DE MEMORIA 20	
MEMORIA RAM ESTACION	
PAGINA 1/3	S I S 2-8
VERSION 1.1	
U R O M 11/06/00	



SISTEMA DE DESARROLLO 20	
PUNTERO 8080	
PAGINA 1/1-2	S I S Z - 8
VERSION 1-1	
U N O M 05/00/00	



DESIGNA DE DESARROLLO	3313
MODELO Y DESCRIPCION	3313
PARTE I / T-A	3313
VERSION 1.1	3313
13 0 0 0 0 0 0 0 0	3313

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 acceder cualquier byte de memoria a excepción de los primeros 4096 bytes de memoria de programa.

b) A través de las señales SA0-SA11 y SDO-SD7 independientemente del estado de la señal AMSZ8. En este caso solo se pueden acceder 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 accedido 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 esto complicó el alambrado.

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

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 28 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 AMS2B.

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 28 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 28 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 SDO-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 accederse 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 accederla en memoria de datos externa, donde sí son posibles los ciclos de escritura a memoria.

El SIS 28 emplea dos señales para controlar el origen de los buses: AMS2B y MEMINT. Al inicializar (RESET), la señal AMS2B 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 líneas SA0-SA11 y SDO-SD7 del Z8612. El programa monitor interno configura los Puertos 0 y 1 como buses, modifica el estado de la señal AMS2B 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 28 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 SDO-7. Este es el caso del comando 80, 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 AMSZB 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 AMSZB 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 previamente por el seguidor amplificador U20, debido a que el fan out al que está sometida, sobrepasa las especificaciones eléctricas máximas.

AMSZB después del inversor U42, habilita a U14, cuya función 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 AMSZB 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 SDO-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, así 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 mas adelante. La combinación de estas dos señales se realiza en la OR U41. DMHAB fue 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, siempre 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 U9 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 801 para que al inicio de su programa, el Z8612 quede con todos sus registros programados como inmediatos a un RESET.

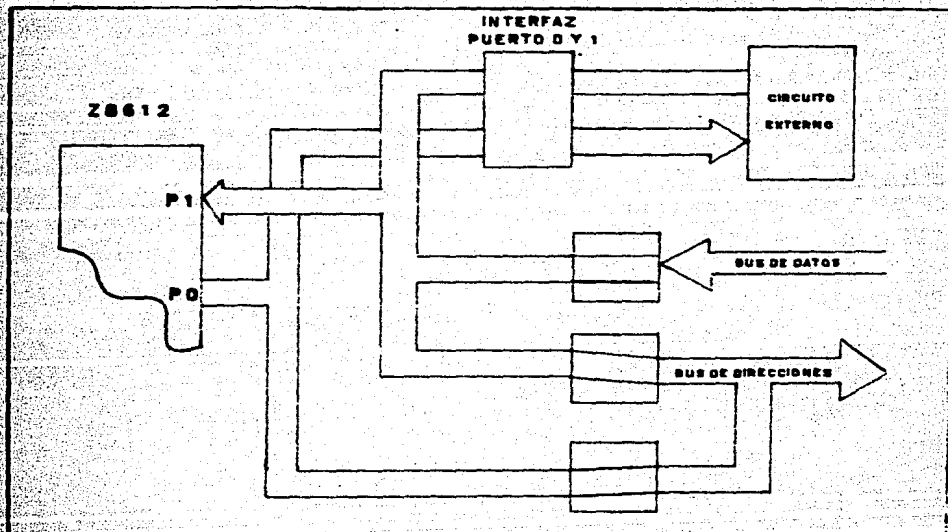


FIGURA (5.6)

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 POIM 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 AMS2B 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 AMS2B.

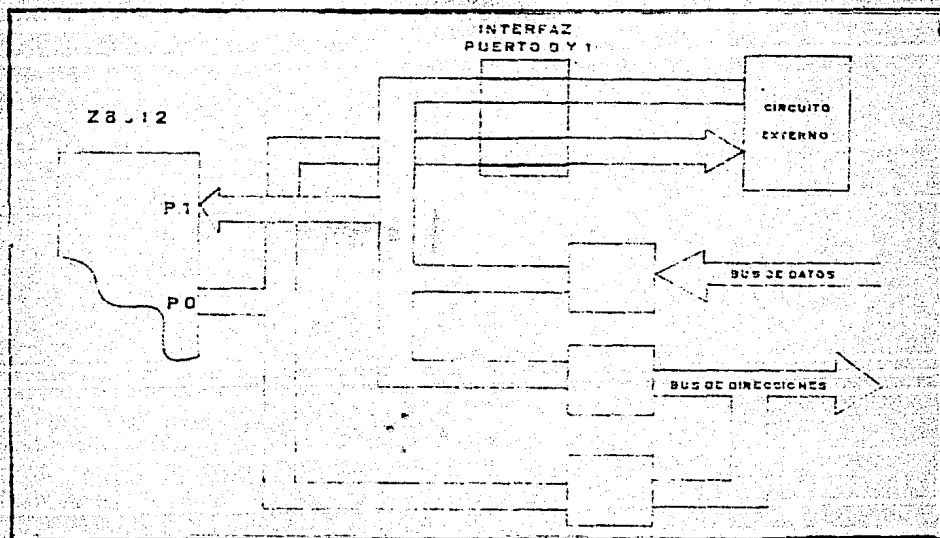


FIGURA (5.7)

Cuando el contador de programa llega a una localidad indicada como punto prueba, la lógica de Break Points produce una petición de interrupción IRGO, 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 2KB 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.

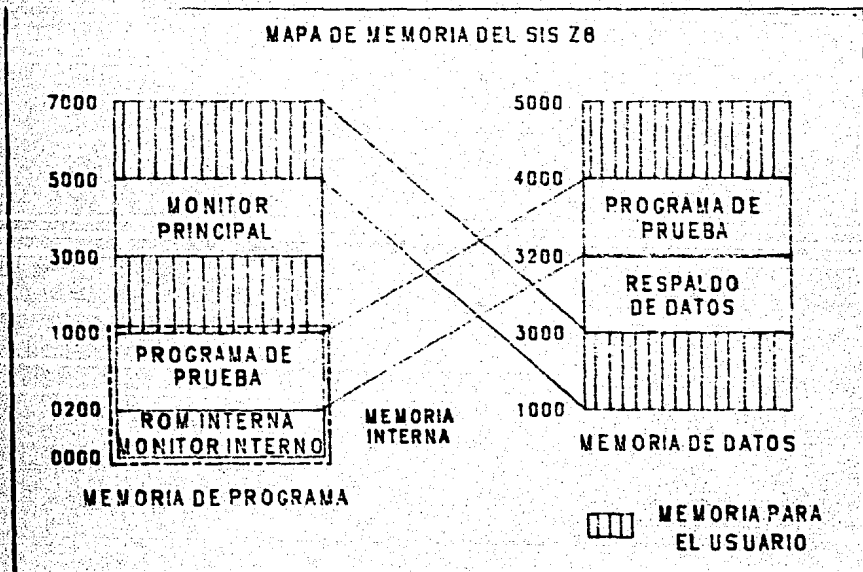


FIGURA (5.8)

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 8Kb x 8. 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 I 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 DMSZB 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 OSW de las direcciones A12-A15. Como este 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 HAB1 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 0200_H.

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 éstas selecciona a U26, a través del pin 6 de U11 en compañía de un nivel alto de DMSZB, 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 (DMSZB baja), a partir de la dirección 1000_H a la 2FFF_H (MEM1 o MEM2 de U14). La segunda es con memoria de programa (DMSZB 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 DMSZB es baja junto con MON3 o MON4.

• Simula a la memoria de programa interna del Z8 y solamente puede leerse cuando las señales HAB1 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 asíncronos 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 envío 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 ecos o quedar deshabilitada. Como ya vimos desde el capítulo 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 DMHAB que ya analizamos en la sección anterior.

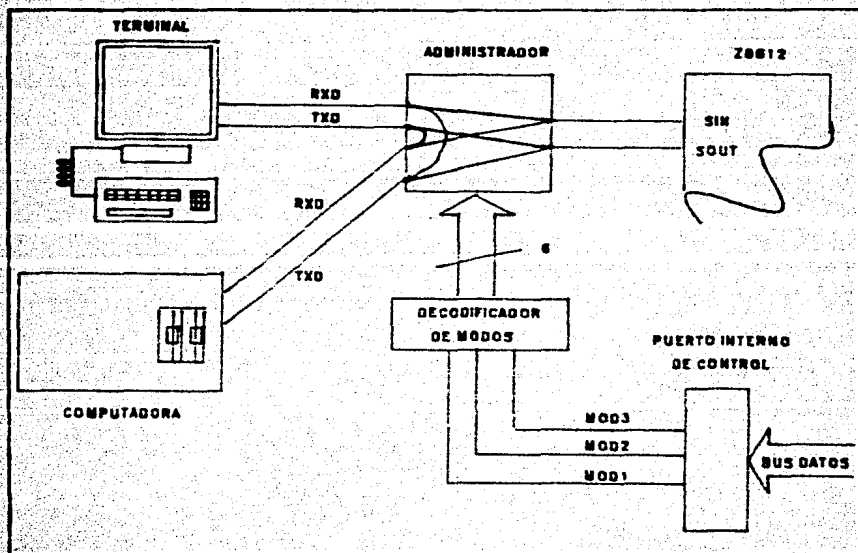


FIGURA (5.9)

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, este repite el código y lo transmite a la computadora.

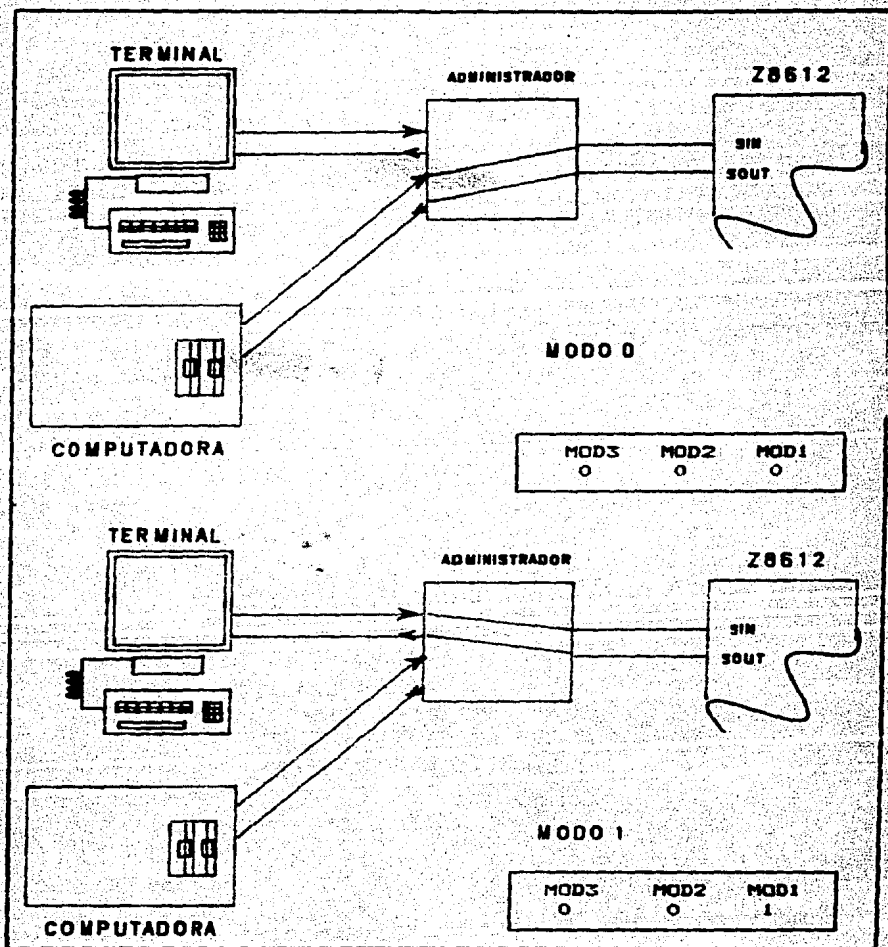


FIGURA (5.10 a y b)

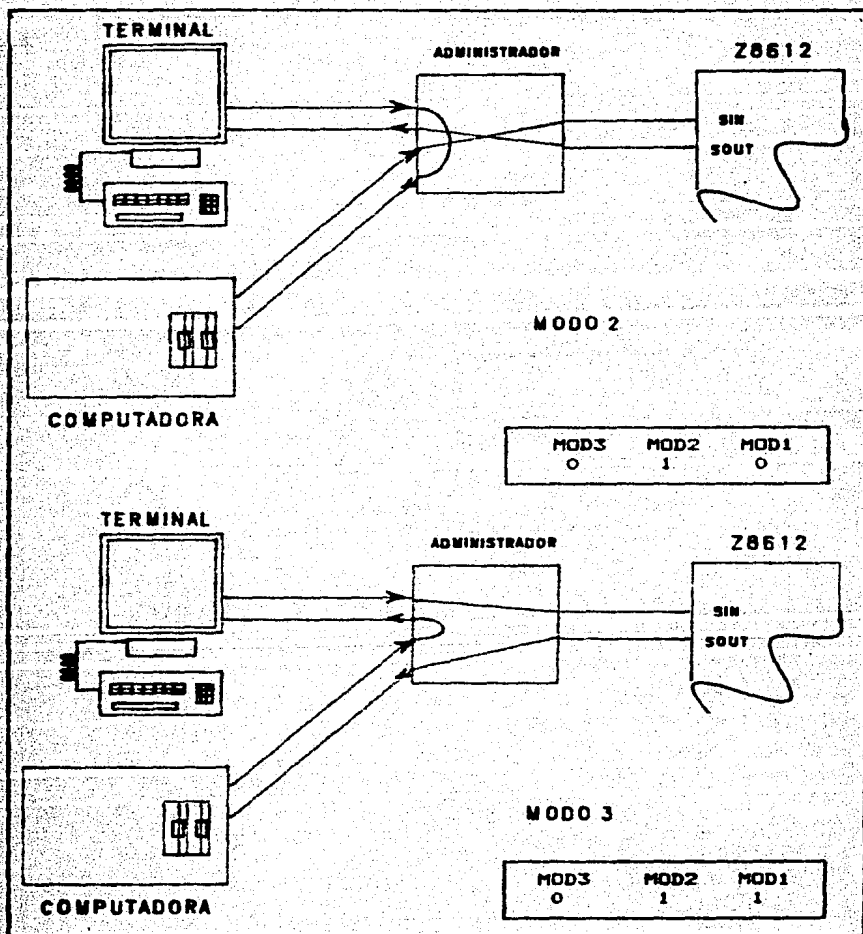


FIGURA (5.10 c y d)

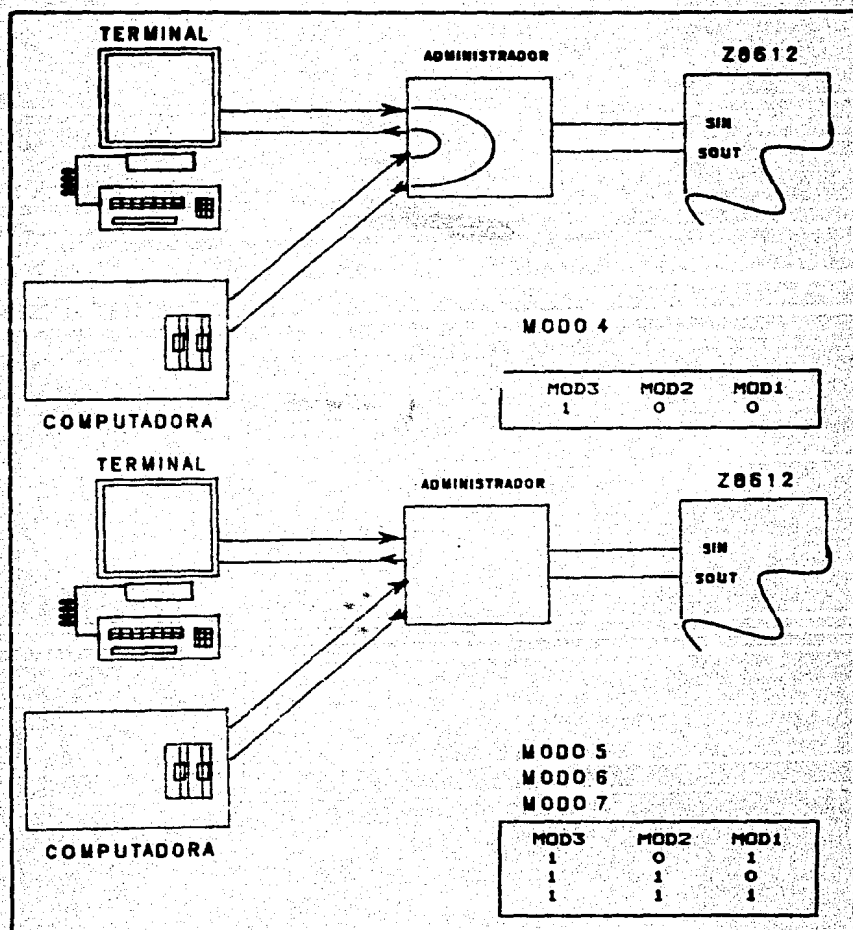


FIGURA (5.10 • y f)

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 acceder 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 las 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, U8, U7, U12 y U6. Los circuitos integrados U3 y U6 (1498 y 1499), 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, SALTSTR y SALTSTR tienen un nivel alto para interrumpir 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 SALCSEL.

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.

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 F000_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 U39, 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, fue la solución al control y rompimiento de la ejecución del programa de prueba. La solución vía programación no fue 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.

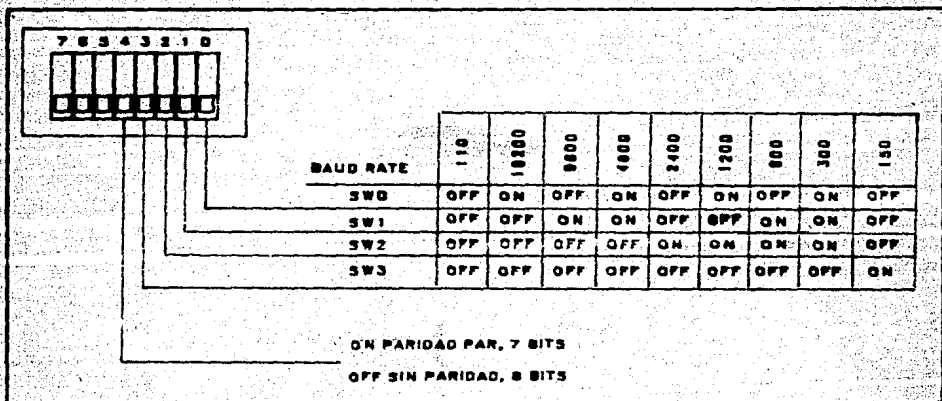


FIGURA (5.12)

La alternativa escogida es por medio de circuiteria 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 ZB, 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 capítulo, 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 capítulo 2, donde se ha añadido la señal SYNC y la operación del contador U35.

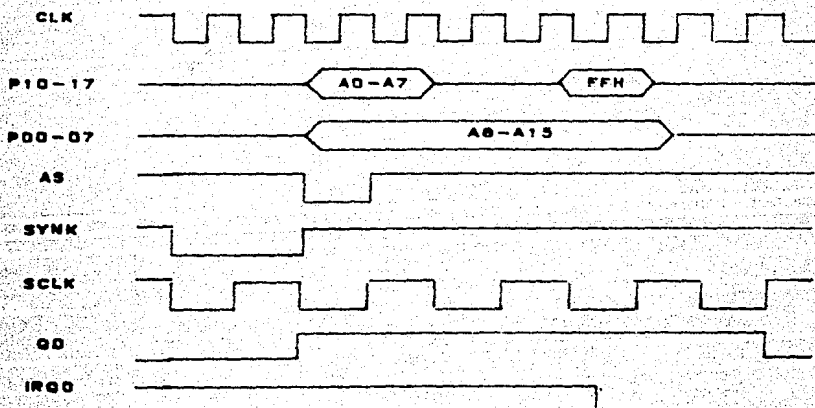


FIGURA (5.13)

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 78612. 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 FFH 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 FFH, por lo menos una de las salidas (8, 12 o 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 FFH, 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 28 programa al puerto de control interno con HT alto. Sin embargo, esto no produce rompimiento de la ejecución del SCAN 28, 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 IRQ0 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. Faltaba ahora la otra mitad de su funcionamiento: El Software del SCAN Z8, del cual nos ocuparemos en el siguiente capítulo.

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 IRQ0 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 capítulo.

CAPITULO VI

EL SCAN Z8

Este capítulo 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 mas 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 28

DECLARACION DE CONSTANTES		
M O N I T O R	I N T E R N O	PROGRAMACION DE REGISTROS DE CONTROL
		SIMULACION DE VECTORES DE INTERRUPCION
		RUTINAS AUXILIARES DE E/S PARA CONTROL DE PROGRAMAS DE PRUEBA
M O N I T O R	S U P E R I O R	CONTROL DE E/S
		MONITOR
		RUTINAS DE USO GENERAL
		COMANDOS
		SERVICIO A IRQ0
		TABLA DE COMANDOS
		MENSAJES DIVERSOS

FIGURA (6.1)

La declaración de constantes permite una mejor interpretación del código fuente, ya que los datos numéricos son sustituidos con nombres genéricos 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 0000₁₆ y 0200₁₆.

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

El monitor superior comienza en la localidad 3000₁₆ 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 implican controles de E/S, control de interrupciones y control de status. Los últimos rectángulos 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 estos, en función de los deseos del usuario.

Al inicializar o encender el SIS Z8, la lectura del programa comienza por la localidad 0000₁₆. A partir de esta hasta la 0026₁₆ 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 40₁₆.

Estas operaciones continúan aún en el monitor superior con las rutinas INIT y XS, 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 esta 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 éste.

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 INR se han deshabilitado individualmente todas las peticiones de interrupción. El caracter ensamblado es depositado en R2.

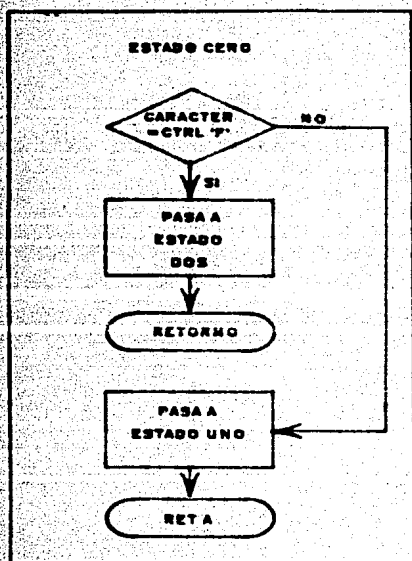
RUTINA RECEPCION.

Es la rutina más 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.

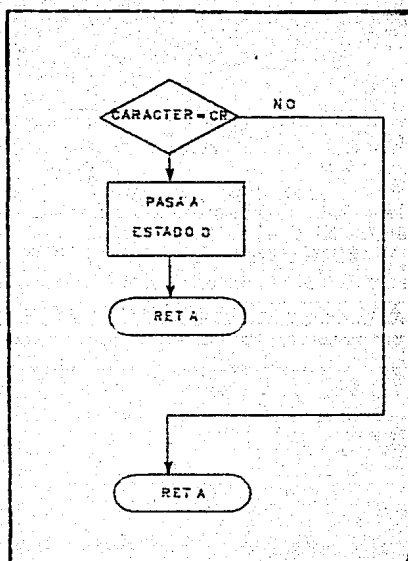
Una vez que la rutina AGUANTA ha detectado un caracter, se manifiesta la intervención de esta rutina por el eco, que ésta efectúa del caracter recibido. En el estado inicial (CERO) de RECEPCION se interpreta el caracter introducido; éste determinará el estado siguiente, siendo el Estado Dos si se tratara de un código ASCII ACK (06H 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 I 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 está 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.



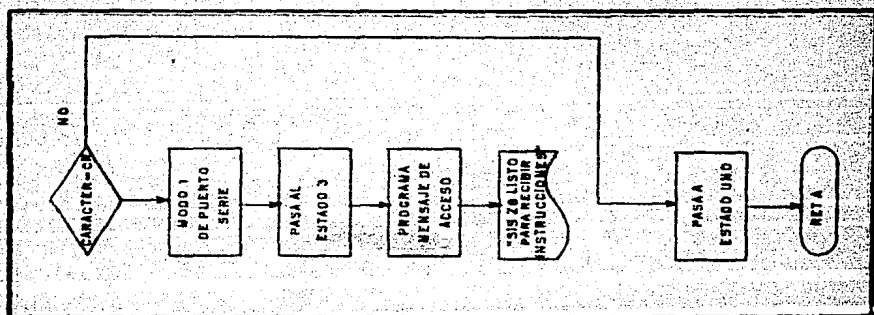
ESTADO CERO



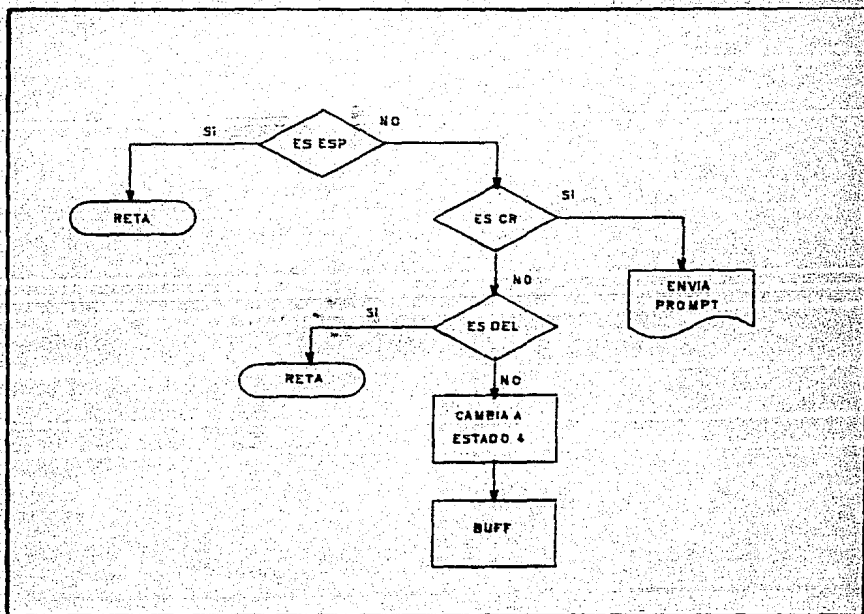
ESTADO UNO

El Estado Tres de la rutina, espera por códigos distintos a CR y ESP (20, 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.

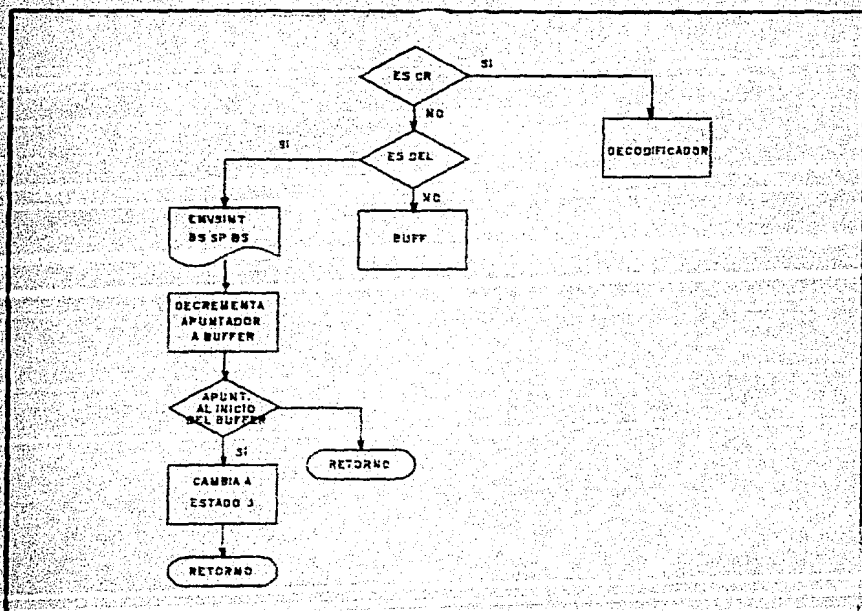
Cuando se recibe una instrucción completa se comprueba su validez a través de un decodificador de instrucciones. El decodificador hará el papel de parser, auxiliado por una tabla predefinida de direcciones y comandos. El decodificador verifica la existencia en dicha tabla de la secuencia anterior a un 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



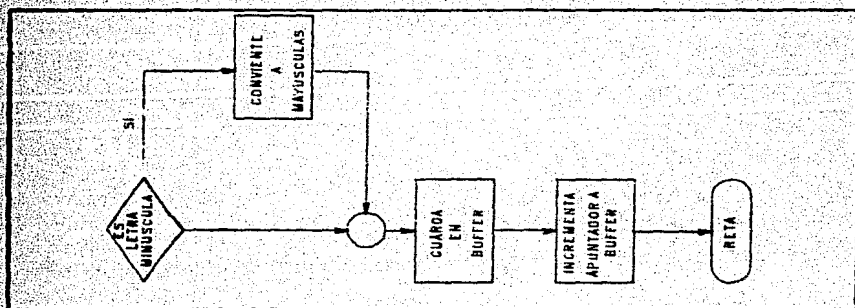
ESTADO DOS



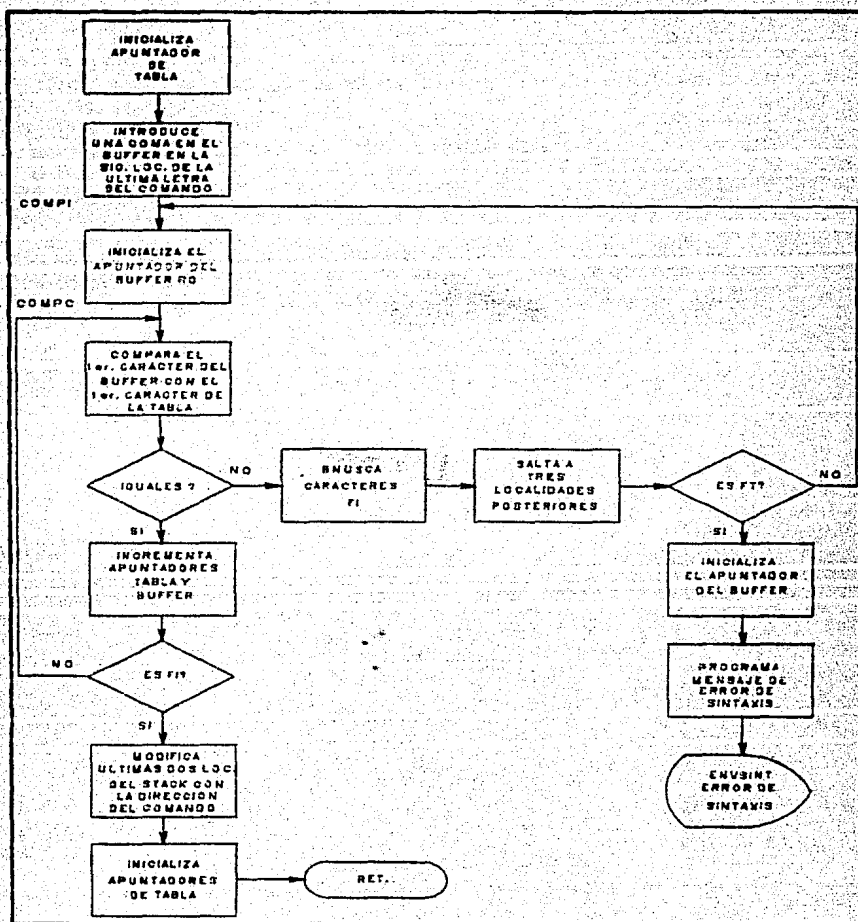
ESTADO TRES



ESTADO CUATRO



BUFF



DECODIFICADOR

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

La tabla de comandos está diseñada de manera que su barrido sea fácil 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, retornará al Estado Tres de esta rutina.

RUTINAS DEL SISTEMA.

En total existen 20 rutinas del sistema, pero solo serán tratadas las más 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 las 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 200H, de esta manera el vector correspondiente a IRQ1, ocupará la localidad 0202H, en lugar de la 0002H, y así sucesivamente. Al ocurrir una interrupción permitida por el usuario en su programa de prueba, por ejemplo IRQ1, el Z8 buscará automáticamente en la localidad 0002H, el vector correspondiente. Este apuntará 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 PREROUT (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 este 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.

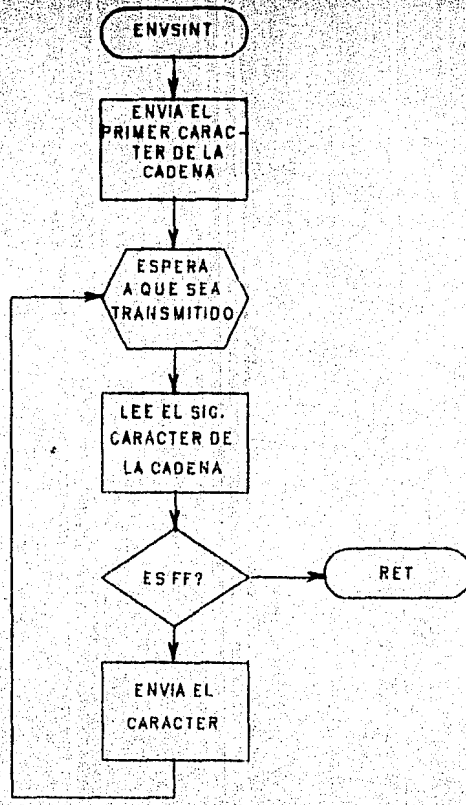
Formato:

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

RUTINA ENVBINT (Dir 32A3_H).

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

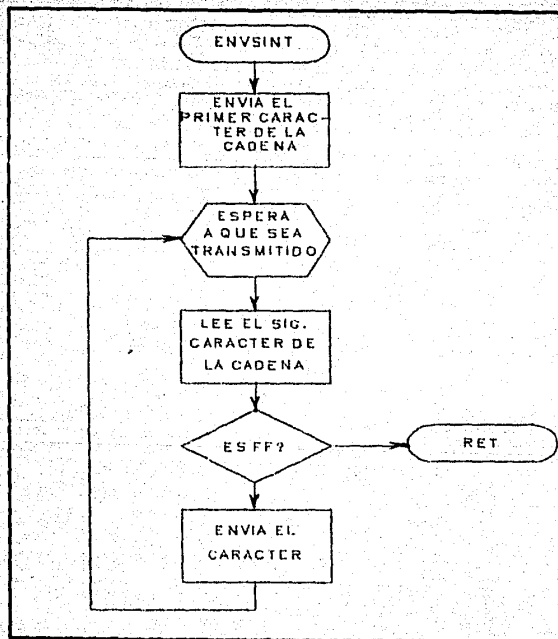
ENVSINT



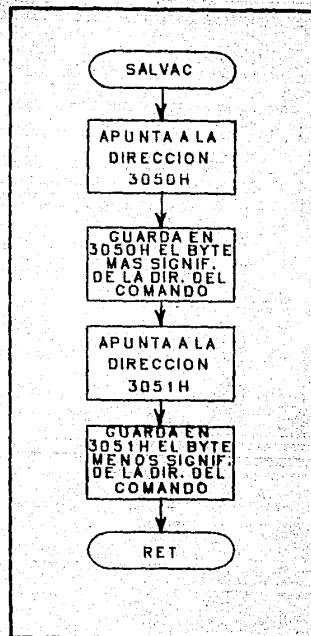
SALVAC



ENVSINT



SALVAC



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

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 envío 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 FFH.

RUTINA HEXASC (Dir 3214H).

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 5EH y 5FH.

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 30H para números y 37H para caracteres.

RUTINA ASCHEX (Dir 31D3H).

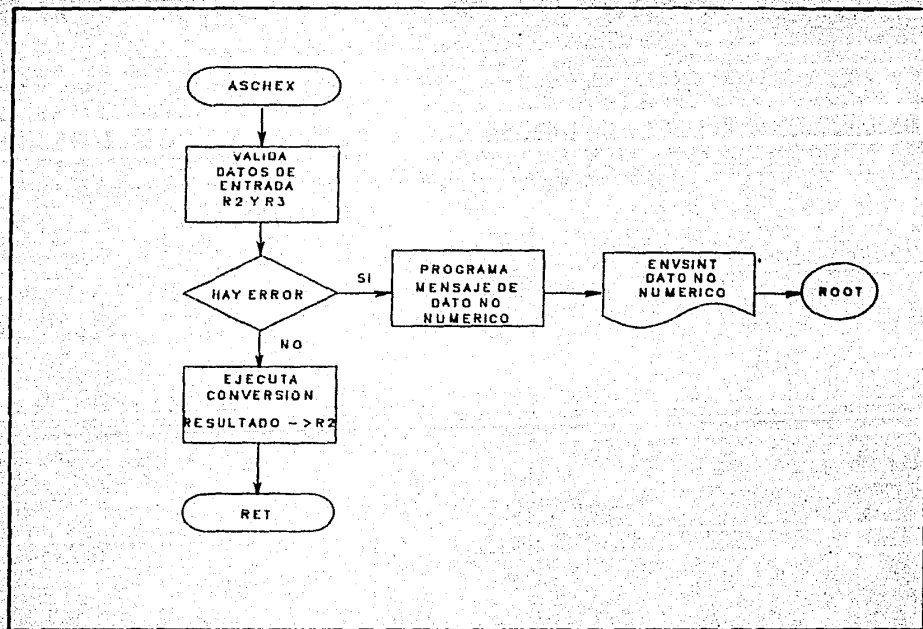
Es la rutina de auxilio que convierte el código ASCII a código hexadecimal; la operación se ejecuta sobre 2 dígitos apoyándose en los registros R2 y R3 del Grupo 1 de trabajo; el resultado de la operación es almacenado en el registro 2 del Grupo 1.

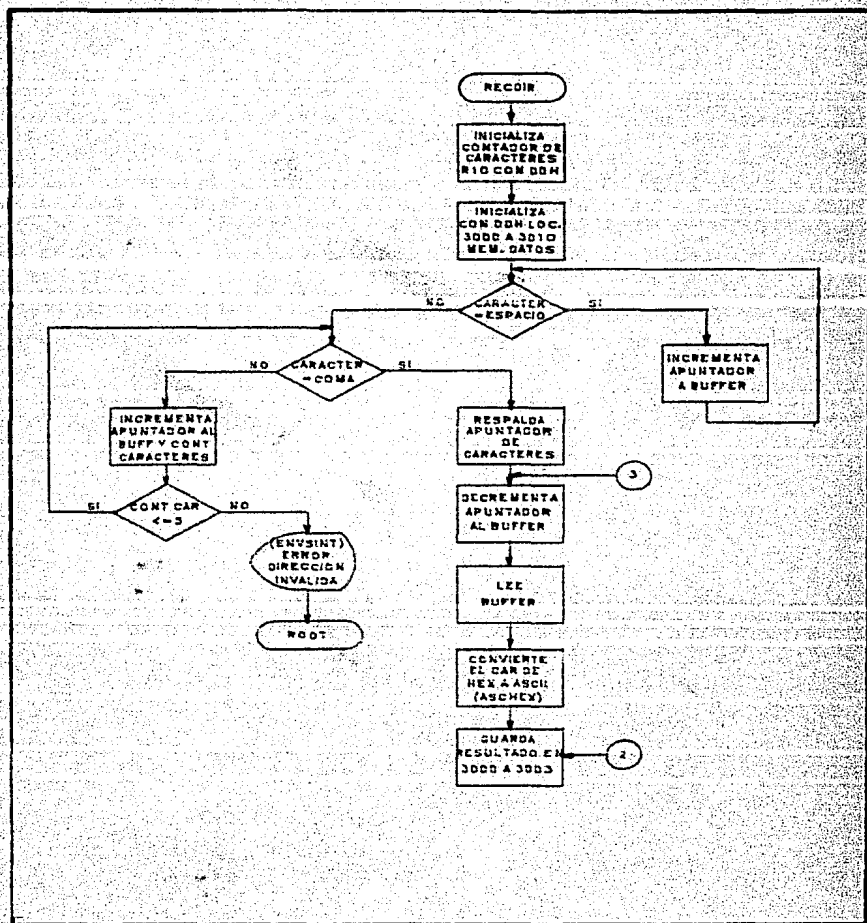
Las principales operaciones que realiza esta rutina son la verificación de los dígitos leídos, es decir, los dígitos a convertir siempre deben estar en un rango del 0 al 9 y de A a F; a estos dígitos en ASCII se les resta un offset (30H para números y 37H para caracteres); posteriormente el contenido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

RUTINA RECDIR (Reconoce Dirección, Dir 32F1H).

Esta rutina realiza el reconocimiento de las direcciones sobre las que actúan los diferentes comandos del SCAN 28. 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 (,).

ASCEX





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, ENVINT y PREROUT. 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.

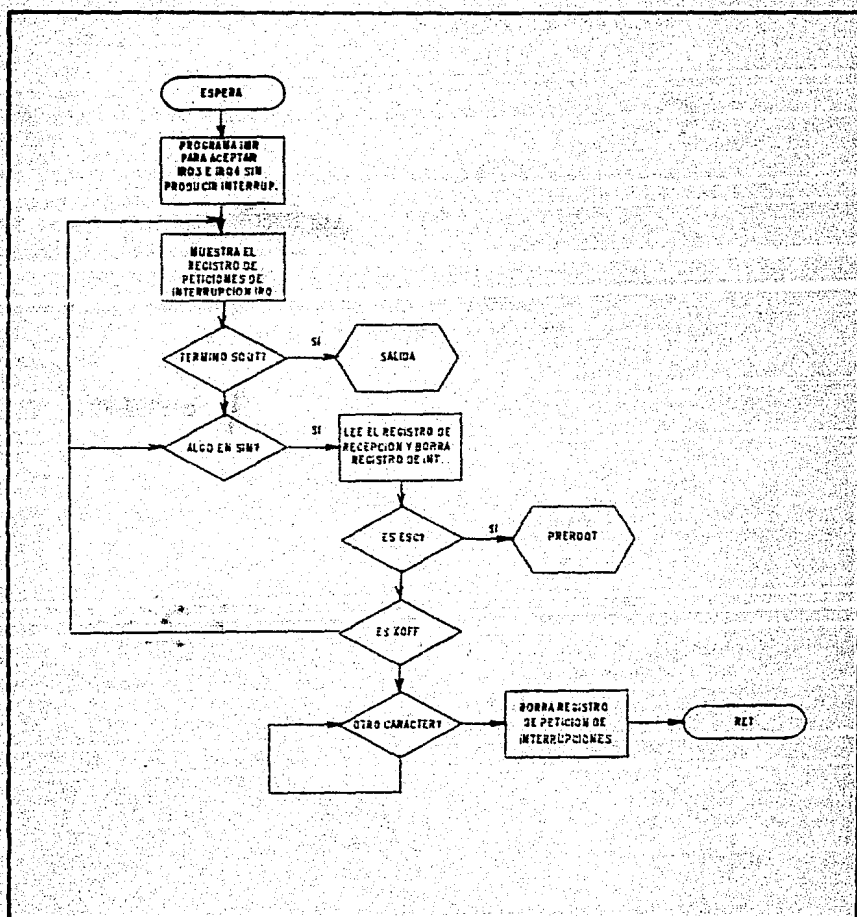
RUTINA ESPERA (Dir. 3272_H).

Es una rutina auxiliar, cuya función, mediante el muestreo del registro IRQ0, es determinar el momento en que ha concluido el envío de un carácter 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 CTRL8 (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 carácter. El único registro afectado es R2 en caso de recepción.

RUTINA ESPDAT (Dir. 3187_H).

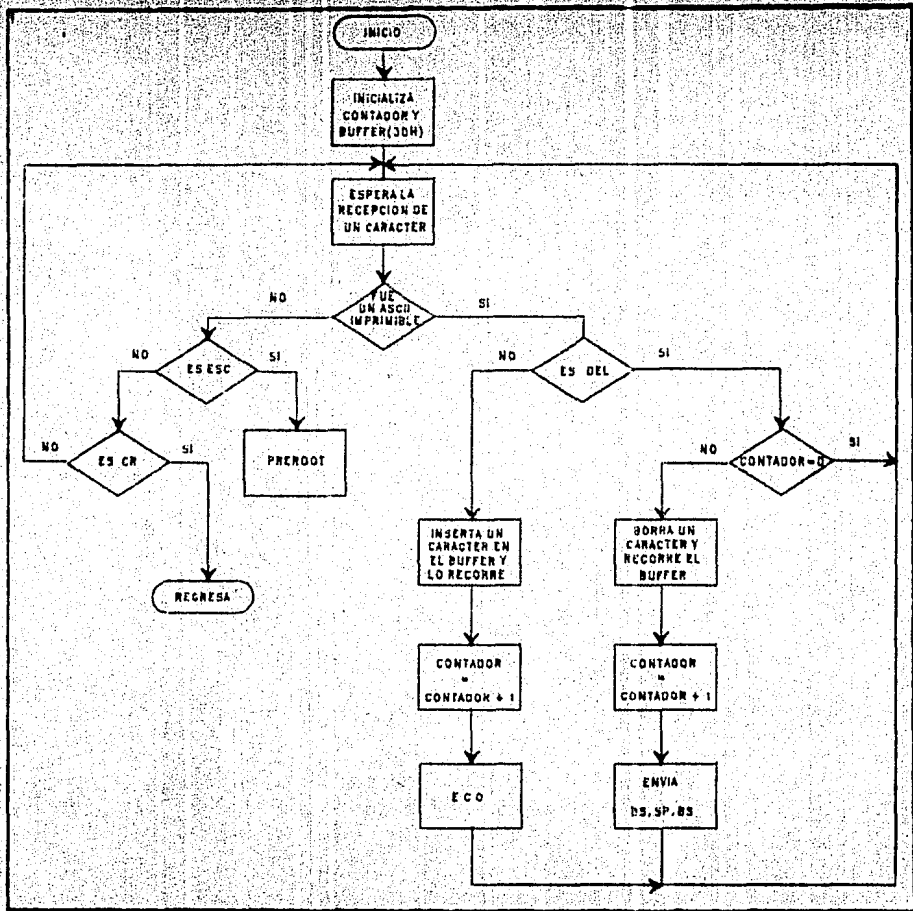
Esta rutina auxiliar es empleada por los comandos MR y SM. Se trata de un pequeño monitor, cuya función es recibir un dato hexadecimal codificado en ASCII, de la consola.

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

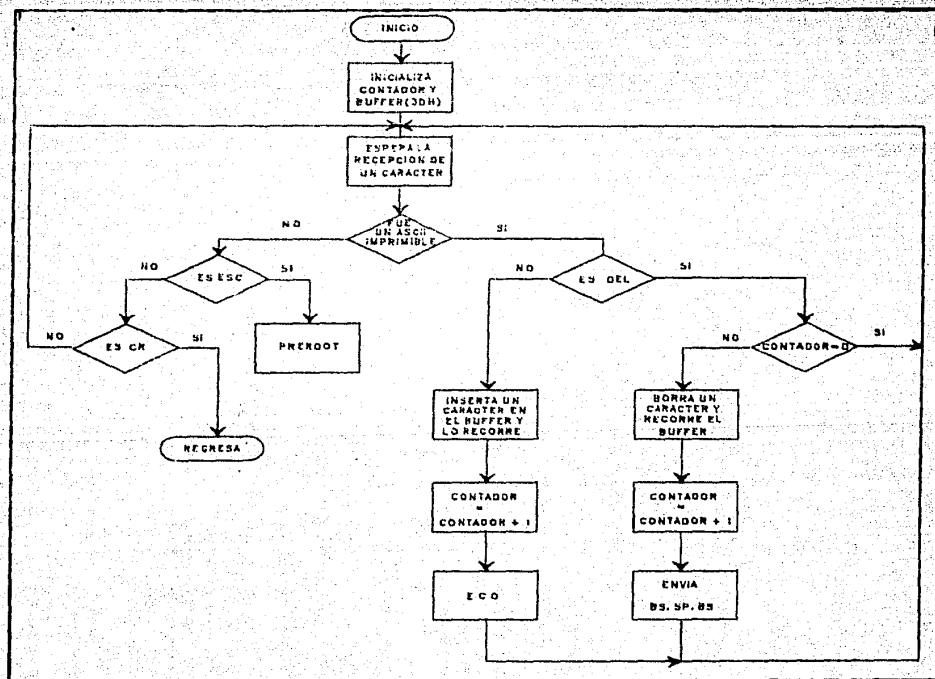


ESPERA

ESP DAT



ESPAAT



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

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

RUTINA CRLF (Dir. 325FH).

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

RUTINA ESPACIO (Dir. 32EAH).

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

RUTINA MODOS (Dir. 325AH).

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. 329CH).

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 5EH y 5FH, los cuales son introducidos en el puerto serie en la rutina OTRA.

RUTINA POD (Dir. 336FH).

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 automática en memoria de datos o programa.

PARAK (Dir. 32BFH).

Esta rutina, al igual que SALVAC, sirve para guardar información útil para el comando K. En especial ésta 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 (E000h hasta EFFFh), 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 entenderán 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 IO 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 IRQ0, se explica al final de esta sección, pero debe recordarse que no se trata de un comando, puesto que no hay manera de accederla por medio del monitor.

IV (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 28 manipula dos tipos de memoria: de Datos y de Programa.

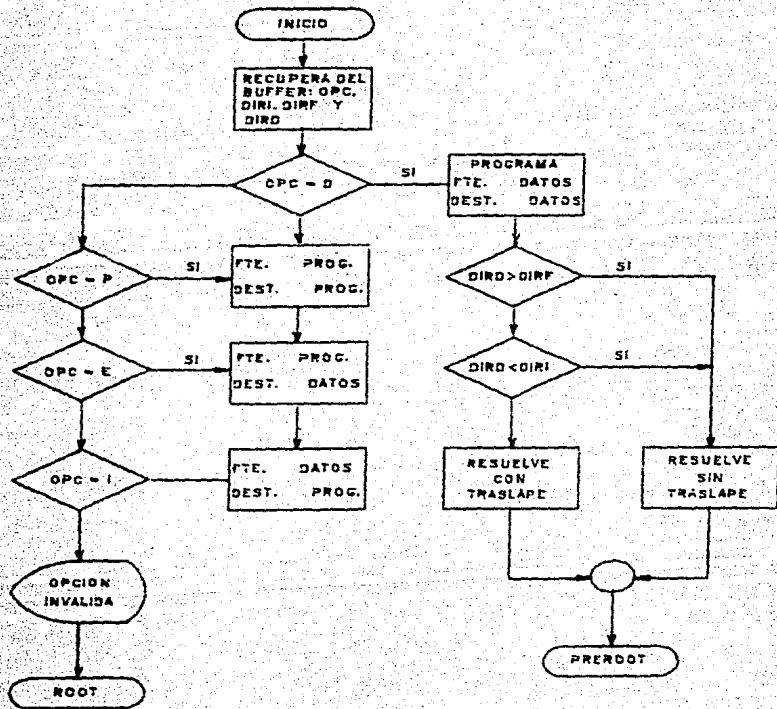
Es necesario, por lo tanto, reconocer las opciones del tipo de traspaso, en otras palabras, el tipo de memoria origen y el tipo de memoria destino; para ello la rutina preestablece cuatro letras distintas, siendo

- 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.

También es indispensable reconocer tres direcciones: el inicio del bloque, el final y la nueva dirección destino.

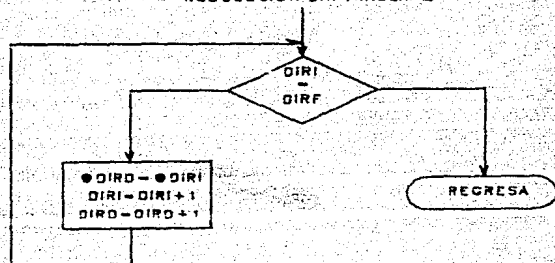
Se auxilia en la rutina de reconocimiento y validación de direcciones y utiliza apuntadores a cada una de las direcciones mencionadas, interpreta las direcciones y determina si existe traslapo para iniciar con el movimiento de contenidos.

La rutina no envía letreros, a menos que existan errores, simplemente regresa un prompt una vez que ha ejecutado la transferencia.

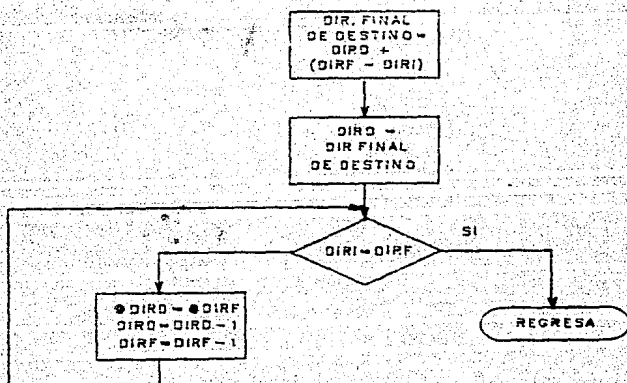


OPC: OPCION
 DIRI: DIRECCION DE INICIO
 DIRF: DIRECCION FINAL
 DIRD: DIRECCION INICIAL DE DEFINITIVO

RESOLUCION SIN TRASLAPE

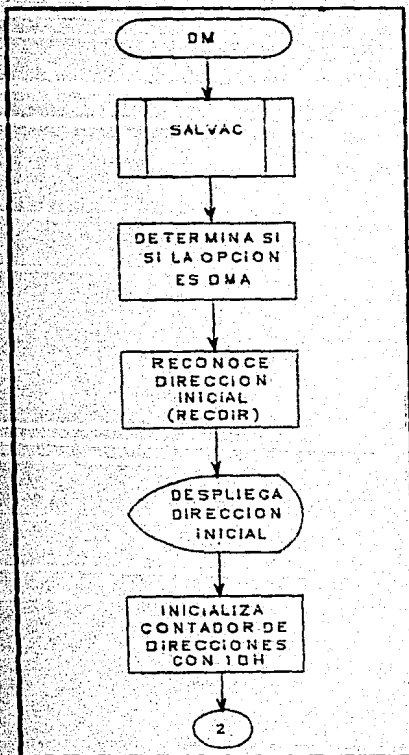


RESOLUCION CON TRASLAPE



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 desplegará memoria de datos y el comando original (DM) actuará sobre memoria de programa. La opción es determinada por la rutina POD.



DM - I

La rutina está habilitada para reconocer dos direcciones, una inicial y otra final, reconociendo de esta manera el rango de direcciones cuyo contenido estará bajo visualización. Las direcciones no deben omitirse y deberán estar separadas por una coma. Se apoya en la subrutina de reconocimiento de direcciones (RECDIR), la cual actúa sobre datos en hexadecimal.

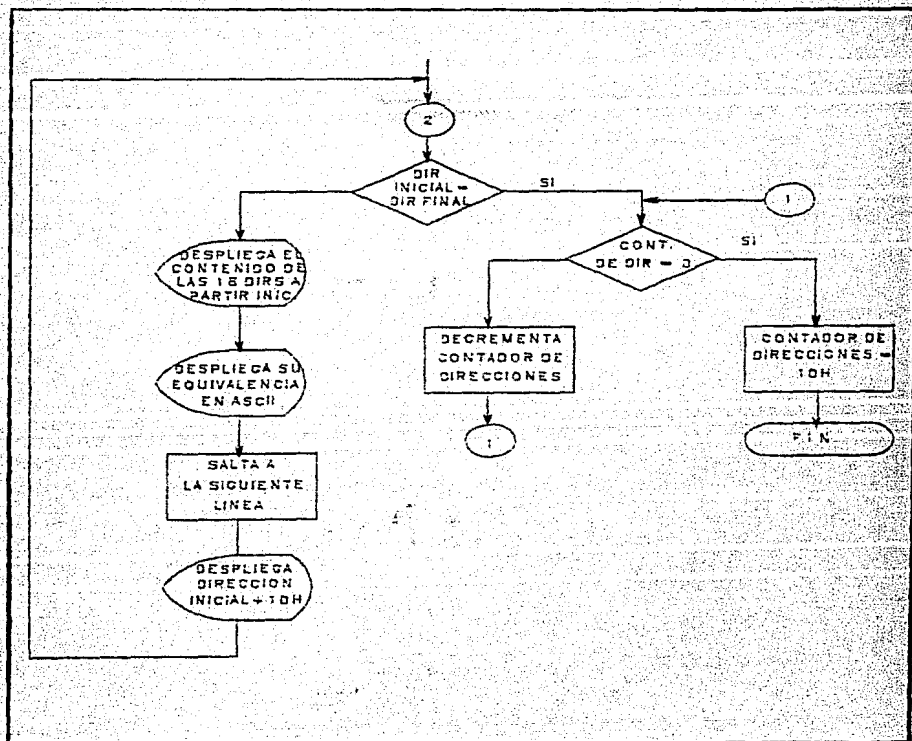
Una vez que se determina el rango de direcciones (recordar también que el despliegue de direcciones entre F000_H y FFFF_H bloqueará al sistema), se procede al envío de información a pantalla. La rutina tiene necesidad de utilizar la mayoría de los registros del grupo uno, debido a la complejidad en la organización del despliegue de contenidos y direcciones ya que se requerirán contadores y apuntadores entre otros elementos.

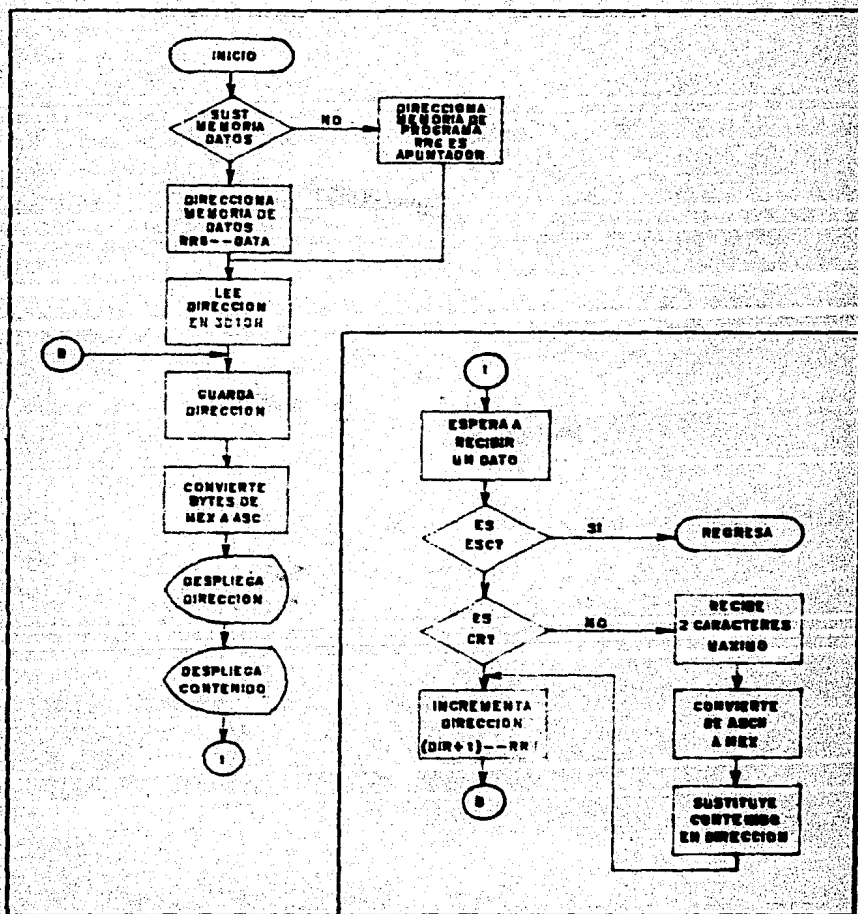
Se despliega la dirección inicial seguida del contenido en hexadecimal de esa dirección y 15 subsecuentes, además se envía el contenido en caracteres ASCII, mandando a pantalla un punto representativo de los caracteres no imprimibles; esta operación se repite hasta cumplir con el envío de la dirección final indicada.

Al usar las rutinas ENVGINT

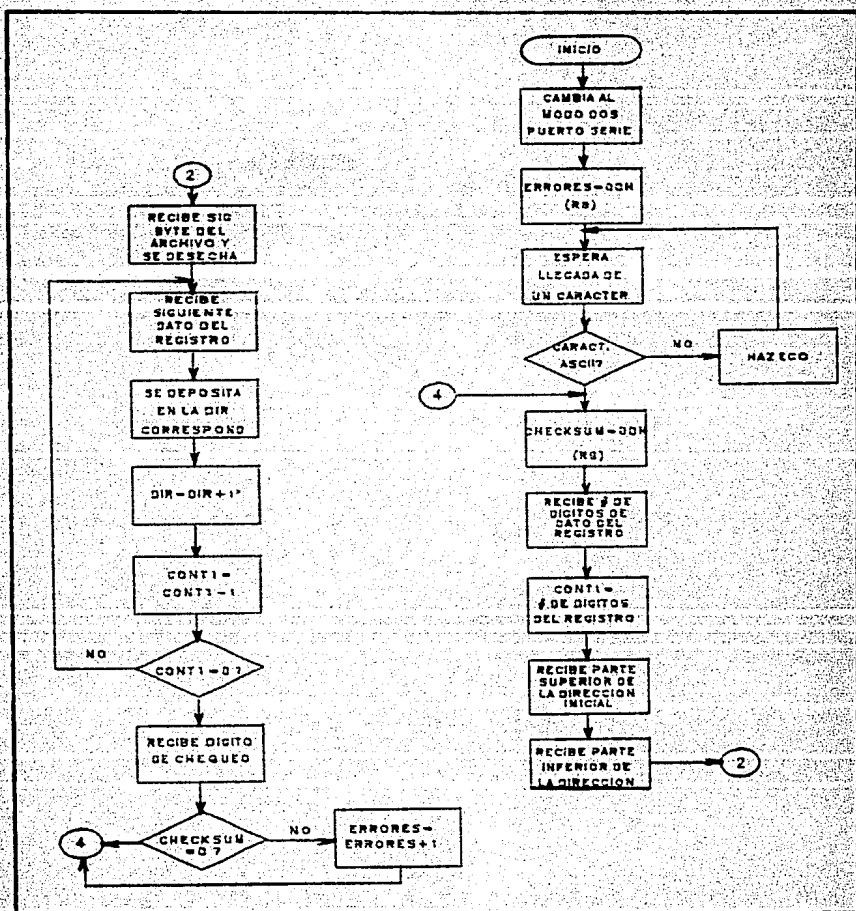
y ESPERA, este comando puede ser interferido por los caracteres de control CTRLS y ESC, teniendo como consecuencia la suspensión momentánea y definitiva del comando respectivamente. La visualización de las siguientes 256 localidades puede ser conseguida con la rutina K (continuación).

Otras subrutinas de apoyo para DM son las de envío de caracteres (ENVSINT), reconocimiento de direcciones (RECDIR) y respaldo de direcciones (SALVAC Y PARAK).

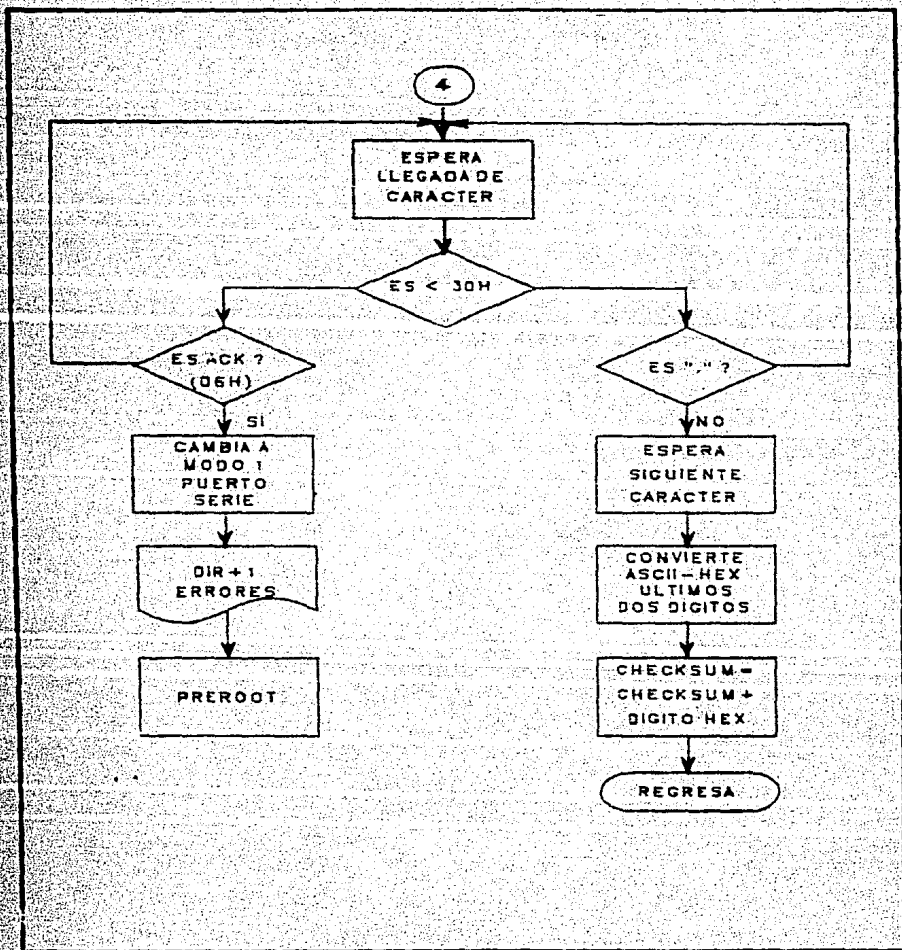




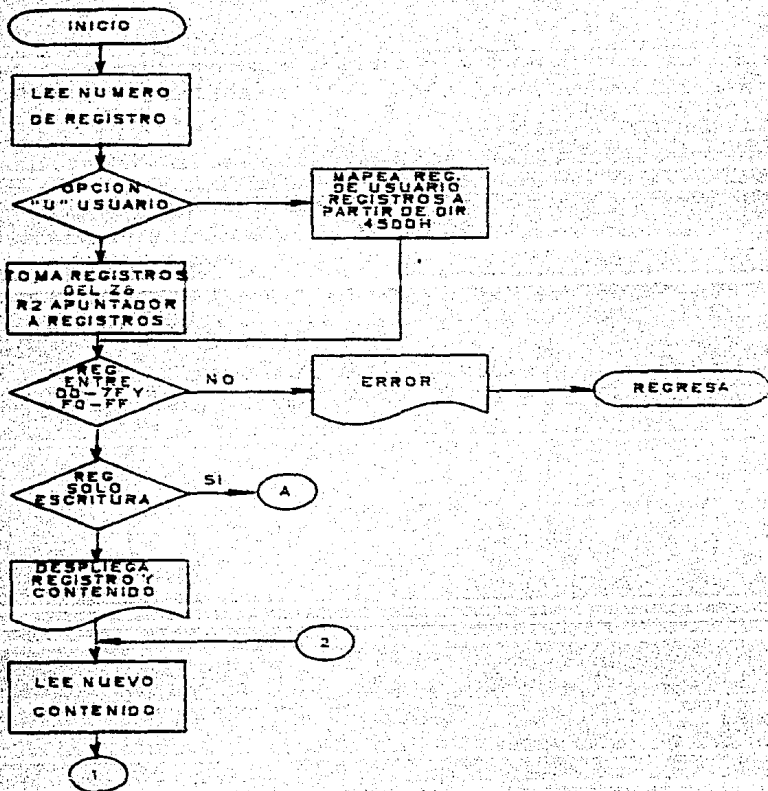
SM

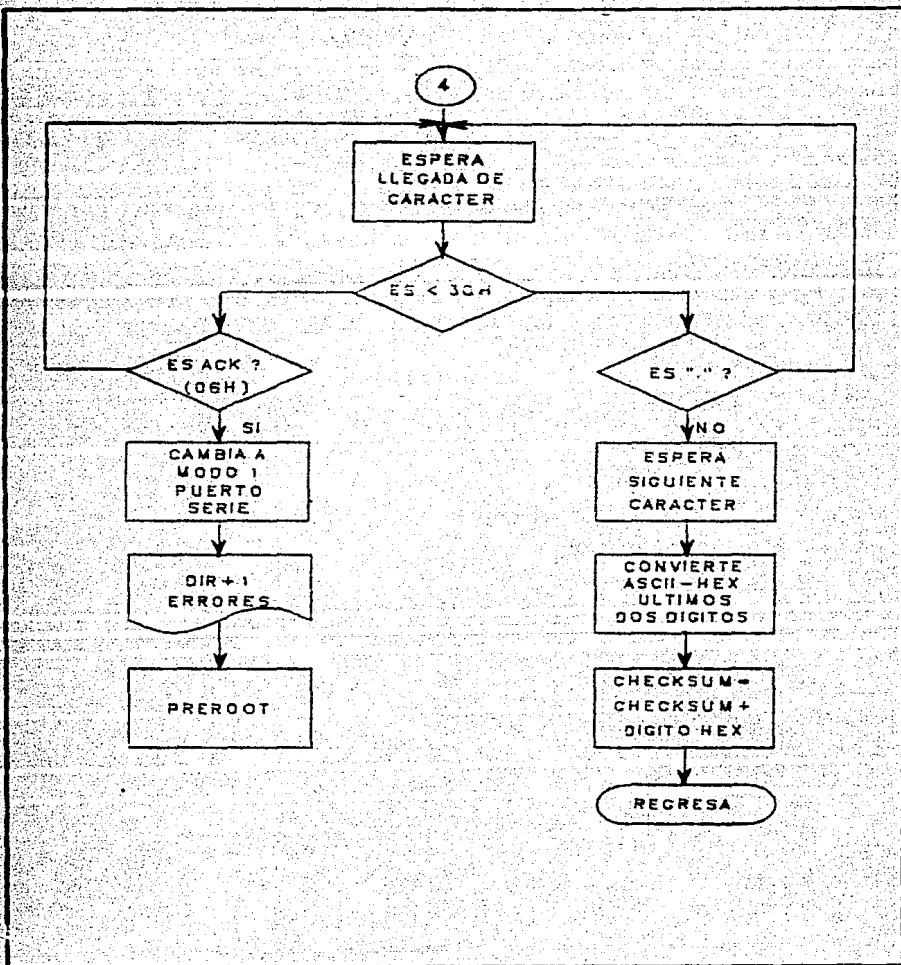


TRM



TRM-1





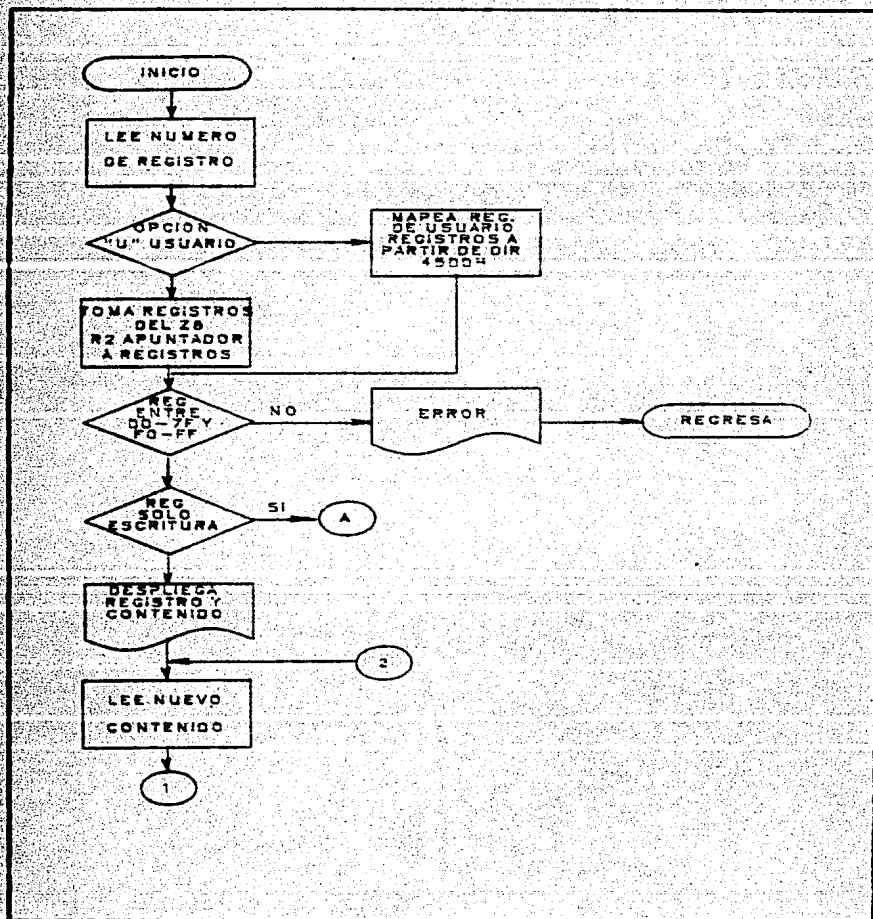
MR y MRU (Movimiento de Registros).

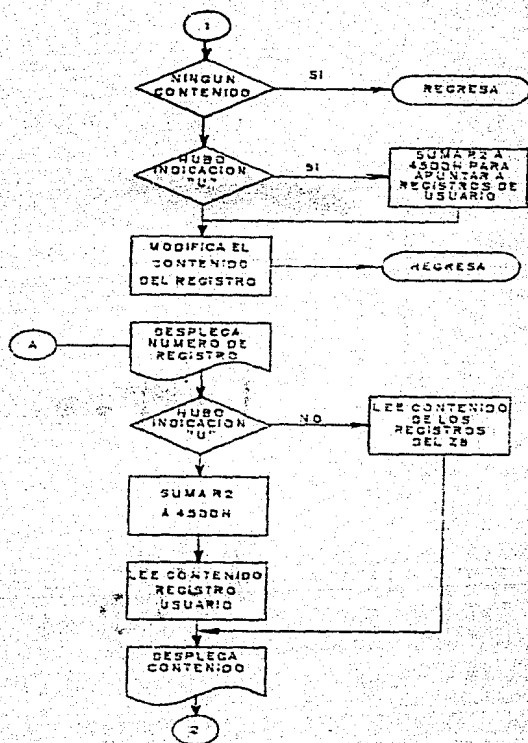
Esta rutina hace posible la modificación del contenido de los registros del Z8, incluyendo los de control. Considera dos alternativas: una se refiere a la modificación de registros de usuario, para lo cual hace referencia al archivo alterno de registros en memoria de datos externa, la otra opción concierne 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 envío 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 códigos.

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 envía un código de escape.





MR-II

DR (Despliegue de Registros).

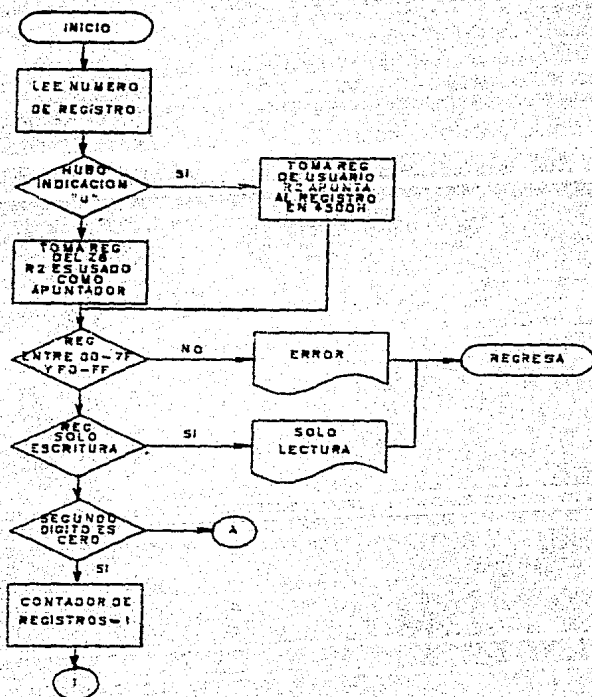
Esta rutina permite la visualización de uno o varios registros del Z8, es decir, la rutina puede enviar a pantalla un solo registro o un grupo (16), dependiendo de la indicación realizada después de las letras DR o DRU.

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.

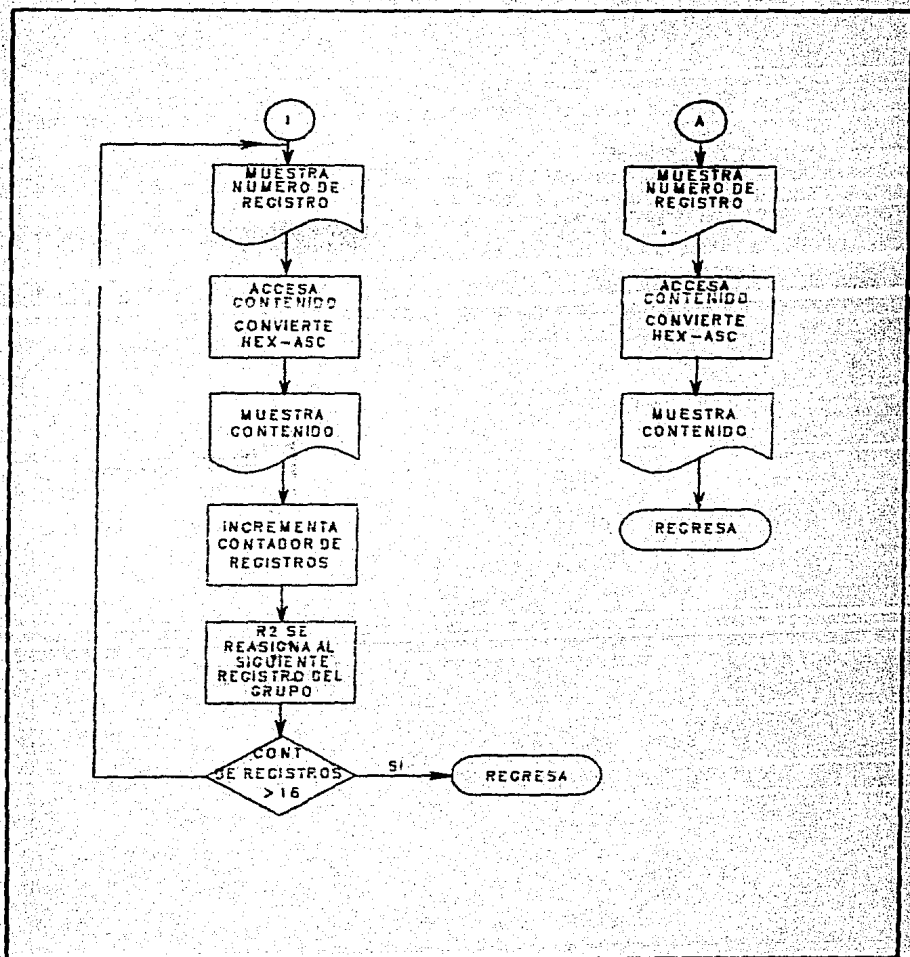
La rutina se auxilia en subrutinas como las siguientes: reconocimiento y lectura de un registro dado en forma hexadecimal, cuando el último dígito de este número sea cero desplegará todo el grupo correspondiente al primer dígito, cuando no sea así se indicará únicamente el contenido del registro referenciado 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 envío 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 (RB) para enviar todos estos.

Puede existir inconsistencia en el desplegado de registros de control (grupo F) debido a que algunos de ellos son registros de solo escritura, mientras que otros son dobles.



DR-I



DR-II

MA (Manda)

Es la rutina que se encarga de probar el correcto funcionamiento de transmisiones a través del puerto serial.

Ejecuta el envío de todos los caracteres ASCII imprimibles, desde el espacio (20h) hasta el "~" (7Eh).

Se auxilia de los registros 14 y 15 para el envío 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 indicaciones 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 envío 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 J, a fin de utilizarlos como contadores y apuntadora 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 -
QA H

CONTADOR DE
DATOS -
QD H

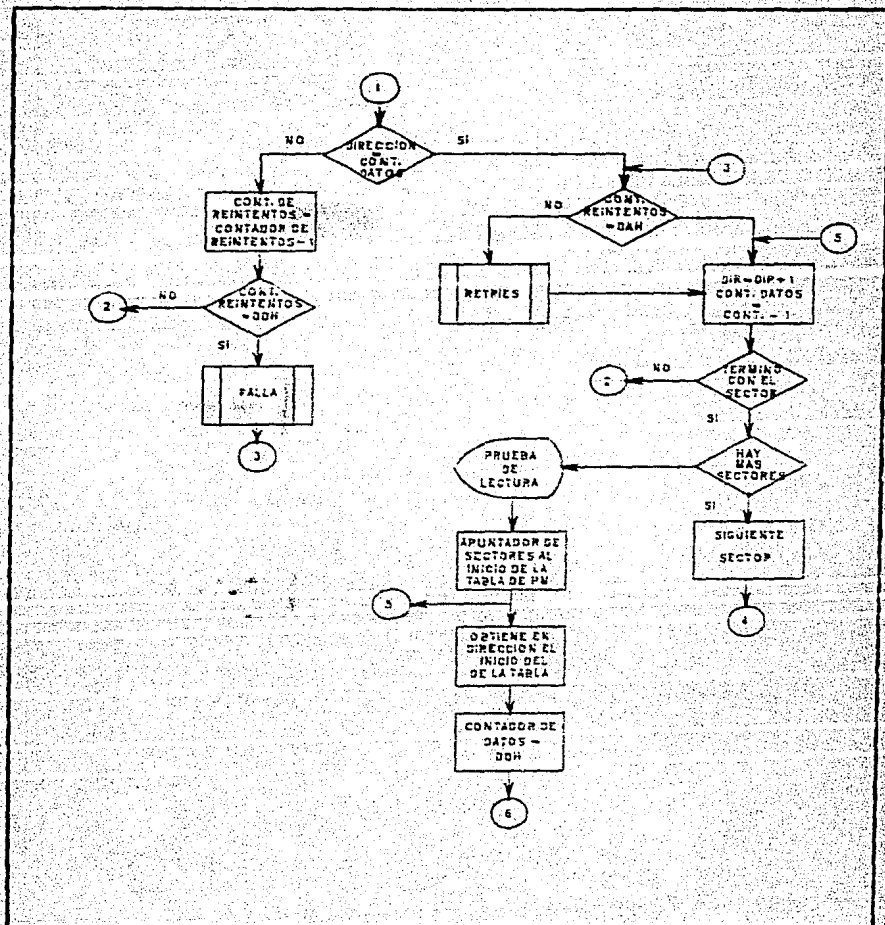
2

GUARDA CONT.
DE DATOS EN
DIR. DE SECTOR
RELATIVO

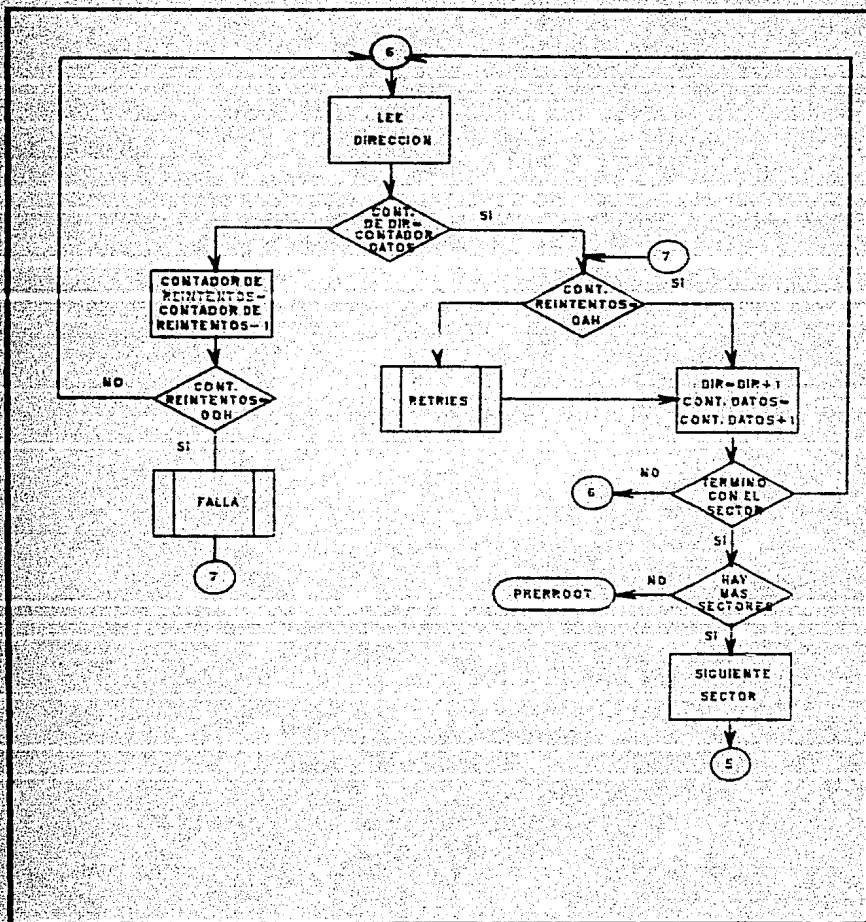
LEE DIRECCION
DE ACABA DE
ESCRIBIR

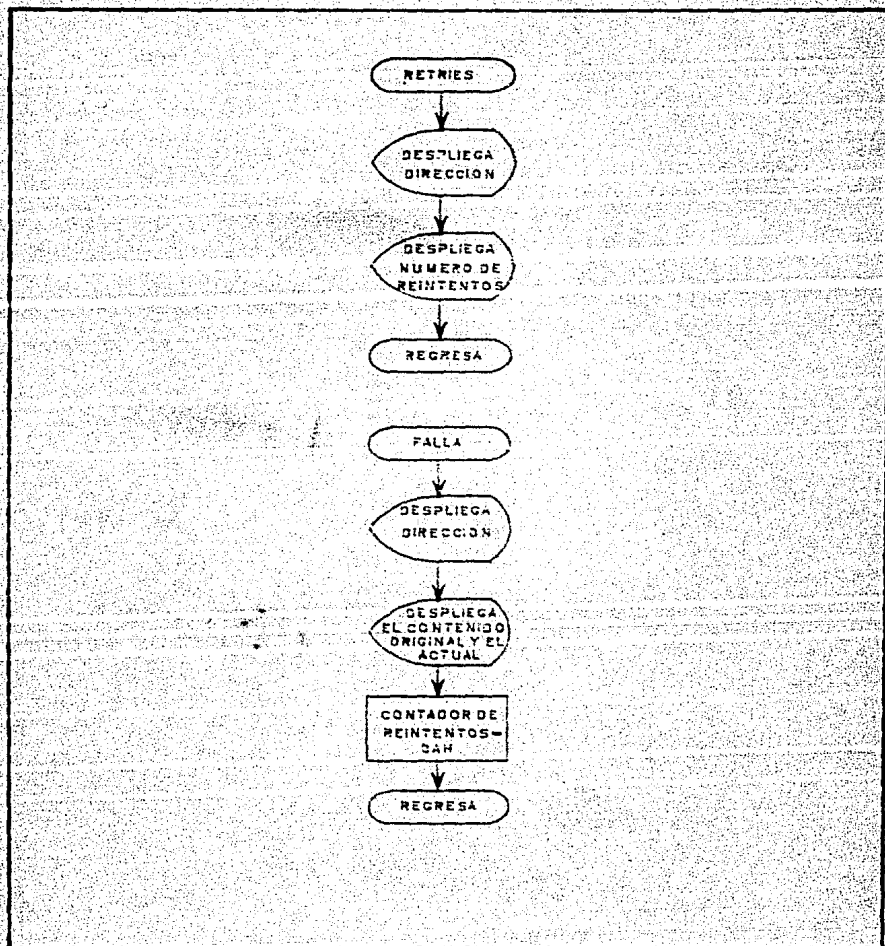
1

PM-I



PM-II





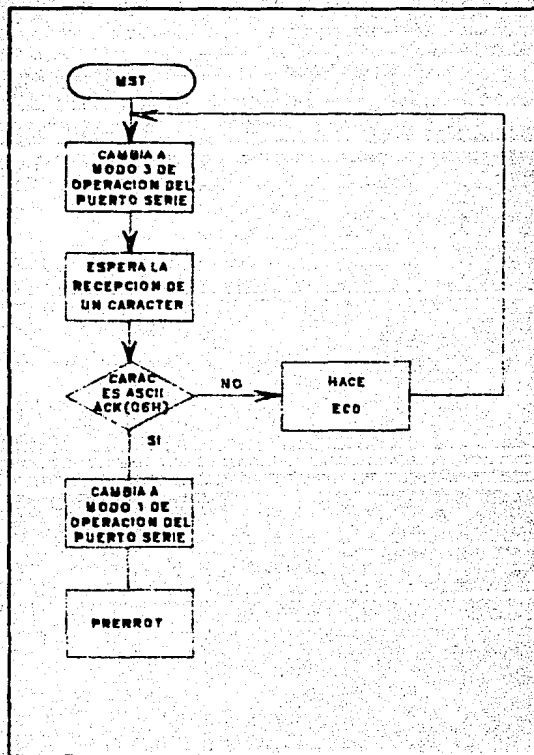
PM-IV

MST (Maestro).

Esta subrutina se encarga de configurar el puerto serie del SIS-28, 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.

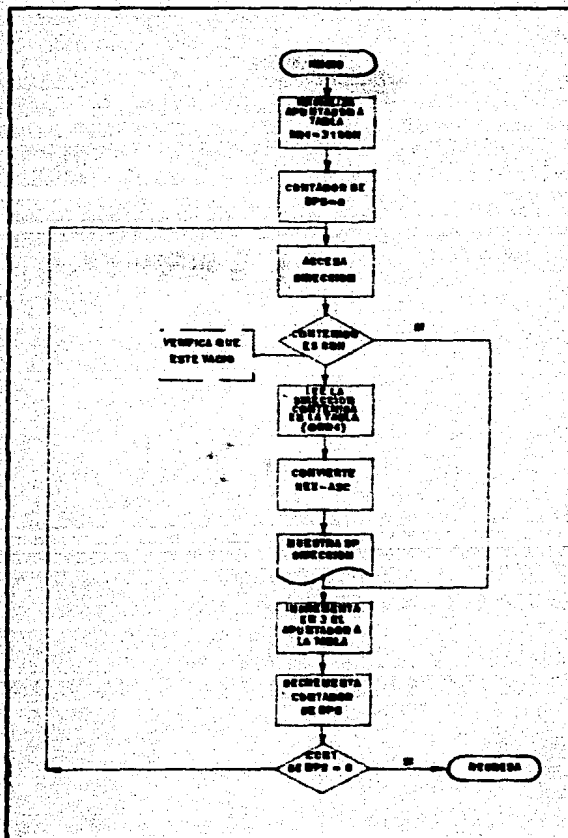
Es obvio que este modo de operación deshabilita las funciones del SIS-28, maniobrando éste únicamente en espera de la clave de retorno al monitor.

Durante este ciclo de espera, la rutina lee los caracteres enviados por la terminal retransmitiéndolos si no se trata de un CTRL F, cuando ha leído este carácter, 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-28.



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 3100H de datos.

La rutina revisa cada uno de los ocho lugares posibles para guardar los Puntos de Ruptura, desechando aquellos que se encuentren vacíos y enviando a la pantalla las direcciones donde se encuentren localizados los Puntos de Prueba.

Esta rutina se apoya en las rutinas generales de envío de caracteres.

BP (Break Point).

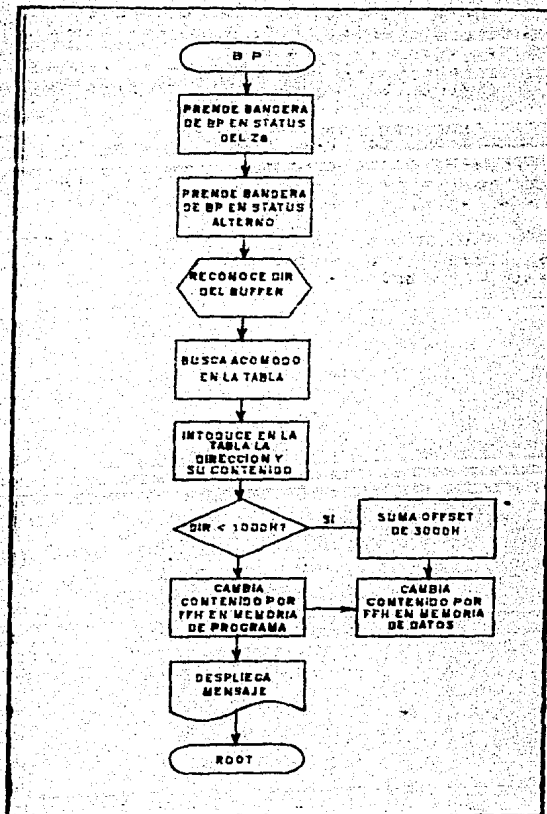
Este comando es usado para colocar puntos de ruptura en localidades incluidas en el rango de direcciones ocupadas con los programas de prueba. Los puntos de ruptura tienen como objetivo el detener la ejecución del programa y mostrar el estado de los registros.

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 FFH.

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.



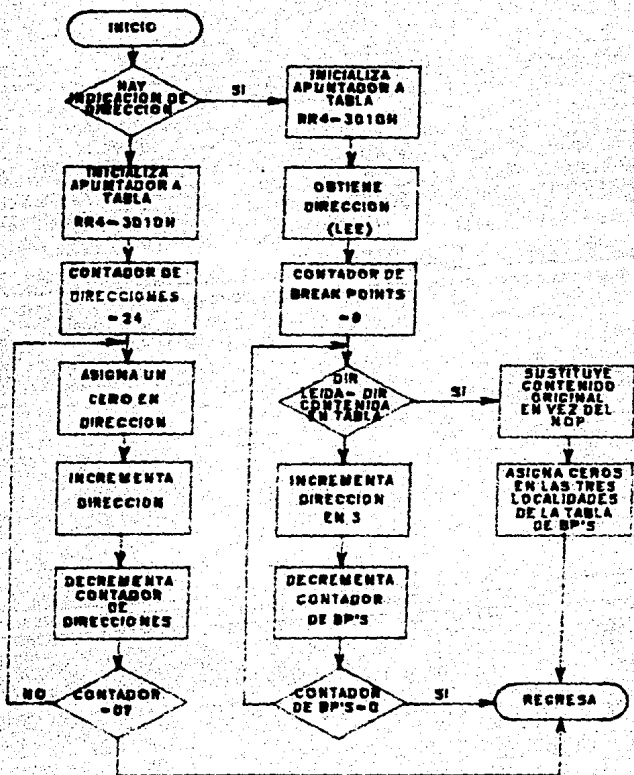
BP

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 específica y en el segundo, en caso de no existir tal indicación, ejecuta la limpieza de toda la tabla.

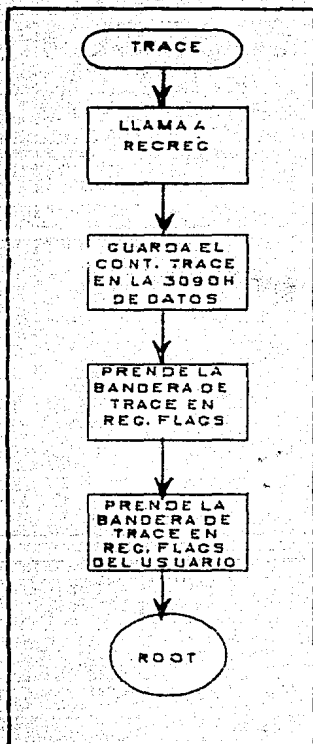
Como se ha mencionado, una vez indicada una dirección específica, verifica que exista en la tabla, y si este es el caso reintegra el código original del programa en lugar del colocado por la rutina de BP (FFH). Obviamente genera un error en caso de no existir un punto de ruptura en la dirección indicada.

Cuando no se le indica una dirección específica coloca 00H en todas las direcciones que abarca la tabla de Break Points (3100H en adelante).



BX

T (Trace).



TRACE

Es la rutina utilizada para la predisposición de ejecución de programas paso a paso o instrucción 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).

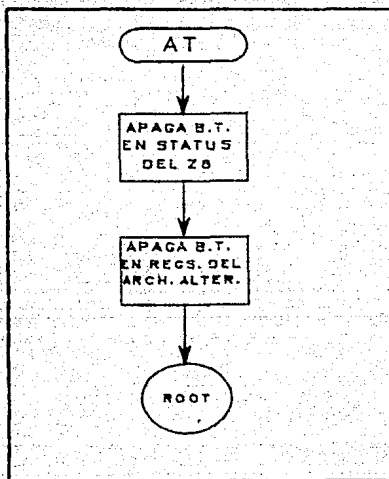
La rutina reconoce el número de instrucciones a ejecutarse paso a paso, a través de la rutina de reconocimiento de dos dígitos hexadecimales (RECRES), utiliza el último bit del registro de estado o banderas para indicar que la opción de Trace está activa.

El número de instrucciones que ejecuta paso a paso es respaldado en la dirección 3090_H de datos y se va decrementada cada vez que se ejecuta una interrupción por IRQ0.

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/SP).

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.

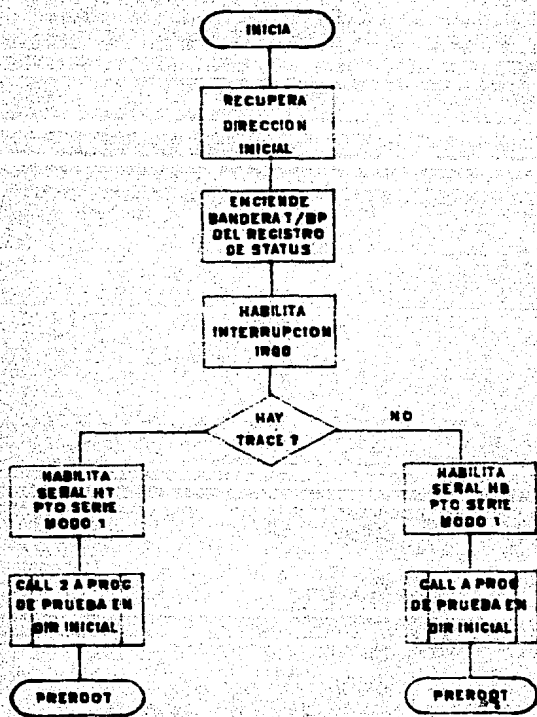
La segunda etapa o TAMCAK prende la bandera de CALL/GO en el bit 2 del registro de status y programa los registros R8 y R7 como sus equivalentes F8_H y F7_H respectivamente. Esto último se hace con el objeto de auxiliar a la rutina de servicio a IRQ0, ya que ésta es inicialmente "ciega" en cuanto que 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 fue 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 acceder 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.



CALL

GO (Arranque de un programa de prueba).

La primera operación de este comando es reconocer la opción [I]. Si esta es válida, 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 fué 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 TANCAK, 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 esto 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 I debe ser habilitada en el puerto interno de control y procede a la transferencia de registros del archivo alterno.

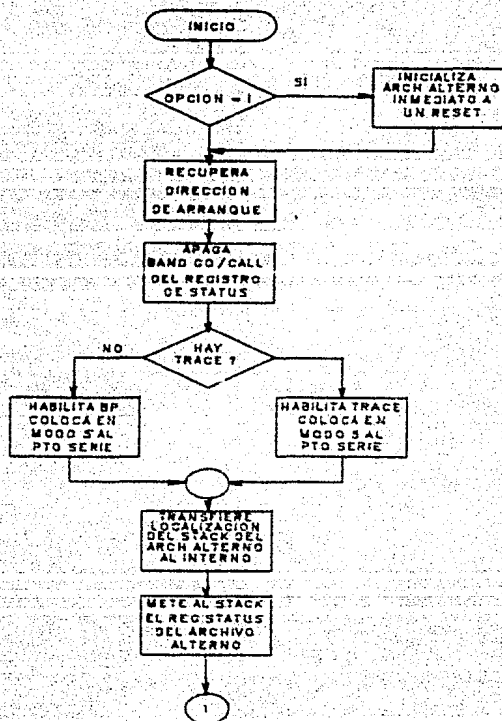
Inicialmente pasa el apuntador al stack de los registros alternos correspondientes, así también determina del registro FB_W alterno la localización del stack.

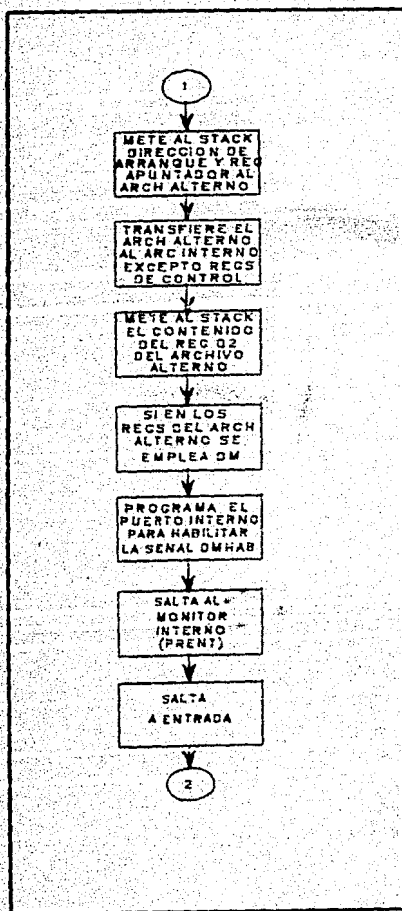
Con el apuntador del stack en la última localidad antes de una interrupción, introduce la nueva dirección, el registro de status, además del registro RP del archivo alterno y posteriormente el registro 02_W previo a la interrupción.

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.

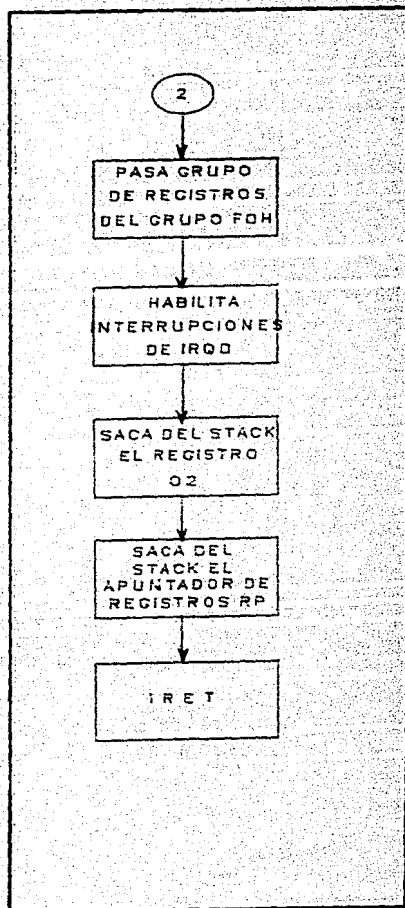
Finalmente si la opción de memoria de datos se encuentra habilitada en registros alternos, entonces habilita la señal DMHAB del puerto interno de control y salta a la rutina ENTRADA.

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_W, los cuales se encontraban dentro del stack. Por último habilita interrupciones por IRD0 y ejecuta una instrucción IRET (recordar que la dirección de arranque fué introducida al stack pasos atrás), con lo cual automáticamente habilita interrupciones.





60-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 IRDO. La función de estas es respaldar en memoria de datos, información pertinente para la ejecución de K:

DIRECCION

MEMORIA DE DATOS

3050

3051

3052

3053

3054

3055

(D) O (EBP) ASCII

**DIRECCION DEL ULTIMO
COMANDO
EJECUTADO**

RECONOCIMIENTO (D)

**ULTIMA DIRECCION DE
DESPLIEGE O DE
RETORNO**

Inspeccionando en esta tabla las localidades 3050., y 3051., determina cual fue 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., y 3055., y pasa a ejecutar estos comandos desde las rutinas TAMGOK y TAMCAK respectivamente.

ROUTINA DE SERVICIO A IRQO.

Este programa esta formado por muchas pequeñas rutinas, comenzando por SALPRO, cuya dirección está contenida en el vector de interrupción correspondiente.

SALPRO.

Esta rutina debe modificar los registros RP, 02, POIM, 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 este se encuentra respaldado en el registro IBW. La modificación al registro POIM 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 según cualquiera de los cuatro casos siguientes:

ROUTINA CTOATO	CALL CON TRACE
ROUTINA CBPOATO	CALL CON BP
ROUTINA GOTOATO	GO CON TRACE
ROUTINA GOBPOATO	GO CON BP

XXXOATO.

Estas rutinas están formadas por combinaciones de otras cuatro:

ATRACE Atención a interrupciones por TRACE.

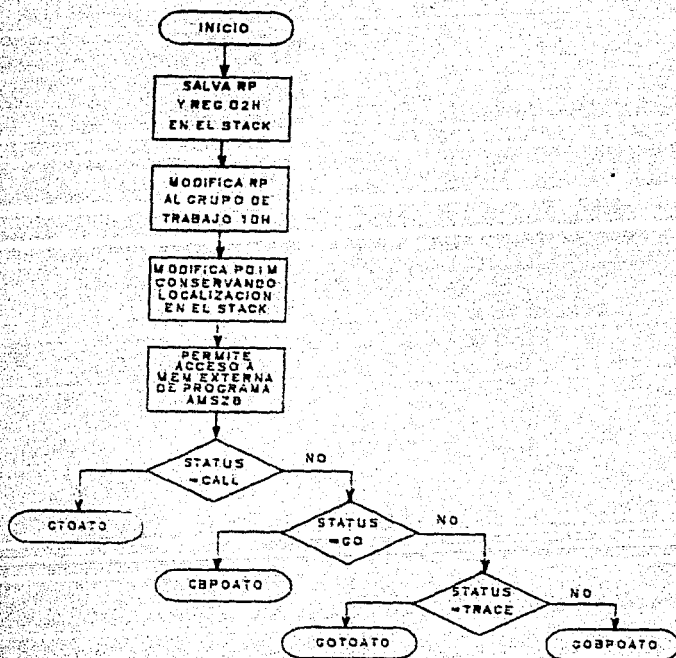
ATBP Atención a interrupciones por BP.

RESPALDAU Rutina que debe ejecutarse en caso de GO.

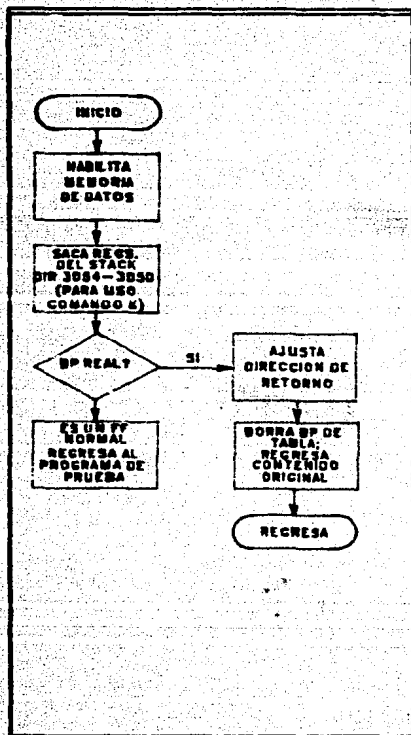
DTRCALL Rutina que debe ejecutarse en caso de CALL.

ATRACE.

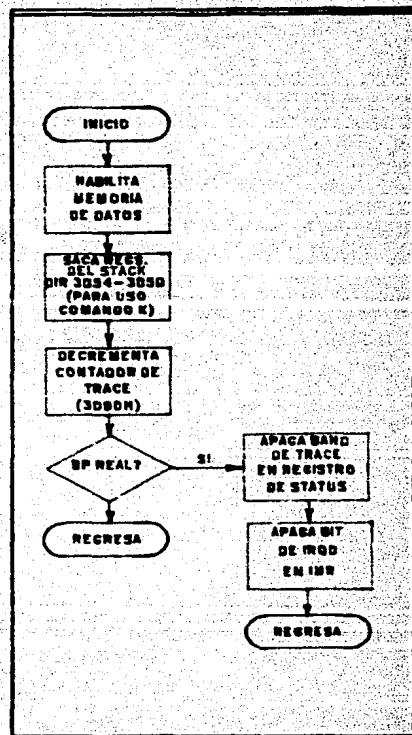
Su función es decrementar el contador de eventos TRACE que se encuentra en la localidad 3090H de memoria de datos. Si éste es cero apaga la bandera de T/BP del registro de status y elimina posteriores interrupciones apagando el bit 1 del registro IMR.



SALPRO



AT BP

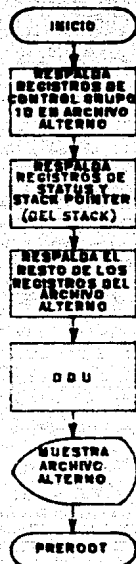


ATRACE



DDU

DTRCALL

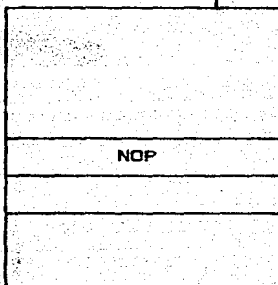


RESPALDAU

ATSP.

Cuando ocurre una interrupción debida a BP, la dirección de retorno está alterada, esto 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:

Dirección de BP
Dirección de BP+1



ESTA ES LA DIRECCION
QUE HA SIDO SUSTITUIDA
POR UN BYTE DE BP.

ESTA ES LA DIRECCION
QUE HA SIDO SALVADA EN
EL STACK, DESPUES DE LA
INTERRUPCION.

Es por esto 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 fue 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%, 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.

SCAN 28

DECLARACIONES

0000	FLR03	EDU	252
0004	EDF	EDU	04
0030	TORAL	EDU	30M
0040	MACHE	EDU	40M
0012	AP	EDU	12M
0013	HUFF	EDU	13M
0013	CONTR	EDU	13M
0004	LF	EDU	04M
002C	COMA	EDU	2CM
0000	NUL	EDU	00
0000	DS	EDU	00
007F	DEL	EDU	7FH
00FE	FI	EDU	0FEH
00FD	FT	EDU	0FDH
00FF	FF	EDU	0FFH
0004	CTILF	EDU	04M
0020	ESP	EDU	20H
0000	CR	EDU	13
0010	REG	EDU	10M
001E	REG1	EDU	1EH
001F	REG2	EDU	1FH

FORM

276

000-20 SISTEMA DE DESARROLLO

```

0000                                FORM
0000
0000 ;
0000 ;"          Rutina de entrada al programa del usuario          "
0000 ;"          SE EJECUTA PARA GO Y CONTINUE          "
0000 ;
0000 ;
0000 00F1  ENTRADA  LD      R12,00F1H ;
0000 00F2  LD      R13,011H ;" PASA REGISTROS DEL GRUPO 10 A "
0000 00F3  LD      R14,000H ;" SUS SIMILARES EN EL GRUPO DE "
0000 00F4  NUEVE   LD      R15,0010 ;" REGISTROS DE CONTROL "
0000 00F5  F2CF   LD      R012,R15 ;
0000 00F6  CE     INC     R12
0000 00F7  DE     INC     R13
0000 00F8  E0F0  DJNZ    R14,NUEVE
0000 00F9  46F001 OR      IRR,0000000010 ; HABILITA TROO
0000 00E0  5002  UNWINDS POP   02 ; SACO DEL STACK EL REGISTRO 02
0000 00F0  50F0  POP    RP ; Y EL REGISTRO RP
0000 00F2  BF     IRET
0000 ;
0000 ;"          Rutina de salida de cualquier programa de usuario          "
0000 ;"          SE ACCESA A ELLA MEDIANTE UNA INTERRUPCION          "
0000 ;
0000 ;
0000 00F3  70F0  SALPRO  PUSH   RP ; INTRODUCE AL STACK A RP Y 02
0000 00F5  7002  PUSH   02
0000 00F7  3110  SRP     10H ; MODIFICA RP AL GRUPO DE TRABAJO
0000 00F9  0CF0  LD      R0,011110110
0000 00FB  40F003 OR      R0,PO1H ; MODIFICA EL REGISTRO PO1H SIN
0000 00FE  50E004 AND     R0,0110101100 ; ALTERN LA LOCALIZACION DE STACK
0000 0001  0F00  LD      PO1H,R0
0000 0003  60F700 LD      P3H,0010100000 ; MODIFICA P3H
0000 0006  C0FE   LD      R12,SPH
0000 0008  00FF   LD      R13,SPL
0000 000C  700C  PUSH   R12
0000 000E  700D  PUSH   R13
0000 0010  50027F AND     02,0011111110 ; HABILITA AWS28
0000 0014  70FC03 TH      FLAGS,00000000110 ; DETERMINA LAS COMBINACIONES
0000 0016  003010 JP      E0,COT010
0000 0018  70FC02 TH      FLAGS,00000000100
0000 001A  003003 JP      E0,C0F010 ; T, GO , CALL V
0000 001C  70FC01 TH      FLAGS,0000000010 ; BP CON QUE FUE EJECUTADO EL
0000 001E  003011 JP      E0,COT010 ; PROGRAMA DE PRUEBA
0000 0020  003017 JP      C0F010

```

SIS-20 SISTEMA DE DESARROLLO

0006

FORM

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

0006

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

SIS-20 SISTEMA DE DESARROLLO

3004		FORM	CP	R2,061H	
3004	AGE261	BUFF	CP	LT,SAL	
3007	1011	JR	CP	R2,070H	; CAMBIO DE MINUSCULAS A MAYUSCULAS
3009	AGE270	CP	JR	0E,SAL	
300C	900C	JR	LD	R3,12H	
300E	3012	LD	SUB	R3,020H	
3000	26E320	SUB	LD	010H,13H	; GUARDA EL CARACTER EN BUFFER E
3003	F51310	LD	INC	R0	; INCREMENTA APONTADOR
3006	0E	INC	JP	,RETA	
3007	003055	JP	LD	010H,12H	
300A	F51210	SAL	INC	R0	
3000	0E	INC	JP	,RETA	
300E	003055	JP	CP	12H,0DEL	; SI EL CARACTER RECIBIDO ES DEL
30C1	A6127F	PDEL	JR	ME,BUFF	; MANDA CARACTER BS(BACK SPACE)
30C4	E0DE	JR	LD	R6,0DELETE>0	; ESP Y BS
30C5	6C30	LD	LD	R7,0DELETE\256	
30C0	7CF0	LD	DEC	R0	
30CA	0100	DEC	CP	10H,050H	; ELIMINA CARACTER DEL BUFFER
30CC	A61050	CP	JP	ME,EMUSINT	; SI SE HA BORRADO TODA LA LINEA
30CF	E03207	JP	LD	R4,0TRES>0	; REGRESA AL ESTADO 3
3002	4C30	LD	LD	R5,0TRES\256	
3004	5C00	LD	JP	,EMUSINT	
3006	003207	JP	CP	12H,0CR	; BUSCA EL CARACTER DE CR
3009	A61200	CUATRO	JR	ME,PDEL	
300C	E0E3	JR	LD	R4,0TRES>0	; SI LO ENCONTRO
300E	4C30	LD	LD	R5,0TRES\256	; COMIENZA EL DECODIFICADOR
3000	5C00	LD	LD	R0,0TABL>0	; TABL CONTINE LA TABLA DE COMANDOS
3002	0C3C	LD	LD	R9,0TABL\256	
3004	9CE1	LD	LD	010H,0CDMA	
3006	E7102C	LD	LDC	R1,0RR0	; ESTA EL PRIMER CARACTER DEL BUFFER?
3009	C210	LDC	LD	R0,050H	
300B	0C50	COMP1	CP	R1,0R0	; SI NO LO ESTA BUSACA EL SIGUIENTE
300D	A310	COMP2	JR	EQ,CONT	; COMANDO EN TABL
300F	601F	JR	INC	R0	
3001	00E0	BUSCAF1	LDC	R1,0RR0	
3003	C210	LDC	CP	11H,0F1	
3005	A611FE	CP	JR	ME,BUSCAF1	
3008	E0F7	JR	ADD	19H,003H	
300A	061903	ADD	ADC	10H,000H	
300D	161000	ADC	LDC	R1,0RR0	
3100	C210	LDC	CP	11H,0FT	
3102	A611FD	CP	JR	ME,COMP1	; SI LLEGO AL FIN DE TABL
3105	E0E4	JR	LD	R6,0MEMSINH>0	
3107	6C3F	SINTAX	LD	R7,0MEMSINH\256	; ENVIA MENSAJE DE ERROR DE SINTAKIS
3109	7030	LD	LD	R0,050H	
3100	0C50	LD	JP	,EMUSINT	
3100	003207	JP			

PROGRAMA SISTEMA DE DESARROLLO

000	0000	CONT	FORM	000	
001	00		INCU	00	
002	0000		INC	R1,0000	; SI EL RESTO DE LOS CARACTERES DEL
003	0000		LDC	11H,0F1	; COMANDO COINCIDEN CON LOS DEL
004	0000		CP	ME,COMP	; BUFFER
005	0000		JR	R1,03FH	
006	0000		LD	001,0000	; TOMA LA DIRECCION DE LA TABLA DE
007	0000		INCU	001,0000	; COMANDOS
008	0000		LDC	R0,0TABL20	; Y LO INTRODUCE AL STACK
009	0000		LDC	R0,0TABL20	
010	0000		LD	R0,0TABL20	; AL EJECUTAR ESTA INSTRUCCION YA
011	0000		LD	R0,0TABL20	; NO REGRESA A ROOT SINO AL COMANDO
012	00		RET		

819-29 SISTEMA DE DESARROLLO

3127

FORM

3127

; INICIALIZACION DE VARIABLES Y DECLARACION DE CONSTANTES

3127

; "

AQUANTA

"

3127

; "

; INICIALIZACION DE VARIABLES Y DECLARACION DE CONSTANTES

3127 BF

AQUANTA

DI

LD 000001000B

; ELIMINA INTERRUPCIONES

3128 E6F000

TDVND

TN

LD 000001000B

; MUESTRO DE IRQ3

3128 76F000

JR

2,TDVND

312E 60F0

LD

LD 000001000B

3130 E6F000

LD

RZ, SIO

; REGRESA CARACTER DEL SIO

3133 20F0

RET

3135 BF

2000

203

SIS-20 SISTEMA DE DESARROLLO

		FORM	
3175			
3175			
3175			
3175 0C00	ESDAT	LD	R0,000H
3177 1C00		LD	R1,030H
3179 9C00		LD	R9,030H
317B 3C00		LD	R3,030H
317D 06325A	MDAT	CALL	ESPERA ; ESPERA UN CARACTER DE CONSOLA
3180 A61230		CP	12H,030H
3183 1B29		JR	17,CROEL ; DETERMINA SI EL CARACTER ES
3185 A6127F		CP	12H,00EL ; NO IMPRIMIBLE
3188 EB16		JR	NE,ACONODA
318A A6E900		CP	R0,000H
318D 60EE		JR	EQ,MDAT
318F 3019		LD	R3,19H ; SI ES DEL ACONODA EL BUFFER
3191 9711		LD	R9,11H
3193 1C00		LD	R1,030H
3195 00E9		DEC	R0
3197 6C00		LD	R6,00ELETE>0
3199 7CF0		LD	R7,00ELETE\256
319B D63207		CALL	EMUSINT
319E 8000		JR	MDAT
31A0 29F0	ACONODA	LD	S10,R2 ; HACE UN ECO DEL CARACTER Y
31A2 D6325A		CALL	ESPERA ;
31A5 1019		LD	R1,19H ; LO INTRODUCIENDO AL BUFFER
31A7 9013		LD	R9,13H
31A9 3012		LD	R3,12H
31AB 0E		JNC	R0
31AC 00CF		JR	MDAT
31AE A61200	CROEL	CP	12H,00C
31B1 EBCA		JR	NE,MDAT
31B3 A6E900		CP	R0,000H
31B6 60E2		JR	EQ,0YE
31B8 2019		LD	R2,19H
31BA 0F	0YE	RET	

000-20 SISTEMA DE DESARROLLO

FORM

A S C - N E X

CONVERSION ASCII- NEX DE DOS DIGITOS
 NECESITA CUATRO PARAMETROS LOS REGISTROS
 R2 Y R3 DEL GRUPO1. EL RESULTADO LO DEJA
 EN EL REGISTRO R2.

0000	00E247	ASCNEX	CP	R2,047H	
0000	0001F2		JP	OE,HEXERR	
0000	00E25F		CP	R2,02FH	
0000	2031F2		JP	LE,HEXERR	;VERIFICA DIGITOS
0000	00E247		CP	R3,047H	
0000	0001F2		JP	OE,HEXERR	
0000	00E25F		CP	R3,02FH	
0000	2031F2		JP	LE,HEXERR	;DIGITO MENOS SIGNIFICATIVO ; CONTIENZA CONVERSION
0000	00E240		CP	R2,040H	
0000	0005		JR	GT,UN01	
0000	20E230		SUB	R2,030H	
0000	0005		JR	CONTIN1	
0000	20E237	UN01	SUB	R2,037H	
0000	000340	CONTIN1	CP	R3,040H	
0000	0005		JR	GT,UN02	
0000	20E330		SUB	R3,030H	
0000	0005		JR	CONTIN2	
0000	20E337	UN02	SUB	R3,037H	
0000	00E2	CONTIN2	SNAP	R2	
0000	4223		OR	R2,R3	
0000	00		RET		
0000					; MENSAJE DE ERROR
0000					
0000	0000	HEXERR	LD	R6,0000 > 0	
0000	0000		LD	R7,0000 \ 256	
0000	000007		CALL	ERRMSG	
0000	00000E		JP	PERROUT	
0000					

810-20 SISTEMA DE DESARROLLO

FORM

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

31FC

70EA

E61A9E

F3E21A

S71A0F

201A

F3E21A

S71A0F

F03F

E61A9E

A71A0F

AD05

071A30

0003

071A37

201A

A61A04

EDEC

50EA

AF

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

3220

NEXASC

1WTC10

C1GUE

LETRAS

CONT1

CP

JR

POP

RET

LD

LD

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

PUSH

LD

LD

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

AND

R10

REG,05EH

REG,02

REG,00001111B

REG

REG,02

REG,011110000B

SFN

REG,05EH

REG,009H

GT,LETRAS

REG,030H

CONT1

REG,037H

REG

REG,060H

ME,C1GUE

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

R10

101-20 SISTEMA DE VERIFICACIÓN

500

E N V A S I N T

- SE ENCARGA DE ENVIAR CADENAS DE MENSAJES
- SIN USO DE INTERRUPCIONES. LA RUTINA REGRESA
- CUANDO EL CARACTER A ENVIAR ES UN FF HEXA-
- DECIMAL

ENVSINT	CALL	ZIGUE	
ENVSINT	CALL	ESPERA	; ENVIO SIN INTERRUPCION
	LDC	R2,RR6	; APUNTAOR AL LETRERO DE STATUS
	CP	12H,00FFH	; SI ES FINAL DE LA CADENA REGRESA
	JP	E0,PREREG	; DESPLIEGA NUMERO DE SECCION
	CALL	ZIGUE	
	JP	ENVSINT	
PREREG	RET		
ZIGUE	LDC	R2,RR6	; ENVIO DE CARACTERES
	LD	S10,R2	
	INCW	RR6	
	RET		
; *****			
PARM6	LD	R12,030H	
	LD	R13,054H	
	LDE	RR12,R4	
	INCW	RR12	
	LDE	RR12,R5	
	RET		

```
LD      R12,R30H
LD      R13,R54H
LDR     R12,R4
INCW    R12
LDR     R12,R5
```

ROUTINA DE REGRESO AL MONITOR

```

PREROOT      LD      IR0,000H
              LD      R6,0PROMPT10
              LD      R7,0PROMPT1256      ; ENVIA EL PROMPT A CONSOLA
              CALL    ENV$INT
              CALL    CAN0                ; REGRESA AL ESTADO 3
              CALL    LIMP10B             ; PONE EN CEROS LAS LOCALIDADES
              LD      R0,#50H             ; 3000 - 300F
              LD      SPL,0400H
              LD      SPH,0400H
              JP      ROOT

```

ESPACIO	LD	\$10, RESP
	CALL	ESPERA
	RET	

SIS-28 SISTEMA DE DESARROLLO

[illegible]

800-00 SISTEMA DE DESARROLLO

0000		PCON		
0000	DE	INC	R0	; INCREMENTA APUNTAOR AL BUFFER
0001	DE	INC	R10	; INCREMENTA EL CONTADOR R10
0002	000003	CP	R10,003H	; COMPARA EL CONTADOR CON 3H
0003	0005	JR	LT,CC00H	; SI ES MENOR SE VA A CC00H
0004	000E	EDR1	LD	R6,0LET>0
0005	7C04	LD	R7,0LET>236	; MANDA EL MENSAJE DE "CARACTER INVALIDO"
0006	003007	CALL	EMPSINT	; LO ENVIA A LA CONSOLA
0007	00000E	JP	PRE00T	; REGRESA A PRE00T
0008				; FIN DE LA RUTINA QUE RECONOCE UNA DIRECCION;
0009				; *****
000E	0C30	SALVAC	LD	R4,030H
000F	3C30	LD	R5,030H	
0012	003040	CALL	DATA	
0015	2013	LD	R2,13H	
0017	00C4	INCU	XP4	
0019	003040	CALL	DATA	
001E	0F	RET		
001F				; *****
0020	0C30	CARG	LD	R4,0TRES>0
0027	3C30	LD	R5,0TRES>256	
0031	0F	RET		
0032	02C4	DATA	LDE	R2,0RR4
0044	0F	RET		
0046	C224	PROG	LDC	R2,0RR4
0047	0F	RET		
0048	9224	DATA	LDE	0RR4,R2
0049	0F	RET		
004B	0224	PRNG	LDC	0RR4,R2
004D	0F	RET		

819-20 SISTEMA DE DESARROLLO

334E		FORM	
334E			
334E	0C30	POD	LD R4, 030H
3350	0C32		LD R5, 0520
3352	A71044		CP 010H, 044H
3355	680F		JR EB, DAT
3357	6C33		LD R6, 0PR06>8
3359	7C45		LD R7, 0PR06\256
335B	AC33		LD R10, 0PR06>8
335D	0C48		LD R11, 0PR06\256
335F	2C20		LD R2, 0ESP
3361	D63348		CALL DRATA
3364	880E		JR FINPOD
3366	6C33	DAT	LD R6, 0DATA>8
3368	7C42		LD R7, 0DATA\256
336A	AC33		LD R10, 0DATA>8
336C	0C48		LD R11, 0DATA\256
336E	2C44		LD R2, 044H
3370	D63348		CALL DRATA
3373	0E		INC R4
3374	AF	FINPOD	RET
3375			
3375	0C50	LINPIAG	LD R0, 050H
3377	1C20		LD R1, 020H
3379	E7E020	GHUY	LD R0, 0ESP
337C	0E		INC R4
337D	1AF4		DJNZ R1, GHUY
337F	AF		RET
3380			
3380			

800-20 SISTEMA DE DESARROLLO

FORM

C O M A N D O S

T R N

8000	2032	TEN	LD	R2,002	
8002	063242		CALL	MODOS	; CAMBIA A MODO 2 DEL PUERTO SERIE
8005	0000		LD	R0,000H	
8007	063127	NON	CALL	ADAMITA	
800A	061206		CP	12H,0CTLF	; ESPERA HASTA RECIBIR CTRLF
8000	0004		JR	E0,RECIBE	; PARA EMPEZAR A
800F	20F0		LD	S10,R2	; ALMACENAR DATO
8071	00F4		JR	NON	
8073	0000	RECIBE	LD	R9,000H	
8075	063307		CALL	D0CP	
8070	C012		LD	R12,12H	
8070	063307		CALL	D0CP	
8070	06E210		CP	R2,010H	
8080	000F		JR	GE,PREPRO	
8082	06E230		ADD	R2,030H	
8080	F012		LD	R15,12H	
8087	071044		LD	010H,0440H	
8080	06334E		CALL	P00	
8080	001F		LD	R4,1FH	
808F	0000		JR	P0AJ0	
8091	071040	PREPRO	LD	010H,000H	
8094	F012		LD	R15,12H	
8096	06334E		CALL	P00	
8097	001F		LD	R4,1FH	; ALMACENA DIRECCION
8090	063307	P0AJ0	CALL	D0CP	; EN R4 Y R5
809E	0012		LD	R5,12H	
80C0	063307		CALL	D0CP	; COMIENZA LECTURA Y ALMACENAMIENTO
80C3	063307	T0F0	CALL	D0CP	
80C6	0000		CALL	00R10	; DE BYTES DEL REGISTRO
80C0	0004		INC0	NON	
80C0	00F7		0AJ02	R12,T0F0	
80CC	063307		CALL	D0CP	
80CF	06E700		CP	R9,000H	; R9 ES EL REGISTRO DE CHECKSUM
80D2	000F		JR	E0,RECIBE	
80D4	0E		INC	R0	; R0 RECIBE CONTADOR DE ERRORES
80D5	000C		JR	RECIBE	

915-Z8 SISTEMA DE DESARROLLO

```

3307                                FORM
3307                                ;
3307                                ;
3307                                ; D D C P
3307                                ;
3307                                ;
3307                                ;
3307                                ;
3307                                ; EN ESTA RUTINA SE RECIBEN CARACTERES Y SE INTERPRETAN
3307                                ; DIRECTAMENTE DE LA COMPUTADORA AMFITRIONA
3307                                ; EL RESULTADO DE ESTA RUTINA QUEDA EN R2
3307                                ;
3307                                DOCP
3307                                CALL AGUANTA
3307                                CP 12H,030H
3307                                JP LT,PRCF
3307                                CP 12H,03AH
3307                                JR EQ,DOCP
3307                                LD R1,12H
3307                                CALL AGUANTA
3307                                LD R3,12H
3307                                LD R2,11H
3307                                CALL ASCHEX
3307                                ADD R9,R2 ; AQUI SE HACE LA SUMA PARA CHECKSUM
3307                                RET
3307                                PRCF
3307                                CP 12H,0CTRLF ; AQUI SE PRUEBA SI SE RECIBE EL SEGUNDO
3307                                JR NE,DOCP ; SEGUNDO CTRL F
3307                                LD R15,15H
3307                                LD R14,14H
3307                                LD R2,001H ; DE SER CTRL F REGRESA AL MODO 1 PUERTO
3307                                CALL MODOS ; SERIE
3307                                CALL CRLF
3307                                LD R6,0SIGN08
3307                                LD R7,0SIGN256
3307                                CALL EMVSINT
3307                                INCM RR14
3307                                LD R2,1EH
3307                                CALL DESP
3307                                LD R2,1FH
3307                                CALL DESP
3307                                CALL CRLF
3307                                LD R6,0TRERR08 ; MENSAJE DEL NUMERO DE ERRORES
3307                                LD R7,0TRERR256
3307                                CALL EMVSINT
3307                                CALL ESPACIO
3307                                LD R2,1BH
3307                                CALL DESP
3307                                LD R2,001H ; RETORNO A ROOT
3307                                CALL MODOS
3307                                JP PREROOT

```

010-20 SISTEMA DE DESARROLLO

		FORM		
0402				
0402				
0402				
0402				
0402	SC03	LD	R2, 003H	
0404	063242	CALL	MODOS	; CAMBIA AL MODO 3 DEL PUERTO SERIE
0407	063127	CALL	AGUMTA	
0408	061204	CP	12H, 0CTRLF	; Y ESPERA A RECIBIR CTRLF
0409	0200	JR	ME, ESCL	; PARA REGRESAR AL MODO1
0407	201	LD	R2, 001H	
0404	063242	CALL	MODOS	
0404	00320E	JP	PWR007	
0407	27F0	ESCL	LD	S10, R2
0409	00EC	JR	STAND	

SIS-28 SISTEMA DE DESARROLLO

3400

FORM

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

3400

200-20 SISTEMA DE DESARROLLO

2003		FORN	
2004	240200	REST	R2,00E0H
2005	00F2	ADD	R15,R2
2006	00E0	LDE	R2,00R14
2007	003000	CALL	DESP
2008	00F040	LD	SIO,00ACHE
2009	003020	CALL	ESPERA
2010	0030C0	CALL	ESPACIO
2011	000007	JP	FINUM
2012	0005	LD	R2,005
2013	003000	CALL	DESP
2014	00F040	LD	SIO,00ACHE
2015	003030	CALL	ESPERA
2016	0030C0	CALL	ESPACIO
2017	00	FINUM	PET
2018			
2019			
2020			
2021			
2022			
2023			
2024	0000	GRUPO	LD R0,000H
2025	003000	GRUPO	CALL UNREG
2026	00		R0
2027	00		R5
2028	0015		R2,15H
2029	00100F		CP 18H,00FH
2030	0002		JR GT,FINGRUP0
2031	00F2		JR GRUPO
2032	00	FINGRUP0	RET

ROUTINA QUE DESPLIEGA UN GRUPO DE REGISTROS R0 ES UN CONTADOR DE " 16 REGISTROS "

SIS-20 SISTEMA DE DESARROLLO

3569

FORM

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

3569

SIS-28 SISTEMA DE DESARROLLO

3908		FORM
3909	EXC	
3909	063247	CALL
3910	3912	LD
3911	06F932	LD
3912	063250	CALL
3913	063290	CALL
3914	06F940	LD
3915	063256	CALL
3916	06F930	LD
3917	063256	CALL
3918	063257	CALL
3919	0F	RET

CRLF
 RS, 12N
 SIO, 052N
 ESPERA
 DESP
 SIO, 0WACNE
 ESPERA
 SIO, 0IGUAL
 ESPERA
 ESPACIO

; DUTING PARA IMPRIMIR EL NÚMERO DE
; REGISTRO A DESPLAZARSE EN LA FORMA
; SIGUIENTE

STG-20 SISTEMA DE DESARROLLO

3633			FROM	
3635	IME4	SIG1	CALL	ORRG
3637	A61220		CP	12W,020W
3639	0002		JR	GT,EYO
363C	2C2E		LD	R2,02EN
363E	2W0	EYO	LD	S10,R2
3640	063220		CALL	ESPER0
3643	00E4		INCU	RR4
3645	063203		CALL	PH0RK
3648	00E3		DJM2	R0,COMP
3649	A20E	SRLT0	CP	R4,R14
364C	ED04		JR	ME,S102
364E	020F		CP	R5,R15
3670	000F		JR	ED,FIN
3672	063247	S102	CALL	CHLF
3675	CC30		LD	R12,030H
3677	0C10		LD	R13,010H
3679	024C		LDE	0RR12,R4
3670	00EC		INCU	RR12
3670	025C		LDE	0RR12,R5
367F	000F		JR	DES01R
3681	00320E	FTN	JP	PREF0RT
3684		0000		
3684	CC30		LD	R12,030H
3686	0C10		LD	R13,010H
3689	024C		LDE	R4,0RR12
368A	00EC		INCU	RR12
368C	025C		LDE	R5,0RR12
368E	00EC		INCU	RR12
3690	02EC		LDE	R14,0RR12
3692	00EC		INCU	RR12
3694	02FC		LDE	R15,0RR12
3696	0F		RET	
3697	0224	000	LDE	R2,0RR4
3699	00E4		INCU	RR4
3690	0234		LDE	R3,0RR4
3690	00E3		SWAP	R3
369F	4223		OR	R2,R3
36A1	0F		RET	

818-20 SISTEMA DE DESARROLLO

3702		FORM			
3702	00E4	DECU	RR4	:	\
3704	0204	LDE	R0,RR14	:	:
3706	00E4	DECU	RR4	:	:
3708	0214	LDE	R1,RRR4	:	:
370A	50E9	POP	R9	:	:
370C	50E9	POP	R9	:	:
370E	50E9	POP	R9	:	:
3710	50E9	POP	R9	:	:
3712	70E1	PUSH	R1	:	:
3714	70E0	PUSH	R10	:	:
3716	70EC	PUSH	R12	:	:
3718	70E0	PUSH	R0	:	/
371A	0009	LD	R0,09H	:	\
371C	EC45	LD	R14,045H	:	:
371E	FC10	LD	R15,010H	:	:
3720	920E	LDE	RRR14,R0	:	:
3722	DC7B	LD	R13,06FH+12	:	:
3724	EC45	LD	R14,045H	:	:
3726	FC00	LD	R15,000H	:	:
3728	CC00	LD	R12,000H	:	:
372A	03CE	LDEI	RR12,RRR14	:	:
372C	A6EC1C	CP	R12,01CH	:	:
372F	ED3736	JP	NE,ESTOSI	:	:
3732	CC20	LD	R12,020H	:	:
3734	FC20	LD	R15,020H	:	:
3736	A6EC02	CP	R12,002H	:	:
3739	ED06	JR	NE,NOES2	:	:
373B	020E	LDE	R0,RRR14	:	:
373D	70E0	PUSH	R0	:	:
373F	A0EE	INCU	RR14	:	:
3741	04E7	DJNZ	R13,0027	:	/
3743	F017	LD	R15,17H	:	\
3745	50EF10	AND	R15,000011000B	:	:
3748	600A	JR	Z,DMHAB	:	:
374A	76EF00	TH	R15,0000010000B	:	:
374D	600E	JR	Z,PRENT	:	:
374F	76FF10	TH	R15,0000100000B	:	:
3752	6009	JR	Z,PRENT	:	:
3754	C010	LD	R12,10H	:	:
3756	46EC20	OR	R12,0001000000B	:	:
3759	ECF0	LD	R14,00F0H	:	:
375B	02CE	LOC	RRR14,R12	:	:
375D	0F	DI		:	/
375E	000030	JP	ENTRADA	:	:

NETE AL STACK RP Y STATUS DEL
ARCHIVO ALTERNO ASI COMO
DIRECCION DE ARRANQUE

TRANSFIERE ARCHIVO ALTERNO AL
ARCHIVO INTERNO EXCEPTO
REGISTROS DE CONTROL. ESTOS
QUEDAN EN EL GRUPO 10.

DETERMINA SI EN LOS REGISTROS DEL
ARCHIVO ALTERNO ESTA HABILITADA
DN PARA HABILITAR LA SENAL DMHAB
ANTES DE SALTAR AL MONITOR
INTERNO.

800-20 SISTEMA DE DESARROLLO

```

8061 FORM
8062
8063
8064 ESTO ES UTIL PARA POSTERIORES USOS DE CALL
8065
8066
8067 CALL LD R2,0CALL:0
8068 0061 LD R3,0CALL:256
8069 0060C CALL COMAN
8070 00FC82 TANCAK OR FLAG,000000100 ; PRENDE BANDERA CALL / GO
8071 00FC01 TH FLAG,000000010
8072 000779 JP ED,FUETRACE
8073 0081 LD R2,001H
8074 003242 CALL NODOS
8075 00077E JP INTERNA
8076 0041 FUETRACE LD R2,041H
8077 003242 CALL NODOS
8078 00F INTERNA DT
8079 00F001 LD INH,0000000010
8080 0036 LD R0,0110101100
8081 7C30 LD R7,0010100000
8082 F EI
8083 00EC CALL 00R12 ; LLAMADA A LA DIRECCION DE ARRANQUE DEL
8084 00020E JP PIEROOT ; PROGRAMA DE PRUEBA
8085
8086
8087 ESTO RUTINA ES COMUN A CALL Y GO
8088
8089
8090
8091 COMAN CALL SALVAC
8092 0030 LD R12,030H
8093 0010 LD R13,010H
8094 00300 CALL RECDIR
8095 0030 LD R2,030H
8096 0010 LD R3,010H
8097 00C2 LDE R12,00R2
8098 00C2 INCH R12
8099 0002 LDE R13,00R2
8100 00F RET

```


820-20 SISTEMA DE DESARROLLO

SPCC	FORM			
SPCC				
SPCC		P	N	
SPCC				RUTINA QUE PRUEBA LA MEMORIA DE PROGRAMA Y LA MEMORIA DE DATOS
SPCC				
SPCC	EC3E	PH	LD	R14,0TAPW)8
SPCC	FC14		LD	R15,0TAPW)256
SPCC	GC3F		LD	R5,0UDE)8 ; MENSAJE DE "PRUEBA
SPCC	7C0B		LD	R7,0UDE)256 ; DE VERIFICACION DESPUES DE
SPCC	063207		CALL	EMUSINT ; ESCRITURA "
SPCC	063030	PRINC	CALL	C4CB
SPCC	0C31	CONW	LD	R8,004H ; COMIENZA CICLO PRINCIPAL
SPCC	3C04	C104CB	LD	R3,000H ; CONTADOR DE CARACTERES R3
SPCC	0C04	CONRET	LD	R12,00AH ; CONTADOR DE INTENTOS
SPCC	2013	SIEMPRE	LD	R2,12H
SPCC	04CB		CALL	04CB ; GUARDA CONTENIDO
SPCC	04CB		CALL	04CB ; LEE CONTENIDO
SPCC	0223		CP	R2,R3
SPCC	0805		JR	E0,PASA ; VERIFICA IGUALDAD
SPCC	0AF4		DJNZ	R12,SIEMPRE ; SI HAY ERROR NUEVO INTENTO
SPCC	063049		CALL	FALLA ; SI DESPUES DE 10 INTENTOS
SPCC	A61C04	PASA	CP	1CB,00AH ; MARCA ERROR
SPCC	0803		JR	E0,NORETRIES ; SI HUBO REINTENTO CONTINUA
SPCC	063063		CALL	RETRIES
SPCC	0804	NORETRIES	INCW	R4
SPCC	30E3		DJNZ	R3,CONRET
SPCC	080F		DJNZ	R0,C104CB
SPCC	100B		DJNZ	R1,CONRET
SPCC	A6EF34		CP	R15,0TAPW)256*32 ; SI NO HAY MAS SECTORES CONTINUA
SPCC	E003		JR	NE,PRINC
SPCC	0C3F		LD	R6,0L0W)8
SPCC	7C07		LD	R7,0L0W)256 ; MENSAJE DE PRUEBA DE LECTURA
SPCC	063207		CALL	EMUSINT
SPCC	EC3E		LD	R14,0TAPW)8
SPCC	FC14		LD	R15,0TAPW)256

SIS-20 SISTEMA DE DESARROLLO

300F		FORM			
300F 063030	COMLEC	CALL	CARD		; CICLO PRINCIPAL DE PRUEBA DE
3012 0004	LUNK	LD	R0,000H		; LECTURA
3014 3000	LC104	LD	R3,000H		; CONTADOR DE CARACTERES R3
3016 0000	LECR	LD	R12,000H		
3018 0000	LSIEN	CALL	DIR10		; LEE DIRECCION
301A 0223		CP	R2,R3		; COMPARA CON CONTADOR DE CARACTERES
301C 0005		JR	E0,01EH		
301E 04F0		DJNZ	R12,LSIEN		; SI NO ES IGUAL NUEVO INTENTO
3020 063009		CALL	FALLA		; SI DESPUES DE 10 INTENTOS ERROR
3023 06100A	DIEN	CP	1CH,000H		
3026 0003		JR	E0,MDINT		
3028 063065		CALL	RETRIES		; SI HUBO INTENTOS DESPLIEGA ERROR
302B 00E4	MDINT	INCH	R04		; DE REINTENTOS
302D 30E7		DJNZ	R2,LECR		
302F 04E3		DJNZ	R0,LC104		
3031 100F		DJNZ	R1,LUNK		
3033 04EF34		CP	R15,07APW*256+32		; SI EXISTEN NAS SECTORES REGRESA
3036 0307		JR	NE,COMLEC		; AL CICLO PRINCIPAL
3038 00320E		JP	PREROUT		
3039					
3039 0C14	CARD	LD	R12,014H		
303D 0038		LD	R13,003H		
303F C3CE	CAREG	LDCI	0R12,00R14		; OBTIENE INFORMACION REFERENTE
3041 04FC		DJNZ	R13,CAREG		; A CADA SECTOR DE LA TABLA DE
3043 1C08		LD	R1,000H		; PRUEBA DE MEMORIA
3045 063207		CALL	ENVSINT		
3048 AF		RET			
3049 C012	FALLA	LD	R12,12H		
304B 0013		LD	R13,13H		
304D 063075		CALL	DIRECCION		; DESPLIEGUE DE ERRORES
3050 2010		LD	R2,10H		
3052 063200		CALL	DESP		
3055 0632C7		CALL	ESPACIO		
3058 201C		LD	R2,1CH		
305A 063200		CALL	DESP		
305D 063207		CALL	CALF		
3060 C00A		LD	R12,000H		
3062 3010		LD	R3,10H		
3064 AF		RET			
3065 0C0A	RETRIES	LD	R13,000H		
3067 220C		SUB	R13,R12		
3069 063075		CALL	DIRECCION		
306C 2010		LD	R2,10H		
306E 063200		CALL	DESP		
3071 063207		CALL	CRLF		
3074 AF		RET			
3075 2014	DIRECCION	LD	R2,14H		; DESPLIEGUE DE ERRORES POR
3077 063200		CALL	DESP		; REINTENTOS
307A 2015		LD	R2,15H		
307C 063200		CALL	DESP		
307F 0632C7		CALL	ESPACIO		
3082 AF		RET			

800-28 SISTEMA DE DESARROLLO

0105			FORN	
0105	A000	PROC	CP	R13,R11
0107	E03942		JP	ME,MUE1
0108	0000		CP	R12,R10
010C	E03942		JP	ME,MUE1
010F	003905		JP	FINTRAS
010E	0010	MUE1	LD	R5,10H
0104	001C		LD	R4,1CH
0106	04E6		CALL	00R6 ; LEE
0100	001F		LD	R5,1FH
0100	001E		LD	R4,1EH
010C	04E9		CALL	00R8 ; GRABA
010E	00EC		DECU	RR12
0100	00EE		DECU	RR14
0102	003925		JP	PROC
0100	0F	FINTRAS	RET	

0100				
0100				
0106				" Rutina que mueve bloques de memoria en el caso de que no exista "
0006				" TRASLAP "
0106				
0106				
0100				
0100				
0100				
0106	A200	TRAS1	CP	R11,R13
0100	E03963		JP	ME,MUE
0100	A20C		CP	R10,R12
0100	E03963		JP	ME,MUE
0100	003976		JP	FINTRAS
0103	0010	MUE	LD	R4,1AH
0105	0010		LD	R5,10H
0107	04E6		CALL	00R6
0109	001E		LD	R4,1EH
0100	001F		LD	R5,1FH
0100	04E0		CALL	00R8
010F	00E0		INCH	RR10
0101	00EE		INCH	RR14
0103	003956		JP	TRAS1
0104	0F	FINTRAS	RET	

630-20 SISTEMA DE DESARROLLO

3001		FORM		
3002	0024	CLLEND	R2,RR4	
3003	0031FC	CALL	MEASAC	
3004	F30F00	LD	000,5FH	
3005	00	INC	R0	
3006	F30E00	LD	000,5EH	
3007	00E4	INCW	RR4	
3008	00	INC	R0	
3009	0F	RET		
3010	0034	FB	R5,054H	
3011	0004	LDE	R13,RR4	; RECUPERA LA DIRECCION DE RETORNO
3012	00E4	INCW	RR4	; Y SALTA A TANGOK
3013	00C4	LDE	R12,RR4	
3014	0036C6	JP	TANGOK	
3015				
3016	0C34	STC	R5,054H	
3017	0204	LPE	R13,RR4	; RECUPERA LA DIRECCION DE RETORNO Y
3018	00E4	INCW	RR4	; SALTA A TANGAK
3019	00C4	LDE	R12,RR4	
3020	003768	JP	TANGAK	
3021				
3022				
3023				
3024				
3025				
3026				
3027				
3028				
3029				
3030				
3031				
3032				
3033				
3034				
3035				
3036				
3037				
3038				
3039				
3040				
3041				
3042				
3043				
3044				
3045				
3046				
3047				
3048				
3049				
3050				
3051				
3052				
3053				
3054				
3055				
3056				
3057				
3058				
3059				
3060				
3061				
3062				
3063				
3064				
3065				
3066				
3067				
3068				
3069				
3070				
3071				
3072				
3073				
3074				
3075				
3076				
3077				
3078				
3079				
3080				
3081				
3082				
3083				
3084				
3085				
3086				
3087				
3088				
3089				
3090				
3091				
3092				
3093				
3094				
3095				
3096				
3097				
3098				
3099				
3100				
3101				
3102				
3103				
3104				
3105				
3106				
3107				
3108				
3109				
3110				
3111				
3112				
3113				
3114				
3115				
3116				
3117				
3118				
3119				
3120				
3121				
3122				
3123				
3124				
3125				
3126				
3127				
3128				
3129				
3130				
3131				
3132				
3133				
3134				
3135				
3136				
3137				
3138				
3139				
3140				
3141				
3142				
3143				
3144				
3145				
3146				
3147				
3148				
3149				
3150				
3151				
3152				
3153				
3154				
3155				
3156				
3157				
3158				
3159				
3160				
3161				
3162				
3163				
3164				
3165				
3166				
3167				
3168				
3169				
3170				
3171				
3172				
3173				
3174				
3175				
3176				
3177				
3178				
3179				
3180				
3181				
3182				
3183				
3184				
3185				
3186				
3187				
3188				
3189				
3190				
3191				
3192				
3193				
3194				
3195				
3196				
3197				
3198				
3199				
3200				
3201				
3202				
3203				
3204				
3205				
3206				
3207				
3208				
3209				
3210				
3211				
3212				
3213				
3214				
3215				
3216				
3217				
3218				
3219				
3220				
3221				
3222				
3223				
3224				
3225				
3226				
3227				
3228				
3229				
3230				
3231				
3232				
3233				
3234				
3235				
3236				
3237				
3238				
3239				
3240				
3241				
3242				
3243				
3244				
3245				
3246				
3247				
3248				
3249				
3250				
3251				
3252				
3253				
3254				
3255				
3256				
3257				
3258				
3259				
3260				
3261				
3262				
3263				
3264				
3265				
3266				
3267				
3268				
3269				
3270				
3271				
3272				
3273				
3274				
3275				
3276				
3277				
3278				
3279				
3280				
3281				
3282				
3283				
3284				
3285				
3286				
3287				
3288				
3289				
3290				
3291				
3292				
3293				
3294				
3295				
3296				
3297				
3298				
3299				
3300				
3301				
3302				
3303				
3304				
3305				
3306				
3307				
3308				
3309				
3310				
3311				
3312				
3313				
3314				
3315				
3316				
3317				
3318				
3319				
3320				
3321				
3322				
3323				
3324				
3325				
3326				

SIS-20 SISTEMA DE DESARROLLO

3035	FORM			
3035				
3035			S P	
3035			RUTINA DE COLOCACION DE BREAK POINT	
3035				
3035				
3035	54CFE	BP	AND	FLAGS,0111111100
3038	4C45		LD	R4,045H
3039	5C1C		LD	R5,01CH
303C	8214		LDE	R1,00R4
303E	56E1FE		AND	R1,0111111100
3041	9214		LDE	00R4,R1
3043	CC30		LD	R12,030H
3045	9C10		LD	R13,010H
3047	063200		CALL	DECDIR
304A	4C31		LD	R4,031H
304C	5C00		LD	R5,000H
304E	8C30		LD	R0,030H
3050	9C10		LD	R9,010H
3052	8260		LDE	R6,00R0
3054	00E0		INCM	RR0
3056	8270		LDE	R7,00R0
3058	063093		CALL	BUSIAC
305B	9264		LDE	00R4,R6
305B	00E4		INCU	RR4
305F	9274		LDE	00R4,R7
3061	00E4		INCU	RR4
3063	C206		LDC	R0,00R6
3065	9204		LDE	00R4,R0
3067	20FF		LD	R2,00FFH
3069	06E610		CP	R6,010H
306C	0007		JR	GT,CONOSIEN
306E	06E630		ADD	R6,030H
3071	9226		LDE	00R0,R2
3073	0002		JR	GUION
3075	B226	CONOSIEN	LDC	00R6,R2
3077	E6F820	GUION	LD	S10,020H
307A	063250		CALL	ESPERA
307D	E6F041		LD	S10,041H
3080	063250		CALL	ESPERA
3083	0632C9		CALL	ESPACIO
3086	2016		LD	R2,16H
3088	063200		CALL	DESP
308B	2017		LD	R2,17H
308D	063200		CALL	DESP
3090	00320E		JR	PREROUT
3093	0C00	BUSIAC	LD	R0,000H
3095	8214	BUSIAC	LDE	R1,00R4
3097	A210		CP	R1,R0
3099	6000		JR	EB,Y0
309B	00E4		INCU	RR4
309D	00E4		INCU	RR4
309F	00E4		INCU	RR4
30A1	00F2		JR	BUSIAC
30A3	0F	Y0	RET	

; R06 CONTIENE LA DIRECCION DEL BP

; CARGA EN LA TABLA

; CARGA EL CONTENIDO EN LAS TABLAS

; COLOCA UN NOP EN LA DIR

; ENVIAR UN GUION

; ENVIAR UN B'P

319-20 SISTEMA DE DESARROLLO

3011	FORM		
3011	; *****		
3011	ROUT	ROUTINA DE ATENCION A IRDS	
3011	; *****		
3011 063029	GOONTO	CALL	ATRACE
3014 00305F		JP	RESPALDAU
3017 06304F	GOONTO	CALL	ATBP
3018 00305F		JP	RESPALDAU
3010 063029	CTOATO	CALL	ATRACE
3020 0030EF		JP	DTRCALL
3023 06304F	COONTO	CALL	ATBP
3026 0030EF		JP	DTRCALL
3029	; *****		
3029 0045	ATRACE	LD	R0,045H
3020 ECF0		LD	R14,00F0H
3020 D20E		LDC	0RR14,R0
302F 063020		CALL	IRDSALUA
3032 ECF0		LD	R14,030H
3034 FCF0		LD	R15,050H ; TRAE EL CONTADOR DE TRACE Y LO DECREMENTA
3036 820E		LDE	R0,0RR14
3038 00E0		DEC	R0
303A 603040		JP	Z,APAGAT
3030 920E		LDE	0RR14,R0
303F AF		RET	
3040 56FCFE	APAGAT	AND	FLAGS,011111110B
3043 920E		LDE	0RR14,R0
3045 0C01		LD	R0,001H ; SI EL CONTADOR ES CERO APAGA BANDERA
3047 ECF0		LD	R14,00F0H ; DE TRACE Y SE PONE EN MODO 1 DEL PUERTO
3049 D20E		LDC	0RR14,R0 ; SERIE APAGA EL BIT DE IRDS DE INH
3048 56F0FE		AND	INH,011111110B
304E AF		RET	
304F	; *****		
304F 0C05	ATBP	LD	R0,0C5H
3051 ECF0		LD	R14,00F0H
3053 D20E		LDC	0RR14,R0
3055 063002		CALL	IRDSALUA
3058 ECF0		LD	R14,030H ; AL OCURRIR UN BP ES NECESARIO AJUSTAR
305A FCF0		LD	R15,054H ; LA DIRECCION DE RETORNO YA QUE ESTA ES
305C 820E		LDE	R13,0RR14 ; GUARDADA COMO DIR BP+1
305E 00EE		INCH	RR14
3060 32CE		LDE	R12,0RR14
3062 00EC		DECU	RR12
3064 92CE		LDE	0RR14,R12
3066 00EE		DECU	RR14
3068 920E		LDE	0RR14,R13

819-28 SISTEMA DE DESARROLLO

706A		FORM		
706A	EC31	LD	R14,031H	/
706C	FC00	LD	R15,000H	:
706E	MC00	LD	R0,000H	:
7070	70E0	PUSH	R0	:
7072	000E	LDE	R0,RRR14	:
7074	000C	CP	R0,R12	:
7076	000A	JR	NE,0AT01	:
7078	00EE	INCU	RR14	:
707A	000E	LDE	R0,RRR14	:
707C	0200	CP	R0,R13	:
707E	0034	JR	EQ,0ATDES	:
7080	00EE	DECU	RR14	:
7082	00EE	INCU	RR14	:
7084	00EE	INCU	RR14	:
7086	00EE	INCU	RR14	:
7088	00E0	POP	R0	:
708A	00E4	DMZ	R0,0AT02	/
708C	00EE	POP	R14	:
708E	00EE	POP	R14	:
7090	00EE	POP	R14	:
7092	00EE	POP	R14	:
7094	F017	LD	R15,17H	:
7096	50EF10	AND	R15,00001000B	:
7098	000A	JR	Z,SINH	:
709A	70EF00	TH	R15,000001000B	:
709C	0000	JR	Z,C0NH	:
709E	70EF10	TH	R15,000010000B	:
70A0	0004	JR	Z,C0NH	:
70A2	CCF0	LD	R12,00F0H	:
70A4	CC05	LD	R14,00ASH	:
70A6	00EC	LDC	RRR12,R14	:
70A8	E410F0	LD	P01H,10H	:
70AA	0417F7	LD	P3H,17H	:
70AC	00004E	JP	VAR0005	:
70AD			/	
70AE	00EE	INCU	RR14	:
70B0	000E	LDE	R0,RRR14	:
70B2	00EC10	CP	R12,010H	:
70B4	0020C3	JP	GT,C0MB0AT	:
70B6	00EC30	ADD	R12,030H	:
70C1	000C	LDE	RRR12,R0	:
70C3	0002	JR	ELININA	:
70C5			:	:
70C7	000C	LDC	RRR12,R0	:
70C9	0003	LD	R12,000H	:
70CA	000E	LD	R0,003H	:
70CB	000E	LDE	RRR14,R12	:
70CC	000E	DECU	RR14	:
70CF	00FA	DMZ	R0,0UITA	/
70D1	0F	RET		

DETERMINA SI ESTA DIRECCION CONTENIA
UN BP VERDADERO COMPARANDO CON LA
TABLA DE BP

DE NO ENCONTRAR ESTA DIRECCION
EN LA TABLA REGRESA AL PROGRAMA
DE PRUEBA

ELIMINA EL BP DE LA TABLA DE BP
Y LA DIRECCION PARTICULAR

319-20 SISTEMA DE DESARROLLO

3002		FORM		
3002				
3002	0C08	INOSALVA	LD	R0,000H ; \
3004	CC30		LD	R12,000H ;
3006	0C30		LD	R13,000H ;
3008	50EF	NECESARIOS	POP	R15 ;
300A	52FC		LOE	RR12,R15 ;
300C	00EC		DECU	RR12 ;
300E	0AF0		DJNZ	R0,NECESARIOS ; RECUPERA LOS REGISTROS QUE ESTABAN
3008	0C08		LD	R0,000H ; EN EL STACK Y LOS GUARDA EN MEMORIA
3012	00FE		LD	R12,SPH ; DE DATOS A PARTIR DE LA DIRECCION
3014	00FF		LD	R13,SPH ; 3054 HASTA LA 305D
3016	00EC	CORRIGE	DECU	RR12 ;
3018	0AF0		DJNZ	R0,CORRIGE ;
301A	C7FE		LD	SPH,R12 ;
301C	D7FF		LD	SPL,R13 ;
301E	AF		RET	; /
301F				
302F	0632D	OTRCALL	CALL	DOU ;
3032	8C10		LD	R0,010H ;
3034	0C08		LD	R0,000H ;
3036	063247		CALL	CRLF ; CONTADOR DE REGISTROS R0 Y R0 APUNTA A REGISTROS
3039	7C08		LD	R7,000H ; \
303B	9C08		LD	R9,000H ;
303D	063C0E		CALL	MUESTRAHE ;
3040	8C10		LD	R0,010H ; DESPLIEGA EL CONTENIDO DE LOS REGISTROS
3042	0C08		LD	R0,00F0H ; DEL ARCHIVO INTERNO
3044	7C01		LD	R7,001H ;
3046	9C08		LD	R9,00F0H ;
3048	063C0E		CALL	MUESTRAHE ; /
304B	0032AE		JP	PREROUT ;
304E	2819	MUESTRAHE	LD	R2,19H ;
3010	063280		CALL	DESP ;
3013	0632C9		CALL	ESPACIO ;
3016	0632C9		CALL	ESPACIO ;
3019	E320	Z2	LD	R2,R0 ;
301B	063280		CALL	DESP ;
301E	0632C9		CALL	ESPACIO ;
3021	0E		INC	R0 ; SIGUE REGISTRO
3022	9E		INC	R9 ;
3023	0AF4		DJNZ	R0,22 ; SIGUE DESPLIEGANDO LOS REGISTROS DEL MISMO GRUPO
3025	0C10		LD	R0,010H ; REINCORPORA CONTADOR
3027	063247		CALL	CRLF ;
302A	7AE2		DJNZ	R7,MUESTRAHE ;
302C	AF		RET	;
302D				
302D	2C01	DOU	LD	R2,001H ; \
302F	063242		CALL	MODOS ;
3032	063136		CALL	XS ; CAMBIA MODO 1 DE PUERTO SERIE
3035	4C30		LD	R4,030H ; DETERMINA VELOCIDAD DE TRANSMISION
3037	5C54		LD	R5,054H ;
3039	8C28		LD	R0,040 ;
303B	063247		CALL	CRLF ;

925-29 SISTEMA DE DESARROLLO

9238		FORM			
9238	0632C9	FORMAT	CALL	ESPACIO	:
9241	00F0		DJNZ	R0,FORMAT	:
9243	6C3F		LD	R0,0000IR>0	:
9245	7C0A		LD	R7,0000IR>256	:
9247	0632B7		CALL	EMUSINT	:
9248	0224		LDE	R3,00R4	:
924C	00E4		INCH	RR4	:
924E	0224		LDE	R2,00R4	:
9250	0632B0		CALL	DESP	:
9253	2B13		LD	R2,13H	:
9255	0632B0		CALL	DESP	:
9258	063247		CALL	CRLF	:
9259	063247		CALL	CRLF	:
925C	AF		RET		:
925F					
926F	EC09	RESPALDA	LD	R14,009H	:
9281	CC45		LD	R12,045H	:
9283	0C11		LD	R13,011H	:
9285	0C11		LD	R0,011H	:
9287	500C	PASALOS	LDEI	00R12,000	:
9289	0AFC		DJNZ	R14,PASALOS	:
928B	0CFA		LD	R0,00FAH	:
928D	530C		LDEI	00R12,000	:
928F	530C		LDEI	00R12,000	:
9291	0C1F		LD	R13,01FH	:
9293	2003		LD	R2,003H	:
9295	50E1	SPNFLAG	POP	R1	:
9297	921C		LDE	00R12,R1	:
9299	00EC		DECH	RR12	:
929B	20F0		DJNZ	R2,SPNFLAG	:
929D	EC30		LD	R14,030H	:
929F	FC53		LD	R15,053H	:
92A1	021E		LDE	R1,00R14	:
92A3	921C		LDE	00R12,R1	:
92A5					:
92A7	0C00		LD	R13,000H	:
92A9	0C00		LD	R0,000H	:
92AB	106F		LD	R1,06FH	:
92AD	930C	LOSDEMAS	LDEI	00R12,000	:
92AF	06E010		CP	R0,010H	:
92B0	ED3C97		JP	NE,SIMPRO	:
92B2	0C20		LD	R0,020H	:
92B4	0C20		LD	R13,020H	:
92B6	10F2	SIMPRO	DJNZ	R1,LOSDEMAS	:
92B8	CC45		LD	R12,045H	:
92BA	0C02		LD	R13,002H	:
92BC	EC30		LD	R14,030H	:
92BE	FC57		LD	R15,057H	:
92C0	021E		LDE	R1,00R14	:
92C2	921C		LDE	00R12,R1	:

DESPLIEGA LA DIRECCION DONDE
OCURRIO LA INTERRUPCION

RESPLAZA LOS REGISTROS SPN, SPL Y STATUS
QUE SE ENCUENTRAN EN EL STACK

PASA EL RESTO DE LOS REGISTROS AL ARCHIVO
ALTERNO

SIS-20 SISTEMA DE DESARROLLO

3C45	D63C2D	OTREG	CALL	DDU	:	DESPLIEGA DIRECCION DE INTERRUPCION
3C46	4C45		LD	R4,045H	:	
3C4A	3C30		LD	R5,000H	:	
3C4C	7C38		LD	R7,000H	:	
3C4E	9C10		LD	R9,010H	:	
3C50	9C30		LD	R9,000H	:	
3C52	D63CC1		CALL	MUSTR6	:	
3C55	7C31		LD	R7,001H	:	
3C57	9C38		LD	R9,00F0H	:	
3C59	5C10		LD	R5,010H	:	
3C5B	D63CC1		CALL	MUSTR6	:	
3C5E	80324E		JP	PRER00T	:	DESPLIEGA EL ARCHIVO DE REGISTROS ALTERN0
3C61					:	TAL COMO QUEDARON ANTES DE LA
3C61	2019	MUSTR6	LD	R2,19H	:	INTERRUPCION
3C63	D63220		CALL	DESP	:	
3C66	D632C9		CALL	ESPACIO	:	
3C69	D632C9		CALL	ESPACIO	:	
3C6C	8224	OTREG	LDE	R2,0RR4	:	
3C6E	D63280		CALL	DESP	:	
3C71	D632C9		CALL	ESPACIO	:	
3C74	00E4		INCW	RR4	:	
3C76	9E		INC	R9	:	
3C77	80F3		DJNZ	RR,OTREG	:	
3C79	9C10		LD	R8,010H	:	
3C7B	D63247		CALL	CRLF	:	
3C7E	7AE1		DJNZ	R7,MUSTR6	:	
3C80	AF		RET		:	/

SIS-Z8 SISTEMA DE DESARROLLO

3E34	0000044 47524543 4344474E 20444E36 41004944 41004F	LET	DB	MUL,CR,LF,'DIRECCION INVALIDA',CR,FF
3E40	00000432 45444953 54524F20 44452053 4F4C4F20 45544252 49545352 41004AFF	SULDESC	DB	MUL,CR,LF,'REGISTRO DE SOLO ESCRITURA',CR,LF,FF
3E68	0000044E 55404532 4F204445 20524547 49535452 4F20494E 56414C49 444F0D0A FF	REGIMU	DB	MUL,CR,LF,'NUMERO DE REGISTRO INVALIDO',CR,LF,FF
3E8C	00000453 49532050 30204C49 53544F20 50415241 20524543 49424952 20494E53 54525543 43494F4E 45530D0A 2AFF	REMACC	DB	MUL,CR,LF,'SIS Z8 LISTO PARA RECIBIR INSTRUCCIONES',CR,LF,"",FF
3EBA	00000453 4543544F 52202320 31303030 40205AFF	SECMO	DB	MUL,CR,LF,'SECTOR 0 1000H P',FF
3ECE	00000453 4543544F 52202320 35303030 40205AFF	SECDOS	DB	MUL,CR,LF,'SECTOR 0 5000H P',FF
3EE2	00000453 4543544F 52202320 31303030 40204AFF	SECTRES	DB	MUL,CR,LF,'SECTOR 0 1000H D',FF

END-28 SISTEMA DE DESARROLLO

00000000	SECCION	DB	MUL,CR,LF,'SECTOR 0 30000 D',FF
00000001			
00000002			
00000003			
00000004			
00000005	MEMESIN	DB	MUL,CR,LF,'ERROR DE SINTAXIS',CR,LF,'N',FF
00000006			
00000007			
00000008			
00000009			
00000010			
00000011			
00000012			
00000013	SIGN	DB	MUL,CR,LF,'SIGUIENTE LOCALIDAD =',CR,LF,FF
00000014			
00000015			
00000016			
00000017			
00000018			
00000019			
00000020	OPINMA	DB	MUL,CR,LF,ESP,'OPCION INVALIDA DE MOVIMIENTO DE MEMORIA'
00000021			
00000022			
00000023			
00000024			
00000025			
00000026			
00000027			
00000028			
00000029			
00000030			
00000031			
00000032			
00000033			
00000034			
00000035			
00000036			
00000037			
00000038			
00000039			
00000040			
00000041			
00000042			
00000043			
00000044			
00000045			
00000046			
00000047			
00000048			
00000049			
00000050			
00000051			
00000052			
00000053			
00000054			
00000055			
00000056			
00000057			
00000058			
00000059			
00000060			
00000061			
00000062			
00000063			
00000064			
00000065			
00000066			
00000067			
00000068			
00000069			
00000070			
00000071			
00000072			
00000073			
00000074			
00000075			
00000076			
00000077			
00000078			
00000079			
00000080			
00000081			
00000082			
00000083			
00000084			
00000085			
00000086			
00000087			
00000088			
00000089			
00000090			
00000091			
00000092			
00000093			
00000094			
00000095			
00000096			
00000097			
00000098			
00000099			
00000100			
00000101			
00000102			
00000103			
00000104			
00000105			
00000106			
00000107			
00000108			
00000109			
00000110			
00000111			
00000112			
00000113			
00000114			
00000115			
00000116			
00000117			
00000118			
00000119			
00000120			
00000121			
00000122			
00000123			
00000124			
00000125			
00000126			
00000127			
00000128			
00000129			
00000130			
00000131			
00000132			
00000133			
00000134			
00000135			
00000136			
00000137			
00000138			
00000139			
00000140			
00000141			
00000142			
00000143			
00000144			
00000145			
00000146			
00000147			
00000148			
00000149			
00000150			
00000151			
00000152			
00000153			
00000154			
00000155			
00000156			
00000157			
00000158			
00000159			
00000160			
00000161			
00000162			
00000163			
00000164			
00000165			
00000166			
00000167			
00000168			
00000169			
00000170			
00000171			
00000172			
00000173			
00000174			
00000175			
00000176			
00000177			
00000178			
00000179			
00000180			
00000181			
00000182			
00000183			
00000184			
00000185			
00000186			
00000187			
00000188			
00000189			
00000190			
00000191			
00000192			
00000193			
00000194			
00000195			
00000196			
00000197			
00000198			
00000199			
00000200			
00000201			
00000202			
00000203			
00000204			
00000205			
00000206			
00000207			
00000208			
00000209			
00000210			
00000211			
00000212			
00000213			
00000214			
00000215			
00000216			
00000217			
00000218			
00000219			
00000220			
00000221			
00000222			
00000223			
00000224			
00000225			
00000226			
00000227			
00000228			
00000229			
00000230			
00000231			
00000232			
00000233			
00000234			
00000235			
00000236			
00000237			
00000238			
00000239			
00000240			
00000241			
00000242			
00000243			
00000244			
00000245			
00000246			
00000247			
00000248			
00000249			
00000250			
00000251			
00000252			
00000253			
00000254			
00000255			
00000256			
00000257			
00000258			
00000259			
00000260			
00000261			
00000262			
00000263			
00000264			
00000265			
00000266			
00000267			
00000268			
00000269			
00000270			
00000271			
00000272			
00000273			
00000274			
00000275			
00000276			
00000277			
00000278			
00000279			
00000280			
00000281			
00000282			
00000283			
00000284			
00000285			
00000286			
00000287			
00000288			
00000289			
00000290			
00000291			
00000292			
00000293			
00000294			
00000295			
00000296			
00000297			
00000298			
00000299			
00000300			
00000301			
00000302			
00000303			
00000304			
00000305			
00000306			
00000307			
00000308			
00000309			
00000310			
00000311			
00000312			
00000313			
00000314			
00000315			
00000316			
00000317			
00000318			
00000319			
00000320			
00000321			
00000322			
00000323			
00000324			
00000325			
00000326			
00000327			
00000328			
00000329			
00000330			
00000331			
00000332			
00000333			
00000334			
00000335			
00000336			
00000337			
00000338			

SIS-20 SISTEMA DE DESARROLLO

3FB8	00000430	UNE	DB	MUL,CR,LF, 'PRUEBA DE VERIFICACION DESPUES DE ESCRITURA',FF
	52304542			
	41204445			
	28364532			
	49464943			
	4143494F			
	4E204445			
	53205345			
	53204445			
	28435343			
	52495435			
	5241FF			
3FE7	00000430	LDN	DB	MUL,CR,LF, 'PRUEBA DE LECTURA',CR,LF,FF
	52554542			
	41204445			
	28404543			
	54535241			
	0004FF			
0000	ERRORS			

810-20 SISTEMA DE DESARROLLO

SIGNAL TABLE

ACORDADA	3100	AGUANTA	3127	ALTERA	3600	AP	0012
APROBAT	3040	ASCEX	3100	AT	3700	ATBP	304F
ATENT1	002C	ATENT2	0032	ATENT3	0030	ATENT4	003E
AVENTS	0044	ATRACE	3029	BBP	30AC	BLEN	3023
BP	3035	BS	0000	BLUFF	3044	BURNS	3095
BAND	3043	BUSCAFI	30F1	BUSCAINT	323E	BUSMAC	3093
BK	3004	BK2	30E2	BK21	3000	BK3	3000
BK4	3000	BKERR	3F6C	BKERR	3001	BKRRS	30E2
BVE	3100	CIDACB	370C	C4KB	3030	CALL	3761
CAME	3063	CARG	3330	CAN4	309A	CAMBOAT	30C3
CANAB	3050	CAREG	303F	CBOPOAT0	3023	CCORNA	32EA
COONE	3990	CERO	3050	CIGUE	3214	CINC	3909
CLEMAN	3903	CORA	002C	CONOSIEM	3075	COMP	3640
COMP	39ED	COMP1	39EB	COMUN	372C	CONON	30AB
CONLEC	300F	CONRET	370E	CONT	3110	CONTR	0013
CONFI	3221	CONTINI	31E0	CONTIN2	31ED	CONTRR	35A6
CONLINE	30E6	CR	0000	CROEL	31AE	CRLF	3247
CTOAT0	3010	CTRLF	0006	CUATRO	30D9	CUNCB	37DA
DADIR	3F00	DAT	3366	DATA	3342	DATDAT	30D2
DATPROM	30E0	DB	39F9	DBDEC	3019	DBRAX	39FF
DBRAX	301E	DBSIES	3011	DDCP	3307	DDU	3C20
DEC	3319	DEL	007F	DELETE	30F0	DESDIR	3610
DESDIR1	3619	DESGRUPO	34CA	DESP	3200	DESPL2	3627
DESPL3	363E	DESPL4	3640	DIR1	32F1	DIRECCION	3075
DI	35F9	DINAB	3754	DOS	3073	DOSBY	3A26
DI	349E	DIRATA	3340	DTRCALL	30EF	DTRGO	30A5
DIND	3004	ELINIM	30C7	ENTRADA	0050	ENRASC	3569
ENVSINTNT	3200	ENVSINT	3207	EOF	0004	ERR	3310
ENRI	3324	ESCL	3447	ESCOMT	30D6	ESP	0020
EPPO	34E0	ESPACIO	32C9	ESPACIO2	34E9	ESPDAT	3175
EPVERA	3250	ESTONO	3732	ESTOS1	3736	EVO	365E
FALLA	3049	FF	00FF	FG	39E3	FI	00FE
FIN	3601	FINGRUPO	3560	FINRU	3900	FINOTRAS	3976
FINPO	3374	FINTRAS	3955	FINRUK	350C	FINUN	3557
FLACS	00FC	FORPAT	3C3E	FT	00FD	FUETRACE	3779
0027	3720	GBA	3697	GBAREG	3300	GENERAL	0044
GOVY	3379	GO	36A2	GOBPOAT0	3017	GOTOAT0	3011
GRUP1	3550	GRUPO	3550	GUION	3077	HACHE	0040
HEMOR	31F2	HEMASC	31FC	IGUAL	0030	INI	32E2
INICIO	3201	INIREG	30BF	INIT	3005	INTERNA	377E
INOSALVA	3002	K	3977	LCID4	3014	LON	3FE7
LEZ	3016	LET	3634	LETRAS	321E	LF	000A
LIMP10	3200	LIMP10	3375	LINER	3570	LINEROK	3560
LOGRERAS	3000	LSIEN	3010	LINK	3012	MA	3569
ROBIR	3460	REMI	3921	REHACC	30BC	RENESEN	3F0A

SIS-20 SISTEMA DE DESARROLLO

MESS	30F0	MIBOS	3242	NON	3387	MONSUP	3808
NR	330A	MIBUNA	3506	NST	3432	MUE	3963
NR1	3942	MIBSTRANE	3C0E	MUEVE	0963	MIBSTRG	3CC1
NR	3003	MIBT	3170	NECESARIOS	3000	MIBS2	3741
NR1NT	3020	MIBMETRIES	3777	NORMAL	360F	MIBTRAS	3956
NR000	3309	MIL	0800	MUNDOS	34F1	MURKX	34F8
NR0000	34E6	0AT02	3070	0ATDES	3004	0ATO1	3882
OPINION	3F30	0TRA	3220	0TREG	30CC	PARAK	32A3
PARIDAD	3154	PAS0	37EF	PASALOS	3C67	PBAJA	3300
PCBL	30C1	PA	37CC	POD	334E	PRNG	3348
PCF	33F4	PMBT	3730	PREPRO	3301	PREREG	3290
PREBOOT	320E	PRINC	3707	PROC	3935	PROG	3345
PROBAT	30F3	PROGPROG	3000	PROMPT	30F3	PRUENO	34A6
PSC	390E	QUITA	30C8	REC01R	3200	RECEPCION	3053
RECIDE	3393	RECREG	3400	REG	001A	REG1	001E
REG2	001F	REGINN	3660	REG20	3549	RESPALAMU	3C5F
REST	3533	REST1	39CC	RETA	3955	RETRIES	3965
RIBT	3040	RDX	3500	SAL	30BA	SALIDA	327C
SALPRO	0073	SALTO	366A	SALUNC	332E	SECCUAT	3EF6
SELDOS	36CE	SECTRES	3EE2	SECUNO	3EBA	SIEMPRE	37E0
SIG	362F	SIG1	3653	SIG2	3672	SIGN	3F22
SINON	30A5	SINPRO	3C97	SINTAX	3187	SN	3448
SOLDESC	3640	SPOFLAG	3C75	STAND	3437	STC	39EE
T	37A1	TABL	3CE1	TANCAK	3768	TANGOK	36C6
TAPI	3E2C	TAPH	3E14	TDMO	312B	TODOS	380D
TRANSFER	3605	TRAS	390E	TRAS1	3956	TRASHO	3903
TRASSI	3908	TRERR	3F9E	TRES	3088	TRFR	33C3
TRN	3300	UNO	3068	UNO1	3100	UNO2	31EA
UNREG	3500	USUARIO	3408	UAMONOS	006E	VDE	3FB8
VEATNC	3490	UTRAS	30FB	XCALC	314E	XDECOD	313A
XEND	316E	XHMSUEL	3148	XOFF	0013	XDN	3277
XS	3136	XSON	3168	YA	3AA3	YABX	3AC6
Z2	3C19	ZIGUE	329C				

CAPITULO VII

DISEÑO DEL SIS Z8

El objetivo de este capítulo 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 esto 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 curso una materia específica 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 empírico podría 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 si 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 ZB.

Volviendo un poco al SIS ZB, para realizar un prototipo de laboratorio se requiere de muchos Mapas de Karnaugh, del cálculo de resistencias 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 ZB 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 quien 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

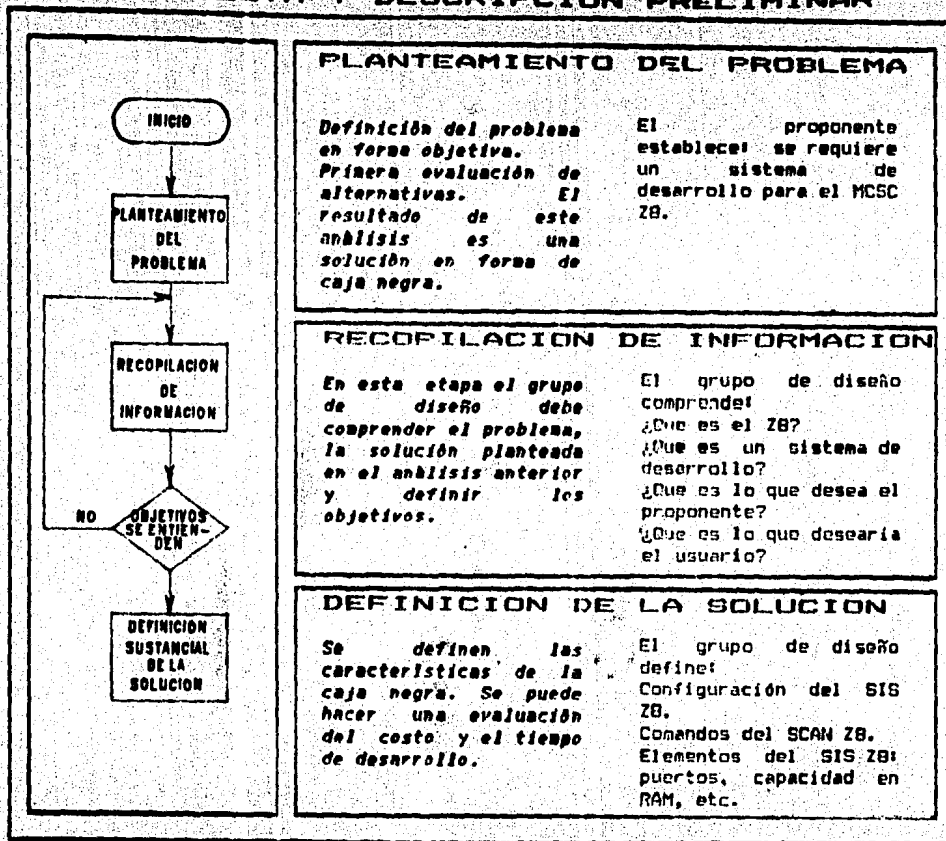


FIGURA (7.2 a)

PROPUESTA Y DESCRIPCION PRELIMINAR

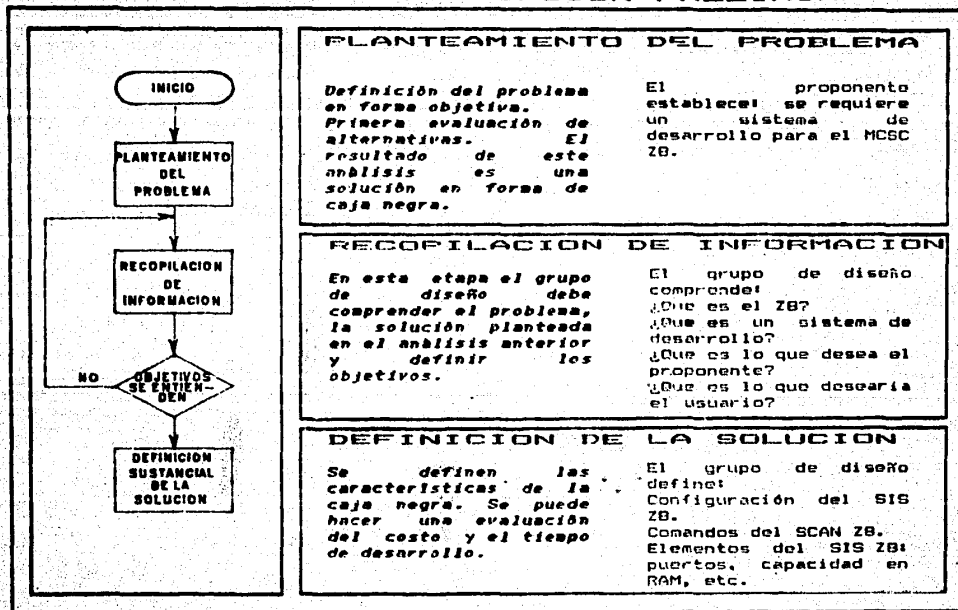


FIGURA (7.2 a)

DISEÑO PRELIMINAR

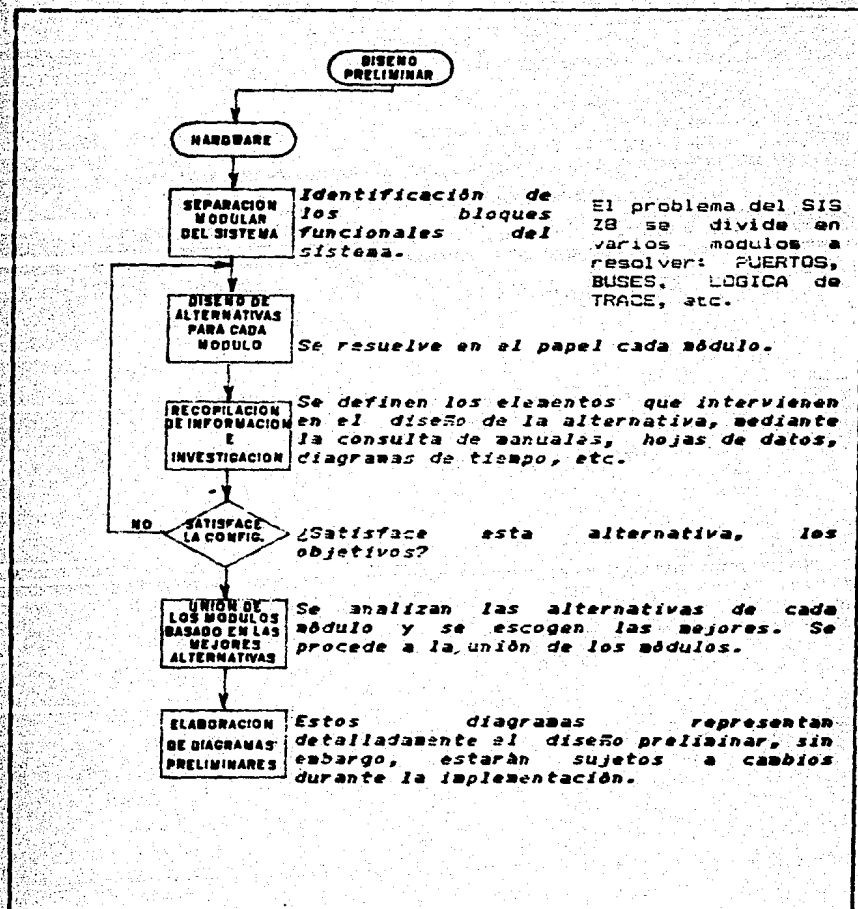


FIGURA (7.2 b)

DISEÑO PRELIMINAR (SOFTWARE)

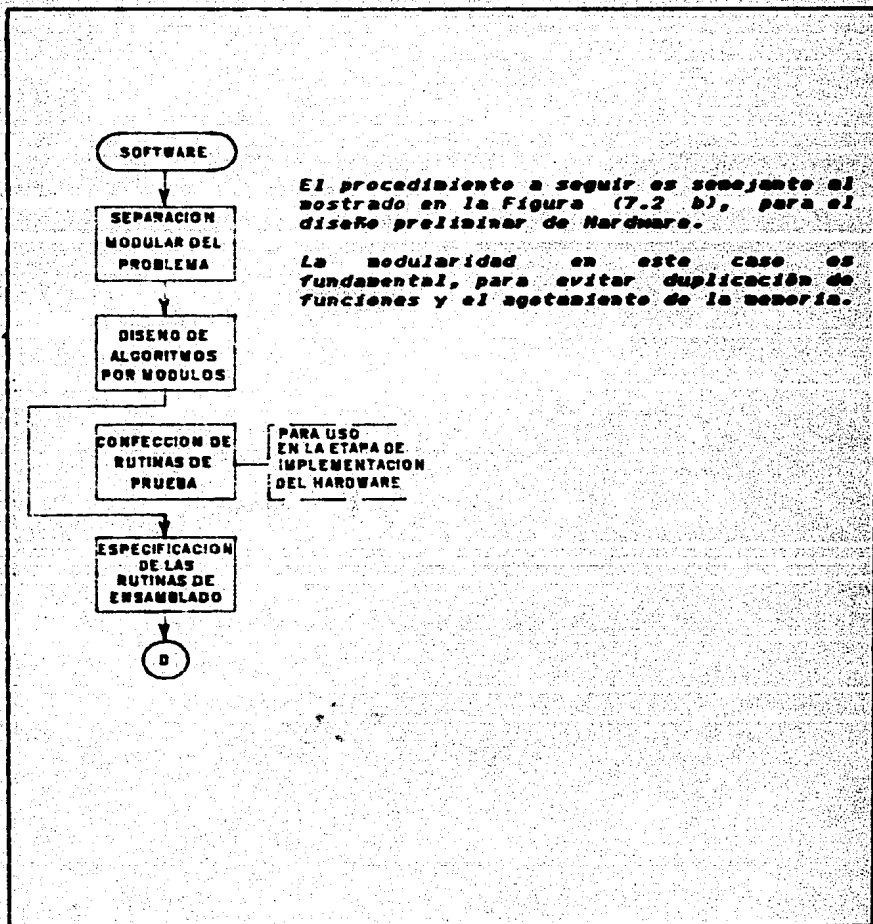
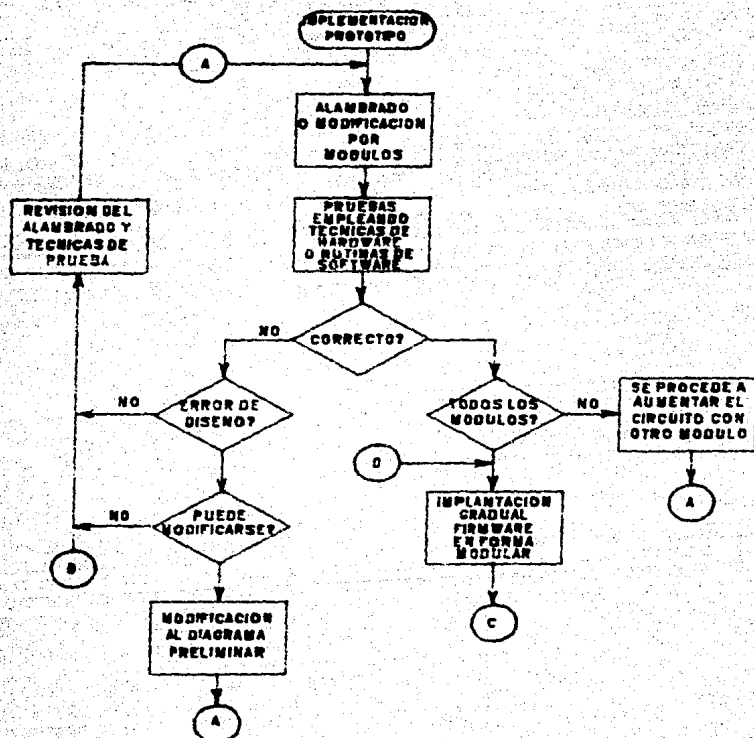


FIGURA (7.2 c)

IMPLEMENTACION DEL PROTOTIPO Y DOCUMENTACION (a).



Con este procedimiento se busca facilitar el proceso de depuración del Hardware y del Software.

FIGURA (7.2 d)

IMPLEMENTACION DEL PROTOTIPO Y DOCUMENTACION (b).

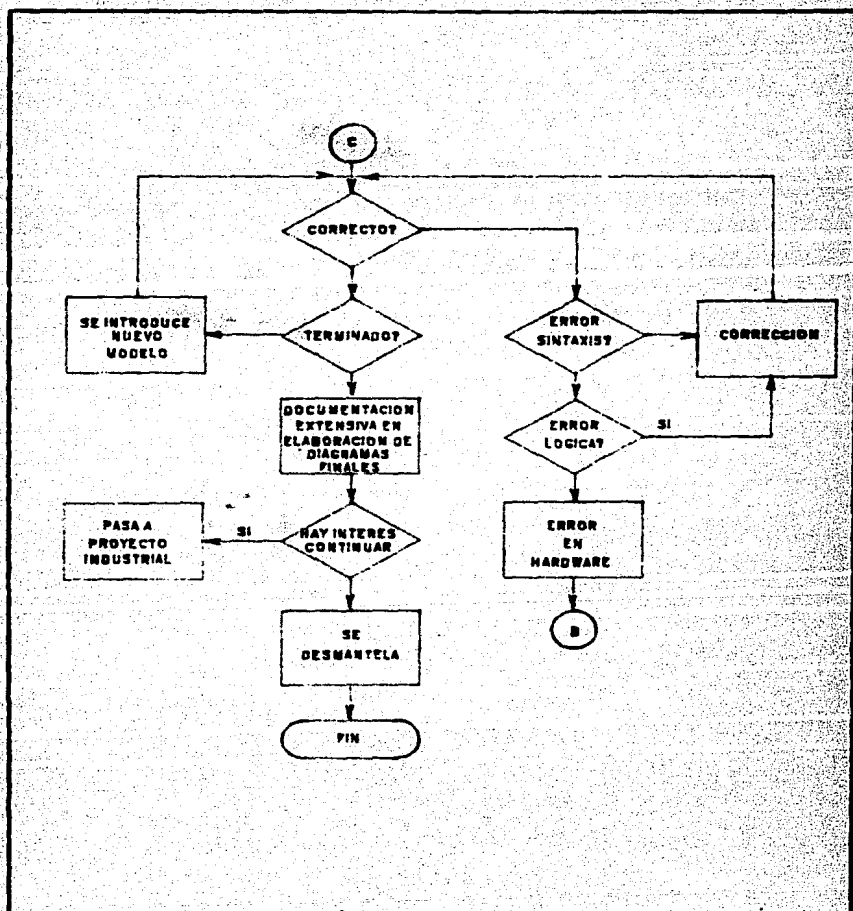


FIGURA (7.2 a)

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 cual 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 cual 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 ZB 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 SICOM ZB, 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 incluidas, por quedar fuera de presupuesto y tiempo. Por esto, consideramos ampliamente recomendable agregar una evaluación de costos y tiempo, que en nuestro caso no fue 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 esto 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 esto 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 discernirse de los restantes para después atacar, independientemente, cada uno hasta completar todos los módulos.

Así 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 líneas 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 módulos:

- 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 fue estudiado en su oportunidad en el Capítulo 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 fue 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 (aunque 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 periódicas 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 28. 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 empaquetan una gran cantidad de funciones. Estos pueden resolver fácilmente muchos problemas, sin embargo, el mercado nacional de estos dispositivos no es vanguardista y al 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 Capítulo 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) fue 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 fue 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 esto, el SIS Z8 fue 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 o 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 fue necesaria una labor previa de mantenimiento. Aún así, obsoleto o no, este equipo, gracias al pleno conocimiento que de él teníamos, fue 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 almacén de aparatos de laboratorio, que al final mueren en la obsolescencia 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.

3.-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 28.

Rotón de
Bont



- Anillos de
conectores

- Cables de
Unión

Fotografía # 11 El prototipo del SIS 28,
debajo de los cables de unión se encuentra la
tarjeta impresa con el 20412.

Arnés de
conectores

Led de
Encendido

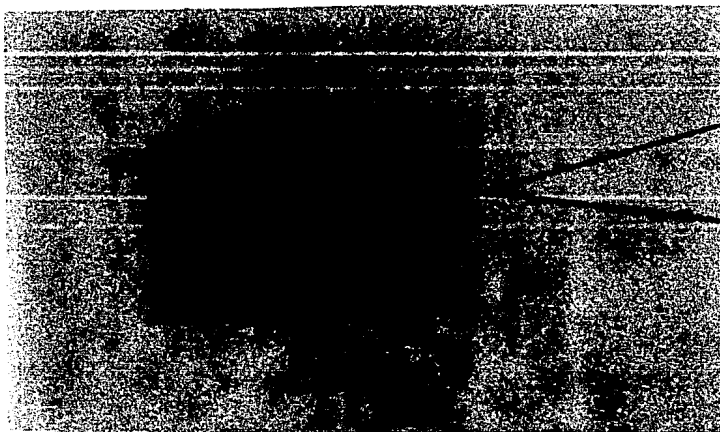


Memorias
RAM

Monitor
Principal

Monitor
Interno

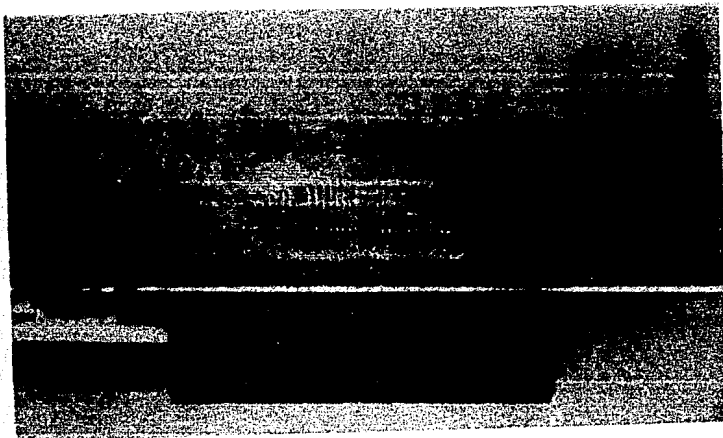
Fotografía # 2: Tarjeta principal del SIS 28.
Se muestra la disposición de los elementos.



Z8612

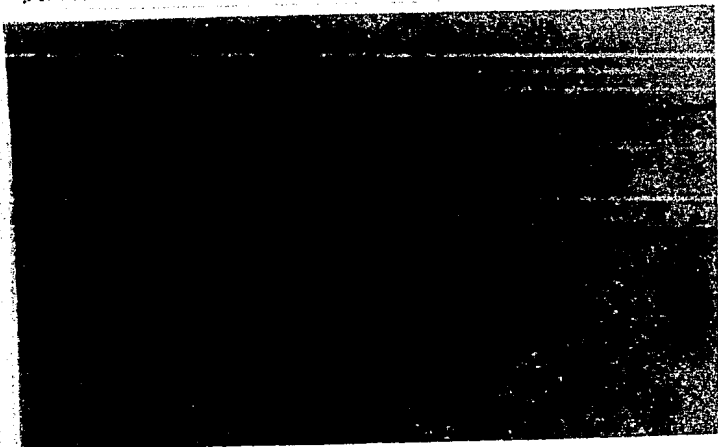
Bases para
Cables de
Unión

Fotografía # 3: Tarjeta secundaria (Z8612).



Cristal de
cuartzo

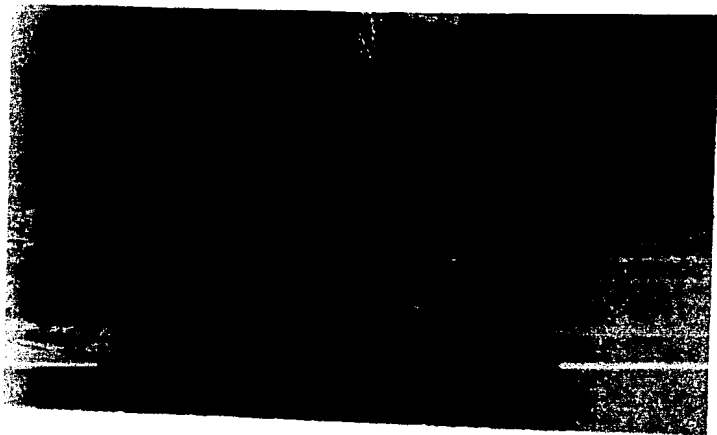
Fotografía # 4: Vista anterior de la tarjeta secundaria. El cristal de cuarzo (1) soldado a esta tarjeta debido a la soldadura de sus patillas.



Cable de
Unión

Bases para
Cables de
Unión

Fotografía # 5: Tarjeta Secundaria.



Cables de
Unión

Bases para
Cables de
Unión

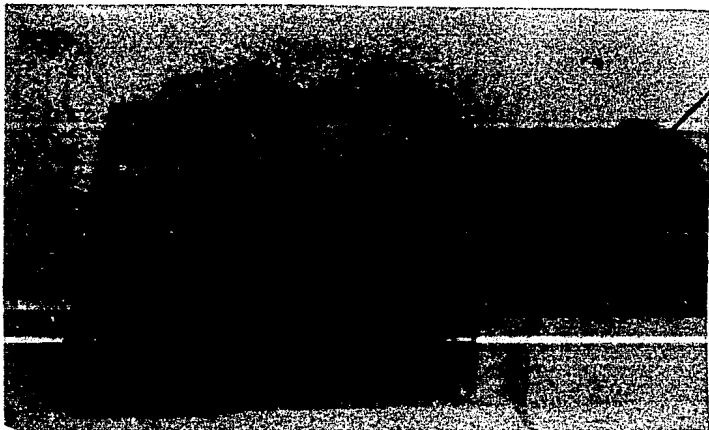
Fotografía # 6: Detalle de conectores de cables de unión.



Caimanes
para
Fuentes
de Poder

Conectores
DB-25 de
Puentes
Serie

Fotografía # 7: Detalle de arneses de conectores.

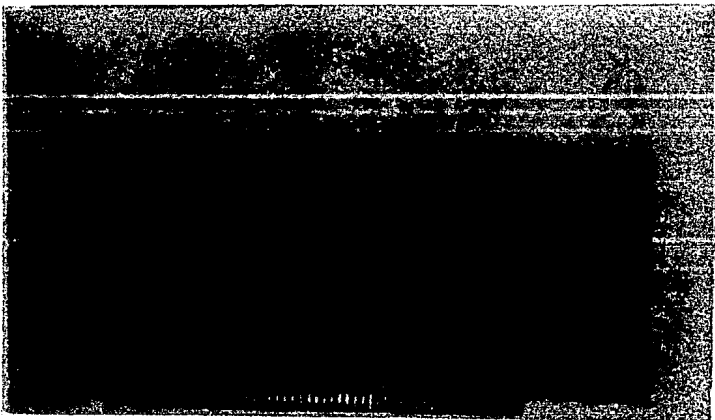


Fuentes de
Poder.

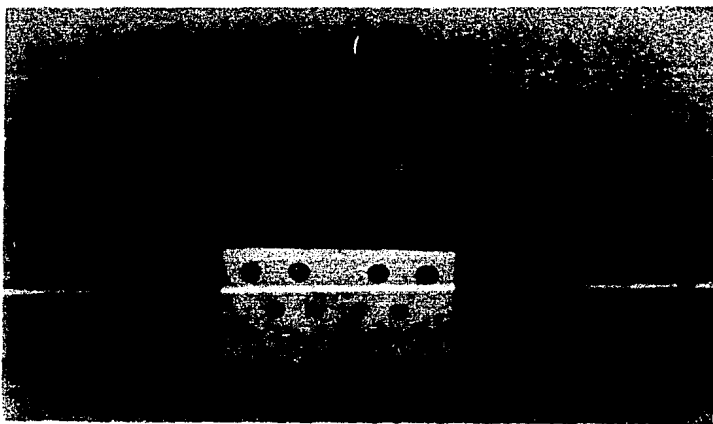
CPU
CROMEMCO

Unidad de
discos
flexibles

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 1/4".



Fotografía # 9: Tarjeta 32KBYTESAVER empleada en el grabado de EPROMS 2716.



Fuente fija
de -12V

Fuentes
variables
empleadas
para
suministrar
+5V y +12V

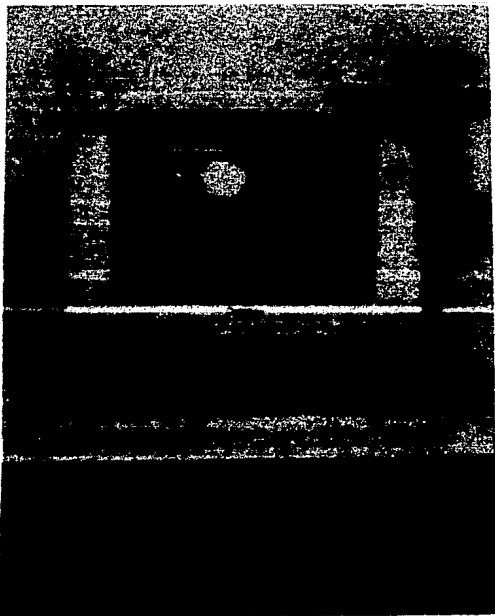
Fotografía # 10: Las fuentes de Poder empleadas para energizar al SIS Z8.

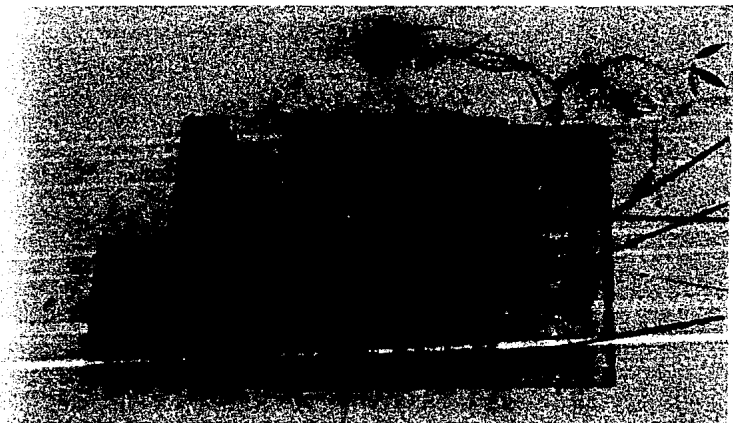


Etiquetas

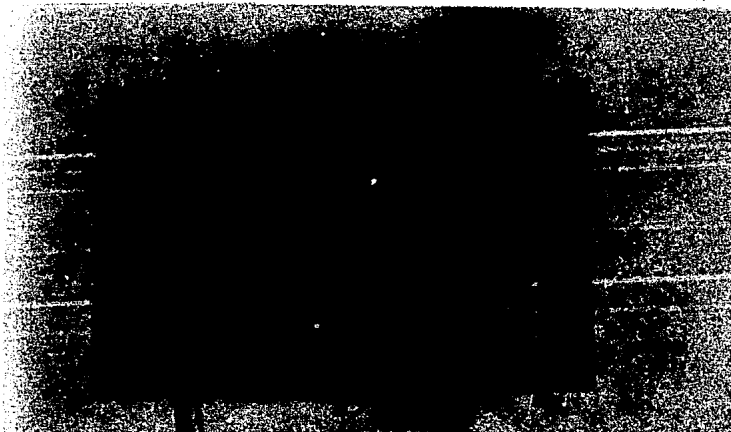
Fotografía # 11: Detalle de conexión del SIS Z8 a las fuentes de poder. Cada caimán tiene una etiqueta que indica su conexión.

Fotografias # 12 y 13
Terminal empleada para
desarrollo del SIS Z8
y detalle de conección
del puerto serie del
SIS Z8 con la terminal.





Fotografia # 14: Distribución de los bloques del SIS Z8 sobre la Tarjeta Principal.



Fotografia # 15: El bus de datos/direcciones de tal manera que podía zafarse un cable sin afectar la operación del SIS Z8.

Puertos
 Serie
 Control
 de buses
 Lógica de
 Trace y BP
 Puerto Int.
 de control

Sin contar el tiempo de importación del Z8612, la implementación del SIS Z8 consumió por sí sola aproximadamente un año. Considerando que, aproximadamente, se disponía diario de 2 a 3 horas de laboratorio, 2 o 3 días 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 fue concluida en un tiempo record, por lo menos sí fue adecuada. Uno de los aciertos que permitieron concluir el SIS Z8, fue la limpieza sistemática del alambrado. Solo en dos ocasiones fue 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 concluida, 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 más cercanos, los microprocesadores.

Como consecuencia de lo anterior, existen aplicaciones para las que específicamente 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 más importantes de este dispositivo son en instrumentación, interfaces y procesamiento 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 la 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, fue como surgió 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 características 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 múltiples grabaciones de programas en EPROMS, siendo las correcciones y depuración de los mismos una tarea tediosa, amén 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 ágilmente 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, así 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 anfitrión son típicamente 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 de las características que se le pueden adaptar, serían 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 podría ser una microcomputadora personal.

3.- Generar más rutinas en forma de utilerías, para que de esta manera el sistema fuese más general y en caso

de su comercialización, lograr una aceptación adecuada por parte del usuario.

El SIS ZB requiere aún de ciertas mejoras, sin embargo posee lo esencial 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 esencial 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 ZB 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 ZB lo ha confirmado. Quisieramos, 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 DESIGN,
William 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 DESIGN,
John B. Peatman,
Mc. Graw Hill.

INTRODUCTION TO MICROPROCESSOR SYSTEM DESIGN,
Harry Garland,
Mc. Graw Hill.

LOGIC HANDBOOK,
Volumen II,
National.

TTL HANDBOOK,
Texas Instruments,

ZILOS COMPONENTS DATABOOK,
ZILOS 1980.

ZILOS Z8 HANDBOOK,
ZILOS 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

MINIARY

NEW DEVICE

(IE-10)

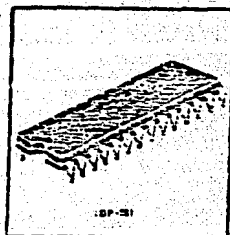
July, 1982



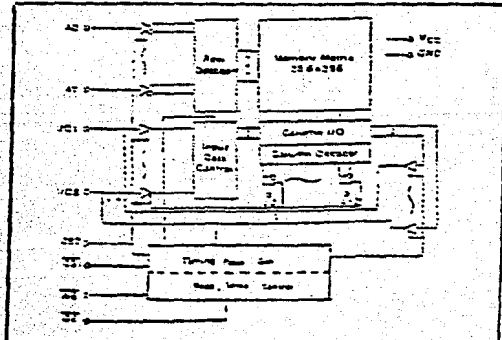
H1-CMOS 64K STATIC RAM
HM5264P-10/-12/-15, HM5264LP-10/-12/-15

8192 x 8 BIT HIGH SPEED STATIC H1-CMOS RAM

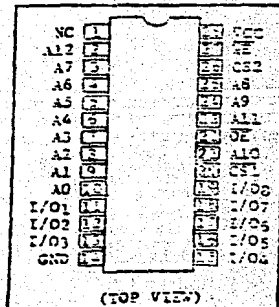
- Single 5V. Supply.
- High speed: Fast Access Time 100ns/120ns/150ns max.
- Equal Access and Cycle Time.
- Completely Static RAM: No Clock or Timing Strobe Required.
- Low Power Standby and Low Power Operation;
Standby: 10µW typ.
Two Chip Selection for Battery Back up.
- Operation: 200mW typ.
- Capability of Battery Back up Operation (LP)
- Common Data Input and Output, Three State Outputs.
- Directly TTL Compatible: All Inputs and Outputs.
- Standard 28-pin Package Configuration.
- Pin Out Compatible with 64K EPROM HM482764.



Functional Block Diagram



Pin Arrangement



• Truth Table

Mode	WE	CS1	CS2	OE	I/O	Current	Note
Not Selected (Power Down)	H	H	L	H	High-Z	ISB, ISB1	
Output Disabled	H	L	H	H	High-Z	ISB, ISB2	
Read	H	L	H	L	Data	ICC, ICC1	
Write	L	L	H	L	Data	ICC, ICC1	Write Cycle(1)
	L	L	H	L	Data	ICC, ICC1	Write Cycle(2)

• Don't Care (H or L)

Note: The specifications of this device are subject to change without notice.
Please contact your nearest Hitachi's Sales Dept. regarding specifications

• Absolute Maximum Ratings

Parameter	Symbol	Rating	Unit
Voltage on Any Pin Relative to GND	V _{IN}	-0.5 to 7	V
Operating Temperature	T _{OP}	0 to 70	°C
Storage Temperature	T _{STG}	-55 to 125	°C
Temperature under Bias	T _{Bias}	-10 to 85	°C
Power Dissipation	P _T	1.0	W

• -3.0V (pulse width 50ns)

• Recommended DC Operating Conditions
(T_a=0°C to 70°C)

Parameter	Symbol	min	typ	max	Unit	Note
Supply Voltage	V _{CC}	4.5	5.0	5.5	V	
	GND	0	0	0	V	
Input High (logic 1) Voltage	V _{IH}	2.2		6.0	V	
Input Low (logic 0) Voltage	V _{IL}	-0.3		0.8	V	• -3.0V (Pulse width 50ns)

• DC and Operating Characteristics
(T_a=0°C to 70°C, V_{CC}=5V±10%, GND=0V)

Parameter	Symbol	HM6264P(LP) -10/12/13			Unit	Test Conditions	Notes
		min	typ	max			
Input Leakage Current	I _{LI}			2	μA	V _{IN} =GND to V _{CC}	
Output Leakage Current	I _{LO}			2	μA	CS1=V _{IH} or CS2=V _{IL} or OE=V _{IH} V _{I/O} =GND to V _{CC}	
Operating Power Supply Current: DC	I _{CC}		40	80	μA	CS1=V _{IL} CS2=V _{IH} I _{I/O} =0mA	
Average Operating Current	I _{CC1}		60	110	μA	Minimum Cycle duty=100%	
Standby Power Supply Current (1)	I _{SB}		1	3	μA	CS1=V _{IH} or CS2=V _{IL} I _{I/O} =0mA	
Standby Power Supply Current(2): DC	I _{SB1}	P	0.0	2	μA	CS1 = V _{CC} -0.2V, CS2 = V _{CC} -0.2V or CS2 = 0.2V	[2]
		LF	2	100	μA		
Standby Power Supply Current(3): DC	I _{SB2}	P	0.0	2	μA	CS2 = 0.2V	[2]
		LF	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 V_{CC}=5.0V, T_a=-25°C and specified loading.
2) V_{ILmin} = -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)

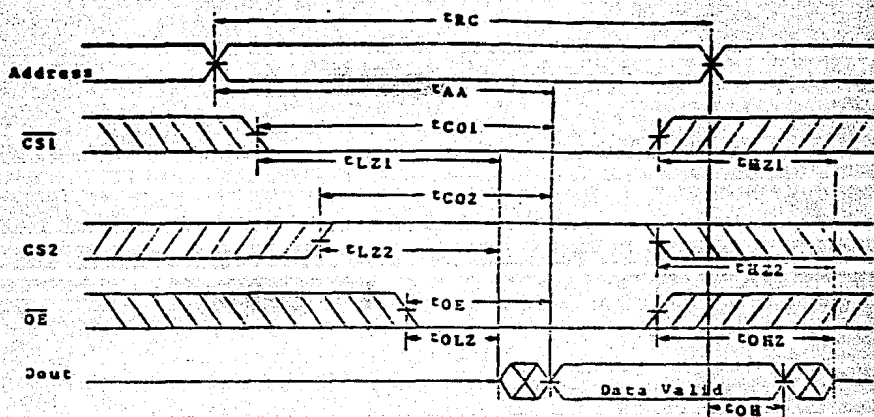
• Read Cycle

Parameter	Symbol	HM6264 P(LP)-10		HM6264 P(LP)-12		HM6264 P(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 (CS1) to Output	t _{CO1}		100		120		150	ns
Chip Selection (CS2) to Output	t _{CO2}		100		120		150	ns
Output Enable(OE) to Output Valid	t _{OE}		50		60		70	ns
Chip Selection (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(OE) to Output in Low Z	t _{OLZ}	5		5		5		ns
Chip Deselection (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(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_{HZ}max is less than t_{LZ}min 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)

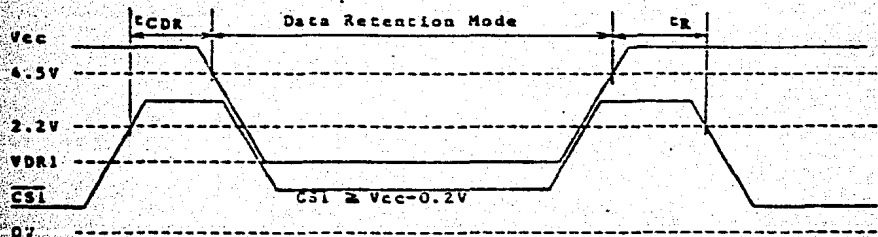
($T_a = 0^\circ\text{C}$ to 70°C)

[1] $V_{ILmin} = -0.1\text{V}$

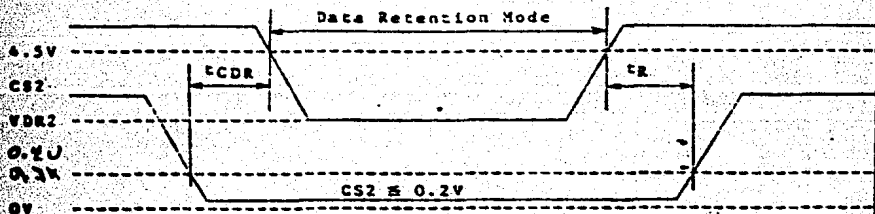
[2] $t_{RC} = \text{Read Cycle Time}$

Parameter	Symbol	Test Conditions	min	typ	max	unit
Vcc for Data Retention (1)	V_{DR1}	$CS1 \geq V_{cc} - 0.2\text{V}$ $CS2 \geq V_{cc} - 0.2\text{V}$	2.0			V
Vcc for Data Retention (2)	V_{DR2}	$CS2 \leq 0.2\text{V}$	2.0			V
Data Retention Current (1)	I_{CCDR1}	$V_{cc} = 3.0\text{V}$ $CS1 \geq V_{cc} - 0.2\text{V}$ $CS2 \geq V_{cc} - 0.2\text{V}$ or $CS1 \leq 0.2\text{V}$		1	[1] 50	μA
Data Retention Current (2)	I_{CCDR2}	$V_{cc} = 3.0\text{V}$ $CS2 \leq 0.2\text{V}$		1	[1] 50	μA
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 (CS1 Controlled)



Low Vcc Data Retention Waveform No.2 (CS2 Controlled)



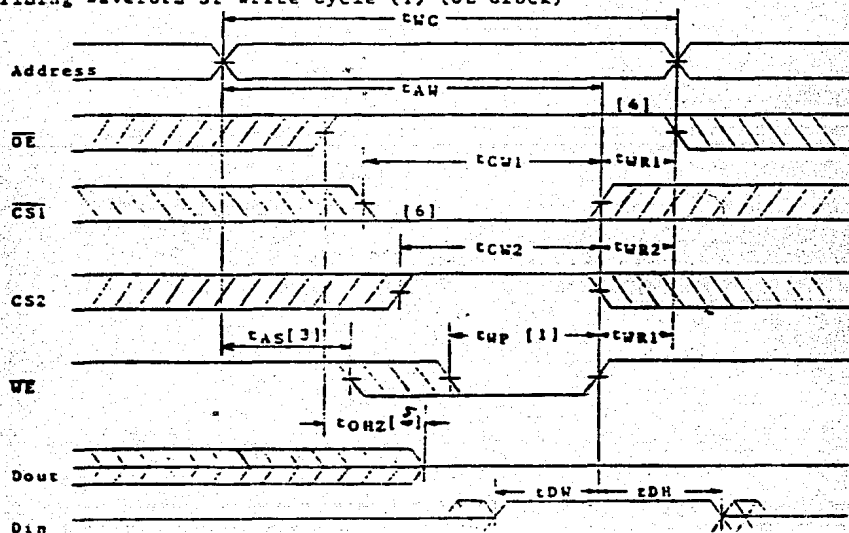
Note: 1) $CS2$ controls Address buffer, $\overline{WE}$ buffer, $\overline{CS1}$ buffer and $\overline{Din}$ buffer. If $CS2$ controls data retention mode, $\overline{Vin}$ levels (Address, $\overline{WE}$, $\overline{CS1}$, I/O) can be in the high impedance state. If $\overline{CS1}$ controls data retention mode, $CS2$ must be $CS2 \geq V_{cc} - 0.2\text{V}$. The other inputs (address, $\overline{WE}$, I/O) can be in the high impedance state.

or $CS1 \leq 0.2\text{V}$

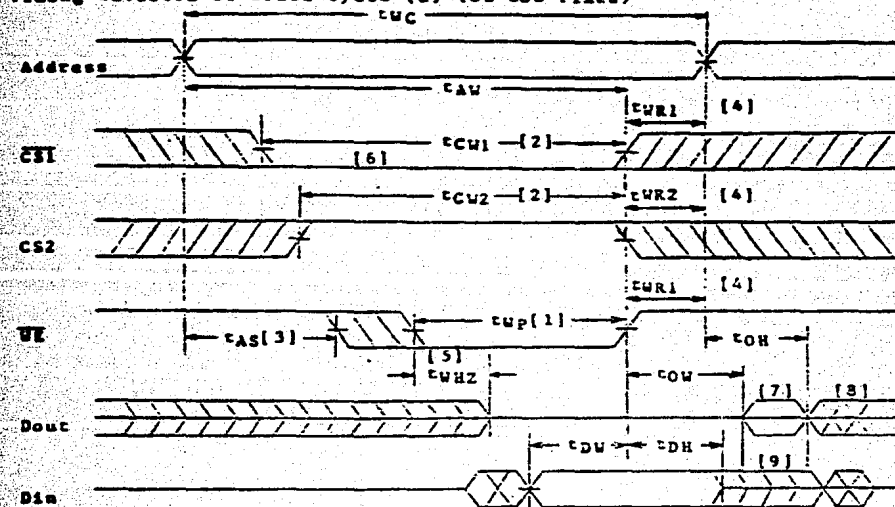
• Write Cycle

Parameter	Symbol	HM6264 P(LP)-10		HM6264 P(LP)-12		HM6264 P(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 $\overline{CS2}$	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)



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. t_{WP} is measured from the beginning of write to the end of write.
- 2) t_{CW} is measured from the later of $\overline{CS1}$ going low or $CS2$ going high to the end of write.
- 3) t_{AS} is measured from the address valid to the beginning of write.
- 4) t_{WR} is measured from the end of write to the address change. t_{WR1} applies in case a write ends at $\overline{CS1}$ or $\overline{WE}$ going high. t_{WR2} 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 instrucción. Este conjunto de instrucciones son fundamentales para poder establecer una acción bien definida que ejecutará el microprocesador.

Cada instrucción de este conjunto, es un simple patrón binario, el cual debe de ser introducido como información hacia el microprocesador para que sea interpretada como una instrucción.

QUE ES UN PROGRAMA?

Un programa escrito en cualquier lenguaje de alto nivel ó en lenguaje ensamblador, es una serie de instrucciones que causan que una computadora desarrolle una tarea particular. En forma más precisa, se puede afirmar que un programa de computadora incluye algo más que instrucciones, también contiene los datos y las direcciones de memoria que el microprocesador (de la computadora) necesita para complementar y ejecutar adecuadamente cada instrucción del programa.

Después de tener el programa fuente, éste es traducido a un conjunto de números binarios, este conjunto es lo que se conoce como PROGRAMA OBJETO ó LENGUAJE DE MÁQUINA. Toda vez que el programa objeto está creado, está listo para ser ejecutado.

Para que una computadora pueda ejecutar un programa, éste debe de estar en lenguaje de máquina, 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 máquina.

Si no se tuvieran este tipo de programas, para una aplicación dada se tendría que programar en binario, lo que ocasionaría que los programas fueran difíciles 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 algún lenguaje de alto nivel, es utilizando instrucciones, a las cuales se les asigna un nombre específico. Este nombre que se le da a cada código de instrucción es llamado un "NEMONICO". Este nemónico deberá describir de alguna manera que función realiza la instrucción.

Cada microprocesador tiene su propio conjunto de nemónicos para el conjunto de sus instrucciones.

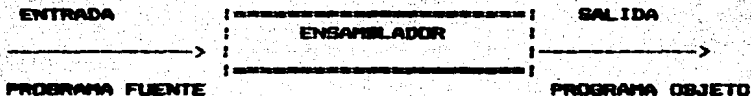
EL LENGUAJE ENSAMBLADOR

Una vez que se ha decidido programar utilizando un lenguaje de nemónicos, para que el programa pueda ser ejecutado deberá de ser traducido a lenguaje de máquina. Cada fabricante de microprocesadores aparte de proporcionar el conjunto de nemónicos asociados a sus instrucciones, proporciona su código binario equivalente, por lo que un programa escrito en nemónicos puede ser traducido a lenguaje de máquina a manualmente.

Si el programa traducido es pequeño, la generación del código objeto puede no ocasionar problemas, pero si se trata de un programa de un número considerable de instrucciones, su traducción manual puede ocasionar pequeños errores que tendrán como resultado una errónea ejecución del mismo.

Para evitar este problema, en lugar de realizar la traducción a lenguaje de máquina manualmente, se utiliza un programa llamado **ENSAMBLADOR**. El programa ensamblador traduce un programa de usuario o programa fuente escrito en nemónicos, a un programa escrito en lenguaje de máquina.

Esquemáticamente la función 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 sintaxis específica, un apropiado control de la información e inclusive el uso en el lugar apropiado, de nombres y números.

CARACTERISTICAS DE LOS ENSAMBLADORES

Los ensambladores en realidad pueden efectuar más funciones, que el solo hecho de traducir nombres de mónicos y registros en sus equivalentes binarios. Algunas de las características adicionales de los ensambladores son :

- 1.- Permitir al usuario asignar nombres a localidades de memoria, a dispositivos de entrada y salida y aún, a una secuencia de instrucciones (definiendo MACROS).
- 2.- Convertir datos o direcciones de varios sistemas numéricos (por ejemplo de hexadecimal ó decimal) a binario y convertir caracteres de código ASCII ó EBCDIC a su código binario equivalente.
- 3.- Informar al programa cargador (loader program), que partes de la memoria de programa ó de datos deberá ser procesado.
- 4.- Permitir al usuario asignar áreas de memoria para almacenamiento temporal de datos y colocar datos fijos en áreas de memoria de programa.
- 5.- Proporcionar la información necesaria para poder incluir otros programas externos, dentro del programa que se esté utilizando.

Es obvio que todas estas características se reflejan en un mayor espacio en memoria que ocupará el ensamblador para su ejecución.

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 está 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 información puede ser irrelevante para la función 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 pueda ser ejecutado en otra máquina que tenga un microprocesador diferente.

ENSAMBLADORES

Esta parte del capítulo describe las funciones desarrolladas por los ensambladores, comenzando por las características comunes de varios tipos de éstos, hasta capacidades más elaboradas, tales como MACROS y condiciones de ensamblado.

CARACTERISTICAS DE LOS ENSAMBLADORES

Como se mencionó anteriormente, hoy en día las funciones que pueden desarrollar los ensambladores, van más allá que el simple hecho de traducir nemónicos en lenguaje ensamblador a códigos binarios.

INSTRUCCIONES DEL ENSAMBLADOR

Las instrucciones o proposiciones del lenguaje ensamblador, están divididas en un número de CAMPOS, como se muestra a continuación :

CAMPO DE ETIQUETA	CODIGO DE OPERACION O CAMPO DE NEMONICO	OPERANDO O CAMPO DE LA DIRECCION	CAMPO DE COMENTARIO
ROOT:	CALL CALL JR	AGUANTA RECEPCION ROOT	; LLAMA A LA RUTINA ; ; SALTA AL CONTROL ; DEL PROGRAMA
RECEPCION:	JP	RRR4	; VA A LA RUTINA ; DIRECCIONADA POR ; RRR4

El campo de código de operación, es el único campo que nunca puede estar vacío, ya que siempre contiene un neónico 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 o puede estar en blanco.

Los campos de comentario y etiqueta son opcionales. Un programador asignará una etiqueta a una instrucción o agregará un comentario si considera que es necesario, este último principalmente. Esta decisión 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 neónicos en un momento dado, puede causar problemas de comprensión.

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 sintaxis 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 específicamente 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 ó 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) ó 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 3RR2, 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 fácil 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 definición 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 razón, es sencillo modificar un programa mediante la inserción de otros programas ó simplemente mediante la reorganización de la memoria.
- 4.- Mediante el uso de etiquetas es fácil utilizar programas separados y formar una biblioteca de programas. Por ejemplo, cada uno de estos programas se puede anexas para formar un programa totalmente diferente.

Una vez establecidas las razones de porqué 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 número de caracteres utilizados en el nombre de etiquetas, el caracter inicial y los caracteres permitidos en dicho nombre. Una vez cumplidas estas especificaciones la decisión de que etiquetas se deben de utilizar es del programador.

El nombre de etiqueta seleccionado, debe ser tal que sugiera el propósito de la misma. Por ejemplo, el nombre PRUEBAMEMO, debe de corresponder a una rutina en la cual 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 \$T000. Este tipo de etiquetas llevan una secuencia que permite su fácil detección, aunque no ayudan a la documentación del programa.

Existen algunas reglas para la selección 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 mnemónico o a algún código de instrucción, ya que puede crear confusión aún al propio ensamblador.
- 2.- No utilizar etiquetas cuyo nombre exceda a la longitud permitida por el ensamblador, pues en la mayoría de las veces, son truncadas al número permitido de caracteres, pudiendo perder su significado.
- 3.- Evitar escoger caracteres especiales (no alfabéticos o no numéricos), ó letras minúsculas, ya que algunos ensambladores no lo permiten. La regla más sencilla a seguir a éste respecto, es seleccionar solo letras mayúsculas y números.
- 4.- Comenzar el nombre de cada etiqueta con una letra. Este tipo de etiquetas son siempre aceptadas.
- 5.- No utilizar dos o más nombres de etiquetas que puedan confundirse. Evitar las letras I, O, Z y los números 0, 1, 2. También evitar nombres como XXXX y XXXXX ya que carecen de todo sentido.
- 6.- Cuando no se está 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 0 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 números son decimales a menos que se especifique algún otro tipo.

2) Otros sistemas numéricos :

Algunos ensambladores también aceptan números binarios, octales o hexadecimales. Estos tipos de números se deben de identificar de alguna manera, por ejemplo antes o después del número se pone un carácter especial. Algunos de estos identificadores que se utilizan son :

B ó	para números binarios.
O, O ó C	para números octales.
H ó h	para números hexadecimales.
D	para números decimales.

El carácter D es opcional siendo el tipo por omisión.

Los ensambladores requieren que un número hexadecimal comience con un dígito decimal (que en la mayoría de veces es el cero), para distinguir 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 \$ 0 \$).

Es útil principalmente en instrucciones de salto(JMP), por ejemplo

JP 46

causa un salto a la localidad de memoria 4 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 símbolos tales como A ó C. Pocos ensambladores también permiten cadenas EBCDIC.

6) Combinación de 1 a 5 con operadores lógicos o aritméticos:

Casi todos los ensambladores permiten simples operaciones aritméticas como INICIO+2. Algunos ensambladores también permiten multiplicación, división, funciones lógicas, recorrimientos etc.

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 listas 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 parámetros 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 o para compensar alguna deficiencia en el conjunto de instrucciones del microprocesador que se está utilizando.

Para evitar escribir repetidamente la misma secuencia de instrucciones se puede utilizar lo que se conoce como un **MACRO**.

Los **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

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 P2
instrucción P3

MACRO1

instrucción P4
instrucción P5
instrucción P6
instrucción P7

MACRO1

instrucción P8
instrucción P9
:
:

PROGRAMA OBJETO

instrucción P1
instrucción P2
instrucción P3

instrucción 1
instrucción 2
instrucción 3
instrucción P4
instrucción P5
instrucción P6
instrucción P7

instrucción 1
instrucción 2
instrucción 3

instrucción P8
instrucción P9
:
:

Las macros no son lo mismo que las subrutinas. Una subrutina aparece una vez en el programa, y la ejecución del programa transiada a la subrutina cada vez que se llama. Una macro es EXPANDIDA a la actual secuencia de instrucciones cada vez que la macro ocurre, esto 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 más corto.
- 2) Ofrecen una mejor documentación de un programa.
- 3) Los programas que utilizan macros, son más fáciles de corregir pues si un error ocurre en la macro, se puede librar de errores gran cantidad del programa.
- 4) Los programas son fáciles de modificar tan solo modificando la definición 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) Repetición de las mismas instrucciones en lenguaje de máquina, 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 estandarización en su uso puede hacer al programa difícil 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 código 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 difícil 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 instrucción. Los comentarios deben de establecer cosas como "PROGRAMANDO PUERTOS", "EXAMINANDO BANDERAS", "ALMACENANDO REGISTROS". Los comentarios no deben decir cosas como: "SALTA 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 documentación.
- 3.- Comentar TODOS los puntos clave.
- 4.- No comentar instrucciones estandar 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 áreas 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 fácil 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 varían enormemente. No trataremos de describir todos los tipos de ensambladores existentes, solamente se definirán los términos e indicaremos algunas características de varios tipos de ensambladores.

CROSS-ASSEMBLER

Un cross-assembler es un ensamblador que "corre" en una computadora ajena a la que ensambla los programas.

La computadora en la cual el cross-assembler se ejecuta, es típicamente 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 cual el cross-assembler ensambla los programas es típicamente una microcomputadora como la Z80. Varios cross-assemblers 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 cual ensambla los programas.

El autoensamblador requiere espacio de memoria y periféricos, y puede correr más 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 específico 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 después 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 símbolos, en la segunda pasada el ensamblador reemplaza las referencias con las definiciones actuales. Un ensamblador de dos pasadas resuelve fácilmente 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 consisten 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 inválidas (por ejemplo 2 operandos en una sola línea).
- 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 21030

Procesador de Textos WORD PERFECT 4.0.

Computadora PRINTFORM 5203.

Para el diseño y depuración del SIS 28, 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.

Multímetro Digital marca FLUKE 74.

Ensamblador Cruzado de 28 para sistemas Z80-8080.

Paquete de Depuración DEBUG para CDS (CROMEMCO).

APENDICE D

HOJAS DE ESPECIFICACIONES DEL 28

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 refer-
enced to GND. Positive current flows into

the reference pin. Standard conditions are
as follows: $+4.75V \leq V_{CC} \leq +5.25V$,
GND = 0V, $0^\circ C \leq T_A \leq +70^\circ C$.

3 DC Char- acteristics

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	$0 \leq V_{IN} \leq +5.25V$	
I_{OL}	Output Leakage			μA	$0 \leq V_{IN} \leq +5.25V$	
I_R	Reset Input Current			μA	$V_{RL} = 0V$, $V_{CC} = +5.25V$	
I_{DD}	V_{DD} Supply Current			mA		
I_{MM}	V_{MM} Supply Current			mA		

1. For A_0, A_{11} , RDS, STX, SCLK and IACK on the 28 pin version, $I_{OH} = -100 \mu A$ and $I_{OL} = 1.0 mA$.

Data Sheets on pages 8-02 through 8-08 are reprinted by permission of Zilog, Incorporated.

4 External Instruction Fetch, I/O or Memory Read Timing

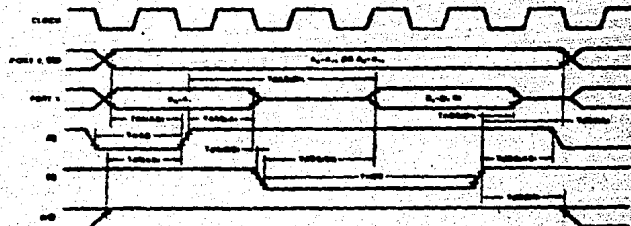
Symbol	Parameter	Min	Max	Units	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdASA	Address Strobe to Address Float Delay Time	60		ns	Test Load 1	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		380	ns	Test Load 1	3
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdADS	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		180	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 mount 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 1-9 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 three clock periods must be added to TdAS(DI) and the change in three clock periods added to TdDS(DI).

4. All timing relationships assume 2.0V for a logic "1" and 0.5V for a logic "0."



I/O External I/O or Memory Write Timing

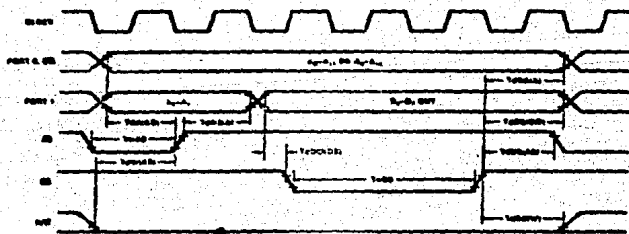
Symbol	Parameter	Min	Max	Unit	Condition	Note
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdAS(A)	Address Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdO(AS)	Data Out Valid to Data Strobe Delay Time	30		ns	Test Load 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
TdDS(AS)	Data Strobe to Address Strobe Delay Time	50		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 clock period 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. All timing references assume 2.0V for a logic "1" and 0.5V for a logic "0."



5 **Memory Port** **(28/64)** **Timing**

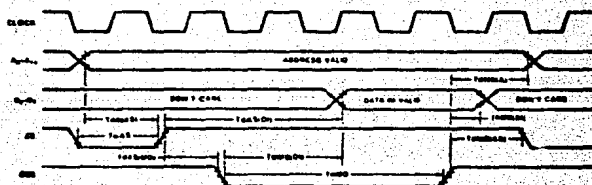
Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 2	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		280	ns	Test Load 2	3
TwAS	Address Strobe Width	60		ns	Test Load 2	1
TdAS(MD)	Address Strobe to Memory Data Strobe Delay Time	60		ns	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 period 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 structures being reserved. 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 at 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

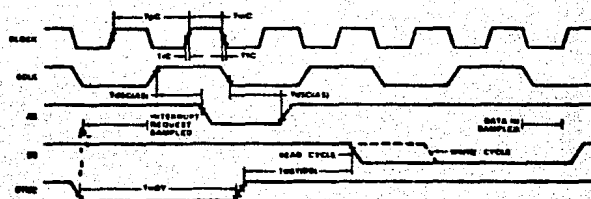
Symbol	Parameter	Min	Max	Unit	Condition	Notes
T_{pC}	Input Clock Period	125		ns		
T_{rC}, T_{fC}	Input Clock Rise and Fall Times			ns	From External Clock Generator	
T_{wC}	Input Clock Width			ns	From External Clock Generator	
$T_{dSC}(AS)$	System Clock Out to Address Strobe Delay Time			ns		1
$T_{dSY}(DS)$	Instruction Sync Out to Data Strobe Delay Time			ns		1, 2
T_{wSY}	Instruction Sync Out Width			ns		1, 2

1. Test Conditions use Test Load 1 for $SCLK$ and $STNC$ -data output through their respective Port 1 pins and Test Load 2 on the $SCLK$ and $STNC$ direct outputs on the 64 pin version.

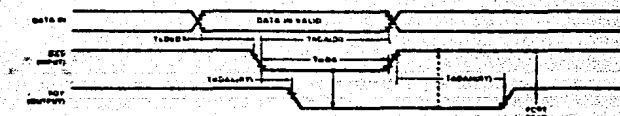
2. Times given assume an 8 MHz crystal input frequency.

For lower frequencies, the change in two clock periods must be added.

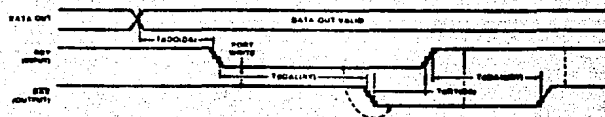
3. All timing references assume 2.0V for a logic "1" and 0.5V for a logic "0."



Symbol	Parameter	Min	Max	Unit	Condition
8 Handshake Timing					
TsD(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	0		ns	Input Handshake Test Load 1 Output Handshake Test Load 1
TdDAH(RY)	Data Available High to Ready Delay Time	0		ns	Input Handshake Test Load 1 Output Handshake Test Load 1
TdDOxDA)	Data Out to Data Available Delay Time			ns	Test Load 1
TdRY(DA)	Ready to Data Available Delay Time			ns	Test Load 1



Input Handshake



Output Handshake

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

