UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

EXPERIENCIAS SOBRE MICROCOMPUTADORAS

T E S I S

que para obtener el título de :

INGENIERO MECANICO ELECTRICISTA
(COMUNICACIONES Y ELECTRONICA)

p r e s e n t a :

HECTOR GUERRERO GUADARRAMA

Ciudad Universitaria, D. F. 1979

CONTENIDO

INTRODUCCION:

La actual revolución industrial y científica, es el resultado de un gran cúmulo de descubrimientos en muchas áreas de la ciencia. La electrónica, es una de las partes que más ha contribuido a este adelanto.

Dentro del campo de la electrónica, los microcomputadores destacan como los dispositivos con mayores perspectivas en aplicaciones comerciales, científicas, industriales, domésticas, etc.

El estudio de éstos, el que siendo parte de la electrónica y computación, debe considerarse como básico, para un mejor desarrollo de nuestra tecnología. Por tanto imprescindible para ello, es el contar con los elementos necesarios para introducirse en esta atractiva materia.

El progreso de los microcomputadores ha sido sorprendente; en el presente, a no muchos años de su aparición, es posible encontrarlos sintetizados en una sola tarjeta de circuito impreso, o en un solo dispositivo integrado, que incluye memoria, puertos de entrada y/o salida, etc. Sin embargo, la estructura de estos elementos es básicamente la misma. De aquí que el estudio de un microcomputador en particular, nos dé los conocimientos fundamentales para la evaluación de estos dispisitivos en general.

- OBJETIVOS PRIMARIOS.

Obtener los conocimientos esenciales acerca de la estructura de un microcomputador, su funcionamiento, modo de operación y aplicaciones. Para lograr este objetivo se propone:

a) Establecer, a partir de sus características particulares, una manera eficiente de aprovechar los elementos que lo integran.

b) Adicionar las terminales, que fueren necesarias, para conformar un sistema de empleo general; y diseñar los acopladores propios de ellas para lograr que sean compatibles al sistema inicial.

c) Evaluar los elementos necesarios para una aplicación particular, tomando como base los que ya están incluidos en el sistema en cuestión (AMI 6800).

- OBJETIVOS SECUNDARIOS.

Proporcionar a las personas interesadas en el tema, unas notas que representan, en gran parte, el reporte de trabajo de la puesta en marcha de una microcomputadora.

Dar un aspecto general de los elementos de programación, necesarios para el desarrollo de rutinas o programas que son o serán empleados en el sistema.

Indicar los pasos a seguir para lograr la implementación de un microcomputador, como soporte fundamental de una aplicación específica.

Estos últimos objetivos sólo serán alcanzados cuando una, o más personas, hagan uso de estos apuntes y logren obtener de ellos, un provecho substancial.

C O N T E N I D O:

En el primer capítulo se proporcionan las caracterísiticas propias de los dispositivos de la FAMILIA S6800 (MC6800), el modo en que pueden ser interconectados y, además, se indica el mínimo de elementos que conforman un sistema básico.

El segundo capítulo versa sobre la estructura del sistema EVK 300 de AMI (AMERICAN MICROSYSTEMS, INC.) y su capacidad en SOFTWARE y HARDWARE.

En el capítulo tres se expone el modo como se logró resolver el problema de la comunicación del sistema con las terminales disponibles.

La intención del capítulo cuatro es dar una introducción al lenguaje ensamblador. Esto se trata de lograr en base a ejemplos; primero construidos mediante un programa fuente y posteriormente trasladarlos al programa objeto.

Las ventajas que ofrecen los programas auxiliares, adicionados a un sistema, son tratadas en el Capítulo Cinco; se hace ver la posibilidad del empleo de una máquina de mayor tamaño, como colaboradora para el mejor aprovechamiento del microcomputador.

El capítulo seis incluye tres ejemplos de aplicación, que proporcionan una mejor visión del manejo y posibilidades de un microcomputador. El problema de la terminal de impresión se vuelve a tratar, pero desde otro punto de vista, ya que ahora su implementación se logra en base a la programación. Se hace una evaluación de los elementos necesarios para una aplicación particular. Por último, se da una introducción a la implementación de un "mini" sistema operativo, en la parte que corresponde al programa ejecutivo.

CAPITULO I

ELEMENTOS DEL SISTEMA

El progreso de los sistemas digitales, para el procesa-- miento de la información, ha estado primordialmente impulsado por los avances tecnológicos desarrollados a partir de mediados del siglo XX; aunado a ello, también la existencia y el - desarrollo de las herramientas matemáticas necesarias para generar potentes dispositivos en aplicaciones generales.

El mejoramiento de las técnicas de fabricación de elementos digitales, ha dado lugar a la aparición de numerosas unidades funcionales integradas que se pueden clasificar en tres grandes grupos:

1.- Dispositivos de baja densidad de integración -SSI- - (compuertas lógicas, biestables).

2.- Dispositivos con densidad media de integración -MSI- (registros, contadores).

3.- Dispositivos de elevada densidad de integración - - -LSI- (memorias, microprocesadores).

Aunque ya se habla de la generación de los circuitos integrados a grandísima escala -VLSI-, cada uno de estos circuitos lógicos estará sintetizado en una pastilla de semiconductor.

Ante el avance acelerado de la Microelectrónica, el desarrollo de las máquinas digitales camina a la par de los dispositivos lógicos, de tal manera que, gran cantidad de sistemas han quedado obsoletos al aparecer en el mercado nuevos dispositivos de circuitos integrados.

Los computadores y demás máquinas digitales de la década de los sesentas, estaban implementados con elementos discre-- tos (diodos, transistores, etc.). De aquí que, el desarrollo se fundara únicamente a partir de la máquina elemental básica:

la compuerta NAND o NOR; no olvidemos que es posible realizar cualquier sistema digital con circuitos lógicos NAND (o NOR)- por tratarse de operadores funcionalmente completos, capaces- de realizar las funciones lógicas más diversas, por muy com-- plejas que éstas sean.

Con la llegada de los circuitos integrados, se simplificó el diseño lógico; en lo que respecta al HARDWARE, ya que - se contó con módulos funcionales especializados como pueden - ser: compuertas lógicas, sumadores, biestables, registros, -- etc.

Por último, en la década de los setentas, se presenta la posibilidad de integrar en un módulo LSI circuitos complejos; consecuentemente, aparece la necesidad de adoptar una nueva - alternativa en la realización de sistemas digitales, como solución al problema de compatibilizar la utilización del alto -- grado de integración, con la construcción de circuitos no limitados en sus posibilidades funcionales.

La facilidad de tener un sistema cuya característica estructural fuera independiente de las funciones posibles a realizar, constituía una solución muy atractiva. Si además, dicho sistema podía realizar cualquier tipo de función, sin hacer modificaciones en su estructura, como en el caso de sistemas digitales de empleo general o computadores, se convertía- en una solución ideal. En la investigación de aproximaciones a dicha solución ideal, aparecieron los microcomputadores.

Un microcomputador es, pues, un sistema digital de utilización general realizada en uno o varios módulos de circuitos integrados, esto es, una máquina formada por una o varias pastillas LSI, cuya función puede ser programada por el diseña-- dor.

El elemento principal de un computador es la unidad central de proceso, de ahí que, el de un microcomputador también lo sea. El microprocesador es la unidad central aritmética y lógica de un microcomputador; junto con sus elementos asociados está reducida a una escala tal, que permite tenerla en -- una sola pastilla de silicio (a veces varias). Una pastilla- de microprocesador típica mide medio centímetro de lado y contiene varias decenas de miles de transistores, resistencias y otros elementos del circuito.

Como la unidad central de proceso (CPU, del inglés Central Processing Unit) de un gran computador, el trabajo del - microprocesador consiste en: admitir datos en forma de cadena de dígitos binarios (ceros y unos); almacenarlos en memoria - para su procesamiento futuro; realizar operaciones aritméti-- cas y lógicas con los datos, de acuerdo con instrucciones pre

viamente almacenadas, y, entregar los resultados para que, a-través de un puerto, sea mostrada la información al usuario;-dicho puerto puede ser un panel con dispositivos luminosos, -un teletipo, la pantalla de un monitor de televisión y/o un -graficador.

Los sistemas modernos de procesamiento de la información y de control, precisan de un rápido almacenamiento y recuperación de la información digital. Las operaciones de almacenaje ("escribir") y la recuperación ("leer"), se encuentran por completo bajo control electrónico.

Existen varios tipos de memoria, su utilización dentro -de un sistema queda definido por la aplicación que se trate.-Las memorias más empleadas son las de lectura-escritura (RWM-READ/WRITE MEMORY-O-RAM-RANDOM ACCESS MEMORY-). Algunas aplicaciones requieren de memorias que contengan información almacenada de modo permanente, o que se altere pocas veces. Tal-almacenamiento lo proporcionan las memorias sólo de lectura -(ROM, del inglés "READ ONLY MEMORY). La información se sitúa en la matriz de almacenamiento cuando se fabrica la pastilla-o "CHIP". Como alternativa para formar este tipo de conexiones en la fabricación inicial de la memoria, las células pueden fabricarse con una pequeña conexión fusible. Para ubicar luego en la matriz una configuración de información dada, basta con aplicar un conjunto de señales eléctricas lo suficientemente intensas como para destruir las conexiones no deseadas. Estas memorias programables de únicamente lectura (PROM, del inglés "Programmable Read Only Memory") se encuentran en-varias versiones, como pueden ser: las memorias eléctricamente alterables sólo de lectura (EAROM, del inglés "Electrically Alterable Read Only Memory"), en la que se puede modificar cierta sección de la memoria y, volver a programar la misma. O también, la alterable por medios ópticos, es decir, exponer la pastilla a rayos ultra-violeta (AROM, del inglés "Alterable Read Only Memory").

Existen dispositivos con los que se facilita la comunicación Máquinas-usuario, éstos son de empleo un poco más específico, es decir, sólo son auxiliares del sistema, dado que el-nivel de integración de las memorias y del procesador es alto, era necesario que estas partes tuvieran una versatilidad afín.

Estos dispositivos, llamados acopladores de entrada y/o-salida (I/O), consisten en pastillas de circuitos integrados, conteniendo los circuitos lógicos necesarios para hacer una -correspondencia de datos en serie o en paralelo, su denominación depende de la familia de microprocesador que se trate; -más adelante haré hincapié en esto.

El sistema microcomputador AMI EVK contiene, en su estructura, los elementos antes mencionados; estos dispositivos están dispuestos de tal manera, que constituyen una máquina de desarrollo y aplicación. Contiene además, circuitos auxiliares como son: un generador de pulsos programable, para obtener de él la velocidad de transmisión requerida para la comunicación en serie con terminales, como un teletipo o una pantalla con teclado; un acoplador (interfase) para conectarse a un teletipo o una terminal, con I/O RS 232; una base de tiempo para procesos de interrupción y algunas características más que mencionaré posteriormente.

El diagrama de bloques de la figura I.1, muestra la composición interna del microcomputador, las líneas indican una conexión entre dispositivos que las incluyen; dichas líneas pueden ser unidireccionales o bidireccionales; en ambos casos, las líneas pueden ser múltiples (BUS).

Los detalles de los circuitos auxiliares se explican en el Capítulo II; atenderemos a éste, las características del microprocesador y de los dispositivos de soporte.

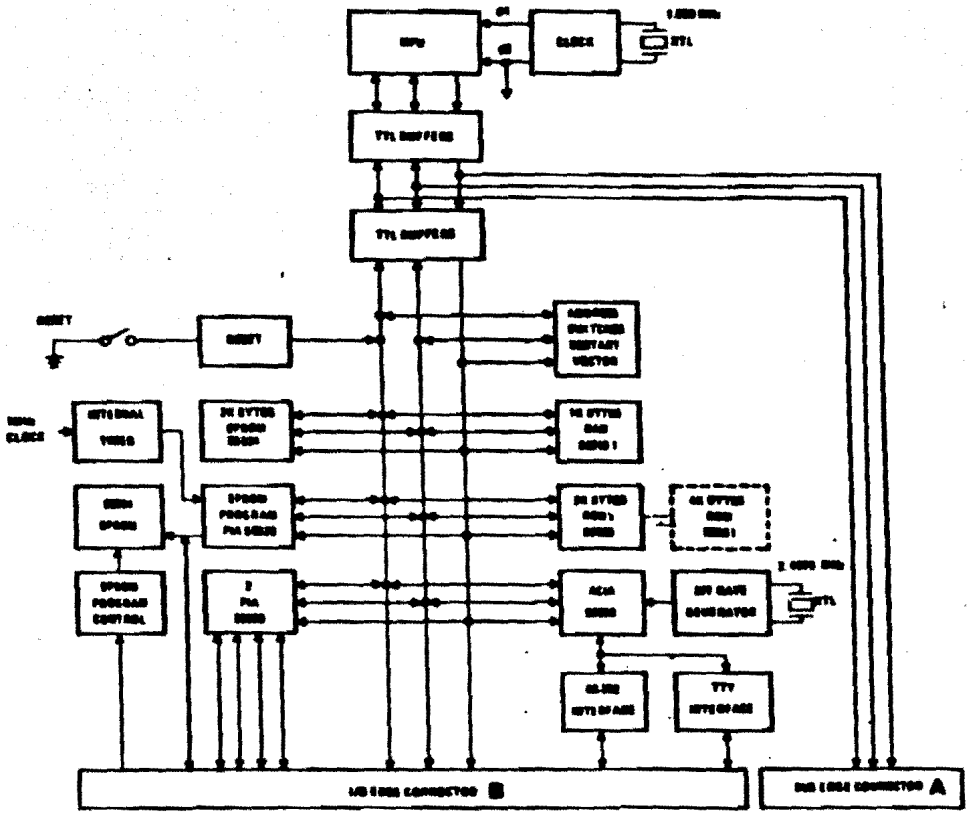


FIGURA I.1.- DIAGRAMA DE BLOQUES DEL SISTEMA AMI S6800.

CAPITULO IA

MICROPROCESADOR S6800

Los implementos de un sistema computador, como son: la memoria, dispositivos de I/O, etc., deberán ser interconectados para formar el sistema. La manera en que los componentes se agrupan y se comunican con cada uno de los otros módulos afectará las características del funcionamiento del sistema.

Una manera eficiente de interconectar los componentes del conjunto es, usando el llamado BUS simple. El BUS consiste en un grupo de líneas conductoras paralelas. Esto se muestra en la figura IA.1a, donde diversas líneas o conductores, que forman el BUS, pasan a través y se conectan a un mínimo de unidades o módulos. En general, cada módulo puede leer del BUS o escribir en él.

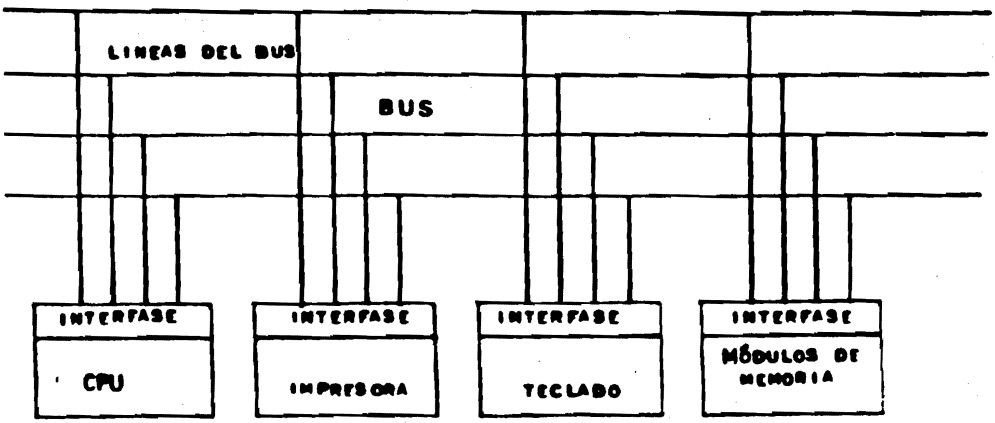
Este BUS se subdivide en tres distintos:

1.- El BUS de datos.

2.- El BUS de direcciones.

3,- El BUS de control.

El BUS de datos es un grupo de ocho líneas bidireccionales que facilitan el flujo de información a todas las partes del sistema. El BUS de direcciones implica dieciseis líneas, que se conectan de acuerdo a las necesidades del conjunto. El BUS de control, permite el manejo de los dispositivos de I/O y memoria.



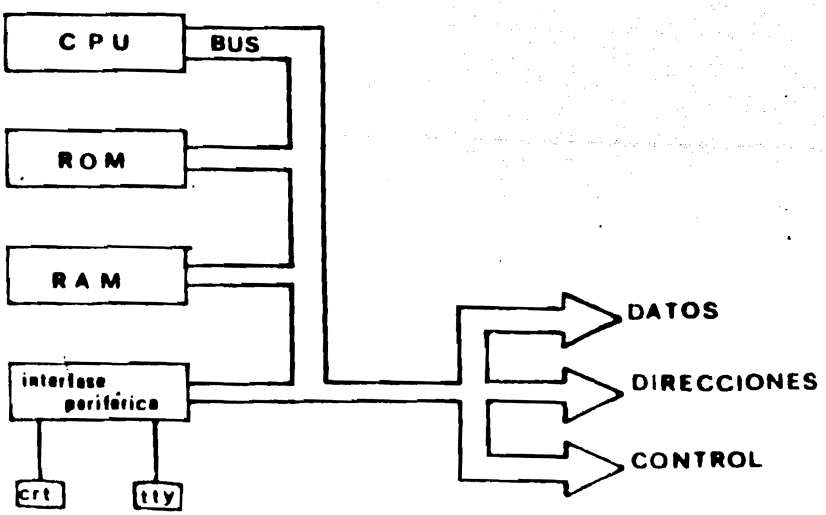


FIGURA IA.1a.- ORGANIZACION DEL COMPUTADOR USANDO UN BUS SIMPLE.

El BUS de control consiste en una mezcla heterogénea de-señales que regulan la operación del sistema. De la figura - IA.1b, podemos mencionar brevemente las características de ca da una de estas líneas de control. Ø2 es una de las fases del reloj aplicadas al MPU. Es empleada para seleccionar o habilitar la entrada de algún dispositivo determinado, asegurando que el elemento elegido es habilitado sólo cuando el BUS de - direcciones y la línea de dirección válida de memoria (VMA) - son estables.

La línea de $\overline{\text{RESET}}$ es usada para poner al sistema en una-situación inicial prescrita (RESET) e iniciar al MPU (START)-desde una condición de encendido. La señal de $\overline{\text{INTERRUPT}}$ - $\overline{\text{REQUEST}}$ ($\overline{\text{IRQ}}$) es generada por la PIA, ACIA o por el usuario - que solicita al MPU que le atienda.

READ/WRITE (R/W) y VMA son salidas del MPU, caracterizan do al BUS de datos y al BUS de direcciones, respectivamente.- R/W indica si el MPU está en el modo de lectura o escritura - para cada ciclo. VMA indica a la memoria e I/O que el MPU está ejecutando una operación de lectura o escritura en un ci-- clo dado. Esta señal es empleada para seleccionar o habili-- tar la entrada del dispositivo seleccionado, de tal manera -- que, inhabilita la transferencia de datos cuando VMA es nivel cero.

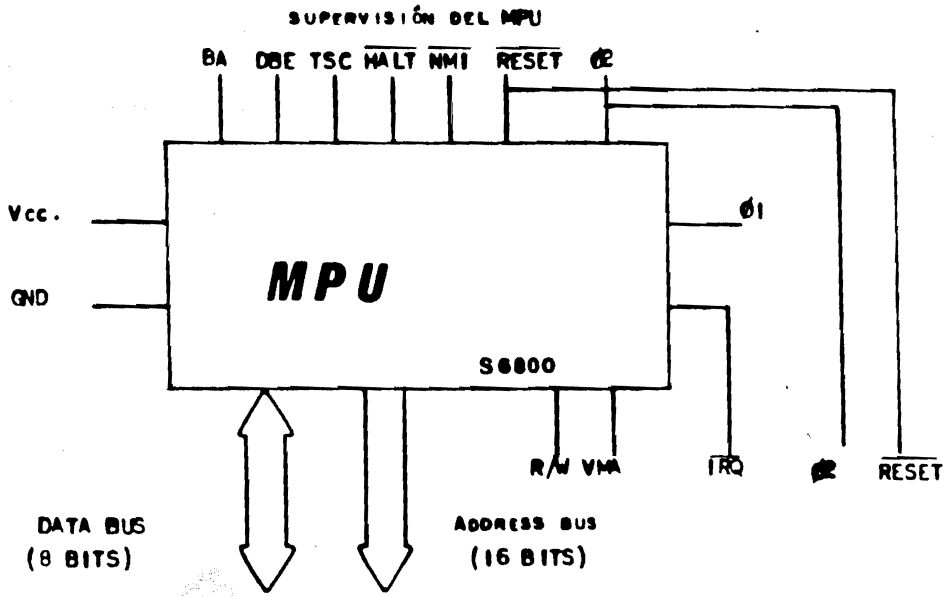


FIGURA IA.1b.- "BUSES" DATOS, DIRECCIONES Y CONTROL DEL MPU S6800.

El grupo de señales, supervisión del MPU, son usadas para controlar al MPU; tres de ellas son compartidas con el BUS de control y afectan, tanto a la memoria como a los dispositivos de I/O.

Ø1 es una de las fases de reloj para el MPU. NON-MASKABLE INTERRUPT (NMI) es similar a la entrada de solicitud de interrupción (IRQ), mencionada anteriormente, excepto que NMI ignora el contenido del bit de bandera de interrupción (I), referida posteriormente con más detalle. DATA BUS ENABLE (DBE) es la señal de control que maneja el estado que guarda el BUS de datos, es decir, si los "BUFFERS" se encuentran en el modo de alta impedancia a la salida. Generalmente, esta señal es dada por Ø2, una de las fases del reloj. THREE STATE CONTROL (TSC) maneja al BUS de direcciones de manera similar que DBE al BUS de datos. Esta señal puede ser usada, verbigracia, cuando se desea usar el modo de alta impedancia a la salida del BUS de direcciones y la señal de R/W, para hacer un acceso directo a la memoria (DMA). La señal supervisora restante es el alto (HALT). Cuando HALT es nivel cero el MPU detendrá el proceso. En el modo de HALT todas las líneas con lógica de tres estados, estarán en el modo de alta impedancia (direcciones, datos y R/W) VMA será nivel cero y la señal de BUS disponible (BA) será nivel uno.

La señal, salida del MPU, BA es normalmente nivel cero. Siendo nivel uno cuando ocurre un nivel cero en la entrada HALT o por la ejecución de una instrucción de WAIT. En uno u otro caso, el MPU detendrá la ejecución del programa y fijará a nivel uno a la señal de BA, indicando que los BUFFERS de tres estados están en el modo de alta impedancia. Si el MPU es detenido como resultado de una instrucción de WAIT, éste esperará hasta que una señal de interrupción sea dada, en tanto BA permanecerá activa mientras una señal de interrupción no enmascarada (NMI) o una señal de solicitud de interrupción (IRQ), ocurran. BA puede ser usada como una señal externa que indica que el MPU está fuera de función para aplicaciones de multiprogramación o acceso directo a memoria (DMA).

La arquitectura del microprocesador incluye registros y unidades de control que lo hacen tener las funciones de una máquina digital de empleo general. El CPU engloba seis registros accesibles por programa. A saber: 1, 2. Acumulador A y B (A,B) (ACCA, ACCB).

El acumulador es un registro que guarda un operando, pudiendo realizar diversas operaciones aritmêticas y/o lógicas que incluyen aquel operando y (cuando corresponde) otro operando; algunas veces, el resultado de la operación queda en el acumulador sustituyendo al operando original. Su tarea in

cluye ser depósito temporal de datos que son trasladados de - memoria para ser almacenados en alguna otra localidad. El -- S6800 contiene dos acumuladores: A y B, de ocho bits (1 BYTE) cada uno. Ver figura IA.2

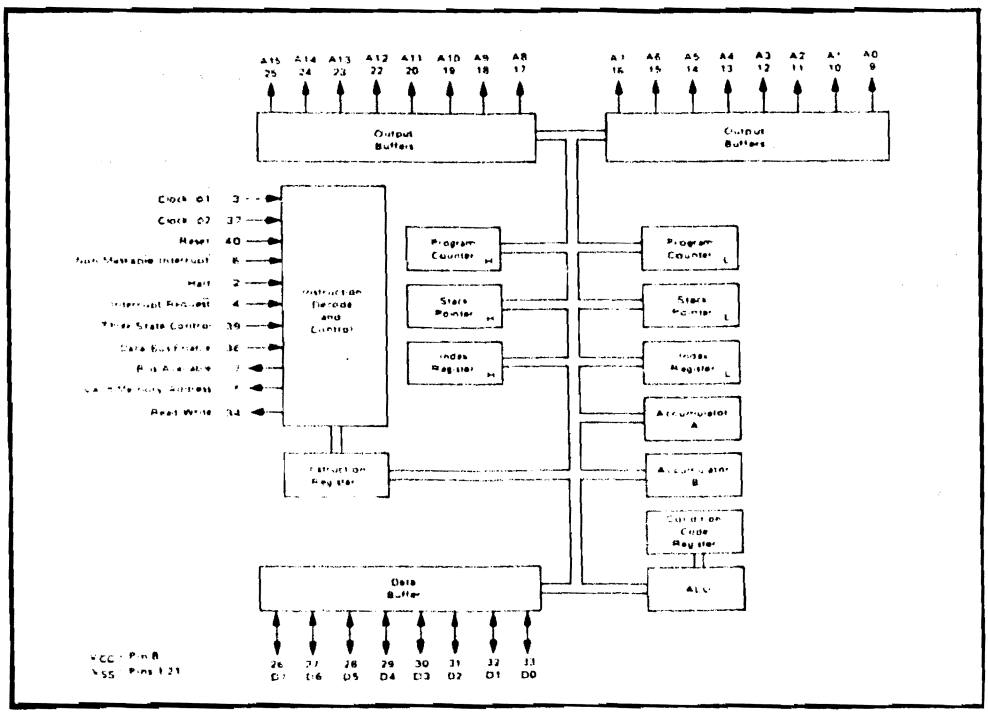


FIGURA IA.2.- DIAGRAMA QUE MUESTRA LOS REGISTROS DEL MPU, ADEMAS, DE LOS BUSES DE DATOS, DIRECCIONES Y CONTROL.

3.- Contador de Programa (P) (PC)

El contador de programa es un dispositivo que registra - la localización de almacenamiento de la palabra de instruc- - ción, se acciona al seguir la palabra de instrucción en uso - vigente. El contador de programa puede seleccionar la locali zación de almacenamiento en orden de secuencia, obteniendo -- así, la siguiente palabra de instrucción a partir de la ubica ción de almacenamiento subsecuente, a menos que el CPU encuen tre otra institución especial o de transferencia. La direc-- ción de la instrucción inmediata que ha de realizarse se en-- cierra en un registro de diez y seis BITS.

4.- Registro Indice (X)

Un registro cuyo contenido se puede sumar o restar de -- una dirección, antes o durante la ejecución de una instruc- - ción. El registro índice permite la modificación automática- de la dirección referida, sin alterar las instrucciones alma- cenadas en la memoria.

	DIRECCION	CONTENIDO
a) X	0000	CE
	0001	FB
	0002	01
b) X	0003	7C
	0004	9A

a) Contenido en X 0000
b) Contenido en X 0003

FIGURA IA.3.- UTILIZACION DEL REGISTRO INDICE -X-.

Como ejemplo, podemos citar el siguiente: siendo los con tenidos, datos arbitrarios, en el inciso a), de la figura - - IA.3, el contenido del registro índice es (0000)h; de ahí que, cuando nos referimos a la información almacenada en la direc- ción del registro índice, hablemos del contenido (CE)h y, - - cuando hacemos una relación al contenido en la dirección del- registro índice más 1 (+1), hablemos del contenido (FB)h. En el inciso b), se realizó alguna operación en el registro X, - de tal manera que, el contenido del registro X es ahora - - (0003)h, de modo que, cuando nos referimos al contenido de la información almacenada en la dirección del registro índice --

(0,X), hablemos del contenido (7C)h, y cuando sumemos una diferencia de 1 (1, X), el contenido será (9A)h, y así sucesivamente. En cualquier caso, no damos como dato la dirección contenida en el registro índice, sino que es la diferencia entre la dirección contenida en el registro índice y la dirección que contiene la información a la que nos referimos.

5.- Apuntador de STACK (s) (SP)

Un STACK estriba en registros de memoria (localizados en RAM) que son direccionados automáticamente durante el llamado o retorno de una subrutina. Otro punto de vista es, que consiste en una serie de artículos, construída y mantenida de tal forma, que el siguiente contenido a sacar es, siempre, el que se puso recientemente en la lista. Funciona como una memoria, "el último en entrar es el primero en salir" -LAST IN-FIRST OUT-, o sea, la llamada memoria LIFO.

Cuando un BYTE de información es almacenado en el STACK, éste se almacenará en la dirección contenida en el apuntador de STACK (STACK POINTER). El STACK POINTER es decrementado (en uno), inmediatamente después de almacenar en el STACK un BYTE de información. Asimismo, el STACK POINTER es incrementado (en uno) antes de recuperar un BYTE de información proveniente del STACK y, el BYTE es obtenido desde la dirección contenida en el STACK POINTER. (16 BITS).

La operación del STACK POINTER, por medio de las instrucciones de recuperar (PULL) o almacenar (PUSH), mencionados en el párrafo anterior, es ilustrado en la figura IA.4 e IA.5. Estas instrucciones pueden estar referidas al acumulador A ó al B; este ejemplo se refiere al A.

La instrucción PUSH (PSH A), empujar o almacenar el contenido del acumulador A a la localidad dada por el STACK POINTER-, genera la secuencia mostrada en la figura IA.4. El inciso a) indica el estado del acumulador A, el del contador de programa y el del STACK POINTER antes de ejecutar la instrucción PSH A. Supongamos que el registro SP contiene la dirección M, el contenido del acumulador A, sea 7C)h y el contador de programa indica la dirección que contiene la instrucción PSH A. El inciso b) muestra los mismos dispositivos pero, en el estado en que se hallan después de la operación PSH A. El apuntador de STACK se ha decrementado en uno, el contenido de la dirección M es ocupado por el contenido del acumulador A y, el contador de programa contiene la dirección de la siguiente instrucción a ejecutar.

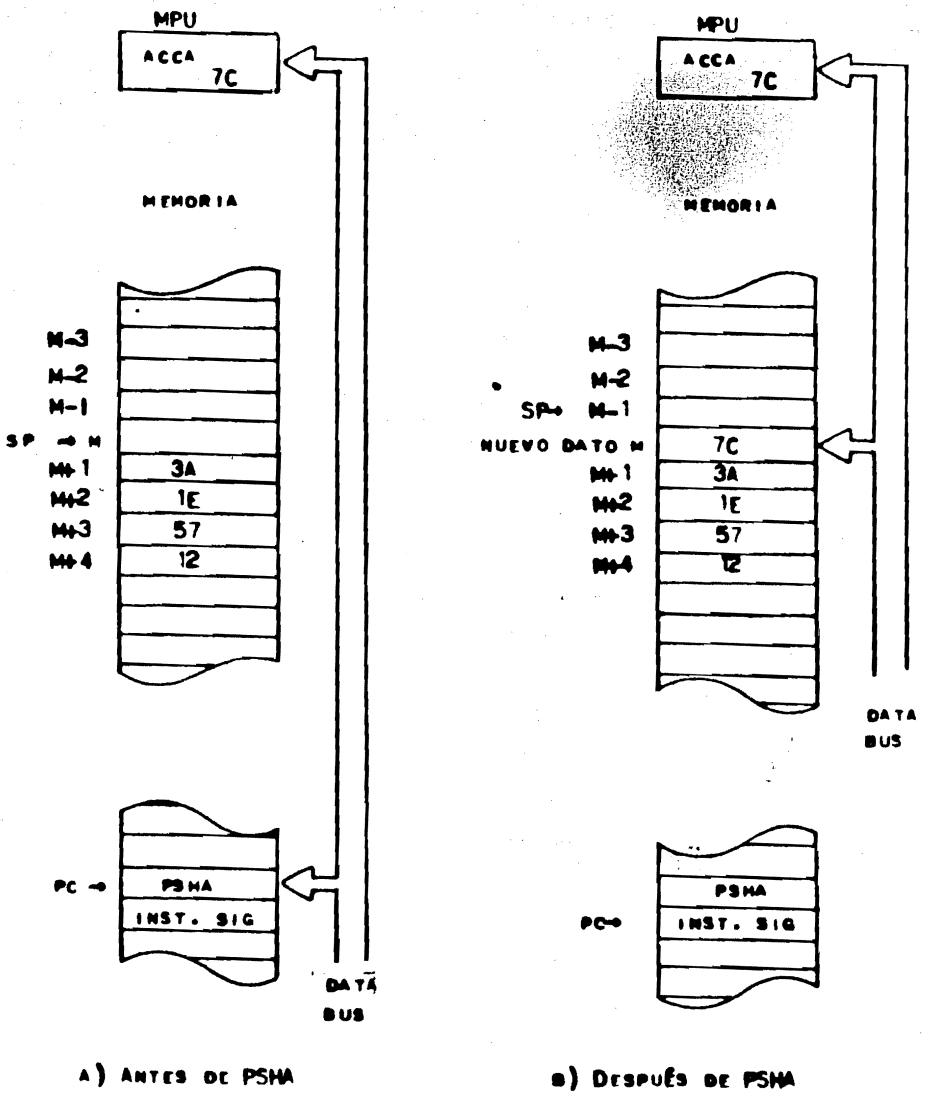


FIGURA IA.4.- SECUENCIA DE LA INSTRUCCION PUSH.

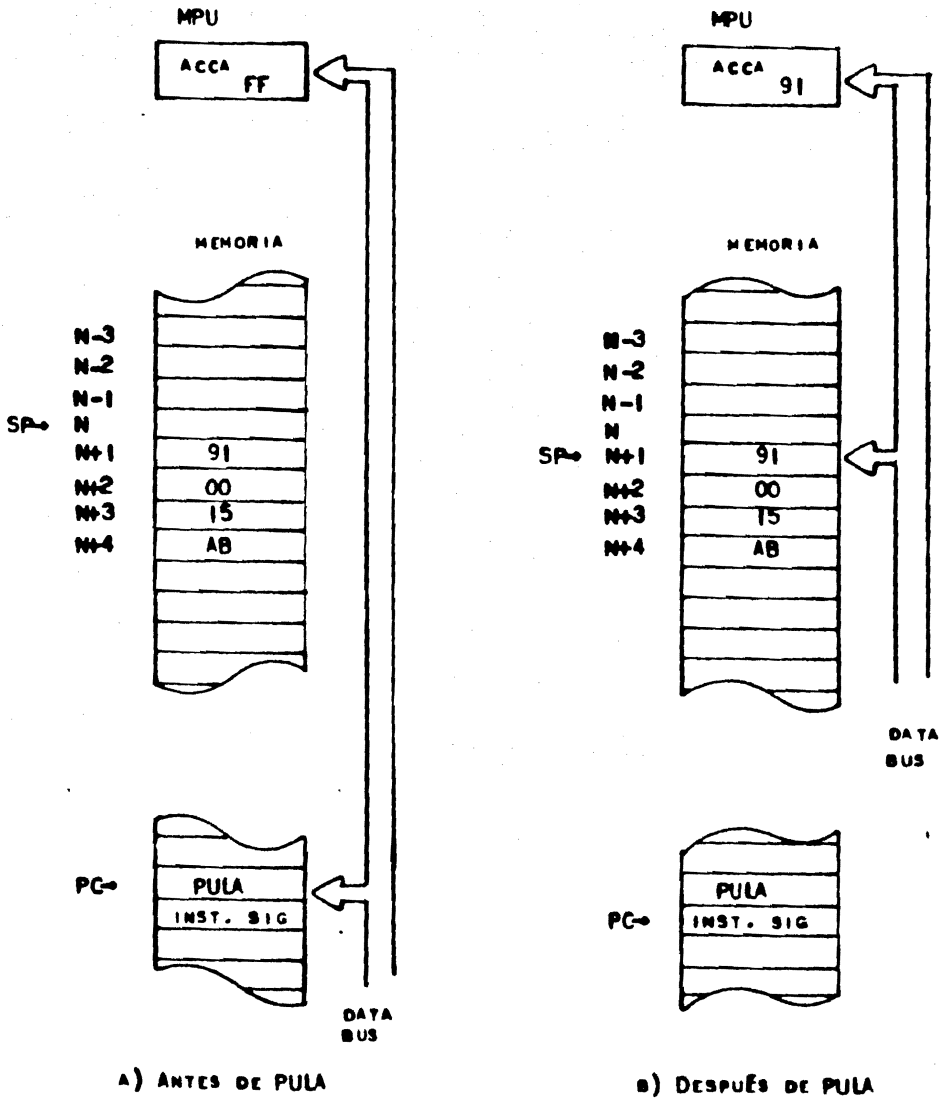


FIGURA IA.5.- SECUENCIA DE LA INSTRUCCION PULL.

La instrucción PULL (PUL A) hace el proceso inverso al - de la instruccipon PSH A. La figura IA.5 muestra su evolu- - ción. En el inciso a) se expone el estado del acumulador A,-

el contador de programa y el apuntador de STACK. En este caso, supongamos que el apuntador contiene la dirección N, el - contenido del acumulador A es FF)h y el contador de programa- indica la localidad que contiene la instrucción PUL A. La -- condición que guardan los dispositivos después de ejecutar la instrucción PUL A, se indica en el inciso b) de la figura IA. 5. El STACK POINTER se ha incrementado en uno, el acumulador ahora contiene la información que indica el apuntador y el -- contador de programa encierra la dirección de la siguiente -- instrucción a ejecutar.

6.- Registro de Condición (CC) (CCR)

Este registro contiene ocho BITS seis de los cuales pueden ser afectados automáticamente por operaciones efectuadas- en el MPU, o también por instrucciones propias de este registro. Los dos bits restantes estarán fijos a nivel uno. La - figura IA.6 indica de qué manera está constituido el registro de condición.

1	1	H	I	N	Z	V	C
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

FIGURA IA.6.- REGISTRO DE CONDICION.

C.- CARRY; es nivel uno si hay un acarreo desde el bit - más significativo del resultado de alguna operación. De otra manera es nivel cero.

V.- OVERFLOW; es nivel uno si ocurre un exceso de la capacidad del registro como resultado de alguna operación. Delo contrario es nivel cero.

Z.- ZERO; es nivel uno cuando el resultado de alguna operación es igual a cero. De otra manera es nivel cero.

N.- NEGATIVE; es nivel uno cuando el bit más significativo es nivel uno. Si no es nivel cero.

I.- INTERRUPT MASK; es nivel uno por una operación de - HARDWARE ó SOFTWARE INTERRUPT o por la instrucción SEI; es nivel cero por la ejecución de la instrucción CLI, mostrados en el conjunto de instrucciones del MPU. Es llevado a nivel ce-

ro como resultado de la ejecución de la instrucción RTI; si - I fue almacenado en el STACK con esa condición.

H.- HALF CARRY; este BIT sólo es afectado por las ins- - trucciones ADD, ABA y ADC (señaladas en el conjunto de ins- - trucciones), y será nivel uno cuando haya un acarreo del BIT- 3 al BIT 4 del resultado de la ejecución de alguna de las - - tres instrucciones mencionadas anteriormente. Cuando no ocurre un acarreo del BIT 3 al BIT 4 H será nivel cero.

Los BITS b6 y b7 del registro de condición estarán siempre fijos a nivel uno.

Veamos ahora con más detalle las señales de control que- supervisan la ejecución del MPU.

La figura IA.7 es un compendio de las señales de reloj,- Ø1 y Ø2 requeridas por el S6800. Las señales Ø1 y Ø2 deben - ser complementarias, no traslapadas de 5 volts.

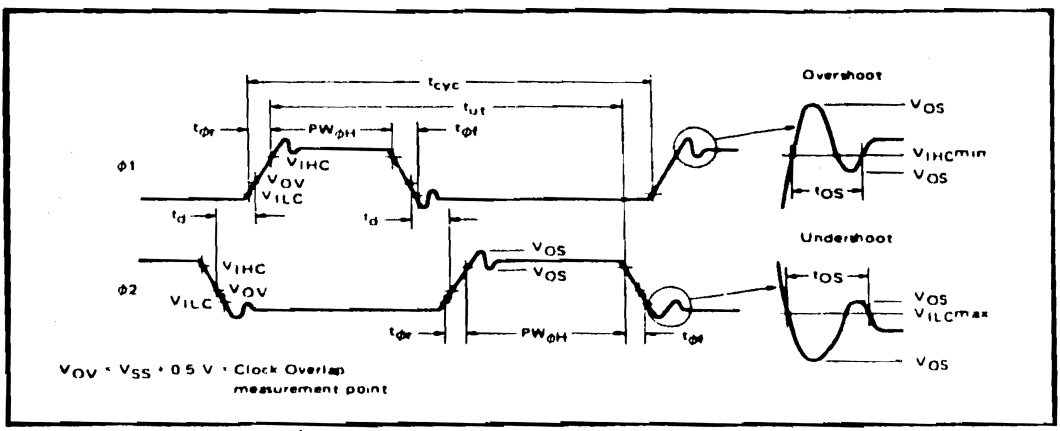


FIGURA IA.7.- FORMA DE ONDA PARA EL RELOJ DEL MPU.

Las características de las señales se dan en el apéndice B.

Como anteriormente señalábamos, el VMA funge como señal-

de control que indica cuando se realiza una operación de lectura o escritura, siendo para este propósito la señal de un nivel uno. El R/W nos dice si hay una operación de lectura, nivel uno, y cuando hay una operación de escritura, nivel cero, desde una localidad de memoria o de algún periférico. La figura IA.8, nos muestra estas dos operaciones.

Es necesario indicar la importancia que tiene la señal VMA. De tal manera que, este control habilita el periférico o memoria sólo cuando se realiza una operación de lectura o escritura. Como se puede ver en la figura IA8.b, el DBE (habilitador del BUS de datos), generalmente Ø2, acciona los BUFFERS del DATA BUS durante el nivel uno de su período, de tal manera que sólo puede escribir durante esta etapa. De aquí que, la mejor manera de asegurar una buena operación de lectura o escritura en memoria o en un periférico es usar la señal VMA en conjunción con Ø2, como muestra la figura IA.9.

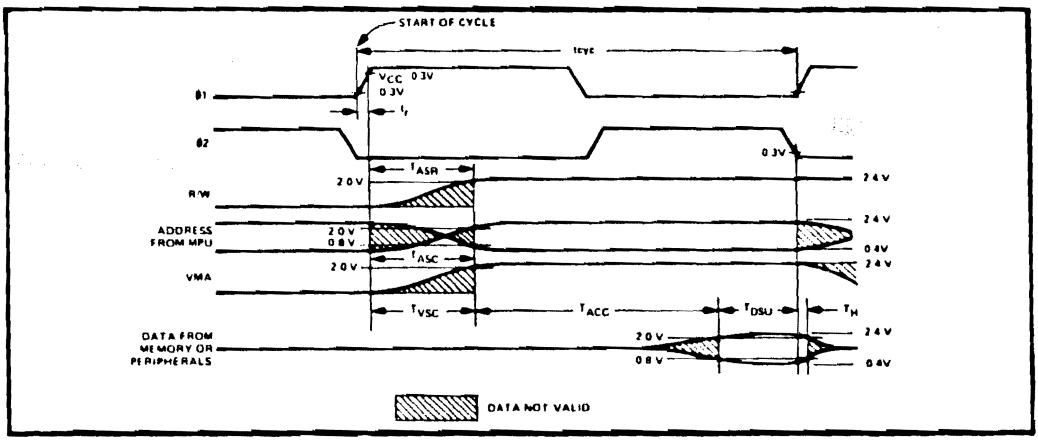


FIGURA IA9.a.- LECTURA DE DATOS DESDE MEMORIA O PERIFERICOS.

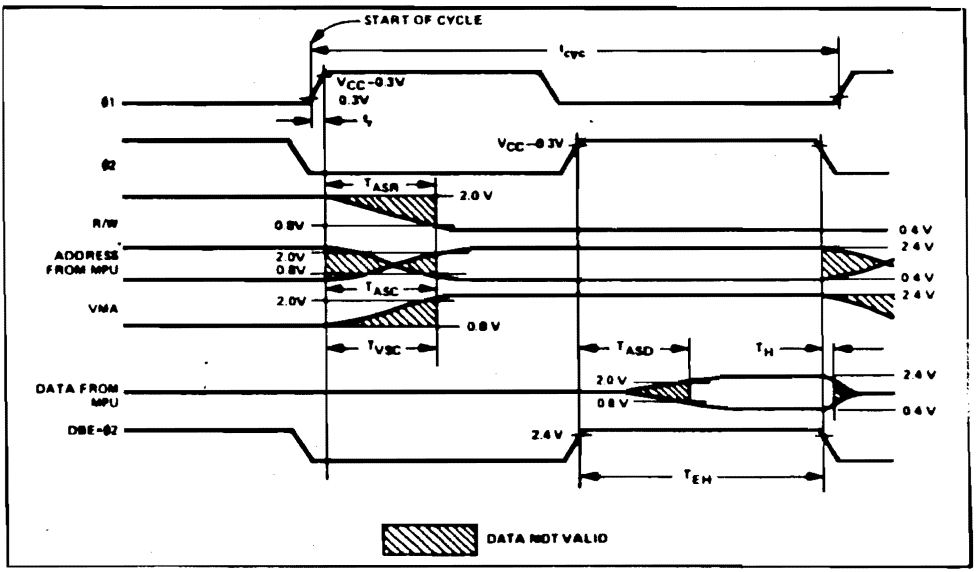


FIGURA IA8.b.- ESCRITURA DE DATOS EN MEMORIA O PERIFERICOS.

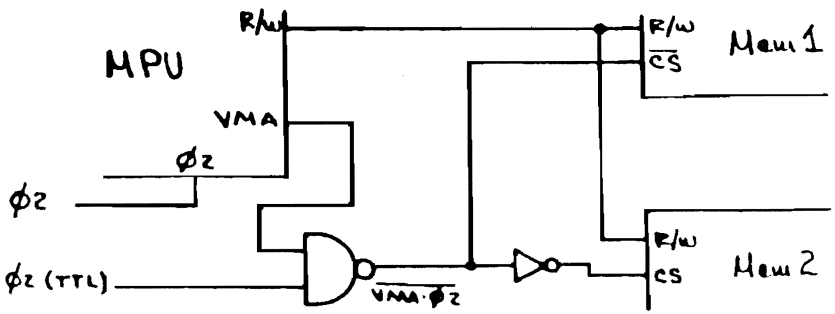


FIGURA IA.9.- MODO DE CONEXION DEL VMA, Ø2 y R/W A LOS PERIFERICOS O DISPOSITIVOS DE MEMORIA.

La entrada CS (CHIP SELECT) o E (ENABLE), habilita al -- dispositivo que lo contiene, de tal manera que en el caso de- Mem.1 es necesario dar la señal negada para que se habilite - cuando VMA y Ø2 son nivel uno, ya que su CS es negado y vice- versa en Mem. 2. La señal de R/W se conecta directamente a - memoria o periférico.

El contenido del registro contador de programa (PC, dentro del MPU), al encender el sistema, o cuando se ha tenido - una falla en el programa ejecutado, es totalmente aleatorio,- de modo que para que el MPU pueda dirigirse a una dirección - específica, necesitamos de una señal de control que logre este propósito. Esta señal es el "RESET"; con este control podemos "mandar" al procesador al inicio de un programa que tam bién fije, si es que así lo deseamos, el contenido del SP, X, ACCA, ACCB, y/o iniciar un ACIA o una PIA, etc. Esta señal - debe ser nivel cero cuando queramos realizar la operación anterior y nivel uno para que el MPU trabaje normalmente. $\overline{\text{RESET}}$ ocasiona que el MPU inicie una secuencia en la que fija el -- contenido del PC de la siguiente manera:

Al presentarse un frente de onda positivo en la entrada- $\overline{\text{RESET}}$ el MPU comenzará la secuencia de reset. El bit 4 del - registro de condición (I) es llevado a nivel uno, el contenido de las direcciones más altas de memoria (FFFE, FFFF) son - cargadas en el PC, de tal manera que, este registro encierra- ahora la dirección de inicio de un programa o subrutina, este programa recibe el nombre de MONITOR. Cuando RESET es bajo - (teniendo en cuenta que sea un pulso nivel cero, de una duración de ocho ciclos como mínimo), las señales de salida del - MPU tendrán el siguiente estado: VMA y BA nivel cero; R/W nivel uno; DATA BUS en alta impedancia y el ADDRESS BUS contendrá la dirección FFFE. La figura IA.10 ilustra la secuencia- a partir de que ha sido encendida la fuente de poder.

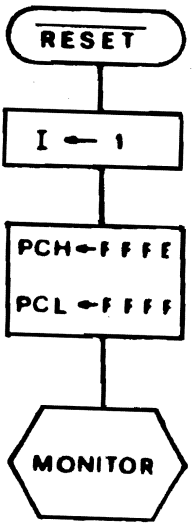


FIGURA IA.10.- SECUENCIA DE LA INTERRUPCION DE RESET.

Además de la señal de interrupción $\overline{\text{RESET}}$, existen tres más, son: $\overline{\text{NMI}}$, $\overline{\text{IRQ}}$ y $\overline{\text{HALT}}$.

Las señales de interrupción $\overline{\text{IRQ}}$ y $\overline{\text{NMI}}$ pueden ser dadas por el usuario manualmente o por programa, éstas pueden ser proporcionadas mediante un interruptor o a través de un periférico como una PIA, ACIA o algún otro dispositivo. Como el $\overline{\text{RESET}}$, el $\overline{\text{IRQ}}$ y $\overline{\text{NMI}}$ generan una secuencia definida.

Ambas señales deberán ser nivel cero para que ocurra la secuencia. En el caso de $\overline{\text{IRQ}}$ (solicitud de interrupción por HARDWARE), el bit 4 del registro de condición (CCR) es censado para aceptar o no la interrupción, es decir, después de finalizar la instrucción corriente y probar el estado del bit $\overline{\text{I}}$ del CCR, y si éste es nivel cero, el MPU almacenará el contenido de los registros internos (CCR, ACCB, ACCA, X y PC) en las localidades de memoria dadas por el SP. Este almacenamiento llena siete ciclos de escritura en memoria: dos para

cada uno de los registros PC y X y uno para cada uno de los registros ACCA, ACCB y CCR. El SP será decrementado en siete y estará apuntando a la siguiente dirección disponible.

Posterior a esto, el MPU fijará el bit I a nivel uno para que así no ocurra otra interrupción mientras no se haya entrado al programa solicitado por el $\overline{IRQ}$, inmediatamente después, el PC almacenará el contenido de las direcciones FFF8)h y FFF9)h para más tarde, saltar al inicio del programa definido por la dirección encerrada en el PC. Esta secuencia es mostrada en la figura IA.11.

La secuencia de $\overline{NMI}$ (interrupción no enmascarada) es similar a la de $\overline{IRQ}$, sólo que ésta no censa el contenido de I.- Después de haber terminado la ejecución de la instrucción corriente, el MPU almacena el contenido de los registros internos, mencionados en $\overline{IRQ}$, lleva a nivel uno el bit I y obtiene la dirección de inicio de la rutina de servicio del $\overline{NMI}$, esta dirección está contenida en las localidades FFFC)h y FFFD)h.- La figura IA.12 muestra la secuencia que sigue de la señal de $\overline{NMI}$.

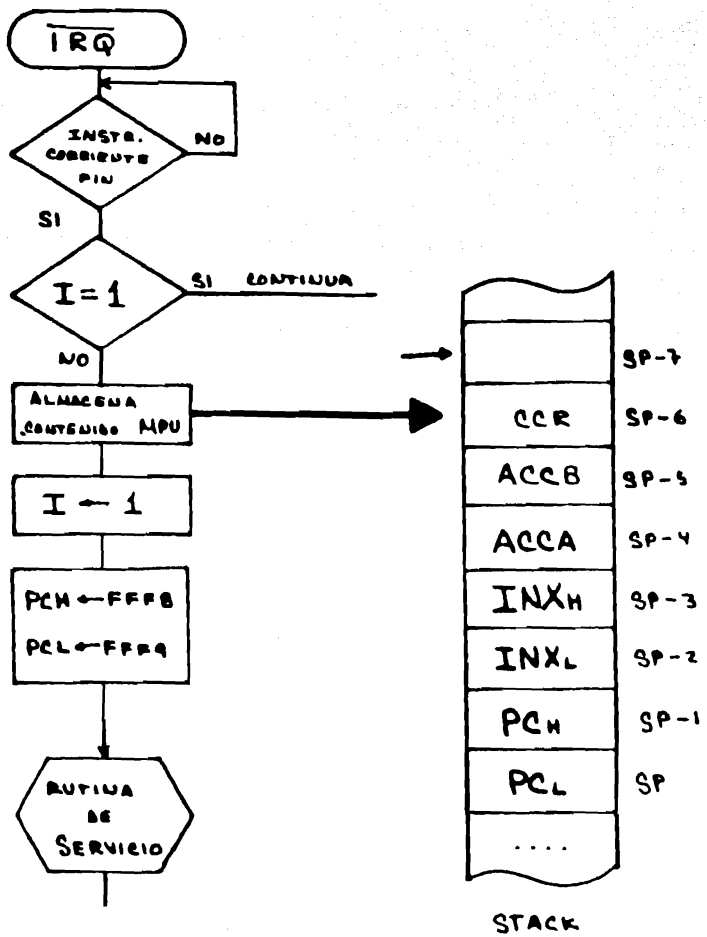


FIGURA IA.11.- SECUENCIA DEL $\overline{IRQ}$

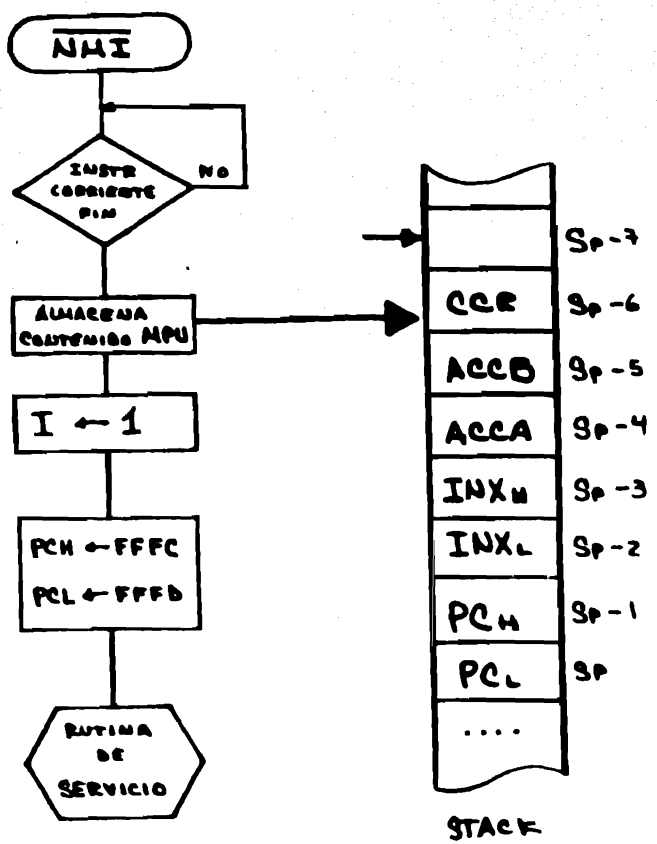


FIGURA IA.12.- SECUENCIA DEL $\overline{NMI}$

La señal de $\overline{HALT}$ permite la opción de detener la ejecución del MPU, de tal modo que aparenta el haber sido desconec

tado del sistema de BUSES.

Cuando el MPU está en el modo de Halt, toda la actividad del programa es suspendida, y si $\overline{NMI}$ o $\overline{IRQ}$ ocurren durante este período, serán guardados y actuarán tan pronto como el MPU sea sacado del modo de HALT. Si una señal de $\overline{RESET}$ ocurre -- cuando el MPU está detenido, se presentan los siguientes estados en las señales de salida del MPU: VMA y BA, serán nivel - cero; DATA BUS, presentará alta impedancia; R/W, estará en el modo de lectura, nivel uno, y el ADDRESS BUS contendrá la dirección FFFE, mientras el $\overline{RESET}$ sea nivel cero. Tan pronto - como la señal de $\overline{HALT}$ sea nivel uno el MPU obtendrá el contenido de las direcciones FFFE)h y FFFF)h para iniciar la rutina de RESET.

La figura IA.14 muestra el diagrama de tiempo implicado- cuando el MPU está en el modo de HALT. Cuando $\overline{HALT}$ es nivel- cero, el MPU será detenido después de haber completado la ejecución de la instrucción corriente. La transición de $\overline{HALT}$, - de nivel uno a cero, deberá ocurrir tpcs antes de suceder el- frente de onda negativo del ciclo último de Ø1 que completa - el final de la instrucción corriente. BA será nivel uno, una vez transcurrido el tiempo tBA, después del último ciclo de - la instrucción corriente. Para este instante el VMA será nivel cero y R/W, ADDRESS BUS y DATA BUS estarán en el modo de- alta impedancia.

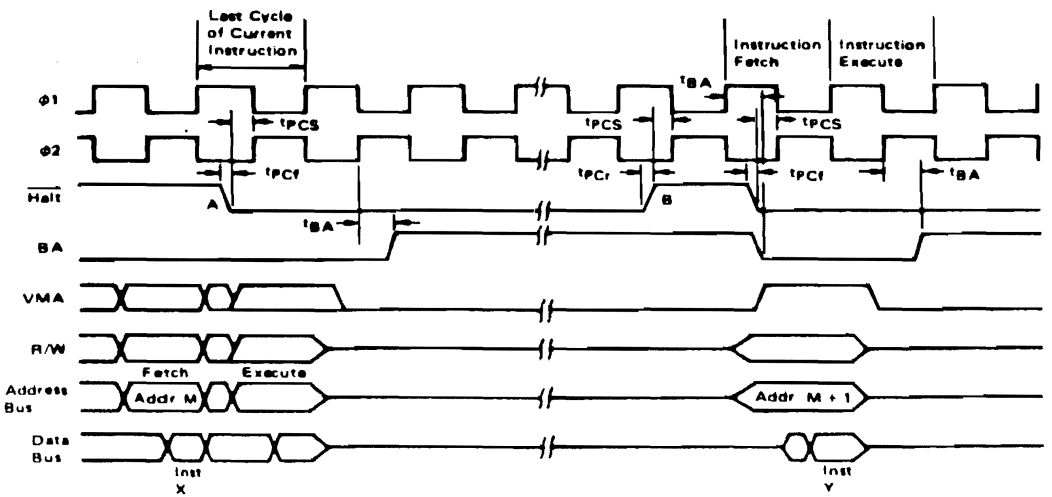


FIGURA Ia.14.- DIAGRAMA DE TIEMPOS QUE IMPLICA EL MODO - DE HALT CUANDO EXISTE LA EJECUCION DE UNA INSTRUCCION CORRIENTE.

El MPU cuenta con una instrucción que realiza una interrupción, ésta es la interrupción por SOFTWARE, SWI. La ejecución de dicha instrucción de SWI inicia la secuencia mostrada en la figura IA.15. La secuencia es similar a la de una interrupción por HARDWARE, excepto que SWI es dado por programa y que las localidades que guardan la dirección de inicio de la rutina de servicio son FFFA)h y FFFB)h.

Esta interrupción puede ser usada para adicionar al programa principal las subrutinas que indiquen la condición que guarda el MPU en el momento de un SWI; estos puntos son llamados puntos de interrupción o BREAK-POINTS.

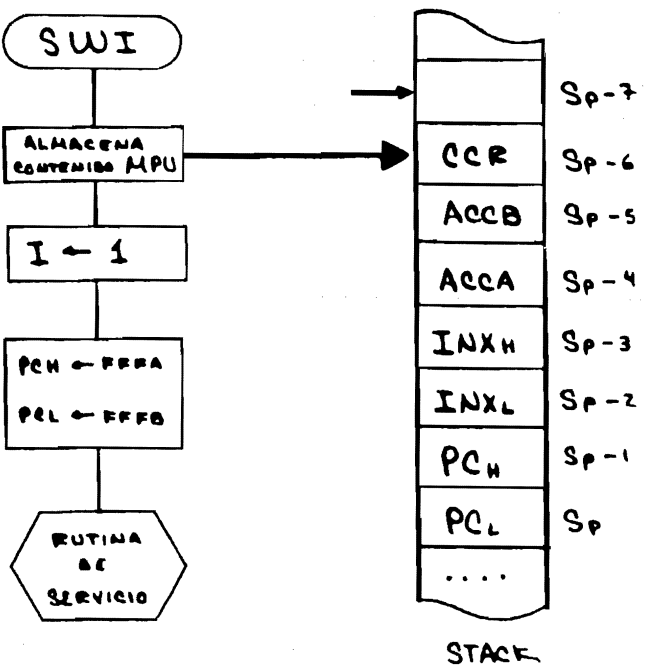


FIGURA IA.15.- SECUENCIA DE SWI.

Las señales de interrupción: $\overline{\text{RESET}}$, $\overline{\text{IRQ}}$ y $\overline{\text{NMI}}$, y la instrucción SWI, generan una secuencia que incluye un salto a una rutina de servicio. Las localidades que contienen las direcciones de inicio de esas rutinas son llamadas vector de interrupción; estas localidades no deben ser ocupadas por algún periférico o algún otro dispositivo diferente a la memoria. La figura IA.16 nos muestra un resumen de este vector de interrupción.

VECTOR		ESPECIFICACION
MS	LS	
FFFE	FFFF	$\overline{\text{RESET}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	$\overline{\text{SWI}}$
FFF8	FFF9	$\overline{\text{IRQ}}$

FIGURA IA.16.- MAPA DE MEMORIA PARA VECTORES DE INTERRUPCION.

El conjunto de instrucciones del MPU S6800 proporciona maneras diferentes de acceder el operando para la función que le ha sido asignada por el código de operación.

El operando puede contener datos que solamente serán llamados para procesarse en el MPU, o también encerrará la dirección de alguna localidad de memoria que contenga la información solicitada. El S6800 es capaz de direccionar hasta 65,536 diferentes localidades de memoria, que requieren 16 líneas de dirección para seleccionar alguna de ellas. De tal manera que, si queremos referirnos a alguna localidad en especial, tendremos que hacerlo utilizando dos BYTES como operando. Pero si la dirección de la localidad que contiene la información requerida está cercana, entonces podremos hacerlo con menos de 16 bits. de dirección.

Cuando la información a la que queremos tener acceso está ordenada de una manera definida y, ha sido almacenada previamente, podemos hacerlo de dos modos distintos, esto gracias a los registros SP y X, no importando lo lejana que esté dicha información, ya que SP y X contienen 16 bits. En el caso de SP, como vimos anteriormente, sólo podemos obtener un byte de información a la vez, para esto, sólo se necesita el código de operación, ya que está implícita la dirección conte

nida en SP.

El otro registro de dirección que puede ser precargado - para tomarse como referencia, es el registro índice (X) y las instrucciones que acceden memoria relativa a X permiten el direccionamiento hasta de 256 localidades, con sólo utilizar un BYTE como operando.

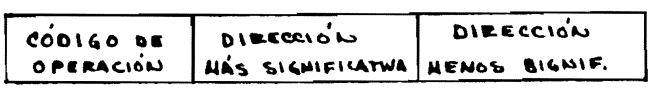
Por otra parte, las ramificaciones y saltos también se - pueden lograr de diferentes maneras. En el caso de ramificaciones (BRANCH), éstas pueden ser condicionadas al estado que guarda el CCR; el contador de programa se verá incrementado o decrementado, según se requiera, en +127 ó - 128 localidades. Necesitándose para esto, un operando de un BYTE. Los saltos- (JUMPS) pueden lograrse a cualquier dirección que se requiera, para esto, se necesitan dos BYTES como operando, o uno si la- dirección a la que se quiera saltar, es relativa al registro- X.

El S6800 cuenta con 72 instrucciones básicas. Las ins-- trucciones pueden ser usadas en diferentes modos de direccionamiento, para "ahorrar" tiempo de ejecución y espacio de memoria. Algunos modos de direccionamiento son disponibles en- una variedad de instrucciones, otros son sólo para una pequeña clase de instrucciones. En muchos casos, donde las ins- - trucciones sólo difieren en el modo de direccionamiento, el - mismo código de operación es usado para ambas y, el modo de - direccionamiento es especificado de otra manera.

Los modos de direccionamiento se pueden englobar en seis distintos:

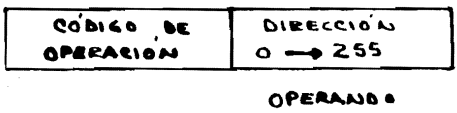
1) Direccionamiento extendido:

Este modo de direccionamiento permite al MPU referirse a cualquier localidad de memoria comprendida entre los 65,536 - BYTES que el MPU puede acceder directamente. Las instrucciones usadas en este direccionamiento consisten en tres BYTES;- el primero, es el código de operación. El segundo BYTE con-- tiene los ocho BITS más significativos del operando y, el tercero, los menos significativos. La dirección contenida en el operando es una dirección absoluta.



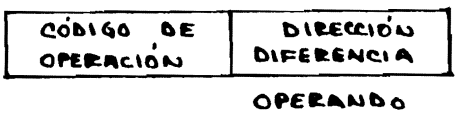
2) Direccionamiento directo:

Este modo faculta al MPU el acceder cualquier localidad de memoria incluída en los primeros 256 BYTES, conocida como "página cero". Las instrucciones usadas con este direccionamiento consisten en dos BYTES: el primero corresponde al código de operación y el segundo a la dirección contenida en la página cero.



3) Direccionamiento indiciado.

Con este modo de direccionamiento, la dirección numérica es variable y dependiente del contenido del registro índice, esto permitirá tener acceso a cualquier localidad de memoria, comprendida entre el valor del contenido de X y 255 localidades próximas. La dirección que incluye el registro índice no se verá afectado con esta operación. Este modo de direccionamiento consiste en dos BYTES: código de operación y operando, éste último, define la diferencia que existe entre la dirección a la cual queremos tener acceso y el contenido del registro índice.



4) Direccionamiento relativo.

Esta forma de direccionamiento es disponible sólo para las instrucciones de ramificación condicional, ramificación incondicional (BRA) y ramificación a subrutina (BSR). Ninguna de estas instrucciones pueden ser usadas con algún otro modo de direccionamiento.

El rango para la ramificación está limitado por:

$(PC+2) -128 \leq D \leq (PC+2) +127$

donde:

PC = dirección del primer BYTE de la instrucción de ramificación.

D = dirección del destino de la instrucción de ramificación.

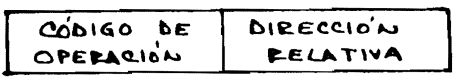
Cuando es necesaria la transferencia de control más allá del rango de la instrucción de ramificación, pueden ser usadas otras instrucciones como JMP (salto incondicional) ó JSR (salto a subrutina). Estas instrucciones no usan el modo de direccionamiento relativo.

La correspondencia que existe entre la dirección relativa y la dirección absoluta del destino de una instrucción de ramificación está dado por:

$$D = (PC + 2) + R$$

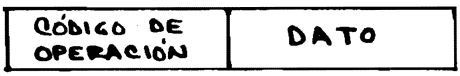
donde:

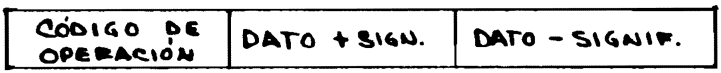
R = número binario de 8 BITS, en complemento a dos, almacenada en el segundo BYTE de la instrucción de ramificación.



5) Direccionamiento inmediato:

En este tipo de direccionamiento, el operando de la instrucción incluye la información requerida. El operando podrá contener un BYTE para instrucciones referidas a los acumuladores o dos BYTES, cuando esté referida al registro X o al SP.- En ambos casos, el código de operación hará implícito a cuálde esos registros se refiere.





OPERANDO

6) Direccionamiento inherente:

Las instrucciones que no necesitan incluir un operando - para definir la operación a realizar, caen dentro de este modo de direccionamiento. Operaciones tales como: "limpiar un acumulador" (CLR A o CLR B) o incrementarlo (INC A o INC B) no - requieren ningún otro dato adicional para efectuar dicha operación.

Estas instrucciones, serán definidas con sólo indicar el código de operación.

En el capítulo IV B se verán ejemplos de aplicación para estos tipos de direccionamiento.

CAPITULO II

DISPOSITIVOS DE SOPORTE

Los dispositivos que completan un sistema computador se-pueden enumerar de acuerdo a la capacidad o versatilidad que-se le quiera dar al conjunto.

Partamos ahora, de la condición nivel mínimo; es decir,-cuántos y cuáles dispositivos son necesarios para concretar - un sistema con el que se pueda evaluar la familia del micro-- procesador en cuestión:

1.- Requerimos de un dispositivo que nos proporcione la-capacidad de almacenar y recuperar la información que estemos procesando. Este dispositivo será la memoria de lectura/es-- critura. (RWM o RAM). S6810.

2.- Es importante contar con un programa alambrado que - nos auxilie en el proceso de I/O; de tal modo que, haya la posibilidad de conectarse al sistema por medio de un TTY o equivalente. Este programa recibe el nombre de monitor, ya que - se encarga de iniciar (programarlos para la comunicación) el-o los dispositivos facultados para el enlace, además de otras tareas explicadas posteriormente. Este programa deberá ser - inalterable; este modo de acceso lo proporciona la memoria de sólo lectura. (ROM). S6830.

3.- Si deseamos agregar un programa fijo, pero de uso -- particular, requerimos de una memoria programable de sólo lectura (PROM). S6834.

4.- Para la comunicación máquina-usuario, se requiere de un acoplador que facilite el manejo de la información; cuando la correspondencia es necesario hacerla a través de un TTY ó-una pantalla, requerimos que los datos se manejen en serie; - este modo de comunicación nos lo proporciona un acoplador o - interfase convertidor en paralelo-serie y/o serie-paralelo. - (ACIA). S6850.

5.- Cuando los datos se requieran manejar en paralelo; - verbigracia: un convertidor D/A o A/D, una impresora o un teclado, etc., se cuenta con un acoplador o interfase en paralelo. (PIA) S6820.

CAPITULO IB.1

RAM

Las memorias digitales más utilizadas son las memorias - de lectura-escritura, cuya expresión significa que realizan - las operaciones de leer y de escribir a una velocidad similar.

Otras características importantes son: la velocidad de - operación (definido en términos de tiempo de acceso) y la duración del ciclo. El tiempo de acceso (tacc) es simplemente- al tiempo necesario para leer o para escribir en cualquier -- ubicación del almacenamiento. Las memorias de acceso aleatorio (RAM), pueden realizar la totalidad de las operaciones de leer y de escribir en un período mínimo especificado, conocido como duración del ciclo. (tcyc).

Hay que notar un punto importante cuando consideremos velocidades de operación, antes de que una palabra pueda ser -- leída, es necesario localizarla. El tiempo requerido para localizar y leer una palabra desde memoria es llamado tiempo de acceso. El procedimiento para localizar la información puede ser dividido en dos clases: acceso aleatorio y acceso secuencial. Una memoria de acceso aleatorio (RAM) es cuando cualquier localidad, en el dispositivo, puede ser seleccionada al azar, el acceso a la información es directo, y el tiempo de - acceso es aproximadamente igual para cada una de las localidades.

Un dispositivo de acceso secuencial es, cuando el arribo a la localidad escogida puede ser precedida por el paso por - otras localidades, de tal modo que, el tiempo de acceso varía de acuerdo a la localización del almacén; por ejemplo: si la- información deseada se localiza en una cinta magnética y el - dato se encuentra a la mitad de la longitud de la cinta, tendremos que recorrer la mitad de la misma para acceder a la palabra buscada.

Una memoria estática es aquella que sólo necesita que la información se le de una vez para que la mantenga, mientras - no se le cambie; FLIP FLOPS, núcleos magnéticos, cintas y tarjetas perforadas son ejemplos de este tipo de almacenamiento.

Una memoria dinámica requiere que a cada ciclo se le - vuelva a dar la información para que no la pierda. (El "re--- frescamiento" evita ese cambio de estado).

La memoria S6810, soporte del S6800, es una memoria de - acceso aleatorio tipo estático, ya que no requiere de un "refrescamiento" de su información almacenada, mientras la fuen-

te de poder esté aplicada en sus terminales. Obviamente, no- es el único tipo de memoria con facilidad de uso en el sistema, pero es la contenida en el conjunto AMI S6800.

Este dispositivo de almacenamiento cuenta con 128 localidades de 8 BITS cada una. Por lo tanto, requiere de 7 líneas para seleccionar alguna de estas localidades; además de estas líneas, cuenta con otras 6 que sirven para habilitar al dispositivo, de tal modo que, queda capacitado para leer o para escribir cuando en los CS'S (CHIP SELECT) hay nivel uno y cuando en los $\overline{CS'S}$ hay nivel cero. Ahora bien, como recordaremos, el CPU genera una señal de control llamada R/W, de tal modo - que, para leer de la memoria, el R/W debe ser nivel uno y para escribir en la memoria R/W será nivel cero. Como se puede observar en la figura IB.1.1, R/W en conjunción con las líneas de selección controlan los BUFFERS de tres estados, además de la lógica asociada, de tal modo que, la información será accesible cuando en el punto A se registre el nivel uno. - Una vez habilitada se podrá leer cuando en la terminal R/W se registre el nivel uno, y escribir cuando en R/W se registre - un nivel cero.

A partir de las 7 líneas de selección y de las 6 líneas- de habilitación, podemos conformar la ubicación de la memoria, de tal modo que, dividiremos nuestro rango de selección, o -- sea, 65536 BYTES entre el número de BYTES que contenga nuestra memoria S6810. Así, podemos agregar al sistema 512 dispositivos S6810 acomodados de 128 en 128, 80)h.

Veamos un ejemplo: necesitamos implementar dos dispositivos S6810 que estén direccionados, empezando en la localidad- 0000)h. De tal modo que, el primero será alambrado para que- responda desde la dirección 000)h hasta 007F)h y el siguiente desde 0080)h hasta la dirección 00FF)h. Hay que recordar que para habilitar correctamente nuestra memoria debemos incluirlas señales VMA, Ø2 y R/W.

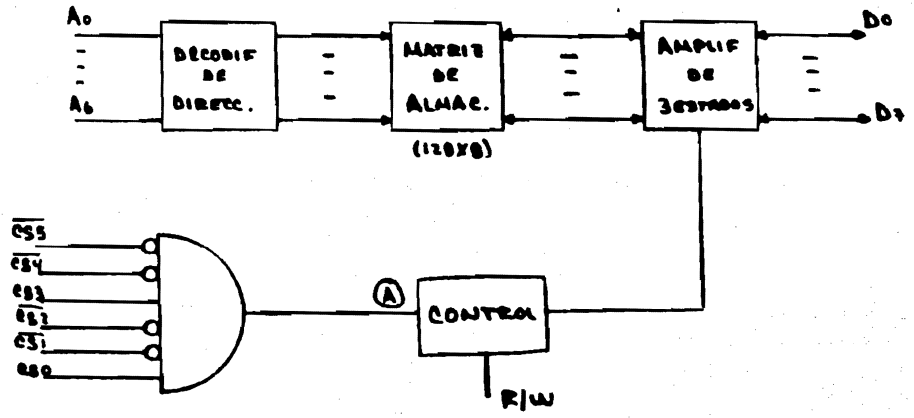
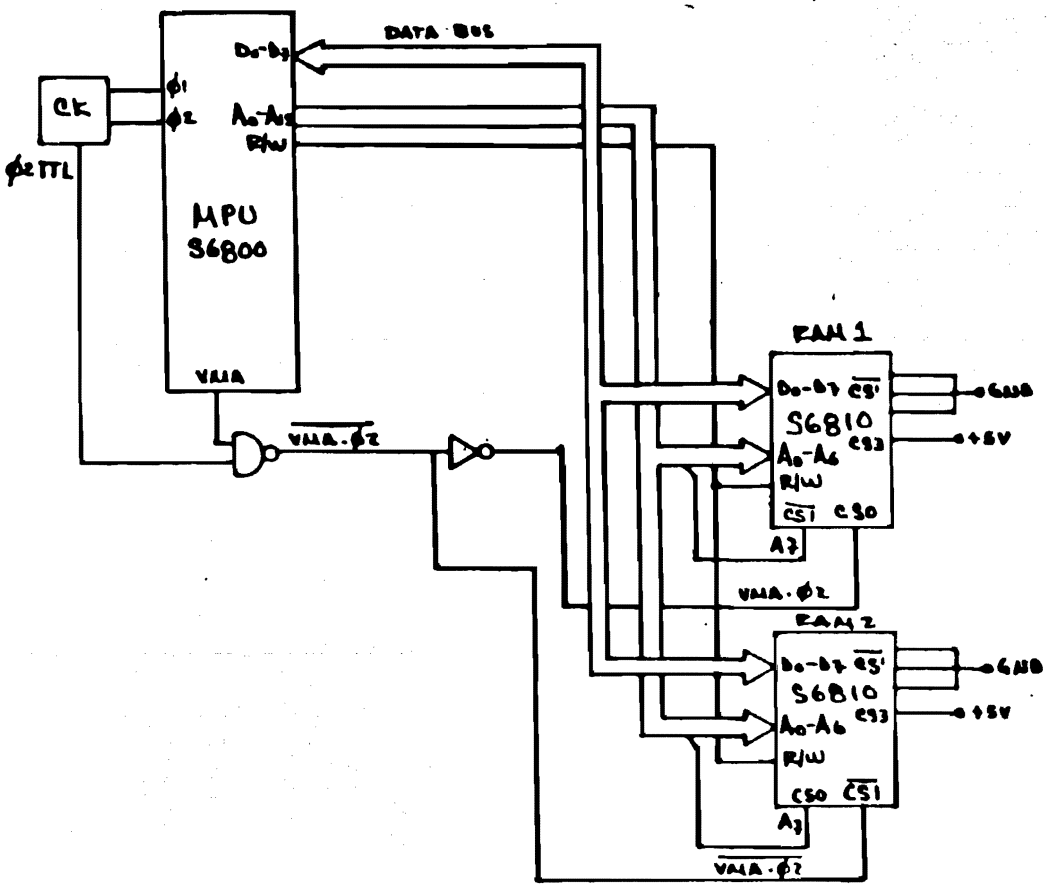


FIGURA IB.1.1.- DIAGRAMA DE BLOQUE DEL S6810 (128 X 8 RAM).

Las líneas de selección de la memoria se conectan directamente a las de dirección menos significativas del MPU y, -- las líneas de habilitación se conectan como muestra la figura IB.1.2. Como sólo son dos dispositivos, únicamente se requiere de una línea de dirección adicional quelos diferencie uno- del otro. VMA y Ø2 los implementamos en otra línea de habilitación en cada dispositivo.

Las características eléctricas y de tiempos de R/W se -- dan en el apéndice.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	A1
RAM1 00-7F	—	—	—	—	—	—	—	—	0	X	X	X	X	X	X	X	RAM 1
RAM2 80-FF	—	—	—	—	—	—	—	—	1	X	X	X	X	X	X	X	RAM 2

— LÍNEA NO UTILIZADA

X LÍNEA VARIABLE.

FIGURA IB.1.2.- CONEXIONAMIENTO DE DOS S6810 CON EL MPU.

CAPITULO IB.2

ROM

Estos dispositivos de memoria tienen la característica de que se puede leer en ellos, pero no se puede escribir información dentro de los mismos. La información almacenada en estas memorias es, por lo tanto, introducida en los dispositivos desde su manufactura.

El ROM tiene almacenada información que conviene mantener en forma permanente. Dentro del conjunto de una microcomputadora, usualmente se tiene en los ROMs programas o subprogramas de uso frecuente, de tal manera que, faciliten o simplifiquen la aplicación de la microcomputadora.

Fundamentalmente, un ROM es un elemento con varias líneas de entrada y otras de salida, de tal modo que, el estado que guardan las salidas depende de la condición de las entradas. O sea, que siguen la ley de una tabla de verdad definida por el diseñador.

La memoria S6830 contiene 1024 palabras de 8 BITS cada una, su configuración se muestra en la figura IB.2.1. El tiempo de acceso a cualquier ubicación del almacenamiento es aproximadamente el mismo. La implementación en el sistema se hace de forma similar a un RAM, tal vez la única diferencia sea que en el ROM la señal de R/W no es conectada, esto debido a que sólo se habilitará para lectura.

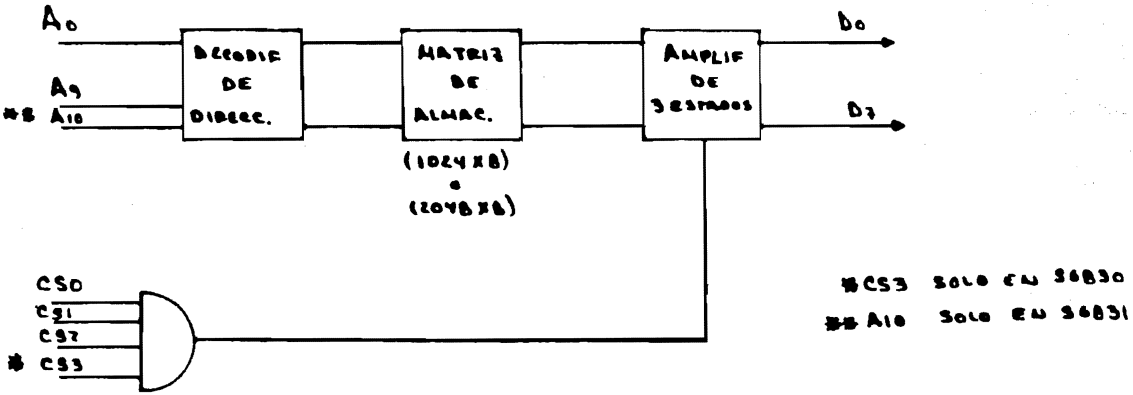


FIGURA IB.2.1.- DIAGRAMA DE BLOQUE DE S6830/1.

Como el S6830 contiene 1024 BYTES de información, requiriéndose 10 líneas de selección, podemos agregar al sistema, dentro de nuestro rango de selección, 64 dispositivos S6830. En el caso del S6831, sólo 32, ya que éste contiene 2048 BYTES de información. Obviamente, el rango de selección debe ser compartido con la memoria RAM, así como con otros elementos. De tal modo que, se deben definir las áreas de almacenamiento, tanto para RAM como para ROM, y los dispositivos adicionales.

Ahora, supongamos que queremos agregar a nuestro sistema mínimo, mencionado en el capítulo referido a RAM, un elemento ROM. Este deberá localizarse en la dirección de inicio - - E000)h; como el dispositivo contiene 1024 BYTES (400)h) la última dirección incluída será la E3FF)h. Una vez más, VMA y Ø2 deberán estar presentes en las líneas de habilitación, que en este dispositivo son cuatro, todas ellas deberán ser nivel uno para que se habilite el dispositivo. Teniendo la opción de solicitar al fabricante que los niveles activos de los - - CS's sean diferentes, es decir, uno o dos de ellos pueden ser activos con nivel uno y tres o dos de los restantes activos -

con nivel cero, etc.

En la figura IB2.2 podemos observar la forma en cómo se- agrega un S6830 a nuestro sistema mínimo. Sin embargo, incu- rrimos en un error, ya que, cuando accedemos alguna localidad contenida en el rango 00)h y FF)h, inclusive, habrá una falsa lectura, ya que tanto los RAM como el ROM responderán dentro- del rango antes mencionado. Para evitar esta lectura equivo- cada necesitamos diferenciarlos de alguna manera; una solu- - ción será agregar la línea de dirección A15. De tal manera - que, ésta deberá ser nivel cero, nivel diferente al del ROM,- para seleccionar los elementos RAM. Con esta configuración,- cuando A15 sea nivel cero, la parte de memoria posible de ha- bilitar serán los dispositivos RAM y nivel uno para el ROM, - es decir, cada dirección debe corresponder a una y sólo una - localidad de memoria.

La figura IB2.3 muestra la conexión correcta.

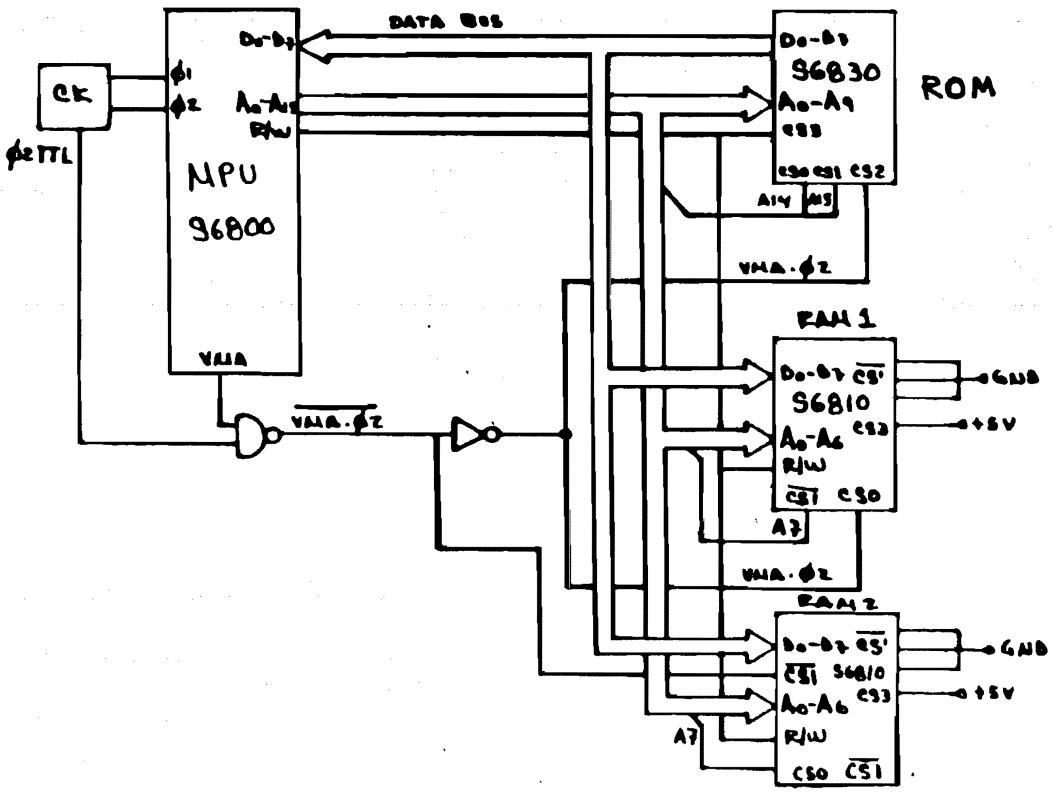
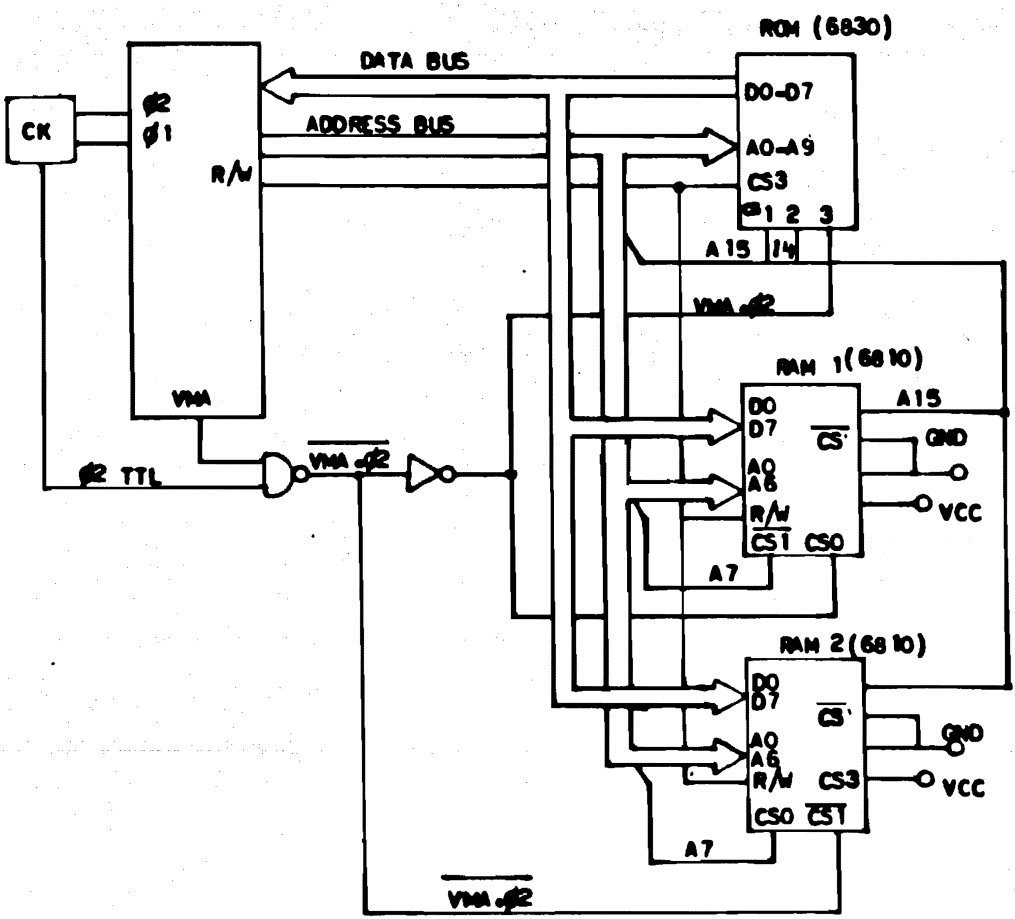


FIGURA IB2.2.- CONEXIONAMIENTO DEL S6830 y S6810 CON EL MPU.



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	-	-	-	-	-	-	-	0	X	X	X	X	X	X	X
0	-	-	-	-	-	-	-	1	X	X	X	X	X	X	X
1	1	-	-	-	-	X	X	X	X	X	X	X	X	X	X

DIS.	DIR
RAM1	0-7F
RAM2	80-FF
ROM	E000-E3FF

FIGURA IB2.3.- CONEXIONAMIENTO CORRECTO DEL S6830
Y S6810 CON EL MPU.

CAPITULO IB.3

PROM

Un PROM tiene características similares a una memoria ROM, es decir, el programa contenido en él es permanente, aún desconectada la fuente de poder. Sin embargo, el programa es alterable, en el caso del S6834, por medios ópticos.

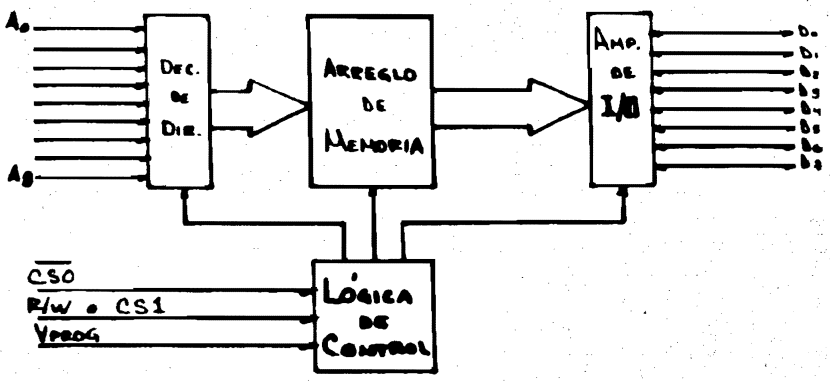
La programación de un PROM se efectúa de manera similar a la escritura de una memoria RAM; la información que se desea grabar se mantiene presente en la dirección deseada, mientras en una terminal indicada para este propósito se aplica un voltaje adecuado. Esto hace que en la celda direccionada, queden los datos contenidos en ella.

La memoria PROM S6834 contiene 512 BYTES de capacidad de almacenamiento; su organización se muestra en la figura IB.3.1. Sus aplicaciones, además de ser almacén de información, pueden ser variadas, junto con el ROM, sirven como decodificadores, generadores de caracteres, codificadores, etc.

Cada modelo y marca de PROM tiene una manera diferente de programarse, aunque esta diferencia no es extrema, la forma particular de programar al S6834 es como sigue:

Inicialmente, y después de un "borrado", todos los BITS del S6834 están en nivel cero. La señal R/W es usada para seleccionar el modo de lectura o escritura. Cuando el R/W es nivel cero, el dispositivo está en el modo de escritura. Las salidas son inhabilitadas. La palabra direccionada es seleccionada de la misma manera que el modo de lectura. El dato que será programado es presentado con ocho BITS en paralelo, y después que el dato y la dirección son estables, un voltaje de programación (Vprog = -50 Vdc) es aplicado. Vprog escribe el dato dentro del arreglo de memoria.

La cantidad de energía requerida para programar, y asegurar que la memoria ha retenido los datos, puede ser definida como una función del número de pulsos de programa (n) y el ancho del pulso de programa (tpw); de tal modo que, ntpw es mayor o igual que 60 mseg. Así si tpw es igual a 3 mseg., se requiere de un mínimo de 20 pulsos de programa y si se aplican pulsos de duración igual a 5 mseg., se requieren de 12 pulsos como mínimo.



. FIGURA IB.3.1.- ORGANIZACION DEL S6834 PROM.

La operación de lectura va acompañada de un nivel uno - en la terminal R/W y con la terminal Vprog conectada a Vcc.

El proceso de borrado se efectúa por exposición del arreglo a una radiación intensa de rayos ultra violeta, por período de 7 a 10 minutos.

El conexionamiento de un dispositivo PROM se efectúa de la misma manera que un ROM. Pero como en el PROM las líneas de datos son bidireccionales, es necesario fijar la línea R/W en el modo de lectura.

La utilización de un PROM implica que el computador se ha destinado para un uso particular o para agregar una subrutina o programa fijo. Por esto, es necesario establecer su aplicación antes de decretar su ubicación, siendo ésta seleccionada por el usuario.

CAPITULO IB.4

ACIA

Este dispositivo permite el manejo, en forma asincrona,- de información en serie. Esta información puede ser manejada de diversas maneras, gracias a la capacidad del elemento, los diferentes modos son seleccionados por el usuario mediante la programación del dispositivo. La velocidad de transmisión -- también puede ser elegida por el programador y su máxima es - de 500,000 BITS por segundo (BPS). El reloj que controla dicha velocidad de transmisión puede ser modificado por programa.

La programación del ACIA (ASYNCHRONOUS COMMUNICATION INTERFACE ADAPTER) se hace por medio del DATA BUS y seleccionada como localidad de memoria. La figura IB.4.1 muestra en -- forma fragmentada y simplificada los registros en el S6850 - ACIA que intervienen en la programación, transmisión, recep-- ción y supervisión de la misma; estos cuatro registros son como siguen:

a) Registro de transmisión de datos.- En este registro - debemos almacenar la información que deseamos transmitir (TDR).

b) Registro de control.- En este registro se almacena el código relativo a la manera en que se desea transmitir la información (CR).

c) Registro de estado.- Aquí se nos muestra el estado -- que guarda la transmisión/recepción de la información que se maneja a través del ACIA (SR).

d) Registro de recepción de datos.- De este registro podemos leer la información que ha sido recibida del exterior - (RDR).

Los registros de control y de transmisión de datos, son- registros de sólo escritura. Los registros de estado y recepción de datos son registros de sólo lectura. Los cuatro re-- gistros anteriores son accesibles gracias a una línea de se-- lección llamada selector de registro (RS). De tal manera que, cuando RS es nivel cero, seleccionamos a dos de los cuatro registros y tendremos acceso a los dos restantes cuando RS es - nivel uno. Gracias a que dos de ellos son registros de sólo-lectura, podemos accederlos cuando la línea de R/W sea nivel-uno; nuevamente los dos sobrantes serán seleccionados cuando-R/W sea nivel cero, ya que son registros de sólo escritura.

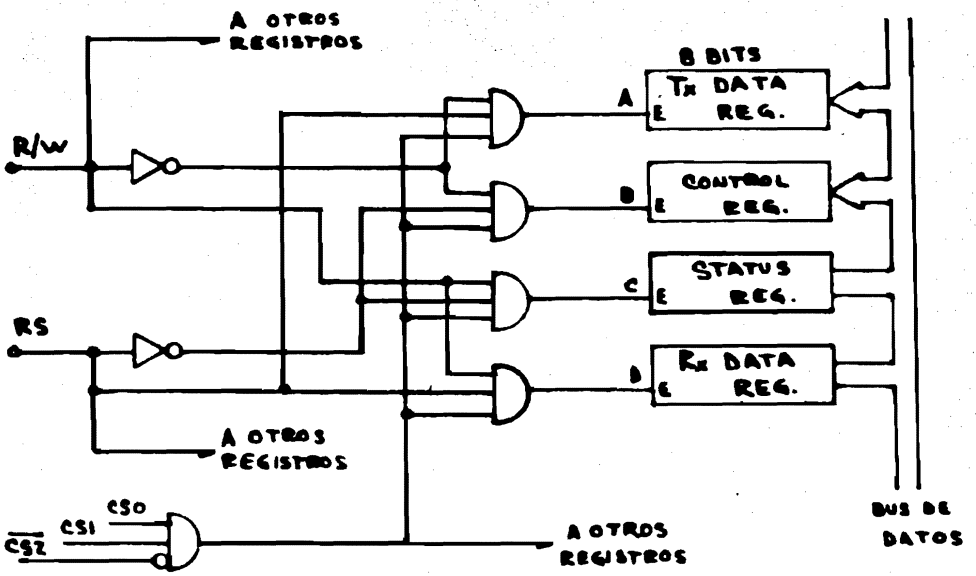


TABLA DE VERDAD

	RS	R/W	A	B	C	D	
CONTROL = $\overline{RS} \cdot \overline{R/W}$	0	0	0	1	0	0	SÓLO ESCRITURA
STATUS = $\overline{RS} \cdot R/W$	0	1	0	0	1	0	SÓLO LECTURA
Tx DATA = $RS \cdot \overline{R/W}$	1	0	1	0	0	0	SÓLO ESCRITURA
Rx DATA = $RS \cdot R/W$	1	1	0	0	0	1	SÓLO LECTURA

FIGURA IB.4.1.- CONFIGURACION FRAGMENTADA DEL ACIA.

Generalmente, el selector de registro (RS) se conecta a- una de las líneas de dirección del MPU; ésta será la menos -- significativa, o sea, A0. De tal manera que, cuando R/W sea - nivel uno seleccionaremos los dos registros de sólo lectura,- que son el registro de estado y el registro de recepción de - datos. Cuando R/W sea nivel cero, habremos escogido los re-- gistros de sólo escritura, es decir, el registro de control y el de transmisión de datos.

La figura IB.4.1. contiene la tabla de verdad que nos -- muestra la manera de habilitar el registro seleccionado.

Veamos un ejemplo: Supongamos que queremos agregar a - - nuestro sistema mínimo un dispositivo ACIA. Como mencionamos anteriormente, se requiere de dos localidades para poder manejar nuestra información a través de un ACIA, de ahí que, co-- nectemos A0 al selector de registro (RS). Ahora bien, estas- dos localidades las podemos ubicar dentro de nuestro rango de selección; recordemos que el área ocupada por los dos elementos RAM incluyen la dirección 00)h hasta FF)h, y nuestro ROM- ocupa el área comprendida entre E000 hasta E7FF, inclusive. - De tal modo que, no debemos implementar nuestra ACIA dentro - de algunas de estas áreas. Tomemos las direcciones FBCE y -- FBCF como localidades referidas al ACIA. La figura IB.4.2. - muestra el conexionamiento.

Una vez más se tiene que diferenciar al ACIA del ROM y - RAM's, para eso, se utiliza la línea A12 logrando ese propósito. Habiendo implementado el dispositivo en cuestión, analicemos el acceso a los registros internos del ACIA.

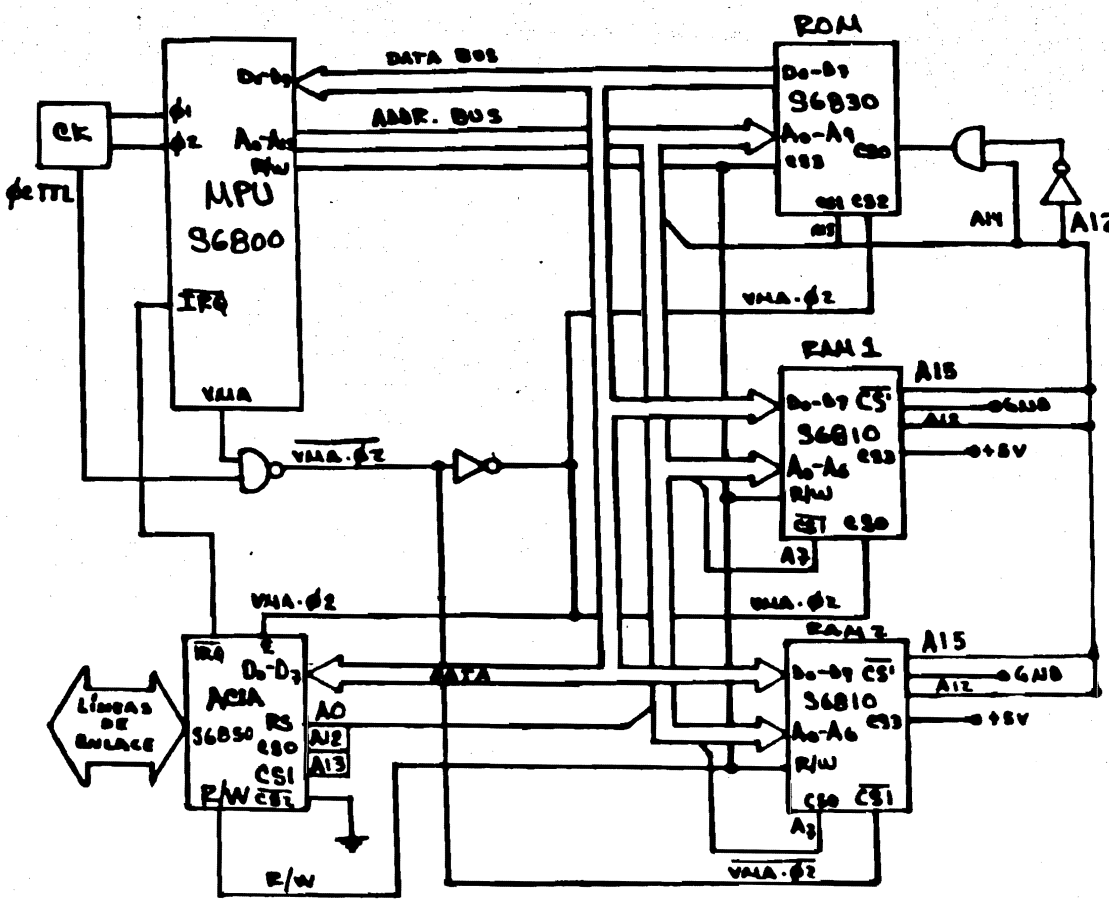
Puesto que A0 fue conectada al selector de registro - (RS) y la señal del CPU, R/W fue conectada al R/W del ACIA, - podemos concluir que:

a) En la dirección FBCE)h están contenidos los registros de control y estado.

a') Durante el ciclo de lectura (R/W nivel uno) habilitaremos el registro de estado (SR).

a'') Durante el ciclo de escritura (R/W nivel cero) habilitaremos el registro de control (CR).

b) En la dirección FBCF)h están contenidos los registros de datos.



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	A.L
0	—	—	0	—	—	—	—	0	X	X	X	X	X	X	X	FAM1
0	—	—	0	—	—	—	—	1	X	X	X	X	X	X	X	FAM2
1	1	—	0	—	—	X	X	X	X	X	X	X	X	X	X	ROM
—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	X	ACIA

FIGURA IB.4.2.- DIAGRAMA DEL CONEXIONAMIENTO DEL ACIA CON EL MPU.

b') En el modo de lectura (R/W nivel uno) seleccionare-- mos el registro de recepción de datos.

b'') En el modo de escritura (R/W nivel cero) selecciona remos el registro de transmisión de datos.

Para transmitir (recibir) la información bastará con escribir (leer) los datos en (de la) localidad FBCF)h.

Para incurrir en el modo de control (supervisión), será- suficiente con escribir (leer) los datos en (de la) localidad FBCE)h.

El modo de lectura/escritura de las localidades anteriores es similar a la de un RAM.

Veamos ahora la función de los registros de estado y de- control.

Las líneas implicadas en la recepción/transmisión de información en serie, se indican en la parte derecha del diagrama de bloques del ACIA mostrada en la figura IB.4.3. En la-- línea RXDATA se recibe la información proveniente de la terminal o dispositivo con el que se tiene contacto. Por la línea TXDATA se envía o transmite la información hacia el o los dispositivos adecuados

Pues bien, el registro de estado (SR) nos da la oportunidad de supervisar la situación que guardan las líneas y registros implicados en la recepción/transmisión de datos por el - ACIA. Este registro de 8 BITS puede ser leído por el procesador a través de las líneas de datos (DO-D7); la figura IB4.4A muestra como está conformado el registro de estado. Los BITS incluídos en este registro nos indican las siguiente condiciones:

1) RECEIVE DATA REGISTER FULL (RDRF), BIT 0- Registro de recepción de datos completo. Indica que el dato recibido ha- sido transferido al registro de recepción de datos (RDR), los BITs PE, OVRN y FE son fijados de acuerdo a la situación del- dato. RDRF es limpiado después que el MPU haya leído del registro de recepción de datos o que se aplique un MASTER RESET (en el registro de control).

2) TRANSMIT DATA REGISTER EMPTY (TDRE), BIT 1- Registro- de transmisión de datos vacío. Indica que el contenido del - registro de transmisión de datos ha sido transferido y que un nuevo dato puede ser admitido desde el MPU. El BIT b1 será-- limpiado cuando se efectúe una operación de escritura en el - TAR o cuando ocurra un nivel uno en la línea $\overline{CTS}$.

3) DATA CARRIER DETECT ($\overline{DCD}$), BIT 2 Detector de BIT de - paridad o detector de BIT de acarreo. Indica que el BIT de - paridad no está presente. Un frente de onda positivo en la - línea de entrada $\overline{DCD}$ causa que el BIT2 sea nivel uno y genera una interrupción (BIT 7=1), cuando RIE (BIT del registro de - control) es nivel uno; $\overline{DCD}$ e IRQ (BIT7) serán nuevamente ni-- vel cero cuando sea leído el registro de estado y el registro

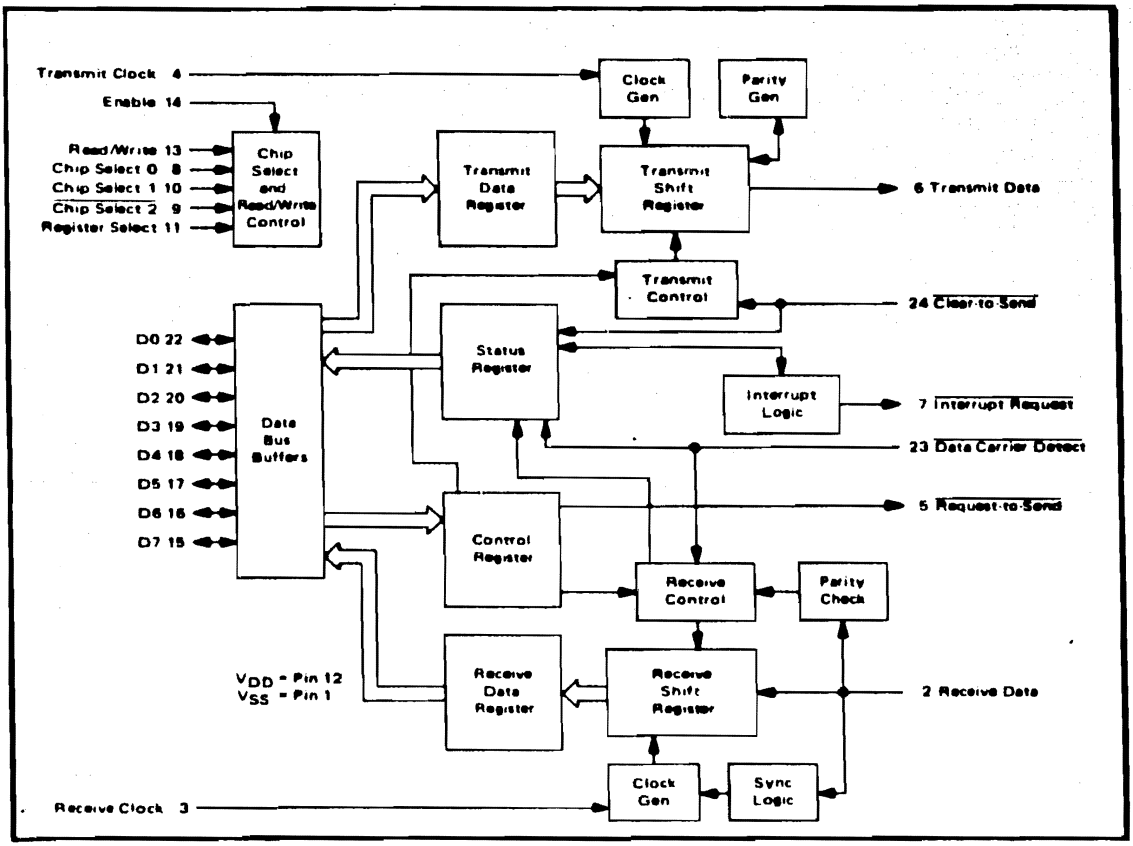


FIGURA IB.4.3. DIAGRAMA DE BLOQUES DEL ACIA.

Data Bus Line Number	Buffer Address			
	RS • R/W Transmit Data Register (Write Only)	RS • R/W Receive Data Register (Read Only)	RS • R/W Control Register (Write Only)	RS • R/W Status Register (Read Only)
0	Data Bit 0*	Data Bit 0	Counter Divide Select 1	Receive Data Register Full (RDRF)
1	Data Bit 1	Data Bit 1	Counter Divide Select 2	Transmit Data Register Empty (TDRE)
2	Data Bit 2	Data Bit 2	Word Select 1	Data Carrier Detect ($\overline{DCD}$)
3	Data Bit 3	Data Bit 3	Word Select 2	Clear to Send ($\overline{CTS}$)
4	Data Bit 4	Data Bit 4	Word Select 3	Framing Error (FE)
5	Data Bit 5	Data Bit 5	Transmit Control 1	Receiver Overrun (OVRN)
6	Data Bit 6	Data Bit 6	Transmit Control 2	Parity Error (PE)
7	Data Bit 7***	Data Bit 7**	Receive Interrupt Enable	Interrupt Request (IRQ)

* Leading bit = LSB = Bit 0
** Data bit will be zero in 7 bit plus parity modes.
*** Data bit is "don't care" in 7 bit plus parity modes.

FIGURA IB.4.4. REGISTROS INTERNOS DEL ACIA

de recepción de datos o cuando se aplique un MASTER RESET.

4) CLEAR TO SEND ($\overline{CTS}$), BIT 3. Esta señal es un reflejo del estado que guarda la línea $\overline{CTS}$. En el estado nivel uno, el TDRF es inhibido. El BIT $\overline{CTS}$ no es afectado por el MASTER RESET.

5) FRAMING ERROR (FE), BIT4. Error estructural. Indica la ausencia del primer BIT de STOP como resultado de un error de sincronía, transmisión defectuosa o una condición de BREAK. Será nivel cero cuando el próximo dato transferido sea correcto.

6) RECEIVER OVERRUN (OVRN), BIT5. Este BIT indica que uno o más caracteres en el flujo de datos ha sido perdido. Esto es, un carácter o un número de caracteres fueron recibidos pero no leídos por el RDR antes de que un nuevo carácter haya sido recibido. Este BIT es limpiado (nivel cero) cuando se efectúa una operación de lectura en RDR.

7) PARITY ERROR, BIT6. El BIT de error de paridad indi-

ca que el número de unos (nivel uno) en el carácter no concuerda con la paridad seleccionada (paridad par o impar). La paridad impar es definida como que el número total de unos es impar. Este BIT es habilitado por medio de los BITS; WS3, WS2 y WS1 del registro de control.

8) INTERRUPT REQUEST (IRQ), BIT7. El BIT de solicitud de interrupción indica el estado de $\overline{IRQ}$, o sea, es el complemento de $\overline{IRQ}$. IRQ es limpiado por una operación de lectura en el RDR o una operación de escritura en el TDR.

El registro de control (CR) nos proporciona la facilidad de programar el modo de transmisión/recepción que convenga al tipo de dispositivo de I/O con el que se cuente. Este registro incluye 8 BITS de sólo escritura, mostrado en la figura IB.4.4 y que son como sigue:

a) COUNTER DIVIDE SELECT (CDS2, CDS1) BIT1 y BIT0. La velocidad de transmisión/recepción de datos es afectada por estos dos BITS. En las terminales TXCLOCK y RXCLOCK se aplica un tren de pulsos con una frecuencia determinada; la velocidad de transmisión/recepción real, será el cociente de la frecuencia aplicada, entre un número determinado por el código almacenado en estos dos BIT's.

Adicionalmente estos BITs son utilizados para proveer un MASTER RESET para el ACIA, que limpia el registro de estado (SR) (excepto a los BIT'S $\overline{CTS}$ y $\overline{DCD}$).

BIT 1 CDS 2	BIT 0 CDS 1	FUNCION
0	0	÷ 1
0	1	÷ 16
1	0	÷ 64
1	1	MASTER RESET

b) WORD SELECT (WS3, WS2, WS1), BIT4, BIT3 y BIT2. Este selector es usado para elegir la longitud de la palabra, tipo de paridad y el número de BITs de STOP. La codificación es la siguiente:

BIT 4	BIT 3	BIT 2	F U N C I O N		
WS3	WS2	WS1	LONGITUD	PARIDAD	STOP BITs
0	0	0	7 BITs	PAR	2
0	0	1	7	IMPAR	2
0	1	0	7	PAR	1
0	1	1	7	IMPAR	1
1	0	0	8	NINGUNA	2
1	0	1	8	NINGUNA	1
1	1	0	8	PAR	1
1	1	1	8	IMPAR	1

C) TRANSMITTER CONTROL (TC2, TC1), BIT6 y BIT5. La sección transmisora del ACIA es controlada por los BITs 5 y 6. - Las cuatro combinaciones proveen la transmisión de un comando, de BREAK, un comando de solicitud de envío (RTS) y un habilitador/inhibidor de solicitud de interrupción (IRQ) para el modo de transmisión.

BIT 6	BIT 5	
TC2	TC1	F U N C I O N
0	0	$\overline{RTS}$ = 0, inhibe interrupción
0	1	$\overline{RTS}$ = 0, habilita interrupción
1	0	$\overline{RTS}$ = 1, inhibe interrupción
1	1	$\overline{RTS}$ = 0, transmite un BREAK e - inhibe interrupción.

d) RECEIVE INTERRUPT ENABLE (RIE), BIT7. Cuando RIE es nivel uno, habilita la salida de $\overline{IRQ}$ en el modo de recepción. Si RIE es nivel cero inhibe la salida de $\overline{IRQ}$ en el modo de recepción.

- Transmisión. La transmisión de un carácter puede ser como sigue: Leyendo el registro de estado (SR) podemos saber si el registro de transmisión está vacío; si es así, se colocará el dato con un comando de escritura, en el registro de -

transmisión. Este carácter es transferido al registro de corrimiento que hará la conversión paralelo serie y sumará las características dadas en el registro de control (1 ó 2 BITs de STOP, etc.). Antes de que se intente transmitir otro carácter, se deberá leer el registro de estado para cerciorarse que el registro de transmisión está vacío, una vez realizada esta operación, se podrá volver a cargar otro dato y enviarlo al periférico.

- Recepción. La secuencia típica en el modo de recepción es como sigue: El registro de estado es leído para determinar si un carácter ha sido recibido desde un periférico. Si el RDR está lleno, el carácter es colocado en el BUS del ACIA, cuando la misma ha recibido un comando de lectura proveniente del MPU; pudiéndose verificar las condiciones establecidas. Mediante un programa, el carácter leído, es almacenado o usado para otro fin. El registro de estado puede ser leído nuevamente para determinar si otro carácter ha sido recibido.

CAPITULO IB.5.

PIA

Este dispositivo permite al usuario recibir/transmitir información paralelo-paralelo, para usos como acoplar un Digital/Analógico o un Analógico/Digital o también para una impresora de línea o algún otro elemento que emplee la información en paralelo. Asimismo para manejar un transistor o similar o un elemento TTL, etc. Esto, gracias a que el dispositivo contiene 18 líneas de enlace con el exterior, y que todas ellas son líneas bidireccionales (entrada o salida), pudiéndose usar una o todas al mismo tiempo. Dos líneas más son sólo empleadas como entrada (CA1, CB1).

El S6820 (PIA, PERIPHERAL INTERFACE ADAPTER), contiene registros con los cuales es posible programar al dispositivo, para efectuar diversas tareas determinadas por el usuario. La figura IB5.1 muestra el diagrama de bloques del S6820, en él se pueden observar los registros que intervienen en la programación (CRA, CRB), en la recepción/transmición de datos (ORA, ORB) y en la determinación de dirección para las líneas exteriores (DDRA, DDRB) . Cada uno de estos registros representa una localidad de memoria, de igual manera que en un RAM.

El S6820 se puede dividir en dos partes, la sección A y la B. La información escrita o leída por el MPU, por medio del DATA BUS, puede destinarse o provenir de la sección A o de la B del PIA. Por ello cada sección cuenta con registros similares que realizan las operaciones independientemente.

Los seis registros representan cuatro ubicaciones de memoria, estas localidades se pueden diferenciar gracias a los selectores de registro RS0 y RS1 de la siguiente manera :

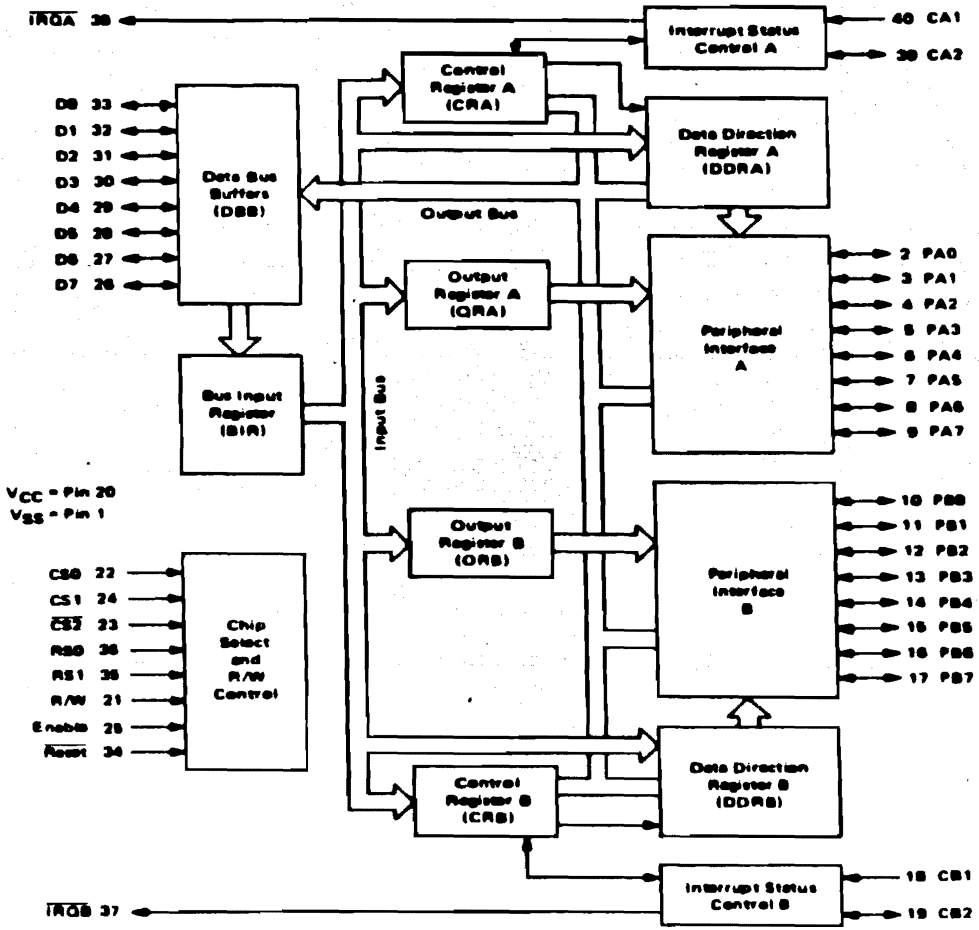


FIGURA IB.5.1.- DIAGRAMA DE BLOQUES DE LA PIA.

RS1	RS0	SELECCION
0	0	ORA o DDRA
0	1	CRA
1	0	ORB o DDRB
1	1	CRB

Las líneas RS0 y RS1 son conectadas a las líneas de dirección menos significativas del MPU. Estas, en conjunción con las tres líneas de CHIP SELECT (CS0, CS1, CS2) y lade E (ENABLE), nos dan la facilidad de ubicar el dispositivo dentro de nuestro rango de selección. Supongamos que la PIA esté ubicada en las localidades FBC8)h hasta FBCB)h, inclusive, de tal modo que los bits menos significativos queden como sigue:

DIRECCION	A1	A0	REGISTRO SELECCIONADO
FBC8	0	0	ORA o DDRA (Registro periférico A)
FBC9	0	1	CRA (Registro de control A)
FBCA	1	0	ORB o DDRB (Registro periférico B)
FBCB	1	1	CRB (Registro de control B).
	RS1	RS0	

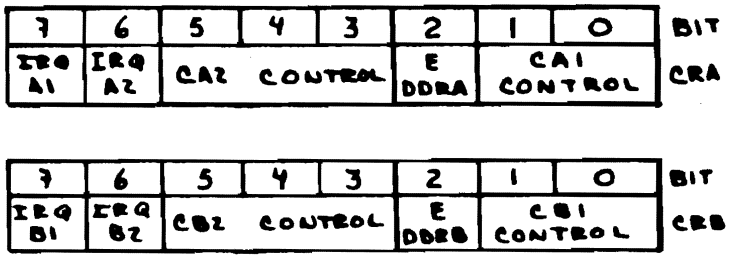
por lo tanto, cuando nos referimos a la localidad FBC8 estaremos accediendo la información relativa al registro periférico A, y así sucesivamente.

Ahora bien, el registro periférico A(B) en realidad representa a dos, pero como les corresponde la misma dirección, se puede, al referirse a alguno de ellos, utilizar el mismo identificador para ambos.

ORA (ORB) Registro de salida de datos A (B) (8 Bits). Es un registro de lectura/escritura en el que se lee o almacena la información a manejar.

DDRA (DDRB) Registro de dirección de datos A (B) - - (8 BITS). Este registro contiene la información que determina la dirección de transmisión de datos (entrada o salida), - de tal modo que si en el BIT i del DDRA (DDRB) existe un ni-- vel cero, implicará que el BIT i del ORA (ORB) está actuando- como entrada. Del mismo modo, si en el BIT i de DDRA (DDRB)- existe un nivel uno, implicará que el BIT i del ORA(ORB) está actuando como salida.

CRA (CRB) Registro de Control A (B) (8 BITS). Los dos - registros de control, CRA y CRB, permiten al MPU establecer y controlar los modos de operación de las líneas periféricas de control CA1, CA2, CB1 y CB2, ubicadas en la parte derecha del diagrama de la figura IB5.1. De tal modo que el CRA se relaciona con las líneas CA1 y CA2, y CRB se relaciona con las -- líneas CB1 y CB2 como se indica en la siguiente tabla.



Se puede observar que el BIT2 de ambos registros de control (CRA, CRB) indican un habilitador de DDRA (DDRB). Estonos permite la opción de escoger entre el registro de salidade datos (ORA, ORB) o el registro de dirección de datos - - (DDRA ,DDRB); recordemos que tanto el registro de salida de - datos como el de dirección de datos, están ubicados en la mis ma dirección, por lo tanto, el BIT2 de ambos registros de con trol, nos da el diferenciador que nos hacía falta. De tal mo do que si en el BIT2 del registro de control (CRA, CRB) codificamos un nivel cero, estaremos seleccionando al registro de dirección de datos (DDRA, DDRB) y si el nivel es uno, estaremos accediendo al registro de salida de datos (ORA, ORB).

Los 5 BIT's menos significativos de ambos registros de - control (BIT 0-5) pueden ser seleccionados tanto para lectura o escritura por el MPU, siempre y cuando en las CSs correspondien

tes se dé la señal de habilidad apropiada. Los BIT's 6 y 7 de -- los dos registros son solo de lectura, y sus niveles son modificados por señales externas provenientes de las líneas de -- control CA1, CB1, CA2 y CB2, de acuerdo al código establecido en los BITs del registro de control correspondiente. Las líneas de control CA1, CB1, CA2 y CB2, además de afectar los -- BITs 6 y 7 del registro de control, pueden modificar el estado de las líneas $\overline{IRQA}$ ($\overline{IRQB}$) del MPU.

-Líneas de control CA1 y CB1. Estas dos líneas sólo pueden actuar como entrada y su función es programada en los - BITs 0 y 1 del registro de control (CRA, CRB). Al detectarse un cambio de nivel en estas líneas, es afectado el estado del BIT7 del registro de control correspondiente y opcionalmente- también a la señal de $\overline{IRQ}$. La siguiente tabla indica los modos posibles.

USO DE CA1 (CB1)

BIT 1 CRA (CRB)	BIT 0 CRA (CRB)	LÍNEA CA1 (CB1)	BIT 7 CRA (CRB)	$\overline{IRQA}$ ($\overline{IRQB}$)
0	0	↓	NIVEL UNO EN ↓ DE CA1 (CB1)	INHIBIDO
0	1	↓	NIVEL UNO EN ↓ DE CA1 (CB1)	NIVEL CERO CUANDO BIT 7 ES UNO
1	0	↑	NIVEL UNO EN ↑ DE CA1 (CB1)	INHIBIDO
1	1	↑	NIVEL UNO EN ↑ DE CA1 (CB1)	NIVEL CERO CUANDO BIT 7 ES UNO

El BIT7 de CRA (CRB) es limpiado cuando el MPU ejecuta - una operación de lectura en el registro de salida de datos: - ORA (ORB).

-Líneas de control CA2 y CB2. Los BITs 3, 4 y 5 del registro de control A (B) afectan el modo de actuación de estas líneas. CA2 y CB2 pueden programarse para que trabajen como- entrada o como salida. Cuando estas dos líneas de control actúen como entrada, un cambio de nivel en ellas modificará el- BIT6 del registro de control correspondiente y, opcionalmente, también a la señal de $\overline{IRQ}$. Para que CA2 o CB2 puedan ser- usadas como entrada, el BIT5 del CRA o CRB deberá ser nivel- cero, como se muestra en la siguiente tabla.

CA2 (CB2) COMO ENTRADA

BIT 5 CRA(CRB)	BIT 4 CRA(CRB)	BIT 3 CRA (CRB)	LÍNEA CA2 (CB2)	BIT 6 CA2 (CB2)	IRQA (IRQB)
0	0	0	⌐	NIVEL UNO EN ⌐ DE CA2 (CB2)	INHIBIDO
0	0	1	⌐	NIVEL UNO EN ⌐ DE CA2 (CB2)	NIVEL CERO CUANDO BIT6 ES UNO
0	1	0	⌐¬	NIVEL UNO EN ¬ DE CA2 (CB2)	INHIBIDO
0	1	1	⌐¬	NIVEL UNO EN ¬ DE CA2 (CB2)	NIVEL CERO CUANDO BIT6 ES UNO

El bit6 de CRA (CRB) es limpiado cuando el MPU ejecuta - una operación de lectura en el registro de salida de datos -- ORA (ORB).

-La función de CA2 y CB2 como salidas son un poco dife-- rentes; en este caso el BIT5 de CRA (CRB) deberá ser nivel -- uno. CA2 (CB2) pueden, como salidas, actuar en uno de tres mo dos:

1o. Si el BIT4 de CRA (CRB) es igual a nivel uno CA2 -- (CB2) seguirá el estado contenido en el BIT3 de CRA (CRB).

-Si el BIT4 de CRA (CRB) es nivel cero, CA2 (CB2) puede- ser usada en alguno de los dos modos restantes. El funciona- miento de las dos secciones A y B difieren para estos dos ti- pos de operación.

2o. BIT3 de CRA (CRB) nivel cero.

CA2 es llevada a nivel cero por efecto de un frente de - onda negativa del pulso de E (ENABLE), después de una opera-- ción de lectura del MPU en el registro de salida (ORA). CA2- retorna a nivel uno cuando ocurre el próximo frente de onda - de CA1.

CB2 es llenada a nivel cero por efecto de un frente de - onda positivo del pulso de E, después de una operación de es- critura del MPU en el registro de salida (ORB). CB2 retorna- a nivel uno cuando ocurre el próximo frente de onda de CB1.

3o. BIT3 de CRA (CRB) nivel uno.

CA2 es llevada a nivel cero por efecto de un frente de -

onda negativo del pulso de E, después de una operación de lectura del MPU en el registro de salida (ORA) CA2 retorna a nivel uno en el próximo frente de onda negativa del pulso de E.

CB2 es llevada a nivel cero por efecto de un frente de onda positivo del pulso de E, después de una operación de escritura del MPU en el registro de salida (ORB). CB2 retorna a nivel uno en el próximo frente de onda positivo del pulso de E.

Ejemplos del uso de la PIA se verán en el capítulo VI.

CAPITULO II

Un sistema computador está integrado por dos porciones - heterogéneas, pero sumamente ligadas; ambas son indispensa- - bles para conformar al sistema.

La parte tangible es implementada de acuerdo a los objetivos que persigue el diseñador y a los elementos a su disposición, esta parte está compuesta por los dispositivos físicos agrupados en la tarjeta del computador y recibe el nombre de HARDWARE.

La parte intangible es la encargada de que la sección física sea aprovechada al máximo, dando el mayor número de facilidades al usuario para la operación del sistema; los programas, subrutinas, etc., componen esta parte llamada SOFTWARE

CAPITULO IIA

HARDWARE

El sistema AMI 6800 está implementado en una sola tarjeta de circuito impreso, en ella están contenidos todos los -- dispositivos mencionados en el capítulo I.

La organización interna del sistema se muestra en la figura II A.1. Podemos observar que, cuenta con dos conectores de acceso; uno de ellos, el B, contiene las líneas para los - elementos de entrada y/o salida (I/O); el A, se emplea como - derivación de los BUSES de datos, direcciones y control.

Los elementos asociados al MPU y los dispositivos de soporte son adicionados de acuerdo al objetivo que persigue el - sistema; como AMI 6800 fue diseñado como un computador de evaluación y desarrollo, contiene diversas opciones de gran utilidad en la implementación y diseño de sistemas de uso más -- particular. Obviamente, también contiene los elementos nece-

sarios para el funcionamiento en sí del MPU y los dispositi-- vos de soporte, como son: los generadores de los relojes Ø1 y Ø2, el generador del pulso de RESET, etc.

Algunas de las caracterfsticas adicionales son las si- - guientes:

1.- Sistema de acceso directo a memoria (DMA).

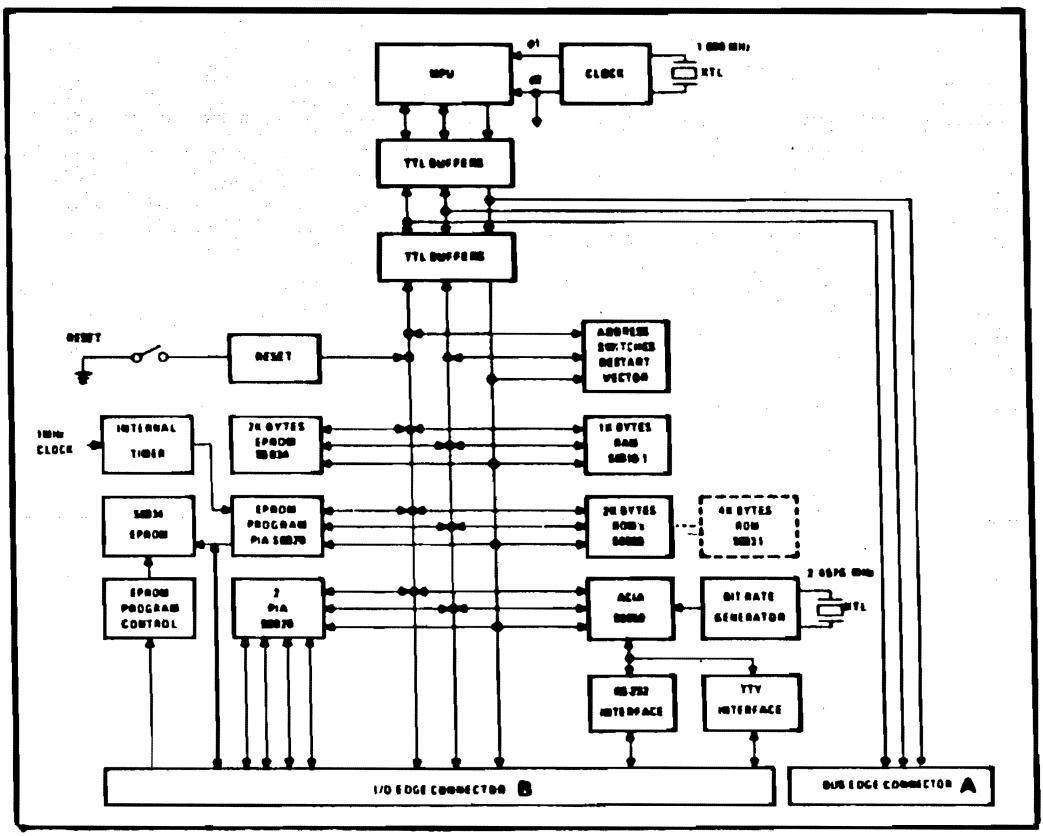


FIGURA II A.1.- DIAGRAMA DE BLOQUES DEL AMI 6800.

2.- Una base de tiempo interna (accesibles las señales - de un mseg y 100 microseg).

3.- Un programador de EPROMs interconstruído.

4.- Interfases para un teletipo con acoplador de 20 - - mamp y RS232C.

5.- Tres opciones para generar Ø1 y Ø2.

6.- Doble "BUFFEREADO" para todas las líneas del BUS de- datos y de direcciones.

7.- 512 BYTES de RAM se pueden localizar en la parte alta o baja del rango de selección, etc.

Se detallará sólo en las más importantes.

Circuito de reloj para S6800:

AMI cuenta con tres opciones para generar las señales de reloj, que cumplen con los requerimientos para el buen funcionamiento del procesador (MPU).

a) Generado por dos monoestables. Esta opción cumple -- con las características mencionadas en el capítulo I. Su forma se muestra en la figura II A.2. El no traslape de Ø1 y Ø2, es generado por los retardos de propagación propios de los monoestables. Como el MPU requiere de niveles específicos muy - cercanos a Vcc y referencia, es necesario un circuito auxi--- liar que consiste en dos transistores acoplados complementa-- riamente, que generan los niveles exigidos. Estos transistores se caracterizan por tener voltaje colector-emisor de saturación muy baja, VCE sat típico de 0.05 VOLTS dc. La figura- II A.2b muestra la configuración.

b) La segunda opción consiste en generar la base de tiempo por medio de un oscilador de cristal, éste se añade al subsistema anterior de la siguiente manera: en la figura II A.2A se muestra el SWITCH de cambio que selecciona la realimenta-- ción del segundo monoestable al primero o la señal proveniente del generador de cristal. Esta selección incluye a los - monoestables, ya que gracias a ellos se obtiene el retardo -- que hace el no traslape de fases, además de poder ajustarse - los ciclos de trabajo de cada una de las señales Ø1 y Ø2. La- figura II A.3 muestra esta opción.

c) La última alternativa consiste en un generador externo, pudiendo conectarse a través de la línea EXTERNAL CLOCK - disponible en el conector de acceso. La figura II A.3 mues-- tra cómo seleccionarla.

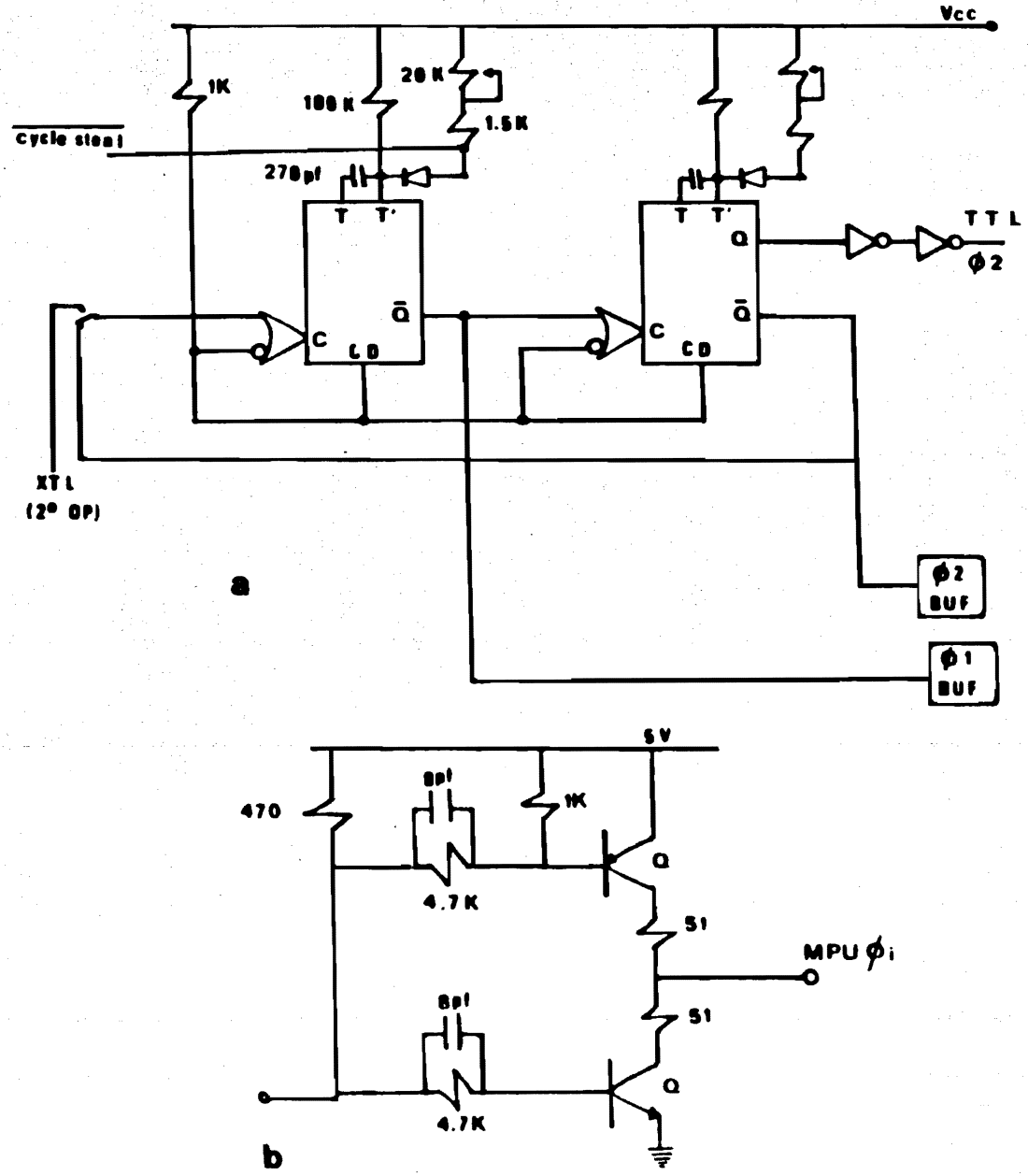


FIGURA II A.2.- a) Generador de señal de reloj.
b) BUFFER para la señal Ø I.

Circuito generador del pulso de RESET y secuencia de - - RESTART.

La figura II A.4 muestra el circuito que genera el pulso de RESET, se observa que es muy sencillo, ya que consiste solamente en un dispositivo integrado implementado como monoestable. Este se dispara con un SWITCH manejado por el usuario en el exterior del microcomputador.

El microcomputador AMI S6800 ofrece la capacidad de capturar, durante la secuencia de RESET, una dirección establecida en los interruptores que se encuentran en la tarjeta, la figura II A.5 nos muestra el circuito empleado.

El funcionamiento de esta sección es como sigue:

a) Si se desea dar comienzo al sistema, se tienen que ordenar los interruptores, de tal manera que, indiquen la primera dirección donde se ubica el programa monitor. Esta dirección es la F000)h.

b) Una vez iniciado el ACIA, se puede direccionar a cualquier localidad de memoria; pudiendo hacerlo, por ejemplo, a la dirección de inicio del BASIC o a la del ensamblador - - (E000 ó E802 respectivamente).

c) En ambos casos, cuando se presiona el SWITCH de RESET, el microprocesador carga los contenidos de las direcciones altas (FFFE, FFFF)h y los almacena en el contador de programa, para así iniciar en la dirección indicada por los interruptores; al mismo tiempo inhabilita el resto de la memoria, de tal modo que, los selectores de 2 a 1 (74S257) actúan como memoria de sólo lectura, es decir, sólo están habilitados cuando R/W tiene nivel uno -debido a que la salida de los selectores es de tres estados- la dirección más baja, o sea, Ao sirve para seleccionar los BITs más significativos o menos significativos cuando su nivel es cero o uno respectivamente.

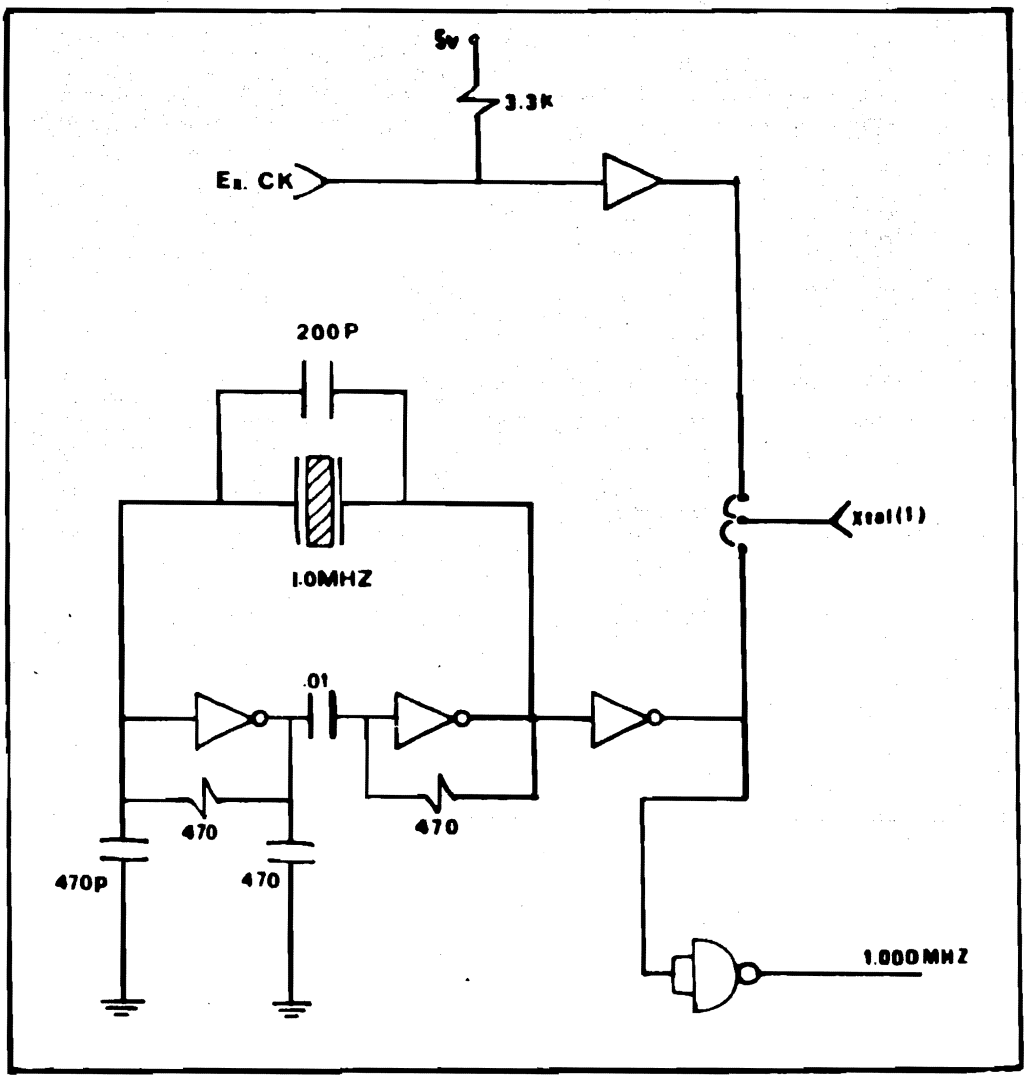


FIGURA II A.3.- OSCILADOR DE CRISTAL Y LA OPCION A RELOJ EXTERNO.

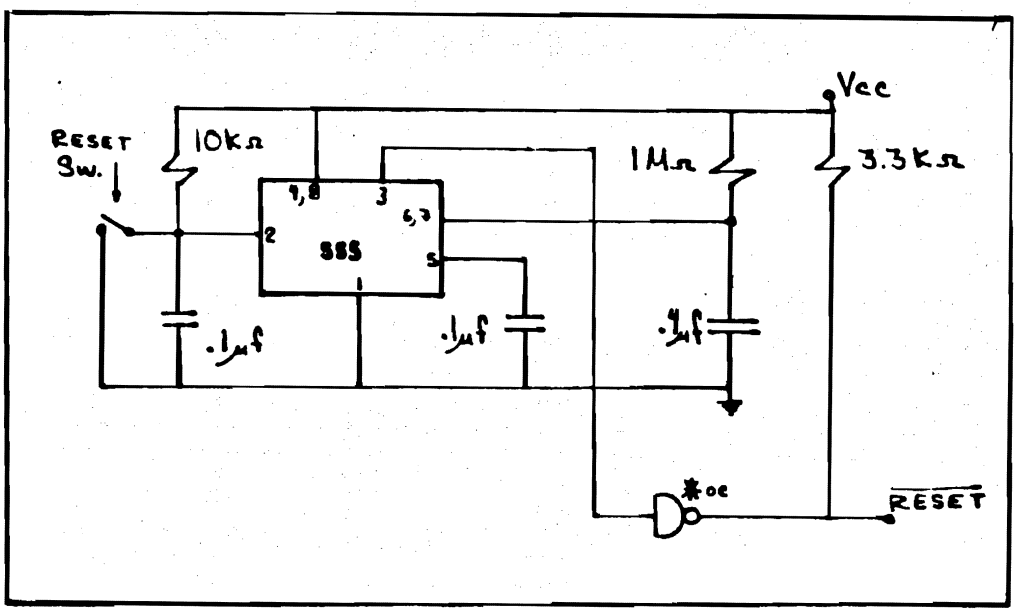


FIGURA II A.4.- CIRCUITO GENERADOR DEL PULSO DE RESET.

MEMORIA:

AMI 6800 contiene 1024 BYTEs de RAM, 512 de ellos se hallan direccionados en la parte alta del rango de selección, desde la dirección FE00)hex a FFFF)hex. Los otros 512 se ubican desde la dirección FC00)hex a FDFF)hex, inclusive, u opcionalmente en la parte inferior del rango, es decir, desde la dirección 0000)hex a 01FF)hex, ésto, gracias a un interruptor selector (LOW o HIGH RAM) que condiciona en HIGH a que las líneas de dirección A10 a A15 sean nivel uno o, cuando es LOW a que A15 sea nivel cero.

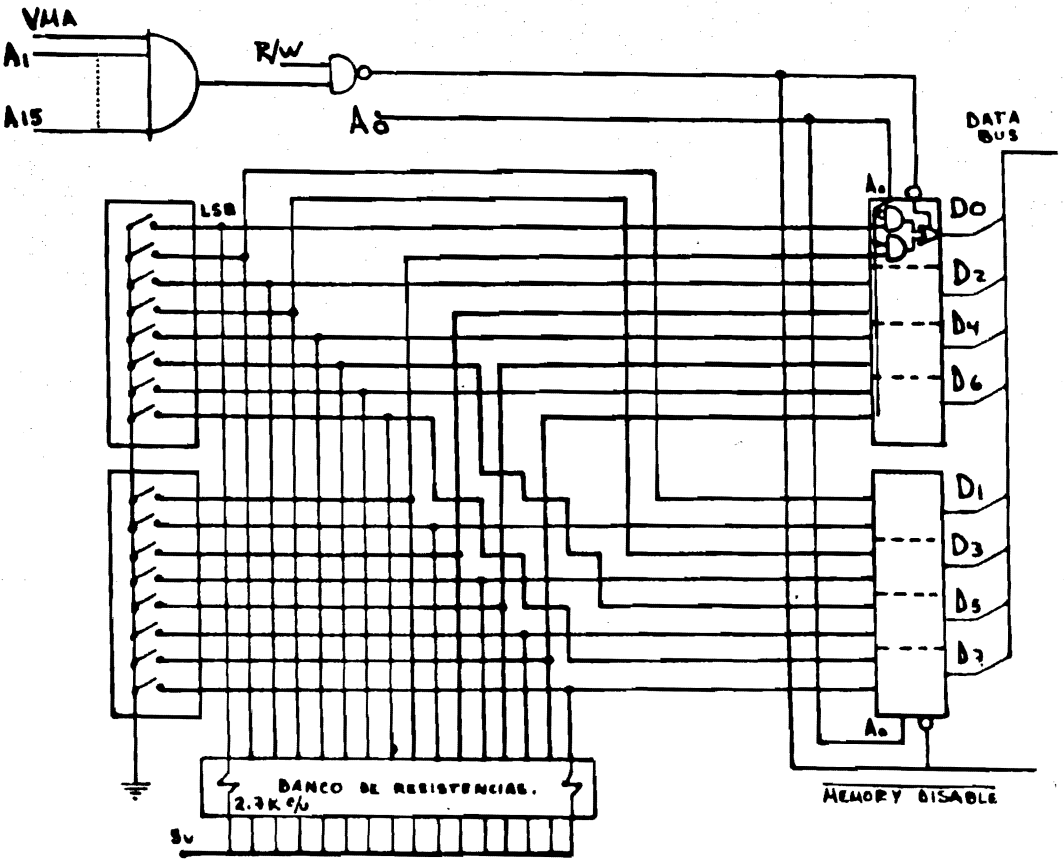


FIGURA II A.5.- DIAGRAMA DEL CIRCUITO DE RESTART.

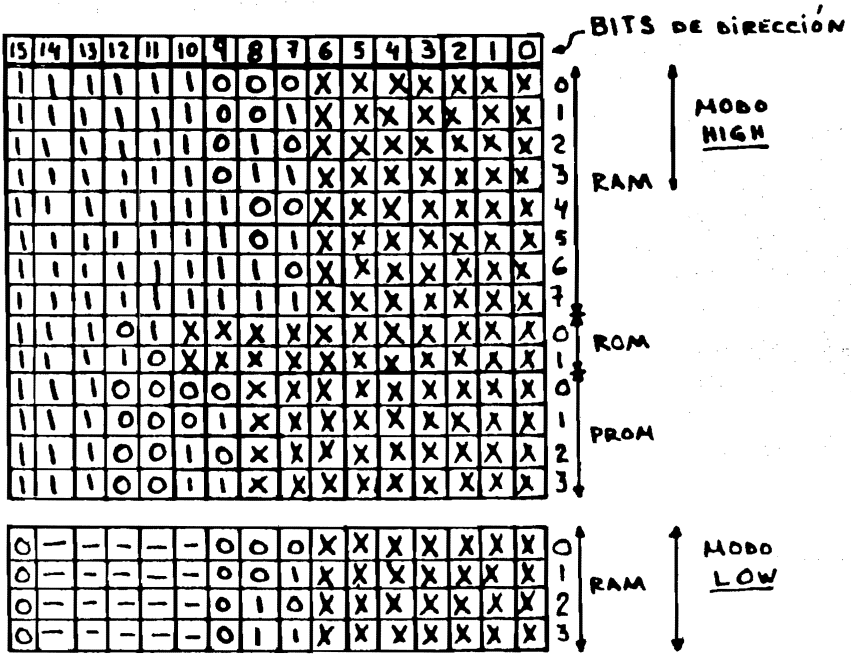


FIGURA II A.6.- TABLA DE ORGANIZACION DE LA MEMORIA.

En lo correspondiente a ROM AMI puede contener dos dispositivos S6831 (2K BYTEs c/u), el primero de ellos ubicado desde la dirección E800)hex a EFFF)hex, inclusive, en él está almacenado en forma permanente el programa ensamblador. El segundo ROM está direccionado a partir de la localidad F000)hex a F7FF)hex, inclusive y, éste contiene el programa monitor.

La tarjeta tiene lugar para cuatro PROMs S6834 (512 BYTEs c/u), las direcciones de inicio y fin de éstos son: E000)hex y E7FF)hex, respectivamente; en ellos está programado el lenguaje BASIC.

Decodificación de memoria:

Los dispositivos encargados de hacer la decodificación de los elementos de memoria, son los CIs 74S138 (CI 44 y 54), estos son decodificadores de 3 a 8 líneas, con tres entradas de habilitación.

Como son 8 los elementos RAM, se requieren de 3 líneas adicionales para discriminarlos, estas señales son: A7, A8 y A9 conectadas a las entradas del CI 44, además, éste sólo será habilitado cuando el resto de las líneas de dirección (A10 a A15) sean nivel uno (salida 7 del CI 54), en el modo de HIGH RAM. En el modo de LOW los RAMs 0, 1, 2, 3, serán habilitados cuando A15 sea nivel cero.

Para decodificar la memoria programable y de sólo lectura (PROM y ROM) se emplea el CI54. En los dos elementos ROM las señales discriminadoras provienen de las líneas A11 y A12 del MPU, de tal modo que, cuando A11 sea nivel uno y A12 nivel cero se habilitará el ROM que contiene al ensamblador. Cuando A11 sea nivel cero y A12 sea nivel uno se habilitará el ROM que contiene el programa monitor, ambos dispositivos quedarán habilitados cuando A13, A14 y A15 sean nivel uno, ya que éste facultará al CI 54, que a su vez capacitará el EO de ambos ROM's cuando corresponda.

Como los elementos PROM son cuatro y están localizados desde la dirección E000)h, se requiere que A13, A14 y A15 sean nivel uno y, además, que las líneas A11 y A12 sean nivel cero, ver figura II A.6. En este caso, las señales discriminadoras serán A9 y A10, utilizando sus cuatro combinaciones, de tal modo que, sigan la siguiente tabla de verdad.

		PROM SEL			
A10	A9	P0	P1	P2	P3
0	0	H	I	I	I
0	1	I	H	I	I
1	0	I	I	H	I
1	1	I	I	I	H

Donde:

I = inhibido

H = habilitado cuando $\overline{CS}$ PROM es nivel cero.

El CI 54 nos proporciona la decodificación mencionada en el párrafo anterior, de la siguiente manera:

Cuando: A15 = A14 = A13 = Nivel uno

A12 = A11 = Nivel cero

a) A10 nivel cero

Q0 nivel cero

Q1 nivel uno

b) A10 nivel uno

Q0 nivel uno

Q1 nivel cero

Y según la tabla de verdad:

			$\overline{CS}$ PROM			
Q1	Q0	A9	0	1	2	3
0	0	0	X	X	X	X
0	0	1	X	X	X	X
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Donde:

X = No importa

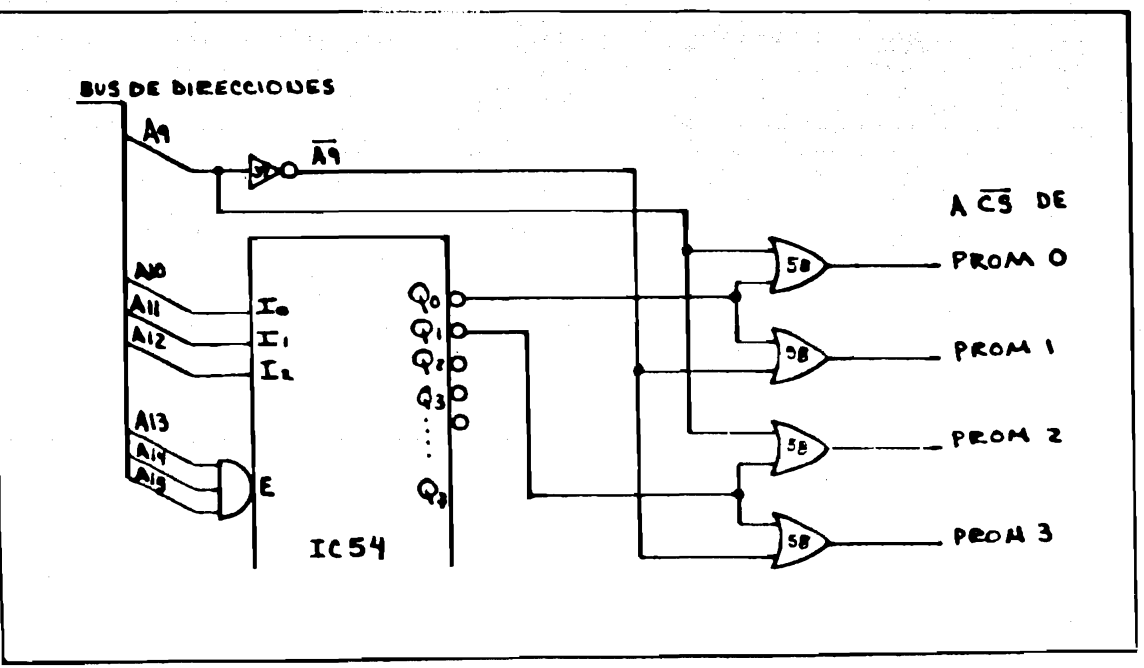
Podemos obtener el diagrama lógico de decodificación, resultando:

$\overline{CS}$ PROM 0 = A9 + Qo

$\overline{CS}$ PROM 1 = $\overline{A9}$ + Q0

$\overline{CS}$ PROM 2 = A9 + Q1

$\overline{CS}$ PROM 3 = $\overline{A9}$ + Q1



Las salidas Q6 y Q7 del CI 54 indican, cuando ambos sean nivel uno, que se está realizando una operación de lectura en un dispositivo PROM ó ROM. Estas mismas salidas indican, - - cuando alguna de ellas es nivel cero, que se está accediendo- alguna localidad en RAM. Por lo tanto, la conjunción negada- de ambas nos indicará:

Q7	Q6	SM
0	0	X
0	1	1
1	0	1
1	1	0

X = No se presenta

Pues bien, SM y R/W nos sirven para generar tres señales de control utilizadas en el sistema de BUFFERs, en el BUS de-datos. Cuando el MPU lee los elementos RAM, PROM o ROM, R/W- debe ser nivel uno y, cuando escribe en RAM, R/W debe ser nivel cero, como lo muestra la siguiente tabla:

SM	R/W	RDO	RDI	RBE1
0	0	1	1	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

donde:

RDO: RAM DATA OUT
RDI: RAM DATA IN
RBE: ROM BUS ENABLE 1

Las ecuaciones son:

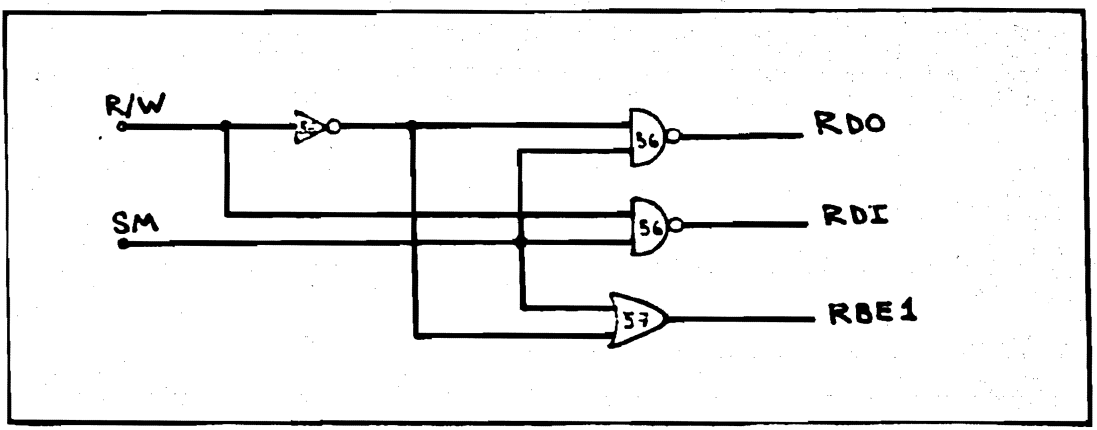
$\overline{RDO} = SM \cdot \overline{R/W}$

$\overline{RDI} = SM \cdot R/W$

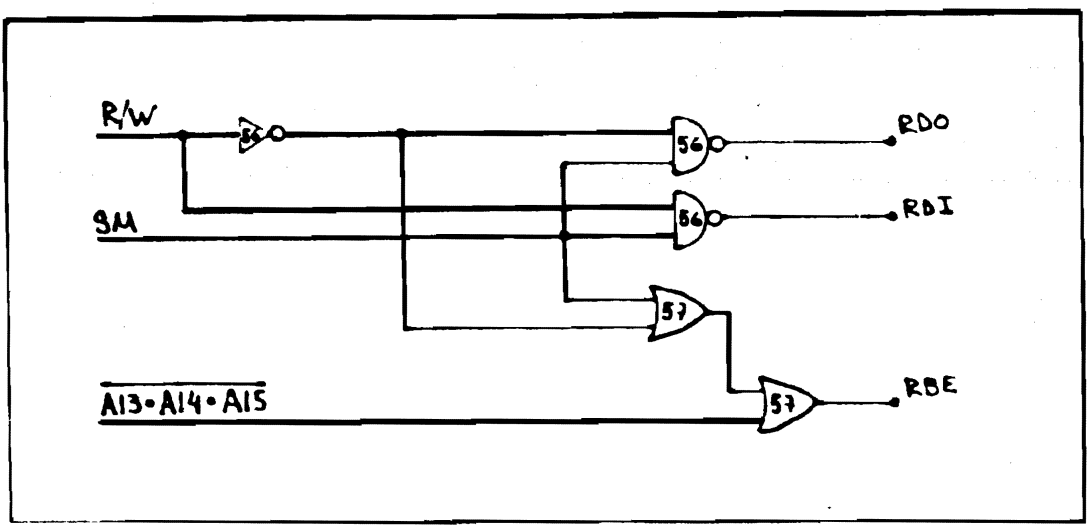
$\overline{RBE1} = SM \cdot R/W$

$RBE1 = SM + \overline{R/W}$

y el diagrama lógico:



Para asegurar que se está efectuando una lectura en - PROM ó ROM A13, A14 y A15 se conjugan y se alternan con RBE1. Para generar el comando RBE, que será empleado para habilitar el BUS de datos en el modo de lectura en ROM o PROM, el dia-- grama final será como sigue:



Acceso directo a memoria:

Este modo de acceso se utiliza cuando la velocidad de - transmisión de datos, relativa al MPU y periféricos involucra dos, es lenta o impráctica, de tal manera que, esta operación

es independiente de la actividad del procesador.

Cuando el MPU ejecuta una instrucción normal, éste tiene completo dominio de los BUSES de dirección y datos, definiendo que localidad de memoria será accedida para leer o para escribir. Sin embargo, se puede lograr que el MPU libere los BUSES antes mencionados, pudiéndose emplear en otras actividades, como la de acceso directo a memoria (DMA).

AMI cuenta con tres opciones para generar un acceso directo a memoria (DMA): detener el procesador (HALT); "alargar" el tiempo del ciclo (CYCLE STEAL) y opción múltiple o multiplex (MULTIPLEX O MUX).

En el modo de HALT el MPU abandona las líneas de datos y dirección, después de haber completado la instrucción en que la línea de $\overline{\text{HALT}}$ cambió a nivel cero. La figura II A7 ilustra los elementos involucrados en el DMA usando la señal de $\overline{\text{HALT}}$; el diagrama de tiempo se ilustra en la figura II A8.

La solicitud de DMA (DMA REQUEST) debe estar sincronizada con el frente de onda positiva de Ø1. Una vez que el acceso directo a memoria ha sido aceptada ($\overline{\text{DMA GRANT}}$), el dispositivo de DMA puede tomar el tiempo necesario para hacer la transferencia a memoria. La señal de $\overline{\text{DMA GRANT}}$ inhabilita la sección de BUFFERS conectados al procesador (ver figura II A11); tanto el BUS de datos como el de direcciones quedarán en el modo de alta impedancia, las señales RAM DATA OUT, RAM DATA IN y ROM BUS ENABLE, que controlan la sección de BUFFERs auxiliares, dependerán de la señal R/W del dispositivo de DMA. Además $\overline{\text{DMA GRANT}}$ también pone en el estado de alta impedancia las señales de $\overline{\text{VMA}}$ y R/W (CI 64), pudiéndose entonces, dar estos comandos externamente.

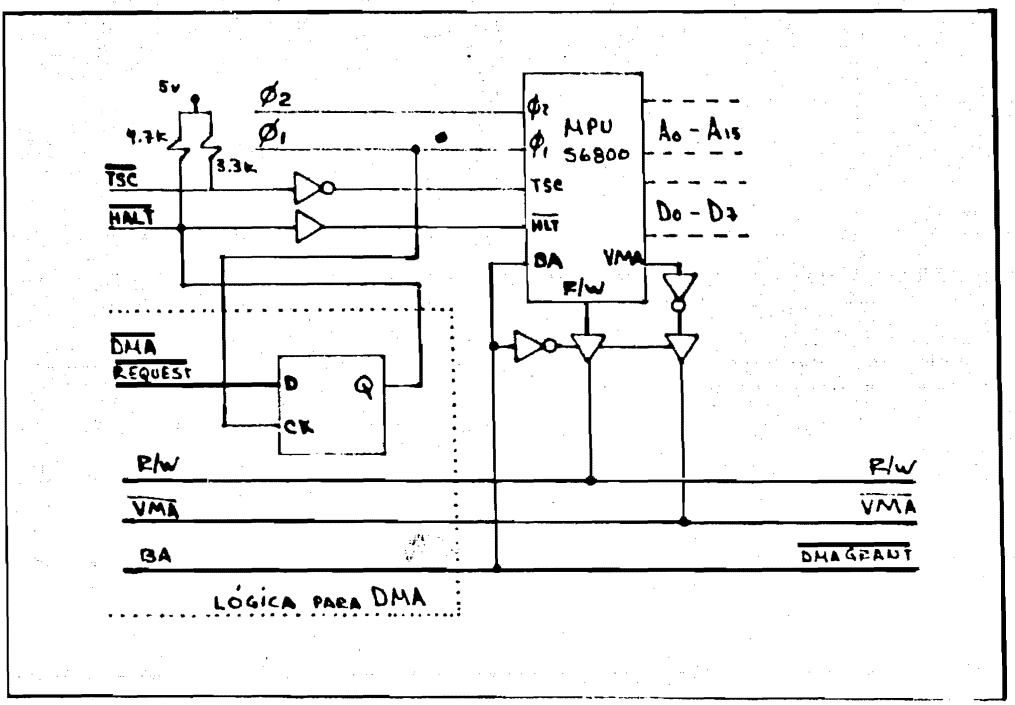


FIGURA II A7.- DMA usando la señal de HALT

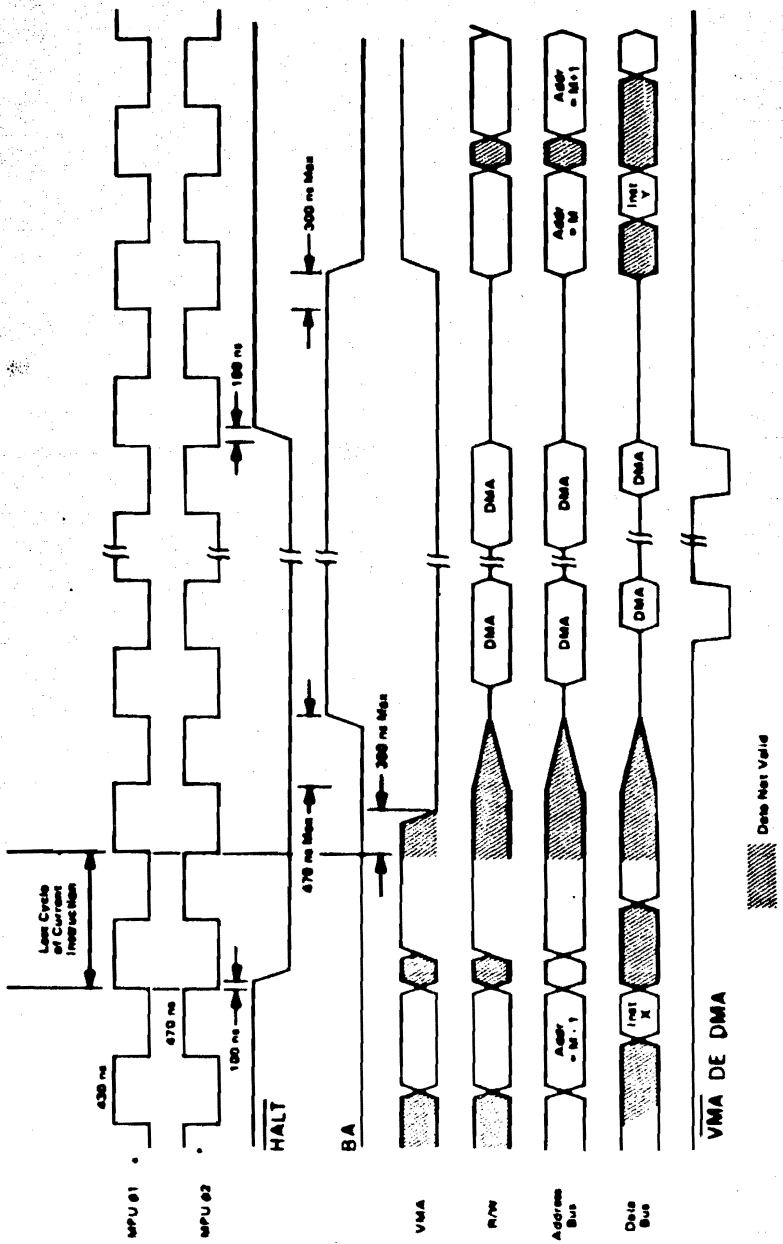


FIGURA II A8.- DIAGRAMA DE TIEMPO PARA HALT

La siguiente opción corresponde al uso de la línea CYCLE STEAL, mostrada en la figura II A2, y la línea de entrada TSC; ver figura II A7. El propósito es lograr que el ciclo de trabajo de Ø1 sea alargado hasta cuatro microsegundos de duración, durante los cuales, la señal de TSC podrá ser usada para deshabilitar el control que el MPU ejerce sobre el BUS de direcciones. Como DBE (usado para controlar el BUS de datos) se activa sólo durante Ø2, el periférico tiene hasta tres microsegundos de DMA en cada ciclo del MPU. El diagrama de tiempo es mostrado en la figura II A9.

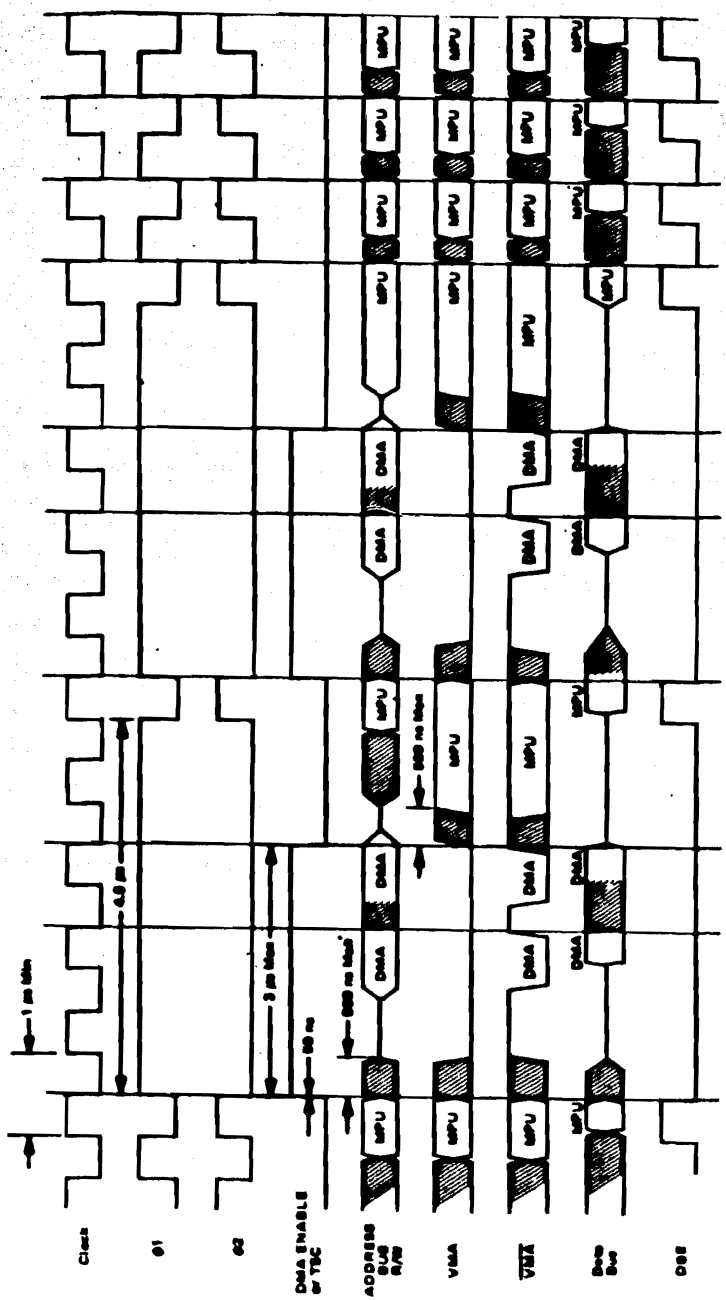


FIGURA II A9.- DIAGRAMA DEL TIEMPO PARA CYCLE STEAL.

La última opción es MUX DMA (múltiplex DMA) y es una variación del CYCLE STEAL DMA; esta técnica requiere de un sistema de regular complejidad, aunque una parte está integrado en la tarjeta AMI. La figura II A10 muestra un diagrama de bloque de esta operación, y la figura II A11 indica simplificadamente cómo se halla implementado en AMI. Durante Ø2, el MPU controla los BUSES y durante Ø1, los controla el periférico. De este modo, el MPU realiza su trabajo intercalando acceso a memoria con la operación de DMA. Este método requiere de elementos de memoria rápidos, de tal modo que, si no se cuenta con ellos, se deberá bajar la frecuencia de operación del sistema, para así proveer de un adecuado retardo de tiempo de acceso para la memoria actual. La figura II A12 muestra el diagrama de tiempos para el MUX DMA. Durante Ø1 nivel uno, los BUFFERs asociados a Ø1 son habilitados y los asociados con Ø2 se llevan al estado de alta impedancia. El DMA SYNCH (Ø1) ocasiona que el canal de DMA haga válidas las direcciones, R/W, VMA y las señales de datos en la interfase de DMA. Cuando Ø2 cambia a nivel uno, los BUFFERs, desde la interfase de DMA, son puestos en el estado de alta impedancia, y los del MPU son habilitados, de tal manera que, el MPU generará ahora las señales de R/W y VMA.

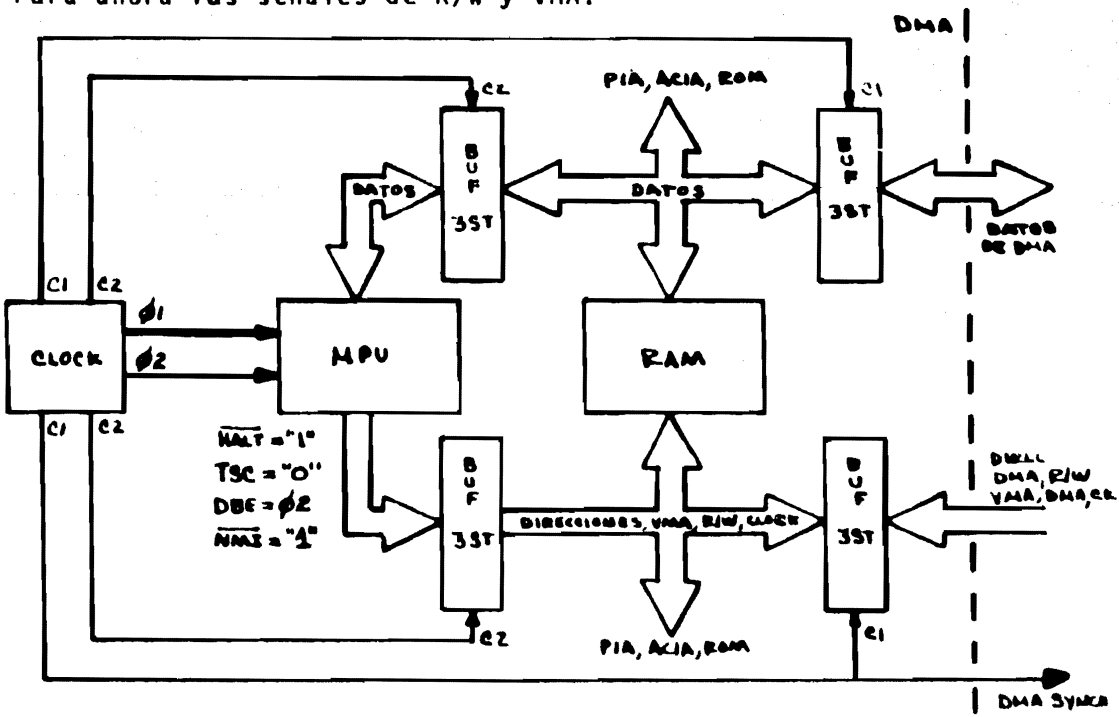


FIGURA II A10.- DIAGRAMA DE BLOQUE DE MUX DMA.

Esta técnica es, también, fácilmente adaptada en sistemas que requieren de dos MPUs conectados a la misma memoria.- En este caso, los dos MPUs trabajarán con relojes desfasados- 180°, de manera que, Ø1 para un MPU es Ø2 para el otro y viceversa.

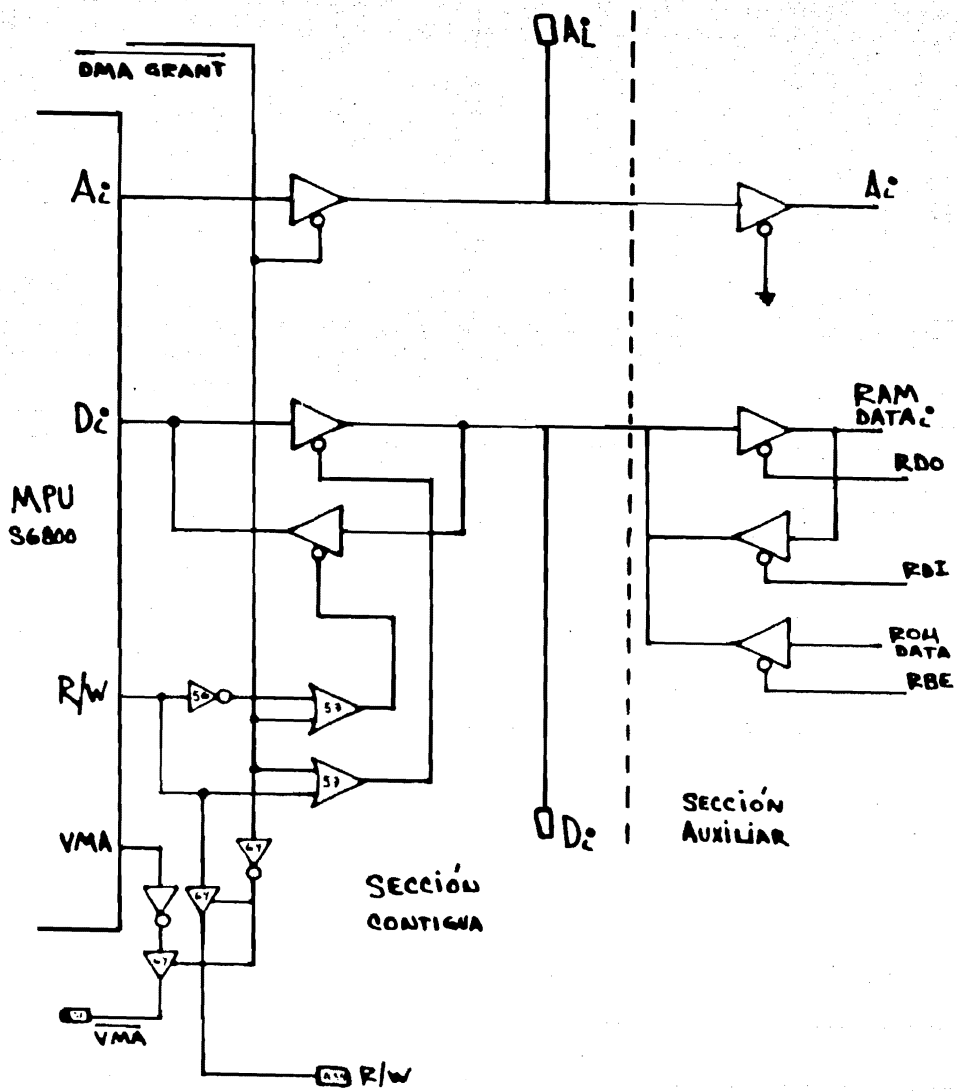


FIGURA II A11.- DIAGRAMA SIMPLIFICADO DE LOS BUFFERs DEL MPU.

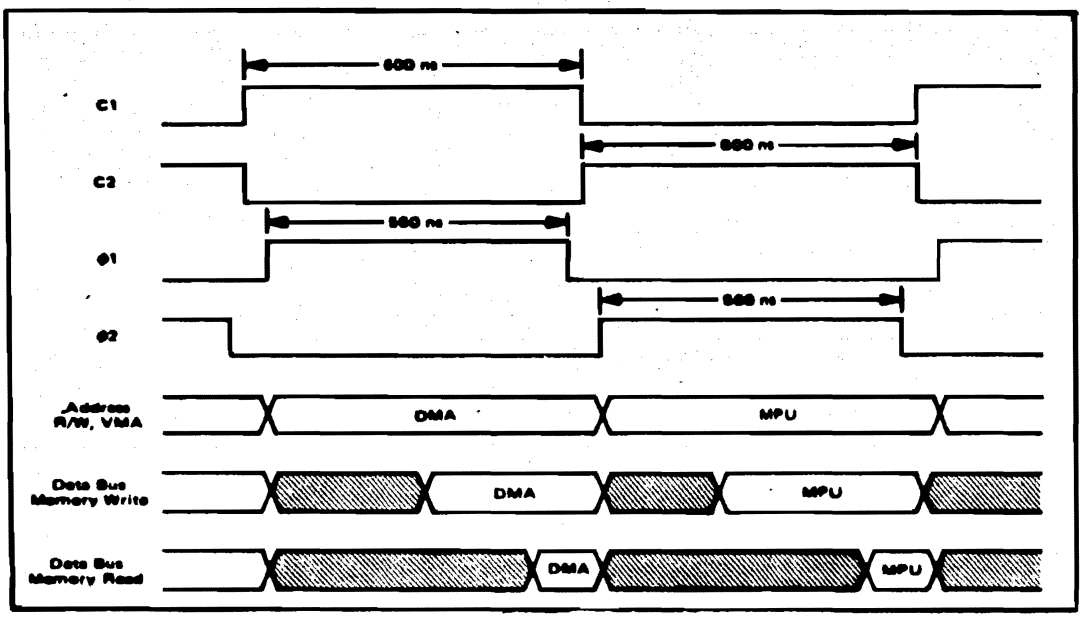
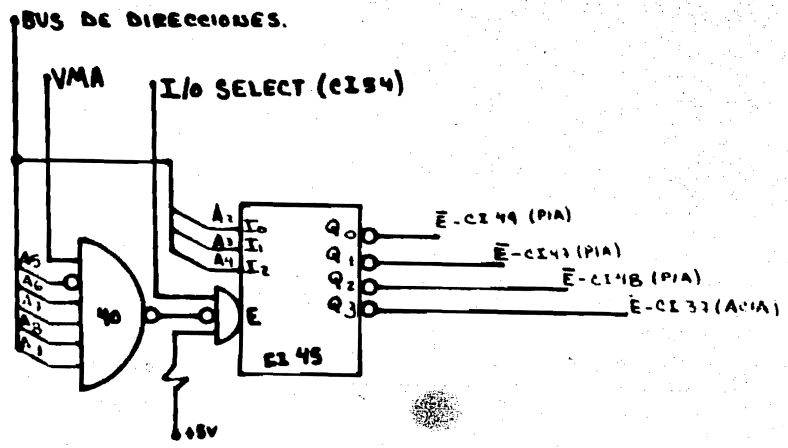


FIGURA II.A.12.- DIAGRAMA DE TIEMPOS PARA MUX DMA.

Dispositivos acopladores de entrada/salida (I/O).

El sistema cuenta con cuatro dispositivos de I/O, tres - de ellos para recibir/transmitir datos en paralelo (PIA) y - una para recibir/transmitir datos en serie (ACIA). Los cua-- tro elementos están direccionados como muestra el mapa de I/O de la figura II.A.13; en ésta misma, se encuenta el diagrama- lógico de la decodificación; el CI 45 se encarga de realizar- lo.

Dos de las PIAs (1,2) son direccionables y sus salidas - se usan en tareas varias. La PIA (3) restante se utiliza pa- ra generar las señales necesarias en el circuito programador- de PROMs, auxiliado sólo por una línea proveniente de la PIA2 (CA2). El circuito del programador así como todo el sistema- se muestra en el apéndice.



CI	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
49	1	1	1	1	1	0	1	1	1	1	0	0	0	0	X	X
47	1	1	1	1	1	0	1	1	1	1	0	0	0	1	X	X
48	1	1	1	1	1	0	1	1	1	1	0	0	1	0	X	X
37	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1	X

BITS DE DIRECCIÓN

DE I/O SELECT (CI 54) — DE (CI 40)

X= BIT VARIABLE

CI 49 FBC0-FBC3
CI 47 FBC4-FBC7
CI 48 FBC8-FBCB
CI 37 FBCE-FBCF

FIGURA II A13.- DIAGRAMA LOGICO DE DECODIFICACION DE I/O Y MAPA DE ASIGNACION DE DIRECCION PARA - I/O

En la parte periférica del ACIA se hallan implementadas las interfases para TTY 20MA y RS232. El circuito de la figura II A14 muestra dichas interfases.

Las líneas TTY IN (+) y TTY IN (-) se emplean como entradas de señales provenientes de un teletipo con interfase de 20 MA (generalmente un SWITCH que abre y cierra de acuerdo a la palabra enviada), para manejar al transistor Q5.

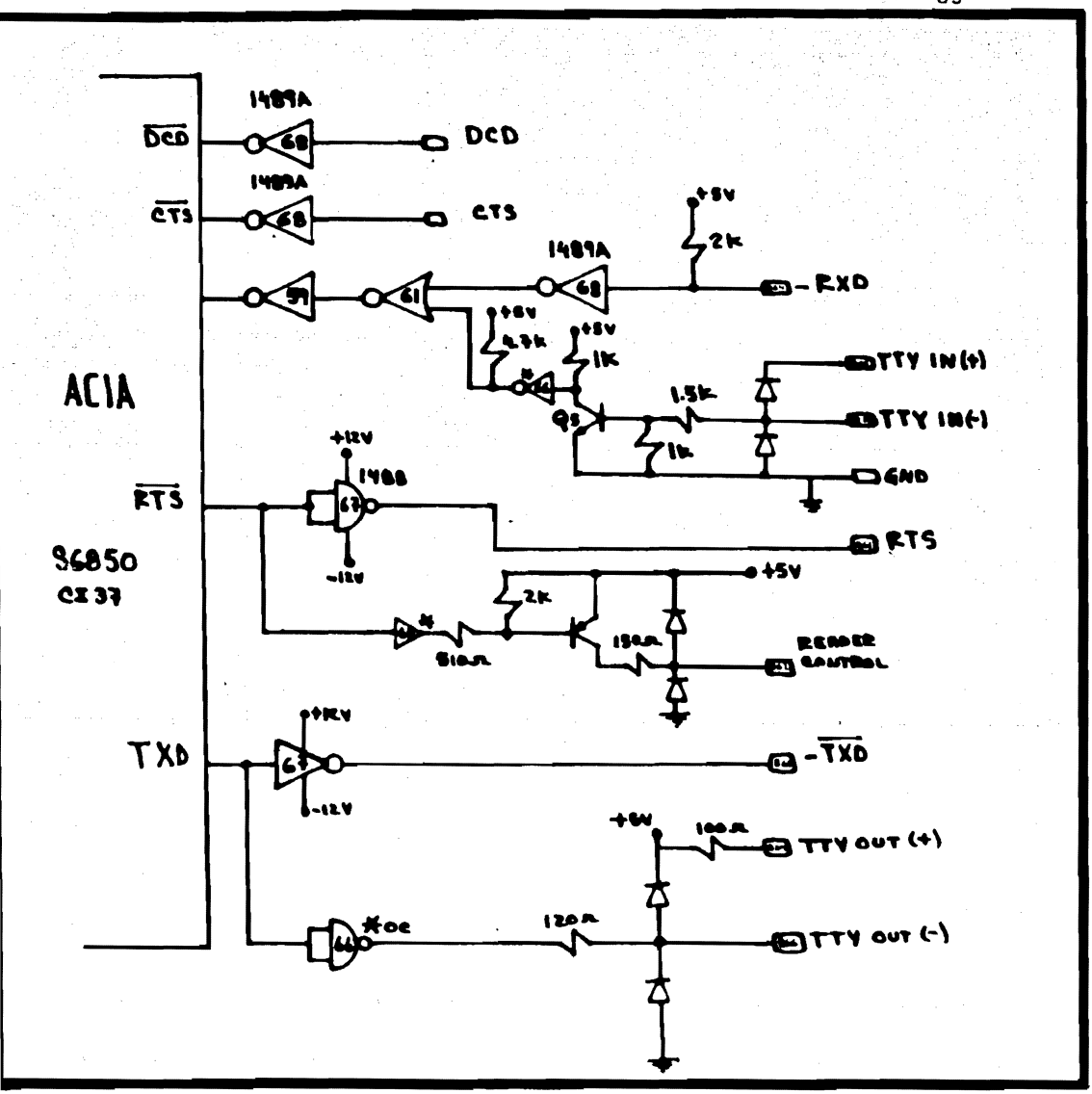


FIGURA II A14.- DIAGRAMA DEL CIRCUITO DE LA INTERFASE - TTY 20 MA y RS232 ACOPLADOS AL ACIA.

Las líneas TTY OUT (+) y TTY OUT (-) se utilizan para enviar la información hacia un teletipo con interfase de 20 MA- (generalmente la bobina de un relé que se activa de acuerdo a la palabra recibida). La señal READER CONTROL puede ser usada para manejar la bobina de un relé que acciona el mecanismo de la lectura de cintas en un teletipo.

Las líneas restantes se emplean en el manejo de un terminal con interfase RS232; la salida de datos se efectúa mediante el dispositivo 1488 (IC 67), que convierte niveles TTL a +12V y -12V, requeridos en este tipo de acopladores, la señal de RTS también es transformada de este modo. La entrada de datos, así como DCD y CTS, son modificadas de niveles de voltaje de +12 y - VOLTS a nivel TTL mediante el dispositivo 1489 (IC 68).

CAPITULO II B

SOFTWARE

Todo sistema computador cuenta con programas que le auxilian en el control y supervisión de los elementos que lo forman, estos programas pueden ser tan complejos como los diseñadores quieran y como las necesidades lo demanden. En computadores de mediano y gran tamaño, este conjunto de programas reciben el nombre de sistema operativo; sus funciones van desde la aceptación, ejecución y presentación de problemas, además de controlar etapas intermedias como pueden ser: la compilación, decodificación, etc.

En computadores pequeños, sobre todo en modelos prototipo, los programas incluidos tienen sólo las subrutinas necesarias para arrancar al sistema y realizar algunas de las tareas antes mencionadas; se les suele denominar programas monitor.

En AMI, este programa está residente en el elemento ROM-S6831 que inicia en la dirección F000)h. Una parte del programa está dedicada a establecer la comunicación con el usuario y, sus subrutinas están diseñadas para ser llamadas por entero vía un teletipo o similar. Esta parte recibe el nombre de PROTOTYPING OPERATING SYSTEM (PROTO) y el número de comandos que puede manipular son once. Un comando consiste en un identificador de comando (un carácter), seguido por parámetros adicionales, si son necesarios, separados por un espacio o coma, finalizando con una señal de CARRIAGE RETURN (CR).

Lista de comandos:

S, ADDR, BYTE 1, BYTE 2, ..., BYTE N.

El comando SET escribe las palabras de 8 BITS especificadas por BYTE 1 a BYTE N, dentro de un conjunto consecutivo de localidades de memoria iniciando en la dirección ADDR.

Ejemplo:

S00 86 05 97

Dirección	Contenido Inicial	Contenido Final
0000	X X	86
0001	X X	05

Dirección	Contenido Inicial	Contenido Final
0002	X X	97
0003	X X	X X

Donde X = No importa

Los caracteres subrayados indican que deben ser teclados por el usuario.

D, ADDL, ADDH.

El comando DISPLAY imprime el contenido de memoria dendel rango comprendido entre ADDL y ADDH, inclusive, en formato hexadecimal.

Ejemplo:

D 00 02

Salida al TTY

0000 86 05 97

R

El comando REGISTERS imprime el contenido de las localidades de memoria comprendidas entre la dirección FFEB)hex a - FFF3)hex, inclusive, que contienen el valor de los registros- CCR, ACCB, ACCA, X, PC y SP (en este orden) cuando el programa fue interrumpido por última vez.

Ejemplo:

R

D0 01 78 FB0A 0001 F720

G, ADDR

El comando GO comienza la ejecución del programa especificado por la dirección ADDR.

M ADDL, ADDH, DEST

El comando MOVE copia el contenido de memoria desde el - rango comprendido entre las direcciones ADDL y ADDH, inclusive, a las localidades en RAM comenzando en la dirección DEST.

L, ADDL, ADDH, OFFSET

El comando LOAD TAPE carga los datos desde una cinta con formato hexadecimal dentro de la memoria entre el rango ADDL- y ADDH, inclusive. El OFFSET, cuyo valor deberá ser en com-- plemento a dos y en hexadecimal, es sumado a la dirección de- memoria especificada en la cinta. Cuando el rango es omitido, PROTO asignará el rango completo (0000-FFFF).

Ejemplo:

L 00 101

Carga en memoria la información comprendida en el rango- de 000)h a 101)h contenida en la cinta; PROTO seguirá leyendo la cinta hasta encontrar un EOT (END OF TAPE) si se ha omiti- do el rango.

P, ADDL, ADDH, OFFSET

El comando PUNCH ocasiona que PROTO perfore una cinta de papel en el teletipo, cuyo contenido incluirá los datos en me moria comprendidos entre las direcciones ADDL y ADDH, inclusI ve. Al inicio de cada RECORD se perforará una dirección de - cuatro dígitos en hexadecimal. Esta dirección es derivada de la dirección de inicio del conjunto de BYTES que será perfora do más el valor del OFFSET. El OFFSET es opcional y, si es - omitido asumirá el valor cero.

Ejemplo:

P 00 10

Perfora el contenido de las localidades entre 00)h y - - 10)h, inclusive, con un formato hexadecimal, ver apéndice.

E

El comando de transmisión (END) es usado para causar un- carácter EOT que será perforado en la cinta de papel. Esto - indicará a PROTO, al momento de leer la cinta, que ha sido el último RECORD por leer.

Ejemplo: E

S9030000FC

B, ADDL, ADDH, ROMAD

El comando BURN copia el contenido de la memoria dentro- del EPROM colocado en el SOCKET del programador, comenzando -

con la localidad de memoria ADDL a ADDH, inclusive, a la localidad del EPROM comenzando con la dirección ROMAD. Cada BYTE es grabado con 20 pulsos de -50 VOLTS de 3 Mseg. de duración, aplicados en la terminal correspondiente en el EPROM SOCKET.

I, ADDL, ADDH, ROMAD

El comando INPUT copia el contenido de un EPROM, colocado en el SOCKET del programador, dentro de la memoria comenzando en la dirección ADDL a ADDH, inclusive, desde la dirección del EPROM ROMAD.

V, ADDL, ADDH, ROMAD

El comando VERIFY compara el contenido de memoria entre ADDL y ADDH, inclusive, con las correspondientes localidades en el EPROM colocado en el SOCKET del programador, comenzando con la dirección del EPROM ROMAD. Si hay alguna discrepancia PROTO la indica.

El sistema dará la opción al usuario de emplear cualquier comando una vez presionado el SWITCH de RESET, PROTO contestará imprimiendo en el teletipo el carácter " > ", posterior a ello, el programador tendrá control sobre el computador.

La parte restante del programa monitor incluye un número de 25 subrutinas llamadas REENTRANT SELF-RELATIVE SUBRUTINE ROMS ((RS)3). Estos programas ofrecen gran utilidad, ya que sus características de uso e implementación, permiten al usuario llamarlas en cualquier momento, sin necesidad de preocuparse por la ubicación de las mismas, ya que son relativas a un número asociado a la instrucción de SWI.

Estas subrutinas utilizan direccionamiento relativo para controlar al programa y, al registro índice y al STACK POINTER para la manipulación de datos. Esto permite a las subrutinas ser reentrantes, es decir, el programa puede ser llamado nuevamente sin haber completado la ejecución de la llamada anterior. Para invocar la subrutina deseada, el usuario utiliza un BYTE índice precedido por la instrucción de SWI, donde el índice indica la función que será ejecutada. Después que la subrutina ha sido ejecutada, el programa del usuario continuará con la instrucción que sigue del BYTE índice. Esto puede ser considerado como una extensión del conjunto de instrucciones que ofrece el S6800.

Las 25 subrutinas que comprenden el (SR) 3 grosso modo son como sigue: La explicación comienza con el BYTE índice con el que será llamado, nombre de la subrutina, descripción y el número de BYTEs en el STACK usados en el llamado (inclu-

yendo el SWI).

INDICE	NOMBRE	DESCRIPCION	STACK
00	PUSH ALL	Almacena los registros.	14

Se guardan cinco BYTEs en el STACK, conteniendo, respectivamente, CCR, ACCB, ACCA y el registro índice (XH, XL).

Registro alterado: SP.

01	POP ALL	Recobra los registros.	9

Se obtiene el contenido de los últimos 5 registros del STACK y son colocados, respectivamente, en CCR, ACCB, ACCA y el registro índice.

Registros alterados: CCR, ACCB, ACCA, X y SP.

02	TXAB	Transfiere a X al ACCA y ACCB	9

Se copian los 8 BITs más significativos del registro índice en ACCA, y los 8 menos significativos en ACCB.

Registros alterados: A y B.

03	TABX	Transfiere ACCA y ACCB a X.	9

El contenido del acumulador A se copia en el BYTE más significativo del registro X y, el del acumulador B en el BYTE menos significativo del registro X.

Registro alterado: X

04	XABX	Cambia A y B con X	12

Se intercambia el contenido del registro índice y los dos acumuladores.

Registros alterados: A, B, X.

INDICE	NOMBRE	DESCRIPCION	STACK
05	PUSH X	Almacena al registro índice	11

Se almacena el contenido del registro índice en el STACK.

Registro alterado: SP

06	PULL X	Recobra al registro índice	9

Se obtienen dos BYTEs del STACK y son colocados en el registro índice.

Registros alterados: X, SP.

07	ADDXAB	Suma X a ACCA y ACCB	14

Suma el contenido de X a los dos acumuladores, como una suma de 16 BITs, llevando la suma a los dos acumuladores.

Registros alterados: ACCA, ACCB, CCR.

08	ADDABX	Suma A y B al registro índice	9

Suma el contenido de los dos acumuladores al registro índice, la suma está en el registro índice.

Registros alterados: X, CC.

09	ADDAX	Suma ACCA a X.	9

Suma el contenido de ACCA al contenido de X, la suma en X.

Registros alterados: X, CCR.

0A	ADDBX	Suma ACCB a X.	9

Suma el contenido de ACCB al contenido de X, la suma en X.

Registros alterados: X, CC.

INDICE	NOMBRE	DESCRIPCION	STACK
0B	SUBXAB	Substrae X de A y B	14

Substrae el contenido de X de los acumuladores A y B como una resta de 16 BITs. La diferencia en A y B.

Registros alterados: A, B, CC.

0C	SUBABX	Substrae A y B de X.	9

Substrae el contenido de A y B del contenido del registro índice. La diferencia en X.

Registros alterados: X, CC.

0D	SUBAX	Substrae A de X.	9

Substrae el contenido de A del contenido de X y la diferencia la pone en X.

Registros alterados: X, CC.

0E	SUBBX	Substrae B de X	9

Substrae el contenido de B del contenido de X y la diferencia la pone en X.

Registros alterados: X, CC.

0F	P2HEX	Imprime un BYTE en hexadecimal.	15

El BYTE apuntado por la dirección en X se convierte a notación hexadecimal en ASCII, y se transmite al teletipo vía - ACIA.

Registro alterado: X

INDICE	NOMBRE	DESCRIPCION	STACK
10	P4HEX	Imprime dos --- BYTEs en hexadecimal.	15

Dos BYTEs en memoria apuntados por la dirección contenida en X se convierten a cuatro caracteres en código ASCII e - impresos en el teletipo.

Registro alterado: X

11	PRINTA	Imprime el BYTE- en ACCA.	10

Se imprime el contenido del acumulador A en el teletipo- vía ACIA.

Registro alterado: ninguno

12	PMSG	Imprime un mensaje	12

Una cadena de caracteres, el primer BYTE de la cadena deberá estar apuntado por la dirección en X, se imprime vía - - ACIA. El final de la cadena deberá contener un ASCII EXT - - (=04) hex).

Registro alterado: X

13	VALAM	Alfanumérico válido.	11

El carácter apuntado por el registro índice es analizado y, la bandera de CARRY es fijada si es un dígito o letra; sino es un dígito hexadecimal la bandera OVERFLOW es nivel uno.

Registro alterado: CCR

INDICE	NOMBRE	DESCRIPCION	STACK
14	INPUTA	Carga en ACCA - un BYTE del - - ACIA.	9

Se carga un BYTE desde el ACIA.

Registro alterado: A.

15	CONHB	Convierte una - cadena en hex a binario.	11

Se revisa una cadena de caracteres en memoria, comenzando en la dirección contenida en X (en busca de dígitos hexadecimales válidos), cuando se halla alguno, es colocado en el - BYTE menos significativo de ACCB y continua inspeccionando -- hasta localizar otro, éste será almacenado en el BYTE más significativo de ACCB, haciendo la misma operación en ACCA.

Registros alterados: ACCA, ACCB, X, CCR.

16	INDEX	Multiplica ACCA- x ACCB y suma el resultado al registro fndice.	12

Se multiplica el contenido del acumulador A por el contenido en B, el producto se suma al registro fndice. La suma - en X.

Registros alterados: CCR, X.

17	MUL 8	Multiplica A tantas veces B.	12

Multiplica el contenido de A tantas veces el contenido - en B, el producto será un número de 16 BITs, con la parte más significativa en ACCA.

Registros alterados: ACCA, ACCB, CCR.

BREAK POINTS

Los BREAK POINTS permiten al usuario detener la ejecu- -

ción del programa y examinar el contenido de los registros internos. PROTO provee dos tipos de BREAK POINTS. Su modo de llamada es similar al de (SR) 3, es decir, una instrucción de SWI y un BYTE asociados.

1.- Imprime registros: SWI, 80)h. Imprime el contenido de CC, ACCB, ACCA, registro índice (X), contador de programa (PC) y STACK POINTER (SP), en este orden y, regresa el control a PROTO.

2.- Impresión instantánea (SNAPSHOT); SWI, 81)h. Imprime el contenido de los registros internos, como en SWI 80)h, pero en SNAPSHOT el programa del usuario continua con la ejecución de la siguiente instrucción donde fue invocada SWI, 81

Además de este SOFTWARE, el sistema fue equipado con dos programas adicionales: el programa ASSEMBLER/DISASSEMBLER y el programa de intérprete BASIC.

CAPITULO III

COMUNICACION CON EL SISTEMA AMI 6800

Los dispositivos de entrada/salida (I/O), suministran los medios de comunicación entre el computador y el mundo exterior. Para resolver el problema de procesamiento de datos, éstos y las instrucciones serán introducidos en la máquina y, la computadora dará el resultado de los cálculos. La cuestión es que siempre ha sido complicado compaginar la lentitud de los dispositivos de I/O con la gran velocidad de los sistemas de computación. Otro de los factores importantes en estos elementos, es el alto costo; siendo necesario hacer uso del ingenio para proveerse de las terminales esenciales cuando no se cuenta con los medios económicos suficientes.

La secuencia de I/O a un sistema computador grande o pequeño es muy semejante: primero, se escribe un programa que incluye la descripción de los cálculos y la secuencia del proceso que realizará la máquina. El programa y los datos se organizan para que sean cargados en el sistema. Posteriormente, la máquina leerá el programa, los datos y las órdenes de control necesarios, completándose así, la secuencia de entrada. Después de haber completado la acción de la serie de instrucciones dadas, el sistema imprimirá los resultados (si se le dan las instrucciones adecuadas).

Los dispositivos de entrada suelen ser teclados, lectoras de cinta magnética o de papel perforado, teletipos, etc. Los dispositivos de salida comprenden: perforadores de cinta de papel, grabadoras de cinta magnética, pantalla de VIDEO, teletipos, etc. La utilización de uno o varios elementos depende de la aplicación que se le dará al computador y, lógicamente, de las terminales disponibles.

Generalmente, los sistemas de computación no se conectan directamente a los dispositivos I/O, ya que sus características de acoplamiento no son compatibles; es necesario, entonces, construír los acopladores o interfases adecuadas.

La familia S6800 cuenta con acopladores para este trabajo; éstos son: el S6850 (ACIA), para comunicación en serie, y

el S6820 o S6821 (PIA) para el acceso paralelo. Sin embargo, las características eléctricas de las terminales y los dispositivos antes mencionados pueden seguir, en algunos casos, -- siendo incompatibles, de aquí que, se tenga que agregar otra interfase; pudiendo ser ésta el último eslabón, aunque suele suceder que se necesite de otro más, si es que se quiere poder conectar diversos tipos de dispositivos de I/O. Ver figura III.1.

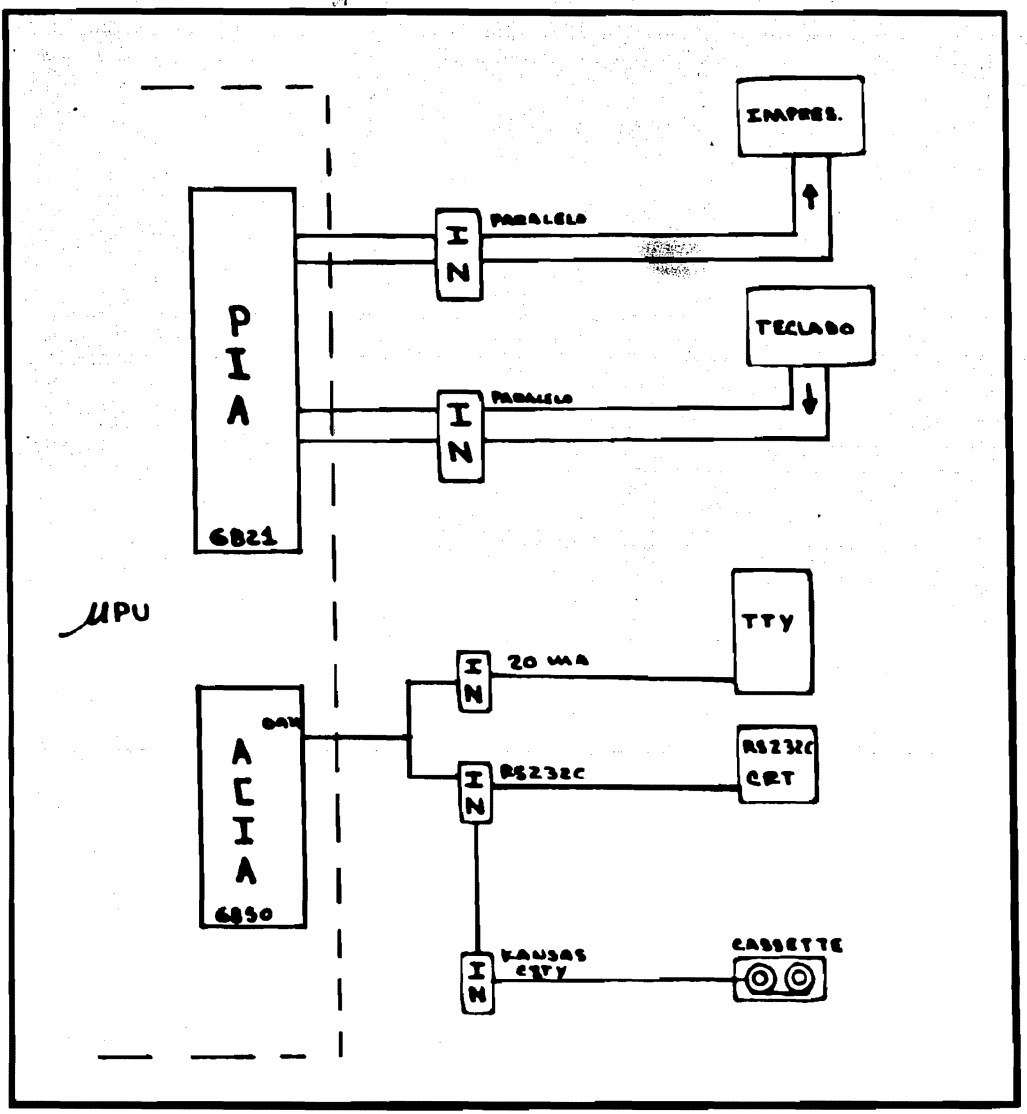


FIGURA III.1.- POSIBLE CONEXIONAMIENTO DE UN SISTEMA CON SUS TERMINALES ASOCIADAS.

Estas interfases pueden consistir: desde un transistor o un elemento TTL, hasta un complejo circuito digital; A/D o -- D/A. AMI 6800 cuenta con dos tipos de interfases: una para - conectarse con un teletipo (20 mamp) y otra para una terminal con RS232C, ambas vía ACIA. (Ver capítulo IIA).

En adición, se cuenta con dos terminales para la comunicación: una de salida -impresora MPR-40- y otra de entrada - -teclado TEXAS INSTRUMENTS-.

CAPITULO III A

TERMINAL DE SALIDA

Uno de los medios más prácticos para poder visualizar los programas y la información procesada, es una terminal de impresión. Esta permite tener siempre a la mano, los datos necesarios para usos, como: la corrección o inserción de artículos, comentarios o subrutinas adicionales. Aunque es uno de los dispositivos con mayor lentitud de operación, también, es de los más útiles; de tal modo que, se optó por incluirlo dentro del sistema.

Impresora MPR-40: Entre sus principales características se pueden mencionar las siguientes:

1.- Su longitud de impresión en línea es de 40 caracteres.

2.- Puede escribir 75 líneas por minuto.

3.- Recibe la información en paralelo, codificada en ASCII.

Modo de funcionamiento: La impresora cuenta con siete líneas de acceso (D0 a D6) y dos líneas de control (DATA READY y DATA ACCEPTED).

En la línea DATA READY, el transmisor indica a la impresora que un carácter se halla presente en las terminales de datos; la terminal captará los datos cuando en DATA READY se de un nivel cero.

Cuando el carácter ha sido capturado, éste se almacena en una de las cuarenta localidades de memoria con las que cuenta la impresora. El proceso de impresión se inicia cuando el archivo mencionado se ha completado, es decir, cuando la máquina ha recibido los cuarenta caracteres que componen una línea. Otra manera de que la terminal imprima la información contenida en memoria, es que ésta reciba un CARRIAGE RETURN (CR).

La línea de control restante, DATA ACCEPTED, indica al transmisor, que la terminal se halla en proceso de impresión o de almacenamiento de datos; durante este lapso, la máquina queda inhabilitada para aceptar nueva información. Esta condición es señalada por la impresora con un pulso nivel cero en la línea DATA ACCEPTED. La figura III.A.1 muestra el diagrama de bloques de la terminal en cuestión.

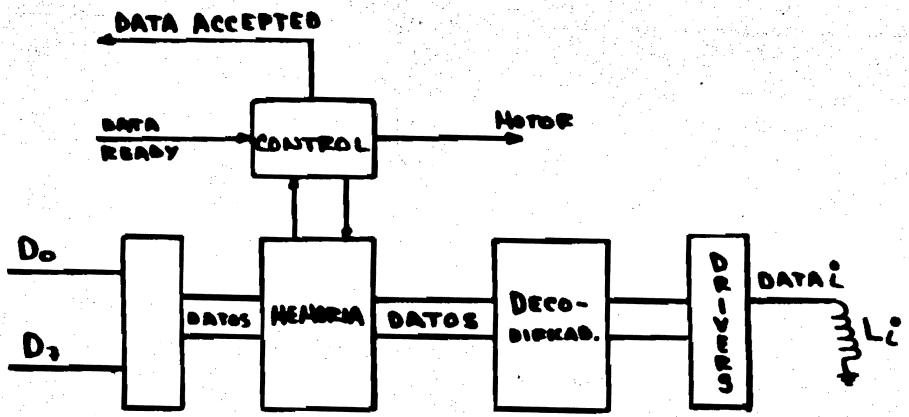


FIGURA III A.1.- DIAGRAMA DE BLOQUES DE LA TERMINAL MPR-40.

La impresión se efectúa sobre papel ordinario de 10 cm.- de ancho, mediante siete agujas que golpean sobre una cinta - entintada; las agujas, son movidas por solenoides accionados- por transistores de potencia.

El acoplamiento de esta terminal con la computadora se- puede realizar de dos maneras:

1.- Hacer un acoplamiento directo mediante una PIA.

2.- Diseñar y construir una interfase con la cual nos pu diésemos comunicar en serie por medio de un ACIA.

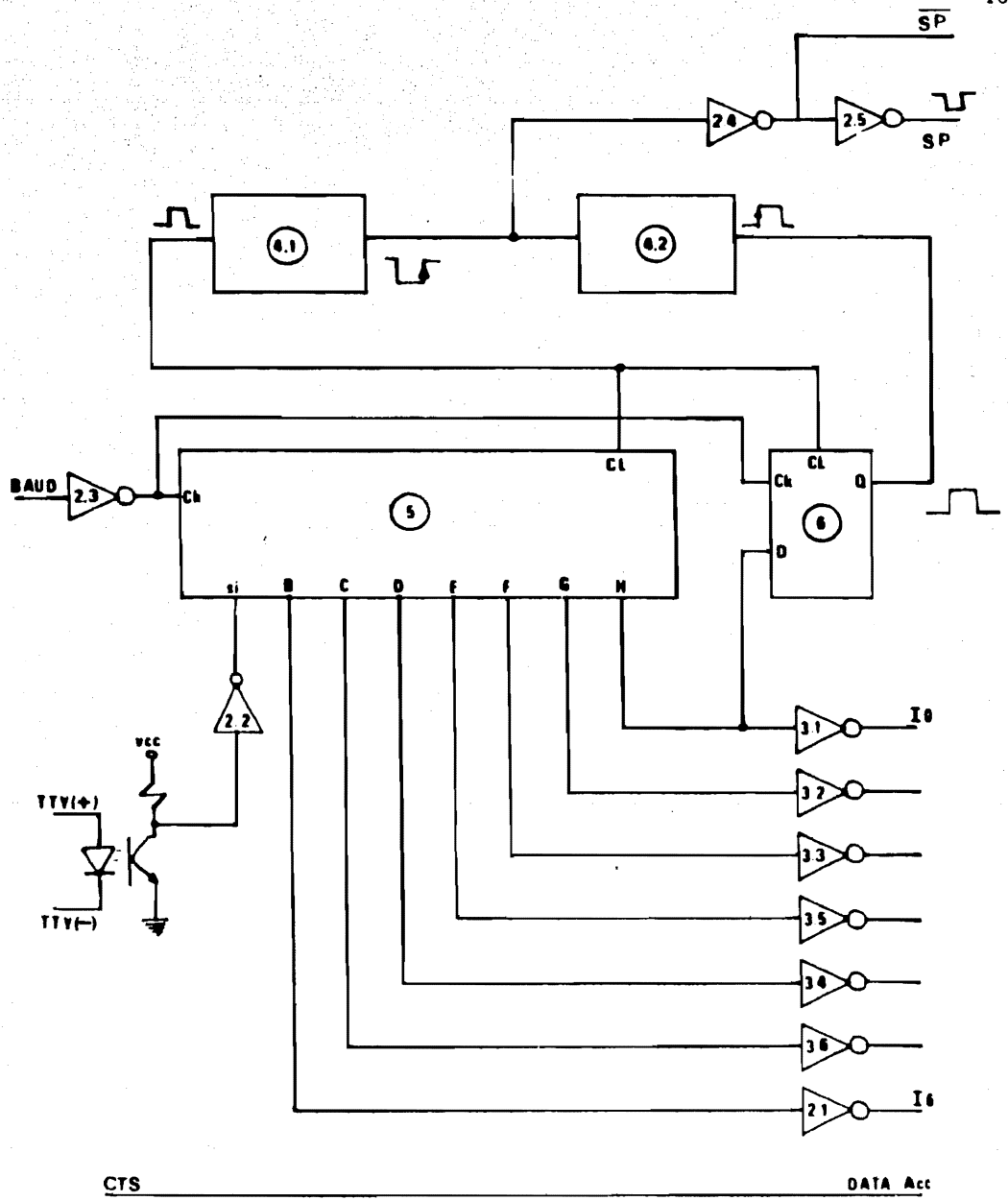


FIGURA III A.2.- DIAGRAMA DE LA INTERFASE SERIE-PARALELO.

La primera alternativa presenta ventajas desde el punto de vista de HARDWARE, ya que no hay necesidad de implementar alguna interfase adicional.

Sin embargo, la segunda alternativa presenta el atractivo de utilizar a la impresora en cualquier computadora con acoplador de 20 ma. (TTY). De aquí que, la interfase se diseñó de tal manera que queda abierta la posibilidad de usar la impresora en ambos modos de operación.

Una terminal con interfase 2o mamp, significa que los niveles lógicos están identificados por la ausencia de corriente o la presencia de ésta, generalmente, una corriente nominal de 20 mamp. En este tipo de acoplamiento, la computadora es la que proporciona la corriente y la terminal, la que cierra la malla por la que circula dicha corriente.

Dado que los elementos empleados en esta interfase utilizan el voltaje como nivel lógico, es necesario: primero, hacer la conversión corriente-voltaje; este modo lo proporciona el acoplador óptimo ICI. Ver figura III A.2.

Como la impresora admite la información en paralelo y el transmisor lo envía en serie, es preciso el empleo de un registro de corrimiento que recibe los datos en serie y los entrega en paralelo (IC5); la velocidad del corrimiento se hace por medio del reloj CK BAUD; éste es el resultado de una división entre dieciséis del reloj X16 BAUD proporcionado por AMI o por un generador propio de la terminal.

A partir de las características de norma con que es enviado un carácter en código ASCII, ver figura III B.3, se diseñó el circuito formado por los elementos IC4-1, 2 e IC6- -figura III. A.2.- IC6 da oportunidad a que los datos (Io, - I_6) estén presentes a la salida de los inversores, e IC4-2, - detecta el frente de onda positivo del pulso de START (en este caso, nivel uno, ya que el carácter codificado es negado - por IC2-2). IC4-2 (monoestable) genera, a partir de que es - disparado, un pulso de aproximadamente 60 micro seg., éste es utilizado para proporcionar a la impresora el pulso de STROBO (SP) indicando que el dato está listo, es decir, esta línea - (SP) se conecta a la de DATA READY de la terminal. Cuando se ha capturado el carácter, es necesario "limpiar" el registro de corrimiento para impedir que algún BIT de datos con nivel uno vuelva a disparar los monoestables. La secuencia ante- - rior es indicada en la figura III A.3.

La línea DATA ACCEPTED de la impresora se conecta directamente al CTS del ACIA.

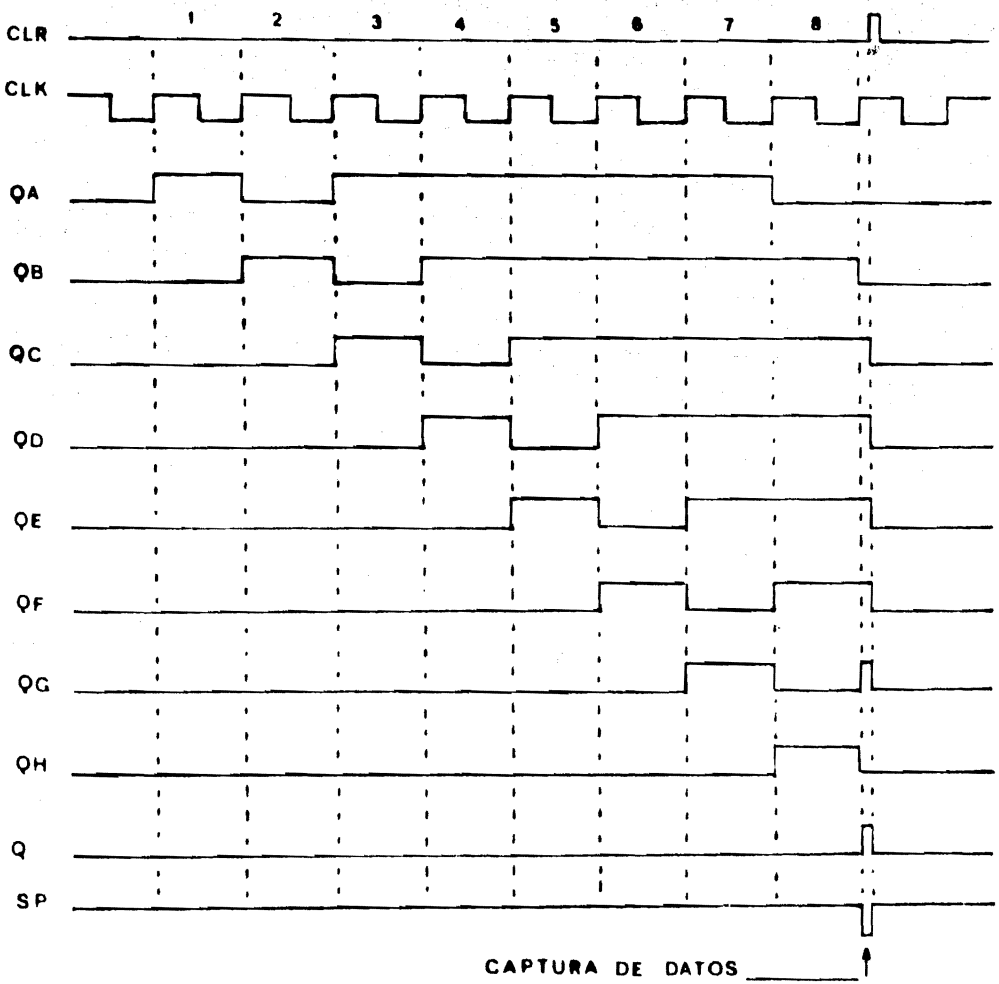


FIGURA III A.3.- SECUENCIA DE LA CONVERSION S/P.

CAPITULO III B

TERMINAL DE ENTRADA

Esta terminal es un teclado tipo máquina de escribir que proporciona la información en paralelo; sus características - principales son:

1.- Salidas compatibles con TTL.

2.- Los caracteres los proporciona codificados en ASCII

3.- Genera BIT de paridad.

4.- Pulso de estrobo positivo (SP).

5.- Cuenta, a la salida, con registros de almacenamiento temporal.

El acoplador para esta terminal se pensó con la misma filosofía que el anterior, ya que se quería lograr una terminal universal, o sea, tipo TTY. La figura III B.1 muestra el diagrama de bloques del teclado.

El circuito de la interfase es sencillo: consiste en un- registro de corrimiento en serie y obtención de dato en paralelo (CI), ver figura III B.2. IC3-1 agrega un BIT de corri- - miento más, necesario para el método de conversión. El hecho de que el teclado tenga a la salida registros de almacenamiento temporal, nos permite efectuar la conversión muy eficientemente, de tal modo que, al presentarse el pulso de estrobo -- (SP), negado por IC4-1, la información está concurrente durante todo el período del pulso, dando así, una gran seguridad a la transmisión, ya que cualquier variación en los datos, durante SP, repercutiría en el registro de corrimiento que adquieren los datos de la siguiente manera: cuando SP es nivel- uno, el registro captura la información, y cuando SP es nivel cero, los datos son corridos; de tal manera que, QH contendrá un BIT con nivel cero al inicio del corrimiento.

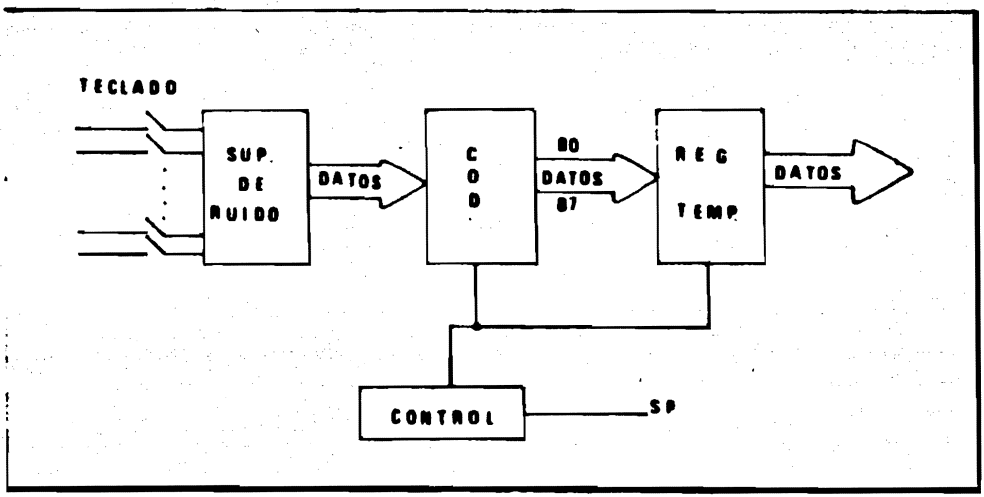


FIGURA III B.1.- DIAGRAMA DE BLOQUES DEL TECLADO TEXAS INSTRUMENTS KB-3.

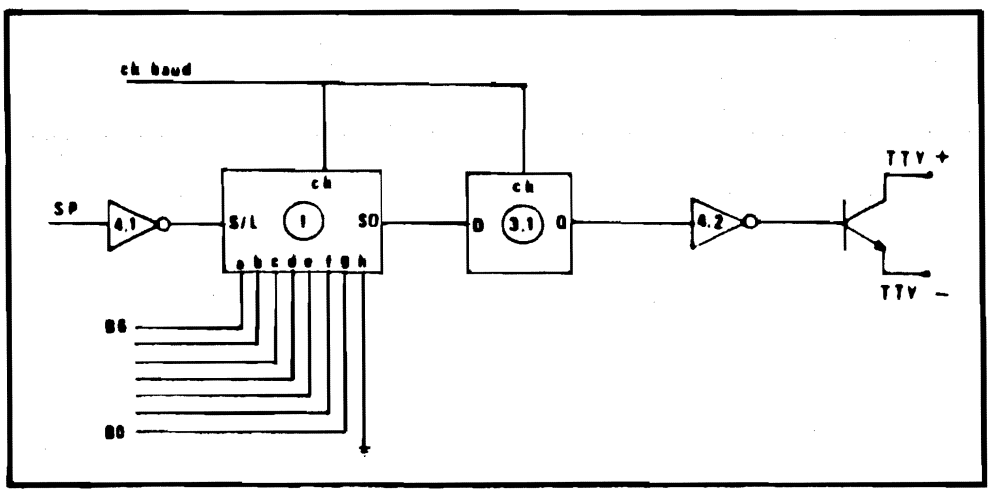


FIGURA III B.2.- CIRCUITO DE LA INTERFASE PARALELO-SERIE.

Los pulsos de STOP son generados por la entrada serie de ICI (SI), dando al acoplador características de normas.

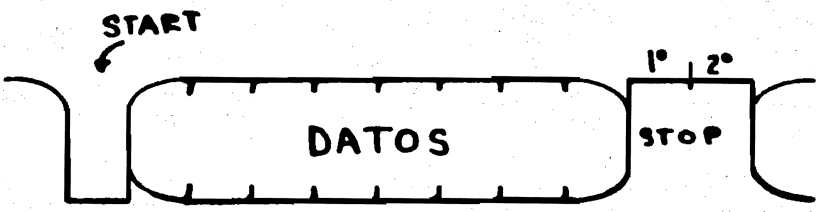


FIGURA III B3.- GENERACION DE UN CARACTER EN CODIGO ASCII.

Estas características (fig. III B.3) son como siguen:

1.- El tren de pulsos debe ser iniciado con un pulso de START, nivel cero; ésto se genera con sólo llevar a tierra el BIT H de ICI.

2.- Los datos están codificados en siete BITS; ellos se transmiten al correr del BIT A al BIT G.

3.- Para generar los pulsos de STOP, la línea que pertenece a la entrada serie (SI) del registro de corrimiento lo llevamos a nivel uno, así es como al final de la transmisión del carácter obtenemos los BITS de STOP necesarios.

IC4-2 nos da el carácter a un nivel TTL y el transistor nos proporciona al acoplamiento a 20 mamp para el computador.

Finalmente, el sistema queda conformado como muestra la figura III B.4. Es conveniente notar que el agrupamiento nos da un dispositivo bastante completo y fácilmente expandible.

Ya en la práctica, la microcomputadora AMI se está manejando también con un teletipo, una pantalla con teclado y una interfase para grabadora de CASSETTES, terminales con las cuales funciona bastante bien.

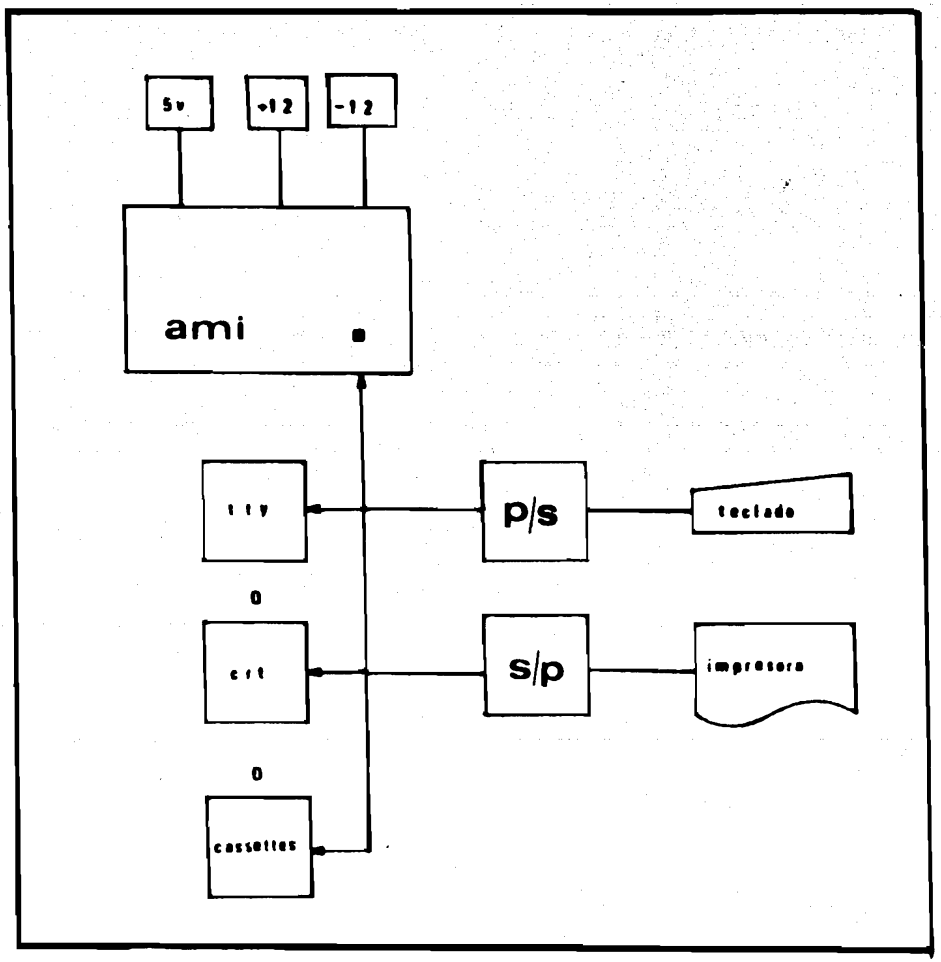


FIGURA III B.4.- AGRUPAMIENTO DEL SISTEMA AMI, FUENTES Y TERMINALES.

CAPITULO IV

INTRODUCCION AL LENGUAJE

El lenguaje de la computadora es básicamente numérico, y es dependiente del diseño y requerimientos de la máquina. - - Esas condiciones hacen que el proceso de escritura de programas en lenguaje de computadoras (llamado lenguaje de máquina) sea una tarea complicada y tediosa. Algunos de los problemas que se encuentran son los siguientes:

1.- El programador trabaja casi exclusivamente con números.

2.- Las direcciones de almacenamiento al estar referidas a una localidad anterior o posterior, hacen que el programador tenga que llevar una relación de estas direcciones por separado, ya que en el transcurso del programa no hay certeza - donde se hallen ubicadas.

3.- El programador tiene que llevar una cuenta o archivo de las localizaciones de instrucciones, de constantes, de datos y de almacenamiento usado y disponible, etc.

A raíz de las dificultades presentadas, hubo la necesidad de idear formas de transcripción de programas de origen o fuente, en una forma que puedan leer las máquinas y que disminuya los errores de la puesta en código; obviamente la computadora misma es el mecanismo ideal para este propósito. Se requiere de un PROGRAMA que realice esta tarea.

Como entrada, el PROGRAMA usa la puesta en código registrada por el programador presentada en un lenguaje mnemónico, comúnmente llamado lenguaje simbólico. Adjunto al PROGRAMA, - se cargan constantes y variables, que se usarán para la traducción y para el registro de la asignación de direcciones en el programa OBJETO (el programa puesto en código en el lenguaje de máquina). El PROGRAMA puede producir el programa objeto en la forma adecuada para que puedan leerlo las máquinas, - así como imprimirlo; también puede proporcionarnos el mapa de almacenamiento y un listado de errores cometidos al transcribir al código original.

El usuario que emplea un PROGRAMA tendrá que precisar la disposición de los registros de entrada y de salida, deberá - especificar qué clase de caracteres son válidos, y en que orden deberán leerse o manejarse. Esta distinción da como re-- sultado lo que se llama el lenguaje del sistema procesador.

El primer nivel de lenguaje se llama lenguaje símbolico, de ensamble, o de una a una. Como lo denota el último término, la particularidad de este lenguaje es que hay una línea - de código para cada instrucción de máquina, como aparece fi-- nalmente en el programa objeto.

Como el lenguaje utilizado tendrá que traducirse al de-- la máquina, el formato presentará cierta semejanza. Las tres partes de una instrucción son:

1.- Operador

2.- Operando o parte de dirección y

3.- La dirección de la localidad de la instrucción misma.

o de la siguiente manera:

ETIQUETA	OPERADOR	OPERANDO
Dirección de la instrucción.	El símbolo que describe la acción.	La descripción de- las cosas que hay que operar.
LABEL	OPERATOR	OPERAND

El empleo de un lenguaje simbólico ofrece varias venta-- jas al programador. Una de ellas es el uso de operadores mnemónicos en inglés. La figura IV.1. muestra la forma en que - se especificaría esto, para el procesador S6800, el operador- mnemónico sugiere la función que se ejecutará. El programa - simbólico ofrece la ventaja de facilitar las referencias a -- otras instrucciones, usando nombres emblemáticos en lugar de- números (etiquetas).

NOMBRE	OPERADOR MNEMONICO	CODIGO DE MAQUINA
COMPARE ACCUMULATORS	CBA	11 (HEX)
ADD ACCUMULATORS	ABA	1B
DEC STACK POINTER	DES	34
INC. INDEX REG.	INX	08, etc.

FIGURA IV.1. CODIGO MNEMONICOS DE OPERACION (S6800)

Después de proporcionar operadores mnemónicos y etiquetas simbólicas, sólo es necesario hacer lo mismo con las constantes y los datos; lo que se lleva a cabo con un mecanismo llamado comúnmente operación declaratoria, que consiste de una secuencia de puesta en clave integrada por una etiqueta simbólica, un operador mnemónico declaratorio y un operando.

EJEMPLO:

ETIQUETA	OPERADOR MNEMONICO	OPERANDO
CALCIO	LDX	$0100

Ya que ésta declaración es construida para que pase al proceso de traducción a lenguaje de máquina (lenguaje objeto) recibe el nombre de declaración de origen o fuente (SOURCE STATEMENT), y al agrupamiento ligado de estas declaraciones se le denomina programa fuente.

CAPITULO IV A

PROGRAMA FUENTE

Una declaración fuente contiene desde uno hasta cuatro - campos que son como sigue:

ETIQUETA; OPERADOR (MNEMONICO); OPERANDO; COMENTARIOS

Las etiquetas se emplean en la declaración que involucra el destino de un salto o ramificación y también en la definición de símbolos.

El operando debe ser incluido en las declaraciones que - lo requieran.

El comentario es opcional y puede ser insertado en las - declaraciones para facilitar la interpretación del programa.

Los campos son definidos por medio de uno o varios caracteres separadores (espacio) ASCII SPACE (SP) según el caso.

El lenguaje fuente para el S6800 (MOTOROLA M6800) está - constituído por cerca de 72 instrucciones mnemónicas y 12 seudo instrucciones (que varían en número y características de - acuerdo a la máquina con que se trabaje). Los caracteres válidos usados en el lenguaje fuente para el ensamblador S6800- son como sigue:

(contenidos en el código ASCII).

1.- El alfabeto de A a Z

2.- Los enteros de 0 a 9

3.- Cuatro operadores aritméticos: + - * /

4.- Caracteres especiales usados como prefijos:

indica el modo de direccionamiento inmediato.

$ indica un número hexadecimal.

@ indica un número octal.

% indica un número binario.

' indica un carácter ASCII

& indica un número decimal

- Instrucciones ejecutables.

El campo de las instrucciones reconocidas por el lenguaje fuente consiste de tres caracteres alfabéticos. Sin embargo, se cuenta con instrucciones que pueden estar referidas a alguno de los dos acumuladores; para distinguir a cuál de ellos se está relacionando se agrega un carácter más, A o B, al campo del operador.

EJEMPLO

OPERADOR	OPERANDO	COMENTARIO
LDAA	LAB	CARGAR EN ACCA EL CONTENIDO DE LAB
LDAB	LIB	CARGAR EN ACCB EL CONTENIDO DE LIB
NEGA		NEGAR ACCA
NEGB		NEGAR ACCB
NEG	CASA	NEGAR EL CONTENIDO DE CASA

Donde LAB, LIB y CASA son identificadores que denominan una ubicación en memoria.

El repertorio de instrucciones se enumera en el apéndice, pero las empleadas en los ejemplos se explican brevemente en la columna de comentarios en la declaración fuente.

- Modos de direccionamiento.

Los modos de direccionamiento fueron comentados en el Capítulo IA, la explicación ahora se dará mediante ejemplos.

El campo correspondiente a la instrucción mnemónica (operador), al ser traducido al lenguaje de máquina, da lugar a un BYTE en código de máquina; el correspondiente al operando, si es que existe, puede generar uno o dos BYTEs según el modo de direccionamiento empleado.

- INHERENTE

En este modo el operador proporciona la información necesaria para que la instrucción sea ejecutada. Por lo tanto en la declaración fuente solo aparecerá el operador y la etiqueta, si es que corresponde.

EJEMPLO:

ETIQUETA	OPERADOR	OPERANDO	COMENTARIOS
	ABA		Suma ambos acumuladores, el resultado en ACCA
	CBA		Compara acumuladores (ACCA--ACCB) sin afectar su contenido, solo fija el CCR.
	CLRA		Limpiar ACCA
	CLRB		Limpiar ACCB
	INCA		Incrementa en uno ACCA

En la instrucción CLR cabe mencionar que se puede emplear dos tipos más de direccionamiento, extendido e indiciado, o sea referido a una localidad de memoria o relativa al registro índice. Al mencionado en este modo de direccionamiento (inherente) cuando se trata de algún acumulador, se le suele llamar también direccionamiento acumulador.

- Relativo

Este modo se emplea en las instrucciones de ramificación (BRANCH) cuyo destino se halla a +127 0 - 128 localidades de su origen, por lo tanto basta un BYTE para indicar la extensión de la transferencia. Es decir, un BYTE para el operador y uno más para el operando donde se indica la dirección relativa de la ramificación.

EJEMPLO:

ETIQUETA	OPERADOR	OPERANDO	COMENTARIOS
	CLRA		ACCA = 0
INA	INCA		ACCA = ACCA + 1
	BPL	INA	RAMIFICA SI ES POSITIVO
	BRA	OUT	RAMIFICA SIEMPRE

El pequeño programa anterior puede servir para generar un retardo en la ejecución de algún programa de mayor tamaño. Las instrucciones BPL y BRA emplean el modo de direccionamiento relativo, ya que la ramificación es relativa a la localización de la instrucción. En BPL se prueba el estado del BIT-N del registro de condición (CCR), si éste es nivel cero se hará la transferencia, es decir, estará ramificando a INA hasta que el valor contenido en el ACCA sea 1 000 0000) binario - (80)hex). En BRA el procesador hará la transferencia a OUT de inmediato.

- Directo

Cuando la localidad a la que queremos tener acceso se ubica en la página cero (rango 00)hex a FF)hex), se puede emplear el modo de direccionamiento directo.

EJEMPLO:

ETIQUETA	OPERADOR	OPERANDO	COMENTARIOS
	LDAA	CASA	CARGAR EN ACCA EL CONTENIDO DE CASA.
	LDAB	HOGAR	CARGAR EN ACCB EL CONTENIDO DE HOGAR.
	ABA		
	STAA	SUMA	ALMACENAR ACCA EN SUMA
	BRA	02	

Donde CASA, HOGAR y SUMA son identificadores de localidades ubicadas dentro del rango antes mencionado, de aquí, que el BYTE del operador más el del operando serán suficientes para definir la instrucción.

- Extendido

Cuando la localidad a la que queremos tener acceso se halla dentro de todo el rango de selección (0000 a FFFF)hex), tenemos que utilizar el modo de direccionamiento extendido. De aquí que, para construir la instrucción se necesiten tres BYTEs, uno para el operador y dos para el operando.

EJEMPLO:

ETIQUETA	OPERADOR	OPERANDO	COMENTARIO
	LDX	HONG	CARGA EN XH EL CONTENIDO DE-HONG Y EN XL EL CONTENIDO DE HONG+1
	INX		
	CPX	KONG	COMPARA HX CON EL CONTENIDO-DE KONG Y XL CON EL CONTENI-DO DE KONG+1
	BMI	CHINA	
	BRA	JAPON	
CHINA	LDX	0120	CARGA EN XH EL CONTENIDO DE-LA DIRECCION 0120 Y EN XL EL DE 0121

Donde HONG y KONG son identificadores de localidades - - dentro del rango de selección antes mencionado.

- INDICIADO

En este modo de direccionamiento, la dirección numérica-es variable y depende del contenido del registro índice. La-forma de indicar este tipo de direccionamiento es como sigue:

Número, X

Símbolo, X

Expresión, X

Donde el número, símbolo o expresión comprende el rango-de 00 a FF)hex.

EJEMPLO:

ETIQUETA	OPERADOR	OPERANDO	COMENTARIOS
1	LDX	$20, X	CARGA A XH CON EL CONTENIDO DE X+20)hex y a XL CON EL CONTENIDO DE X+21)hex.
2	STAA	$00,X	ALMACENA A ACCA EN LA LOCALIDAD CONTENIDA EN LA DIRECCION DE X
3	STAB	LIBRA,X	ALMACENA A ACCB EN LA LOCALIDAD CONTENIDA EN LA DIRECCION DE X + LIBRA
4	INX		
5	LDA A$03,X		CARGA EN ACCA EL CONTENIDO DE LA DIRECCION EN X+3

Del ejemplo: Supongamos que el contenido de X sea 0100)hex al inicio del programa.

Los contenidos de la dirección 0120)hex y 0121)hex sean 03)hex y FF)hex respectivamente. Después de ejecutada la instrucción 1 el contenido del registro índice (X) será 03FF)hex, por lo tanto, después de la instrucción 2, el acumulador A será el contenido de la dirección 03FF)hex y, después de la instrucción 3, el contenido en ACCB será el de 03FF + LIBRA. Después de ejecutada la instrucción 4 el contenido en X será de 0400)hex, de aquí que, después de la ejecución de la instrucción 5, el contenido en A sea el de la dirección 0403)hex.

Este tipo de direccionamiento emplea dos BYTEs para definir la instrucción, una para el operador y otro para el operando, ya que éste último sólo puede tener el valor del rango entre 00)hex y FF)hex.

- INMEDIATO

En este direccionamiento el operando indica el valor que sea operado, es decir, si queremos cargar en algún acumulador un valor determinado, pero que no se halla en alguna localidad de memoria, lo podemos expresar en el operando como una constante o una variable, de tal modo que el formato del operando sea como sigue:

Número

Símbolo

Expresión

#'Carácter

El número, símbolo o expresión puede tener un contenido entre 00)hex a FF)hex expresado con un BYTE, de 0000)hex a FFFF)hex, expresado con dos BYTEs. El carácter puede ser cualquiera contenido en el código ASCII, expresado en hexadecimal, dentro del rango 20 (SPACE) a 5F(-)

EJEMPLO:

ETIQUETA	OPERADOR	OPERANDO	COMENTARIO
1	LDAA	#$0F	CARGA A ACCA CON $0F
1	LDAB	#$30	CARGA A ACCB CON $30
3	ABA		
4	STAA	SUM	
5	LDX	#$0500	CARGA AX CON $0500
6	STAA	0,X	
7	LDAA	#'S	CARGA ACCA CON $53
8	STAA	1, X	

Del ejemplo: El contenido del acumulador A, después de ejecutar 1, será de 0F)hex y el de B, después de ejecutar 2, será de 30)hex, por lo tanto, después de ABA, el contenido en ACCA será de 3F)hex. En la instrucción 5 el contenido de X será de 0500)hex y en ACCA, después de la instrucción 7, estará 53)hex que es el valor en hexadecimal correspondiente al carácter ASCII **S**.

-Seudo instrucciones del ensamblador.

Las directivas o seudoinstrucciones permiten al programador el control del ensamble de las instrucciones ejecutables en código de máquina, incluyendo el control de las localidades de memoria y asignación de valores a datos. Las seudoinstrucciones proveen también el modo de formato para la salida del ensamblador. Las 12 seudoinstrucciones son como sigue:

CODIGO:

ORG Asigna el valor de inicio del contador de programa (PC). Define la dirección numérica del primer BYTE de un segmento subsecuente del programa codificado.

EJEMPLO:

DIRECCION	DATO	ETIQUETA	OPERADOR	OPERANDO
0064			ORG	& 100
AF28			ORG	$ AF28

EQU Iguala un símbolo a un operando. Hace correspondiente un símbolo a un valor numérico, a otro símbolo o a una expresión.

EJEMPLO:

DATO	ETIQUETA	OPERADOR	OPERANDO
0AB7	BAR	EQU	$AB7
001E	TAB	EQU	& 30
0AB7	TRI	EQU	BAR
001F	TRU	EQU	TAB+$1

FCB Formar un BYTE constante

FCC Formar caracteres constantes (ASCII— hexadecimal),

FDB Formar dos BYTEs constantes y

FMB Reserva BYTEs en memoria. Asignan valores y direcciones de datos.

EJEMPLO:

DIRECCION	DATO	ETIQUETA	OPERADOR	OPERANDO
0000	F7	TAP	FCB	$F7
0001	43	MSG1	FCC	/CASA/
0002	41			
0003	52			
0004	41			
0005	F017	MAR	FDB	$F017
0007	000F	LIST1	RMB	& 15
0016	0014	LIST2	RMB	& 20
002A	10	MARI	FCB	$10

END Define el fin del programa fuente.

MON Regresa a la consola. Permiten controlar la secuencia de un programa fuente a través del ensamblador.

NAM Nombre del programa.

OPT Opciones para el control del ensamblador y

SPC Espacio vertical en el listado del programa. Permiten el control del formato. En el caso de OPT existen varias opciones para la impresión de la salida, por ejemplo:

SYMBOL ó S, ocasiona que se imprima la tabla de símbolos.

DB16, el desplegado se efectúa en hexadecimal.

DB8, el desplegado se efectúa en octal.

MEMORY ó M, da instrucciones al ensamblador para almacenar el programa objeto en un archivo permanente, etc.

EJEMPLO IV.A.1. Estructuración de un programa fuente.

Se cuenta con un archivo cuyo contenido puede ser un inventario de productos de alguna compañía o empresa. Un programa se encarga de mostrar al usuario el contenido de dicho archivo, cada ubicación podrá encerrar información acerca del producto que se trate, como su precio, existencias, etc. El usuario tecleará en la consola el nombre del producto y el programa se encargará de mostrar alguna de las características antes mencionadas. El nombre del producto será una combinación de dos caracteres, el primer carácter del arreglo podrá ser cualquier letra de A-Z (ASCII) y el segundo cualquiera de las cinco vocales. De tal modo que la longitud del archivo será de 130 localidades (82)hex).

Se requiere de una subrutina cuya entrada sea la combinación de los caracteres y su salida sea una dirección dentro del rango del archivo; ésta dirección podrá ser entonces utilizada por el programa principal para obtener la información requerida.

La ubicación de los caracteres será proporcionada por el programa principal y corresponde a dos localidades que son llamadas PRIM (almacén del primer carácter) y SEGD) almacén del segundo carácter); el archivo recibe el nombre de ARCH y la localidad que contendrá la dirección generada por la combi

nación de caracteres se llamará DIRE, a la subrutina se le -- dio el nombre de FINC

```
 1          NAM         FINC        DEFINE EL NOMBRE DEL PRO-
                                    GRAMA
 2          OPT         S           IMPRIME LA TABLA DE SIMBO
                                    LOS
 3   *GENERACION DE UNA DIRECCION A
 4   *PARTIR DE DOS CARACTERES.
 5   *PRIM ALMACEN DEL PRIMER CARACTER (A,Z)
 6   *SEGD ALMACEN DEL SEGUNDO CARACTER (A,E,I,O,U)
 7   *DIRE DIRECCION GENERADA.
 8   PRIM   EQU      $018D       DEFINE UBICACION DEL PRIM
 9   SEGD   EQU      $018E       IDEM PARA SEGD
10   DIRE   EQU      $018F       IDEM PARA DIRE
11   FINC   EQU      $0190       DIRECCION DE INICIO DE- -
                                 FINC
12   *
13   ARCH   RMB      $0082       RESERVA EL ARCHIVO
14   *
15   *
16          ORG      $0190       ORIGEN DE LA SUBRUTINA
17   FINC   LDAA     PRIM        DIRECCIONAMIENTO EXTENDI-
                                 DO
18          SUBA     #$41        DIRECCIONAMIENTO INMEDIA-
                                 TO.
19          LDAB     SEGD        DIRECCIONAMIENTO EXTENDI-
                                 DO.
20          CMPB     #$41        DIRECCIONAMIENTO INMEDIA-
                                 TO.
21          BEQ      FCOM        DIRECCIONAMIENTO RELATIVO.
22          ADDA     #$1A        ACCA=ACCA+$1A, INMEDIATO
23          CMPB     #$45        COMPARA ACCB CON $45, IN-
                                 MEDIATO.
24          BEQ      FCOM        (ACCB)=$45,SI?...FCOM
25          ADDA     #$1A        ACCA=ACCA+$1A,INMEDIATO.
```

```
26          CMPB    #$49        (ACCB)=$49,FIJA (CCR)
27          BEQ     FCOM        DIRECCIONAMIENTO RELATIVO
28          ADDA    #$1A        (ACCA)=(ACCA)+$1A, INM.
29          CMPB    #$4F        (ACCB)-4F
30          BEQ     FCOM        (ACCB)=4F,SI?...FCOM
31          ADDA    #$1A        (ACCA)=(ACCA)+$1A, INM
32  FCOM    STAA    DIRE        DIRECCIONAMIENTO EXTENDI-
                                DO
33          RTS                 DIRECCIONAMIENTO INHEREN-
                                TE
34          END                 FIN DEL PROG.
```

FIG. IV A.1. PROGRAMA FINC LISTADO FUENTE.

UBICACION DE VARIABLES

Variable		Dirección
PRIM	(1o. LETRA)	018D
SEGD	(2o. LETRA)	018E
DIRE	(DIRECCION GENERADA)	018F

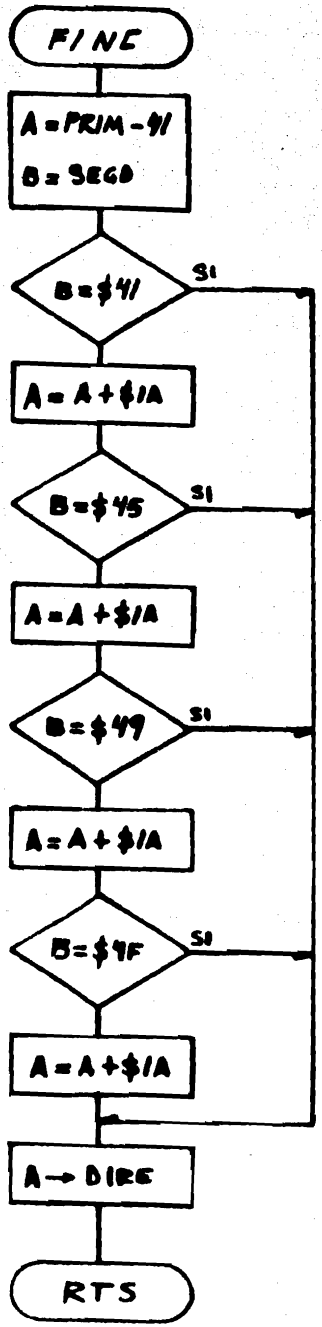


FIG. IV.A.2. DIAGRAMA DE FLUJO DEL PROMANA FINC.

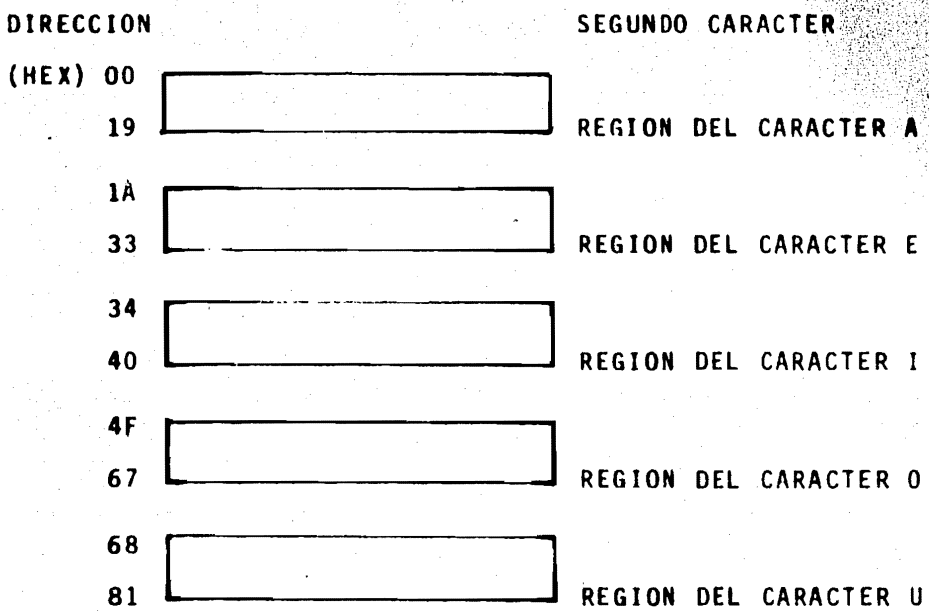


FIGURA IV.A.3. MAPA DEL ARCHIVO ARCH QUE CONTIENE LA INFORMACION RELATIVA A LA DIRECCION DIRE

La solución que se le dio al problema consiste en asignar una región del archivo ARCH a cada una de las vocales, de tal manera que para una combinación, por ejemplo XA le corresponde una dirección dentro de la primera región, y para la combinación XE le corresponde una localidad dentro de la segunda región, y así sucesivamente. Cada sección consta de 26 localidades, que es el número de letras del alfabeto de A-Z (ASCII), de tal modo que para una combinación, por ejemplo, AA la dirección generada será la primera localidad (dada por el primer carácter) de la primera sección (dado por el segundo carácter); para el arreglo BA la dirección generada corresponderá a la segunda localidad de la primera sección, y así sucesivamente.

Resumiendo, el segundo carácter definirá la región en la que se encuentra la localidad. El primer carácter generará una dirección dentro de la primera región de tal modo que solo bastará con sumar la diferencia que halla de la primera a alguna otra región, si es que el segundo carácter es diferente de A, ver Figura IV.A.3.

Los caracteres codificados en ASCII de A a Z, siguen una secuencia que comienza con el valor 41)hex y termina con 5A)-hex, de aquí que restemos el valor 41)hex a el valor del primer carácter de la combinación de entrada, de este modo el carácter A le corresponderá la dirección 00 y a Z la 19)hex. - Ahora bien, como el segundo carácter define la sección se preguntará sucesivamente de cuál se trata, y se irá incrementando de sección en sección (sumando 1A)hex) hasta hallar la dirección buscada. El diagrama de flujo de la figura IV.A.2. - muestra esta secuencia.

El programa fuente se muestra en la figura IVA.1, ejemplifica el uso de las seudoinstrucciones, el de las instrucciones mnemónicas, etiquetas y diferentes modos de direccionamiento. Ahora el paso a seguir es construir el programa objeto.

CAPITULO IV B

PROGRAMA OBJETO

El ensamblado del programa fuente se puede realizar de - diversas maneras, utilizando un ensamblador residente, cons-- truyendo el programa objeto a mano, haciendo el ensamblado -- por medio de una máquina de otro tipo (CROSS ASSEMBLER), etc.

Una vez obtenido el programa objeto, el usuario tendrá - varias opciones para introducirlo al microcomputador AMI, esto dependiendo de que modo de ensamblado se haya efectuado; - si el ensamblado se hace a mano la única alternativa es pre-- sentar el programa por medio del teclado, empleando para ello el programa monitor (comando S). Si se emplea un CROSS - - ASSEMBLER éste puede generar un listado del programa objeto y el programador utilizara la técnica anterior; sin embargo el- CROSS ASSEMBLER puede crear el listado objeto en una cinta de papel perforada o algún otro medio, para ello el usuario usará el monitor y cargará el contenido de la cinta en la microcomputadora AMI (Comando LOAD TAPE-L-).

Cuando se cuenta con un ensamblador residente el programa objeto podrá ser cargado como se mencionó anteriormente o- se podrá hacer automáticamente, etc.

AMI cuenta con un ensamblador/desensamblador contenido - en un S6831 (ROM), aunque éste no cuenta con las caracterísi cas mencionadas en el capítulo IVA es un auxiliar bastante -- útil, sus particularidades se mencionan en el siguiente capítulo.

El Laboratorio de Cibernética, lugar donde se realizó es ta tesis, cuenta con un programa CROSS ASSEMBLER implementado en una minicomputadora; gracias a él se pueden lograr progra mas de gran longitud; éste ensamblador se comentará en el Capítulo V.

Veamos ahora cómo se construye a mano un programa objeto, a partir de los datos y declaraciones proporcionadas por un - programa fuente.

EJEMPLO IV B.1

Ensamblado a mano de un programa fuente.

Este programa efectúa una multiplicación entre dos números. El primer factor se encuentra almacenado en la locali--

dad llamada SD01 y el segundo en SD02. El producto deberá -- ser guardado en la localidad de nombre PROD, por último imprimirá el contenido de los registros del MPU. Posterior a la - ejecución el programa regresará el control de la máquina al - usuario.

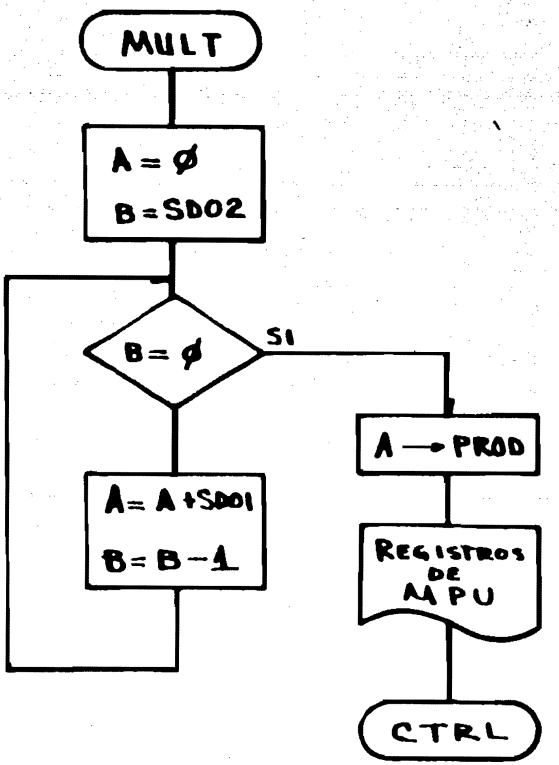


FIGURA IVB.1. DIAGRAMA DE FLUJO DEL PROGRAMA EJEMPLO IV B.1 MULT (PROD=SD01XSD02)

```
    ETIQUETA    OPERADOR    OPERANDO    COMENTARIO
 1              NAM         MULT
 2              OPT         S
 3  *MULTIPLICA DOS NUMEROS SIN SIGNO.
 4  *RESULTADO EN PROD
 5  CTRL        EQU         $F047
 6  SD01        EQU         $0180
 7  SD02        EQU         S0181
 8  PROD        EQU         $0182
 9  *
10  *
11             ORG         $0000
12             CLRA                    LIMPIA ACCA
13             LDAB        SD02        CARGA B CON SD02
14  VERNE      CMPB        #$00        COMPARA B CON CERO
15             BEQ         JULIO       SI B = 0 TRANSFIERE-
                                       A JULIO
16             ADDA        SD01        SUMA A ACCA SD01
17             DECB                    DECREMENTA ACCB
18             BRA         VERNE       RAMIFICA A VERNE
19  JULIO      STAA        PROD        ALMACENA ACCA EN PROD
20             SW1                     IMPRIME CONTENIDO
21             81                      DE REGISTROS.
22             JMP         CTRL        SALTA A MONITOR
23             END                     FIN DEL PROG
```

FIGURA IVB.2. LISTADO FUENTE DEL PROGRAMA MULT (EJEMPLO IVB.1)

El diagrama de flujo de la figura IV.B.1 muestra el algoritmo que sigue el programa del ejemplo, y la figura IVB.2. - indica el listado de las declaraciones fuente.

Para hacer el ensamblado a mano se puede emplear una forma como la mostrada en la figura IV.B.3; la primera columna - sirve para hacer referencia al número de línea que corresponde a cada declaración. Las siguientes tres columnas corresponden a las declaraciones objeto: dirección de la instrucción (DIRE), código de operación (CO) y valor del operando o- dirección efectiva (OPER). Las columnas restantes son propias de las declaraciones fuente: etiqueta (ETIQ), operador - mnemónico (OPMN), operando (OPERD) y la última es reservada - para los comentarios.

El programa fuente es transcrito a la forma como se indica en la figura IV.B.3, posteriormente se adicionan las direcciones correspondientes a cada instrucción, empezando con la- dirección definida por la seudoinstrucción ORG, declaración - 110.

La declaración 120 define la primera instrucción ejecutable del programa, por lo tanto, le corresponde la dirección - de inicio, o sea 0000, a ésta dirección se le sumará el número de BYTEs que ocupa la primera instrucción (CLRA) para obtener la dirección de la segunda (0001); como la segunda ins- - trucción utiliza el direccionamiento extendido (SD02 mayor -- que FF), se emplean 3 BYTES para definirla, por lo tanto a la tercera le corresponde la dirección 0004, y asf sucesivamente.

Después de ésto se podrá definir la dirección correspondiente a las instrucciones referidas por etiquetas, declaraciones 140 y 190; o sea, se completará la tabla de símbolos - que queda como sigue:

TABLA DE SIMBOLOS

CTRL	F047
SD01	0180
SD02	0181
PROD	0182
VERNE	0004
JULIO	000E

-Esta tabla fue solicitada por medio de la declaración - 10 (OPT S) -

El siguiente paso es anotar el código de operación co- - rrespondiente a cada operador mnemónico, ésto se logra consultando el repertorio de instrucciones del S6800 (MC6800, MOTO-

prog: PRODUCTO (MULT) | fecha FEB/79 | lugar CIBERNÉTICA

usuario HÉCTOR EN AMI | pág. 1 de 1 | EJEMPLO IV. B 1

LIN	OBJETO DIRE	CO	OPER	FUENTE ETIQ	OPMN	OPERD	COMENTARIOS
1					NAM	MULT	* NOMBRE
10					OPT	S	SIMBOLOS
20				* MULTIPLICA			
30				* DOS NUMEROS			
40				* SIN SIGNO			
50			F047	CTRL	EQU	$F047	
60			0180	SD01	EQU	$0180	
70			0181	SD02	EQU	$0181	
80			0182	PROD	EQU	$0182	
90				*			
100				*			
110	0000				ORG	$0000	* ORIGEN
120	0000	4F			CLRA		(ACCA) ← ϕϕ
130	0001	F6	0181		LDAB	SD02	(ACCB) ← SD02
140	0004	C1	00	VERNE	CMPB	#$00	(ACCB) − ϕϕ
150	0006	27	06		BEQ	JULIO	(ACCB) = ϕ, SI → JULIO
160	0008	BB	0180		ADDA	SD01	(ACCA) = * + (SD01)
170	000B	5A			DECB		(ACCB) = (ACCB) − 1
180	000C	20	F6		BRA	VERNE	→ VERNE
190	000E	B7	0182	JULIO	STAA	PROD	(ACCA) → (PROD)
200	0011	3F			SWI		MUESTRA REGISTROS
210	0012	81			81		DEL MPU.
220	0013	7E	F047		JMP	CTRL	REGRESA A CONTROL
230					END		
	CTRL		F047				
	SD01		0180				
	SD02		0181				
	PROD		0182				
	VERNE		0004				
	JULIO		000E				

FIGURA IV.B.3. PROGRAMA MULT, DECLARACIONES FUENTE Y OBJETO.

ROLA) proporcionado en el apéndice, por ejemplo:

MNEMONICO	DIRECCIONAMIENTO	CODIGO
CLRA	(INHERENTE)	4F ,
LDAB	(EXTENDIDO, OPERANDO MAYOR QUE FF)	F6 ,
CMPB	(INMEDIATO)	C1 ,
BEQ	(RELATIVO)	27 , etc.

El valor del operando puede estar definido por la tabla de símbolos, declaraciones 130, 150, 160, 180, 190 y 220, o por el operando en la declaración fuente, línea 140. Sin embargo, para el caso de las instrucciones con direccionamiento relativo, el operando de la declaración objeto se define a partir de la dirección absoluta de la ramificación o transferencia, dada en la tabla de símbolos, y la dirección donde se localiza dicha instrucción, ejemplo:

ADDR	OC	OPND	LABEL	OPER	OPENDO
0004	C1	00	VERNE	CMPB	#$00
000C	20	(?)		BRA	VERNE (=0004)

La relación está dada por

$$D = (PC+2) + R$$

Donde:

PC : Dirección de la instrucción con direccionamiento relativo.

D : Dirección destino de la transferencia.

R : Número binario 8 BITs, en complemento a 2, almacenado en el segundo BYTE de la instrucción (operando).

En este ejemplo:

D = 0004 (ver tabla de símbolos)

PC = 000C

y entonces

R = D - (PC + 2)

y

R = F6 que es el operando de la instrucción definida - en la declaración 180.

El procedimiento para correr el programa en AMI es como- sigue: Los caracteres subrayados indican que son tecleados - por el usuario, y los comentarios son adicionados sólo para - facilitar la comprensión, éstos fueron escritos posteriormente. Ver página siguiente.

CAPITULO V

PROGRAMAS AUXILIARES

La potencialidad de un sistema computador estriba no solo del HARDWARE, que lo constituye, sino que también del soporte en SOFTWARE que se le adicione, ésto último es imprescindible cuando al conjunto se le quiere utilizar con versatilidad y eficiencia, inclusive. Cabe la posibilidad de emplear una máquina de mayor tamaño, como coadyuvante en la realización de un proyecto en el que se requiera de un SOFTWARE extenso.

Los programas adicionales usados con AMI son dos: un ensamblador/desensamblador y un programa CROSS-ASSEMBLER implementado en una minicomputadora HP-2114.

CAPITUO V A

ASSEMBLER/DISASSEMBLER

Cuando se requiere de un programa, y éste es de pequeñas dimensiones, lo más práctico es utilizar el ASSEMBLER/DISASSEMBLER, ya que nos proporciona la posibilidad de construir nuestra rutina sin necesidad de hacer uso de las seudoinstrucciones y demás requisitos; es más, podemos hacer correcciones sobre la marcha y editar los programas si es que es necesario; inclusive podemos desensamblar secciones de programas, ya almacenados en memoria, es decir, a partir de un programa objeto se cuenta con la posibilidad de obtener el programa fuente.

Las limitaciones que tiene el ASSEMBLER/DISASSEMBLER (MA/D), entre otras, son que no se puede hacer uso de etiquetas ni operandos simbólicos; sin embargo su ayuda es suficiente.

Este programa está residente en un ROM S6831 (2K BYTES) que está implementado a partir de la dirección E800)hex; además utiliza cierto número de BYTES en RAM como área de trabajo, ésta es localizada en la parte baja de la página cero (00-78) dec.

Para tener acceso al MA/D se hace uso del comando G del-

monitor de la AMI, la dirección de inicio del programa es - E802)hex. Una vez en MA/D se podrán emplear los comandos propios del programa, que son los siguientes:

@ NEWLOC

NEW LOC es una dirección en hexadecimal que define una - nueva localidad donde se habrá de ensamblar o desensamblar -- una o varias instrucciones.

EJEMPLO:

002A: @0080

0080: @90

0090:

!ADDRESS

El signo de admiración seguido por una dirección originará que MA/D llame a una subrutina cuya localidad de inicio es ADDRESS. Si solo se emplea el signo de admiración, el origen de la subrutina será la dirección de la localidad corriente.

-COUNT

Imprime COUNT BYTES en hexadecimal a partir de la dirección corriente.

X X

X

Se almacenan uno o dos dígitos en la dirección corriente y ésta es incrementada en uno. X es un dígito hexadecimal.

'CHARACTER

Este comando origina que el carácter sea convertido a hexadecimal y almacenado en la dirección corriente.

"STRING

El uso de comillas origina que la cadena de caracteres - que las siguen, sean codificados en hexadecimal y almacenados

a partir de la dirección corriente

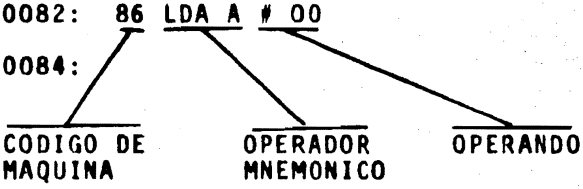
& ADDRESS, COUNT

&, ADDRESS y COUNT (ADDRESS y COUNT de uno a cuatro dígitos hexadecimales) originan que la dirección corriente sea-lle ada a ADDRESS más COUNT.

$ COUNT

El signo de $ seguido de uno o dos dígitos en hexadeci--mal, da como resultado que MA/D desensamble "COUNT" instruc--ciones a partir de la dirección corriente, ejemplo:

```
0080:  $ 03
0080:  01 NOP
0081:  3F SWI
0082:  86 LDA A # 00
0084:
```



(CR)

La señal de control CARRIAGE RETURN es equivalente a $01.

Este programa MA/D se puede implementar en cualquier má-quina que contenga como microprocesador al 6800, ya que se -- pueden configurar los puertos de salida y las subrutinas de I/O, además no hace uso del monitor de AMI. En las direcciones -- 00 a 02, inclusive, se localiza la subrutina de salto al pro-grama de impresión de un carácter.

En las direcciones 03 a 05, inclusive, se localiza la -- subrutina de salto al programa de obtención de un carácter.

Veamos más ejemplos de como se maneja el MA/D.

EJEMPLO VA. 1

Suma de dos números.

El propósito de este ejemplo es ilustrar el uso del MA/D de tal modo que el programa es sencillo, ver figura VA1:

Secuencia del programa:

- Primero se cargan los acumuladores (A y B) con un valor definido por las dos primeras instrucciones.

- Posteriormente se hace la suma del contenido de ambos acumuladores, empleando para ello la instrucción ABA. El resultado de la suma estará contenido en el acumulador A.

- Después se almacena el contenido del acumulador A en la localidad cuya dirección es 0090)hex. La instrucción es STA A 90.

- Por último saltamos a la dirección de inicio de una subrutina (en PROTO) llamada CONTROL (F047)hex, ésta ocasiona que AMI regrese al ámbito del monitor.

La primera sección de la figura VA1 muestra como se introduce el programa a MA/D.

En la sección siguiente, se hace el desensamblado del programa, comando ($05), se solicita la transferencia a la dirección que contiene el resultado, comando (@0090), se pide el contenido de la dirección 90)hex, comando (=01), y por último se pide el listado del programa objeto, comando (=0A).

FIGURA VA.1. EJEMPLO DEL USO DE MA/D

```
                    ** CARGA DE DATOS INICIALES **
** LOS SIGUIENTES CARACTERES LOS DA EL MICROPROCESADOR

                                * CODIGO OPERATIVO DE SALTO DE
                                * SUBRUTINA INCONDICIONAL
                                * Y LO DA POR DEFECTO DEL MA/D

                                *ESTABLECE EL PUNTERO
0000                            *PARA EL CODIGO

(C) 1976, A.M.I.  6800 LISTA ASSEMBLER/DISASSEMBLER - 1.0
0000:CE0000                     * DIRECCION DE INICIO DE DATOS
0003:DF A7                      * GUARDA EN ADDR A7
0005:DE A7 [01] ???             * INDICA QUE HAY UN ERROR
0007:DE A7 [01]                 * CARGA EL DATO EN ACC  B
0009:CBA                        * ACC= ACC+ACCB
000A:37A                        * SWITCH A 9600 BAUDIOS
000B:8E E0A7                    * PUNTERO A CONTROL (MONITOR)
000E:7E0000>                    * EJECUTA EL PROGRAMA SUBRUTINA
                                * EN LA DIRECCION 0000.
                                * EL CARACTER ">" INDICA QUE EL
                                * MONITOR ESTA PRESENTE.

>0000                           * PROTEGE/NO PROTEGE LA MA/D

(C) 1976, A.M.I.  6800 MICRO ASSEMBLER/DISASSEMBLER - 1.0
0000:CE0000                     * LA DIRECCION DEL PROGRAMA
0003:A6                         * PROTEGIDO/PLANOS E INSTRUCCIONES
0005-> 01  LDA A #07
0007-> 06  LDA B #01
0009-> 1B  ABA
000A-> 07  STA A  00
000C-> 7E  JMP    E0A7
000F:CE000F                     * LA DIRECCION DEL RESULTADO
0012:A1                         * EL CONTENIDO DE LA DIRECCION
0014-> 00
                                * EN 0012 SE TIENE EL RESULTADO DE
0015:CE0                        * 07 (EN ACCA) + 01 (EN ACCB)
0018:E7                         *ALMACENA RESULT.
001A-> 3C 3E 01 01 1F 07 00 3F
001C-> FF FF
001E:                           * FIN DEL SUBRUTINA.
```

EJEMPLO V.A2

Precio de un producto en almacén (PRINCIPAL).

En este ejemplo, además de mostrar el uso del MA/D, se ilustra el manejo de las subrutinas del (SR)3, ver capítulo IIB, éstas subrutinas son parte del monitor de AMI y las empleadas en este ejercicio son las siguientes:

SUBRUTINA PMSG

SWI

12 El monitor imprime una cadena de caracteres, empezando con el carácter contenido en la dirección que apunta el registro índice, el final de la cadena debe ser un EOT (04)hex.

Ejemplo: Si el mensaje se halla a partir de la dirección 01B9, se deberá cargar a X con esa dirección, de tal modo que queda como sigue.

Dirección de la cadena.

01B9: "MARCA DEL PRODUCTO?

01CC: 07

01CD: 04

El 07 indica al TTY que toque la campana (BELL) y el 04 indica al monitor que es el final de la cadena de caracteres (EOT).

Dirección de la instrucción

0114: LDX # 01B9 .

0117: SWI . IMPRIME M2

0118: 12 .

Al correr esta sección del programa se imprimirá en el TTY lo siguiente:

MARCA DEL PRODUCTO?

- En el programa que le llamaremos PRINCIPAL, se hallan-

contenidos seis mensajes:

```
M1    1o. mensaje: hace un cambio de renglón (01B4)
M2    2o. mensaje: MARCA DEL PRODUCTO?       (01B9)
M3    3o. mensaje: EL PRECIO DE              (01CE)
M4    4o. mensaje: ES  $                     (01DC)
M5    5o. mensaje: .00 MN                    (01E2)
M6    6o. mensaje: EL PRODUCTO NO ESTA EN EL
                   ALMACEN...                (01E9)
```

De tal modo que para imprimir cualquiera de ellos sólo - se escribirán las 3 declaraciones.

```
LDX  #  MX
SWI
12
```

y el monitor imprimirá la cadena indicada.

SUBRUTINA INPUTA

SWI
14 AMI aguarda a que el usuario teclee en el TTY un carác-- ter; éste será guardado en el acumulador A.

SWI PRINTA
II El monitor imprimirá el BYTE contenido en el acumulador- A.

- Estas subrutinas frecuentemente se emplean acopladas,- ya que es cómodo que el usuario al presionar la tecla de un - carácter, éste se imprima en el TTY. A este modo se le suele llamar el Modo de ECO.

Ejemplo: El usuario presiona en el teletipo el carácter Z, y se desea que Z se almacene en algún lugar de memoria y,- además, el computador de ECO; el programador puede formar és- to del modo siguiente:

DIRECCION DE LA INSTRUCCION

```
0119:  SWI          . CARGA EN ACCA
011A:  14           . UN BYTE DE ACIA
011B:  STAA 018E    . ALMACENA EN 018D
011E:  SWI          . IMPRIME ACCA
011F:  11           . EN TTY (ECO)
```

SUBRUTINA TABX

SWI
03 Se transfiere el contenido del acumulador A al registro-índice en su parte más significativa y el BYTE contenido en el acumulador B se transfiere al registro índice en - su parte menos significativa.

Ejemplo:

Antes contenido del acumulador A - 00
contenido del acumulador B - 7F
contenido del registro X - 0100

después de SWI

03
Contenido del registro X - 007F
ACCA y ACCB no se alteran.

SUBRUTINA P2HEX

SWI
OF El contenido de la dirección encerrada en el registro índice se imprime en el teletipo en FORMATO hexadecimal.

Ejemplo:

Contenido de X - 107F

Contenido de la localidad 107F - B4

En TTY después de ejecutada la subrutina

B4

Un programa semejante a "principal" puede ser usado en - una tienda o almacén; el empleado o cliente teclea el nombre del producto y la computadora puede mostrar el precio, la - existencia en almacén, características, etc. Para ello se -- tendrá que contar con sistemas de almacenamiento auxiliar, como discos o cintas magnéticas, ya que AMI está muy limitado - en capacidad de memoria, "Principal" sólo pretende ser un - - ejemplo del uso de AMI 6800 y sus elementos de SOFTWARE.

En este programa "principal" se utiliza la subrutina del ejemplo IVA.1 (FINC), ver capítulo IVA, ésta subrutina genera una dirección a partir de un arreglo de dos caracteres; en la figura VA.2 se lista el programa FINC empleando el MA/D.

La secuencia del programa "PRINCIPAL" es la siguiente:

Dirección que contienen las instrucciones implicadas en la secuencia - De 010F a 0118)hex.	SECUENCIA
	1. Pregunta el usuario por el juego de caracteres; usando el mensaje 1 y 2. Se supone que los-dos caracteres definen la marca del producto. Mensaje en TTY: MARCA DEL PRO--DUCTO?.
De 0119 a 0126	2. El programa espera a que el - - usuario presione la marca del - artículo, y contesta con el ECO. Para ésto "principal" ya almacenó la marca en las localidades-PRIM y SEGD, dirección 018D y - 018E respectivamente.
De 0127 a 0129	3. Salto a la subrutina que verifica que los dos caracteres sean-válidos. De regreso el BIT C - del CCR del MPU es nivel cero - cuando el juego es válido. De - otro modo salta a la subrutina-ABOR (dirección 015B)
012B	4. Salto a la subrutina FINC. De - regreso la dirección del precio del producto se halla en la lolicadad 018F.
De 0120 a 0145	5. "Principal" imprime el mensaje-uno, cambio de renglón, el mensaje tres, el nombre del producto y el mensaje cuatro.
De 0146 a 014B	6. Se limpia el contenido del acumulador A y En B se carga la dirección obtenida por FINC. El-contenido de ambos acumuladores se transfiere al del registro - índice (X).
De 014C a 014D	7. Se imprime el precio del producto, siendo éste el contenido de

la localidad encerrada en X.

De 014E a 0158

8. Por último "principal" imprime los mensajes 1 y 5, y regresa el control al monitor.

De 015B a 0165

9. Esta subrutina, ABOR, es empleada por "principal" cuando el juego de caracteres no es válido, imprimiendo el mensaje M6.- (El producto no está en el almacén).

De 0167 a 018C

10. Subrutina de verificación (VERI) este subprograma certifica que los caracteres tecleados por el usuario estén dentro del rango admitido por FINC. Si el juego es válido se fijará el valor de C, del CCR del MPU, a nivel cero, de otro modo a nivel uno.

De 0190 a 01B3

11. Subrutina FINC, comentada en el capítulo IVA.

De 01B4 a 020E

12. Archivo que contiene los mensajes (M1 a M6).

La figura VA3A, B, C muestra el listado del programa - - PRINCIPAL, el de las subrutinas ABOR y VERI, el del archivo-- de mensajes y ejemplos de la corrida de "principal".

```
*** PROGRAMA PRINCIPAL ***

>G E802

A.M.I. 6800 MICRO ASSEMBLER/DISASSEMBLER - 1.0
(C) 1976, A.M.I.
802A:0010F                          ** SOLICITUD DE DIRECCION DE PRIN
010F:S22
010F-> CE  LDX    #01B4             *
0112-> 3F  SWI                      *  IMPRIME M1
0113-> 12  12                       *
0114-> CE  LDX    #01B9              *
0117-> 3F  SWI                       * IMPRIME M2
0118-> 12  12                        *
0119-> 3F  SWI                      *  CARGA EN ACCA
011A-> 14  14                       *  EL PRIMER BYTE DE ACIA
011B-> B7  STA A  018D               * ALMACENA EN PRIM
011E-> 3F  SWI                      *  IMPRIME PRIM (ECO)
011F-> 11  CBA                      *  EN TTY
0120-> 3F  SWI                       * CARGA EN ACCA
0121-> 14  14                        * EL SEGUNDO BYTE DE ACIA
0122-> B7  STA A  018E               *  ALMACENA EN SEGD
0125-> 3F  SWI                       * IMPRIME SEGD (ECO)
0126-> 11  CBA                       * EN TTY
0127-> 8D  BSR    0167              *  VERIFICA EL PRODUCTO Y
0129-> 25  BCS    015B              *  RAMIFICA SI ES ILEGAL
012B-> 8D  BSR    0190               * TRANSFIERE A "FINC"
012D-> CE  LDX    #01B4             *
0130-> 3F  SWI                      *  IMPRIME M1
0131-> 12  12                       *
0132-> CE  LDX    #01CE              *
0135-> 3F  SWI                       * IMPRIME M3
0136-> 12  12                        *
0137-> B6  LDA A  018D              *
013A-> 3F  SWI                      *  IMPRIME PRIM
013B-> 11  CBA                      *
013C-> B6  LDA A  018E               *
013F-> 3F  SWI                       * IMPRIME SEGD
0140-> 11  CBA                       *
0141-> CE  LDX    #01DC             *
0144-> 3F  SWI                      *  IMPRIME M4
0145-> 12  12                       *
0146:S28                             * LISTA LA SIG. PAGINA.
```

FIGURA VA.3.A. LISTADO DE PRINCIPAL

```
                *** PROGRAMA PRINCIPAL (PAGINA 2) ***


0146-> 4F   CLR A                                  *  A=0
0147-> F6   LDA B   018F                           *  B=DIRE
014A-> 3F   SWI                                    *  XH=00
014B-> 03   03                                     *  XL=DIRE
014C-> 3F   SWI                                    *  IMPRIME EL
014D-> 0F   SEI                                    *  CONTENIDO DE DIRE
014E-> CE   LDX    01E2                            *
0151-> 3F   SWI                                    *  IMPRIME N5
0152-> 12   12                                     *
0153-> CE   LDX    01B4        -FIN-               *
0156-> 3F   SWI                                    *  IMPRIME M1
0157-> 12   12                                     *
0158-> 7E   JMP    F847                            *  SALTA A CONTROL
015B-> CE   LDX    01B4        -ABOR-              *
015E-> 3F   SWI                                    *  IMPRIME M1
015F-> 12   12                                     *
0160-> CE   LDX    01E9                            *
0163-> 3F   SWI                                    *  IMPRIME M6
0164-> 12   12                                     *
0165-> 20   BRA     0153                           *  SALTA A CONTROL
0167-> B6   LDA A   018D        -VERI-             *  A=PRIM
016A-> 81   CMP A 40                               *  PRIM<=40?
016C-> 23   BLS     0169                           *  SI > NOVA
016E-> 81   CMP A 5B                               *  PRIM=>5B?
0170-> 24   BCC     0169                           *  SI > NOVA
0172-> B6   LDA A   018E                           *  A=SEGD
0175-> 81   CMP A 41                               *
0177-> 27   BEQ     016B                           *
0179-> 81   CMP A 45                               *
017B-> 27   BEQ     016B                           *  SEGD=A,E,I,O,U ??
017D-> 81   CMP A 49                               *
017F-> 27   BEQ     016B                           *  SI > SIVA
0181-> 81   CMP A 4F                               *
0183-> 27   BEQ     016B                           *
0185-> 81   CMP A 55                               *
0187-> 27   BEQ     016B                           *
0189-> 0D   SEC                 -NOVA-             *  ARREGLO NO VALIDO C=1
018A-> 39   RTS                                    *
018B-> 0C   CLC                 -SIVA-             *  ARREGLO VALODO C=0
018C-> 39   RTS                                    *
018D:01B4
01B4:=40                        *** CONTENIDO DE ARCHIVO MENSAJES .
```

FIGURA VA.3.B. LISTADO DE PRINCIPAL

```
01B4-> 0D 0A 00 00 04 4D 41 52
01BC-> 43 41 20 44 45 4C 20 50
01C4-> 52 4F 44 55 43 54 4F 3F
01CC-> 07 04 45 4C 20 50 52 45
01D4-> 43 49 4F 20 44 45 20 04
01DC-> 20 45 53 20 24 04 2E 30
01E4-> 30 20 4D 4E 04 45 4C 20
01EC-> 50 52 4F 44 55 43 54 4F
01F4-> 20 4E 4F 20 45 53 54 41
01FC-> 20 45 4E 20 45 4C 20 41
0204-> 4C 4D 41 43 45 4E 2E 2E
020C-> 2E 07 04
020F:
```

```
 ***** EJEMPLO DE LA CORRIDA DE PRINCIPAL + FINC *****

>G 010F

MARCA DEL PRODUCTO?BA
EL PRECIO DE BA ES $34.00 MN
>G

MARCA DEL PRODUCTO?KU
EL PRECIO DE KU ES $23.00 MN
>G

MARCA DEL PRODUCTO?AI
EL PRECIO DE AI ES $65.00 MN
>G

MARCA DEL PRODUCTO?ML
EL PRODUCTO NO ESTA EN EL ALMACEN...
>G

MARCA DEL PRODUCTO?SA
EL PRECIO DE SA ES $05.00 MN
>G

MARCA DEL PRODUCTO?0P
EL PRODUCTO NO ESTA EN EL ALMACEN...
>G

MARCA DEL PRODUCTO?MA
EL PRECIO DE MA ES $40.00 MN
>
```

FIGURA VA.3.C. ARCHIVO DE MENSAJES Y EJEMPLO DE LA CORRIDA DE PRINCIPAL.

CAPITULO V B

CROSS-ASSEMBLER

Como se vió en el capítulo anterior, el MA/D (MICRO- -ASSEMBLER/DISASSEMBLER) la única ventaja que tiene es que se- emplean operadores mnemónicos en lugar de códigos de opera- - ción, y que se usan direcciones absolutas (en las instruccio- nes de ramificación BRA, BSR, etc.) en lugar de direcciones - relativas; de aquí que haya la necesidad de hacer uso de un - SOFTWARE más sofisticado. Obviamente los lenguajes de alto - nivel (FORTRAN, BASIC, ALGOL, etc.) son los que presentan el- mayor número de atributos al respecto; sin embargo AMI, como- está implementada en la actualidad, no tiene la capacidad de- memoria para contener alguno de estos lenguajes, inclusive ni para un ensamblador residente.

Una solución es que se utilice una máquina de mayor di-- mensión, ésta puede ser otra microcomputadora con más capaci- dad de memoria, una minicomputadora o una computadora de - - gran tamaño. En cualquier caso encontramos varias alternati- vas; una de estas es realizar el programa fuente en la compu- tadora disponible, ésta, una vez efectuado el proceso de en-- samblado, nos entregará el programa objeto que se correrá en- la AMI o en cualquier microcomputadora que incluya el S6800.

Naturalmente el empleo de una máquina adicional implica, que ésta última, represente ún costo mucho mayor que si se su mara la memoria necesaria a AMI para implementar en ella un - ensamblador residente; pero cuando se tiene a la mano este -- medio no hay razón para no hacer uso de el.

La máquina disponible es una minicomputadora HEWLETT -- PACKARD 2114A que cuenta con los elementos siguientes:

a) Una lectora de cintas de papel perforado.

b) Una perforadora de cintas de papel.

c) Un sistema de almacenamiento auxiliar basado en dis-- cos flexibles (FLOPPY DISK), con capacidad de dos de- éstos.

d) Un teletipo.

e) Una pantalla de VIDEO con teclado, etc.

El programa en cuestión está almacenado en disco y el -- lenguaje en el que está implementado es el ensamblador propio

de H.P.; aunque se pudo haber realizado en algún otro lenguaje con los que cuenta la H.P. 2114A, éstos son ALGOL, FORTRAN y BASIC.

El sistema operativo de la "mini" nos proporciona la facilidad de poder editar el programa fuente que será ensamblado, para ello se almacena en disco y se trabaja en él hasta que el programa se ha depurado.

El sistema de SOFTWARE cruzado es muy empleado cuando no se quiere hacer una inversión, en una máquina que tal vez no se opere en su capacidad total, en la aplicación que se le destine. El programa al que me refiero es un CROSS-ASSEMBLER diseñado e implementado para el micro-procesador MC6800 (MOTOROLA) o S6800 (AMI), este ensamblador es de dos pasos:

Primer paso: (Una vez editado y obtenida la cinta fuente). Se carga el programa fuente por medio de la foto colectora (lectora de cintas de papel perforado). Al correr el CROSS ASSEMBLER la "mini" escribe en el TTY la tabla de símbolos (SYMBOL TABLE) e indica que se encienda la perforadora de cintas de papel.

Segundo paso: Nuevamente se corre el CORSS ASSEMBLER y también se vuelve a cargar el programa fuente. Ahora la "mini" lista los programas fuente y objeto en el TTY y además nos da la cinta objeto o binaria por la perforadora.

Veamos un ejemplo.

EJEMPLO VB

Este programa muestra como el CROSS-ASSEMBLER nos lista los programas fuente y objeto; de manera similar que si se tratase de un ensamblador residente (ver capítulo IV A y B).

El ejemplo consiste en la conversión de un número hexadecimal a uno decimal (BCD); el rango admitido es de 0000 a FFFF (0 a 65535) dec, es decir, el máximo número de dígitos hexadecimales es de 4. A la entrada de esta subrutina, que llamaremos HEXDE, los dos dígitos más significativos (MSD) deberán hallarse en el acumulador A y los dos menos significativos (LSD) en el acumulador B.

A la salida HEXDE entregará los dígitos en decimal a partir de la dirección contenida en el registro X, este dígito será el más significativo del resultado.

El algoritmo empleado para la conversión consiste en di-

vidir el número en hexadecimal entre 10)dec (0A)hex); el residuo será el dígito en decimal menos significativo del resultado. El cociente será dividido entre 10)dec, y nuevamente el residuo será el siguiente dígito del resultado, y así sucesivamente.

El listado del programa, una vez efectuados los pasos respectivos, se muestra en la figura VB.1; en ella se puede observar el programa objeto y el programa fuente dados por el CROSS-ASSEMBLER. Los comentarios son indicados por un asterisco y describen grosso modo la operación de cada una de las instrucciones. La tabla de símbolos (SYMBOL TABLE) define las etiquetas empleadas en la subrutina.

```
    FE E/FINAF
LISTADO= L  FINAFIO= F  AMFOS= -
F


INFUT FFOG.

SYMECOL TAFLF
HEXDE     0050
MO        005E
M1        0064
M2        006D
M3        0077
M4        007A
CELA      008C
CONT      008D
COC       008E
COC1      008F
DEC4      0090
DECO      0094
INFLT FFOG.

0000 1  0050      005C            0FG   $0050
***    FFOGRAMA FARA CONVERTIF UN NUMERO
***    HEXADECIMAL
***    LONGITUL MAXIMA: CUATRO DIGITOS
***    LOS DOS MSD EN ACCA
***    LOS DOS LSD EN ACCF
***
***    EL FESULTADO: CINCO DIGITOS EN
***    DECIMAL.
***    Y APUNTA LA DIRECCION DEL
***    DIGITO MAS SIGNIFICATIVO.
***
*      INICIO DE LA SUBRUTINA DE CONVERSION.
*
0001E   005C  E7  0000     HEXDE   STAA   CELA
0001F   005E  CE  0004             LDX    #DECC
00020   0061  86    05              LDAA   #$05
00021   0063  E7  0000             STAA   CONT
00022   0065  E6  0000             LDAA   CELA
00023   0068  7F  0000     MO      CLR    CCC
00024   006B  7F  0000             CLR    COC1
00025   006E  CC    0A     M1      SUBD   #$0A
00026   0071  24    05             BCC    M2
00027   0073  4A                   DECA
```

FIGURA VB.1.A. LISTADO DEL PROGRAMA HEXDE (CROSS ASSEMBLER)

```
0025  0069  81     FF              CMPA  #$FF
0026  006B  27     0A              BEQ   M2
0027  006D  7C  0080      M2      INC   COC
0028  0070  26     F2              BNE   M1
0029  0072  7C  0081              INC   COC1
0030  0075  20     EE              BRA   M1
0031  0077  4C              .2      INCA
0032  0078  CB     0A              ADDB  #$0A
0033  007A  E7     00              STAB  00,X
0034  007C  09                      RTS
0035  007D  7A  0082          M4  DEC   CONT
0036  0080  27     01              BEQ   M5
0037  0082  F6  0082              LDAB  COC
0038  0085  86  0081              LDA.  COC1
0039  0088  20     F1              BRA   M0
0040  008A  09              .5      RTS
0041  008B  3F                      RTI
****
****   ALMACENES AUXILIARES.
****
0045  008C          01      CELA    RMB   $01
0046  008D          01      CONT    RMB   $01
0047  008E          01      COC     RMB   $01
0048  008F          01      COC1    RMB   $01
*
*      SALIDA DE RESULTADOS.
*
0051  0090          04      DECA    RMB   $04
0053  0094          01      DECC    RMB   $01
0054                                  END
**NO ERRORS **
```

FIGURA VB.1.B. LISTADO DEL PROGRAMA HEXDE (CROSS ASSEMBLER)

CAPITULO VI. A

EMPLEO DE UNA PIA COMO INTERFASE PARA UNA IMPRESORA (MPR-40)

En el capítulo III se trató sobre los acopladores para la impresora de línea y un teclado, descritos en esa parte, ahora veremos como se puede efectuar el enlace con la impresora por medio de una PIA, siendo este método más sencillo desde el punto de vista de HARDWARE.

En realidad, la solución consiste de un programa que efectúe la iniciación y programación de la PIA empleada para este propósito; el programa incluye varias subrutinas que pueden ser llamadas por el usuario desde una rutina cualquiera. La figura VI.1 indica el acoplamiento AMI (vía PIA)-impresora.

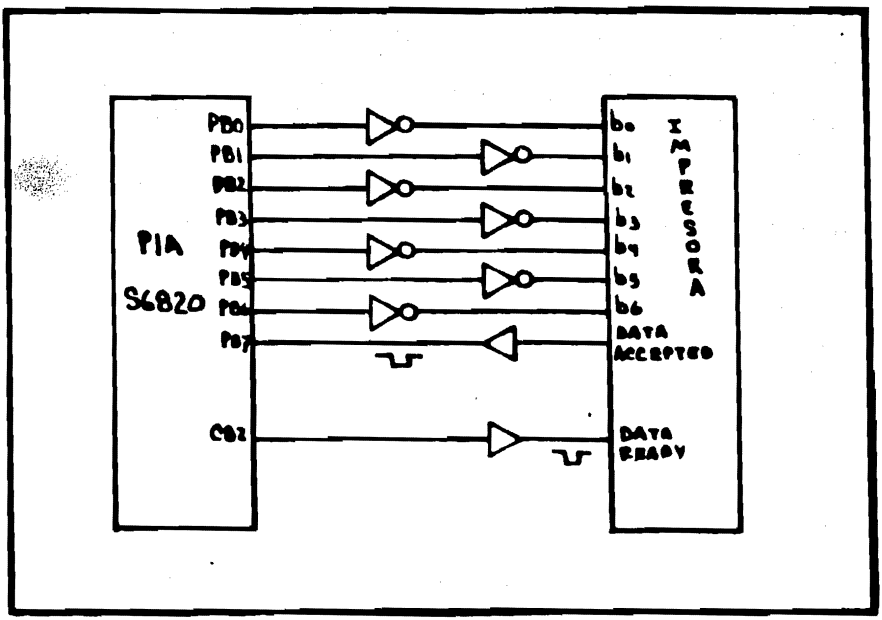


FIGURA VI.1. LINEAS EMPLEADAS PARA EL ACOPLAMIENTO PIA-MPR40.

Recordemos que la PIA es un dispositivo de acoplamiento en paralelo y que puede ser conectada directamente al MPU. - Las líneas de enlace con éste último, consisten de un BUS de datos (DO a D7) y varias líneas de control (Ø2,VMA, etc.). - Las líneas de comunicación con el exterior son 20, 10 de - - ellas correspondientes al canal A y las restantes conforman- el canal B. Las funciones que desempeñan ambas secciones -- son semejantes, y pueden ser programadas por el usuario me-- diante el MPU (ver capítulo IB5). El canal empleado para es te ejemplo es el B y sus líneas se indican en la figura VI.I

Tanto en la sección A como en la B se incluyen tres registros:

-CRB. Registro de control (8 BITS); en él se almacena- el código correspondiente al modo de operación de las líneas CB1 y CB2; nos indica el cambio de estado de éstas y nos da la opción de seleccionar alguno de los dos registros restantes (mediante el BIT 2 del CRB).

-DDRB. Registro de dirección de datos (8 BITS); nos -- permite el escoger el modo de operación de las líneas PB0 a- PB7, es decir, si en el BIT N del DDRB se almacena un nivel- uno, la línea PBN será determinada para usarse como salida;- y si el BIT N es nivel cero PBN será entrada.

-ORB. Registro de salida de datos; en este registro se puede leer o escribir, según el modo escogido, el estado que guardan las líneas PB0 a PB7.

El MPU puede tener acceso al CRB, DDRB u ORB del mismo- modo que si se tratase de una localidad de memoria. El re-- gistro de control (CRB) está localizado en la dirección FBCB y la dirección FBCA corresponde al registro de salida de datos (ORB) y al registro de dirección de datos (DDRB); recuérdese que el BIT 2 del CRB nos proporciona el discriminador - de ambos registros, de tal modo que al referirnos a algunos- de estos dos (DDRB u ORB) lo haremos con un solo identificador que es PRB (registro periférico B).

- Primera Subrutina.- Esta subrutina inicia a la PIA en cuestión para ser acoplada a la impresora.(INIP).

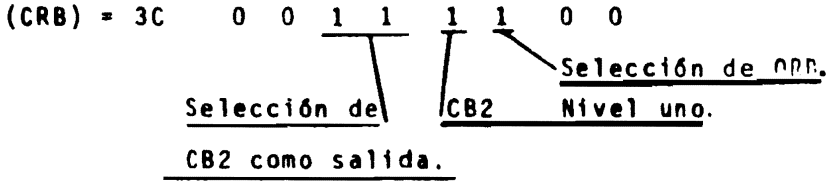
```
INIP    PSHA            -ALMACENA ( ACCA) EN STACK
        CLRA            -(ACCA) = 00
        STAA    CRB     -COMENTARIO 1
        LDAA    #$7F    -(ACCA) = 7F)HEX
        STAA    PRB     -COMENTARIO 2
        LDAA    #$3C    -(ACCA) = 3C) HEX
        STAA    CRB     -COMENTARIO 3
        PULA            -RECUPERA (ACCA) DEL STACK
        RTS
```

- COMENTARIOS:

-1: Ocasiona que se seleccione el DDRB (BIT2 DEL CRB es nivel cero).

-2: Se seleccionan como salidas las líneas PB0 a PB6 y como entrada PB7

-3: Ocasiona que se seleccione el ORB (BIT2 del CRB es nivel uno).



CB2 se utiliza, conectada a DATA READY de la impresora- (ver figura VI.1), como un pulso de estrobo para indicar a - la terminal, cuando CB2 sea nivel cero, que el dato puede -- ser capturado.

Registros alterados: registros en PIA y CCR.

- Segunda Subrutina.- Al invocar esta subrutina se im-- prime el carácter almacenado en el acumulador A. (ESCAR)

```
ESCAR       SWI          -+ALMACENA LO REG
            00           -+DEL MPU EN STACK
            COMA         -COMPLEMENTA (ACCA)
            STAA   PRB   -COMENTARIO 1
            LDAA   #$34  -(ACCA)= 34
            STAA   CRB   -COMENTARIO 2
            NOP          -NO OP
            LDAA   #$3C  -(ACCA)=3C
            STAA   CRB   -COMENTARIO 3
ESP         TST    PRB   -COMENTARIO 4
            BPL    ESP   -COMENTARIO 5
            SWI          -+RECUPERA EL MPU
            01           -+SUS REGISTROS DE
                           STACK
            RTS
```

- COMENTARIOS:

-1: El carácter a imprimir se almacena en el registro - de salida de datos (ORB).

-2: La línea CB2 se lleva a nivel cero, indicando a la- impresora que tome el carácter.

-3: La línea CB2 es llevada nuevamente a nivel uno.

-4: Se prueba el estado del registro de salida de datos (ORB).

-5: Se transfiere a ESP si el BIT7 del registro de sali da de datos (ORB) es nivel cero; indicando que la - impresora está en proceso de almacenamiento o de im presión.

Registros alterados: Ninguno.

INSTRUCTION	MNEM	INH	IMM	DIR	INX	EXT	REL	OPERATION	H	I	N	Z	V	C
Add accumulators	ABA	1B						A + B → A	A		A	A	A	A
Add with carry	ADCA		89	99	A9	B9		A + M + C → A	A		A	A	A	A
	ADCB		C9	D9	E9	F9		B + M + C → B	A		A	A	A	A
Add	ADDA		8B	9B	AB	BB		A + M → A	A		A	A	A	A
	ADDB		CB	DB	EB	FB		B + M → B	A		A	A	A	A
Logical and	ANDA		84	94	A4	B4		A · M → A			A	A	R	
	ANDB		C4	D4	E4	F4		B · M → B			A	A	R	
Shift left arith.	ASL				68	78					A	A	6	A
	ASLA	48									A	A	6	A
	ASLB	58									A	A	6	A
Shift right arith.	ASR				67	77					A	A	6	A
	ASRA	47									A	A	6	A
	ASRB	57									A	A	6	A
Branch if carry clear	BCC						24	if C = 0						
Branch if carry set	BCS						25	if C = 1						
Branch if equal to zero	BEQ						27	if Z = 1						
Branch if greater or equal to zero	BGE						2C	if N ⊕ V = 0						
Branch if greater than zero	BGT						2E	if Z+(N⊕V)=0						
Branch if higher	BHI						22	if C + Z = 0						
Bit test	BITA		85	95	A5	B5		A · M			A	A	R	
	BITB		C5	D5	E5	F5		B · M			A	A	R	
Branch if less or equal to zero	BLE						2F	if Z+(N⊕V)=1						
Branch if lower or same	BLS						23	if C + Z = 1						
Decimal adjust accumulator A	DAA	19									A	A	A	3
Decrement	DEC				6A	7A		M − 1 → M			A	A	4	
	DECA	4A						A − 1 → A			A	A	4	
	DECB	5A						B − 1 → B			A	A	4	
Decrement stack pointer	DES	34						SP − 1 → SP						
Decrement index	DEX	09						X − 1 → X				A		
Exclusive or	EORA		88	98	A8	B8		A ⊕ M → A			A	A	R	
	EORB		C8	D8	E8	F8		B ⊕ M → B			A	A	R	
Increment	INC				6C	7C		M + 1 → M			A	A	5	
	INCA	4C						A + 1 → A			A	A	5	
	INCB	5C						B + 1 → B			A	A	5	
Increment stack pointer	INS	31						SP + 1 → SP						
Increment index	INX	08						X + 1 → X				A		

- Tercera Subrutina.- Imprime una lista de caracteres,- empezando en la dirección en X y termina cuando el carácteres un EOT (04)hex). (PLIST).

```
PLIST   SWI             -+ALMACENA LOS REG.
        00              -+DEL MPU EN EL STACK
PDLIS   LDAA 0,X        -(ACCA)= ((X))
        INX             -(X) = (X)+1
        CMPA #$04       -(ACCA)-04, FIJA CCR
        BEQ  SPL        -(ACCA)=04?, ENTONCES SPL
        JSR  ESCAR      -IMPRIME (ACCA)
        BRA  PDLIS      -CONTINUA PROXIMO CARAC
SPL     PULA            -RECUPERA (ACCA) DEL
        TAP             -STACK y TRANSFIERE ACCR
        PULB            -RECUPERA (ACCB)
        PULA            -RECUPERA (ACCB)
        INS             -INC EL STACK PARA
        INS             -ELIMINA X (ANTERIOR)
        RTS             -
```

Registros alterados:

- Cuarta Subrutina.- Imprime el BYTE apuntado por la - dirección en X en formato hexadecimal. (I2HEX). El BYTE - - apuntado por la dirección contenida en el registro fndice -- (X) se convierte a notación hexadecimal en ASCII y se imprime en la terminal.

```
I2HEX   SWI             -+ALMACENA LOS REG
        0C              -+DEL MPU EN STACK
        LDAA 0,X        -(ACCA) = ((X))
        BSR     NIBL    -COMENTARIO 1
        BSR     ESCAR   -IMPRIME (ACCA)
        LDAA    0,X     -(ACCA) = ((X))
        BSR     NIBR    -COMENTARIO 2
        BSR     ESCAR   -IMPRIME (ACCA)
        SWI             -+RECUPERA EL
        01              -+MPU SUS REG
        INC             -(X) = (X)+1
        RTS             -REGRESA A PROG.
NIBL    LSRA            -CORRE UN BIT DER
        LSRA
        LSRA
        LSRA
NIBR    ANDA    #$0F    -(ACCA)=(ACCA).0F
        ADDA    #$30    -(ACCA)=(ACCA) + 30
        CMPA    #$39    -(ACCA)-39, FIJA CCR
        BLS     RTSB    -(ACCA)MENOR Q 39, RTSB
        ADDA    #$07    -(ACCA)=(ACCA)+07
RTSB    RTS
```

-COMENTARIOS:

-1: La subrutina NIBL convierte a notación hexadecimal- en ASCII los 4 BITS más significativos del (ACCA)

-2: La subrutina NIBR convierte a notación hexadecimal- en ASCII los 4 BITS menos significativos del (ACCA)

Registro alterado: X (se incrementa en 1).

APLICACION DEL EJEMPLO VI.1

Una de las particularidades del MA/D consiste en que las rutinas de I/O se pueden configurar de tal modo, que es posible emplear alguna otra terminal adicional o implementar el MA/D en otra máquina con el S6800 como procesador.

Las subrutinas del ejemplo VI.1 se pueden emplear en este caso, de esta manera se puede trabajar en un periférico X, verbigracia, una terminal CRT de alta velocidad, y solicitar los listados ya corregidos en la impresora MPR40; o también el usuario puede dar la orden de impresión de resultados del programa en cuestión. Para lograr esto, se requiere de un pequeño programa que modifique las subrutinas de I/O del MA/D.

Se recordará que el MA/D en las direcciones 00 a 05 almacena las localidades donde se ubican las subrutinas de I/O de tal modo que se substituirá la que se refiere a la salida, o sea, la subrutina ESCAR.

El programa es como sigue:

```
        NAM     IMAD
IMAD    LDX     #$3F14      *CARGA LAS
        STX     00          *SUBRUTINAS
        LDX     #$397E      *DE I/O
        STX     02          * EN LAS
        LDX     #$FC11      *LOCALIDADES
        STX     04          *PROPIAS
        LDX     #$0414      *4 NULLS
        STX     06          *Y14 CPL
        JSR     INIP        *INICIA PIA
        JMP     MAID        *
```

La figura VI.2 muestra el listado de las subrutinas del ejemplo VI.1, dadas por la impresora MPR40 ya implementadas en ésta INIP y ESCAP.

La tabla de símbolos de las subrutinas son:

Para INIP

INIP FC00
CRB FBCB
PRB FBCA

Para ESCAR

ESCAR FC11
PRB FBCA
CRB FBCB
ESP FC22

Para PLIST

PLIST FC2A
SPL FC38
ESCAR FC11
PDLIS FC2C

Para I2HEX

I2HEX FC3F
NIBL FC55
ESCAR FC11
NIBR FC59
RTSB FC63

Para IMAD

IMAD FC64
INIP FC00
MA/D E800

Para correr el MA/D empleando la impresora de línea basta dar el comando GO, del monitor de la AMI, a la dirección de IMAD o sea

G FC64

entonces la impresora es controlada por el usuario mediante el MA/D.

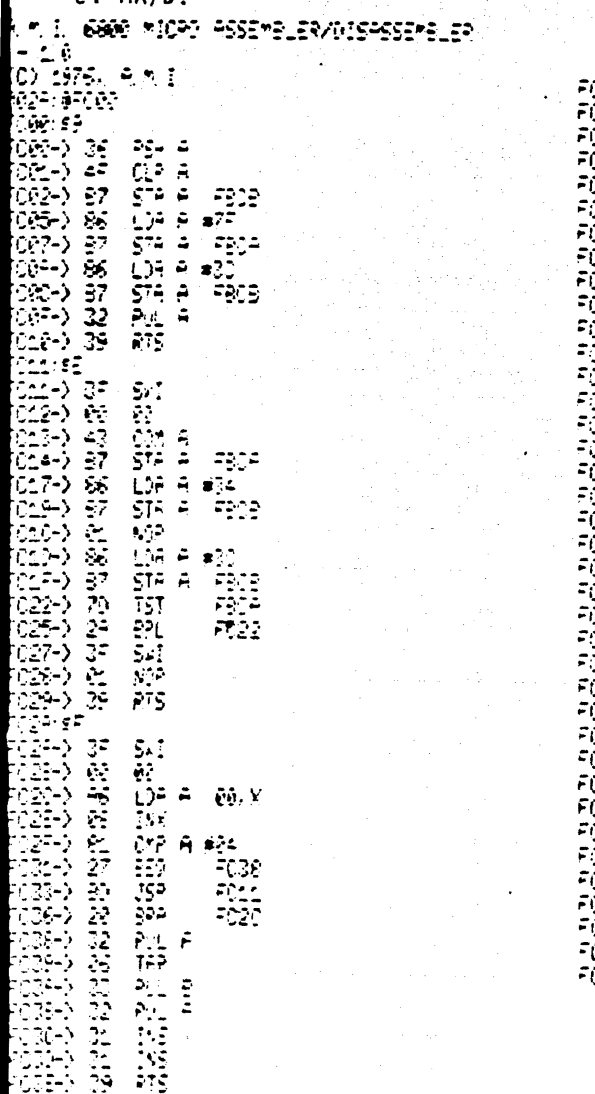


FIGURA VI.2. LISTADO DE LAS SUBRUTINAS DEL EJEMPLO VI.1. DADAS CON LA IMPRESORA MPR-40.

CAPITULO VI-B

EVALUACION Y SIMULACION DE UNA BASCULA DE SUPERMERCADO.

Las aplicaciones comerciales que se le dan a la electrónica, en todas sus ramas, han contribuído a que ésta se desarrolle con mayor rapidez que otras áreas de la ciencia. Los microcomputadores no podían ser la excepción, los podemos encontrar, cada vez con más frecuencia, en un sinnúmero de artefactos de uso común: lavadoras, hornos de microondas, automóviles, aviones, etc.

La aplicación que se indica en este ejemplo, es una de tantas que se le pueden dar a un microprocesador; en particular la tecnología actual permite que el conjunto de dispositivos empleados para esta evaluación, sean substituídos por uno solo, como en realidad debe ser, sin embargo, se intenta hacer notar que primero se necesita una apreciación de los elementos que conformarán el sistema definitivo.

Una báscula, como la mostrada en la Fig. VI-B.1, se encuentra en la mayoría de los supermercados, en las secciones donde se hace el pesado de artículos comestibles.

La capacidad de la báscula es de 9.999 Kg. y el máximo precio por Kg. es de 999.99 pesos/Kg. El transductor peso-voltaje nos proporciona un rango de salida de 0 a 1 volt. y un A/D nos da la conversión con una resolución de 8 BITS (aprox.- 0.4%). El método de conversión empleado es el de aproximaciones sucesivas y se menciona más adelante. Cabe hacer notar que el error de medición es muy grande, comparado con el rango que se intenta alcanzar (0-9.999 Kg), necesitándose una resolución de 14 BITS para lograr un error no significativo (menor de un gramo).

El despliegue de la información es mostrada por un módulo que contiene los DISPLAYS de siete segmentos necesarios. Este tiene la capacidad de recibir los datos codificados en BCD y almacenarlos en secuencia, de tal manera que siempre indica los dígitos en el orden en que fueron recibidos. Este módulo contiene 12 líneas de entrada: cuatro para los datos (D0-D3), dos de habilidad (E0-E1) y seis para dar los comandos de almacenar y limpiar los registros internos (3 x CLR y DP).

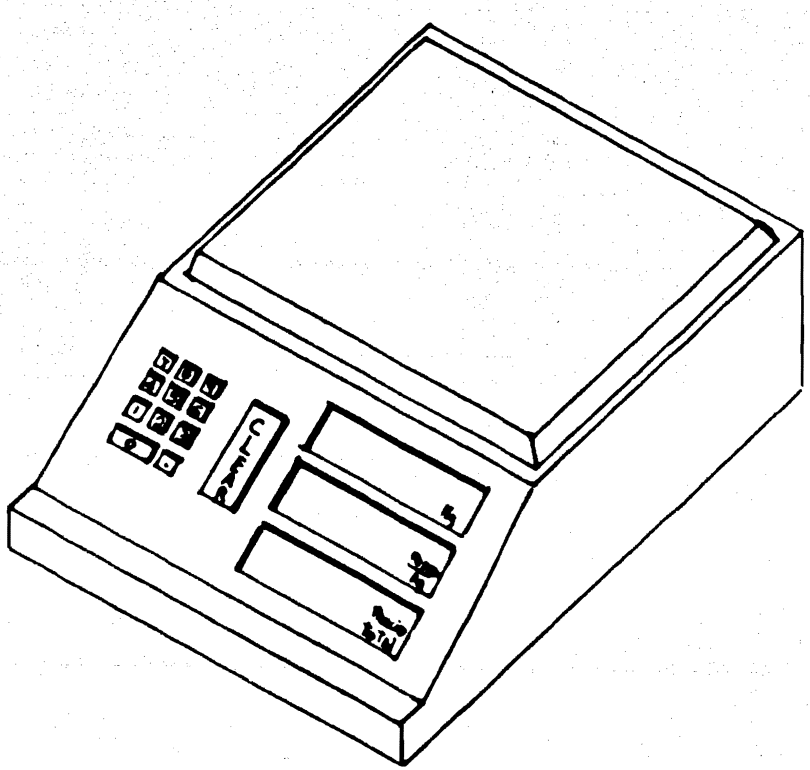


FIGURA VIB.1. PRESENTACION COMERCIAL DE UNA BASCULA DE SUPERMERCADO.

Además se incluye un teclado con diez dígitos, punto decimal y una tecla de CLEAR.

El sistema necesario, usando los elementos que integran AMI, son los mostrados en la figura VI B.2 Cabe hacer notar que el número y capacidad de los dispositivos empleados en este modelo se pueden reducir aún más; ya que las localidades de memoria usada son muy pocas, y que multiplexando la salida de una PIA se puede lograr que realice las funciones de las dos empleadas.

De acuerdo al diagrama de flujo, mostrado en la fig. VI B.4, la secuencia del proceso de pesado es como sigue:

1) Al encender la báscula se da un RESET, de tal modo que se inicializan ambas PIAS y los registros necesarios (SP, FLAG, PUNTO, ETC.). Se carga el vector de interrupción correspondiente (IRQ), para que éste contenga la dirección de la rutina de servicio (KINTR); este programa será invocado cuando el usuario presione cualquier tecla de dígito o el de punto, ver fig. VI B.5.

2) El siguiente paso es capturar un valor de la sección del A/D y mostrarlo en el DISPLAY.

   El valor capturado va de 00 a F9) Hex (249 decimal), de modo que bastará con multiplicar por 28 hex (40 decimal) para obtener el valor correspondiente al peso obtenido (subrutina MULT 8, ver cap. II B). La subrutina que adquiere el valor del A/D es muy sencilla, consiste en dar un valor inicial al registro de salida de la PIA2, se hace la conversión D/A (dispositivo IC1, fig. VI B.3), y se compara con el valor analógico de entrada; si éste último es mayor, se incrementará el contador hasta que se igualen. En este momento el comparador (IC2) hará un cambio de nivel, que se registrará en la línea CA1, indicando que se ha completado la conversión; entonces el valor digital actual, en el registro de salida de la PIA, será el valor buscado (subrutina VOLA1).

   Con el valor resultante de la multiplicación en ambos ACUMULADORES, se entra a la subrutina HEXDE; ésta hace la conversión hexadecimal - BCD (ver cap. V B).

   De aquí se parte hacia la subrutina que muestra el resultado en el DISPLAY correspondiente (DPESO). Después de esto, el programa regresará hasta el punto REGEE, si es que el usuario no ha terminado de teclear el precio del producto.

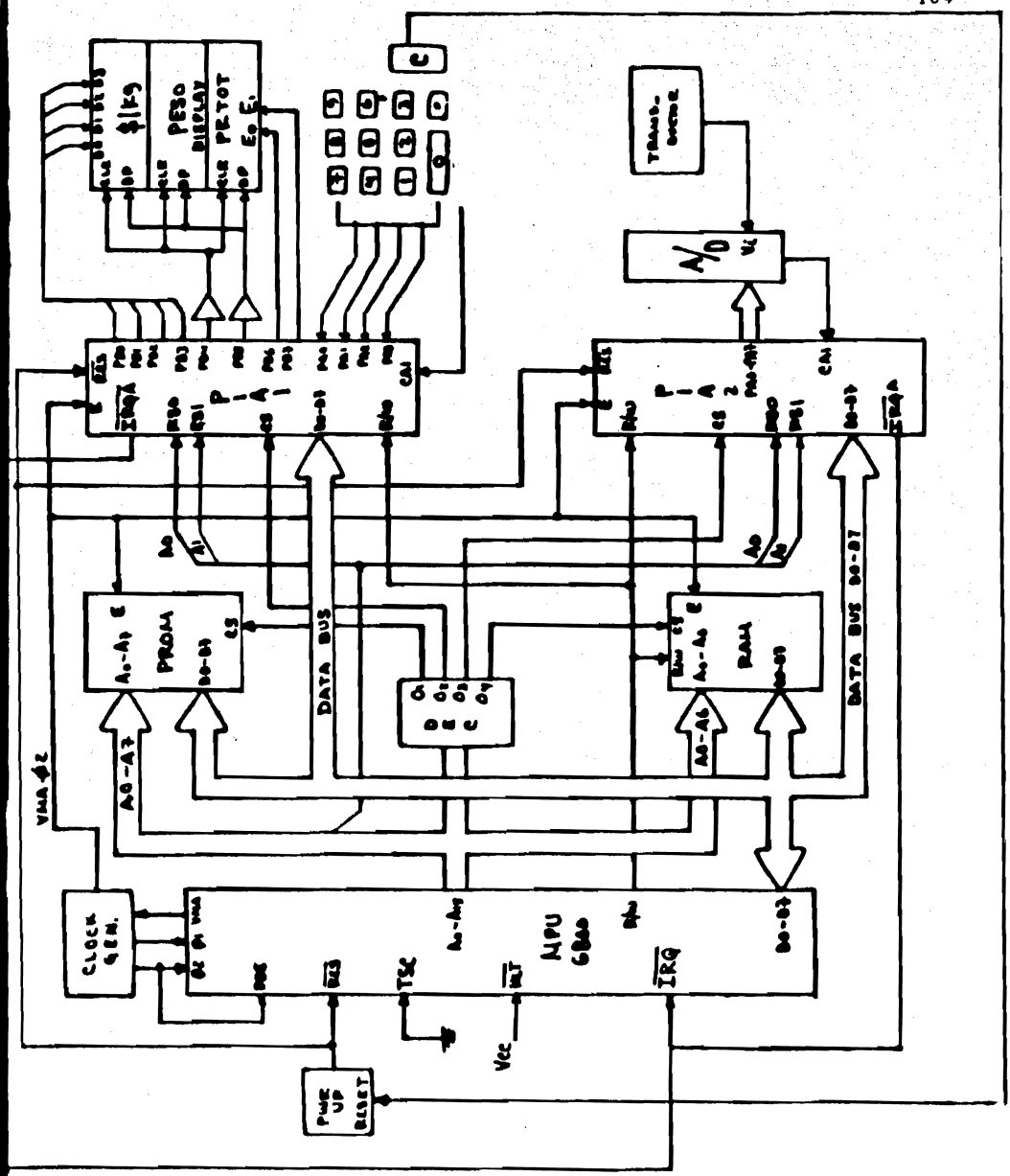


FIGURA VIB.2. DIAGRAMA DEL CIRCUITO EMPLEADO PARA LA EVALUACION DE LA BASCULA.

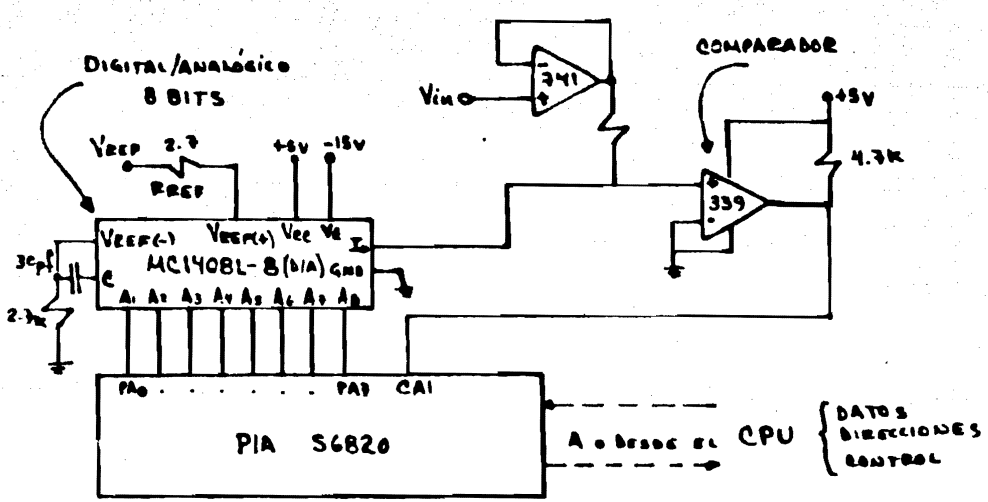


FIGURA VIB.3 CIRCUITO DE A/D EMPLEANDO UN DISPOSITIVO D/A.

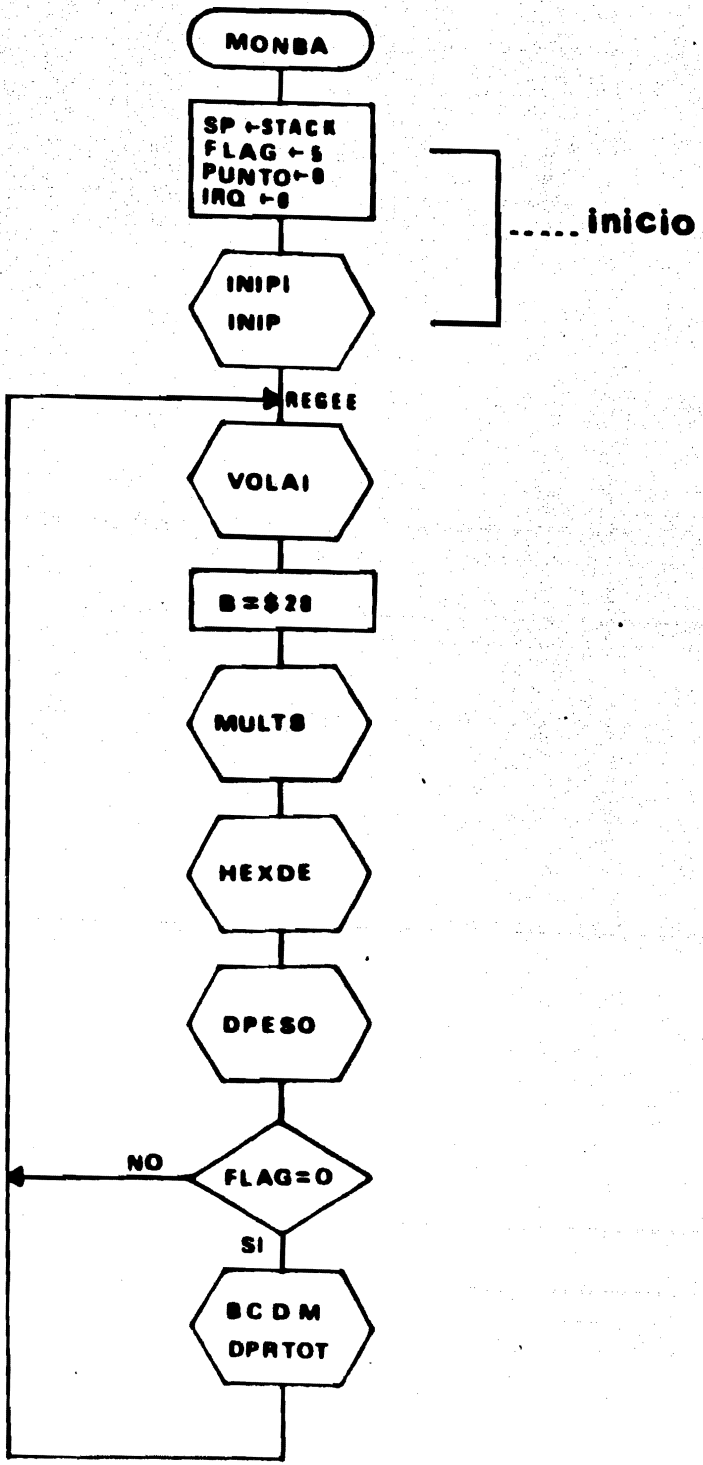


FIGURA VIB.4 DIAGRAMA DE FLUJO DEL PROGRAMA MONBA

3) Cuando se ha completado la secuencia de teclado, el programa se desvía hacia la subrutina BCDM y DPRTOT.

BCDM hace la multiplicación en BCD del precio/kg. marcado por el peso obtenido, y lo almacena en memoria.

DPRTOT toma el contenido correspondiente al resultado de la multiplicación y lo muestra en el DISPLAY; de aquí regresará al punto REGEE.

A partir del punto tres, el programa ignora al operador, hasta que éste, no ha presionado la tecla de CLEAR. Esta genera un pulso de RESET que iniciará la secuencia anterior.

El programa KINTR genera la secuela mostrada en el diagrama de la fig. VI B.5.

4) Cuando el usuario presiona una tecla el programa MONBA es interrumpido, y el control se translada al programa KINTR, éste como primer paso, captura el dato desde la PIA 1. La información es recibida en código BCD, correspondiéndole al punto decimal el valor de OA.

Este programa fue diseñado a prueba de errores, de tal modo que el usuario tiene completa seguridad que el resultado es confiable.

Los errores que se detectan son:

A) El usuario marca más de un punto decimal.
B) El valor entero excede de 999.
C) El valor fraccionario excede de 99.

En los tres casos el exceso es ignorado.

D) El usuario no marca punto decimal.
E) No marca parte fraccionaria.
F) No marca parte entera.

En D y E el programa espera hasta que el usuario no haya tecleado, mientras MONBA seguirá mostrando los cambios que se efectuen en el peso.

En F, el punto decimal define que no hay parte entera y el programa se ejecutará normalmente.

Cada vez que es marcado un dígito o punto decimal, y habiéndose completado la secuencia de captura, el con

trol regresará a MONBA con un RETORNO DE INTERRUPCION (Instrucción RTI, ver apéndice C). Cuando el usuario ha completado la secuela de marcar el precio por Kg. deseado, el programa KINTR, por medio de SEISR, inhibirá el BIT I (ver Cap. I A) de tal modo que no permitirá una interrupción más, hasta que no se haya presionado la tecla de CLEAR.

El listado de esta simulación se da a continuación. Para generar el programa objeto, a partir de la fuente, se utilizó el CROSS ASSEMBLER (ver cap. V B).

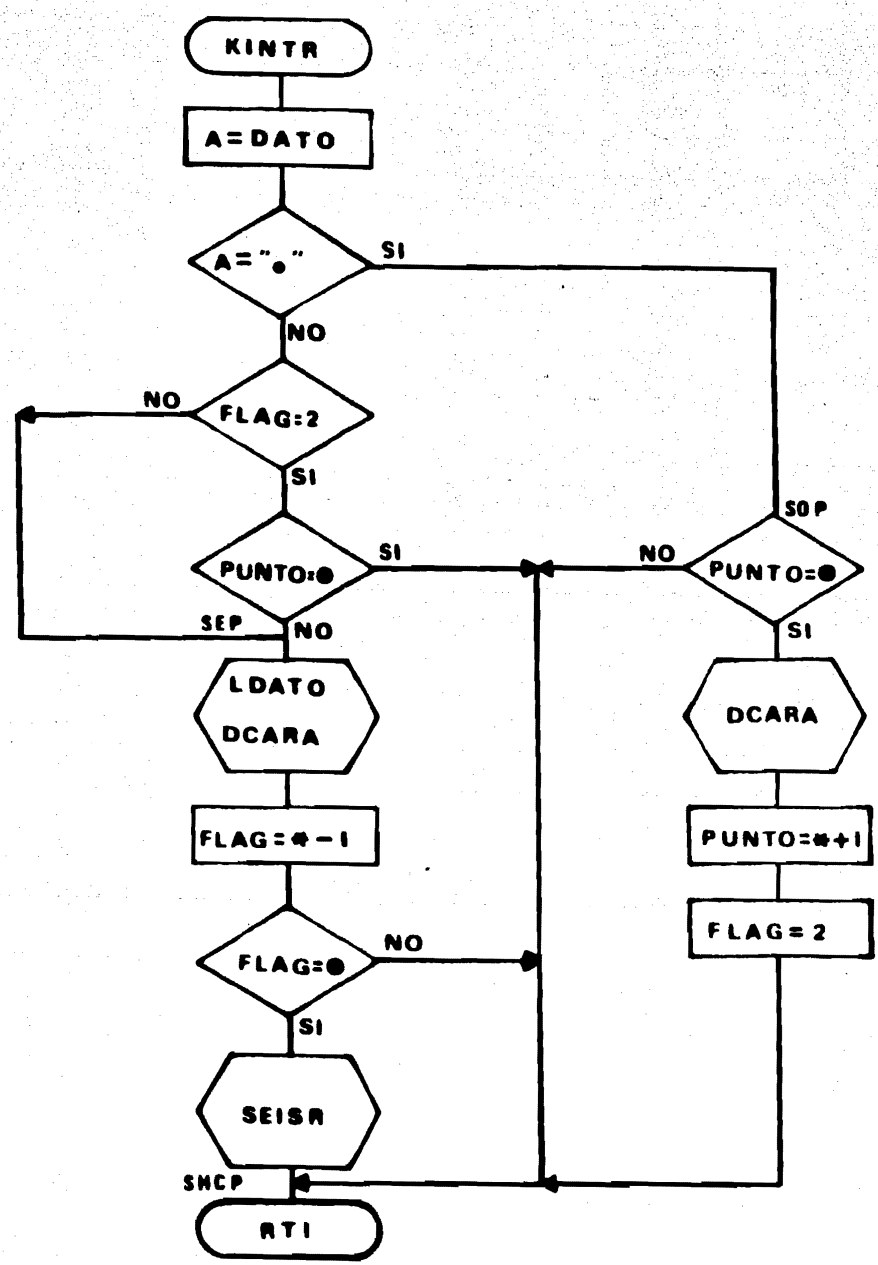


FIGURA VIB.5 DIAGRAMA DE FLUJO DEL PROGRAMA KINTR

```
0000 1  3000        3000              ORG   $3000
*
* * * PROGRAMA PRINCIPAL DE LA EVALUACION * * *
*       * SIMULACION DE UNA PASCUA *       *
*       *      * MONBA *      *      *
*
*
*       SE DEFINE EL VECTOR DE INTERRUPCION.
*
0001C  FFFE        FFFE     IRQV    EQU   $FFFE
*
*       AHORA LAS DIRECCIONES DE
*       LOS REGISTROS DE LAS PIAS
*
00015  FFCE        FFCP     PPA1    EQU   $FFCP
00016  FFC9        FFC9     CPA1    EQU   $FFC9
00017  FFCA        FFCA     PPB1    EQU   $FFCA
00018  FFCB        FFCB     CPB1    EQU   $FFCB
00019  FFCC        FFCC     PPA2    EQU   $FFCC
0001A  FFC1        FFC1     CPA2    EQU   $FFC1
*
*       AHORA EL STACK
*
00024  FF00        FF00     STACK   EQU   $FF00
*
*       INICIO DE MONBA.
*
00028  3000  CE  31FF     MONBA   LDX   #KINTF
00029  3003  FF  FFF8             STX   IRQV
0002B  3006  8E  FF00             LDS   #STACK
00031  3009  8E     25           BSR   LIMP
00032  300B  86     05           LDAA  #$05
00033  300D  B7  31FB             STAA  FLAG
00034  3010  7F  31FC             CLR   PUNTO
00035  3013  8E     27           BSR   INIP1
00036  3015  8E     56           BSR   INIP2
00037  3017  0E                  CLI
00038  3018  8E     61     PEGPE   BSR   VOLA1
00039  301A  06     28           LDAB  #$28
0003A  301C  3F                  SWI
```

FIGURA VIB.6A MOMBA Y KINTR (1)

```
*       CONSTANTE = 17 PARA (SP)3 >INDICE<.
0C042   3C1D          17            FCBE    1,17
0C043   3C1E  8D      7E            BSR     MEDIE
0C044   3C2C  ED  3CD5              JSR     LEESC
0C045   3C23  7D  21FE              TST     FLAG
0C046   3C26  26    F0              BNE     FBGEE
0C047   3C28  BD  30F7              JSR     PCIN
0C048   3C2F  BD  3169              JSR     EES7C
0C049   3C2E  20    EE              BRA     BBGEE
*
*       SUBRUTINA PARA LIMPIAR EL
*       "BUFFER" DE DATOS.
*
0C054   3C3C  CE  312C      LIMP    LDX     #BUFF
0C055   3C33  6F    00      FI      CLR     00,X
0C056   3C35  CB                    INX
0C057   3C36  8C  3201              CPX     #FIN+1
0C058   3C39  26    F8              BNE     FI
0C059   3C3B  39                    RTS
*
*       SUBRUTINA DE INICIACION DE LA
*       PIA1  (TECLADO Y DISPLAY)
*
0C064   3C3C  4F            INIPI   CLRA
0C065   3C3D  B7  FFCB              STAA    CRB1
0C066   3C4C  43                    COMA
0C067   3C41  B7  FFCA              STAA    DRA1
0C068   3C44  86    04              LDAA    #$04
0C069   3C46  B7  FFCB              STAA    CRB1
0C070   3C49  8D    11              BSR     CLPKG
0C071   3C4B  8D    13              BSR     CLRES
0C072   3C4D  8D    15              BSR     CLPRT
0C073   3C4F  4F                    CLRA
0C074   3C5C  B7  FFC9              STAA    CRA1
0C075   3C53  B7  FFC8              STAA    DRA1
0C076   3C56  86    05              LDAA    #$05
0C077   3C5B  B7  FFC9              STAA    CRA1
0C07B   3C5B  39                    RTS
0C079   3C5C  86    7C      CLPKG   LDAA    #$7C
0C07B   3C5E  CC    06              BRA     SALE
```

FIGURA VIB.6B MOMBA Y KINTR (2)

```
0C081   306C  86      50      CLFES   LDAA    #$FC
0C082   306E  20      02              BRA     SALE
0C083   3070  86      F0      CLFET   LDAA    #$FC
0C084   3072  B7    FFCA      SALE    STAA    PRE1
0C085   3075  7F    FFCA              CLR     PRE1
0C086   3078  39                      RTS
*
*       SUBRUTINA DE INICIACION
*       DE LA PIA2 (A/D)
*
0C091   307D  4F              INIP2   CLRA
0C092   307E  B7    FFC1              STAA    CRA2
0C093   3081  43                      COMA
0C094   3082  B7    FFCC              STAA    ????
0C095   3085  86      C4              LDAA    #$C4
0C096   3087  B7    FFC1              STAA    CRA2
0C097   308A  39                      RTS
*
*
*       TABLA PARA EL DISPLAY
*
*       PB7 PB6 PB5 PB4  FUNCION
*
*
*        0   0   0   0   NINGUNA
*        0   0   0   1   N
*        0   0   1   0   N
*        0   0   1   1   N
*        0   1   0   0   N
*        0   1   0   1   P/PG CLEAR
*        0   1   1   0   P/PG LOAD
*        0   1   1   1   P/PG CLEAR
*        1   0   0   0   P
*        1   0   0   1   PB5 CLEAR
*        1   0   1   0   PB5 LOAD
*        1   0   1   1   PB5 CLEAR
*        1   1   0   0   BLASTING
*        1   1   0   1   PB7 CLEAR
*        1   1   1   0   PB7 LOAD
*        1   1   1   1   PB7 CLEAR
```

FIGURA VIB.6C MOMBA Y KINTR (3)

```
*
*       SUBRUTINAS VOLAI Y ABORT
*       OBTENCION DE UN VALOR DEL
*       A/D ( DE SALIDA EN ACC# )
*
0C12E  3075  7F   FECC     VOLAI   CLR    FFA?
0C127  3078  8F                    CLRF
0C128  3079  B6   FECC             LDAA   FFA?
0C129  307C  5C                    INCB
0C13C  307D  C1     FA             CMPB   #$FA
0C131  3081  26     02             BNE    $3C11
0C132  3083  20     0E             BRA    /----
0C133  3085  F7   FECC     $3C11   STAB   FFA?
0C134  3088  81     C9             CMPA   #$C9
0C135  308A  7I   FEC1             BGT    $3C1A
0C136  308D  2A     E5             BPL    FFLOT
0C137  308F  7C   FECC             LDAA   FFA?
0C13E  3092  39                    RTS
*
0C14C  3093  86     CC     ABORT   LDAA   #$CC
0C141  3095  B7   FECA             STAA   FFE1
0C142  3098  20     FE             BRA    ABORT
*
*       SUBRUTINA MOVIE
*       CONVERSION DE MEMA-PCE
*
0C147  309A  36                    MEMDE  PSHA
0C148  309B  CE   31E8             LDX    #$1MC
0C149  309E  86     C4             LDAA   #$CA
0C15C  30A4  B7   31FE             STAA   CONT
0C151  30A7  32                    PULA
0C152  30A8  7F   31FE     MC      CLR    COC
0C153  30AB  7F   31FF             CLR    COC1
0C154  30AE  CC     CA     M1      STDE   #$CA
0C155  30B0  84     C9             ANDA   M2
0C156  30B2  4A                    DECA
0C157  30B3  81     FF             CMPA   #$FF
0C158  30B5  27     0A             BEQ    M2
0C159  30B7  7C   31FE     M2      INC    COC
0C16C  30BA  26     F2             BNE    M1
```

FIGURA VIB.6D MOMBA Y KINTR (4)

```
0C161   30EC   7C   31FF            INC    COC1
0C162   30EF   20     ED            BRA    M1
0C163   30C1   4C           M3      INCA
0C164   30C2   CE     CA            ADDB   #$CA
0C165   30C4   E7     00            STAB   CC,X
0C166   30C6   08                   INX
0C167   30C7   7A   31FE            DEC    CONT
0C168   30CA   27     C8            BEQ    M4
0C169   30CC   F6   31FE            LDAB   COC
0C170   30CF   B6   31FF            LDAA   COC1
00171   30D2   20     D4            BRA    M2
0C172   30D4   39           M4      RTS
*
*       SUBRUTINA PARA MOSTRAR EL PESO OBTENIDO
*
0C176   30D5   BD   30A0    DESC    JSR    CLEAR
0C177   30D8   C6     C1            LDAB   #$C1
0C178   30DA   CE   31ED            LDX    #DIM1
0C179   30DD   09           M1      DEX
0C17C   30DE   A6     00            LDAA   CC,X
00181   30E0   8A     0A            ORAA   #$0A
0C1E2   30E2   B7   FRCA    SRUNT   STAA   PRB1
00183   30E5   8D     0F            BSR    DEL5
CC184   30E7   7F   FRCA            CLR    PRB1
00185   30EA   5A                   DECB
00186   30EB   26     04            BNE    NRUN
0C187   30ED   86     AA            LDAA   #$AA
0C18E   30EF   20     F1            BRA    SRUNT
00189   30F1   8C   31E9    NRUN    CPX    #DIM2
0C190   30F4   26     E7            BNE    DEX1
0C191   30F6   39           DEL5    RTS
*
*       SUBRUTINA PARA MULTIPLICAR EL PESO
*       OBTENIDO POR EL PRECIO/KG.
*
0C196   30F7   86     04    MCDN    LDAA   #$04
0C197   30F9   B7   31FA            STAA   CNT2
0C198   30FC   CE   31E0            LDX    #DIFFO
0C199   30FF   4F                   CLRA
0C200   3100   A7     00    CLL     STAA   CC,X
```

FIGURA VIB.6E MOMBA Y KINTR (5)

```
00201  3102  08              INX
00202  3103  8C  31E9        CPX   #TIM2
00203  3106  26    F8        BNE   CLL
00204  3108  FF  31FE        STX   CONX3
00205  310B  CE  31EC        LDX   #TIFFC
00206  310E  FF  31F6        STX   CONX2
00207  3111  FF  31F4  SALT2  STX   CONX1
00208  3114  FE  31FE        LDX   CONX3
00209  3117  E6    00        LDAB  00,X
00210  3119  CE  31E9        LDX   #TIM2
00211  311C  FF  31F2        STX   CONX0
00212  311F  F7  3000        STAB  TTT
00213  3122  F6  3000  SALT1  LDAB  TTT
00214  3125  4F              CLRA
00215  3126  5D        SAL1  TSTB
00216  3127  27    05        BEQ   SAL2
00217  3129  AB    00        ADDA  00,X
00218  312B  5A              DECB
00219  312C  20    F8        BRA   SAL1
00220  312E  7C  31F3  SAL2  INC   CONX0+1
00221  3131  BD    17        JSR   DAY
00222  3133  FE  31F2        LDX   CONX0
00223  3136  8C  31F2        CPX   #CONX0
00224  3139  26    E7        BNE   SALT1
00225  313B  7C  31F7        INC   CONX2+1
00226  313E  FE  31F6        LDX   CONX2
00227  3141  7C  31F9        INC   CONX3+1
00228  3144  7A  31FA        DEC   TMP
00229  3147  26    C8        BNE   SALT2
00230  3149  39              RTS
*       SUBRUTINA DAY
00232  314A  FE  31F4  DAY   LDX   CONX1
00233  314D  A5    00        ADDA  00,X
00234  314F  A7    00        STAA  00,X
00235  3151  A6    00  SIGP  LDAA  00,X
00236  3153  81    0A  SIG1  CMPA  #$0A
00237  3155  25    06        BLT   SIG
00238  3157  8C    0A        SUBA  #$0A
00239  3159  6C    01        INC   01,X
00240  315B  20    F6        BRA   SIG1
```

FIGURA VIB.6F MOMBA Y KINTR (6)

```
00241  315D  A7    C0      SIG     STAA   00,X
00242  315F  08                    INX
00243  3160  8C  31E0              CPX    #IIMP
00244  3163  26    EC              BNE    SIG2
00245  3165  7C  31F5              INC    CONV1+1
00246  3168  39                    RTS
*
*       SUBRUTINA QUE MUESTRA
*       EL PRECIO TOTAL (IMPTOT).
*
00251  3169  BD  3064    IMPTO   JSR    CLEAR
00252  316C  CE    C3              LDX    #...
00253  316E  CE  31E5              LDX    #IIMP
00254  3171  09                  DIG2  DEX
00255  3172  A6    00              LDAA   00,X
00256  3174  8A    EC              ORAA   #$EC
00257  3176  B7  FFCA      SPI     STAA   PRB1
00258  3179  8D    0F              BSR    DELG
00259  317B  7F  FFCA              CLR    PRB1
00260  317E  5A                    DECB
00261  317F  26    04              BNE    NP1
00262  3181  86    FA              LDAA   #$FA
00263  3183  20    F1              BRA    SP1
00264  3185  8C  31E2      NP1     CPX    #IIMP+3
00265  3188  26    E7              BNE    DIG2
00266  318A  39                  DELG  RTS
*
*       PROGRAMA MINTF
*       ES INVOCADO CUANDO SE DETECTA
*       UNA INTERRUPCION DESDE EL TECLADO.
*
00272  318B  B6  FFCF      MINTF   LDAA   PRA1
00273  318E  84    0F              ANDA   #$0F
00274  3190  81    0A              CMPA   #$0A
00275  3192  27    1B              BEQ    SOF
00276  3194  F6  31EB              LDAB   FLAG
00277  3197  C1    00              CMPB   #$00
00278  3199  26    05              BNE    SSF
00279  319B  7C  31EC              INC    CUNTC
00280  319E  27    0E              BEQ    SMOF
```

FIGURA VIB.6G MOMBA Y KINTR (7)

```
0C281  31A0  8D    1B     SEF     ESF    LDATO
0C282  31A2  8D    2C            ESF    DCAFA
0C283  31A4  7A  31FB            DEC    FLAG
0C284  31A7  26    C2            BNE    SHCP
0C285  31A9  8D    20            ESR    SEISF
0C286  31AB  3F              SHCP  RTI
0C287  31AC  7D  31FC    SOF     TST    PUNTO
0C288  31AF  2E    FA            BNE    SHOF
0C289  31B1  8D    1D            BST    DCAFA
0C290  31B3  7A  31FC            DEC    PUNTO
0C291  31B6  C6    02            LDAB   #$02
0C292  31B8  17  31FB            STAB   FLAG
0C293  31BB  20    EE            BRA    SHCP
*
*      SUBRUTINA PARA ALMACENAR UN DATO
*      PROVENIENTE DEL TECLADO (CUANDO
*      ESTE ES VALIDO).
*
0C299  31BD  CE  31F2    LDATO   LDX    #CONXO
0C300  31C0  09                  DEX
0C301  31C1  09              SBE DEX
0C302  31C2  BC  31ED            CPX    #DIMI
0C303  31C5  2E    C6            BGT    SBE
0C304  31C7  E6    00            LDAB   00,X
0C305  31C9  E7    01            STAB   01,X
0C306  31CB  20    FA            BRA    SBE
0C307  31CD  A7    01      SFA   STAA   01,X
0C308  31CF  39                  RTS
*
*      SUBRUTINA QUE MUESTRA EN EL
*      DISPLAY (EFECTO) UN CARACTER.
*
0C313  31D0  8E    60      DCAFA ADDA   #$60
0C314  31D2  E7  FFCA            STAA   PRBI
0C315  31D5  8D    C2            BSR    DEL2
0C316  31D7  7F  FFCA            CLR    PRBI
0C317  31DA  39              DEL2  RTS
*
*
*
```

FIGURA VIB.6H MOMBA Y KINTR (8)

```
*
*       SUBRUTINA PARA FIJAR EL
*       BIT "I" DEL CCR, CUANDO
*       SE HA COMPLETADO EL CICLO
*       DE OBTENCION DE DATOS DESDE
*       EL MODEM.
*
00328   31E8   3E              SEISR   MVI A
00329   31EA   FA      10              OUT     $10
00330   31EC   3E                      MVI A
00331   31EE   3F                      CMC
*
*       FIN DEL PROGRAMA
*
*       AHORA LAS DECLARACIONES PARA
*       RESERVAR LOCALIDADES DE MEMORIA.
*
00332   31EC           00      DIRBC   DMB     $00
00333   31ED           04      DIM2    DMB     $04
00340   31EE           05      DIM1    DMB     $05
00341   31EF           00      CONTC   DMB     $00
00342   31F0           02      CONST1  DMB     $02
00343   31F1           00      CONST2  DMB     $00
00344   31F2           02      CONST3  DMB     $02
00345   31F3           01      PM2     DMB     $01
00346   31F4           01      FLAG    DMB     $01
00347   31F5           01      FLAGC   DMB     $01
00348   31F6           01      CONT    DMB     $01
00349   31F7           01      CCC     DMB     $01
00350   31F8           01      CCC1    DMB     $01
00351   320C           01      BIB     DMB     $01
00352                          END
**NO ERRORS **
```

FIGURA VIB.6I MOMBA Y KINTR (9)

```
SYMBOL TABLE
IFSV    1FFE
FFAI    FFCF
CFAI    FFC9
FFBI    FBCA
CFBI    FLCF
FFA2    FFCC
CFA2    FFC1
STACK   FL00
MCNBA   3000
FEGEE   3010
LILF    3030
FI      3033
IMIFI   303C
CLFIG   3050
CLFES   3060
CLFFT   3064
SALE    3066
IMIF2   306D
VCLAI   307E
MENOR   3082
SIGUE   3089
LELI    3096
AECFT   3097
RE;DE   309F
L.C     30A8
L.1     30AF
L.2     30B7
L.3     30C1
L.4     30E4
LFESC   30D5
LEJ1    30DD
SFUNT   30E2
NFUN.   30F1
DELS    30F6
SCLL.   30F7
CLL     3100
SALT2   3111
SALT1   3122
SALI    3126
SALG    3135
LA2     314A
SIG2    3151
SIG1    3159
FIC     3161
LFFC    3169
CFFS    3171
LL1     3179
WF1     3185
LELG    3191
LINTE   3198
SEF     31AC
SECP    31AE
SCF     31AC
LLATO   31BE
SFE     31C1
SFA     31CE
LCAFA   31EC
DFL2    31DA
SEISF   31D5
LIFFC   31EC
LIL2    31E9
LIM1    31ED
CONXO   31F2
CONX1   31F4
CONX2   31F6
CONX3   31F8
NNT     31FA
FLAG    31FB
FUNTO   31FC
CONT    31FD
COC     31FE
COC1    31FF
STE     3200
INPUT FFCC.
```

FIGURA VIB.61 TABLA DE SIMBOLOS DE MOMBA Y KINTR

CAPITULO VI C

IMPLEMENTACION DE UN MONITOR CON CAPACIDAD DE 20 COMANDOS.

Parte principal de un sistema computador, es el programa-monitor. Este, entre otras cosas, se encarga de realizar el - enlace máquina-usuario y administrar los puertos de entrada/-salida. Sin esta importante sección, el diálogo sería difícil y tedioso. En sistemas de gran tamaño, como se menciona en el capítulo II B, el monitor es parte importante del llamado SISTEMA OPERATIVO. Unicamente en sistemas de aplicación bien particular se puede prescindir de estos programas.

Con este ejemplo, se trata de dar una visión fundamental-de la implementación de un programa monitor.

Independientemente de las tareas que se le encomendarán,-el cuerpo del monitor es, en realidad, sencillo. BASTE DECIR-que uno muy rudimentario, podría ser almacenado en solo 256 - BYTES de memoria. Este monitor es semejante a otros, como es-natural, sin embargo, tiene grandes ventajas con respecto a - los comerciales, ya que las rutinas de los comandos se diseñan a partir de las necesidades del usuario. Como hemos visto, a través de estas notas (principalmente capítulos II, III y V);-es importante la posibilidad, entre otras tareas, poder manipular más de una terminal o, quizá, tener la posibilidad de hacer uso de un dispositivo de almacenamiento masivo; discos o cintas magnéticas, lectora de cintas de papel, etc. Para lo - cual es necesario un comando que inicie a los elementos antes-mencionados.

La primera parte de MONAMI, nombre de nuestro monitor, incluye las subrutinas necesarias para eslabonar la acción de la máquina con la del usuario. Esta primera parte recibe el nombre de MONAMIX y la secuencia que sigue solo permite el indicar al usuario que está en el ámbito de MONAMI, es decir, por-si solo MONAMIX no ejecuta ninguna acción. La segunda parte - incluye las rutinas que indican la acción a ejecutar, dada por el usuario a través del comando. Estas rutinas se denominan - de acuerdo a la tarea designada.

Ahora veamos como está estructurado MONAMIX.

La primera necesidad que surge en la implementación de un monitor, es el diseñar los manejadores (DRIVERS) apropiados - para el dispositivo que servirá de enlace, en este caso un TTY o similar. De aquí, podemos planear las subrutinas asociadas- al proceso de comunicación, como son: imprimir y leer de la - terminal en cuestión.

El programa MONAMIX se encarga de realizar las funciones- antes mencionadas. El ACIA, ver capítulo I B.5, es el dispositivo ideal para este acoplamiento; su versatilidad simplifica- el diseño del manejador y nos permite el desarrollo de una - gran variedad de rutinas.

El programa MONAMIX se lista en páginas posteriores y sus funciones son como sigue, a partir del número de línea indicado en el extremo izquierdo del listado.

No. DE LINEA		FUNCION
40	210	(SCRATC). Se reserva el área de trabajo en donde, MONAMI, hará el almacenaje y lectura de los parámetros - que se requieran al correr las rutinas de comandos.
220	330	Se declaran las etiquetas que corresponden a direcciones constantes; como los registros del ACIA o señales- de control del TTY. NOTA: La subrutina PINIT solo es usada para iniciar la PIA del programador de PROMS, y fue implementada - con anterioridad.
340	610	(MONIT). Este es el programa principal del monitor, y se vale de las - subrutinas que conforman el resto de MONAMIX para establecer la comunicación.
360	430	Se almacenan los registros del MPU,- para que cuando al llevarse a cabo - una interrupción de RESET, éstos puedan ser solicitados por el usuario.
450	480	Se inicializa el ACIA para establecer la comunicación.

500	570	El programa da aviso al usuario de - que ya entró en su ámbito.
580		Se lee un RECORD proveniente del TTY. Este consta de un máximo de 72 caracteres y debe finalizar con una señal de LF.
590	600	El RECORD leído pasa al proceso de - verificación, a partir de los comandos implementados. Al regreso de esta subrutina el BIT C del CCR, será nivel cero si el comando no es válido.
610		Se salta a la rutina que ejecutará - el comando.
620	650	(ABORT). Cuando un error es detectado, en cualquier punto del monitor, ésta subrutina imprimirá un mensaje al usuario.
660	710	(PRTCL). Imprime un LF y CR, adicionando cinco señales de NULL, para - dar tiempo a que el "carro" del TTY- se estabilice.
720	950	(PRINX y PRINT). La primera subrutina imprime el contenido de la dirección en el registro índice, y la segunda el contenido en el acumulador- A.
960	1020	(PRINA). Esta subrutina se encarga- de manejar al TTY en lo que respecta a la impresión.
1030	1160	(RBUFF). Esta es la encargada de - leer y almacenar, en el archivo REC, la línea enviada por el usuario a - través del TTY.
1170	1250	(ESPCA). Aguarda a que el usuario - teclee en el TTY.
1260	1310	(LEEAC). Esta se encarga de manejar al TTY en lo que respecta a la lectura de un carácter.

1320	1410	(BORNT). Es el inicio de una tabla de correspondencia. Esta tabla es empleada por VBUFF para relacionar, los dos caracteres que componen un comando, con la dirección de inicio de la rutina que corresponde al comando dado. Para ello son reservadas 50) mex. localidades de memoria, menos las correspondientes a las rutinas ya implementadas.
1420	1630	(VBUFF). Se encarga de validar el RECORD. Este es analizado y comparado con la tabla antes mencionada, cuando se ha encontrado el conjunto de dos caracteres que componen el comando; VBUFF almacena en (X) el contenido de las dos localidades que siguen y regresa al programa principal. De no ser asi, VBUFF hará nivel cero el contenido del BIT C del CCR indicando que no se ha encontrado relación alguna.
1640	1730	(PRM). Imprime un mensaje, cuyo inicio está apuntado por el contenido en X.
1740	1760	(MER). Son localidades en memoria que contienen el mensaje de error utilizado en ABORT.
1770	1820	(DTCSR). Esta subrutina se encarga de obtener el carácter que es apuntado por el contenido en RECL (apuntador del RECORD).
1830	1970	Estas declaraciones indican los tres comandos que se han implementado.

Las tres rutinas, que corresponden a los comandos AB, CD y DM, son como sigue:

1950	1960	(RUT 2). La rutina dos se implementó solo para probar a MONAMIX. Su función solo consiste, una vez verificado, regresar a control. Es decir, cuando el usuario da el comando CD, VBUFF valida el comando y trans-

		fiere a RUT2, declaración 610; que a su vez regresa a control (FMON). - Punto desde el cual el usuario tiene la oportunidad de volver a dar otra- orden.
1830	1940	(RUT 1). Esta rutina también fué -- implementada para probar algunas partes de MONAMIX. Al detectar el co-- mando AB, MONAMIX transfiere el control a RUT 1: éste verificará si después de AB el usuario a dado el ca-- rácter "," (coma); si es así, RUT 1 imprimirá el argumento de AB. Ver - ejemplo de la corrida en la figura - VI C.1.

La declaración 1970, indica el inicio de la primera rutina formal (RUT 3); esta corresponde al comando DM. A partir - de este punto se podrán adicionar las rutinas que se necesiten. En MONAMIX bastará con insertar cuatro BYTES dentro de la ta-- bla de correspondencia; dos para los caracteres que identifi-- quen al comando (RX, PL, etc.) y las dos restantes para la dirección de inicio de la rutina correspondiente.

Se hace notar que la limitación de 20 comandos, se puede- deshechar; ya que solo se necesita reservar cuatro localidades más para cada comando que se quiera agregar.

Este monitor tiende a ser más elaborado que el propio de- AMI (PROTO, ver capítulo II B), sus comandos serán más poderosos y permitirán mejor aprovechamiento del sistema.

Se incluirán comandos como:

A) AS, UL, UF/MRS.

Donde:

AS es un comando de Asignación que indica que la unidad- UL se asigna a la UF; es decir por ejemplo: UL unidad de im-- presión, es asignada a la unidad UF; que puede ser el TTY o la impresora. De tal manera que después de ejecutado este comando cualquier orden de impresión será ejecutada en UF.

MRS indica el máximo tamaño que puede alcanzar el RECORD- a transmitir, por ejemplo, 72 ó 40 caracteres, etc. Esto debido a que las terminales generalmente no tienen la misma capacidad en línea.

B) IM, DIRECCION/OPCION

Donde:

IM denota un comando que significa Inserción en Memoria.

DIRECCION indica la dirección de inicio de la acción.

OPCION determina la opción: Los artículos que se insertarán podrán estar dados como dígitos hexadecimales (similar al comando S de AMI, ver capítulo II B); o también como caracteres ASCII.

El comando DM, correspondiente a la rutina tres (RUT 3), hace uso de algunas subrutinas incluidas en MONAMIX; así mismo se agregaron otras más. De este modo se hará para cada comando adicional; solo que éstos últimos, a medida que su número aumente, requerirán cada vez menos soportes adjuntos, ya que posiblemente éste esté incluido en los comandos ya implementados.

Esta rutina hace un trabajo de verificación muy exhaustivo; el comando en su forma general es como sigue:

DM, DIRECCION, NUMERO / OPCION (H, A)

Donde:

DM es un comando que indica Despliegue de Memoria.

DIRECCION significa la dirección de inicio del despliegue.

NUMERO denota el número de localidades a desplegar.

OPCION indica la opción: H el despliegue se hace en formato hexadecimal (8 por línea).
A se hace el despliegue de los caracteres, en código ASCII, empezando en DIRECCION (32) dec por línea).

Condiciones por Defacto (DEFAULT).

DIRECCION = 00

NUMERO = 01

OPCION = HEXADECIMAL

De esta manera, las siguientes líneas denotan la misma acción, todas ellas son válidas.

DM, 0000, 0001/H

DM, 0000,/H

DM, 0, 1/H

DM, 0,/H

DM,,

DM (Despliegue del contenido de la localidad 0 - en formato hexadecimal).

Al final de cada comando se debe dar la señal LF.

La secuencia de verificación es mostrada en los Diagramas del Flujo que corresponden a las figuras VI C.2, 3 y 4. Cualquier error detectado origina una transferencia a ABORT. Cuando el proceso termina se ejecuta la acción a partir de la rutina EXECT.

El listado que corresponde a esta rutina se da a continuación. Hay que hacer notar que ésta es más extensa que el programa principal; pero también cabe mencionar que la mayoría de las subrutinas que componen RUT 3, se usarán en las rutinas de comandos que se incluyan posteriormente.

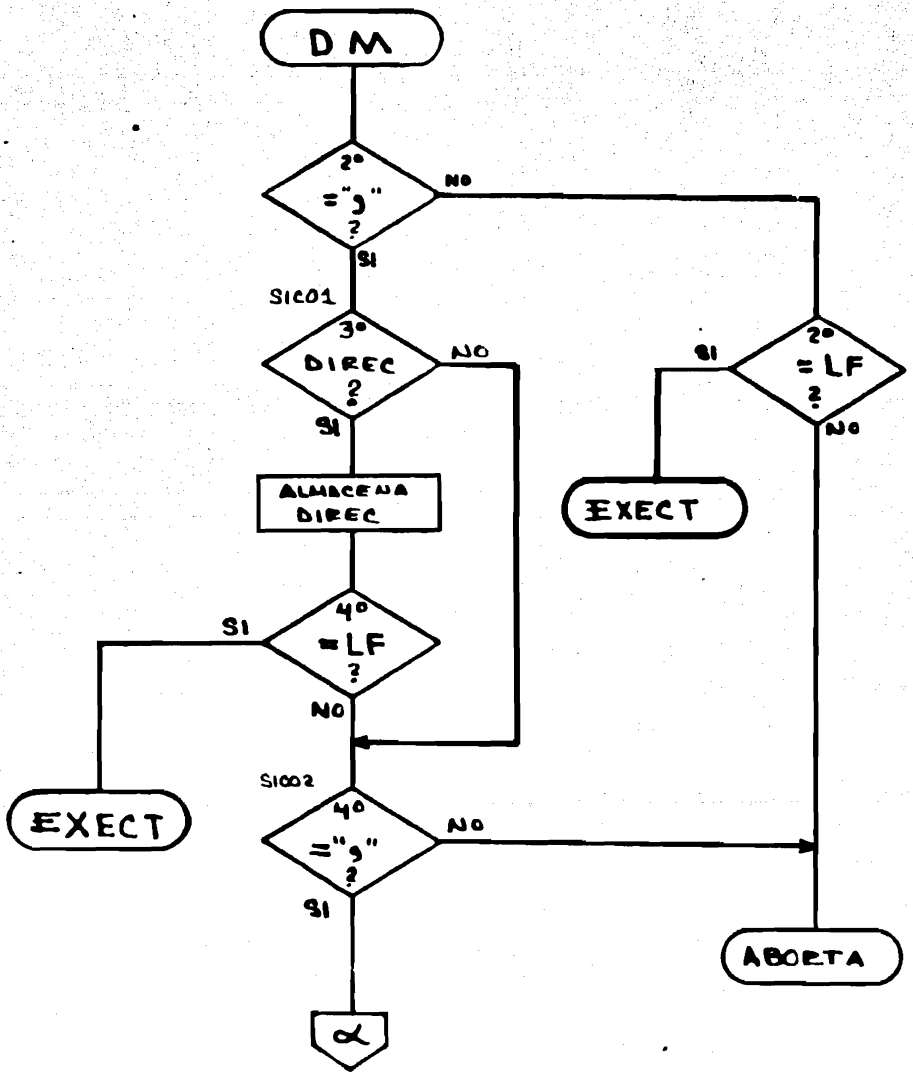


FIGURA VIC.2 DIAGRAMA DE FLUJO DE LA PARTE DE VERIFICACION DE LA RUTINA RUT3 QUE CORRESPONDE AL COMANDO AM (1).

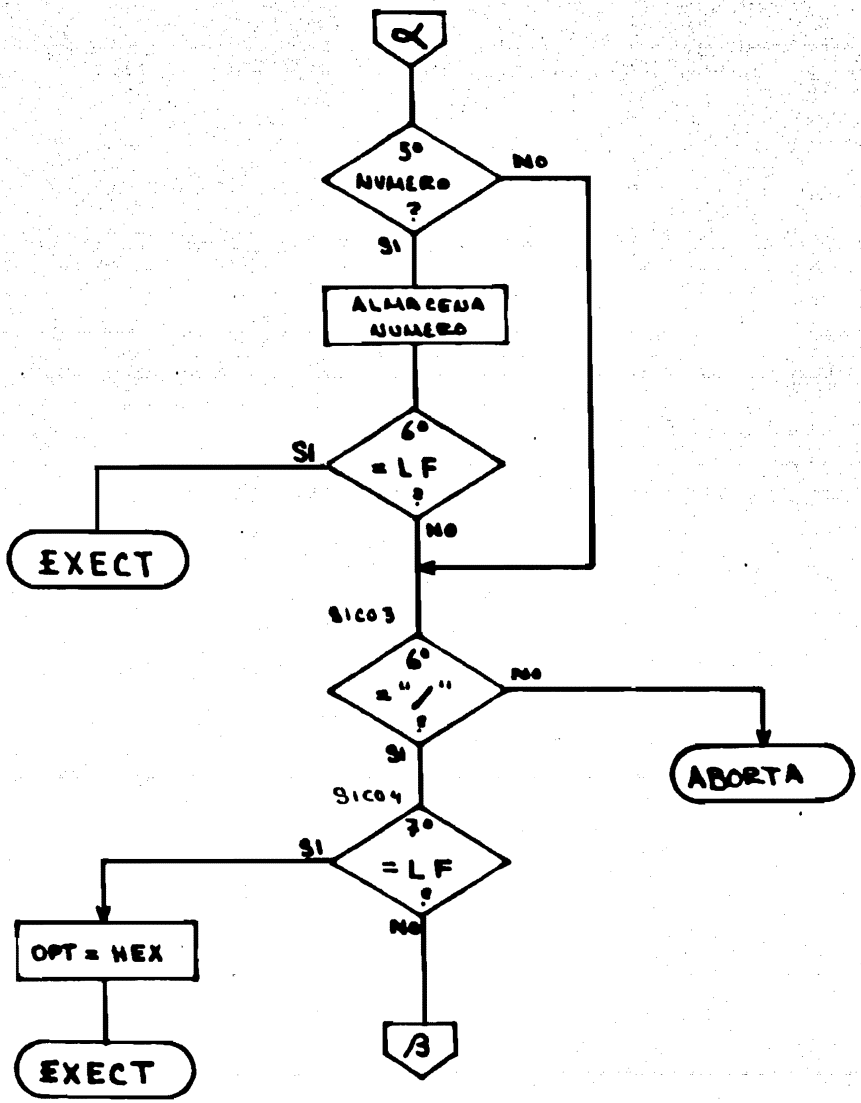


FIGURA VIC.3 DIAGRAMA DE FLUJO DE LA PARTE DE VERIFICACION DE LA RUTINA RUT3 (2).

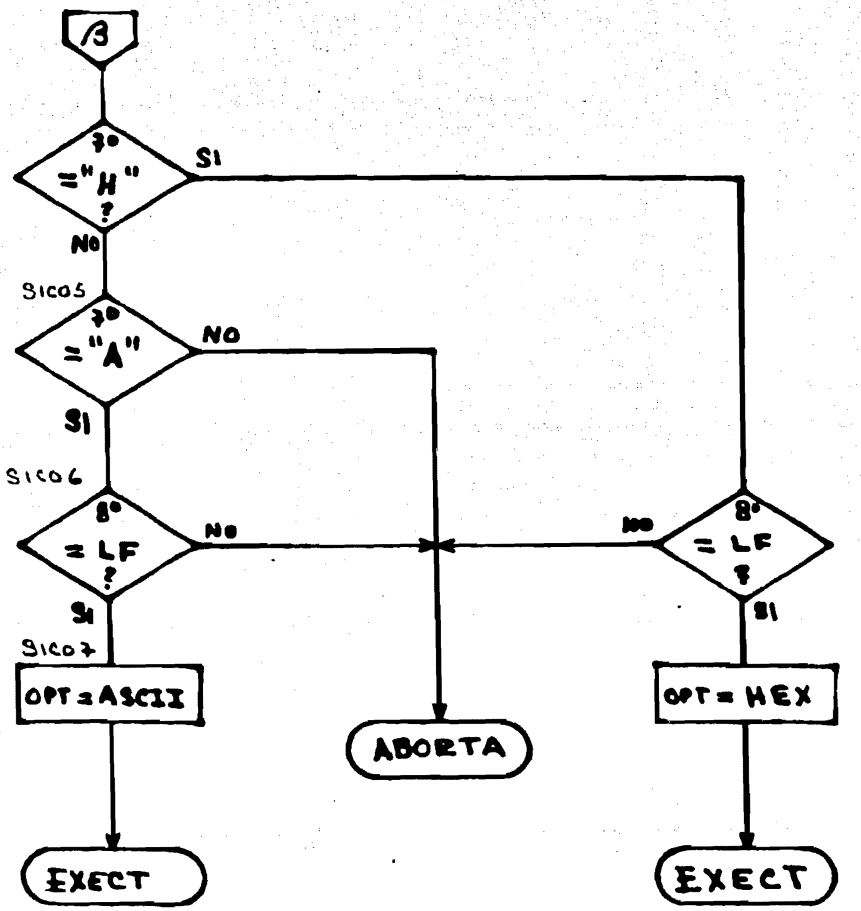


FIGURA VIC.4 DIAGRAMA DE FLUJO DE LA PARTE DE VERIFICACION DE LA RUTINA RUT3 (3)

```
0001C                          NAM     MONAMIX
00020                          OPT     NOO,NOS,NOC
00030 3F90                     ORG     $3F90
00040         3F90   SCRATC  EQU     *       AREA DE TRABAJO
00050 3F90 0048   REC    RMB     72      LONGITUD DEL RECORD
00060 3FD8 0002   DIF    RMB     2       DIFERENCIA
00070 3FDA 0002   DIREC  RMB     2       DIRECCION (COMANDOS)
00080 3FDC 0002   DIREL  RMB     2       DIR BAJA
00090 3FDE 0002   DIREH  RMB     2       DIR ALTA
00100 3FE0 0002   RECL   RMB     2       APUNTADOR DEL RECORD
00110 3FE2 0003                RMB     3       RESERVA
00120 3FE5 0002   SPCEL  RMB     2       ALMACEN DE SP
00130 3FE7 0002   RXCEL  RMB     2       ALMACEN DE X
00140 3FE9 0001   COUNT  RMB     1       CONTADOR
00150 3FEA 0001   COUN   RMB     1       CONT AUX
00160 3FEB 0001   CCRAD  RMB     1       DIRECCION DEL CCF
00170 3FEC 0001   ACBAD  RMB     1       DIRE DE ACCB
00180 3FED 0001   ACAAD  RMB     1       DIRE DE ACCA
00190 3FEE 0002   RXAD   RMB     2       DIRE DE X
00200 3FF0 0002   PCAD   RMB     2       DIRE DE PC
00210 3FF2 0002   SPAD   RMB     2       DIRE DE SP
00220         FD00   STACK  EQU     $FD00   DIRE INICIO DE STACK
00230         FBCE   CRACI  EQU     $FBCE   REG DE CONTROL ACIA
00240         FBCE   SRACI  EQU     $FBCE   REG DE ESTADO ACIA
00250         FBCF   DRACI  EQU     $FBCF   REG DE DATOS ACIA
00260         F416   PINIT  EQU     $F416   SUBR INICIA PROG
00270         0007   BELL   EQU     $07     CLINK... (TTY)
00280         000A   LF     EQU     $0A     UN RENGLON MAS
00290         000D   CR     EQU     $0D     REGRESA EL CARRO
00300         001B   ESC    EQU     $1B     DETENTE Y REGRESA
00310         007F   DEL    EQU     $7F     IGNORADO
00320         0004   EOT    EQU     $04     FIN DEL TEXTO
00330         0020   SPACE  EQU     $20     ESPACIO
00340 3000                     ORG     $3000
00350         3000   MONIT  EQU     *       ORIGEN DEL MONITOR
00360 3000 36         MONIT  PSH A           *SALVA REGISTROS
00370 3001 07                  TPA
00380 3002 B7 3FEB             STA A   CCRAD   *1. CCR
00390 3005 32                  PUL A
00400 3006 B7 3FED             STA A   ACAAD   *2. ACCA
00410 3009 F7 3FEC             STA B   ACBAD   *3. ACCB
00420 300C FF 3FEE             STX     RXAD    *4. X
00430 300F BF 3FF2             STS     SPAD    *5. SP
00440 3012 8E FD00             LDS     #STACK  INICIA SP
00450 3015 86 03               LDA A   #03     *INICIA ACIA
00460 3017 B7 FBCE             STA A   CRACI   *
00470 301A 86 01               LDA A   #01     *
00480 301C B7 FBCE             STA A   CRACI   *
00490 301F BD F416             JSR     PINIT   INICIA PROGRAMADOR
00500 3022 BD 3048             JSR     PRTCL   DA UN CR Y LF
00510         3025   FMON   EQU     *       INICIO DE CONTROL
00520 3025 8E FD00 FMON   LDS     #STACK  REINICIO DE SP
00530 3028 B6 FBCF             LDA A   DRACI   BORRA EL CARAC ANTERIOR
00540 302B 86 07               LDA A   #BELL   *IMPRIME "CLINK... Y * *
```

```
00550 302D BD 3052          JSR     PRINT     •
00560 3030 86 2A            LDA A   #'*       •
00570 3032 BD 3052          JSR     PRINT     •
00580 3035 BD 3089          JSR     RBUFF     LEE UN RECORD
00590 3038 BD 3114          JSR     VBUFF     VALIDA EL RECORD
00600 303B 24 02            BCC     ABORT     VALIDO?
00610 303D 6E 00            JMP     00,X      SI VAL. EJECUTA EL COMANDO
00620      303F     ABORT   EQU     •         COMANDO NO VALIDO
00630 303F CE 314A ABORT    LDX     #MER      •IMP MENSAJE DE ERROR
00640 3042 BD 313B          JSR     PRM       •
00650 3045 7E 3025          JMP     FMON      REGRESA CONTROL
00660      3048     PRTCL   EQU     •         SUBR IMPRIME CR,LF Y 5 NULLS
00670 3048 36       PRTCL   PSH A
00680 3049 86 0A            LDA A   #LF
00690 304B BD 3052          JSR     PRINT
00700 304E 32               PUL A
00710 304F 39               RTS
00720      3050     PRINX   EQU     •         SUBR AL TTY EL CARAC EN X
00730 3050 A6 00    PRINX   LDA A   00,X
00740      3052     PRINT   EQU     •         SUBR AL TTY EL CARAC EN ACCA
00750 3052 37       PRINT   PSH B             SALVA EL ACCB
00760 3053 F6 FBCE          LDA B   SRACI
00770 3056 57               ASR B
00780 3057 24 0A            BCC     PRIT
00790 3059 F6 FBCF          LDA B   DRACI
00800 305C C1 1B            CMP B   #ESC      SCAPE?
00810 305E 26 03            BNE     PRIT
00820 3060 7E 303F          JMP     ABORT     SI, ABORTA
00830 3063 BD 307C PRIT     JSR     PRINA
00840 3066 81 0A            CMP A   #LF       ES LF?
00850 3068 26 10            BNE     RTSI
00860 306A 86 0D            LDA A   #CR       SI, ADICIONA CR Y 5 NULLS
00870 306C BD 307C          JSR     PRINA
00880 306F 4F               CLR A
00890 3070 C6 05            LDA B   #05
00900 3072 BD 307C CAD      JSR     PRINA
00910 3075 5A               DEC B
00920 3076 26 FA            BNE     CAD
00930 3078 86 0A            LDA A   #LF
00940 307A 33       RTSI    PUL B             RECUPERA ACCB
00950 307B 39               RTS
00960 307C 36       PRINA   PSH A             •ACCA ACIA
00970 307D B6 FBCE FREE     LDA A   CRACI     •
00980 3080 85 02            BIT A   #02       •
00990 3082 27 F9            BEQ     FREE      •
01000 3084 32               PUL A             •
01010 3085 B7 FBCF          STA A   DRACI     •
01020 3088 39               RTS               •
01030      3089     RBUFF   EQU     •         SUBR LEE UN RECORD DE TTY
01040 3089 CE 3F9C RBUFF    LDX     #REC
01050 308C FF 3FE0          STX     RECL      SE GUARDA
01060 308F 8C 3FD7 NEXT0    CPX     #REC+71   FIN DEL RECORD?
01070 3092 26 03            BNE     NEXT1
01080 3094 7E 303F          JMP     ABORT     • DE 72 !..FUERA!!!
```

```
01090 3097 BD 30A6 NEXT1  JSR     ESPCA     VE A ESPERAR UN CARAC.
01100 309A 81 0D          CMP A   #CR       CR? IGNORALO
01110 309C 27 F9          BEQ     NEXT1
01120 309E A7 00    NEXT2 STA A   0,X       UN CARAC/?,GUARDALO
01130 30A0 08             INX
01140 30A1 81 0A          CMP A   #LF       ES LF?
01150 30A3 26 EA          BNE     NEXT2
01160 30A5 39             RTS               SI,TERMINAMOS
01170      30A6     ESPCA  EQU     *         ESPERA UN CARACTER DEL ACIA
01180 30A6 BD 30B8 ESPCA  JSR     LEEAC     VE A LEERLO
01190 30A9 81 1B          CMP A   #ESC      ES SCAPE?
01200 30AB 26 03          BNE     ABRT1
01210 30AD 7E 303F        JMP     ABORT     SI, ..FUERA!
01220 30B0 81 7F    ABRT1 CMP A   #DEL      ES RUBAUT?
01230 30B2 27 F2          BEQ     ESPCA     SI, IGNORALO
01240 30B4 BD 3052        JSR     PRINT     DA ECCCCCO
01250 30B7 39             RTS               Y REGRESA
01260 30B8 B6 FBCE LEEAC  LDA A   SRACI     LEE DEL ACIA
01270 30BB 47             ASR A
01280 30BC 24 FA          BCC     LEEAC     YA ESTA LLENO RDRF ?
01290 30BE B6 FBCF        LDA A   DRACI     OK
01300 30C1 84 7F          AND A   #$7F      SE DEPURA
01310 30C3 39             RTS
01320      30C4     BORNT  EQU     *         TABLA DE CORRESPONDENCIA
01330 30C4 41             FCC     /AB/ COMANDO (AB)

01340 30C6 3159           FDB     RUT1
01350 30C8 43             FCC     /CD/ COMANDO (CD)

01360 30CA 3173           FDB     RUT2
01370 30CC 44             FCC     /DM/ COMANDO (DM)

01380 30CE 3176           FDB     RUT3
01390      30D0     ENDTA  EQU     *
01400 30D0 0044           RMB     BORNT+$50-ENDTA
01410      3114     ENDTD  EQU     *
01420      3114     VBUFF  EQU     *         SUBR VALIDA EL RECORD
01430 3114 FE 3FE0 VBUFF  LDX     RECL
01440 3117 A6 00          LDA A   0,X       PRIMER CARACTER DEL COMANDO
01450 3119 E6 01          LDA B   1,X       SEGUNDO CARACTER DEL IDEM
01460 311B CE 30C4        LDX     #BORNT    APUNTA LA TABLA DE CORRESP
01470 311E A1 00    ELSI  CMP A   0,X       *COMPARACION
01480 3120 26 04          BNE     SIGUI     *
01490 3122 E1 01          CMP B   01,X      *
01500 3124 27 0B          BEQ     IGUAL     LO ENCONTRO
01510 3126 08       SIGUI INX               NO LO ENCUENTRA TODAVIA
01520 3127 08             INX
01530 3128 08             INX
01540 3129 08             INX
01550 312A 8C 30D0        CPX     #ENDTA    SE LE TERMINO EL RECORD?
01560 312D 26 EF          BNE     ELSI
01570 312F 0C             CLC               SI!....FUERA!!!
01580 3130 39             RTS
01590 3131 EE 02    IGUAL LDX     2,X       YA LA HIZO JER,JER.
01600 3133 7C 3FE1        INC     RECL+1    *APUNTA EL SIG. CARACTER
01610 3136 7C 3FE1        INC     RECL+1    *
01620 3139 0D             SEC
```

```
0163@ 313A 39              RTS             VAMONOS ..
0164@      313B     PRM    EQU    *        SUBR IMPRIME UN TEXTO
0165@ 313B 36       PRM    PSH A           EMPEZANDO CON (X)
0166@ 313C A6 @@    PRM1   LDA A  @,X
0167@ 313E 81 04           CMP A  #EOT     FIN DEL TEXTO?
0168@ 3140 27 @6           BEQ    TIX
0169@ 3142 BD 3050         JSR    PRINX    NO, IMPRIME ((X))
0170@ 3145 08              INX
0171@ 3146 2@ F4           BRA    PRM1
0172@ 3148 32       TIX    PUL A
0173@ 3149 39              RTS             TERMINO
0174@      314A     MER    EQU    *        MENSAJE DE ERROR
0175@ 314A 4E       MER    FCC    /NSG/

0176@ 314D @7              FCB    BELL,LF,EOT
0177@      315@     DTCSR  EQU    *        ESPACIO,LF,COMA O DIAGONAL ?
0178@      315@     DTCSR  EQU    *        CARGA EN ACCA EL CARACTER EN

0179@ 315@ FE 3FE@ DTCSR  LDX    RECL
0180@ 3153 A6 @@           LDA A  @,X
0181@ 3155 7C 3FE1         INC    RECL+1
0182@ 3158 39              RTS
0183@      3159     RUT1   EQU    *        COMANDO *AB,ARGUMENTO*
0184@ 3159 BD 315@ RUT1   JSR    DTCSR    IMPRIME EL ARGUMENTO
0185@ 315C 81 2C           CMP A  #',
0186@ 315E 27 @3           BEQ    CONTF
0187@ 316@ 7E 303F         JMP    ABORT    LE FALTA LA COMA,FUERA..!!!
0188@ 3163 FE 3FE@ CONTF  LDX    RECL     APUNTA EL RECORD
0189@ 3166 BD 3050 CONTD  JSR    PRINX
0190@ 3169 A6 @@           LDA A  @@,X
0191@ 316B @8              INX
0192@ 316C 81 @A           CMP A  #LF
0193@ 316E 26 F6           BNE    CONTD
0194@ 3170 7E 3025         JMP    FMON     REGRESA A CONTROL
0195@      3173     RUT2   EQU    *        COMANDO *CD*
0196@ 3173 7E 3025 RUT2   JMP    FMON     REGRESA A CONTROL
0197@      3176     RUT3   EQU    *
0198@                     ********************************* HECTOR/79

0199@                     END

TOTAL ERRORS   @@@
```

```
00010                          NAM    RUT3
00020                          OPT    O,NOS
00030          303F    ABORT   EQU    $303F
00040          3048    PRTCL   EQU    $3048
00050          3052    PRINT   EQU    $3052
00060          3025    FMON    EQU    $3025
00070          3150    DTCSR   EQU    $3150
00080          3FD8    DIF     EQU    $3FD8
00090          3FDA    DIREC   EQU    $3FDA
00100          3FE0    RECL    EQU    $3FE0
00110          3FE7    RXCEL   EQU    $3FE7
00120          3FE9    COUNT   EQU    $3FE9
00130          3FEA    COUN    EQU    $3FEA
00140          000A    LF      EQU    $0A
00150          0020    SPACE   EQU    $20
00160 3176                     ORG    $3176
00170          3176    RUT3    EQU    *          COMANDO 'DM'
00180 3176 CE 0000     DMSR    LDX    #0000
00190 3179 FF 3FDA             STX    DIREC      DIREC DEFACTO
00200 317C FF 3FE9             STX    COUNT      INICIA CONTADORES
00210 317F 08                  INX
00220 3180 FF 3FD8             STX    DIF        INICIA DIF DEFACTO=1
00230 3183 BD 3150             JSR    DTCSR      QUE CARACTER SIGUE
00240 3186 81 2C               CMP A  #',        ES COMA ?
00250 3188 27 07               BEQ    SICO1      SI ES ENTONCES SICO1
00260 318A 81 0A               CMP A  #LF        NO?, SERA LF
00270 318C 27 15               BEQ    EXEC0      SI ES ENTONCES EJECUTA
00280 318E 7E 303F             JMP    ABORT      NO, ENTONCES FUERA!!
00290 3191 BD 3257 SICO1       JSR    ARAM
00300 3194 24 10               BCC    SICO20
00310 3196 B7 3FDA             STA A  DIREC
00320 3199 F7 3FDB             STA B  DIREC+1
00330 319C BD 3150             JSR    DTCSR      DAME OTRO CARACTER
00340 319F 81 0A               CMP A  #LF        ES LF?
00350 31A1 26 06               BNE    SICO2      DIFERENTE?, ENTONCES SICO2
00360 31A3 7E 32AA EXEC0       JMP    EXECT
00370 31A6 BD 3150 SICO20      JSR    DTCSR      ONE MORE TIME
00380 31A9 81 2C   SICO2       CMP A  #',        SERA COMA-CHA CA CHA CHAN
00390 31AB 27 03               BEQ    ALFA       SI, ENTONCES ALFA
00400 31AD 7E 303F             JMP    ABORT      NO?,MUERETE!..JI JI
00410          31B0    ALFA    EQU    *          !!!! A L F A !!!!
00420 31B0 BD 3257 ALFA        JSR    ARAM
00430 31B3 24 10               BCC    SICO30     NO?, ENTONCES SICO30
00440 31B5 B7 3FD8             STA A  DIF        SI SI SI
00450 31B8 F7 3FD9             STA B  DIF+1      JI JI JI
00460 31BB BD 3150             JSR    DTCSR      OTRO CARACTER MAS
00470 31BE 81 0A               CMP A  #LF        SERA LF
00480 31C0 26 06               BNE    SICO3      DIFERENTE?, ENTONCES SICO3
00490 31C2 7E 32AA             JMP    EXECT      IGUAL, ENTONCES EJECUTA
00500 31C5 BD 3150 SICO30      JSR    DTCSR      UNO MAS
00510 31C8 81 2F   SICO3       CMP A  #'/        SERA DIAGONAL
00520 31CA 27 03               BEQ    SICO4
00530 31CC 7E 303F             JMP    ABORT      JE JE JE JE
00540 31CF BD 3150 SICO4       JSR    DTCSR
```

```
00550 31D2 81 0A          CMP A  #LF
00560 31D4 26 03          BNE    BETA
00570 31D6 7E 32AA        JMP    EXECT
00580       31D9    BETA  EQU    *          :::: B E T A ::::
00590 31D9 81 48    BETA  CMP A  #'H
00600 31DB 26 0A          BNE    SIC05      DIFERENTE DE "H", SI SIC05
00610 31DD BD 3150        JSR    DTCSR      NO DAME OTRO CARACTER
00620 31E0 81 0A          CMP A  #LF
00630 31E2 27 BF          BEQ    EXEC0      IGUAL A LF, EJECUTA
00640 31E4 7E 303F        JMP    ABORT      DIFERENTE?, A B O R T A
00650 31E7 81 41    SIC05 CMP A  #'A
00660 31E9 27 03          BEQ    SIC06      IGUAL?, ENTONCES SIC06
00670 31EB 7E 303F        JMP    ABORT
00680 31EE BD 3150  SIC06 JSR    DTCSR      UNO MAS
00690 31F1 81 0A          CMP A  #LF
00700 31F3 27 03          BEQ    SIC07      IGUAL ENTONCES SIC07
00710 31F5 7E 303F        JMP    ABORT      NO A B O R T A
00720 31F8 7C 3FE9  SIC07 INC    COUNT      =(COUNT = 1, ASCII)=
00730 31FB 7E 32AA        JMP    EXECT      POR FIN TERMINAMOS
00740       31FE    HEXNU EQU    *          VERIFICA SI ES UN NUMERO HEX
00750 31FE 81 30    HEXNU CMP A  #'0        (C=0;NO NUMERO HEX)
00760 3200 2D 0C          BLT    PAC0       (C=1;SI NUMERO HEX)
00770 3202 81 46          CMP A  #'F
00780 3204 2E 08          BGT    PAC0
00790 3206 81 3A          CMP A  #':
00800 3208 2D 06          BLT    PAC1
00810 320A 81 40          CMP A  #'@
00820 320C 2E 02          BGT    PAC1
00830 320E 0C       PAC0  CLC
00840 320F 39             RTS
00850 3210 0D       PAC1  SEC
00860 3211 39             RTS
00870 3212 36       ASHEX PSH A              (ASCII A HEX
00880 3213 A6 00          LDA A  0,X         ENTRADA EN ((X)) ASCII)
00890 3215 BD 321A        JSR    ASHEA
00900 3218 32             PUL A
00910 3219 39             RTS
00920 321A 36       ASHEA PSH A              (ASCII A HEX
00930 321B 8D E1          BSR    HEXNU       ENTRADA EN ACCA)
00940 321D 24 09          BCC    RTS3        SALIDA EN ACCB -LSN-)
00950 321F 80 30          SUB A  #$30
00960 3221 81 0A          CMP A  #$0A
00970 3223 2B 02          BMI    RTS2
00980 3225 80 07          SUB A  #07
00990 3227 16       RTS2  TAB
01000 3228 32       RTS3  PUL A
01010 3229 39             RTS
01020       322A    PRXHE EQU    *          IMP EN HEX EL BYTE EN ((X))
01030 322A 36       PRXHE PSH A              EN ((X))
01040 322B 37             PSH B
01050 322C 8D 10          BSR    I2HEX
01060 322E BD 3052  TALIS JSR    PRINT
01070 3231 17             TBA
01080 3232 BD 3052        JSR    PRINT
```

```
01090 3237 39              RTS
01100 3238 36       PRAHE  PSH A            ESTA LO HACE EN (ACCA)
01110 3239 37              PSH B
01120 323A 8D 04           BSR    I2HEA
01130 323C 20 F0           BRA    TALIS
01140 323E A6 00    I2HEX  LDA A  0,X
01150 3240 36       I2HEA  PSH A
01160 3241 8D 09           BSR    NIBR
01170 3243 16              TAB
01180 3244 32              PUL A
01190 3245 8D 01           BSR    NIBL
01200 3247 39              RTS
01210      3248     CONVER EQU    *          SUBR CONVIERTE HEX A ASCII
01220 3248 44       NIBL   LSR A             4 BITS MAS SIGNIFICATIVOS
01230 3249 44              LSR A
01240 324A 44              LSR A
01250 324B 44              LSR A
01260 324C 84 0F    NIBR   AND A  #$0F       4 B MENOS SIGNIFICATIVOS
01270 324E 8B 30           ADD A  #$30
01280 3250 81 39           CMP A  #$39
01290 3252 23 02           BLS    RTS4
01300 3254 8B 07           ADD A  #$07
01310 3256 39       RTS4   RTS
01320      3257     ARAM   EQU    *          ADQUIERE DIGITOS DEL RECORD
01330 3257 FE 3FE0 ARAM   LDX    RECL
01340 325A BD 3292         JSR    NODI       DAME EL NUMERO DE DIGITOS
01350 325D B6 3FEA         LDA A  COUN       ESTAN EN COUN
01360 3260 26 02           BNE    FOUR
01370 3262 0C              CLC
01380 3263 39              RTS
01390 3264 81 04    FOUR   CMP A  #04
01400 3266 23 03           BLS    HOMME
01410 3268 7E 303F         JMP    ABORT      SE EXCEDIO
01420 326B FF 3FE7 HOMME   STX    RXCEL
01430 326E 09       FLOZ   DEX
01440 326F BD 3212         JSR    ASHEX
01450 3272 E7 00           STA B  0,X
01460 3274 BC 3FE0         CPX    RECL       TERMINO?
01470 3277 26 F5           BNE    FLOZ       NO?, ENTONCES FLOZ
01480 3279 4F              CLR A
01490 327A 5F              CLR B
01500 327B EB 00    AIRE   ADD B  0,X
01510 327D 08              INX
01520 327E BC 3FE7         CPX    RXCEL
01530 3281 27 0A           BEQ    RAMON      FIN ALMAC. EN LOS ACUMULADORE

01540 3283 58              ASL B
01550 3284 49              ROL A
01560 3285 58              ASL B
01570 3286 49              ROL A
01580 3287 58              ASL B
01590 3288 49              ROL A
01600 3289 58              ASL B
01610 328A 49              ROL A
01620 328B 20 EE           BRA    AIRE
```

```
0163C 328D FF 3FE0 RAMON   STX     RECL      ACTUALIZA APUNTADO!
01640 3290 0D              SEC               FIJA "C" VALIDO
01650 3291 39              RTS
01660       3292     NODI  EQU     *         NUM. DE DIGITOS EN EL RECORD
01670 3292 36        NODI  PSH A
01680 3293 7F 3FEA         CLR     COUN
01690 3296 A6 00    SIHEX  LDA A   0,X
01700 3298 BD 31FE         JSR     HEXNU
01710 329B 24 0B           BCC     RTS5
01720 329D 08              INX
01730 329E 7C 3FEA         INC     COUN
01740 32A1 27 02           BEQ     ESFF
01750 32A3 20 F1           BRA     SIHEX
01760 32A5 7A 3FEA ESFF    DEC     COUN      EXCEDE LA CAPACIDAD
01770 32A8 32        RTS5  PUL A
01780 32A9 39              RTS
01790 ⟹ 32AA       EXECT  EQU     *
01800 32AA 7D 3FE9 EXECT   TST     COUNT     HEX O ASCII
01810 32AD 26 29           BNE     DASC
01820 32AF BD 3319 DHEX    JSR     ADF       Y RESULTA QUE ES HEXA
01830 32B2 FE 3FDA KITEX   LDX     DIREC     CARGA EN (X) EL (DIREC)
01840 32B5 BD 3332         JSR     PRESP     IMP UN ESPACIO
01850 32B8 BD 322A         JSR     PRXHE
01860 32BB 08              INX
01870 32BC FF 3FDA         STX     DIREC     ACTUALIZA APUNTADOR
01880 32BF FE 3FD8         LDX     DIF
01890 32C2 09              DEX
01900 32C3 27 0D           BEQ     FINAL     TERMINAMOS ?
01910 32C5 FF 3FD8         STX     DIF       NO!
01920 32C8 7A 3FE9         DEC     COUNT
01930 32CB 26 E5           BNE     KITEX
01940 32CD BD 3048         JSR     PRTCL
01950 32D0 20 DD           BRA     DHEX      REGRESA
01960 32D2 BD 3048 FINAL   JSR     PRTCL
01970 32D5 7E 3025         JMP     FMON
01980 32D8 BD 3319 DASC    JSR     ADF       Y RESULTA QUE ES ASCII
01990 32DB 86 18           LDA A   #$18      6?,SON NI SE NOTAN
02000 32DD BB 3FE9         ADD A   COUNT     SUMALE OTROS 24
02010 32E0 B7 3FE9         STA A   COUNT     YA VAS
02020 32E3 FE 3FDA KITAS   LDX     DIREC
02030 32E6 BD 333A         JSR     CAPXC
02040 32E9 BD 3052         JSR     PRINT
02050 32EC 08              INX
02060 32ED FF 3FDA         STX     DIREC     ACTUALIZA APUNTADOR
02070 32F0 FE 3FD8         LDX     DIF
02080 32F3 09              DEX
02090 32F4 27 DC           BEQ     FINAL
02100 32F6 FF 3FD8         STX     DIF
02110 32F9 7A 3FE9         DEC     COUNT
02120 32FC 26 E5           BNE     KITAS
02130 32FE BD 3048         JSR     PRTCL
02140 3301 20 D5           BRA     DASC
02150       3303     CBAD  EQU     *         CUANTOS BYTES MOSTRAREMOS
02160 3303 B6 3FDB CBAD    LDA A   DIREC+1    ACUI SE LO DIREMOS
```

```
0217@ 32FB BD 3046          JSR     PRTCL
0218@ 32FE 20 D5            BRA     DASC
02190       3300    CBAD    EQU     *           CUANTOS BYTES MOSTRAREMOS
0220@ 3300 B6 3FDB CBAD     LDA A   DIREC+1     AQUI SE LO DIREMOS
0221@ 3303 84 0F            AND A   #$0F
0222@ 3305 C6 08            LDA B   #08
0223@ 3307 10               SBA
0224@ 3308 26 C5            BNE     DIF8
0225@ 330A F7 3FE9          STA B   COUNT       SON OCHO
0226@ 330D 20 06            BRA     PEL
02270 330F 2A 01    DIF8    BPL     POSIT
0228@ 3311 40               NEG A               DIFERENTE DE 8
0229@ 3312 B7 3FE9 POSIT    STA A   COUNT
0230@ 3315 39       PEL     RTS
02310       3316    ADF     EQU     *           IMPRIME LA DIRECCION
0232@ 3316 BD 3300 ADF      JSR     CBAD
0233@ 3319 CE 0000          LDX     #0000
0234@ 331C BC 3FD8          CPX     DIF
0235@ 331F 27 B1            BEQ     FINAL       TERMINAMOS ?
0236@ 3321 CE 3FDA          LDX     #DIREC      NO
0237@ 3324 BD 322A          JSR     PRXHE       IMPRIME (X)
0238@ 3327 08               INX
0239@ 3328 BD 322A          JSR     PRXHE       IMPRIME (X)
0240@ 332B BD 332F          JSR     PRESP
02410 332E 39               RTS
0242@ 332F 36       PRESP   PSH A               IMPRIME UN ESPACIO
0243@ 3330 86 20            LDA A   #$20
0244@ 3332 BD 3052          JSR     PRINT
0245@ 3335 32               PUL A
0246@ 3336 39               RTS
0247@ 3337 A6 00    CAPXC   LDA A   0,X         CAPTURALO DE ((X))
0248@ 3339 81 20            CMP A   #SPACE      ES ESPACIO?
0249@ 333B 2A 02            BPL     SICAR
0250@ 333D 86 20            LDA A   #SPACE      AHORA ES ESPACIO
02510 333F 39       SICAR   RTS
02520       3340    ULTIMA  EQU     *           LA ANTERIOR FUE LA ULTIMA
02530       3340    INSTR   EQU     *           INSTRUCCION DEL COMANDO DM
02543       ********    ********    ********    ********

02550                   END

TOTAL ERRORS   000
```

```
        // EJEMPLOS DE LA CORRIDA DE MONAMI //

      // LO QUE ESTA ENTRE DIAGONALES SE ADICIONO //

// DE RUT1 //


*AB,ESCRIBE EL ARGUMENTO Y REGRESA A CONTROL          // CORRECTO //
ESCRIBE EL ARGUMENTO Y REGRESA A CONTROL
*AB ESCRIBE EL ARGUMENTO                          // FALTA COMA //
NSG


// AHORA DE RUT3 //     // DESPLIEGUE DE MEMORIA //


*DM,,/                                  // TODO POR DEFACTO //
0000  3F
*DM,10,9,23                  /// ERRORSOTE ///
NSG
*DM,10,9/H                   // A PARTIR DE 0010 NUEVE BYTES EN HEX //
0010  4F 7E ED 4D 7E EC 35 7E
0018  EA
*DM,10,9                     // IGUAL PERO LA H POR DEFACTO //
0010  4F 7E ED 4D 7E EC 35 7E
0018  EA
*DM,,C/H      // 12 BYTE A PARTIR DE LA DIRECCION 0000 EN HEX //
0000  3F 14 39 3F 11 39 04 14
0008  7E ED 26 7E
*DM,,C/A      /// IGUAL PERO EN ASCII ///
0000 7979"M4"
*DM,1000,20          // AHORA A PARTIR DE 1000 20)HEX //
1000  02 16 03 02 C7 12 40 07
1008  03 46 27 62 F3 32 76 02
1010  0B 0E 63 0F 33 92 12 0E
1018  42 43 2A 4A 12 B6 D1 56
*DM,,/M              // M NO ES VALIDO //
NSG
*
```

/// EJEMPLOS MAS DEMOSTRATIVOS ///

```
DM,100,7A                    // A PARTIR DE LA DIRECCION 100 7A)HEX //
0100   45 53 54 4F 53 20 43 41
0108   52 41 43 54 45 52 45 53
0110   20 46 55 45 52 4F 4E 20
0118   41 4C 4D 41 43 45 4E 41
0120   44 4F 53 20 50 52 45 56
0128   49 41 4D 45 4E 54 45 2C
0130   20 53 4F 4C 4F 20 50 41
0138   52 41 20 44 45 4D 4F 53
0140   54 52 41 52 20 45 4C 20
0148   55 53 4F 20 44 45 4C 20
0150   43 4F 4D 41 4E 44 4F 20
0158   44 45 20 44 45 53 50 4C
0160   49 45 47 55 45 20 44 45
0168   20 4D 45 4D 4F 52 49 41
0170   20 3E 3E 44 4D 3C 3C 2E
0178   2E 2E
*DM,100,7A/A                 // IGUAL PERO AHORA EN ASCII //
0100 ESTOS CARACTERES FUERON ALMACENA
0120 DOS PREVIAMENTE, SOLO PARA DEMOS
0140 TRAR EL USO DEL COMANDO DE DESPL
0160 IEGUE DE MEMORIA >>DM<<...
*
```

//FIN DEL EJEMPLO//

/ FIN DE LOS EJEMPLOS /

CONCLUSIONES

Es importante mencionar que al inicio de este trabajo, el conocimiento de los dispositivos estudiados era casi nulo. Las deficiencias fueron subsanadas, mediante un gran esfuerzo, en el transcurso del desarrollo de esta tesis. De ésta, se pueden derivar un conjunto de comentarios y recomendaciones; de las cuales se enumeran las que al parecer son las más importantes.

El participar en una tarea de esta naturaleza, es satisfactorio, y más cuando los resultados nos han proporcionado los conocimientos fundamentales para poder introducirnos de hecho, en el campo de la investigación o estudio de aplicaciones para estos elementos.

Cabe decir, que el sistema empleado es uno de tantos que se encuentran disponibles en el mercado, y aunque se dijo que se trata de un sistema de desarrollo, éste está muy limitado. Sus limitaciones consisten, básicamente en los recursos de memoria y soporte en SOFTWARE. Sin embargo, para las aplicaciones tan reducidas, como las mencionadas en el capítulo VI, los elementos disponibles en AMI son más que suficientes. Ya casi al finalizar se contó con nuevos recursos que fueron empleados en resultados positivos, éstos consisten en lo siguiente:

- Un Módulo de memoria con 16 KBYTES de RAM (montados en AMI).
- Un sistema auxiliar, SWTPC 6800, con suficiente soporte en SOFTWARE (sistema operativo en disco).

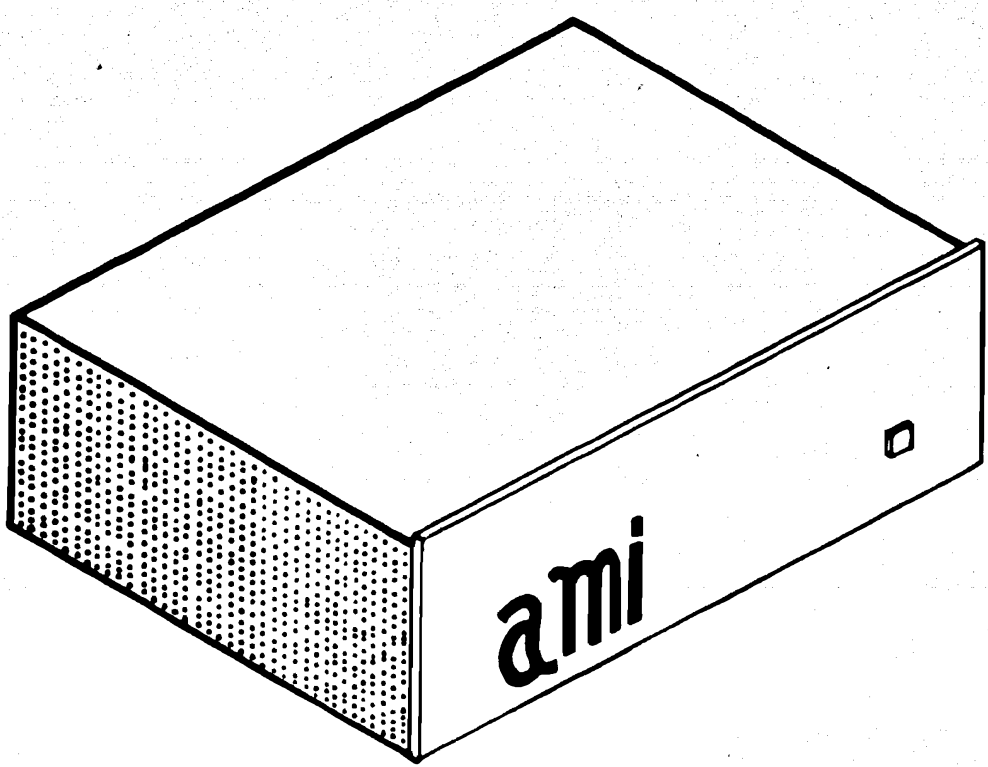
Con estos elementos adicionales, se advierte una eficiencia mayor para el sistema. Es más, el programa de aplicación tres, capítulo VI, se desarrolló con el ensamblador propio de SWTPC, similar al de MOTOROLA (ver capítulo IV).

El funcionamiento del sistema AMI es satisfactorio, los problemas que resultaron de su implementación fueron menores y solucionados sobre la marcha.

La flexibilidad y facilidad de uso de la familia S 6800 - (MC 6800), son los argumentos más importantes para afirmar que es una de las familias con mayor futuro. Y si sumamos a ellolas innovaciones que recientemente ha tenido (MC 6809, MC68000) se refuerza éste concepto. Pero, debido a su evolución, nos - encontramos con que los dispositivos primeros (S 6800, S 6810, S 6820, etc.) tienden a desaparecer, concretamente el S 6810 - que ha sido descontinuado. De aquí que el módulo de memoria - que se adicionó a AMI, no esté estructurado a base del S 6810.

Se recomienda que cualquier intento para introducirse a - este tipo de dispositivos, se haga con un sistema que contenga los elementos indispensables y que, además, éste se encuentre- "listo" para trabajar, es decir, solo se deberá adquirir un - microcomputador desensamblado, o tratar de formar un sistema - cuando ya se tenga la experiencia suficiente para configurarlo.

A partir de la labor desarrollada, se piensa haber cumplido con los objetivos planteados al inicio de esta tesis.


ami

APENDICE A

GLOSARIO DE TERMINOS

ACCESS MODE (MODO DE ACCESO)

Una técnica que se usa para obtener un registro específico de un archivo específico, o para colocar un registro específico en un archivo específico.

ACCESS TIME (TIEMPO DE ACCESO).

El intérvalo que media entre el instante en que un procesador o unidad de control pide una transferencia de datos A/O desde un dispositivo de almacenamiento y el instante en que la operación queda terminada. Así, por ejemplo, tiempo de acceso es la suma del tiempo de espera y del tiempo de transferencia.

ACCUMULATOR (ACUMULADOR).

Un registro que guarda un operando, pudiendo realizar diversas operaciones aritméticas y/o lógicas que incluyen aquel operando y (cuando corresponde) otro operando; generalmente, el resultado de la operación queda en el acumulador, sustituyendo al operando original.

ADDRESS (DIRECCION).

Un nombre, número u otra referencia que indica una ubicación particular en un almacenamiento o alguna otra fuente o destino de datos.

ALGORITHM (ALGORITMO).

Un juego de reglas, bien definidas para la solución de un problema en un número finito de pasos.

ANALOG (ANALOGICO).

Se refiere a los datos presentados bajo forma de cantidades físicas continuamente variables.

ASCII

Un código de siete BITS adoptado como estándar en E.E.U.U. con el fin de facilitar el intercambio de datos entre diversos tipos de equipo de procesamiento y comunicación de datos.

ASSEMBLE (ENSAMBLAR).

Preparar un lenguaje máquina basándose en un programa escrito en codificación simbólica.

ASSEMBLER (ENSAMBLADOR).

Un programa de computador que ensambla programas escritos en codificación simbólica paraproducir programas de lenguaje - de máquina.

BAUD

Una unidad de velocidad de señalización que es igual al - número de condiciones discretas o cantidad de señales por se-- gundo (un BIT por segundo).

BCD (DECIMAL CODIFICADO EN BINARIO).

Se refiere a un método de representación de cada uno de-- los dígitos decimales que van de cero a nueve. por un grupo distinto de dígitos binarios.

BINARY (BINARIO).

Se refiere al sistema de numeración con raiz dos, o a una característica o propiedad de implica una elección o condición en la que existen dos posibilidades.

BIT

1) Un dígito binario; un dígito (cero o uno) en la representación de un número en notación binaria. 2) Una abreviatura de BINARY DIGIT (dígito binario). 3) Un solo pulso de un - grupo de ellos. 4) Una unidad de capacidad de información de- un mecanismo de almacenamiento.

BREAK POINT.

Un punto concreto de un programa en el que este puede ser interrumpido por intervención manual o por una rutina del monitor.

BUFFER.

Un dispositivo o sección de memoria que se emplea para retener información temporalmente.

BUS

El camino principal que se usa para transmitir señales -

desde una o más fuentes a uno o más destinos.

BYTE.

Un grupo de BITS adyacentes con los que se trabaja como una unidad, generalmente es más corto que una palabra (en la mayoría de las máquinas, se ha tomado por convención un BYTE igual a ocho BITS).

CARRY (ACARREO).

Una señal que sale cuando la suma o producto de dos o más dígitos, en una posición de dígito, es igual o excede la raiz del sistema de numeración que se usa.

CENTRAL PROCESSOR (PROCESADOR CENTRAL).

La unidad que en un sistema computador incluye los circuitos que controlan la interpretación y ejecución de instrucciones. Es sinónimo de CPU (CENTRAL PROCESSING UNIT), MPU (MICRO PROCESSING UNIT) y MAIN FRAME (ESTRUCTURA PRINCIPAL).

CLEAR (LIMPIAR).

Limpiar los datos de una ubicación de almacenamiento o de un dispositivo, poniendo todas las celdas de almacenamiento implicadas en un estado dictado de antemano, generalmente a un estado que denote nivel cero.

CODE (CODIGO).

Una serie de reglas sin ambigüedad que especifica la forma exacta en que tienen que ser representados los datos, por ejemplo ASCII, BCD, etc.

COMMENT (COMENTARIO).

Una explicación o identificación, para uso humano, de un paso dentro de una rutina; el comentario no tiene efecto sobre las operaciones del computador que ejecuta la rutina.

CONDITIONAL TRANSFER (TRANSFERENCIA CONDICIONAL).

Una instrucción que puede o no ser causa de un salto dependiendo del resultado de alguna operación.

DATA (DATO).

Una representación cualquiera de un hecho o una idea, en forma que pueda ser comunicada o manipulada mediante algún proceso.

DIGIT (DIGITO).

Un carácter numérico simple que se usa para representar - un entero.

DIGITAL.

Se refiere a los datos representados en forma de dígitos.

DISABLE (INHABILITADO).

Poner un dispositivo en una condición en la que es inca-- paz de responder a las señales de su unidad de control.

ENABLE (HABILITADO).

Poner un dispositivo en una condición que pueda responder a señales procedentes de su unidad de control.

EXECUTE (EJECUCION).

Llevar a cabo una instrucción o una operación o procesar- un programa.

FETCH (OBTENCION).

Obtener datos de una ubicación de almacenamiento.

FLAG (BANDERA).

Un carácter o símbolo que señala el acaecimiento de algu- na condición, como puede ser el final de una palabra o archivo.

HARDWARE.

El equipo físico, como son los dispositivos mecánicos, - magnéticos, eléctricos y electrónicos.

HIGH-ORDER (MAS SIGNIFICATIVO).

Se refiere al dígito o dígitos de un número que tiene el- mayor peso o significado.

INDEX REGISTER (REGISTRO INDICE).

Un registro cuyo contenido se puede sumar o restar de una dirección, antes o durante la ejecución de una instrucción.

INTERFACE (INTERFASE).

Un límite compartido; por ejemplo, un límite entre dos sis-

temas, o entre un computador y uno de sus dispositivos periféricos.

INTERRUPT (INTERRUPCION).

Proceso que permite, a petición del exterior, detener provisionalmente el programa en curso para dejar pasar prioritariamente un programa asociado a la petición, llamado programa o rutina de interrupción o de servicio; designa igualmente a la señal exterior que desencadena el proceso de interrupción.

JUMP (SALTO O TRANSFERENCIA).

Una desviación de la secuencia normal de ejecución de instrucciones en un computador.

LABEL (ETIQUETA).

Un nombre que va prendido o escrito al costado de la entidad que identifica.

LOW ORDER (MENOS SIGNIFICATIVO).

Se refiere al dígito o dígitos de un número que tienen el menor peso o significado.

MNEMONIC (MNEMONICO).

Se refiere a la técnica usada para ayudar a la memoria humana en el reconocimiento de entidades no simbólicas (dígitos o números).

MONITOR ROUTINE (MONITOR).

Una rutina concebida para indicar el progreso del trabajo en un sistema computador. Lo mismo que ejecutivo (ver cap. VI C.1).

MULTIPROCESSING (MULTIPROCESAMIENTO).

La ejecución simultánea de dos o más secuencias de instrucciones en un solo sistema computador. Esto se puede realizar mediante el uso de dos o más procesadores.

OBJECT LANGUAGE (LENGUAJE OBJETO).

Un lenguaje que es el producto de un proceso de traducción.

OBJECT PROGRAM (PROGRAMA OBJETO).

Un programa expresado en lenguaje objeto (por ejemplo un lenguaje máquina que se puede ejecutar directamente por un computador determinado).

OPERAND (OPERANDO).

Una unidad de datos con la que se realiza una operación.

OPERATING SYSTEM (SISTEMA OPERATIVO).

Una colección organizada de rutinas y procedimientos para el funcionamiento de un computador. Estas rutinas y procedimientos realizarán normalmente algunas o todas las funciones siguientes.

1) Señalamiento de tiempos, carga, iniciación, y supervisión en la ejecución de programas.

2) La distribución de almacenamiento, unidades de entrada salida (I/O) y otras conveniencias del sistema computador.

3) La iniciación y control de las operaciones de entrada/salida.

4) Tratamiento de errores y nueva puesta en marcha.

5) La coordinación de comunicaciones entre el operador humano y el sistema de computador.

6) Llevar un registro estadístico de las operaciones del computador.

7) El control de las operaciones en la modalidad de multiprogramación, multiproceso o tiempo compartido.

OPERATION CODE (CODIGO DE OPERACION).

Un código que se usa para representar las operaciones específicas de un computador.

PSEUDO INSTRUCTION (SEUDO INSTRUCCION)

Una instrucción que tiene la misma forma general que una instrucción máquina, pero que no es directamente ejecutable por un computador. Las seudo instrucciones se usan corrientemente en lenguajes orientados a la máquina, para controlar el funcionamiento de un traductor. Es sinónimo de Directiva.

RANDOM ACCESS (ACCESO ALEATORIO).

Se refiere a un dispositivo de almacenamiento en el que - el tiempo de acceso no está afectado de manera importante por- la ubicación de los datos a los que hay que acceder.

READ (LEER).

Obtener datos que están en un almacén o soporte.

RECORD.

Una colección de artículos de datos que guardan relación- entre sí.

REGISTER (REGISTRO).

Un dispositivo capaz de almacenar una cantidad específica de datos, como por ejemplo, una palabra, destinada generalmente para alguna finalidad especial. Entre los registros incluidos en muchos computadores hay uno o varios acumuladores, registros índice, registros de instrucción, contador de programa.

RELATIVE ADDRESS.

Una dirección (contenida generalmente en una instrucción) que se combina con una dirección base para formar la dirección absoluta de una ubicación particular de almacenamiento.

RESET.

Volver a poner un contador, indicador, conmutador o ubicación de memoria en una situación inicial prescrita.

ROUTINE (RUTINA).

Una serie de instrucciones dispuestas en la secuencia idónea para hacer que un computador realice un proceso particular.

SERIAL (SERIE).

Tratar los elementos de una palabra o mensaje (por ejemplo, los BITS o caracteres) uno tras otro, en el mismo dispositivo.

SHIFT (CORRIMIENTO).

Mover todos los BITS, dígitos o caracteres de una palabra un número determinado de posiciones hacia la izquierda o hacia la derecha.

SKIP

Ignorar una o más instrucciones en una secuencia de ins-- trucciones.

SNAPSHOT.

Una copia dinámica de los contenidos de ubicaciones específicas de almacenamiento en puntos o momentos específicos, durante el procesamiento de un programa.

SOFTWARE.

La colección de programas y rutinas que acompañan a un - computador y las cuales facilitan la programación y funcionamiento de un computador.

SOURCE LANGUAGE (LENGUAJE FUENTE).

Un lenguaje que es una entrada para un proceso de traducción.

STORE (ALMACENAR).

Insertar o retener datos en un dispositivo de almacena--- miento.

SUMMATION CHECK - CHECK SUM.

Una confrontación en la que se forma un grupo de dígitos, generalmente sin considerar el acarreo, comparándose el resultado con un valor previamente calculado que se llama -suma de- confrontación-.

TERMINAL.

Un punto o dispositivo en un sistema o red de comunicaciones en el cual los datos pueden entrar o salir.

UNCONDITIONAL TRANSFER (TRANSFERENCIA INCONDICIONAL).

Una instrucción que siempre produce un salto, es decir, - una desviación de la secuencia normal de ejcución de instrucciones.

WORD (PALABRA).

Un grupo de bits o caracteres que son tratados como una - unidad y que pueden ser almacenados en una celda de almacena-- miento.

WRITE (ESCRIBIR).

Registrar datos en un almacén o soporte.

APENDICE B

CARACTERISTICAS E INFORMACION ADICIONAL DE LOS ELEMENTOS EN AMI

— DYNAMIC CHARACTERISTICS (V_{CC} = 5.0 V ± 5%, V_{SS} = 0, T_A = 0 to 70°C unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Address Setup Time	t_{AS}	190	–	–	ns
Allowable Data Delay Time (Read)	t_{DDR}	–	200	380	ns
Memory Read Access Time	t_{acc}	–	–	540	ns
Data Setup Time (Read)	t_{DSR}	100	–	–	ns
Available Data Setup Time (Write)	t_{DSW}	225	–	–	ns
Input Data Hold Time	t_H	10	–	–	ns
Output Data Hold Time	t_H	10	30	–	ns
Address Hold Time	t_{AH}	60	75	–	ns
Address Delay	t_{AD}	–	220	380	ns
Data Delay (Write)	t_{DDW}	–	165	225	ns
Interrupt Request Release Time	t_{IR}	–	0.7	1.2	µs
Frequency of Operation	f	0.1	–	1.0	MHz
Clock Timing (ϕ1 and ϕ2)					
Cycle Time	t_{cyc}	1.0	–	10	µs
Clock Pulse Width (Measured at V_{CC} - 0.3 V) ϕ1	$PW_{\phi H}$	430	–	4500	ns
ϕ2		460	–	4500	ns
Clock Up Time	t_{ut}	940	–	–	ns
Rise and Fall Times ϕ1, ϕ2 (Measured between V_{SS} + 0.3 V and V_{CC} - 0.3 V)	$t_{\phi r}$, $t_{\phi f}$	50	–	90	ns
Delay Time or Clock Separation (Measured at V_{SS} + 0.5 V)	t_d	0	–	9100	ns
Overshoot Duration	t_{OS}	0	–	40	ns
E Enable Pulse Width	PW_E	0.45	–	25	µs
E Enable Rise and Fall Time	t_{Er}, t_{Ef}	–	–	25	ns
Processor Controls					
Processor Control Setup Time (Figure 12, 13, 14, 15)	t_{PCS}	200	–	–	ns
Processor Control Rise and Fall Time (Figures 12, 13, 14, 15)	t_{PCr}, t_{PCf}	–	–	100	ns
Bus Available Delay (Figure 15)	t_{BA}	–	–	300	ns
Three State Enable (Figure 15)	t_{TSE}	–	–	40	ns
Three State Delay (Figure 15)	t_{TSD}	–	–	700	ns
Data Bus Enable Down Time During ϕ1 Up Time (Figure 8)	$t_{\overline{DBE}}$	150	–	–	ns
Data Bus Enable Delay (Figure 8)	t_{DBED}	300	–	–	ns
Data Bus Enable Rise and Fall Times (Figure 8)	t_{DBEr}, t_{DBEf}	–	–	25	ns

Note: Dynamic properties of most of the M6800 peripheral and memory parts exceed the requirements specified above, allowing for system flexibility.

CARACTERÍSTICAS DINÁMICAS DEL CPU.

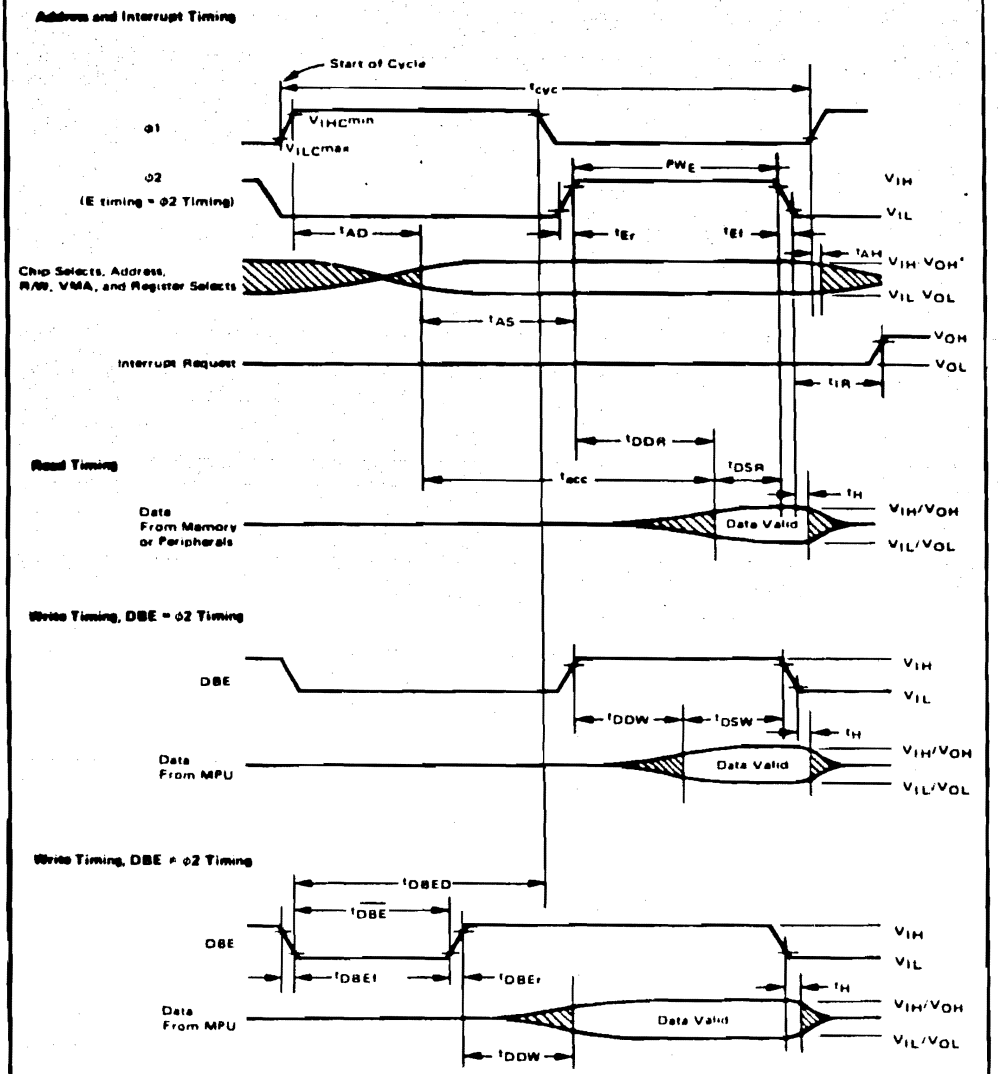
– **STATIC CHARACTERISTICS** (V_{CC} = 5.0 V ± 5%, V_{SS} = 0, T_A = 0 to 70°C unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage – All Inputs except MPU Φ1 and Φ2	V_{IH}	V_{SS} + 2.0	–	V_{CC}	Vdc
MPU Φ1 and Φ2	V_{IHC}	V_{CC} – 0.3	–	V_{CC} + 0.1	
Input Low Voltage – All Inputs except MPU Φ1 and Φ2	V_{IL}	V_{SS} – 0.3	–	V_{SS} + 0.8	Vdc
MPU Φ1 and Φ2	V_{ILC}	V_{SS} – 0.1		V_{SS} + 0.3	
Input Leakage Current (V_{in} = 0 to 5.25 Vdc)	I_{in}				μAdc
(V_{CC} = 5.25 Vdc) All Inputs except MPU Φ1 and Φ2		–	1.0	2.5	
(V_{CC} = 0) MPU Φ1 and Φ2		–	–	100	
Three State (Off State) Input Current	I_{TSI}				μAdc
(V_{in} = 0.4 to 2.4 V, V_{CC} = 5.25 Vdc) D0-D7		–	2.0	10	
A0-A15, R/W		–	–	100	
Output High Voltage (Load A of Figure 8, V_{CC} = 4.75 Vdc)	V_{OH}	V_{SS} + 2.4			Vdc
Output Low Voltage (Load A of Figure 8, V_{CC} = 4.75 Vdc)	V_{OL}	–		V_{SS} + 0.4	Vdc
Output Leakage Current, (IRQ of Peripherals (V_{in} = 2.4 Vdc)	I_{LOH}	–	1.0	10	μAdc
Capacitance* (V_{in} = 0, T_A = 25°C, f = 1.0 MHz) MPU Φ1 and Φ2	C	80	120	160	pF
MPU TSC		–	–	15	
MPU DBE		–	7.0	10	
MPU Logic Inputs		–	6.5	8.5	
All Other Inputs		–	6.0	7.5	
D0-D7, A0-A15, R/W, VMA		–	10	12.5	
IRQ Output		–	3.0	7.0	
All Other Outputs		–	6.0	10	
Φ1 and Φ2 Overshoot/Undershoot – Input High Level	V_{OS}	V_{CC} – 0.5		V_{CC} + 0.5	Vdc
– Input Low Level		V_{SS} – 0.5		V_{SS} + 0.5	
Clock Overlap Voltage (Figure 7)	V_{OV}			0.5	Vdc

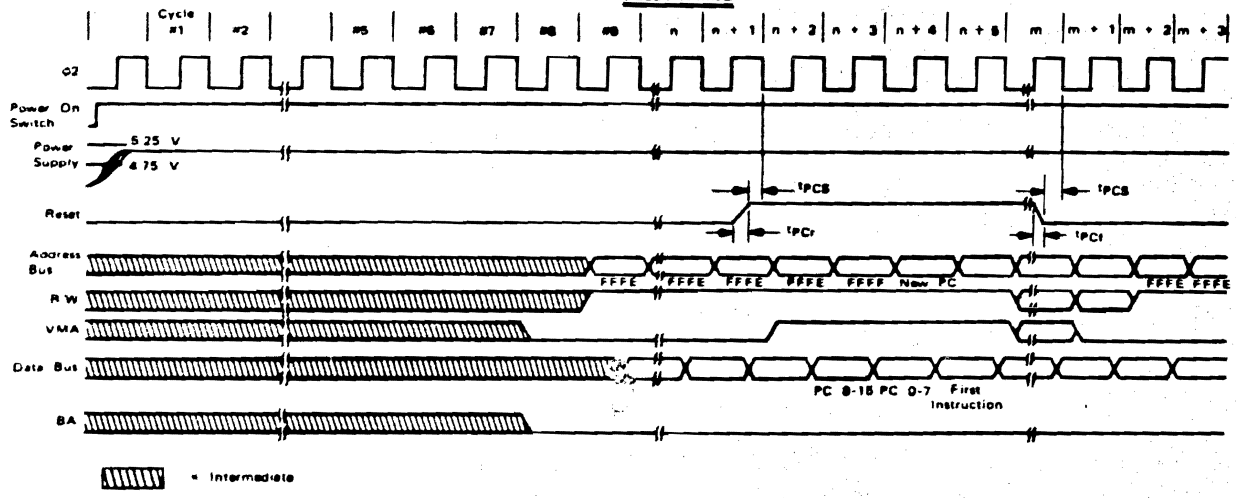
*Capacitances are periodically sampled rather than 100% tested.

CARACTERÍSTICAS ESTÁTICAS DEL C.P.U.

BUS INTERFACE TIMING

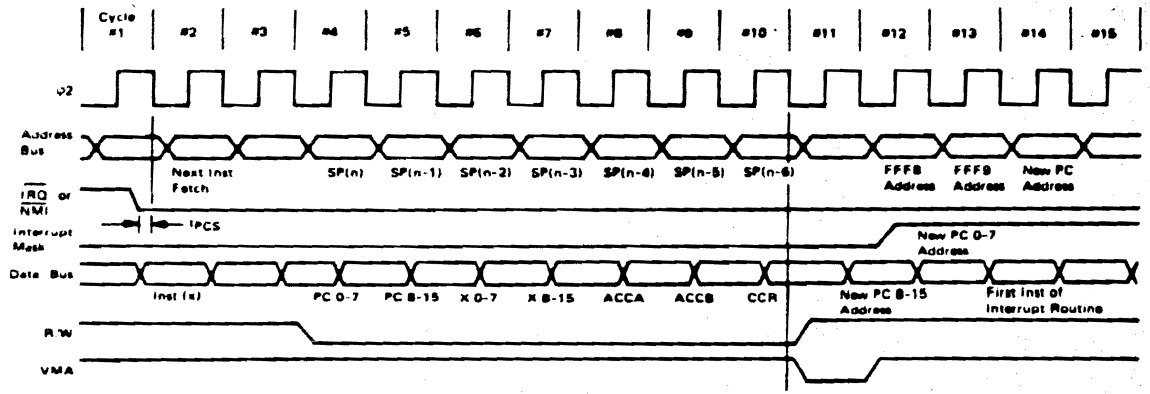


RESET TIMING



= Intermediate

INTERRUPT TIMING



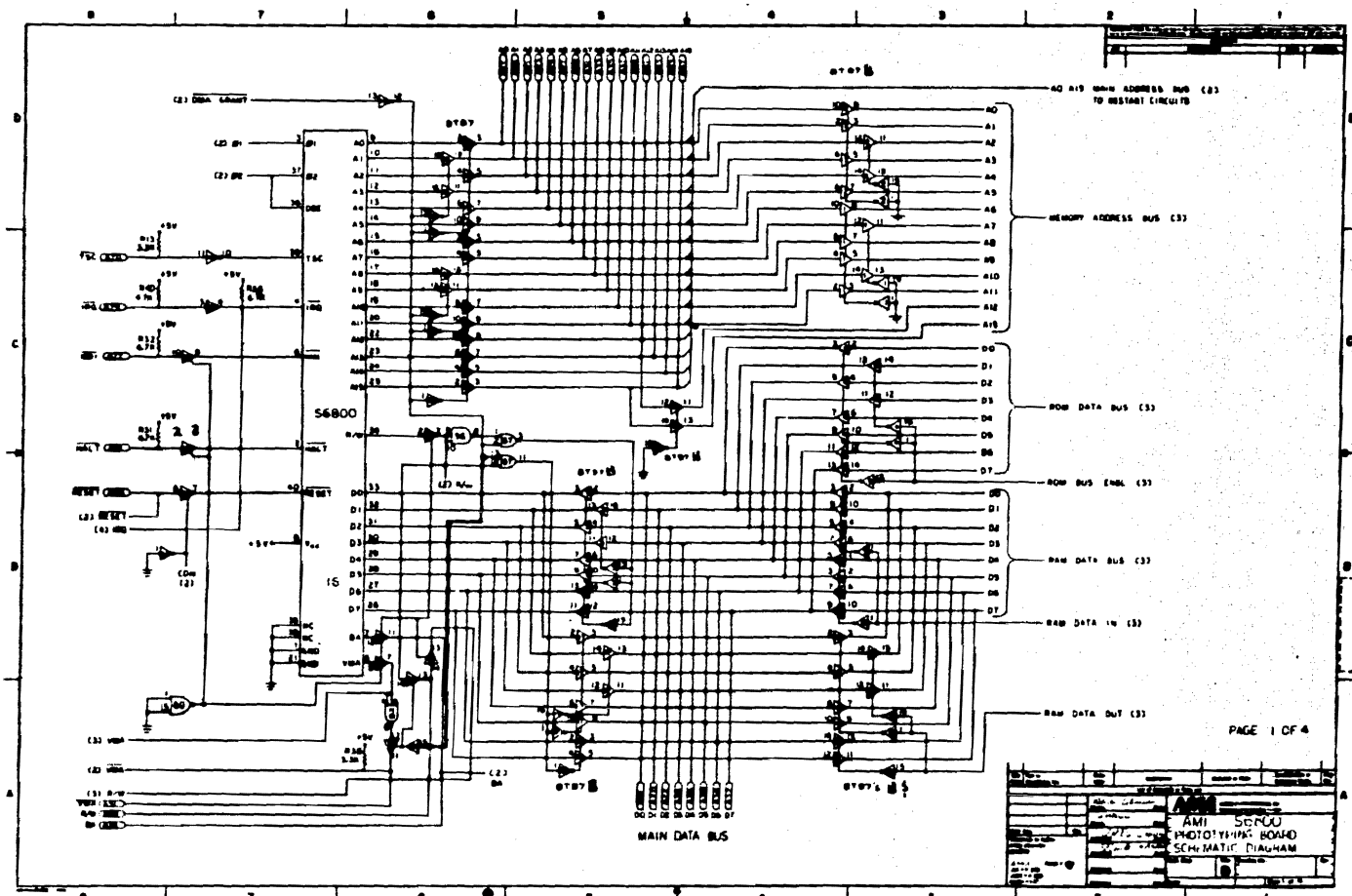


DIAGRAMA DE AMI (1)

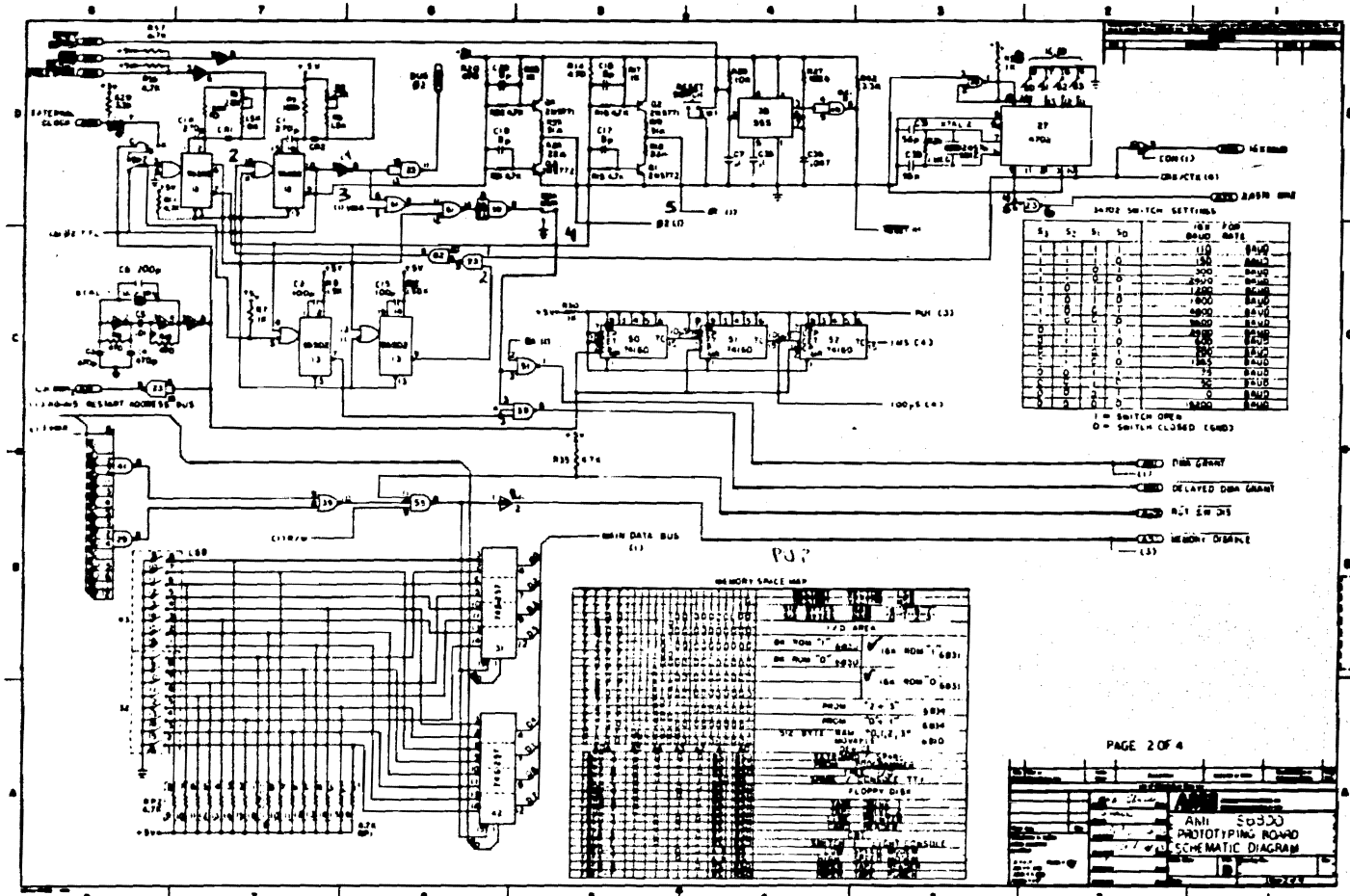


DIAGRAMA DE ANI (2)

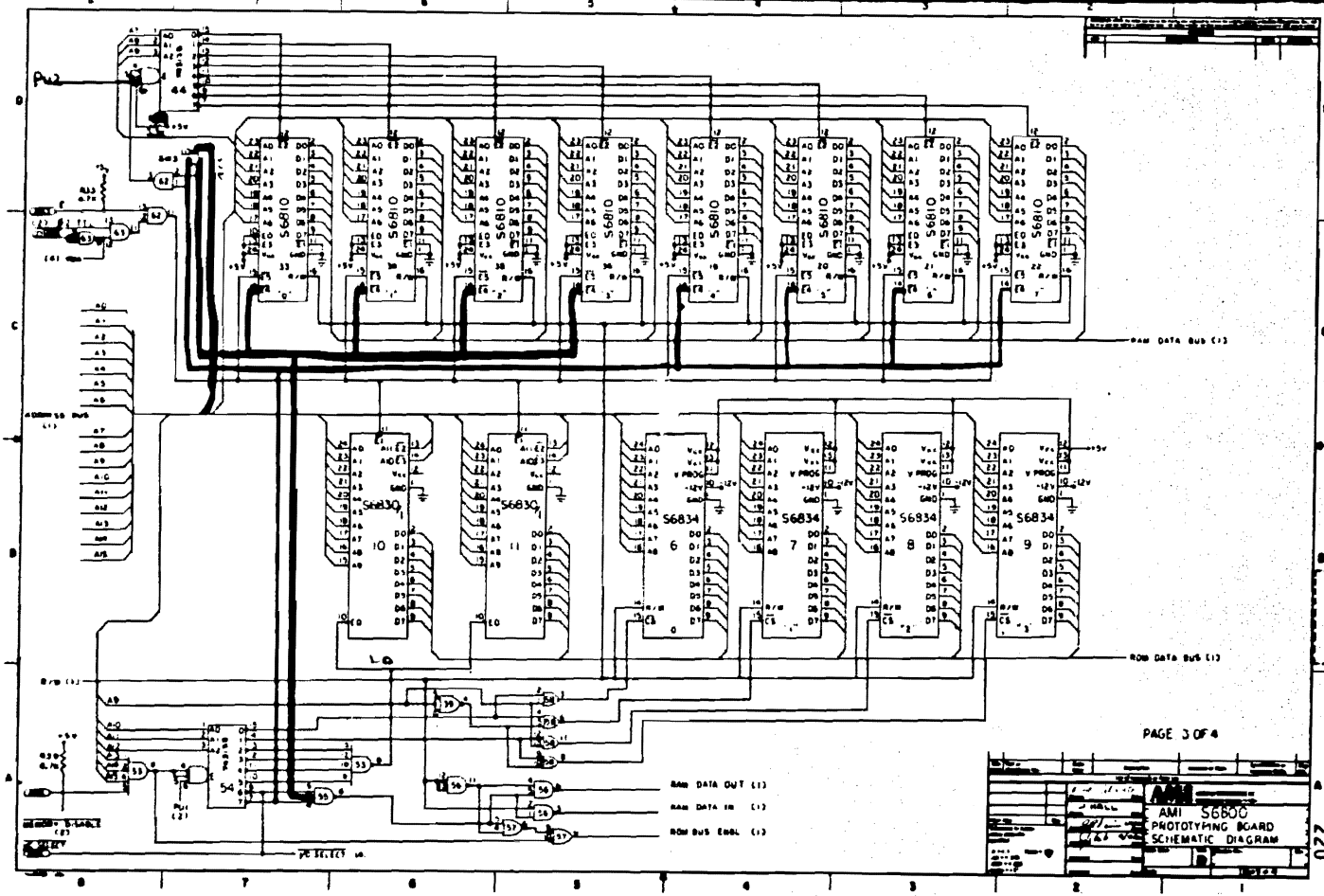


DIAGRAMA DE AMI (3)

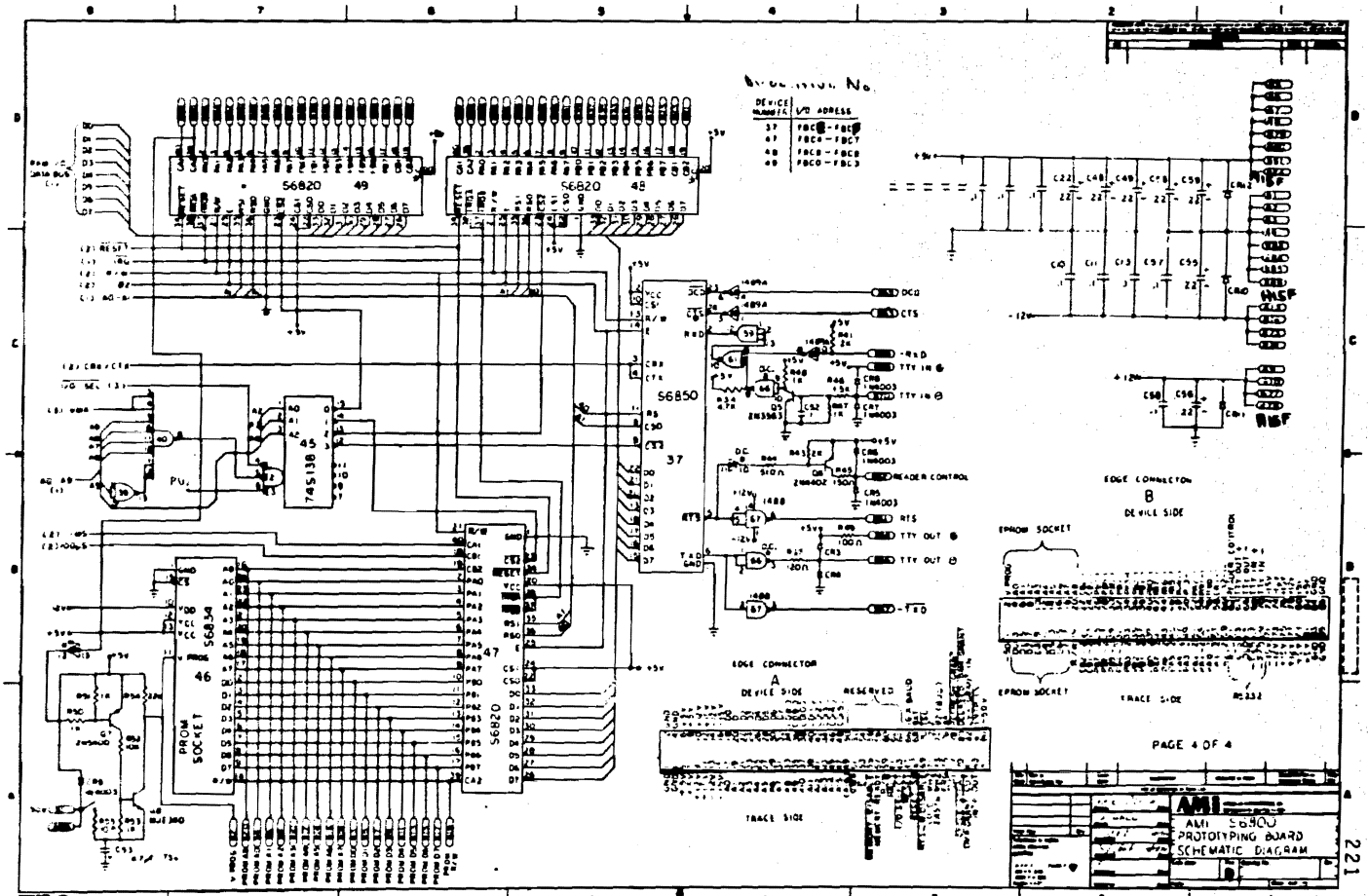
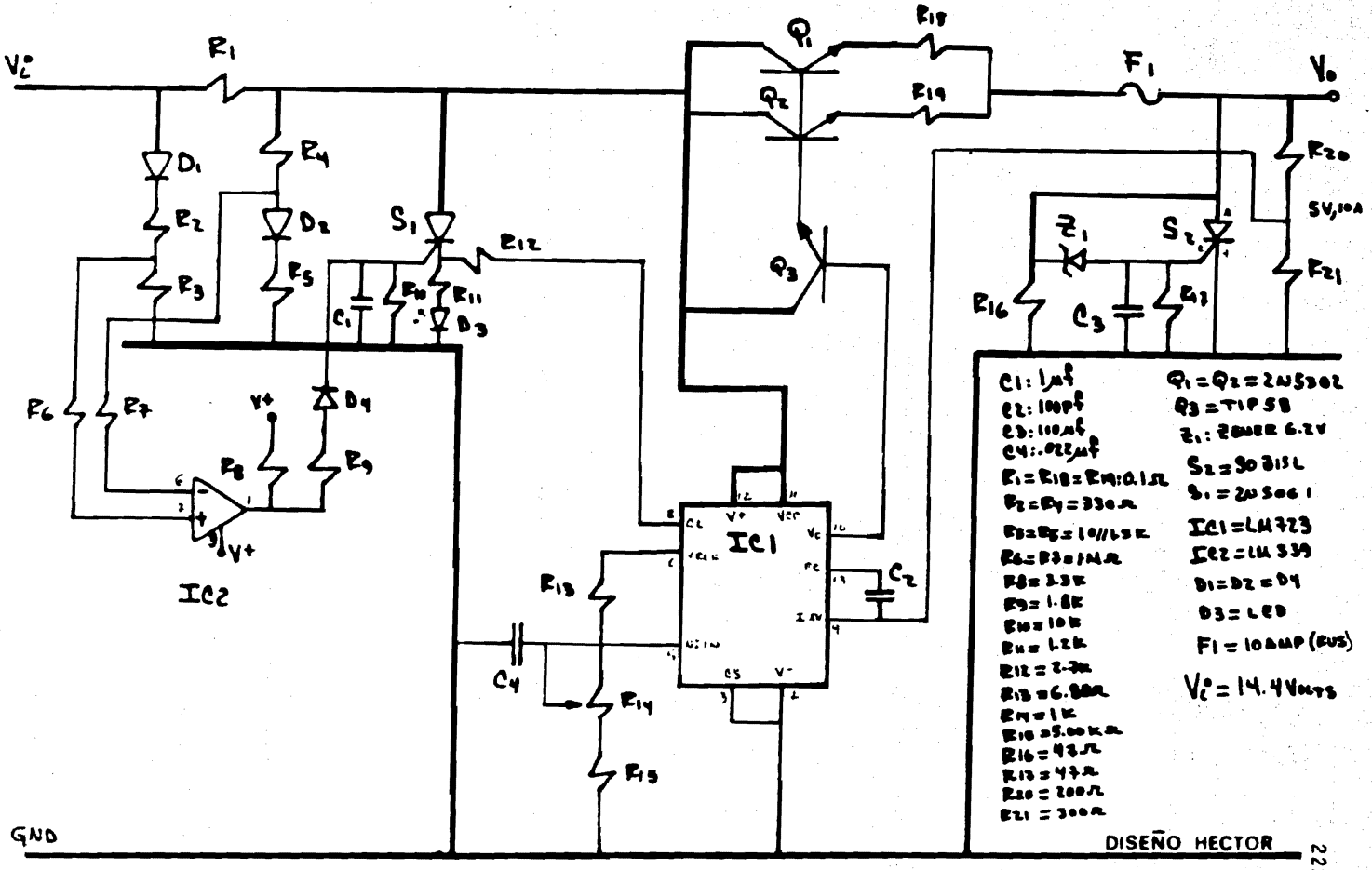


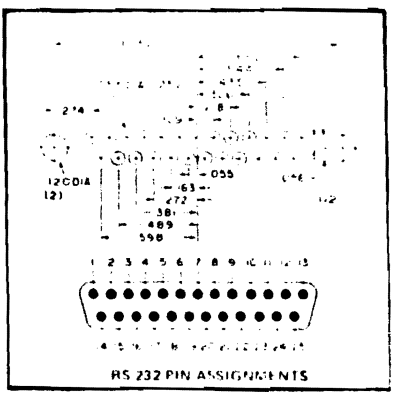
DIAGRAMA DE AMI (4)



FUENTE REGULADA 5V@10A

Signal Name	Mnemonic	Pin	SEÑALES MAS UTILIZADAS RS-232
Protective Ground	GND	1	A connection to the terminals metal chassis.
Transmitted Data	TDATA	2	Outgoing data path from the terminals point of view.
Received Data	RDATA	3	Incoming data path from the terminals point of view.
Request to Send	RTS	4	Activated by the terminal to tell the modem to prepare to receive data from the terminal.
Clear to Send	CTS	5	Activated by the modem to tell the terminal that it is ready to receive data from the terminal.
Data Set Ready	DSR	6	Activated by the modem to tell the terminal that the modem is operational.
Signal Ground	SGND	7	Return path for all other signals on the bus.
Received Line Signal Detector	RLSD	8	Activated by the modem to tell the terminal that the modem has made contact with the computer and can sense the carrier.
Data Terminal Ready	DTR	20	Activated by the terminal to tell the modem that the terminal is operational

Pin	Name	Function
1	FG	Frame Ground (not switched)
2	TD	Transmit Data
3	RD	Receive Data
4	TRS	Request To Send
5	CTS	Clear To Send
6	DSR	Data Set Ready
7	SG	Signal Ground
8	DCD	Data Carrier Detect
9		Positive DC Test Voltage
10		Negative DC Test Voltage
11		Unassigned
12	(S)DCD	Secondary Data Carrier Detect
13	(S)CTS	Secondary Clear To Send
14	(S)TD	Secondary Transmit Data
15	TC	Transmit Clock
16	(S)RD	Secondary Receive Data
17	RC	Receive Clock
18		Receiver Orbit Clock
19	(S)RTS	Secondary Request To Send
20	DTR	Data Terminal Ready
21	SQ	Signal Quality Detect
22	RI	Ring Indicator
23		Data Rate Select
24	ETC	External Transmit Clock
25		Busy



RS 232 PIN ASSIGNMENTS

APENDICE C

REPERTORIO DE INSTRUCCIONES DEL S6800 (MC6800).

continued

INSTRUCTION	MNEM	INH	IMM	DIR	INX	EXT	REL	OPERATION	H	I	N	Z	V	C
Jump	JMP				6E	7E		see special op						
Jump to subroutine	JSR				AD	BD		see special op						
Load accumulator	LDAA		86	96	A6	B6		M → A			A	A	R	
	LDAB		C6	D6	E6	F6		M → B			A	A	R	
Load stack pointer	LDS		8E	9E	AE	BE		$M \rightarrow SP_H, (M+1) \rightarrow SP_L$			9	A	R	
Load index	LDX		CE	DE	EE	FE		$M \rightarrow X_H, (M+1) \rightarrow X_L$			9	A	R	
Shift right logical	LSR				64	74					R	A	6	A
	LSRA	44									R	A	6	A
	LSRB	54									R	A	6	A
Complement (2's)	NEG				60	70		00 − M → M			A	A	1	2
	NEGA	40						00 − A → A			A	A	1	2
	NEGB	50						00 − B → B			A	A	1	2
No operation	NOP	01						PC + 1 → PC						
Inclusive or	ORAA		8A	9A	AA	BA		A + M → A			A	A	R	
	ORAB		CA	DA	EA	FA		B + M → B			A	A	R	
Push data	PSHA	36						$A \rightarrow M_{SP}, SP-1 \rightarrow SP$						
	PSHB	37						$B \rightarrow M_{SP}, SP-1 \rightarrow SP$						
Pull data	PULA	32						$SP+1 \rightarrow SP, M_{SP} \rightarrow A$						
	PULB	33						$SP+1 \rightarrow SP, M_{SP} \rightarrow B$						
Rotate left	ROL				69	79					A	A	6	A
	ROLA	49									A	A	6	A
	ROLB	59									A	A	6	A
Rotate right	ROR				66	76					A	A	6	A
	RORA	46									A	A	6	A
	RORB	56									A	A	6	A
Return from interrupt	RTI	3B						see special op		Note 10				
Return from subroutine	RTS	39						see special op						
Subtract accumulators	SBA	10						A − B → A			A	A	A	A
Subtract with carry	SBCA		B2	92	A2	B2		A − M − C → A			A	A	A	A
	SBCB		C2	D2	E2	F2		B − M − C → B			A	A	A	A
Set carry	SEC	OD						1 → C						S
Set interrupt mask	SEI	OF						1 → I		S				
Set overflow	SEV	OB						1 → V					S	
Store accumulator	STAA			97	A7	B7		A → M			A	A	R	
	STAB			D7	E7	F7		B → M			A	A	R	
Store stack pointer	STS			9F	AF	BF		$SP_H \rightarrow M, SP_L \rightarrow (M+1)$			9	A	R	
Store index	STX			DF	EF	FF		$X_H \rightarrow M, X_L \rightarrow (M+1)$			9	A	R	
Subtract	SUBA		80	90	A0	B0		A − M → A			A	A	A	A
	SUBB		C0	D0	E0	F0		B − M → B			A	A	A	A
Software interrupt	SWI	3F						see special op		S				

6800 INSTRUCTION SET continued

INSTRUCTION	MNEM	INH	IMM	DIR	INX	EXT	REL	OPERATION	H	I	N	Z	V	C
Transfer accumulators	TAB	16						A → B			A	A	R	
Transfer acc A to condition codes	TAP	06						A → CCR		See Note 12				
Transfer accumulators	TBA	17						B → A			A	A	R	
Transfer condition codes to acc A	TPA	07						CCR → A						
Test	TST				6D	7D		M – 00			A	A	R	R
	TSTA	4D						A – 00			A	A	R	R
	TSTB	5D						B – 00			A	A	R	R
Transfer stack pointer to index	TSX	30						SP + 1 → X						
Transfer index to stack pointer	TXS	35						X – 1 → SP						
Wait for interrupt	WAI	3E								11				

LEGEND:

MNEM	Instruction mnemonic
INH	Inherent addressing mode (including Accumulator addressing mode)
IMM	Immediate addressing mode
DIR	Direct addressing mode
INX	Indexed addressing mode
EXT	Extended addressing mode
REL	Relative addressing mode
+	Arithmetic plus, or Boolean OR
–	Arithmetic minus
o	Boolean AND
⊕	Boolean Exclusive OR
M̄	Complement of M
→	Transfer into
0	Bit = zero
00	Byte = zero
M_{SP}	Contents of memory location pointed to by Stack Pointer
M	Contents of addressed memory location

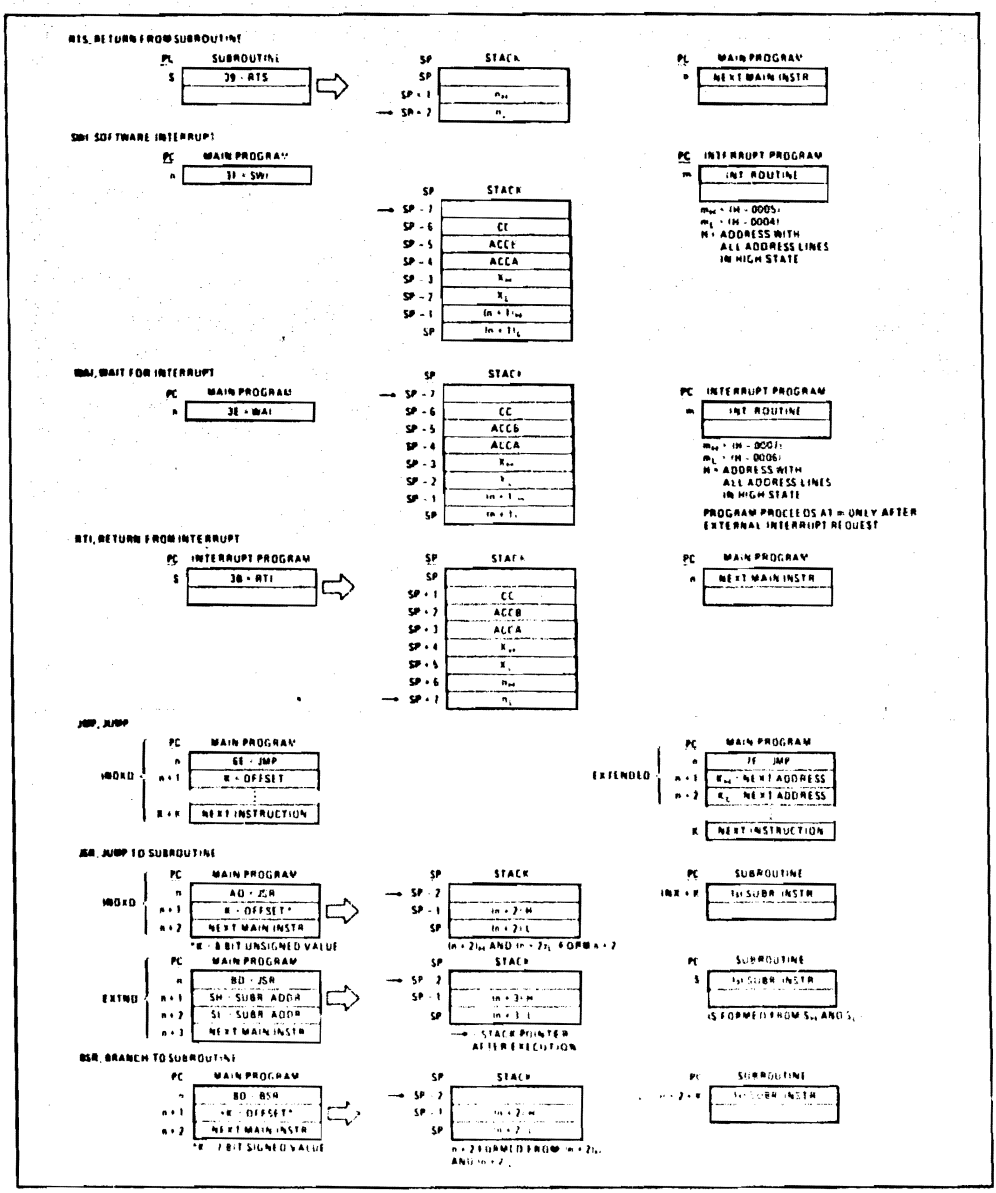
CONDITION CODE SYMBOLS:

H	Half-carry from bit 3
I	Interrupt mask
N	Negative (sign bit)
Z	Zero result
V	Overflow, 2's complement
C	Carry (from bit 7)
R	Reset always to 0
S	Set always to 1
A	Altered to 0 or 1, depending on result
(blank)	Not changed

CONDITION CODE REGISTER NOTES:

1	(V)	Set if result = 10000000; clear otherwise.
2	(C)	Set if result = 00000000; clear otherwise.
3	(C)	Set if most significant BCD digit > 9; else cleared.
4	(V)	Set if operand = 10000000 prior to execution, else cleared.
5	(V)	Set if operand = 01111111 prior to execution, else cleared.
6	(V)	Set equal to N ⊕ C after shift.
7	(N)	Set to sign of most significant byte.
8	(V)	Two's complement overflow from subtraction of most significant bytes.
9	(N)	Set to sign of result (bit 15).
10	(all)	Load condition code register from stack. (See Special Operations)
11	(I)	Set when interrupt occurs.
12	(all)	Set according to contents of accumulator A.

SPECIAL OPERATIONS



LSB \ MSB	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	END	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

TABLA DE CONVERSION ASCII - HEXADECIMAL.

BIBLIOGRAFIA

ABRAMS, P. y CORVINE, W.
Elementos de Proceso de Datos,
México, 1970.
C.E.C.S.A.

ALTMAN, L.
Microprocessors.
New York, 1975.
Mc. Graw Hill.

AUSTRICH, J.
Cross-Assembler HP2114-A-Motorola MC6800
con Ejemplos de Aplicación en la Física.
Tésis Inédita para Licenciatura,
Universidad Autónoma de México, 1978.

AWAD, E.
Procesamiento Automático de Datos,
México 1976,
EDITORIAL DIANA.

BARTEE, T.
Digital Computer Fundamentals.
Harvard University;
Mc. Graw-Hill Kogakusha, 1977.

COHEN, L.
S/O - Sistemas Operativos.
Buenos Aires, 1976,
Ed. El Ateneo.

DESIGNING WITH TTL INTEGRATED CIRCUITS
Texax Instruments Inc,
Mc. Graw-Hill Kogakusha, 1971.

GEREZ, V. y GRIJALVA, M.
El Enfoque de Sistemas.
México, 1976.
Ed. Limusa.

HARDWARE REFERENCE MANUAL.
American Microsistems, Inc.
1976.

MARTIN, J.
Introducción al Teleprocesamiento,
México, 1978.
Ed. Diana.

MICROELECTRONICA.
Barcelona 1977,
Investigación y Ciencia.

MILLMAN, J. y HALKIAS, C.
Integrated Electronics.
Columbia 1972.
Mc Graw Hill Kogakusha.

M6800 MICROPROCESSORS APPLICATION MANUAL.
Phoenix, 1975.
Motorola Semiconductor Products.

M6800 MICROPROCESSORS PROGRAMMING MANUAL.
Phoenix, 1976.
Motorola Semiconductor Products.

OGDIN, CAROL.
EDN Microcomputer Desing Course
EDN. Pags. 139-316.
1976.

PROTOTYPING BOARD MANUAL.
American Microsystems, Inc.
1976.

SOFTWARE DATA BOOK.
American Microsystems, Inc.
1976.

TAUB, H. y SCHILLING, D.
Digital Integrated Electronics.
New York, 1977.
Mc. Graw Hill Kogakusha.

TRAKHTEMBROT, B.
Algoritmos y Computadoras,
México, 1973.
Ed. Limusa.