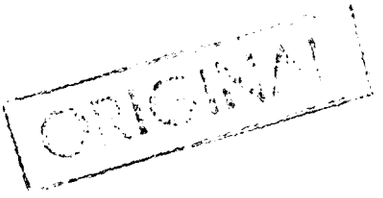
ESCUELA UNIVERSITARIA POLITECNICA

DE

LAS PALMAS DE GRAN CANARIA.



TITULO: DESARROLLO DE SISTEMAS BASADOS EN MICROPROCESADORES (SOFTWARE), CON UN MDS-221.

AUTOR: Francisco Javier Afonso Palmero

TUTOR: Sebastián Suarez Gil

INDICE:

INDICE (CONTINUACION):

1.- ANTECEDENTES.

En los pasados 25-30 años, la industria de los semiconductores ha tenido un avance espectacular.La aparición del circuito integrado (CI), permitió la miniaturización de los equipos electrónicos, acompañados de una fiabilidad de funcionamiento mayor.

A medida que las técnicas de integración fueron evolucionando, se aumentó el número de componentes por circuito,pasando de unas pocas decenas a varias decenas de mil.

Pero aún así, la industria se enfrentaba con un problema de altos costos de la lógica aleatoria y cableada.Para tratar de evitar este problema,una empresa japonesa encargó,un circuito integrado de multiples usos.Y los ingenieros de una empresa norteamericana, la INTEL corporation, comenzaron su diseño.

En 1971, acabaron los estudios con la presentación del microprocesador 4004, un procesador programable de 4 bits en un solo circuito integrado.Era el comienzo.Luego vendrian una serie de microprocesadores,hasta que llegamos a los actuales 8085 de INTEL, Z80 de Zylog, 6800 de Motorola, etc.,que poseen unas altas prestaciones, bajo coste, además de un alto MTBF(Tiempo medio antes de un fallo).

Los procesadores programables se introducen en cualquier tipo de industria.Así desde un juguete hasta un electrodoméstico suelen traer incopporado un sistema basado en microprocesador.

Este proceso de miniaturización y el consiguiente abataramiento ha permitido, que costosos equipos que eran anteriormente de laboratorio,sean ahora utilizados popularment,como es el caso de las calculadoras.

2.- OBJETO DEL PROYECTO.

El objeto de este proyecto és,la realización de un algoritmo de trabajo, donde se expliquen las pautas a seguir por un ingeniero, a la hora de afrontar el diseño de un sistema basado en un microprocesador.Luego se hará una aplicación al sistema de desarrollo MDS-221,de la casa INTEL, describiendo al mismo.

3.- EL SISTEMA DE DESARROLLO EN LA SOLUCION DE EL PROBLEMA DE DISEÑO.

Una vez planteado el problema a resolver y las especificaciones que debe cumplir la solución, comenzamos las tareas de diseño.Tareas que,actualmente se han visto facilitadas gracias a la aparición de los sistemas de desarrollo.

Un sistema de desarrollo,es una herramienta electronica que se usa para la realización del Software de el prototipo y la comprobación del harware.

Surgió debido a la posición intermedia de los microprocesadores, entre el mundo de la lógica cableada y los ordenadores clásicos, ya que lo primero se podía comprobar con polimetros y osciloscopios, pero hacia falta algo más que esto,para poder comprobar el correcto funcionamiento, de un programa destinado a controlar un sistema basado en un microprocesador.Y apareció el sistema de desarrollo.

3.1.- EVOLUCION DE LOS METODOS DE DESARROLLO.

En los últimos 10 años, los métodos de desarrollo de

microprocesadores, han cambiado.Cuando fueron introducidos en 1971, las facilidades de desarrollo no existian y los diseñadores debian trasladar su programa a código maquina y lo introducian en el uP.

El primer paso en la evolución de las herremientas de desarrollo, fué la aparición del cross assembler, un programa que funciona en un sistema distintò y ensambla códigos máquina para el microcomputador, uC.

A medida que la cantidad de memoria en el sistema del uC aumentó, empezaron a estar disponibles los ensambladores residentes.Pero todavia no se utilizaban las ventajas de un editor de textos y el manejo de ficheros.

Y fué hacia mediados de la década, que aparecieron los sistemas de desarrollo de una forma más ó menos completa, incluyendo: Editor de textos, un Ensamblador, y un "Debugger".

Los últimos sistemas se componen de (En general): Una unidad de display ó presentación en pantalla basada en un CRT(Tubo de rayos catódicos), un teclado, una unidad a diskette ó floppy-disk, un procesador central con capacidad para direccionar desde 16 a 64 kbytes de memoria, una impresora, un módulo ICE (Unidad para emular un circuito)y un programador de PROM.Estas són precisamente las partes de las que se compone el sistema de desarrollo MDS-221,de la casa INTEL corporation.

4.- DESCRIPCION DEL SISTEMA DE DESARROLLO MDS-221.

4.1.- INTRODUCCION.

El sistema de desarrollo MDS-221, es un sistema personalizado para el par de uP 8080, 8085,en la parte de software y particularmente para el 8085 ,en la parte de hard

-ware.Para ello trae incorporado,un módulo ICE-85, emulador de los microprocesadores 8085.

El MDS-221 está formado por trés diferentes elementos de computación: El IPB ó IPC, la IOC, y la PIO.

El IPB (Integrated processor board), es una tarjeta que se encarga de el control del sistema MDS-221,cuando se hace referencia a datos basados en el lenguaje del 8080.

El IPC (Integrated processor card), es una tarjeta que se encarga de el control del sistema MDS-221, cuando se hace referencia a datos basados en el lenguaje del 8085.

La IOC (Input/Output controller), está compuesta por una serie de circuitos, que sirven de interface inteligente entre el IPB ó el IPC y la unidad de CRT ó la unidad de discos flexibles.

La PIO (Parallel Input-output), esta compuesta por una serie de circuitos, que sirven de interface inteligente entre la IPB ó la IPC, y los periféricos de desarrollo estandar del sistema MDS-221.

Cada uno de estos sistemas poseen su propio microprocesador, y la suficiente memoria y facilidades de I/O, para realizar las tareas de computación en tiempo real.Pero no pueden actuar por separado, por lo que operan unidos formando un sistema simple integrado.

En la figura 4.1 se ve un esquema del sistema MDS-221 en configuración de bloques.Analogamente, las tablas 4.1 y 4.2 muestran las caracteristicas y especificaciones de el sistema MDS-221.

4.2.- SOPORTE HARDWARE.

El soporte hardware, está compuesto por: La consola,el teclado, la unidad de discos flexibles, la impresora, el

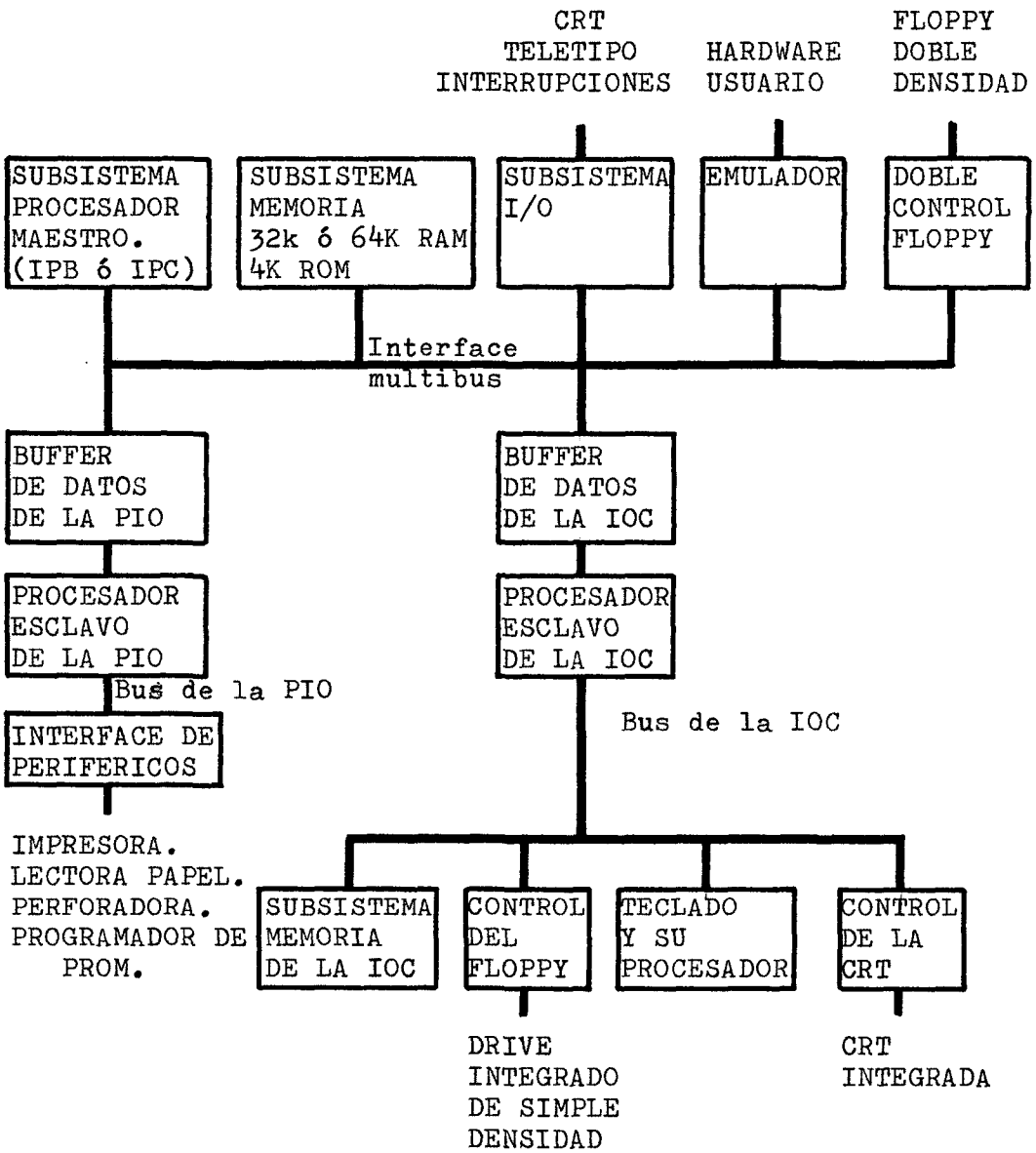


Figura 4.1

Tabla 4.1 (Especificaciones del sistema MDS).

PROCESADOR MAESTRO IPC.	
-Microprocesador:	8085A - 2, operando a 4 Mhz.
-RAM:	64K.
-ROM:	4K.
-Bús:	Arquitectura multibús.
INTERRUPCIONES.	
-Tipos:	Hay 8 niveles de interrupción.
INTERFACES DE I/O.	
-Serie:	Dós canales RS232 a 110-9600 baudios, en modo asíncrono ó 150--5600 baudios en modo síncrono.
-Paralelo:	Interface para la impresora y el grabador de PROM.
ACCESO DIRECTO A MEMORIA(DMA):	Es una capacidad estandar,en la arquitectura de Multibús.
SUBSISTEMA DE DISKETTE.	
-Número de drives:	Uno, de simple densidad.
-Capacidad:	250K bytes.
-Vel. transferencia:	250 bits/segundo.
-Tiempo de acceso:	Pista a pista: 10 ms. Colocación de la cabeza: 10 ms.
-Posicionamiento aleatorio medio:	260 ms.
-Velocidad ángular:	360 rpm.
-Modo de grabación:	FM.
REQUERIMIENTOS DE AC.	
-Voltage de entrada:	220V/240V
-Corriente de entrada:	3.1A

Tabla 4.2 (Capacidades de corriente del sistema MDS)

COMPONENTE DEL SISTEMA	Voltages de alimentación					
	5V	12V	-12V	-10V	15V	24 V
*Capacidad de la fuente----------	30.0A	2.5A	0.3A	1.0A	1.5A	1.7A
*IPB----------	4.0A	0.3A	0.1A	0.01A	--	--
*IOC----------	2.8A	0.1A	--	0.01A	--	--
*IPC----------	4.3A	1.4A	0.2A	0.02A	--	--
*CRT----------	--	--	--	--	1.5A	--
*Teclado----------	0.4A	--	--	--	--	--
*Drive del disco---	1.0A	--	--	--	--	1.7A
*Drenaje total de corriente------	8.5A	1.5A	0.2A	0.03A	1.5A	1.7A

Nota: El drenaje total de corriente está indicado para nuestro sistema.

módulo ICE-85, y el grabador de PROM.

4.2.1.- LA CONSOLA.

La consola del sistema MDS-221, está formada por un tubo de rayos catódicos (CRT), y la circuitería analógica necesaria para la generación y control del haz de electrones.Esta circuiteria, es capaz de generar más de 260 barridos horizontales por cada barrido vertical.

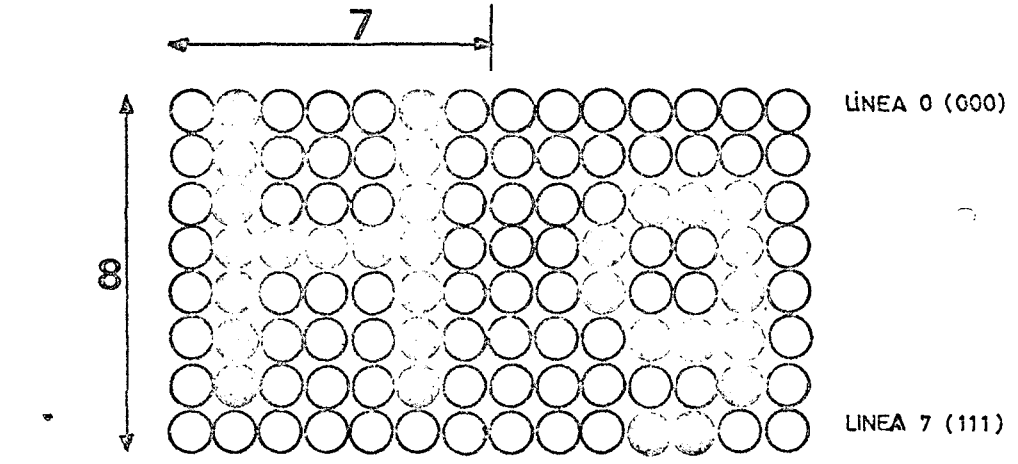
El tipo de pantalla es de alta resolución, lo que ayuda a la buena definición,de los caracteres escritos en ella.

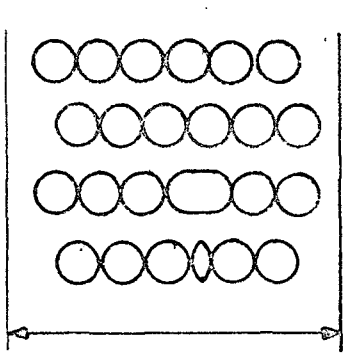
La representación de un caracter em pantalla, se basa en la generación de una matriz de puntos (De 7X8) para las letras mayúsculas, y una un poco más pequeña (De 5X7) para las letras minúsculas.La figura 4.2 muestra este tipo de matriz.

Dentro de la matriz de puntos, se verifica que 6 bits se corresponden con el caracter, y los otros 2 bits tienen un significado especial, que se emplea para el desplazamiento de medio punto en la matriz y conseguir una mejor lectura del caracter.

Este tipo de consola, puede representar hasta un total de 25 lineas de 80 caracteres cada una.Además, la información representada en una pagina de pantalla, no es recuperable,es decir, a medida que se avanza en la ejecución de comandos, la información que aparece,una vez llena la página va desapareciendo, salvo en el caso en que nos encontremos en la edición de un texto, mediante el programa Editor CREDIT,en el que la información va pasando a la memoria del sistema, siendo recuperable en cualquier momento de la edición.

MATRÍZ DE CARACTERES.





CAMPO DE CARACTERES

MATRÍZ DE CARACTERES,

CON PUNTOS DESPLAZADOS.

FÍGURA 4.2

4.2.2.- EL TECLADO.

El teclado del sistema de desarrollo MDS-221, es del tipo convencional de una máquina de escribir, salvo algunas teclas especiales, descritas más tarde.

El teclado está organizado en una serie de filas y columnas, que cubren las 62 teclas individuales mediante una matriz de 8X8 (Quedando, por tanto, dos espacios libres para otras teclas).

Un circuito, el 8041, se encarga de las tareas de detección de las teclas pulsadas.De esta manera, al pulsar una tecla, el 8041 que mientras tanto ha estado generando una serie de habilitadores (Strobes) para las filas,detecta cuando ha sido recibido un pulso de una columna, determinando que tecla ha sido pulsada.En el caso de que se produciera el tecleado simultaneo de dos teclas, el 8041, las almacena sucesivamente en un stack, para proceder a su transmisión secuencial a la unidad de entrada-salida,IOC. Si una de estas dós teclas fuera alguna de las siguientes: CNTL, SHIFT, ó RPT, el 8041 tratará al caracter pulsado conjuntamente con una de ellas, de acuerdo con la función especial para la que se prevee en este caso, y generará el código necesario para su transmisión a la IOC.La figura 4.3, muestra un esquema del teclado.

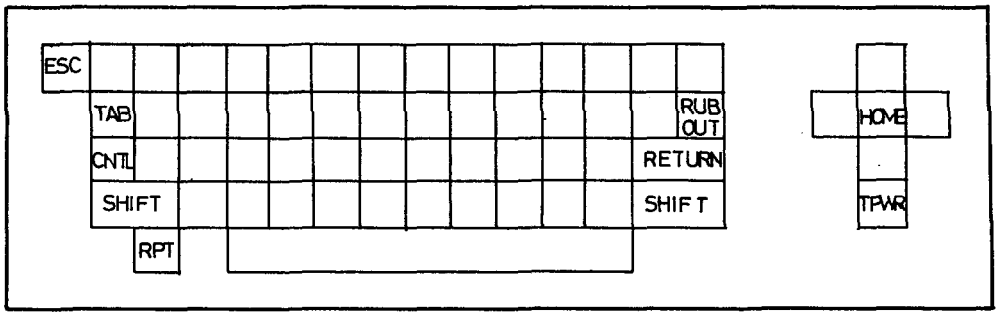


Figura 4.3

4.2.3.- LA UNIDAD DE DISCOS FLEXIBLES.

La unidad de discos flexibles,es un sistema de memoria, que sirve de soporte para almacenar datos permanente.Se encuentra integrada en la caja del MDS-221 y sus caracte rísticas principales se muestran en la tabla 4.1.

Los discos flexibles que se utilizan, son de 8 pulgadas de diámetro y están divididos en 77 pistas de 26 sectores cada una, donde un sector són 128 bytes.Una lúz roja en la trampilla de acceso al drive, indica que se está produ ciendo una lectura ó escritura,cuando está encendida ,avi sando del peligro de la perdida de información, si en ese momento accediéramos al drive.Incluso se podria producir la rotura "física" del disco.

Los circuitos de control de la unidad de discos, están compuestos por un 8271 más los elementos TTL que compren den el separador de datos del diskette.El controlador del diskette, recibe comandos mediante los ports (puertos) de entrada-salida del procesador de la IOC,transfiere datos mediante el canal 1 de la DMA(Acceso directo a memoria), y establece la interface con el drive del diskette inte grado en el MDS.

Cuando se especifica una operación de lectura,el contro lador permanece en el modo "READ".Después de direcciona- do el sector y la pista sobre la que se quiere leer, el controlador comienza a ensamblar en bytes de 8 bits, los datos en serie leidos de el diskette.Cada vez que un by te es ensamblado, el controlador de diskette genera una señal de paso a la DMA (DMA REQ 1).El controlador de la DMA responde al controlador del diskette con la señal, DMA ACK 1, y entonces escribe el dato en la RAM de la IOC. Cuando han sido transmitidos los 128 bytes de un pista,

el IPC, lee en el registro de resultado de la transmisión, para comprobar si esta ha sido satisfactoria.En caso afirmativo lee el bloque de datos de la RAM de la IOC, y los procesa.

Cuando se realiza una operación de escritura, el controlador de diskette pasa de el modo "READ" ,al modo "WRITE", y genera una señal de paso a la DMA, para a través de su controlador,pasar a acceder a el primer dato que se desea escribir de la RAM de la IOC.(La IPC debe escribir,primero, el bloque de datos en la RAM de la IOC, antes de comenzar la operación de escritura)Una vez posicionado,comienza la escritura de un byte, mientras el siguiente es accedido por medio de la memoria RAM.Cuando toda la operación de escritura de un bloque ha sido completada,el controlador coloca una marca (CRC) al final del bloque, en el disco.

4.2.4.- LA IMPRESORA

La impresora utilizada en nuestro sistema es una TRS-80 de Radio Shack, del tipo de matriz de puntos.Incluye una interface para la conección en paralelo con el sistema.

El circuito de control de la impresora esta compuesto por un microprocesador y una memoria ROM.

Tiene una capacidad flexible de escritura , y así, puede escrir :Mayúsculas,minúsculas,simbolos europeos, simbolos americanos, y gráficos.Ademas puede variar el número de caracteres por linea ,así como el numero de lineas por pulgada.

La velocidad de impresión es de 160 caracteres por segundo en el formato normal de densidad de caracteres(Es decir, 132 caracteres por linea).

La impresora está provista de un sistema de alarma, pa

ra que el usuario se percate cuando se acaba el papel.Este sistema de alarma detiene la impresión, y enciende una lúz de alerta en el panel frontal.Después de colocar más papel, es necesario reiniciar a la impresora apretando la tecla de RESTART.

La impresora tiene una serie de teclas de control e indicadores.Estos són:

a) TECLAS:

POWER ON-OFF .-Es el interruptor de apagado y encendido de la impresora.

ON/OFF-LINE .-Esta tecla habilita a la impresora, para recibir información proveniente del MDS.

LINE-FEED .-Esta tecla, produce el avance de un renglón de el papel.Si permaneciera pulsada,el papel avanzaria continuamente.

SELF-TEST.- Esta tecla produce el testeado de todos los posibles caracteres que puede imprimir la impresora.

▽ 1/8 LINE .-Avanza 1/8 de linea a el papel.

△ 1/8 LINE .-Retraza 1/8 de linea a el papel.

RESTART .-Esta tecla se utiliza, para reiniciar a la impresora cuando se acaba el papel, y la señal de alerta está encendida.

b) INDICADORES:

READY .-Lámpara piloto que permanece encendida cuando la impresora está en ON-LINE.

ALERT .-Esta lámpara piloto indica que el papel se ha terminado.

POWER .-Esta lámpara piloto indica que la impresora está alimentada de la red.

Hay 34 códigos de control que se suman a los códigos

de los caracteres imprimibles, según la American Standar Code for Información Interchange (ASCII).Estos códigos de control són enviados como datos, y al ser recibidos por la impresora,son ejecutados como si se tratara de una instrucción.

La tabla 4.3 muestra a estos códigos.

CODIGO			SIMBOLO	FUNCION
DECIMAL	OCTAL	HEX		
0	000	00	NULL	Estos són ignorados
1	001	01	NULL	
10 ó	012	0A	LF	Avanza una linea el
138	ó 312	ó 8A		puntero.
13 ó	015	0D	CR	Retorna al puntero al
141	ó 315	ó 8A		principio de la linea.
14	016	0E	ULE	Finalizado de subrayado.
15	017	0F	ULS	Inicio de subrayado.
27⊕	033⊕	18⊕	CON	Inicializa la impre
14	016	0E	STR	sora para escritura densa.
27⊕	033⊕	1B⊕	CON	Cancela la escritura
15	017	0F	END	condensada.
27⊕	033⊕	1B⊕	BOLD	Fin de caracteres finos
30	036	1E	END	en la impresión.
27⊕	033⊕	1B⊕	BOLD	Comienza la escritura
31	037	1F	STR	de caracteres claros.
27⊕	033⊕	1B⊕	1/12L	Imprime 12 lineas por
28	034	1C		pulgada.
27⊕	033⊕	1B⊕	1/6L	Imprime 6 lineas por
54	066	36		pulgada.
27⊕	033⊕	1B⊕	1/8L	Imprime 8 lineas por
56	068	38		pulgada.
30	036	1E	NOR	Cancela caracteres elongados.
31	037	1F	ELN	Escribe caracteres elongados.
127	177	7F	DEL	Estos són ignorados.
255	377	FF	DEL	

Tabla 4.3

La tabla de caracteres ASCII se encuentra en el apéndice.

4.2.5.- EL ICE-85.

Un módulo ICE(IN CIRCUIT EMULATOR),es un dispositivo que nos permite emular sobre el propio circuito diseñado, cada una de las partes de las que se compone.Es decir, nos permite simular en tiempo real, cualquiera de las partes de el circuito, como puede ser la ROM, la RAM, ó el propio microprocesador y los ports de entrada-salida.Además puede ser utilizado como analizador lógico mediante una sonda de ,en nustro caso, 18 canales.

El sistema MDS-221, con el ICE-85, contiene dos microprocesadores.El procesador del INTELLEC, supervisa las operaciones del sistema, ejecuta comandos del monitor, y controla los sistemas periféricos.Un segundo procesador, el procesador del ICE,trabaja de interface entre el prototipo del diseñador ó el sistema de producción y el sistema INTELLEC.

La conección entre el ICE-85 y el prototipo, se realiza mediante un cable terminado en una clavija de 40 patillas, la cual, se coloca en el lugar destinado a el microprocesador a través de un zócalo.

El ICE-85 está compuesto de dos tarjetas de circuito impreso, la tarjeta de control y la tarjeta de traza, y dos cintas cableadas que conectan el módulo ICE y el módulo de Trazas externas.Ademas el ICE contiene una serie de conectores de patillas que le permiten coordinar la emulación con posibles eventos externos.

El módulo de trazas externas, tiene una serie de 18 canales de prueba para conectar a distintas señales,en el prototipo,que se deseen monitorizar.Estos 18 canales son accesibles por medio de el control de trazas y la emulación.

Un diagrama de bloques de el ICE-85 se muestra en la

la figura 4.4 .

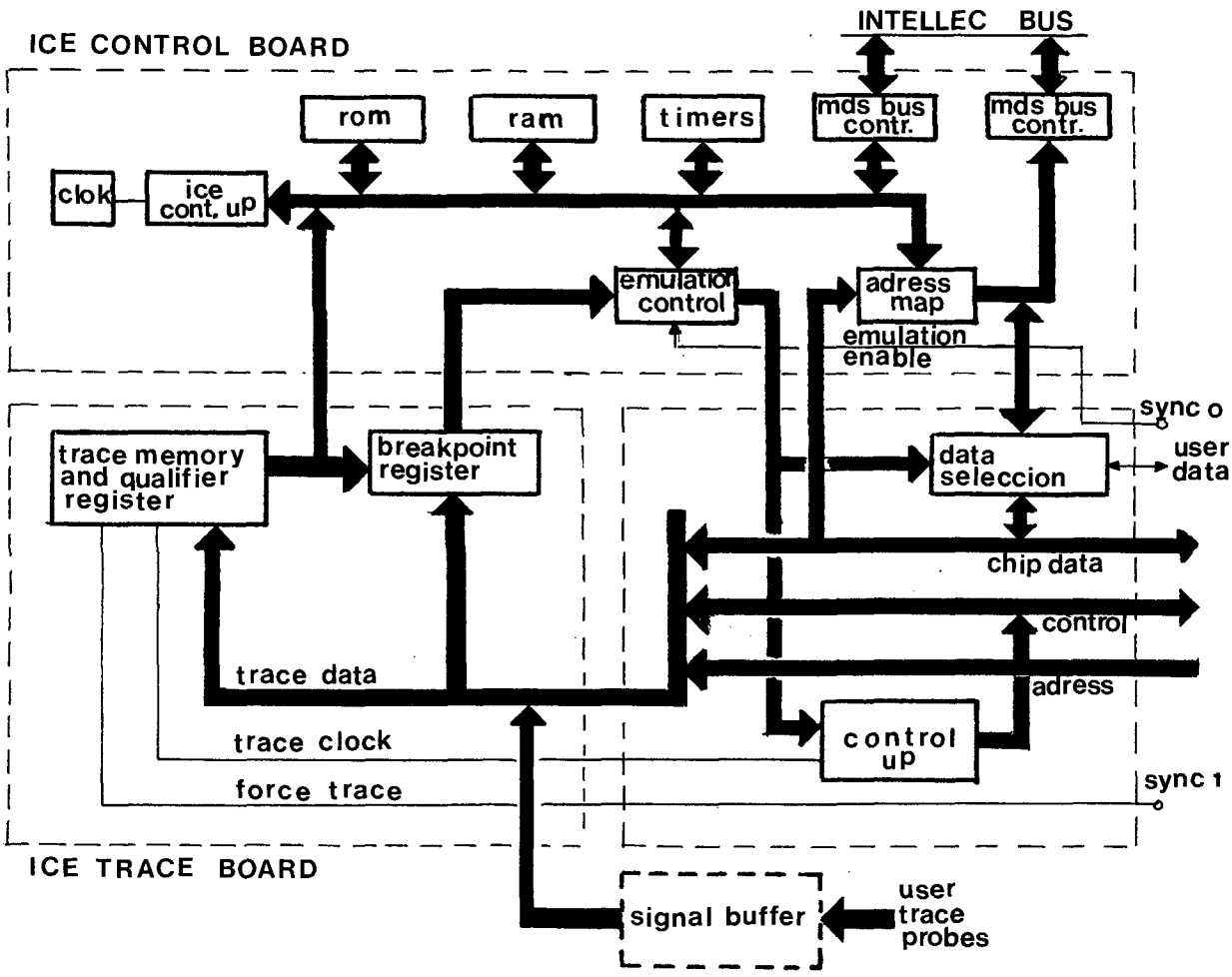


Figura 4.4

4.2.6.- EL GRABADOR DE PROM.

El grabador de PROM, es un unidad periférica de sistema de desarrollo MDS-221, usado en la programación de memorias de solo lectura.

El UPP (Programador de PROM), posee dós zócalos para la grabación simultanea de dós PROM´s, un interruptor de apagado-encendido de linea y su correspondiente piloto, un interruptor de reset, y un indicador de programación mediante una lámpara piloto.La tecla de RESET se usa cuando el sistema MDS no puede conectar con el UPP.Esta tecla no debe usarse nunca cuando el indicador de programación está encendido.

Cada uno de los dós zocalos del panel frontal están controlados por una placa de circuito impreso (PCB),que posee la circuiteria necesaria para controlar la programación de una clase particular de memorias.Para la grabación de un tipo ú otro de memoria, existen diferentes módulos de programación.La tabla 4.4 muestra todos los tipos de memorias programables por el UPP, y la tabla 4.5, muestra los diferentes zócalos necesarios para su instalación en el programador.

4.2.6.1.- PROGRAMACION CON UPM.

El mapeador universal de PROM (UPM),es el software que se aplica para la programación de todas las memorias INTEL del tipo PROM, mediante la utilización de 16 instrucciones diferentes.De estas 16 solo se usan generalmente 6, y las otras 10 son para una utilización muy especializada.

El UPM opera bajo el control del sistema operativo del sistema MDS, el ISIS-II.El formato de carga del UPM es:

-UPM

PROM Type	No. Pins	Organization	Personality Module	Description Section
1602A	24	256x8	UPP-872	5-26
1702A	24	256x8	UPP-872	5-26
1702A-2	24	256x8	UPP-872	5-26
1702A-6	24	256x8	UPP-872	5-26
1702AL	24	256x8	UPP-872	5-26
1702AL-2	24	256x8	UPP-872	5-26
2704	24	512x8	UPP-878	5-29
2708	24	1024x8	UPP-878	5-29
2716	24	2048x8	UPP-816	5-5
2732	24	4096x8	UPP-832	5-10
2758	24	1024x8	UPP-816	5-5
2758 S-1865	24	1024x8	UPP-816	5-5
3601	16	256x4	UPP-361	5-2
3602	16	512x4	UPP-865	5-21
3602A	16	512x4	UPP-865	5-21
3604	24	515x8	UPP-865	5-21
3604A	24	512x8	UPP-865	5-21
3604L-6	24	512x8	UPP-865	5-21
3604AL	24	512x8	UPP-865	5-21
3605	18	1024x4	UPP-865	5-21
3605A	18	1024x4	UPP-865	5-21
3608	24	1024x8	UPP-865	5-21
3621	16	256x4	UPP-865	5-21
3622	16	512x4	UPP-865	5-21
3622A	16	512x4	UPP-865	5-21
3624	24	512x8	UPP-865	5-21
3624A	24	512x8	UPP-865	5-21
3625	18	1024x4	UPP-865	5-21
3625A	18	1024x4	UPP-865	5-21
3628	24	1024x8	UPP-865	5-21
4702	24	256x8	UPP-872	5-26
4702A	24	256x8	UPP-872	5-26
8702A	24	256x8	UPP-872	5-26
8702A-4	24	256x8	UPP-872	5-26
8704	24	512x8	UPP-878	5-29
8708	24	1024x8	UPP-878	5-29
8741	40	1024x8	UPP-848*	5-13
8748	40	1024x8	UPP-848*	5-13
8755	40	2048x8	UPP-855 or 955** with UP1	5-17
8755A	40	2048x8	UPP-855 or 955** with UP2	5-17

TABLA 4.4

PROM	Adapter Required	Adapter Identification Label
2758 S-1865	UPP-555	UPP-555 PWA 4601633
3602 3602A	UPP-562	3602/3622 PWA 1000555
3604L-6 3604AL	UPP-555	UPP-555 PWA 4601633
3605 3605A	UPP-566*	3605/3625 PWA 1000745
3608	UPP-555	UPP-555 PWA 4601633
3621	UPP-562	3602/3622 PWA 1000555
3622 3622A	UPP-562	3602/3622 PWA 1000555
3625 3625A	UPP-566*	3605/3625 PWA 1000745
3628	UPP-555	UPP-555 PWA 4601633
8755	UP1	

TABLA 4.5

Name Type =	Note
1702A	Use also for 1702 and 1602A
2704	Use also for 8704
2708	
2716	
2732	
2758	Use also for 2758S-1865
3601	
3602	
3602A	
3604	Use also for 3604AL
3604A	
3604L	
3605	Use also for 3605A
3608	
3621	
3621A	
3622	
3622A	
3624	
3624A	
3625	Use also for 3625A
3628	
4702A	Use also for 4702
8702A	
8708	
8741	
8748	
8755	Use also for 8755A
For any PROM with a dash number (different speed or power), do not enter '-xx'. For any Mxxxx (Mil-spec parts), do not enter the M.	

TABLA 4.6

El sistema responderáde la manera siguiente:

ISIS-II PROM MAPPER Vx,y

TYPE*

donde TYPE indica al usuario que introduzca el código del tipo de memoria a grabar.Este código viene referido en la tabla 4.6.

Depués de introducido el código de la memoria, el sistema queda preparado para recibir cualquiera de los comandos:PROGRAM, PROGRAM", TRANSFER, COMPARE, READ, ó EXIT.

1.-Comando PROGRAM.

Su formato es:

PROGRAM FROM (Direc. baja) TO (Direc. alta) START (Direc.P.)

donde:PROGRAM es el nombre del comando, FROM (Direc. baja) indica la dirección de comienzo del preograma en la memoria del sitema,TO (Direc. alta) indica la dirección donde termina el programa, y START (Direc.PROM), indica a partir de que dirección de la PROM deseamos que se grabe nuestro programa.

2.-Comando PROGRAM".

Este comando es una variación del comando PROGRAM, y nos permite realizar una secuencia de programación identica a la seguida con un comando PROGRAM anterior.Es decir, repite el comando para la grabación de varias memorias en als mismas condiciones.Ejemplo:

*PROGRAM FROM Ø TO 255 START Ø

*PROGRAM"

*PROGRAM"

Esto hará,que la memoria PROM quede grabada con la información contenida entre las posiciones Ø y 255 de la memoria del sistema,partiendo de la posición Ø de la PROM. Sucesivas memorias PROM podran ser grabadas con al misma información mediante el comando PROGRAM".

3.-Comando TRANSFER.

Este comando se emplea para leer datos de una memoria PROM, y almacenarlo en un area determinada de la memoria del sistema, en posiciones lógicas de palabras.Su formato es:

*TRANSFER FROM (Direc. baja) TO (Direc. alta)

donde TRANSFER es el nombre del comando; FROM (Direc. baja) es la posicioń de memoria menor del bloque a mover y TO (Direc. alta)? es el valor del final del bloque que queremos mover.Ejemplo:

*TRANSFER FROM Ø TO 255

Los datos contenidos en al memoria PROM, desde la posición Ø en adelante, son transferidas a la posición lógica del sistema que va de Ø a la posición 255.

4.-Comando COMPARE.

Este comando hace una comparación byte a byte de los datos grabados en la memoria PROM, sacando en pantalla un aviso, para el usuario, que identifica la variación en el dato requerido.Su formato es:

*COMPARE FROM (Direc. baja) TO (Direc. alta)

donde COMPARE, es el nombre del comando y FROM (Direc. baja) y TO (Dire. alta) tienen el mismo significado que anteriormente.

En caso de encontrar un dato diferente entre la posición lógica y el valor que tiene en la memoria PROM, se saca en pantalla un mensaje como sigue:

*(Direc. PROM) M=(Contenido de Mem.) P=(Contenido PROM)

5.-Comando READ.

Este comando es usado para cargar datos provenientes de un fichero en la memoria INTELLEC para su utilización por el UPM.Su formato es:

*READ Tipo de fichero)FILE(Nombre de f.)INTO(Biass dire.)

donde READ es el nombre del comando, el tipo de fichero hace referencia al tipo de grabación en que está realizado, es decir, si está en código hexadecimal ó código objeto;FILE (Nombre de fichero), indica que fichero se va a cargar e INTO(Biass de dirección.), indica una cantidad constante a sumar a la dirección lógica a la hora de la grabación en la PROM.

6.-Comando EXIT.

Este comando nos saca de el programa UPM, y devuelve el control al ISIS-II.

Existen 10 comandos más, pero no tienen una utilización aplicable a este proyecto directamente.Estos son:

7.-Comando CHANGE.

Cambia el valor de una posición de memoria del sis

tema INTELLEC, y de las siguientes si se deseara.Su forma to es el siguiente:

*CHANGE(Direción lógica)=valol valor2,..etc

donde CHANGE es el nombre del comando y (Dirección lógica), indica la dirección del dato que queremos cambiar(Pero en el sistema).

8.-Comando DATA.

Este comando nos especifica si los datos a grabar en la memoria, ó a operar, són en complemento ó nó.Su formato es:

*DATA= Sentido

donde DATA es el nombre del comando, y sentido ha de ser ó el valor "T" (Datos sin complementar),ó el valor "F", (Datos complementados).

9.-Comando DISPLAY.

Este comando muestra en pantalla una zona de la memoria PROM.Su formato es:

*DISPLAY FROM(Dirección baja) TO (Dirección alta)

donde DISPLAY es el nombre del comando.Ejemplo:

*DISPLAY FROM Ø TO 15H

Este comando haria que apareciese en pantalla lo siguien

te:

*DISPLAY FROM Ø TO 15H

ØØØØ 3E 43 54 45 FE D2 4E 23 78 54 33 3Ø ØE 44 54 55

ØØ1Ø EE FA C3 D5 8F 23

Si al final del comando especificamos la sentencia, BINARY, los datos apareceran en binario.

10.-Comando FORMAT.

Se emplea para guardar datos en memoria de una manera específica.Su formato es:

*FORMAT (Especificación1),(Especificación2),...etc

Ejemplo:

*FORMAT OFFSET 66ØØH LOGICAL 4,OFFSET 67ØØH LOGICAL 4

Hace que los 4 primeros bits de una palabra de 8,se guarden a partir de la posición de memoria 66ØØH, yque los 4 bits menos significativos són almacenados a partir de la posición 67ØØH, pudiendo más tarde proceder a la grabación de dós memorias por separado.

11.-Comando LOGICAL.

Este comando presenta en pantalla ó inicializa el tamaño de la palabra de memoria lógica.Su formato es:

*LOGICAL=(Longitud de la palabra)

donde LOGICAL es el nombre del comando y la longitud de la palabra expresa el número de bits que la componen.

12.-Comando OFFSET.

Indica que posición de la memoria del INTELLEC contiene el primer dato a ser grabado.Displaya ó cambia esta posición de memoria, y si no se utiliza, al inicializar el sistema, se asume como OFFSET la dirección del primer dato disponible para su grabación.Su formato es:

*OFFSET=(Dirección de comienzo de datos)

13.-Comando SOCKET.

Este comando sirve para diferenciar entre uno u otro de los zócalos existentes en el UPP.Su formato es:

*SOCKET=(Número)

donde SOCKET es el nombre del comando y número indica el número del zócalo sobre el que se vá a efectuar la operación de entrada-salida de datos.

14.-Comando TYPE.

Este comando sirve para indicar que tipo de memoria es la que se vá a grabar.Su formato es:

*TYPE=(Tipo de PROM)

15.-Comando WRITE.

Este comando sirve para escribir datos desde la memoria INTELLEC en un fichero de diskette.Su formato es:

*WRITE FROM(Dir. baja) TO(Dir. alta) FILE(Nombre)(Tipo de f.)

donde WRITE es el nombre del comando, el valor de las di

recciones marcan el bloque lógico a grabar;FILE (Nombre de fichero),especifica a que fichero grabaremos los datos; tipo de fichero,indica de que tipo són los datos a almacenar en el fichero (Objeto ó hexadecimal).

16.-Comando STRIP.

Este comando es de uso exclusivo del microprocesador 8086 y aqui no se describe por considerarlo no ha lugar.

La figura 4.5 muestra lo que es una palabra lógica y su significado.

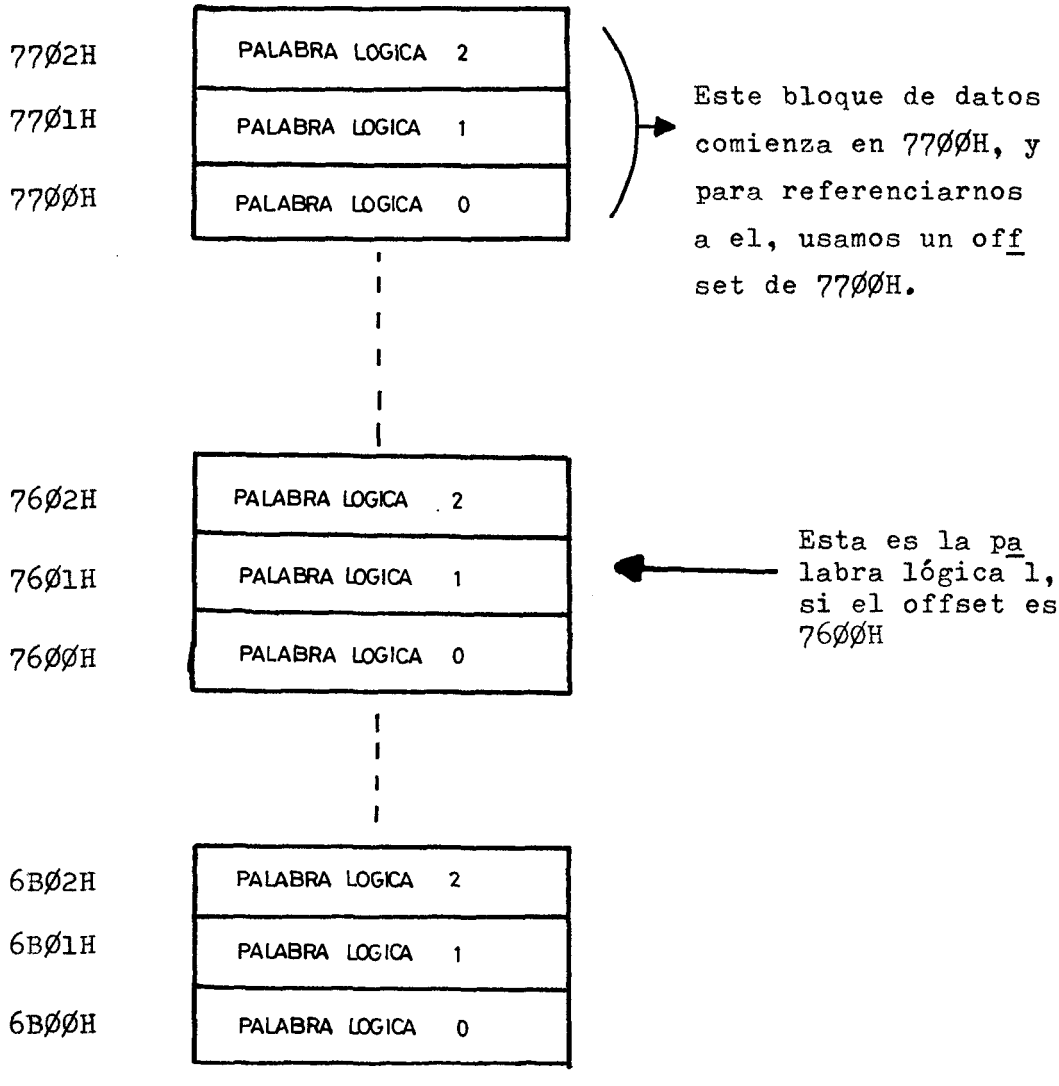


Figura 4.5

4.3.- SOPORTE SOFTWARE.

El soporte software del sistema de desarrollo MDS-221, está compuesto por un programa monitor, un editor de textos CREDIT, un sistema operativo ISIS-II, un programa de macroassembler, y un programa emulador.

4.3.1.- EL MONITOR.

El Monitor es un prggrama de control que supervisa todas las funciones del MDS.Además procesa una serie de comandos introducidos por el usuario, que se usan para diferentes funciones.Estos comandos se dividen en:

1),- Comandos de presentación y modificación.
2).- Comandos para ejecución de programas.
3).- Comandos de lectura y escritura de datos.

El MDS muestra en pantalla un (.) punto cuando se encuentra en el area de monitor, justo al margen izquierdo de la pantalla.El formato general de un comando es:

.Comando (Parámetro) (CR)

donde:

-Comando,es el caracter alfabético simple para el mismo.

-Parámetro, es uno ó más datos variables dados conjuntamente con el comando.El parametro debe ser número ó alfabético.

Estudiemos a continuación los diferentes comandos existentes en el monitor que són de utilidad.

1).- Comandos de control de la memoria.

-- A.-Comando D.

El comando "D", muestra en pantalla una sección de memoria, formateado en lineas de 16 bytes.Su formato es:

.D(Dir. baja),(Dir. alta) (CR)

donde:

-D, es el nombre del comando.

-(Dir. baja), especifica la dirección de comienzo del bloque ha displayar.

-(Dir. alta), especifica la dirección del final del bloque de memoria a presentar en pantalla.

-- B.-Comando F.

El comando "F", escribe una constante de 1 byte en un bloque determinado de memoria RAM.Su formato es:

.F(Dir. baja),(Dir. alta),(Constante) (CR)

donde:

-F, es el nombre del comando.

-(Dir. baja), especifica la dirección de comienzo del bloque de memoria,a fijar con el valor constante en todas sus direcciones.

-(Dir. alta), especifica el final del bloque de memoria.

-(Constante), es el valor hexadecimal que se vá a colocar en el bloque de memoria.

-- C.-Comando M.

El comando "M", copia un área especificada de memoria en otra área determinada de la memoria RAM. Su formato es:

.M(Dir. comienzo),(Dir. final),(Dir. destino) (CR)

donde:

-M, es el nombre del comando.

-(Dir. comienzo), es la dirección donde comienza el bloque de memoria a copiar.

-(Dir. final), es la dirección donde acaba el bloque de memoria a mover.

-(Dir. destino), es la dirección a partir de la la cual se copiará el bloque de memoria.

-- D.-Comando S.

El comando "S", permite la sustitución de un dato almacenado en una posición de memoria, por otro. Su formato es:

.S(Dir.),(Byte de dato),(Byte de dato),...(CR)

2).- Comandos de control de registros.

-- A.-Comando X.

El comando "X", produce la presentación en pantalla de todos los contenidos de lso registros de la CPU, más el registro de máscara.Su formato es el siguiente:

.X (CR)

-- B.-Comando X-registro.

Este comando nos permite displayar ó cambiar el contenido de un registro determinado.Su formato es el siguiente:

.X(Registro),(Dato)

donde:

-X es el nombre del comando.

-(Registro), es el nombre de cualquier registro de la CPU 8085.

-(Dato), es el valor que se vá a sustituir en registro indicado.

3).- Comandos de I/O para cinta de papel.

-- A.-Comando R.

Este comando lee una cinta de papel en formato hexadecimal, y carga los datos en memoria, según la localización asignada en al grabación. Su formato es:

.R(Bias) (CR)

donde:

-R, es el nombre del comando.

-(Bias), especifica un valor que se sumará al dato cargado desde la cinta de papel.

-- B.-Comando W.

Este comando graba el contenido de una sección de memoria en la unidad asignada.Su formato es:

.W(Dir. comienzo),(Dir. final) (CR)

donde:

-W, es el nombre del comando.

-(Dir. comienzo), es la dirección de comienzo de el bloque de memoria a grabar.

-(Dir. final), es la dirección del final del bloque de memoria.

-- C.-Comando E.

Este comando finaliza la grabación de un programa en al cinta de papel.Se puede introducir en el comando "E", un punto de entrada, que es la dirección de la primera instrucción a ser ejecutada.Su formato es:

.E(Punto de entrada) (CR)

donde:

-E, es el nombre del comando.

-(Punto de entrada), es la dirección de la primera instrucción a ser ejecutada.

-- D.-Comando N.

Este comando, inserta un caracter NULL, cuyo valor es ØØH, en la cola ó en al cabeza de la grabación, a efectos de una mayor facilidad para la grabación y lectura de un programa en cinta de papel.Su formato es:

.N (CR)

donde:

↑N, es el nombre del comando.

4).- Comando de ejecución G.

Este comando permite pasar el control, desde el monitor, al programa del usuario. Se pueden especificar, una dirección de comienzo, y hasta dos puntos de ruptura de la secuencia del programa, para estudiar el estado de los registros de la CPU. Su formato es:

.G(Dir. comienzo),(Punto de ruptura),(Punto de ruptura)(CR)

donde G, es el nombre del comando. Ejemplos:

Para pasar el control a la dirección del contador de programa:

.G (CR)

Para pasar el control al programa, cuya dirección de comienzo es la 3ØA:

.G3ØA (CR)

Para pasar el control al programa, cuya dirección de comienzo es la 3ØA y establecer un punto de ruptura en la dirección 4ØØ:

.G3ØA,-4ØØ (CR)

Al colocar la coma (,), aparece el separador (-).

Para pasar el control al programa cuyo punto de entrada es 3ØA , y colocar dos puntos de ruptura en la dirección 4ØØ y en la 5ØØ:

.G3ØA,-4ØØ,-5ØØ (CR)

4.3.2.- EL EDITOR DE TEXTOS

El programa CREDIT,es un editor de textos de tipo de pantalla orientada,para su uso con el ISIS II, en un sistema de Desarrollo INTELLEC de 64K bytes de memoria.
El programa CREDIT nos puede presentar en pantalla un fichero, mover el puntero en cualquier dirección, hacia algún punto en el texto, haciendo inserciones,borrados ú otras correcciones y ver los resultados de los cambios inmediatamente.También nos dá la posibilidad de retroceder o´acceder a una nueva página a lo largo de todo el fichero.

Para añadir un texto en medio de un fichero, moveremos el cursor hasta el punto de inserción, tecleamos Control-A (El caracter-comando para adición de texto),introducir el nuevo texto,y teclear Control-A otra vez. Para borrar texto, desde uno a varios caracteres, se sigue un método muy similar, utilizando el Control-Z.

El programa CREDIT también tiene una serie de comandos para la edición en modo línea.Estos comandos incluyen las funciones de edición mas complejas, tales como mover, copiar,comandos de iteración, macro definición, y operaciones externas de ficheros.A menos que lo deseemos previamente, en el modo de edición de línea no observamos las correcciones realizadas.

El programa CREDIT incluye además un comando de ayuda denominada H que saca en pantalla todas las funciones del CREDIT.Esto evita el uso de los manuales cada vez que surge una duda.

El comando CREDIT inicializa la sesión de edición.El formato del comando CREDIT es:

CREDIT nombre de fichero 1 (TO nombre de fichero 2)
(MACRO) ó (NO MACRO)

donde:

--nombre de fichero 1.-Tiene el formato:

:aparato:nombre.extensión.

-(:aparato:).- En nuestro sistema se omite con lo que CREDIT hace mención al drive :FØ:

-(Nombre).- Es el nombre del fichero,compuesto de 1 a 6 caracteres.

-(.Extensión).- Es el nombre de la extensión compuesta de 1 a 3 caracteres,precedidos de un punto (.).Su utilización es opcional.

--TO nombre de fichero 2.-Especifica el nombre del fichero de salida, permaneciendo el fichero 1 en el disco.Si fuera omitido, la nueva versión del fichero, quedaria con el nombre del fichero 1 y la vieja quedaria con el mismo nombre del fichero 1, pero con una extensión (.BAK).

--MACRO.-Habilita la utilización de un fichero de comandos.

--NOMACRO.-Especifica la nó utilización de macros.

4.3.2.1.- FINALIZACION DE UNA SESION DE EDITADO.

Existen tres maneras de finalizar una sesión de editado.Estas són:

1) Reemplazar la vieja versión del fichero con la versión renovada.
2) Almacenar la versión renovada con un nombre diferente al que tenia el fichero.
3) Ignorar cualquier cambio y dejar el fichero tal como estaba, antes de la edición.

Esto se consigue mediante la utilización de dos comandos: a) Comando de salida EX y b) Comando de salida EQ.

a).- Comando EX: El comando de salida EX, finaliza la edición y almacena la nueva versión del fichero,

de tal manera, que podemos dejar la nueva versión con el antiguo nombre, o con uno diferente.Esto se realiza de la siguiente forma:

El comando EX, tiene el formato:

EX(Nombre de fichero)

donde (Nombre de fichero), es el nombre del fichero de salida para la nueva versión.Si se omite, la nueva versión del fichero queda almacenada con el antiguo nombre, y la vieja queda almacenada co el antiguo nombre más una extensión (.BAK).

b).-Comando EQ: El comando de aborto, ignora todos los cambios producidos en el fichero durante la edición, y lo deja tal como estaba.Para reducir posibilidades de error, el programa CREDIT pregunta al usuario, sí realmente desea abortar los cambios con la palabra siguiente:

QUIT?

Si contestamos "Y" ó "y", se abortan los cambios realizados.Si contestamos cualquier otro caracter, continuará la sesión de editado.

Su formato es :

EQ

4.3.2.2.-CONTROLES DE MODO PANTALLA.

Las funciones de edición, en modo pantalla, son:

a).- Inserción. Hay dos tipos:

1) Añadir texto.Su formato es:

CNTL-A "Texto" CNTL-A

Añade texto a partir del caracter donde se encuentra el cursor.

2) Añadir caracter. Su formato es:

CNTL-C "Caracter"

Añade un caracter en lugar donde se encuentra el cursor.

b).- Borrado.- Hay dós tipos:

1) Borrar texto.Su formato es:

CNTL-Z "Mover cursor" CNTL-Z

Borra,desde donde situamos el primer CNTL-Z, hasta donde se coloca el segundo CNTL-Z,sin borrar este último caracter.

2) Borrado de un caracter.Su formato es:

CNTL-D "Caracter"

Borra el caracter donde se encuentra el cursor.

c) Presentado en pantalla.Són trés:

1) Vista de página.Su formato es:

CNTL-V

Retorna la edición, desde el area de comandos,al area de pantalla.

2) Próxima página.Su formato es:

CNTL-N

Muestra en pantalla la siguiente página del fichero.

3) Página anterior.Su formato es:

CNTL-P

Muestra en pantalla la página anterior del fichero.

Si se nesecitará mayor poder, en las funciones de edición, se puede optar por el modo de edición en lineas de comandos.

4.3.2.3.- CONTROLES DE MODO COMANDO.

Los controles de edición en modo comando se denominan más exactamente comandos.

El programa CREDIT, incluye un juego de comandos para funciones más complejas de edición.Además para mover ó copiar textos de un sitio a otro en el fichero, se pueden definir un juego de funciones por el propio usuario mediante la definición de MACROS, que están compuestas de otros comandos del CREDIT.

La edición en modo comando, se puede definir como el proceso de introducción de comandos para su ejecución, en el área correspondiente de la pantalla.Hay cuatro clases de comandos diferentes:

a).- Comandos de puntero y Tag, los cuales mueven el puntero y establecen marcas auxiliares de posición, a lo largo del fichero,llamadas Tags.

b).- Comandos de texto, los cuales afectan al texto borrando, insertando, reemplazando ó imprimiendo caracteres.

c).-Comandos de busqueda, los cuales localizan series de caracteres.

d).- Comandos de iteración, los cuales producen una repetición de una serie de comandos.

Una vez situados en el area de comandos,mediante la tecla HOME, un asterisco (*), nos indica que podemos comenzar la introducción de comandos.Podemos introducir uno ó varios,separados por (;), pero siempre terminando la serie de comandos con un (CR), para su ejecución.

Ejemplo:

*Comando;Comando;Comando; . . . ;(CR)

Si tuviéramos demasiados comandos, y nó cupieran en una linea, se puede solucionar el problema mediante un ( ).

Ejemplo:

*Comando;Comando;Comando;Comando;&(CR)

**Comando;Comando(CR)

En la segunda linea aparecerá un doble asterisco (**), que nos indicará la posibilidad de seguir extendiendo la linea de comandos.

La sintaxis de un comando,es la siguiente:

*Nombre de comando(Argumento)

donde:

-* .-Es la señal que nos indica que estamos situados en el área de comandos.

-Nombre del comando .-Es el comando a ejecutar.

-Argumento .-Es opcional y puede constar de uno,dos, ó trés terminos.

Ejemplo:

*L3

Significa que el puntero se mueva trés lineas hacia adelante.

*XC,T1,T2

Significa, copiar todo el texto que se encuentra entre los tags T1 y T2 (Osea, un bloque de texto), a partir de la localización del puntero.

El programa CREDIT, provee a el usuario de un comando de ayuda.Este comando es el comando H, cuya sintaxis es:

*H

Este comando hace que aparezca en pantalla, un resumen de todos los comandos del CREDIT.

a) COMANDOS DE PUNTERO Y TAGS.

I.-Comandos de puntero.Se encargan de mover el puntero hacia delante y hacia atrás ó también hacia un tag.Son dós:L y J.

1) Comando de linea L.

Mueve el puntero un número especificado de lineas hacia delante ó hacia atrás, posicionandose en el primer caracter de la linea.Su formato es:

*L(Número)

donde la "L", es el comando de linea y el Número, es el número de lineas que se moverá el puntero y su valor podrá ser "Ø", positivo, ó negativo. con lo que se quedará en la misma linea,avanzará, ó retrocederá respectivamente.Si el Número se omite,el puntero pasa a la siguiente linea.Ejemplo:

*L-3

El puntero se posiciona en el primer caracter de trés lineas atrás.

2) Comando de salto J.

Mueve el puntero un número determinado de caracteres hacia delante, hacia atrás ó hacia un tag determinado.Su formato es:

*J(Número ó tag)

donde "J" es el nombre del comando,"Número especifica el número de caracteres que se desplazará el puntero, hacia delante ó hacia atrás y "Tag", que es opcional indica hacia que tag hay que saltar(Marca del fichero).Ejemplos:

*J-18

Salta 18 caracteres hacia atrás.

*JT1

Salta hacia donde se encuentre el tag número uno.

II.- Comandos de Tags.Estos comandos crean ó borran una marca en el fichero, que nos permite identificar un punto determinado del texto.Se pueden asignar hasta 10 marcas de tags,desde TØ a T9, que se suman a 4 marcas prefijadas por el CREDIT al inicio del editado.Estas són:

-TT .-Indica el comienzo del fichero.

-TE .-Indica el final del fichero.

-TB .-Indica el comienzo del texto en la memoria del sistema.

-TZ .-Indica el final del texto en la memoria.

Existen dos tipos de comandos de tag: TS y TD.

1) Comando de habilitación de tags TS.

Habilita un tag en el fichero.Su formato es:

*TSn

donde "TS", es el nombre del comando, y "n" es el número que se le asigna al tag (De Ø a 9). Ejemplo:

*TS3

Asigna a la posición en la que se encuentra el puntero, el tag número 3 (T3).

2) Comando de borrado de tags TD.

Borra un tag existente en el fichero.Su formato es:

*TDn

donde "TD" es el nombre del comando y "n" es el número del tag a borrar.Ejemplo:

*TD3

Borra el tag número 3.

b)COMANDOS DE TEXTO.

Estos comandos se encargan de la impresión, inserción,borrado,moviento y copia de texto.Hay 5 tipos.

1) Comando de impresión P.

Su formato es:

*P(n ó tag)

donde "P" es el nombre del comando, "n" es el número de lineas a imprimir(Puede ser negativo

positivo o cero), a partir del puntero.En caso de que "n" valga Ø ,se imprime la linea donde se encuentra el puntero.Y tag expresa el número de tag hasta el cual se imprimen las lineas a partir del puntero.

2) Comando de impresión hexadecimal PH.

Imprime el valor en hexadecimal,de un caracter ASCII, sobre el cual se encuentra el puntero en el texto.Su formato es:

*PHn

donde "PH" es el nombre del comando y "n" es el número de caracteres que se van a imprimir en hexadecimal, a partir del puntero."n" puede ser negativo, positivo ó cero.En caso de que valga Ø, no produce la conversión a hexadecimal. Ejemplo:

Si tenemos el puntero sobre la palabra "PEPE", en la primera "P",al ejecutarse: *PH4 ,obtenemos:

*PH4 (CR)
50 45 50 45

que corresponde al valor hexadecimal de PEPE en código ASCII.

3) Comando de inserción I.

El comando "I", inserta caracteres o texto, a partir de la posición del puntero.Su formato es:

*I/texto/

donde "I" es el nombre del comando y texto es

uno ó más caracteres a insertar.Ejemplo:

Si tuvieramos el texto "L2: MOV C,A ", con el puntero situado en 2,pasamos al área de comandos y hacemos:

*I/BYTE/ (CR)

Volviendo al área de pantalla tendremos:

LBYTE2: MOV C,A

4) Comandos de borrado DL y DC.

El formato del comando DL(Borrado de lineas)es:

*DLn

donde "DL" es el nombre del comando y "n" es el número de lineas a borrar, a partir de la posición en la que se encuentra el puntero.

El formato del comando DC(Borrado de caracteres), es:

*DCn

donde DC,es el nombre del comando, y "n" es el número de caracteres a borrar a partir de la posición del puntero."n" puede ser positivo, negativo ó cero.Si vale cero borra el caracter sobre el que se encuentra el puntero.

5) Comando de movimiento de texto XM.

El comando de movimiento XM, realiza una copia de un bloque de texto marcado entre dós tags, a partir de la posición del puntero.Además borra el bloque original de su antigua posición.Su formato es:

*XM Tn,(Tn ó número)

donde, el primer Tn indica el principio del bloque a copiar, el segundo Tn indica el final del bloque a copiar ó si indicaramos "n",el número de lineas a partir del primer tag que deseamos copiar.Ejemplo:

*XM T8,T9

Nos mueve el bloque de texto entre los tags T8 y T9 a partir de la posición de le puntero, borrando el bloque de su posición original.

*XM T3,10

Nos mueve 10 lineas a partir de T3, a la situación del puntero.

6) Comando de copiado de texto XC.

Este comando nos copia un bloque de caracteres definidos por tags, a partir de la posición del puntero, sin borrar el bloque original,es decir, lo duplica.Su formato es:

*XC Tn,(Tn ó número)

donde "XC" es el nombre del comando, el primer Tn indica el comienzo del bloque a copiar, el segundo Tn es el final del bloque a copiar, ó si utilizamos "n", nos indicará el número de lineas a copiar después del primer tag.

c) COMANDOS DE BUSQUEDA.

Hay dós comandos de búsqueda: EL "F" y el "S".

El comando "F", busca una serie de caracteres, a partir de la posición del puntero, y si los encuentra, situa a el puntero un carcter después del final del texto encontrado.

El comando "S", busca una serie de caracteres, y si los encuentra, los sustituirá por el texto definido con el comando.

Existen una serie de caracteres especiales,que nos permiten una flexibilidad en la búsqueda de un texto.Estos són:

1) Caracter interrogación, ?.

Es un comodín que se emplea para simular cualquier caracter en la búsqueda de un texto.

Ejemplo:

A?BC equivale a ATBC,A$BC,A8BC,...etc.

2) Caracter ¡Y (CNTL-Y).

Permite encontrar un texto que posea un nú cualquiera de caracteres, en el lugar donde es utilizado.Ejemplo:

A¡YBC equivale a AccccccBC,A222BC,...etc,

3) Caracter ¡W (CNTL-W).

Permite encontrar uns serie de caracteres, con una dualidad mayúscula-minúscula.Ejemplo:

¡WabcjW equivale a AbC,ABC,abC,Abc,...etc.

Los formatos de los comandos de búsqueda de caracteres son:

1) Comando encontrar, F.

Su formato es:

*F/texto/Tn ónúmero

donde "F", es el nombre del comando, texto es

la serie de caracteres a encontrar, y Tn indi ca hasta que tag (marca en el fichero), se rea liza la búsqueda del texto.Si en lugar de Tn aparece un número "n" en el comando, la búsque da se realiza, desde la posición del puntero hasta el numero de lineas indicadas por "n".

Ejemplo:

*F/LOOP:/T9

Busca la etiqueta LOOP:, entre la posición en la que se encuentra el puntero y el tag nú mero 9 (T9), y en cuanto encuentra el texto,él puntero se situa en el caracter inmediatamente posterior a LOOP: .

Si no se indica un tag ó un número de lineas sobre las que realizar la búsqueda, el comando asume que la búsqueda se realizará hasta el fi nal del fichero.

2) Comando de sustitución, S.

El comando "S", reemplaza una serie de carac teres por otra, una vez encontrada ésta.En el caso de no encontrar la serie de caracteres, se presentará en pantalla un mensaje "NOT FOUND".

El formato del comando "S", es:

*S ó SQ/Texto viejo/Texto nuevo/Tn ó número

donde "S", es el comando de sustitución, la cual se llevará a cabo solamente si se encuen tra el texto viejo; "SQ"se emplea en sustitu ción de "S", para realizar el remplazado de caracteres, solo si se contesta con "Y" ó "y"

al Editor;"Texto viejo", es el texto a sustituir, "Texto nuevo", es el texto que sustituye; Tn indica, hasata que tag se realizara la busqueda de la serie de caracteres, y en caso de que utilizáramos "Numero", este nos indicaria, sobre qué número de lineas a partir del cursor se realizará la búsqueda de el texto viejo.

Ejemplos:

*S/THEN:/NEXT:/

Cuando el comando encuentra THEN: , lo sustituye por NEXT: .En caso de que no lo encontrara ,aparaeceria en pantalla un mensaje que diria:

NOT FOUND

Supongamos que tenemos el texto:

THEN: MOV A,A

donde el cursor se encuentra al principio del fichero, 5 lineas más arriba.Si deseáramos que THEN:, fuera sustituido por NEXT:, pero que el Editor esperara primero a ver si nos interesa esta sustitución, hariamos lo siguiente:

*SQ/THEN:/NEXT:/5

5 THEN: MOV A,A ? (Y)

Retornando al área de pantalla ,veríamos:

NEXT: MOV A,A

Además al aparecer el mensaje de interrogación al usuario,se indica la linea numerada en la que se encuentra el puntero.

d) COMANDO DE ITERACION.

Produce la repetición de una serie de comandos, que se encuentran encerrados en un parentesis.Su formato es:

*n ó !(Comando;Comando;..;etc)(CR)

donde "n", indica el número de repeticiones.Si en vez de "n", utilizamos "!", se produce una repetición indefinida de la serie de comandos, hasta que han sido ejecutados a lo largo de todo el fichero.Ejemplo:

*10(S/MOV A,M/MOV M,A/)

Hace que a partir de la posición del cursor,se realicen 10 sustituciones del texto "MOV A,M", por el texto "MOV M,A" en caso de que fuera posible, ya que pudiera darse el caso de que el numero de textos antiguos que existen, fuera menor de diez.En este caso,aparecería un mensaje en pantalla,una vez hechas las sustituciones posibles, del tipo "NOT FOUND".

Este comando permite una profundidad de actuación de hasta 5 ángulos.

Ejemplo:

*!(S/MOV A,M/MOV M,A/)

Hace que,a partir de la posición del cursor, se realicen una serie de sustituciones indefinidas del texto viejo, "MOV A,M", por el texto nuevo, "MOV M,A", hasta que todos los textos viejos han sido sustituidos. Es decir, cuando todos los textos viejos són sustituidos el comando finaliza su ejecución.

4.3.3.- EL ISIS-II

1).- CONFIGURACION.

ISIS-II, viene del Intel System Implementación Supervisor y es un sistema operativo basado en diskette para el MDS con, como mínimo, 64k byts de memoria RAM.

ISIS-II maneja los siguiente perifericos:

- Programador universal de PROM.
- Impresora de Líneas.
- Lectora rápida de cinta de papel.
- Perforadora rápida de cinta de papel.
- Teletipo.
- Terminales de video.
- Diskettes.

Además puede soportar:

- In-Circuit Emulator (ICE-85).
- Periféricos no standard para los cuales el usuario deberá programar el sofware necesario para manejar estos periféricos.

Mediante sus comandos se puede:

- Editar Ficheros.
- Destruir Ficheros.
- Copiar Ficheros.
- Cambiarles el Nombre.
- Asignarles atributos (protegidos a escritura - sistema ...)

Construir Programas:

- Ensamblarlos o Compilarlos.
- Combinarlos (Link).
- Crear Librererias (Lib).
- Asignarles direcciones de carga (Locate).

- Cargarlos en Memoria y Ejecutarlos.

2).- ORGANIZACION DE MEMORIA.

En la figura 4.6 , se muestra un esquema de la configuración de la memoria del sistema:

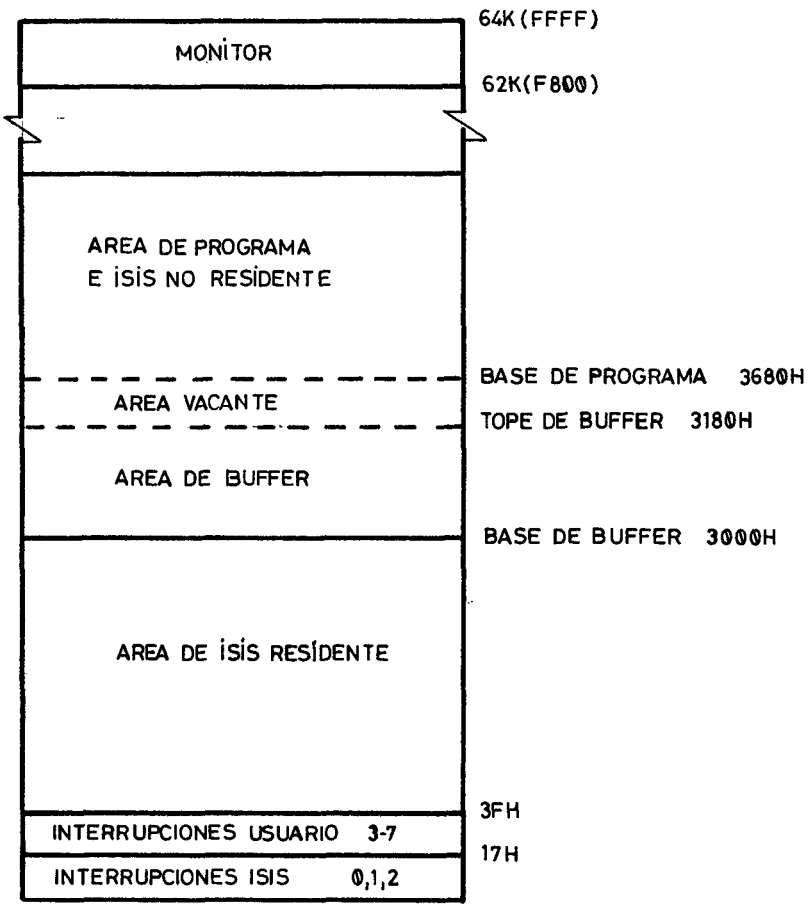


Figura 4.6

Una especial mención, merece el area de Buffer, en la que se encuentran definidos tres buffer de 128 bytes cada uno.Uno de ellos es usado por el ISIS, para su comunicación con la CONSOLA, y los otros dós pueden ser usados por el usuario.Por otra parte, el monitor ocupa los dos K más altos de la memoria, y el resto de memoria será usada por el programa del usuario y por los comandos del ISIS-II, que son usados por el usuario, y que fueron cargados desde un diskette.

3).- ENTRADAS/SALIDAS.

Todas las entradas/salidas las trata el MONITOR excepto los accesos al diskette, que la trata directamente el ISIS-II.

Cuando el ISIS-II necesita realizar una operacion de I/O que no sea la del diskette, llama a la rutina correspondiente del MONITOR.

Las I/O manejadas por el MONITOR son byt a byt, mientras que los accesos al diskette son por bloques (128 byts).

4).- IDENTIFICACIO DE PERIFERICOS.FICHEROS.

El ISIS-II maneja una serie de dispositivos preestablecidos como son:

:F0: a :F9: diskettes.

:T1: Teclado del TTY.

:TO: Impresora del TTY.

:TP: Perforador de cinta del TTY.

:TR: Lector del cinta del TTY.

:V1: Teclado de la unidad de video.

:VO: Pantalla de video.

:HP: Perforadora de cinta de papel de alta velocidad.

:HR: Lectora de cinta de papel de alta velocidad.

:LP: Impresora de Líneas.

Ademas hay otras dos "unidades lógicas" :CO: y :C1: que son la entrada y salida de consola y que pueden asignar(siempre ambos) a cualquier periférico que cumple

la condicion de que puede hacer entradas y salidas, mediante el comando CONSOL.

Los dispositivos se nombran entre (:).

Cuando se esta realizando una operacion con un dispositivo diskette debe decírsele al sistema operativo, - ademas de a que diskette se está accediendo, a la parte, dentro de ese diskette que se quiere acceder. Esto se logra dándole a cada programa dentro del diskette, un nombre y un"apellido".

El nombre debe tener de 1 a 6 caracteres alfanumericos pero el 1º debe ser alfabético y el apellido de 1 a 3 caracteres alfanúmericos. Nombre y "apellido" deben estar separados por un punto (.).

-INTERRUPCION Ø : Devuelve el control al MONITOR.

Puede usarse para parar cualquier proceso que esté ejecutando (incluso si es el ISIS el que tiene el control) y dar control al MONITOR para realizar un seguimiento de un programa, por ejemplo.

Si es un programa el que estaba ejecutando y hemos pedido una interrupcón Ø, podremos volver a relanzar dicho programa dando el comando, del monitor, G. El proceso se reanuda en la instrucción a la que apunte el contador de programa. (ver comando G del MONITOR).

Si el MONITOR tiene el control y queremos pasar el control al ISIS-II, solo podemos hacerlo dando el comando G8, del MONITOR.

-INTERRUPCION 1: Devuelve el control al ISIS-II.

Si se está ejecuntado un programa y queremos devolver el control al ISIS-II se pr siona la interrupcion 1.

Pero el programa no podrá ser relanzado en el punto donde se ha interrumpido.

Podría parecer que si es el MONITOR quien tiene el control y queremos pasarselo al ISIS, bastaría con que pulsásemos la interrupción 1, pero esto no es así ya que el MONITOR ENMASCARA TODAS LAS INTERRUPCIONES EXCEPTO LA Ø, con lo cual el pulsar la 1 no sirve de nada en estos casos.

Hay también que tener en cuenta que cuando se - produce una interrupción 1, sucede (si tiene el control el ISIS):

a) la :C1: y :CO: se reasignan a su valor inicial.

b) Una nueva copia del ISIS- residente se lee del diskette a memoria y pregunta por su - comando.

Cuando tiene control el MONITOR y presionamos la interrupción Ø , las unidades lógicas se reasignan a - sus valores iniciales.

5).- COMANDOS DE INICIALIZACION DE DISKETTES.COMANDO IDISK.

Un diskette en blanco debe ser inicializado antes de poder usarlo para soportar información que deba manejar el ISIS-II.

La sintaxis del comando será:

-IDISK (driver) (nombre) (s)

(driver) especifica el driver que contiene al nuevo diskette que queremos inicializar.

(nombre) es el nombre que le vamos a dar a ese nuevo diskette. Este nombre debe estar construído con las mismas reglas que el de un fichero, es decir (nombre.extensión).

(s) es opcional y específica.

Si se pone S, el nuevo diskette será del sistema y

sólo podrá ser usado como almacenamiento de ficheros de ficheros de usuario, pero no podrá ser colocado en el - DRIVEØ por controlar el microcomputador.

Un diskette del sistema debe tener como mínimo los siguientes ficheros:

-ISIS.DIR ISIS.LAB

-ISIS.MAP ISIS.BIN

-ISIS.TO ISIS.CLI

Sin embargo un diskette de no sistema sólo tendrá que tener:

-ISIS.DIR

-ISIS.MAP

-ISIS.TO

-ISIS.LAB

Los demás ficheros hay que copiarlos de un fichero en otro por COPY.

Ejemplo:

```
-IDISK :F1: SYST.V1 S
 SYSTEM DISKETTE
-
```

6).- COMANDOS DE CONTROL DE FICHEROS.COMANDO DIR.

Lista el directorio del diskette especificado, enviando esa información o´a la consola ó a un fichero que debemos especificar.

La sintáxis será:

-DIR (FOR(fichero)) (TO(fichero)) ((indicadores))

(FOR(fichero)) especifica el fichero del cual queremos obtener su directorio, pero sólo de ese fichero. Es opcional.

(TO(fichero)) especifica el fichero del cual queremos obtener que nos deje la infor acion del directorio, en ve$_z$

de mandárnosla a la pantalla de consola. También es opcional. También se puede especificar un periférico. Si se omite lo manda a consola.

((indicadores)) especifica el diskette del cual queremos toda o parte del directorio. Podrá ser Ø, 1, etc.

Si se omite el sistema interpreta que estamos hablando del diskette que está en el drive Ø.

Además del indicador anterior podremos especificar:

- F; indica que queremos un listado rápido, es decir, sólo nos dará el nombre y sus atributos. Si se omite nos dará toda la información.
- I; ijdica que queremos saber todas las entradas del directorio incluso las que tengan el atributo de invisibles si se omite sólo se listan las no invisibles.
- P; es un indicador que nos permite trabajar con varios diskettes aún en caso de tener un sólo driver. Si se omite interpreta que tenemos varios drivers.

El formato del listado sera:

```
DIRECTORY OF        nombre.extensión
NAME.EXT   BLKS   LENGTH   ATTR
-------------------------------
m/n  BLOCKS USED
```

donde nombre.extensión es el nombre del diskette del cual queremos listar el directorio.

Debajo de NAME.EXT nos dará los nombres de los ficheros de ese diskette.

Debajo de BLKS, nos indicará el número de bloques (conjunto de 128 byts) que ocupa el fichero.

Debajo de LENGTH nos dará su longitud en byts.

Debajo de ATTR nos dará los atributos de cada fichero.

m/n nos dirá:

m= nº de bloques usados del diskette.

n= nº total de bloques del diskette.

n podrá ser 2ØØ2 (simple densidad) ó 4ØØ4 (doble densidad).

Ejemplos:

```
-DIR
 DIRECTORY OF :FØ: ISØØAB.SYS
 NAME       .EXT       BLKS      LENGTH    ATTR
 PROGRA     .HEX         75        96ØØ       W
 SUMS       .OBJ         51        6528
 936/4ØØ4 BLOCKS USED

-DIR FOR PROGRA.    HEX
 NAME       .EXT       BLKS       LENGTH     ATTR
 PROGRA     .HEX         75        96ØØ        W
 936/4ØØ4   BLOCKS USED

-DIR I F
 DIRECTORY OF :FØ: ISØØAB.SYS
 PROGRA.HEX
 SUMS  .OBJ
 ISIS  .CAR
```

* COMANDO COPY.

Copia de un fichero de un sitio a otro. El sitio de donde y en donde se copia puede serficheros o periféricos. La sintáxis del comando será:

- COPY (fichero 1) ((fichero 2) ....) TO (fichero destino) ((indicadores))

(fichero 1) puede ser o un fichero de un diskette o un

periférico. Si es un fichero de diskette hay que indicar le en que driver está. Si es un fichero de diskette hay que indicarle en que driver está. Si se especifican más de un fichero ((fichero 2), (fichero 3) ....) etc., se concatenan en (fichero destino).

(fichero destino) puede ser o un fichero de diskette o un periférico. Si es un fichero hay que indicarle en que driver está.

Si (fichero destino) es un fichero que ya existe y que está protegido a escritura, el sistema pondrá el mensaje.

(fichero destino) WRITE PROTECTED

Si (fichero destino) es un fichero que ya existe y no está protegido a escritura, el sistema dará el mensaje

(fichero destino) FILE ALREADY EXISTS

DELETE?

Si conectamos YES y (CR) nos borra ese fichero y se ejecuta el comando.

Si contestamos NO, aobrta el comando y pregunta por otro. (indicadores) indican opciones de copia. Pueden ponerse u omitirse, uno o varios de los siguientes.

U suprime el 2º mensaje explicado anteriormente.

S cuando se especifica copia los ficheros con artículo (atributo) S.

N copia todos los que sean de NO SISTEMA

Q hace que si se usa con "y"? nos pregunte

COPY (fichero) TO (fichero) ?

Nosotros podemos constestar YES (CR) y lo compilará. Si le decimos NO (CR) pasará a preguntar por el siguiente

(C) es el indicador de atributos ; si C está especificado al nuevo fichero le pondrá los atributos del antiguo. Si no lo especificamos, el nuevo fichero no tendrá ninguń atributo.

(B) Es el indicador de borrado. Si está especificado, borra el fichero destino antes de copiarle el nuevo contenido sin mandaros el mensaje.

(fichero destino) ALREADY EXISTS

DELETE?

(P) Es el indicador de Pausa y nos permite trabajar con varios diskettes, si solo disponemos de un sólo "drive".

Ejemplos:

- COPY BOOK TO :LP:

COPIED BOOK TO :LP:

Copia el fichero BOOK de :FØ: en la impresora de - líneas.

- COPY :HR: TO :F1: SUM. FUE

COPIED :HR: TO :F1: SUM. FUE

Copia un fichero desde el lector rápido de cinta d e papel en el diskette :F1: y le da el nombre SUM. FUE.

NOTA: Se pueden usar * y ? en los nombres de los ficheros pero hay que ceñirse a unas reglas muy estrictas.

* COMANDO DELETE

Nos borrará del directorio, los nombres que le especifiquemos. NO BORRARA AQUELLOS FICHEROS QUE TENGAN LOS ATRIBUTOS DE FORMAT Y PROTEGIDO A ESCRITURA.

La sintáxis será:

-DELETE (fichero 1) (Q) ((fichero2) (Q) ...) (P)

(fichero 1), (fichero 2) .... especifica el nombre del fichero que se quiere borrar. Se pueden especificar en el nombre de los ficheros, * y ? para borrar grupos de ficheros. Si estos ficheros no están protegidos a escritura, se borra el fichero y aparece un mensaje de confirmación de borrado.

Si (fichero) no existe escribirá el mensaje

(fichero) NO SUCH FILE

Si fichero está protegido a escritura escribirá el emnsaje

(fichero) WRITE PROTECTED

Si despues de cada fichero se especifica Q el sistema nos preguntaráantes de borrarlo, si realmente queremos borrarlo.

(fichero) DELETE?

y como siempre, podremos contesta YES (CR) o NO (CR).

(P) establece una pausa en caso de que tengamos varios diskettes y sólo un driver.

Ejemplos:

- DELETE CHAP?.* (CR)

:FØ: CHAP1. TXT, DELETED

:FØ: CHAP2. FUE, DELETED

:FØ: CHAP5. OBJ, DELETED

- DIR (CR)

NAME.	EXT	BLKS	LENGHT	FATR
ASM8Ø	V2	24	4320	W

24/4ØØ4 BLOKCS USED

- DELETE ASM 80.V2 (CR)

:FØ: ASM8Ø.V2? WRITE PROTECTED

- DELETE :F1: TENIS Q,REFLEJ Q, (CR)

:F1: TENIS DELETE? YES (CR)

:F1: TENIS DEEETED

:F1: REFLEJ DELETE? YES (CR)

:F1: REFLEJ DELETED

* COMANDO RENAME

Cambia el nombre de un fichero.Su formato es:

-RENAME (Nombre viejo) TO (Nombre nuevo)

donde (Nombre viejo), es el nombre del fichero que queremos cambiar, (Nombre nuevo) es el nombre que queremos poner al fichero.Ejemplo:

-RENAME MULT TO DIV

* COMANDO ATTRIB

Cambia ó adjudica atributos a un fichero.Su formato es:

-ATTRIB (Fichero) (Lista de atributos)

(fichero) es el nombre del fichero al cual le queremos adjudicar o cambiar los atributos.
(lista de atributos) es uno o más de los parámetros siguiente:

1Ø : NO INVISIBLE

I1 : INVISIBLE

WØ : NO PROTEGIDO A ESCRITURA

W1 : PROTEGIDO

FØ : NO FORMAT

F1 : FORMAT

Hay que tener cuidado al cambiar el FORMAT ya que se puede inicializar mal un nuevo diskette.

SØ : NO SISTEMA

S1 : SISTEMA

Si se especifican dos atributos contradictorios co je el de maś a la derecha como bueno.

Ejemplos:

- ATTRIB PROGRA. * WØ W1 Q

FILE COURRENT ATTRIBUTES

:FØ: PROGRA.FUE

:FØ: PROGRA.OBJ

7).- COMANDOS DE EJECUCION DE PROGRAMAS.

Hay tres formas de ejecutar un programa:

- Llamándolo y ejécutandolo directamente.
- Ejecutándolo bajo control del MONITOR y que nos ayudará en su depuración.
- Que el sistema ejecute el programa sin interven ción del operador.

COMANDO nombre del fichero.

Todos los comandos del ISIS-II (excepto DEBUG) son nombres de ficheros. Cuando se les nombra por consola, se cargan y ejecutan.

Esto mismo podemos hacer con cualquier programa - que por supuesto, esté en objeto.

Ejemplo :

-SUMA

* COMANDO DEBUG

Se realiza una ejecución bajo control del MONITOR. Carga el progrma especificado y pasa el control al moni-

tor, que puede utilizarse para la depuración del programa, cambiar registros, memoria, poner trazos, etc. (ver comando G del MONITOR).

La sintáxis del comando será:

-DEBUG (PROGRAMA) (Parametros)

donde:

(PROGRAMA), es cualquiera de los programas ejecutables en modulo objeto que esten en el diskette.Si se omite se pasa control al monitor.

(Parametros), son los parametros que nesecita el programa para su ejecución.

Una vez que el sistema carga el programa y le dá control al monitor, este nos muestra la dirección de comienzo de nuestro programa.Ejemplo:

-DEBUG SUMA.OBJ (CR)

≠368Ø

.G,-36FF

* COMANDO FORMAT.

Formatea un nuevo disco al igual que el comando IDISK, pero con la diferencia de que copia todos los ficheros que tengan en el atributo "F".Su formato es:

-FORMAT(Driver)(Nombre) (Indicador)

donde (Drive) se refiere a el código del drive que contiene el disco a ser formateado; (Nombre), especifica el nombre del nuevo disco; (Indicador) es uno de los dos valores "A" ó "S".Si es "S", se copian los ficheros de sistema y si es "A", se copian todos los ficheros del disco fuente.

7).- COMANDOS DE CONTROL DE PROGRAMACION MODULAR.

El trabajo de programación se hace más facil si se puede programar por módulos.Las ventajas que se pueden alcanzar con este tipo de programación són:

-Es más facil programar módulos pequeños que uno grande.

-Es más facil encontrar errores en un programa pequeño que en uno grande.

-Es mas asequible la variación de un sector del programa pués solo habria que cambiar un módulo.

Para poder realizar este tipo de programación se usan dos comandos= El comando LINK y el comando LOCATE.

* COMANDO LINK.

Este comando combina diversos ficheros en módulo objeto en un módulo objeto principal,pudiendo ser relocalizable,más tarde en otro fichero distinto.Su formato es:

-LINK (Lista de entrada) TO (Lista de salida)(Controles)

donde:

-(Lista de entrada) podra ser cualquiera de las siguientes opciones:

1- (Filename) (Modname)....

(Filename) especifica un fichero en módulo objeto ó un fichero de una libreria,támbien en módulo objeto.

Si filename es un fichero de libreria y (Modname) osea nomres de módulos de la libreria, están especificados, solo los módulos especificados de la libreria pasarán al módulo final.

Si (Modnames) no són especificados, solo pasaran al fichero de salida aquellos que satisfagan las referen

-(Lista de controles) será uno ó más de los siguientes:

-MAP .-Lista un mapa de memoria, sobre el proceso de de localizado indicando información sobre las posiciones de los segmentos de código,stack,memoria y posiciones absolutas.

-PRINT(Fichero) .-indica en que fichero se vá a guardar el mapa del localizado.Si se omite se presentará en la consola.

-PUBLICS .-Especifica la lista de simbolos que se desean que aparezcan en la tabla de simbolos, y que serán accesibles desde otros programas.

-ORDER(Referencia de segmentos), designa el orden de asignación de los segmentos en memoria.Si se omite, el orden será el siguiente:

CODE

STACK

/COMON/ (Para Fortran-8Ø)

DATA

MEMORY

-CODE(Dirección) especifica la dirección a partir de la cual empezara el segmento de código.

-STACK(Dirección) especifica la dirección a partir de la cual empezara el segmento de stack.

-DATA(Dirección) especifica la dirección donde comenzará el segmento de datos.

-NAME(Filename) especifica el nombre del fichero de salida,del módulo localizado.

-RESTART Ø introduce instrucciones de salto en las direcciones 0,L,2 del módulo absoluto, para preparar las rutinas de tratamiento de la interrupción en el prototipo.

-START(Dirección) especifica la dirección de la prime

cias de programación.

2- PUBLICS (Filename) especifica ficheros que tienen módulos objeto que pueden ser utilizados por el usuario.

-(Lista de salida),especifica un fichero que tendra el módulo montado de salida.Este fichero no deberá estar en la (Lista de entrada).

-(Lista de controles), especifica diferentes opciones de montaje.Estan pueden ser:

-MAP .-Produce un mapa de la operación de montaje del fichero, pudiendo mandarlo a la consola óa un fichero determinado con el comando PRINT.

-NAME(Modname) .-Especifica el nombre del modulo montado.Si se omite toma el nombre del fichero de salida.

-PRINT(Filename) .-Especifica el nombre del fichero donde se vá a guardar el mapa de montaje.

-En el caso de que el comando LINK fuera demasiado grande, se usa un "&" antes de dar un (CR) y luego se continua en la linea siguiente.El plano 1 del apendice muestra un mapa de un linkado.

* COMANDO LOCATE.

Relocaliza,es decir ordena, un fichero en módulo objeto y produce otro con referncias de dirección absoluta.

Su formato es:

-LOCATE (Fichero de entrada) TO (Fichero de salida)(Cont.)

donde:

-(Fichero de entrada), es el fichero que contiene el programa en módulo objeto relocalizable.

-(Fichero de salida), será el fichero que contendrá el programa relocalizado en módulo objeto absoluto.Si se omite el LOCATE genera un fichero igual al fichero de entrada pero sin extensión.

la toma del fichero de entrada.

-STACKSIZE(Valor) especifica el valor en bytes del tamaño del stack.

En el plano 2 del apéndice se muestra el mapa de una operación de localizado.

* COMANDO LIB.

Este comando crea un fichero de libreria.Su formato es:

-LIB

Entonces preguntará por alguno de los comandos:CREATE, DELETE,ADD,LIST, y EXIT.

-CREATE crea un fichero de libreria vacio, al que se le añadiran otros ficheros en módulo objeto:

*CREATE(Nombre de fichero)

-ADD añade nuevos ficheros.Su formato es:

*ADD(Fichero)(Modnames) TO (Fichero de libreria)

-DELETE borra ficheros del programa de libreria.Su formato es:

*DELETE(Fichero de libreria)(Nombres de modulos)

Donde nombre de módulos especifica los módulos a borrar.

-LIST lista los ficheros de libreria y los simbolos publicos.Su formato es:

*LIST(F. de lib.)(Nombre de mod.)TO (F. list.)PUB.

-EXIT devuelve el control al ISIS-II.Su formato es:

*EXIT

Ejemplo:

```
-LIB
 ISIS-II LIBRARIAN V2.Ø
*CREATE FØØ.LIB
*ADD SUMA.OBJ,RESTA.OBJ TO FØØ.LIB
*LIST FØØ.LIB
*FØØ.LIB
*EXIT
```

4.3.4.- EL MACROASSEMBLER.

El programa macroassembler permite pasar de lenguaje ensamblador a lenguaje máquina ejecutable.El formato del macroassembler es:

-ASM8Ø fichero(lista de controles)

donde el "fichero" es el fichero donde se contiene el programa fuente y la "lista de control" incluye uno ó alguno de los siguientes:

-OBJECT(Fichero) Un fichero en módulo objeto es generado con el nombre (Fichero).Si se omite, se toma el nombre del fichero fuente con la extensión ".OBJ".

-NOOBJECT El código objeto no se genera.

-MOD85 Ensambla las instrucciones del 8085.

-DEBUG Si un fichero en código objeto es requerido, la tabla de simbolos es sacada al fichero.DEBUG no es operativo de otra manera.

-NODEBUG La tabla de simbolos no es incluida en el fichero objeto.

PRINT(Fichero) Un fichero de listado,es generado con el nombre del fichero dado en el control.Si se omite, se genera un fichero de listado con el nombre del fichero fuente con una extensión ".LST" .

-NOPRINT Suprime el fichero de listado.

-SYMBOLS Si un fichero de listado es abierto por PRINT, la tabla de simbolos es llevada a ese fichero.

-NOSYMBOLS La tabla de simbolos no se incluye en el fichero creado por PRINT.

-XREF Se introduce un "cross-reference", en

el fichero creado por PRINT.

-NOXREF — El "cross-reference" de simbolos es suprimido.

-MACROFILE — De fine un fichero que contiene instrucciones macros.

-NOMACROFILE — Especifica que el fichero no tiene macros en su interior.

-PAGELENGTH(n) — Indica el número de lineas por página del fichero de listado.

-PAGING — El macroassembler separa el listado en páginas con encabezadores.

-NOPAGING — El listado no es separado con encabezadores de página.

-MACRODEBUG — El ensamblador genera macro-simbolos que són sacados a los ficheros de listado y objeto.

-NOMACRODEBUG — Los macro-simbolos no són sacados en los ficheros de listado y objeto.

-TTY — Salida para teletipo.

-NOTTY — No se habilita la salida para TTY.

Estos controles,son denominados controles primarios.Hay otro tipo de controles,que se denominan controles generales. Estos controles generales són:

-INCLUDE(Fichero) — Introduce lineas fuentes de un fichero determinado, en el fichero que se está operando, dando una facilidad para mantener una macrolibreria.

-LIST — Una lista del fichero ensamblado es llevada al fichero indicado por PRINT.

-NOLIST — Se suprime la lista del ensamblado exepto, las lineas con errores.

-GEN El texto fuente de la macroexpansión en el fichero es listada si se selec ciona LIST.

-NOGEN Se suprime el texto fuente de las ma cro expansiones.

-TITLE('texto') Asigna un titulo al encabezado de ca da página del listado, colocandose en la segunda linea de cada página.

Las omisiones de controles asumidas por el ISIS-II son:

-OBJECT(Fichero.OBJ)

-NODEBUG

-PRINT(Fichero.LST)

-LIST

-SYMBOLS

-GEN

-NOXREF

-NOMACRODEBUG

NOMACROFILE

-PAGING

-PAGELENGTH(66)

-PAGEWIDTH(120)

A la hora de utilizar controles en el código fuente de un programa, se utiliza el simbolo "$".Su formato es:

$Control lista

donde $ debe aparecer en la primera columna del código fuente.Ejemplo:

$LIST DEBUG XREF MOD85

Las lineas de control deben aparecer antes de cualquier otra cosa en el fichero.

Un listado de un programa ensamblado se muestra en el plano 3 de el apéndice.

4.3.5.- EL EMULADOR (ICE-85).

El ICE-85, es una herramienta para el desarrollo de sistemas basados en el uP 8Ø85 de Intel, que nos permite emular, es decir, simular en tiempo real y sobre el circuito, cualquier parte del mismo.Asi podremos comprobar el correcto comportamiento de la ROM, la RAM, el microprocesador, y los ports de entrada-salida (Interfases de conección con el exterior).Pero su caracteristica de desarrollo mas interesante , es la de comprobar el Software del sistema antes de que se haya construido el prototipo Hardware, evitando posibles fallos de funcionamiento del sistema integrado Hard-Soft.

El ICE permite establecer ciertas condiciones en la emulación del Software y del Hardware, bajo las cuales la ejecución del programa se detendrá.Por ejemplo:

> Tenemos un programa de recepción de información en el cual, al recibir un dato, se procesa y se muestra en pantalla.Pues bién, podemos hacer que el programa se ejecute hasta justo antes de que se vaya a recibir el dato, se detenga, y procedamos a la introducción de un valor, comprobando, seguidamenente, que la cecuencia de instrucciones se cumplan como son deseadas.

Estas condiciones de detencion del programa, se denominan Puntos de Ruptura de Software.Además para la emulación Hardware, el ICE toma puntos de ruptura externos, desde cualquier señal lógica del circuito, a travéz de puntas de prueva, completando de esta manera, las señales del bus del uP.

El poder de depurado del ICE, se hace patente también, en la facilidad de cambiar y mostrar las informaciones que describen el estado del prototipo.Los términos especificos que determinen ese estado son:

- Patillas de la CPU, registros y flags.
- El contenido del Stack (Zona reservada de mamoria)
- El contenido de la memoria.
- El contenido de los Puertos de I/O.

Además se puede requerir información sobre:

- La dirección ó el código de operación de la última instrucción emulada.
- Una traza de instrucciones emuladas más recientes, ciclos de máquinas, ó 18 líneas de datos del status del sistema a elección del usuario.
- Información sobre la ejecución de las subrutinas.

Especialmente, el ICE nos permite:

a).- Como emulador de la CPU.

Nos permite:

- Examinar y modificar todos los registros de la CPU y posiciones del RAM del prototipo.
- Ejecutar el programa paso a paso, multipaso ó continuo.
- Especificar puntos de ruptura condicional, con el fin de permitir un control más flexible sobre el depurado del Sofl y del Hard.

b).- Como emulador de ROM o RAM.

Nos permite utilizar como memeria del prototipo un área de la memoria de nuestro sistema de desarrollo.Esta es una importante alternativa, ya que en las primeras etapas del desarrollo del prototipo es posible que no tenga su memoria colocada.

En la sustitución de la memoria debería de haber alguna manera de protegerla de un acceso ilegal

por parte del prototipo.Esa protección se consigue con el ICE mediante, un mapeo de memoria desde el sistema de desarrollo.

c).- Como emulador de Ports de I/O.

El ICE, permite sustituir los Ports de I/O del prototipo por los Ports del sistema.Eso nos permite poder ejecutar los programas sin que fisicamente existan los Ports.Además sirve de comprobación del producto manufacturado.

d).- Como analizador lógico.

El ICE, controla y almacena direcciones, datos, y actividad de las señales de control en la CPU.Su trace de memoria, graba todas las direcciones, datos y señales de control, de manera sincrónica.Sacando en pantalla la memoria, podemos examinar el curso seguido por todos los registros y posiciones de memoria, total ó pausada del programa.De esta manera, la tarea de detección de errores se vé facilitada.

4.3.5.1.- DEPURADO SIMBOLICO.

Una de las facilidades del ICE es la referencia a simbolos y localizaciones de forma imaginaria, en un programa en módulo objeto absoluto.Así por ejemplo, no es necesario conocer, donde está localizada una variable ó donde se encuentra el código correspondiente a una etiqueta en ensamblador.Simplemente se usan los símbolos directamente. Por ejemplo, el comando del ICE-85:

*GO TILL .L2 EXECUTED

asume que L2, es una etiqueta y que la emulación continua

-rá hasta que la instrucción que se encuentra en esa localización ha sido ejecutada.© podemos decir:

*GO TILL .PRODUCTO WRITTEN

asume que la emulación se detendrá cuando el programa almacena cualquier dato el la localización correspondiente al simbolo PRODUCTO.También podemos decir:

*GO FROM .START TILL .L4 EXECUTED

asume que la emulación comenzará con la localización START, y continuará hasta la instrucción correspondiente a la etiqueta L4(Hasta que dicha instrucción ha sido ejecutada).

4.3.5.2.- COMANDOS DE ICE.

El comando de llamada del emulador ICE, tiene la sintaxis:

-ICE85

El ICE respoderá, entonces, con un asterisco (*) y esperará por algún comando.Estos comandos se dividen en tres grupos:

a).- Comandos de emulación.
b).- Comandos de interrogación.
c).- Comandos de utilidad.
d).- Comandos de mapeado.

a).-COMANDOS DE EMULACION.

Son los comandos GO, STEP, y CALL.

1).- Comando GO.

Este comando permite la emulación del prototipo en tiempo real, y hace que la emulacióm continue hasta que una condición de ruptura es satisfecha.Su sintaxis es :

*GO (FROM dirección)(Condiciones de parada)

donde:

-GO es el nombre del comando.

-(FROM dirección) comienzo de la emulación.

-(Condiciones de parada) final de la emulación.

2).- Comando STEP.

Realiza la emulación en pasos simples ó multiples, rompiendo la emulación cuando un determinado número de ins_ trucciones han sido realizadas, ó satisfecha una condición. Su formato es:

*Step (FROM dirección) (Condicones de STEP)

donde:

-STEP es el nombre del comando.

-(FROM dirección) comienzo de la emulación.

-(Condiciones de STEP) especifica las condiciones en las que es realizada la emulación paso a paso.

3).- Comando CALL

Habilita el usuario para introducir la emulación de una subrutina en cualquier punto.Su formato es:

*CALL (subrutina)

b).- COMANDOS DE INTERROGACION.

Este tipo de comandos presenta en pantalla un listado de las instrucciones ejecutadas hasta ese punto, y cambiar algunos terminos que aparezcan en el listado.

1).- Comando BASE.

Este comando establece la base numérica en la que serán presentados en pantalla los datos (Decimal,octal,hexadecimal,binario ó ASCII).Su formato es:

*BASE = (Y/Q/T/H/ASCII)

donde:

-BASE es el nombre del comando.

-Y especifica la base binaria.

-Q especifica la base octal.

-T especifica la base decimal.

-H especifica la base hexadecimal.

-ASCII especifica la base ASCII.

2).- Comando SUFFIX.

Establece la base numérica,en omisión, que se aplica a los datos introducidos por la consola.Su formato es:

*SUFFIX = (Y/Q/T/H)

donde:

-SUFFIX es el nombre del comando.

3).- Comando "Display".

Podemos sacar en pantalla el valor de cualquier termi

-no específico, con solo teclear su nombre.Su formato es :

*(Nombre de termino ó simbolo)

Ejemplo:

Supongamos que el contador de programa PC, tiene el valor 2ØØØH.Si tecleamos:

*PC

aparecerá en pantalla el siguiente mensaje:

*PC

2ØØØH

4).- Comando "Cambio".

El contenido de términos ó símbolos específicos, puede ser alterado introduciendo su nombre seguido por un valor. Su formato es:

*(Símbolo ó término) = (Valor numérico)

Ejemplo:

En un momento determinado de la emulación, deseamos que el PC.adquiera el valor 2Ø5ØH.Procederemos de esta manera:

*PC

3EFØH valor del PC antes del cambio.

*PC = 2Ø5ØH

*PC

2Ø5Ø valor del PC depués del cambio.

5).- Comando PRINT.

Hace que aparezca en pantalla, la información que deseamos observar.Su formato es:

* PRINT(+/- número ó ALL)

donde:

-PRINT es el nombre del comando.

-+/- número, indica un número decimal de líneas a ser presentadas en pantalla, provenientes del buffer de trazas.

-ALL indica que aparezca en pantalla el total de líneas contenidas en el Buffer de trazas.

c).- COMANDOS DE UTILIDAD.

Son los comandos LOAD, SAVE, LIST, DEFINE,EXIT y EVALUATE.

1).- Comando LOAD.

Carga un programa en módulo objeto y su tabla de símbolos,desde un diskette.Su formato es:

*LOAD (:Drive:Filename)

donde:

-LOAD es el nombre del comando.

-:Drive: es el indicativo del drive donde se encuentra el programa que.queremos cargar.

-Filename es el nombre del fichero que contiene el programa a ser emulado.

2).- Comando SAVE.

Almacena un programa en módulo objeto que se ha emulado, en un fichero.Su formato es:

*SAVE (:Drive:Filename)

donde:

-:Drive: especifica sobre que drive se realizará el al almacenado.

-Filename es el nombre del fichero que va a recibir el programa en módulo objeto emulado.

3).- Comando LIST.

Especifica sobre que fichero ó unidad de grabación será guardad la sesión de emulado, en cada una de sus operaciones.Su formato es:

*LIST (Unidad ó fichero)

donde:

-Unidad es el nombre de la unidad sobre la que se escribirá el listado de la sesión de emulado.Ejemplo:

*LIST :LP:

Asume que la sesión de emulado sea escrita en la impresora.

*LIST :FØ:SUMA.ICE

Asume que la sesión de emulado aparezca en el fichero SUMA.ICE .

4).- Comando DEFINE.

Introduce símbolos condicionales y sus valores,en la tabla de simbolos del fichero.Su formato es:

*DEFINE (.Simbolo) = (Valor ó simbolo)

donde:

-(.Símbolo) es el simbolo adicional imaginario introdu´
cido en el programa.

-(Valor ó símbolo) es un valor numérico ó un símbolo
absoluto presente en el programa.

5).- Comando EXIT.

Hace que retorne el control desde el ICE-85 al ISIS-II. Su formato es :

*EXIT

6).- Comando EVALUATE.

Este comando hace que se evalue un dato ó una operación aritmética, y se presente en pantalla, en todas las bases del sistema.Ejemplo:

*EVALUATE 25T - 11H - 8Q - ØØ11Y
53T 35H 65Q 11Ø1Ø1Y ´5´

d).- COMANDOS DE MAPEADO.

Los comandos del mapeade del ICE85, pueden referenciar se a dos tipos de memoria:

*Memoria del sitema prototipo.

Sustitución de memoria del sistema Intellec para la memoria del prototipo.

Para entender como el ICE85, distingue entre estos dos, primero debemos definir los términos "lógicos y fisicos" tal como son usados en el txto de la emulación.Por ejemplo:

Supongamos que todos meses la persona X, recibe el sueldo en su casa, cuya dirección es la 25ØØH.Después está persona cambia de domicilio a la 35ØØH, sin comunicarselo a la empresa."Logicamente", la empresa con

sidera que recide en la 25ØØH, pero "fisicamente" vive en la 35ØØH.Pero esto no es problema para cobrar, pues to que la oficina busca la manera de entregar el suel do (Mapeado de direcciones).

Similarmente, se pueden mapear direcciones lógicas de programa,en localizaciones físicas en la memoria del Intellec.Para el programa, las direcciones permanecen constantes, pero para el sistema Itellec, se produce una transformación a direcciones físicas hábiles.

Usando el comando MAP, se puede espesificar, memoria para el prtotipo y ports de I/O, que serán:No existen tes logicamente(GUARDED), logicamente y fisicamente, existen en la memoria del sistema de prototipo (USER), ó existe logicamente en el prototipo y fisicamente en la memoria del Intellec(INTELLEC).

La memoria del Intellec está compartida en bloques de 2K byters, que unicamente se encuentran en modo GU ARDED, es decir, que logicamente no existen. De esta manera cualquier referencia a una localización de me moria ó ports de I/O, hace que aparezca un mensaje de error en pantalla.

Las direcciones lógicas del programa del usuario, pu eden ir desde Ø a 65:535 (64K), y es dividida en 32 blo ques de 2K bytes.Cada bloque lógico de memoria puede recidir fisicamente el sistema del usuario ó en un blo que de 2K sin usar de la memoria del ICE.

5.- METODOLOGIA DEL DISEÑO DE UN SISTEMA BASADO EN uP.

Un sistema basado en un microprocesador (uP), se compone de cuatro elementos basicos: El uP,Memoria, Unidad de entrada-salida(I/O), y Reloj.

El uP, engloba una unidad aritmético-lógica, una unidad de control y una serie de registros, que en dèfinitiva són sus caracteristicas, las que personalizaran el sistema,con su juego de instrucciones y su arquitectura interna.

La memoria es un dispositivo que permite el almacenamiento de información, tanto de datos como de la secuencia de operaciones a realizar (Programa).La memoria se divide en dós tipos distintos: RAM, que permite la escritura y lectura de datos y ROM que solo permite lectura.

Las unidades de entrada-salida, facilitan el intercambio necesario de información, entre el sistema y el exterior, y en definitiva, són las unidades que permiten la aplicación practica del sistema para el fín para el que ha sido diseñado.

El reloj, se encarga de suministrar las secuencias necesarias (Timing), para poder ir procesando las operaciones que el microprocesador se encuentra en la memoria.El esquema que se muestra en al figura 5.1 describe un sistema basado en un uP.

Observamos que hay una clara via de comunicación, entre las diferentes partes del sistema.Esta via se denomina "BUS DEL SISTEMA".En la práctica , el Bus del Sistema, se compone de tres diferentes buses: Bus de datos, Bus de direcciones y Bus de control.

Ciñiendonos a la arquitectura básica de un uP de 8 bits, la configuración del sistema quedaria como se muestra en la figura 5.2 .

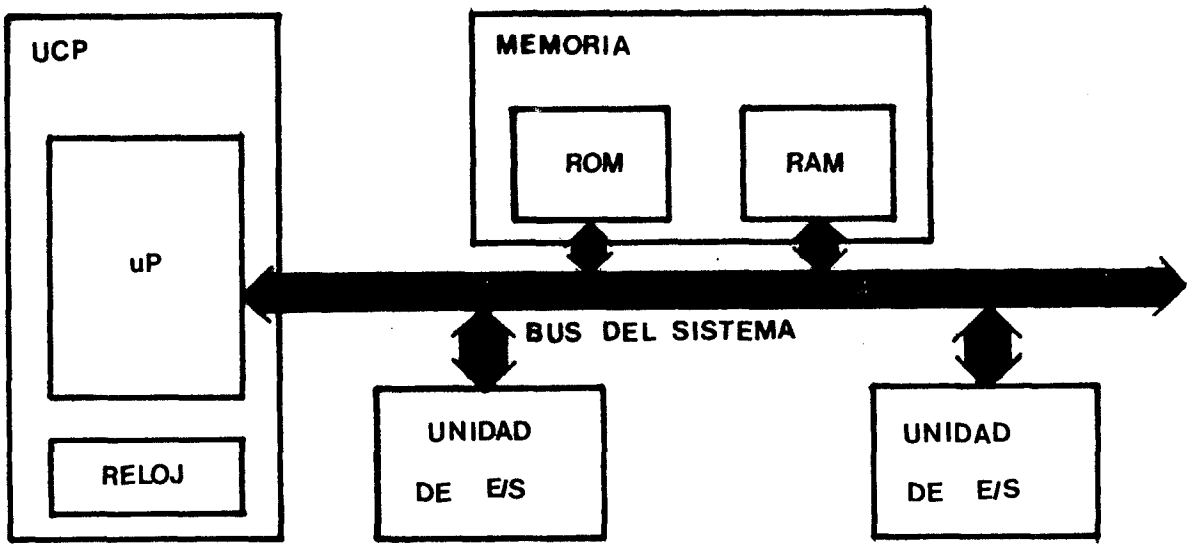


Figura 5.1

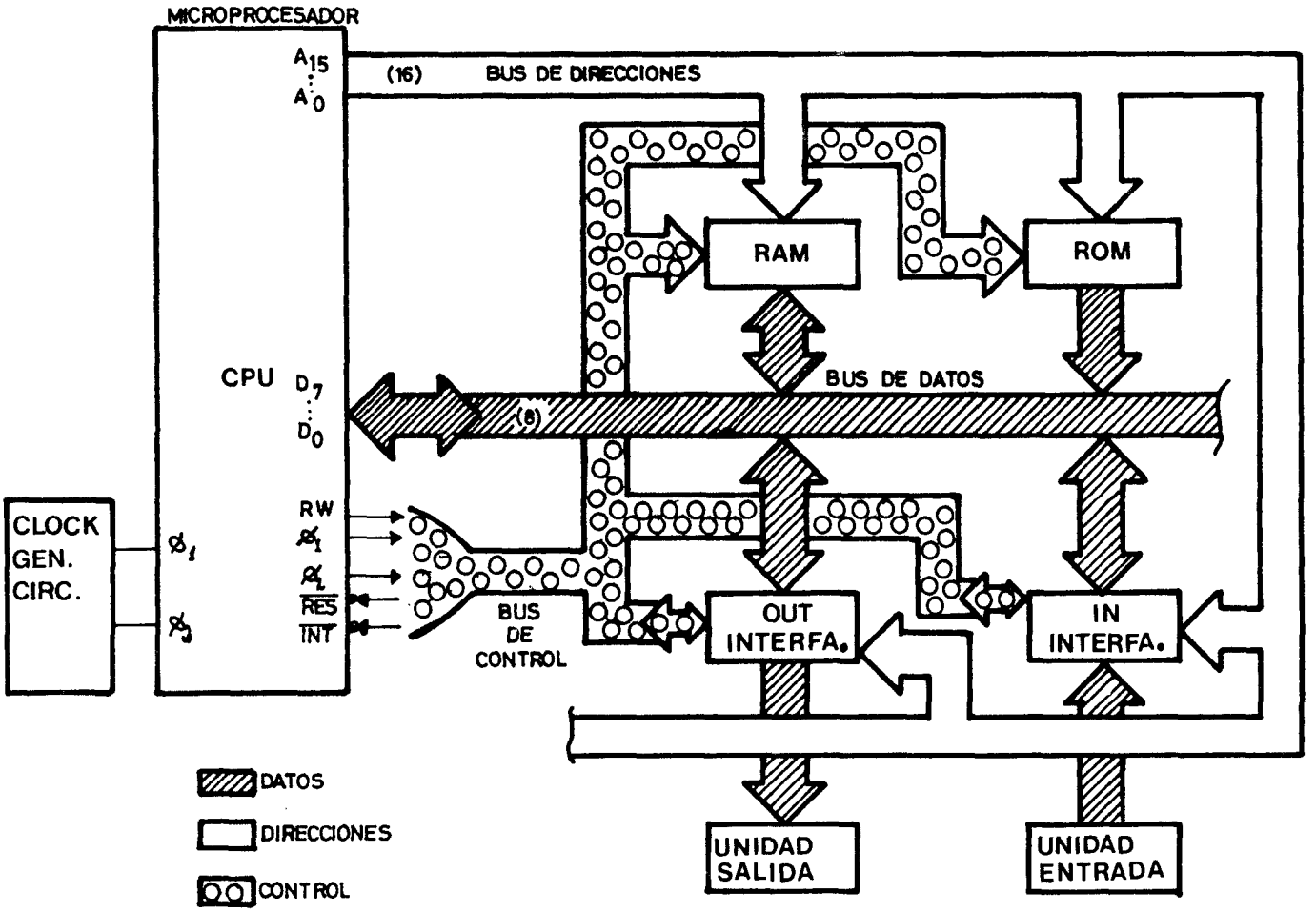


Figura 5.2

A través de los buses, la CPU conecta con el resto de los elementos del sistema, mandando o recibiendo información de una manera controlada.Si esta mandando información se dice que la CPU "escribe" en un dispositivo, y si por el contrario, esta recibiendo información, se dice que esta "leyendo".

Las caracteristicas de los buses són las siguientes:

***Bus de direcciones .- Es un conjunto de lineas unidireccionales, por las que envia la codificación binaria(Dirección), del elemento con el que la CPU va a comunicar.Tipicamente són 16 las lineas, lo que permite un direccionamiento de 64K bytes.Las lineas se designan,AØ...A15.

***Bus de datos .- Es un conjunto de lineas bidireccionales, a través de las cuales se intercambia información entre la CPU y el elemento seleccionado por el bus de direcciones.Es tipicamente de 8 lineas,lo que permite palabras de datos de 8 bits (Byte).Las lineas de este bus,se designan, DØ...D7 .

***Bus de control .- Es un conjunto de lineas, cada una de las cuales, tiene un significado específico, que permite sincronizar todos los elementos que constituyen el sistema.Así nos encontramos con las señales R/W, que indica si la CPU "lee" ó "escribe" un dato sobre una unidad, Ø1 y Ø2, que són señales que controlan el "Timing" del sistema, etc, etc.

5.1.- CONFIGURACION DE UN SISTEMA BASADO EN UN uP.

El primer paso en la configuracion de un microcomputador será, una veź elegidos los elementos necesarios (impuestos por los requerimientos de la aplicación), es la asignación a cada uno de ellos, de una identificación (Dirección) para que puedan ser identificados por el microprocesador de forma biunívoca.

Para ello hay que partir del espacio de memoria disponible, y asignar sobre él,direcciones a los diferentes elementos.Para esta asignación,es necesario tener en cuenta ciertas ligaduras impuestas por el micro elegido.Estas ligaduras son una serie de posiciones reservadas para funciones de control (Interrupciones, inicio,etc.).El resto del mapa es libre para que el usuario ubique cada elemento donde más convenga.Una buena regla es disponer elementos del mismo tipo, es zonas contiguas.Una disposición típica de un mapa de memoria para un microprocesador de 8 bits, puede ser el de la figura 5.3 .

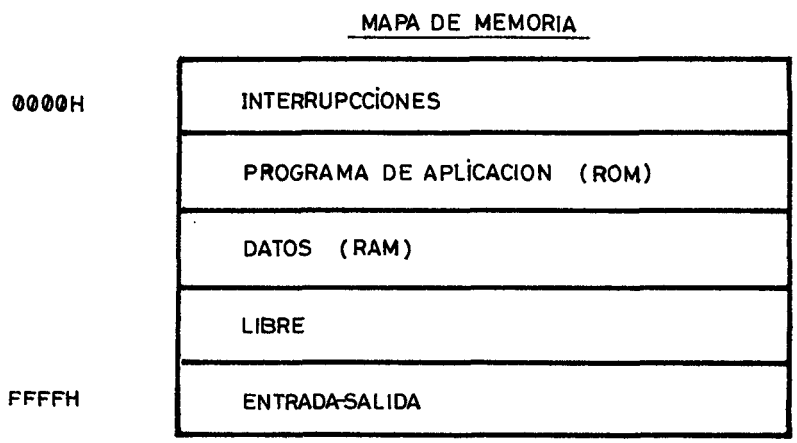


FIGURA 5.3

Una vez construido el mapa correspondiente al espacio de direcciones, hay que asociar a cada elemento su correspondiente circuito de decodificación, de forma paracial ó de forma total.La decodificación parcial es adecuada en pequeños sistemas donde el número de lementos es reducido, en comparación con el espacio total disponible, dando como consecuencia un costo más bajo de realización.En los sistemas medios ó grandes hay que utilizar la decodificación total de los elementos.

La decodificación parcial usa solamente un subconjunto del total de lineas de dirección para identificar el elemento.Así en el ejemplo de la figura 5.4 se utilizan las cuatro lineas más significativas de la dirección.

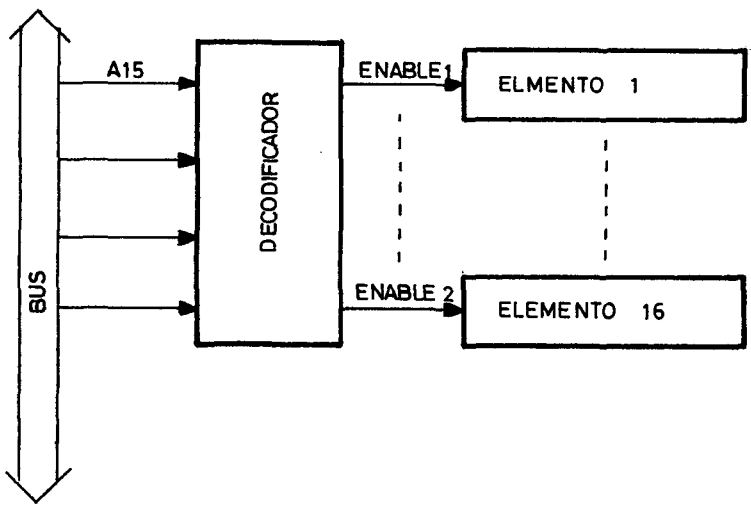


FIGURA 5.4

Todos los elementos conectados al bús del sistema pueden ser seleccionados en esta forma, ya que tanto la memoria como los periféricos, utilizan para la selección, una entrada de capacitación que puede ser activada por la salida del

decodificador.Pero no puede evitarse que al direccionar una determinada posición de memoria, se pierda la correspondencia única dirección-elemeto.

La decodificación total hace desaparecer esta ambiguedad utilizando para cada elemento todas las lineas de dirección necesarias, tal y como se indica ne la figura 5.5 .

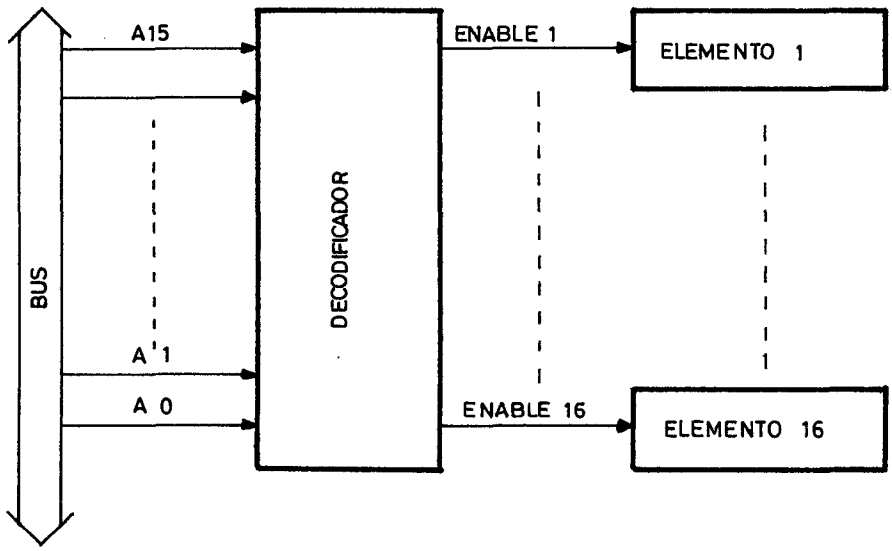


FIGURA 5.5

Esta segunda alternativa es, evidentemente, más costosa puesto que implica unos circuitos de selección más complejos, pero en sistema grandes es la solución válida yá que desaparece cualquier ambiguedad del selección.

El segundo problema asociado a la configuración de un sistema basado en un microprocesador, es el relativo a control del flujo de información sobre el bús del sistema, a fin de prevenir errores por interacción simultanea sobre él de varios elementos.Es decir ,asegurase que una transmisión "a" ó "desde" el microprocesador en un instante dado, sea transmitida ó recibida por un único elemento.La figura 5.6, mustra un esquema, donde se dibujan posibles ca

-minos entre dos elemntos y el microprocesador.

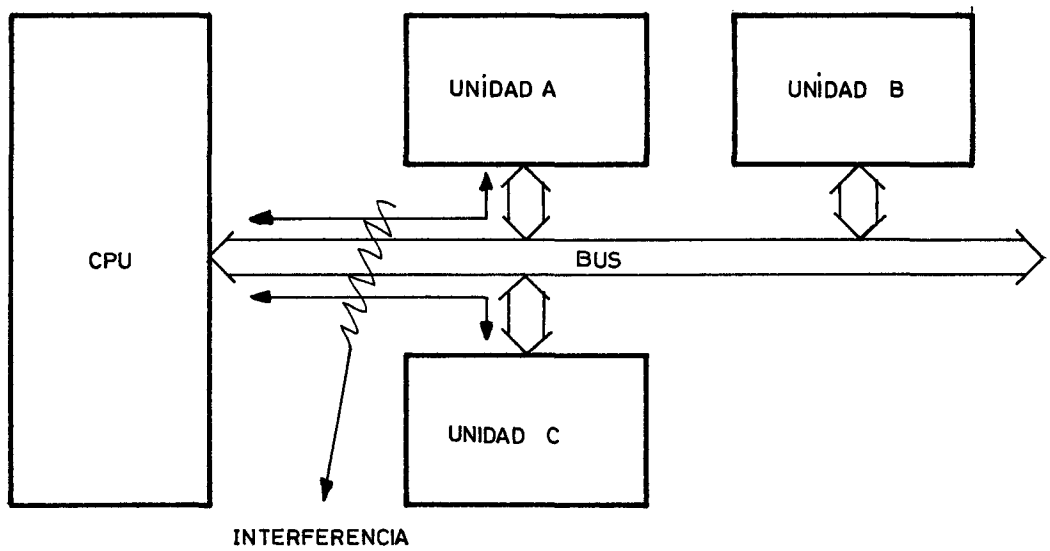


FIGURA 5.6

El camino micro-unidad A ó micro-unidad B, debe estar abierto en un instante dado, y permanecer el otro cerrado al paso de la información.Este problema se presenta fundamentalmente en la sección del bús de datos.

La solución es el empleo de elemtos en tri-state, los cuales en un momento determinado pueden aislar a un elemento del bus del sistema, previniendo interferencias de aquellos elemntos que no van a comunicar.

Una vez tenidos en cuenta estas cuestiones podemos pasar al estudio de la secuencia de trabajo en el diseño de un prototipo basado en la utilización.(En nuetro caso será aplicado más directamente al uP 8085 y a todo el soporte software y hardware que de él existe.)

5.2.- PASOS A SEGUIR EN LA REALIZACION DE UN PROTOTIPO.

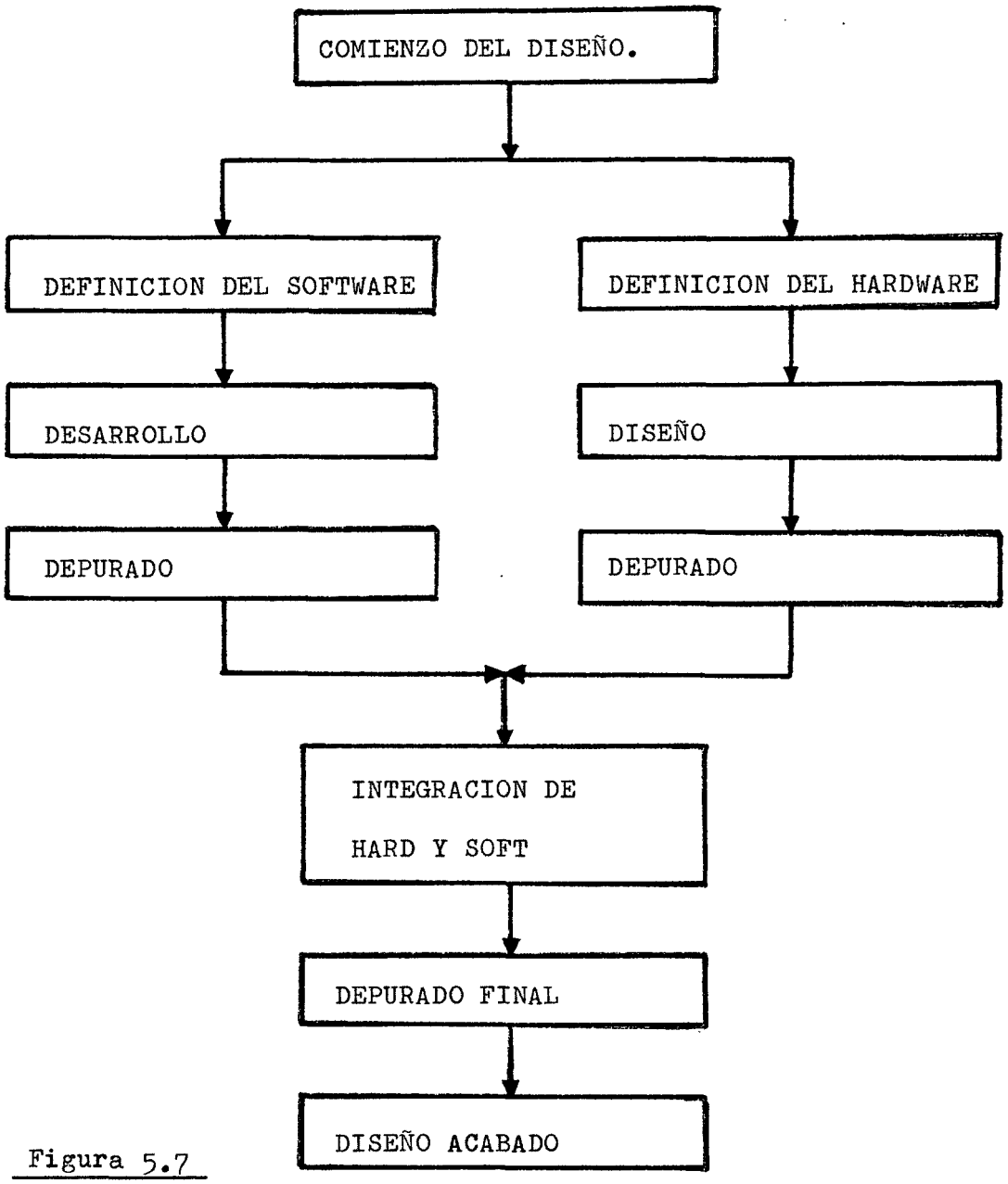


Figura 5.7

La figura anterior, muestra un organigrama que indica la secuencia de tarbajo a seguir para la realización de un desarrollo de un sistema basado en un microprocesador.

Centrandonos en el software, que es objetivo de este estudio, la manera de desarrollar el software viene indicada en la figura 5.8.

El software que se puede desarrollar con el sistema

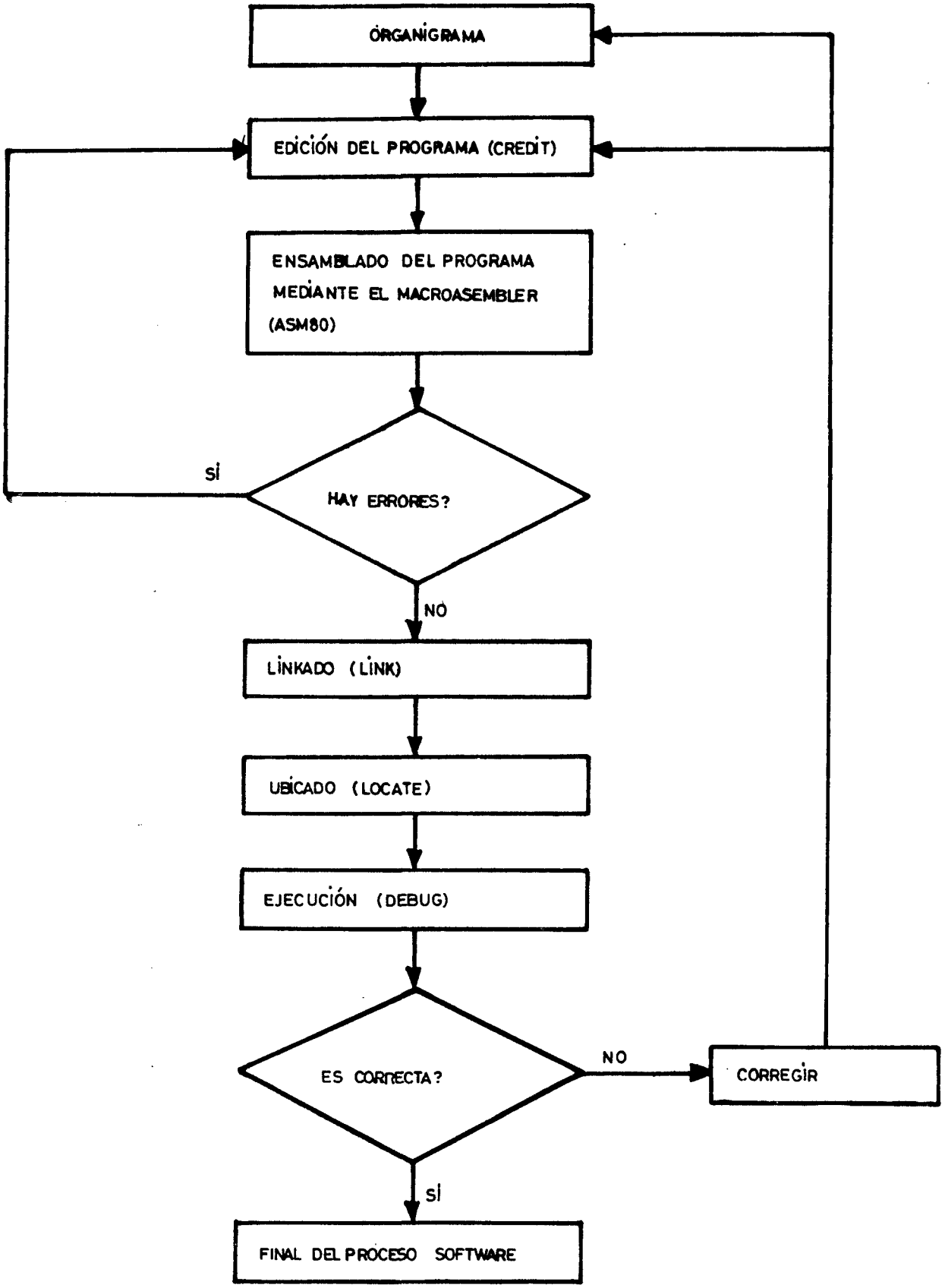


FIGURA 5.8

MDS-221 está basado en el lenguaje ensamblador del uP 8085, en PLM-80, ó en BASIC-80.

El lenguaje de programación usado en el desarrollo de sistemas con poca capacidad suele ser el ensamblador(Pero siempre que no exeda en demasía las mil lineas de programa), y para el programa ejemplo es el que se utiliza.

5.3.- EL PROGRAMA PORTS.

El programa PORTS, es un programa en ensamblador, que se encarga de el control de un horno de proposito general. El control,lo realiza testeando el estado en el que se encuentra un port, el port Ø9 programado en la posición Ø9ØØH, de la memoria PROM del sistema prototipo.Cuando detecta alguna de las condiciones ØØ, Ø1, 1Ø, ó 11 en binario, procede de la manera programada, enfriando, calentando,ó volviendo a comprobar el sistema:

ØØH=ØØ en port Ø9H ------- Temperatura baja
Ø1H=Ø1 en port Ø9H ------- Temperatura en rango
Ø2H=1Ø en port Ø9H ------- Temperatura alta
Ø3H=11 en port Ø9H -------Transductor averiado

Las ordenes de control de temperatura són:

ØØH=ØØ en port ØAH ------- Orden de calentamiento
Ø1H=Ø1 en port ØAH ------- Orden de ventilación
Ø3H=11 en port ØAH ------- Orden de transmisión

El sistema se apoya en un transductor de temperatura que

junto con un conversor A/D, un codificador y un selector de rango de temperaturas constituyen el sistema de lectura y acomodación de datos para el prototipo.Otro sistema, el de control de calentador y ventilador, se encargan de transmitir las ordenes del prototipo a las unidades correspondientes.La descripción de estos sistemas caen fuera del propósito de este proyecto, y solo estudiaremos el capitulo del software del prototipo.

Definidas las especificaciones del producto, el esquema de seguimiento de la figura 5.7, pasa a la definición del software.Para ello comenzamos la realización de los pasos indicados en la figura 5.8 .

1).- ORGANIGRAMA.

En la figura 5.9 se muestar el organigrama del programa PORTS.En él,se describen las distintas situaciones que acontecen al funcionamiento lógico del programa.

2).- EDICION DEL PROGRAMA.

La edición del programa,se efectua mediante el comando CREDIT del sistema operativo ISIS-II.La creación del fichero se realiza con el siguiente comando:

```
-CREDIT PORTS      (Aparecerá en pantalla el mensaje siguiente)
ISIS-II CRT-BASED TEXT EDITOR,V2.Ø
NEW FILE SIZE=ØØ  3Ø5 FREE DISK BLOCKS

---------
```

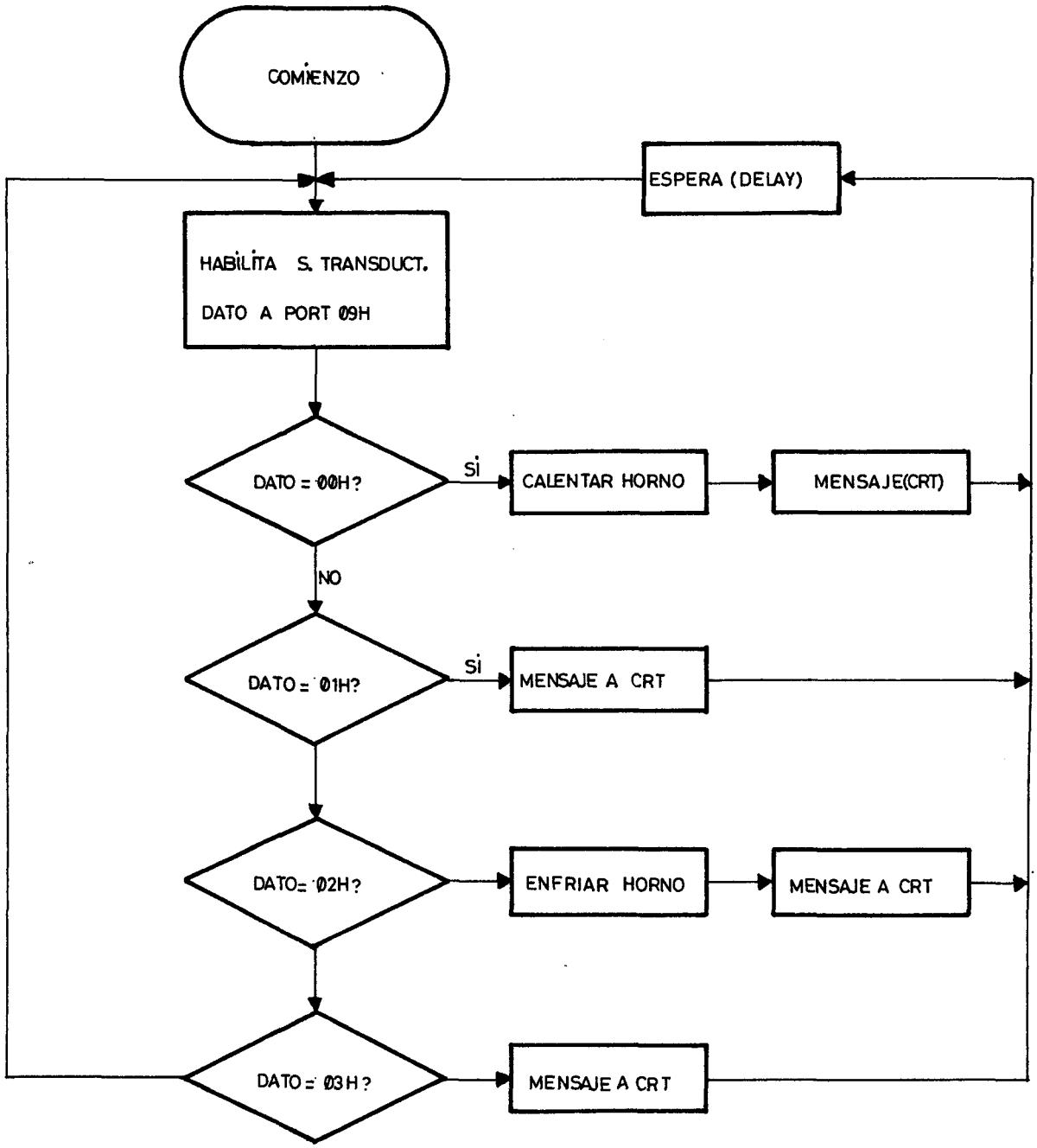


Figura 5.9 .- Organigrama del programa PORTS.

Una vez estamos en la edición, procedemos a la escritura del programa, corrigiendo en caso necesario la edición con comandos ó controles oportunos.

A la hora de describir la localización de el programa PORTS, considero que va a ir cargado en un sistema MDS, puesto que se trata de un ejemplo de desarrollo de software, y por tanto utilizo la Subrutina "CO" de presentación en pantalla de datos.En caso de que quisieramos orto tipo de presentación deberíamos elaborar la subrutina de presentación en pantalla ó dispaly.La programación de prots,se hace a partir de la posición Ø8ØØH, con el port A funcionando como entrada de datos y el port B funcionando como salida.Además considero que el stack, es del tipo relocalizable igual que el código del programa en sí.

Una vez la edición es correcta pasamos al ensanblado del programa con el MACROASSEMBLER.

3).- ENSAMBLADO DEL PROGRAMA.

El comando de ensamblado del programa PORTS, es el siguiente:

```
-ASM8Ø PORTS XREF PAGELENGTH(4Ø)   (Aparecerá en pantalla:)
ISIS-II 8Ø8Ø/8Ø85 MACRO ASSEMBLER,  V4.Ø
ASSEMBLY COMPLETE,   NO ERRORS
ISIS-II ASSEMBLY SYMBOL CROSS REFERENCE,   V2.1
```

Este comando nos creará trés ficheros:PORTS, PORTS.OBJ y PORTS.LST, como yá ha sido explicado anteriormente.En caso

de que el disco sobre el que realizamos la edición no tuviera el programa del MACRASSEMBLER, procederiamos a su copiado (Del programa PORTS), en un disco que si lo tuviera.

Después de esto copiamos, si procede, el fichero PORTS.OBJ en el disco del sistema operativo para proceder a su linkado.

4).- LINKADO.

Una vez situados con el fichero PORTS.OBJ en el disco del sistema operativo,realizamos la función de linkado:

```
-LINK PORTS.OBJ,SYSTEM.LIB(CO) TO PORTS.LNK MAP PRINT(PFCMAP.LNK)
ISIS-II OBJECT LOCATER  V4.2
```

En el fichero PORTS.LNK quedará guardado el módulo resultante de la operación de linkado con la subrutina "CO", de la libreria de sistema.En el fichero PFCMAO.LNK, quedará guardado el mapa de la operación de linkado, dando informacióm sobre los módulos incluidos y sobre la longitud de los distintos segmentos.

Ahora pasamos a la localización del programa.

5).- LOCALIZADO.

La operación de localizado será:

```
-LOCATE PORTS.LNK TO PORTS.LOC CODE(4ØØØH) STACK(49ØØH)
 MAP PRINT(PFCMAP.LOC)
```

Este comando asume que el fichero PORTS.LOC contendrá

el módulo absoluto del programa PORTS, directamente ejecutable bajo las ordenes del monitor del sistema MDS, para conseguir su deourado en caso de encontrar fallos.

6).- DEPURADO.

Según los comandos descritos en la sección 4.3.1, procederemos a la ejecución del programa, verificando su correcto funcionamiento.De esta manera con el comando:

```
-DEBUG PORTS.LOC
.I386Ø
.G4ØØØ
```

el programa se ejecutará siguiendo el valor que en ese momento tenga el port Ø9H.Para comprobar que se ejecuat otra orden, cambiamos el valor del registro A en el punto donde se produce la entrada del port y mandamos que continue la ejecución del programa,para verificar que se cumple según lo deseado:

```
.M42Ø1
XA=Ø3
.G
```

y de esta manera se ejecutará la sustitución para comprobar cualquiera de las otras subrutinas de mensajes y ordenes.

Cuando todo funciona según lo deseado, se ha terminado el proceso de desarrollo del software.

Ahora se pasaria a la realización del prototipo hardware, y se procederia según la figura 5.7.Una vez realizado el prototipo, seintegran el software y el harware y se produce el test del sistema completo y su emulación con el ICE-85, y así detectar posibles fallos.En caso de encontrarlos se vuelve al principio del proceso y se continua su secuencia de trabajo.

En el apendice se encuentra el listado del programa y los diferentes mapas de linkado y localizado del programa PORTS.

6.- BIBLIOGRAFIA.

--Sistemas de desarrollo para microprocesadores.

August Mayer Pújadas.

--Proceadores programables:El microprocesador

Enrique Mandado.

--A guide to Intellec microcomputer development systems.

Daniel D. McCracken

--Manual de utilización del sistema MDS-221

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V4.0          MODULE   PAGE    1


  LOC  OBJ            LINE         SOURCE STATEMENT

                         1 ;        PROGRAMA PARA EL PROYECTO FIN DE CARRERA.
                         2
                         3 ;        AUTOR: DON FRANCISCO JAVIER AFONSO PALMERO.
                         4
                         5          EXTRN    CO                ;DECLARACION DE SUBRUTINA EXTERNA.
                         6
                         7          CSEG                       ;DECLARACION DE RELOCALIZACION.
                         8
  000D                   9 CR       EQU      0DH
  000A                  10 LF       EQU      0AH
                        11
  0000 3D3D3D3D         12 MENS:    DB       '================================================================='
  0004 3D3D3D3D
  0008 3D3D3D3D
  000C 3D3D3D3D
  0010 3D3D3D3D
  0014 3D3D3D3D
  0018 3D3D3D3D
  001C 3D3D3D3D
  0020 3D3D3D3D
  0024 3D3D3D3D
  0028 3D3D3D3D
  002C 3D3D3D3D
  0030 3D3D3D3D
  0034 3D
  0035 0D               13          DB       CR
  0036 0A               14          DB       LF
  0037 3D202020         15          DB       '=              ESTADO GENERAL DEL SISTEMA               ='
```

```
LOC  OBJ           LINE         SOURCE STATEMENT

003B 20202020
003F 20202020
0043 20455354
0047 41444F20
004B 47454E45
004F 52414C20
0053 44454C20
0057 53495354
005B 454D4120
005F 20202020
0063 20202020
0067 20202020
006B 3D
006C 0D              16         DB       CR
006D 0A              17         DB       LF
006E 3D2D2D2D        18         DB       '=----------------------------------------------------------------------------='
0072 2D2D2D2D
0076 2D2D2D2D
007A 2D2D2D2D
007E 2D2D2D2D
0082 2D2D2D2D
0086 2D2D2D2D
008A 2D2D2D2D
008E 2D2D2D2D
0092 2D2D2D2D
0096 2D2D2D2D
009A 2D2D2D2D
009E 2D2D2D2D
00A2 3D
```

```
LOC  OBJ           LINE         SOURCE STATEMENT

00A3 0D              19           DB       CR
00A4 0A              20           DB       LF
00A5 3D202020        21           DB       '=                                                    ='
00A9 20202020
00AD 20202020
00B1 20202020
00B5 20202020
00B9 20202020
00BD 20202020
00C1 20202020
00C5 20202020
00C9 20202020
00CD 20202020
00D1 20202020
00D5 20202020
00D9 3D
00DA 0D              22           DB       CR
00DB 0A              23           DB       LF
00DC 24              24           DB       '$'
                     25
00DD 3D414C41        26 MENS1:    DB       '=ALARMA: TEMPERATURA POR DEBAJO DE VALORES OPTIMOS.  ='
00E1 5F4D413A
00E5 2054454D
00E9 50455241
00ED 54555241
00F1 20504F52
00F5 20444542
00F9 414A4F20
00FD 44452056
```

```
LOC  OBJ            LINE        SOURCE STATEMENT

0101 414C4F52
0105 4553204F
0109 5054494D
010D 4F532E20
0111 3D
0112 0D                 27          DB      CR
0113 24                 28          DB      '$'
                        29
0114 3D414C41           30 MENS2:   DB      '=ALARMA: TEMPERATURA SUPERIOR A VALORES OPTIMOS.     ='
0118 524D413A
011C 20544548
0120 50455241
0124 54555241
0128 20535550
012C 4552494F
0130 52204120
0134 56414C4F
0138 52455320
013C 4F505449
0140 4D4F532E
0144 20202020
0148 3D
0149 0D                 31          DB      CR
014A 24                 32          DB      '$'
                        33
014B 3D202020           34 MENS3:   DB      '=                TEMPERATURA EN RANGO OPTIMO.       ='
014F 20202020
0153 20202020
0157 20544548
```

```
LOC  OBJ           LINE         SOURCE STATEMENT

015B 50455241
015F 54555241
0163 20454E20
0167 52414E47
016B 4F204F50
016F 54494D4F
0173 2E202020
0177 20202020
017B 20202020
017F 3D
0180 0D              35           DB       CR
0181 24              36           DB       '$'
                     37
0182 3D202020        38 MENS4:    DB       '=          FALLO EN TRANDUCTOR DE TEMPERATURA.          ='
0186 20202020
018A 2046414C
018E 4C4F2045
0192 4E205452
0196 414E4455
019A 43544F52
019E 20444520
01A2 54454D50
01A6 45524154
01AA 5552412E
01AE 20202020
01B2 20202020
01B6 3D
01B7 0D              39           DB       CR
01B8 24              40           DB       '$'
```

```
LOC  OBJ            LINE         SOURCE STATEMENT

                      41
01B9 3D202020         42 MENSS:  DB       '=                                    ='
01BD 20202020
01C1 20202020
01C5 20202020
01C9 20202020
01CD 20202020
01D1 20202020
01D5 20202020
01D9 20202020
01DD 20202020
01E1 20202020
01E5 20202020
01E9 20202020
01ED 3D
01EE 0D               43           DB       CR
01EF 24               44           DB       '$'
                      45
01F0 310000    S      46           LXI      SP,STACK
01F3 3E02             47           MVI      A,02H     ;!-- >   PROGRAMACION DE PORTS.SE UTILIZAN A PARTIR DE
01F5 D308             48           OUT      08H       ;!---- > LA POSICION 0800H, QUEDANDO EL PORT A DE EN-
                      49                                ;!-- >   TRADA, Y EL PORT B DE SALIDA.
                      50
01F7 210000    C      51           LXI      H,MENS
01FA CD5D02    C      52           CALL     PRESEN              ;PRESENTACION DEL MENSAJE INICIAL.
01FD 3E03             53 START:  MVI      A,03H
01FF D30A             54           OUT      0AH                 ;MANDA ORDEN DE TRANSMISION (CODIGO 11Y)
0201 DB09             55           IN       09H                 ;COMIENZA A TESTEAR EL ESTADO DEL SISTEMA.
0203 E603             56           ANI      03H
```

```
LOC  OBJ            LINE        SOURCE STATEMENT

0205 FE00             57          CPI     00H
0207 CA1902    C      58          JZ      BAJA
020A FE01             59          CPI     01H
020C CA2C02    C      60          JZ      BIEN
020F FE02             61          CPI     02H
0211 CA3B02    C      62          JZ      ALTA
0214 FE03             63          CPI     03H
0216 CA4E02    C      64          JZ      ERROR
                      65 ;
0219 3E00             66 BAJA:    MVI     A,00H       ;MANDA ORDEN DE CALENTAMIENTO (CODIGO 00Y).
021B D30A             67          OUT     0AH
021D CD7A02    C      68          CALL    CLEAR
0220 21DD00    C      69          LXI     H,MENS1
0223 CD5D02    C      70          CALL    PRESEN
0226 CD6D02    C      71          CALL    DELAY       ;TIEMPO DE ESPERA PARA ACTIVAR EL CALENTADOR.
0229 C3FD01    C      72          JMP     START       ;ESTUDIO DE LA NUEVA SITUACION.
                      73
022C CD7A02    C      74 BIEN:    CALL    CLEAR
022F 214B01    C      75          LXI     H,MENS3
0232 CD5D02    C      76          CALL    PRESEN      ;MENSAJE DE FUNCIONAMIENTO DENTRO DEL RANGO.
0235 CD6D02    C      77          CALL    DELAY
0238 C3FD01    C      78          JMP     START
                      79
023B 3E01             80 ALTA:    MVI     A,01H
023D D30A             81          OUT     0AH         ;MANDA ORDEN DE ENFRIADO (CODIGO 01Y).
023F CD7A02    C      82          CALL    CLEAR
0242 211401    C      83          LXI     H,MENS2
0245 CD5D02    C      84          CALL    PRESEN
0248 CD6D02    C      85          CALL    DELAY       ;TIEMPO DE ESPERA PARA ACTIVAR EL ENFRIADOR.
```

```
LOC  OBJ            LINE     SOURCE STATEMENT

024B C3FD01    C      86            JMP     START           ;ESTUDIO DE LA NUEVA SITUACION.
                      87
024E CD7A02    C      88 ERROR:     CALL    CLEAR
0251 218201    C      89            LXI     H,MENS4
0254 CD5D02    C      90            CALL    PRESEN          ;MENSAJE DE ERROR EN EL TRANSDUCTOR O MEDIDA.
0257 CD6D02    C      91            CALL    DELAY
025A C3FD01    C      92            JMP     START           ;VERIFICACION DEL ESTADO.
                      93
                      94
                      95 ;****** SUBRUTINA DE PRESENTACION EN PANTALLA ******
                      96
025D 00                97 PRESEN:    NOP
025E 7E                98 TRES:      MOV     A,M
025F FE24              99            CPI     '$'
0261 CA6C02    C      100            JZ      FIN
0264 4E               101            MOV     C,M
0265 CD0000    E      102            CALL    CO
0268 23               103            INX     H
0269 C35E02    C      104            JMP     TRES
026C C9               105 FIN:       RET
                     106
                     107 ;****** SUBRUTINA DE ESPERA DE EJECUCION DE LA ORDEN. ******
                     108
026D 16FF             109 DELAY:    MVI     D,0FFH
026F 1EFF             110           MVI     E,0FFH
0271 15               111 UNO:      DCR     D
0272 C27102    C      112           JNZ     UNO
0275 1D               113           DCR     E
0276 C27102    C      114           JNZ     UNO
```

```
  LOC  OBJ            LINE         SOURCE STATEMENT

  0279 C9              115          RET
                       116
                       117 ;****** SUBRUTINA DE LIMPIADO DEL ULTIMO MENSAJE. ******
                       118
  027A 21B901    C     119 CLEAR:   LXI      H,MENS5
  027D 7E              120 DOS:     MOV      A,M
  027E FE24            121          CPI      '$'
  0280 CA8B02    C     122          JZ       FIN1
  0283 4F              123          MOV      C,A
  0284 CD0000    E     124          CALL     CO
  0287 23              125          INX      H
  0288 C37D02    C     126          JMP      DOS
  028B C9              127 FIN1:    RET
                       128          END
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS
CO E 0000

USER SYMBOLS

ALTA	C 023B	BAJA	C 0219	BIEN	C 022C	CLEAR	C 027A	CO	E 0000	CR	A 000D	DELAY	C
DOS	C 027D	ERROR	C 02AE	FIN	C 026C	FIN1	C 028B	LF	A 000A	MENS	C 0000	MENS1	C
MENS2	C 0114	MENS3	C 0143	MENS4	C 0182	MENS5	C 01B9	PRESEN	C 025D	START	C 01FD	TRES	C
UNO	C 0271												

ASSEMBLY COMPLETE, NO ERRORS

```
ALTA      62      80#
BAJA      58      66#
BIEN      60      74#
CLEAR     68      74      82      98      119#
CO         5     102     124
CR         9#     13      16      19      22      27      31      35      39      43
DELAY     71      77      85      91     109#
DOS      120#    126
ERROR     64      88#.
FIN      100     105#
FIN1     122     127#
LF        10#     14      17      20      23
MENS      12#     51
MENS1     26#     69
MENS2     30#     83
MENS3     34#     75
MENS4     38#     89
MENS5     42#    119
PRESEN    52      70      76      84      90      97#
START     53#     72      78      86      92
TRES      98#    104
UNO      111#    112     114
```

CROSS REFERENCE COMPLETE

```
 ISIS-II OBJECT LOCATER V3.0 INVOKED BY:
-LOCATE PORTS.LNK TO PORTS.LOC CODE(4000H) STACK(4900H) MAP PRINT(PFCMAP.LOC)

MEMORY MAP OF MODULE PORTS
READ FROM FILE :F0:PORTS.LNK
WRITTEN TO FILE :F0:PORTS.LOC
MODULE IS NOT A MAIN MODULE

START    STOP LENGTH REL NAME

4000H   4289H    28AH  B  CODE
4900H   490BH      CH  B  STACK
490CH   F6BFH   ADB4H  B  MEMORY




 ASM80 PORTS XREF PAGELENGTH(40)

 ISIS-II OBJECT LINKER V3.0 INVOKED BY:
-LINK PORTS.OBJ,SYSTEM.LIB(CO) TO PORTS.LNK MAP PRINT(PFCMAP.LNK)

LINK MAP OF MODULE PORTS
WRITTEN TO FILE :F0:PORTS.LNK
MODULE IS NOT A MAIN MODULE

SEGMENT INFORMATION:
START    STOP LENGTH REL NAME

                28CH  B  CODE

INPUT MODULES INCLUDED:
 :F0:PORTS.OBJ(MODULE)
 :F0:SYSTEM.LIB(CO)
```

Code			Char.	Code			Char.	Code			Char.
Dec.	Hex	Oct.		Dec.	Hex	Oct.		Dec.	Hex	Oct.	
32	20	40	(Blank)	64	40	100	@	96	60	140	`
33	21	41	!	65	41	101	A	97	61	141	a
34	22	42	"	66	42	102	B	98	62	142	b
35	23	43	#	67	43	103	C	99	63	143	c
36	24	44	$	68	44	104	D	100	64	144	d
37	25	45	%	69	45	105	E	101	65	145	e
38	26	46	&	70	46	106	F	102	66	146	f
39	27	47	'	71	47	107	G	103	67	147	g
40	28	50	(	72	48	110	H	104	68	150	h
41	29	51	)	73	49	111	I	105	69	151	i
42	2A	52	*	74	4A	112	J	106	6A	152	j
43	2B	53	+	75	4B	113	K	107	6B	153	k
44	2C	54	,	76	4C	114	L	108	6C	154	l
45	2D	55	-	77	4D	115	M	109	6D	155	m
46	2E	56	.	78	4E	116	N	110	6E	156	n
47	2F	57	/	79	4F	117	O	111	6F	157	o
48	30	60	0	80	50	120	P	112	70	160	p
49	31	61	1	81	51	121	Q	113	71	161	q
50	32	62	2	82	52	122	R	114	72	162	r
51	33	63	3	83	53	123	S	115	73	163	s
52	34	64	4	84	54	124	T	116	74	164	t
53	35	65	5	85	55	125	U	117	75	165	u
54	36	66	6	86	56	126	V	118	76	166	v
55	37	67	7	87	57	127	W	119	77	167	w
56	38	70	8	88	58	130	X	120	78	170	x
57	39	71	9	89	59	131	Y	121	79	171	y
58	3A	72	:	90	5A	132	Z	122	7A	172	z
59	3B	73	;	91	5B	133	↑ ([)	123	7B	173	{
60	3C	74	<	92	5C	134	↓ (\)	124	7C	174	|
61	3D	75	=	93	5D	135	← (])	125	7D	175	}
62	3E	76	>	94	5E	136	→ (↑)	126	7E	176	~
63	3F	77	?	95	5F	137	— (←)	127	7F	177	(Blank)

Code			Char.	Code			Char.
Dec.	Hex	Oct.		Dec.	Hex	Oct.	
160	A0	240	(Blank)	224	E0	340	(Blank)
161	A1	241	è	225	E1	341	
162	A2	242	ç	226	E2	342	
163	A3	243	£	227	E3	343	
164	A4	244	(Blank)	228	E4	344	
165	A5	245	μ	229	E5	345	
166	A6	246	°	230	E6	346	
167	A7	247	▼	231	E7	347	
168	A8	250	┼	232	E8	350	
169	A9	251	§	233	E9	351	
170	AA	252	(Blank)	234	EA	352	
171	AB	253	©	235	EB	353	
172	AC	254	¼	236	EC	354	
173	AD	255	(Blank)	237	ED	355	
174	AE	256	½	238	EE	356	
175	AF	257	‖	239	EF	357	
176	B0	260	¥	240	F0	360	┌
177	B1	261	Ä	241	F1	361	─
178	B2	262	Ö	242	F2	362	┐
179	B3	263	Ü	243	F3	363	┬
180	B4	264	₵	244	F4	364	├
181	B5	265	(Blank)	245	F5	365	│
182	B6	266	ä	246	F6	366	─
183	B7	267	ö	247	F7	367	┘
184	B8	270	ü	248	F8	370	┴
185	B9	271	ß	249	F9	371	┤
186	BA	272	(Blank)	250	FA	372	┼
187	BB	273	é	251	FB	373	◤
188	BC	274	ù	252	FC	374	◢
189	BD	275	è	353	FD	375	◥
190	BE	276	(Blank)	254	FE	376	◣
191	BF	277	ƒ				