Universidad Nacional Autónoma de México

FACULTAD DE CIENCIAS

DISEÑO Y CONSTRUCCION DE UNA MICROCOMPUTADORA MODULAR PARA LA ENSEÑANZA

T E S I S

Que para obtener el título de:

F I S I C O

P r e s e n t a :

ENRIQUE ORTIZ LUNA

México, D. F. 1982

CONTENIDO

CONTENIDO

INTRODUCCION

En la década de 1970 se lograron notables avances en la tecnología de circuitos integrados fabricandose circuitos - cada vez más complejos formados por miles de transistores; - esto permitio construir microprocesadores con todos sus elementos básicos en un solo circuito integrado.

La idea de usar la tecnología de circuitos integrados - para construir un microprocesador fue concebida en 1969 por el Dr. Ted Hoff de Intel, y fue el Dr. Federico Faggin director del equipo de diseño de Intel quien desarrollo el primer microprocesador; el 4004 para calculadoras programables que apareció en 1971. Este microprocesador de 4 bits no permite interrupciones, su conjunto de instrucciones es muy simple.

La parte más elemental de un sistema digital es el registro, que es un dispositivo físico que puede guardar un - valor que se expresa generalmente como un número. El sistema digital puede ser un circuito de propósito especial o una - computadora de propósito general.

Dado el tiempo suficiente se puede resolver cualquier - problema que tenga una solución convergente con una secuencia apropiada de transferencias entre registros. Lo anterior significa que una computadora puede resolver cualquier problema que tenga una solución definida en un número finito de pasos. La clave de la gran versatilidad de los microprocesadores es que usando el mismo equipo físico (Hardware "dispositivos electrónicos, eléctricos y magnéticos") o uno muy -

parecido se pueden diseñar programas para un gran número de aplicaciones diferentes. Resulta también atractivo su pequeño tamaño y bajo costo. Estas características han permitido al microprocesador infiltrarse en máquinas de control e instrumentos programables aumentando su capacidad.

Las distintas alternativas para diseñar una microcomputadora son:

1.- Construir una microcomputadora completa en una sola tarjeta.

2.- Construir una microcomputadora modular, con cada tarjeta conteniendo uno de los elementos de la microcomputadora como son: CPU (Unidad de Proceso Central), reloj, memoria e interfases de entrada y salida.

Para la enseñanza la opción que resulta más útil es la de una microcomputadora modular, ya que la estructura menos rígida de ésta permite cambiar fácilmente cada tarjeta substituyendola por otra con las características importantes para la aplicación que se requiere. La facilidad que se tiene para colocar instrumentos de medición entre los distintos elementos de la microcomputadora permite que las fallas de hardware puedan aislarse fácilmente. La técnica de alambrado empleada facilita los cambios y pruebas del mismo.

La microcomputadora modular tema de ésta tesis esta formada por cuatro tarjetas, y el microprocesador empleado en la construcción del sistema fue el MC6800 de Motorola.

Las primeras microcomputadoras que aparecieron en el

mercado venian en una sola tarjeta, ésto facilitaba la operación del sistema, que se podía poner en funcionamiento de inmediato pero hizo más difícil su aprendizaje. En éstas tarjetas de circuito impreso resulta difícil implementar algún cambio. La desventaja de la poca transparencia y flexibilidad de éste sistema para la enseñanza fue superada con la construcción de microcomputadoras modulares, que permiten aprender paso por paso el funcionamiento de las mismas.

CAPITULO I

TECNOLOGIA DE MICROPROCESADORES

Actualmente hay en el mercado una gran cantidad de microprocesadores. Para la construcción de éstos se usan principalmente 6 tecnologias de circuitos integrados distintas. Para la selección y aplicación de un microprocesador es importante conocer las características mas importantes de éstas tecnologias.

Los microprocesadores pueden construirse usando transistores bipolares (bipolar junction transistors, BJTs) o transistores de efecto de campo (field-effect transistors, FETs) de tipo MOSFET (metal-oxide-semiconductor field-effect transistor). Los microprocesadores bipolares pueden emplear transistores de tipo ECL (emitter-coupled logic), Schottky o - I^2L (integrated injection logic). Los microprocesadores MOSFET pueden usar circuitos de canal p (PMOS), canal n (NMOS) o simetría complementaria (complementary-symmetry, CMOS).

1.1.- Tecnología Bipolar.

A los transistores bipolares se les llama asi porque en ellos intervienen dos tipos de portadores. Actualmente los - microprocesadores más rápidos se construyen con esta tecnología. Para entender las diferencias entre las distintas tecnologias bipolares consideremos primero la operación de un - transistor bipolar (Figura 1.1). Este es un amplificador inversor, la salida es alta cuando la señal de entrada es baja. Si la señal de entrada a este amplificador fuera un pulso - rectangular, se esperaria que la salida ideal fuera un pulso

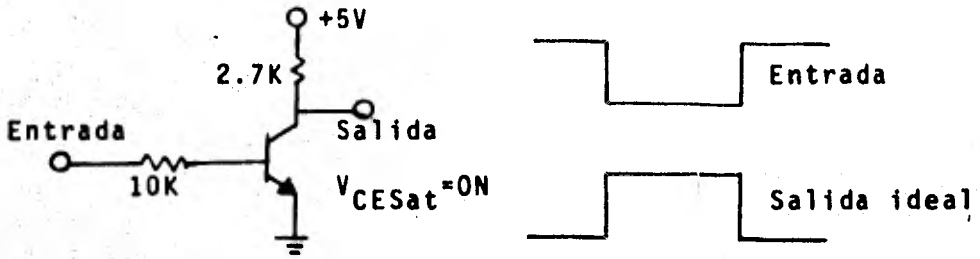


Figura 1.1. Transistor amplificador NPN.

rectangular invertido como se muestra en la Fig. 1.1, sin - embargo la forma de la onda a la salida es como se muestra - en la Fig. 1.2.

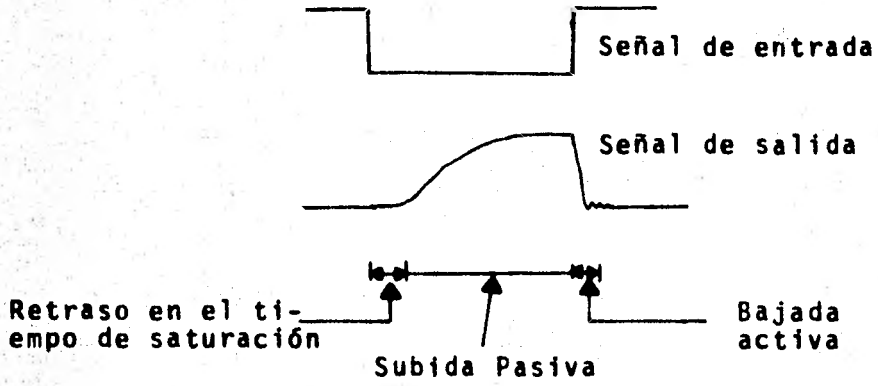


Figura 1.2. Señal de entrada y salida para un transistor amplificador NPN.

La Figura 1.2 muestra 2 efectos no ideales que se pre-- sentan en el transistor bipolar y limitan su velocidad de - operación.

El primer efecto no ideal es el retraso en el tiempo de saturación, que es el tiempo que transcurre entre el pulso - de entrada y la respuesta de la onda que se tiene a la sali- da. Este retraso se produce por el exceso de carga eléctrica

que se acumula en la base del transistor cuando existe una señal de entrada y éste empieza a conducir. El exceso de carga en la base continua proporcionando corriente al colector por un corto periodo de tiempo después que la señal de entrada baja.

El segundo efecto no ideal es el tiempo que transcurre para que la onda a la salida alcance su valor máximo (saturación).

Cuando la señal de entrada regresa a su estado inicial la corriente falla de nuevo para responder en forma inmediata. El transistor tiene a la salida una corriente de colector que es β veces la corriente de base (Beta es la ganancia de corriente del transistor). A causa de esto cualquier capacidad de difusión que se tenga a la salida se descarga rápidamente cuando el transistor se apaga y cuando ésta se carga a través de la resistencia del colector lo hace en forma pasiva. La corriente del colector se incrementa o disminuye siguiendo una curva exponencial, cuya constante de tiempo esta dada por:

$$t_r = h_{FE}(C_c R_c + \frac{1}{W_t}) \simeq 2.2 R_c C_c$$

donde $h_{FE} = \beta = \frac{I_c}{I_B}$ es la ganancia de corriente directa, C_c es la capacitancia de transición del colector y W_t es la frecuencia en radianes para la cual la ganancia de corriente es unitaria.

Los efectos de retraso en el tiempo de saturación y el tiempo de carga pasivo disminuyen la velocidad de operación de un simple transistor bipolar. Como en un microprocesador la velocidad de operación es importante, se han diseñado

circuitos para disminuir éstos efectos. Una forma de elimi-- nar el retraso en el tiempo de saturación es no permitiendo que los transistores alcancen la saturación, lo que se con-- sigue conectando un diodo Schottky de la base al colector - del transistor que impide que la unión base colector del - transistor se polarize en forma directa, ésta es una condi-- ción necesaria para que exista saturación. Un transistor con éste diodo especial se conoce como transistor Schottky que - tiene el símbolo mostrado en la Figura 1.3.

a b

Figura 1.3. a. Diodo Schottky conectado de base a colector.
b. Símbolo de un transistor Schottky.

El transistor Schottky elimina virtualmente el retraso en el tiempo de saturación. Un microprocesador construido - con ésta tecnología es el 2901 de Advanced Micro Devices.

Otro método de evitar retrasos en el tiempo de satura-- ción es polarizar el transistor para que opere solo en su re gión lineal mediante su funcionamiento en clase A, en el que la excursión en corriente de los transistores esta limitada alrededor de un punto de operación escogido por polarización. Esto se logra usando lógica con acoplamiento por emisor (emi tter-coupled logic, ECL). El circuito básico se muestra -

en la figura 1.4. Las grandes computadoras y los circuitos - digitales más rápidos usan actualmente ésta tecnología. Un - microprocesador construido con el empleo de tecnología ECL - es el 10800 de Motorola.

Para resolver el problema de tiempo de carga lento en - el circuito de la figura 1.1 existen varias alternativas. Una de éstas consiste en reducir el valor de la resistencia del colector. Esta solución tiene el efecto indeseable de -

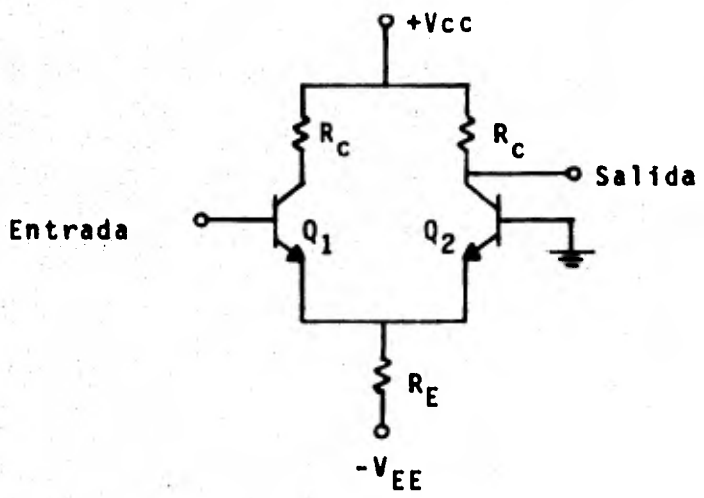


Figura 1.4. Amplificador usado en microprocesadores ECL.

producir un aumento en el consumo de potencia cuando el transistor esta en saturación, lo cual puede verse de la siguiente relación.

$$\text{Potencia Disipada} = \frac{V_{cc}^2}{R_c}$$

La potencia disipada esta dada en miliwatts, el voltaje de - la fuente en volts y la resistencia del colector en Kiloohms.

La técnica que se emplea más comunmente para reducir el

problema de tiempo de carga lento es usar dos transistores - en una configuración "totem pole". En ésta configuración un transistor proporciona una subida activa (pull-up) y el otro proporciona una bajada activa (pull-down). Q_2 y Q_3 son los - transistores que forman la configuración "totem pole".

Q_1 se usa como un inversor de fase para asegurar que Q_2 este en corte cuando Q_3 esta en saturación y viceversa. Para evitar retrazos en el tiempo de saturación se usan transistores Schottky en Q_1 y Q_3.

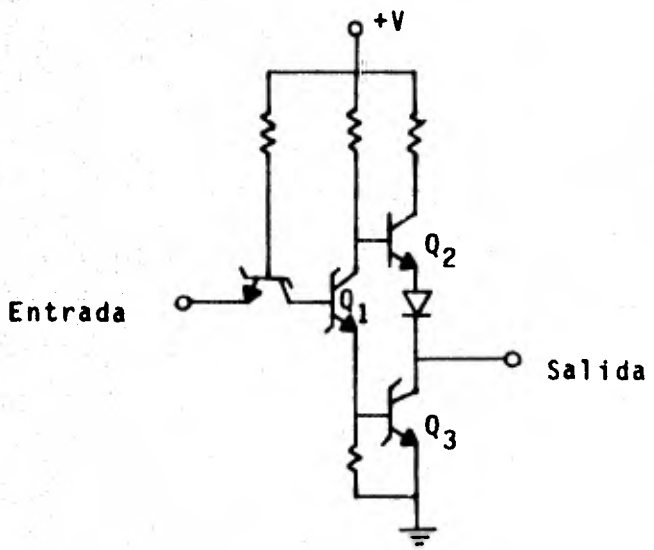


Figura 1.5. Configuración totem pole de un circuito amplificador usado en microprocesadores Schottky.

La tecnología bipolar más reciente que se aplica a microprocesadores es la lógica integrada de inyección (Integrated Injection Logic, I^2L). I^2L usa transistores saturados, - por lo que las velocidades que se tienen son menores que las de circuitos que emplean tecnología Schottky o ECL.

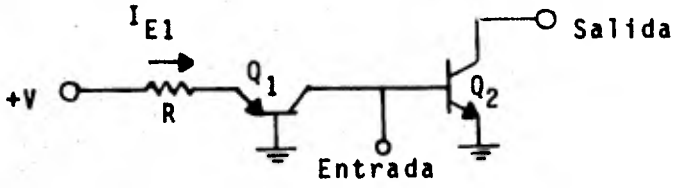


Figura 1.6. Circuito I^2L básico

Cuando la señal de entrada es alta (Figura 1.6), I_{E1} fluye a través de Q_1 y enciende Q_2. Cuando la señal de entrada es baja (al potencial de tierra), la corriente I_{E1} va a tierra y Q_2 se apaga (corte) por no haber corriente de base. Una característica importante de éste circuito es que se tiene una corriente I_{E1} programable. La magnitud de la corriente I_{E1} esta dada por el voltaje de la fuente V y la resistencia en serie:

$$\text{Corriente } I_{E1} = \frac{V - V_{BE}}{R}$$

Dicha corriente esta dada en miliamperes, V en volts y R en Kilo-ohms.

La corriente I_{E1} la fija el usuario. Cuando ésta corriente es pequeña el consumo de potencia es bajo y se reduce la velocidad de operación. Si se aumenta la corriente, la velocidad de operación es mayor y se consume más potencia.

Una característica de un circuito I^2L es que puede emplear una fuente de hasta 0.8V. La poca sensibilidad de los circuitos I^2L al voltaje y su consumo de potencia

programable hacen ésta tecnología atractiva en aplicaciones - donde la fuente de voltaje es una batería.

1.2.- Tecnología MOSFET (metal-oxide-semiconductor field- - effect transistor).

El transistor MOSFET es un transistor unipolar, puesto que solo hay un tipo de portadores.

La mayoría de los microprocesadores que se fabrican actualmente usan transistores MOSFET en sus circuitos integrados. La principal ventaja de la tecnología MOSFET sobre la - bipolar es mayor densidad, lo que permite implementar más - funciones de las que se tienen empleando tecnología bipolar.

Existen dos tipos de transistores de campo MOSFET que - pueden usarse en los circuitos de Microprocesadores. Estos - son: Los de canal P, donde los portadores eléctricos son agujeros en el material semiconductor y transistores de canal N donde los portadores son electrones.

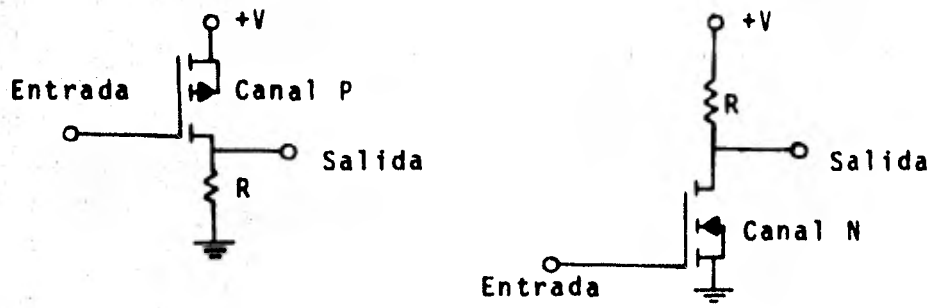


Figura 1.7. Amplificadores MOSFET de canal P y canal N.

Los amplificadores MOSFET básicos de canal P y canal N se muestran en la Figura 1.7.

Los primeros microprocesadores usaron tecnología MOS -- de canal P, cuyas principales desventajas son:

a). Los agujeros tienen una movilidad menor que los electrones, por lo cual son más lentos que los de canal N.

b). Los circuitos de canal P proporcionan una subida activa (pull-up) y una bajada pasiva (pull-down) ante las cargas - externas. Por su parte los de canal N proporcionan una bajada activa (pull-down), que es más efectiva para manejar in-- terfases TTL (Transistor Transistor Logic) empleadas comun-- mente con los microprocesadores.

Actualmente la tecnología de canal N es la más usada en la fabricación de Microprocesadores. El primer microprocesador que empleo tecnología de canal N fue el 8080 que apare-- ció en 1973. La máxima frecuencia del reloj para el 8080 original era de 2MHz. Los avances logrados en la tecnología de canal N permiten construir microprocesadores con frecuencias de reloj de 10 MHz.

Existe una clasificación adicional en los transistores de campo de canal P y canal N y es la de dispositivos de vaciado (depletion mode) o de enriquecimiento (enhancement mode). Los transistores de vaciado normalmente estan en saturación y es necesario aplicar un voltaje para llevarlos a corte, en estos transistores, el canal existe en ausencia de - voltaje y éste se utiliza para hacer desaparecer y bloquear la conducción.

Los transistores de enriquecimiento estan normalmente en corte y es necesario aplicar un voltaje para llevarlos -

a saturación.

Los primeros microprocesadores usaron el tipo de enri-- quecimiento, por lo cual se necesitaba un voltaje adicional. Para eliminar el uso adicional de una fuente ahora se usan los transistores de vaciado.

El tercer tipo de transistores MOS es el transistor complementario CMOS (complementary MOS), que emplea transisto-- res PMOS y NMOS en una sola estructura como se muestra en la figura 1.8.

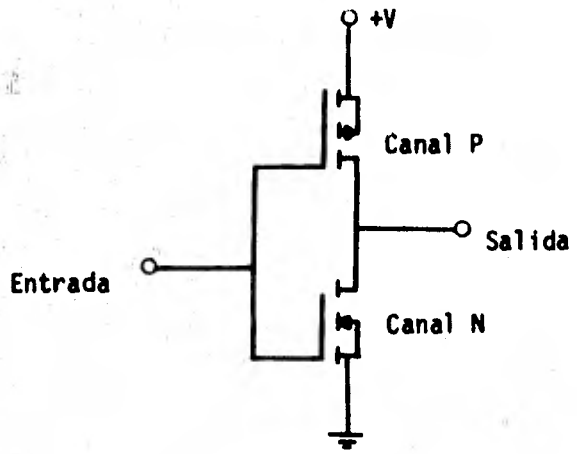
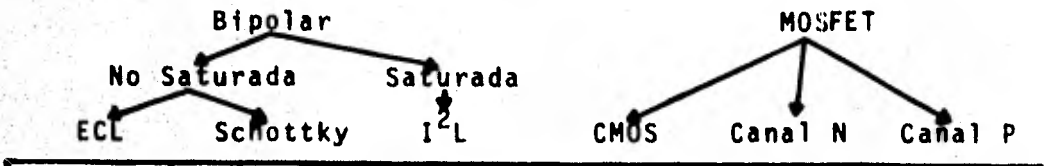


Figura 1.8. Circuito MOS complementario

El consumo de potencia de un circuito CMOS es muy bajo ya que no se tiene un camino continuo al paso de la corriente debido a que siempre esta cortado uno de los dos transistores. La velocidad de un circuito CMOS convencional esta - limitada por las capacidades del sustrato semiconductor (silicio). Para reducir el efecto de tales capacidades se utiliza un sustrato aislante (zafiro) en los circuitos MOS -

(Silicon on Saphire "SOS"). Hewlett Packard desarrollo un - microprocesador empleando esta tecnología.

La figura 1.9 muestra un resumen de las tecnologias de los microprocesadores con ejemplos representativos.



Motorola 10800.	Intel 3001. Advanced Micro Devices Am 2901.	Texas Inst. SBP 0400 SBP 9900.	Intersil 6100. RCA COSMAC	Intel 8080,8085, 8748,8086. Motorola 6800,6802 6809. Fairchild F8. MOS technology MCS 6502 Zilog Z-80,Z-8, Z-8000	Intel 4004 4040 8008 National IMP PACE

Figura 1.9. Tecnologias de microprocesadores más empleados y ejemplos representativos.

CAPITULO II

ELEMENTOS DE UNA MICROCOMPUTADORA

2.1.- Estructura Básica.

Usando las compuertas AND (Función Lógica Y), OR (Función lógica O) y NOT (inversora) como elementos básicos se puede construir la amplia variedad de circuitos lógicos que se encuentran en las computadoras. Estas compuertas pueden usarse como bloques de circuitos más complejos que realizan operaciones aritméticas, lógicas y procesamiento de datos.

Una computadora necesita también elementos de memoria para almacenar dígitos binarios. El grupo de elementos de memoria que trabajan juntos como unidad se llama registro, cuya función más simple es almacenar una palabra binaria.

Los elementos básicos de una computadora son: Unidad Aritmética y Lógica (ALU "Arithmetic Logic Unit), que modifica los datos que se presentan a su entrada de acuerdo con la operación indicada por la instrucción.

La memoria es un medio para guardar instrucciones y datos para su uso posterior.

El sistema de entrada y salida recibe datos para que se procesen en la computadora y los presenta a la salida para su uso. Esta interfase debe resolver problemas de:

a). Tiempo, ya que el equipo periférico (tal como: Teletipo, Terminal de rayos catódicos, convertidores A/D y D/A) es generalmente más lento que la computadora.

b). De Formato de la información transmitida a la computadora (traducción serie-paralelo).

c). Hardware, ya que el equipo periférico no tiene la misma lógica que la computadora.

La unidad de control coordina todas las partes de la - computadora (Figura 2.1), determinando la forma en que se - transfieren los registros en la computadora con la secuencia y tiempo correcto. Esta unidad supervisa la ejecución correcta de cada ciclo, el número de ciclos depende de la instrucción procesada.

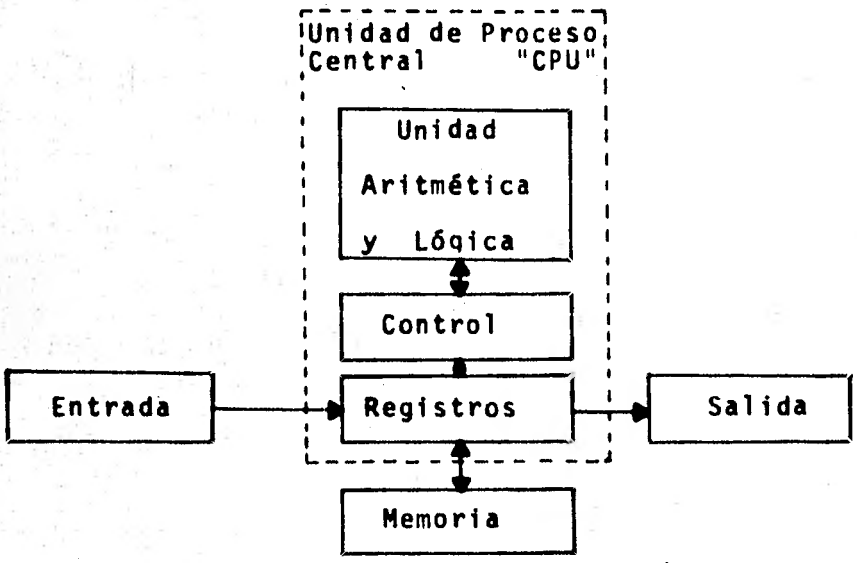


Figura 2.1. Elementos de una Computadora.

Las computadoras estan formadas por registros, y su arquitectura depende de las distintas transferencias que se - pueden realizar entre éstos.

a). Memoria.- La memoria se puede considerar como un conjunto de registros idénticos que pueden seleccionarse individualmente usando otro registro (Figura 2.2). Cuando se quiere seleccionar un registro específico de la memoria, se da - su dirección al registro de direcciones que tiene una entrada de N bits, que decodifica proporcionando una salida de 2^N.

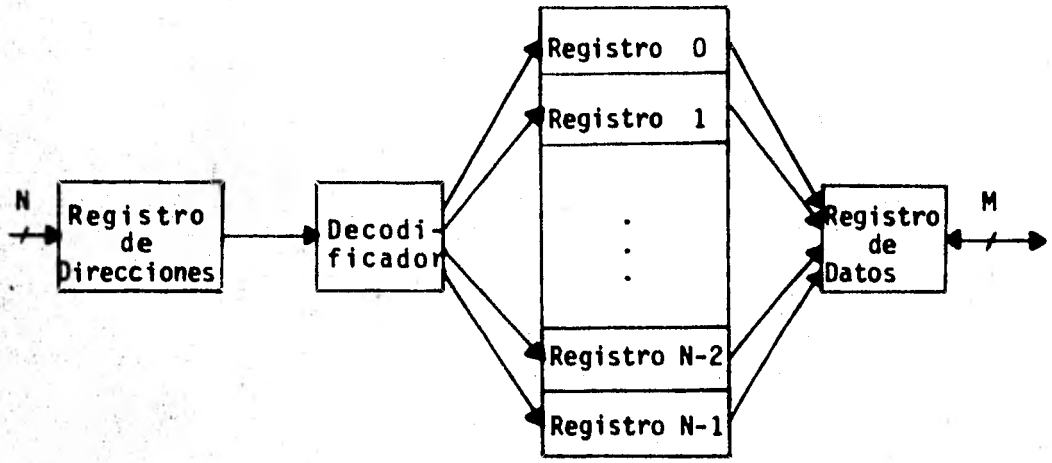


Figura 2.2. Estructura básica de una memoria seleccionable en forma aleatoria.

Si se realiza una lectura del registro seleccionado, el dato en éste se transfiere al registro de datos que determina el tamaño de una palabra en la memoria.

b). Sección de Entrada y Salida. Esta sección esta formada - por registros seleccionables que permiten el intercambio de datos con la computadora.

En el caso del microprocesador, éste proporciona una - combinación de señales que se usan para seleccionar los puertos y controlar la dirección de transferencia de datos.

Para la sección de entrada se necesita un registro que selecciona uno de los puertos de entrada. Este es un decodificador que permite la transferencia del contenido del registro de entrada de datos a la computadora (Figura 2.3).

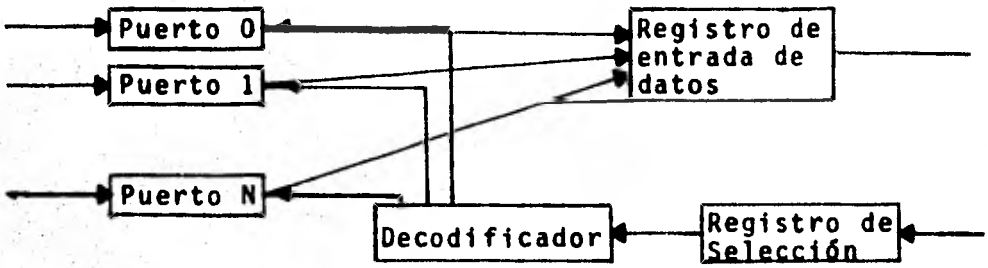


Figura 2.3. Selección de Puertos de Entrada.

c). Unidad Aritmética y Lógica (ALU). Esta es otra colección de registros, que ejecuta instrucciones Aritméticas, Lógicas y corrimientos del CPU. Las operaciones sobre datos se realizan en forma interna como transferencias entre registros y externamente éstos se proporcionan a un registro de datos (Figura 2.4).

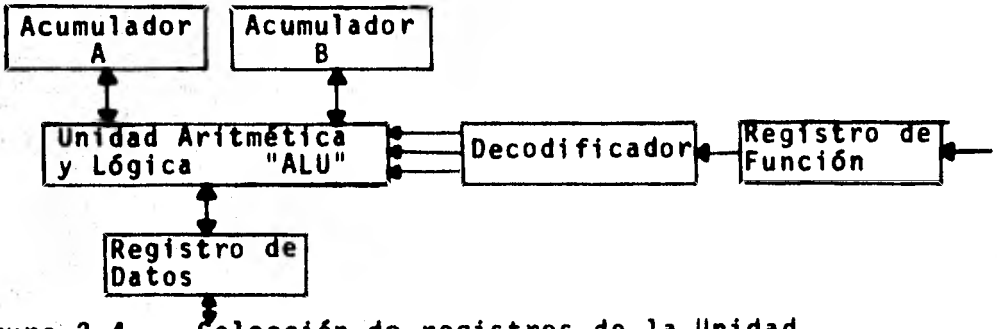


Figura 2.4. Colección de registros de la Unidad Aritmética y Lógica.

La función específica que se realiza sobre éstos se representa como un número binario que se proporciona al Registro de Función de la Unidad Aritmética y Lógica. Esta trabaja generalmente sobre parejas de registros. Asi, para sumar un número a otro es necesario que tenga los valores de los dos datos al mismo tiempo. La unidad Aritmética del MC 6800 tiene dos registros llamados Acumuladores (A,B).

d). Unidad de Control. La sección de control coordina todas las posibles transferencias entre registros que ocurren dentro de la computadora. Para especificar que transferencia - tendrá lugar se da una instrucción a la computadora. La lista de todas las instrucciones posibles se conoce como el conjunto de instrucciones.

Una instrucción típica puede ser una que seleccione un registro de entrada particular y transfiera su contenido a - un registro de memoria específico. Esta transferencia puede consistir de otras transferencias de registros que no están bajo el control directo del programador. Las transferencias internas de registros se conocen como microinstrucciones y son gobernadas por la sección de control de la computadora.

En algunas computadoras la sección de control tiene una memoria solo de lectura (ROM) que contiene una o más pala- - bras para cada instrucción del conjunto de instrucciones - (programa). Este genera las señales de control internas necesarias para controlar las transferencias de registros que - requiere la instrucción. Si la memoria ROM puede modificarse se dice que la computadora es microprogramable. La mayoría - de los microprocesadores se microprograman en la fábrica.

Cuando se ejecuta una instrucción, ésta se busca en la memoria, tal selección la realiza un registro llamado Contador de Programa (PC "Program Counter"), el contenido de éste aparece en el bus de direcciones. Mientras tiene lugar la - acción física de selección de registro (tiempo de acceso) el de programa se incrementa en uno, y cuando se ejecuta la -

instrucción tiene ya la dirección de la siguiente que se ejecutara.

Después de que se tiene el contenido del registro de memoria seleccionado en el registro de datos, la sección de control transfiere su constenido al registro de instrucción. En algunos casos el contenido del PC puede modificarse por el programa mismo, lo que permite ejecutar instrucciones guardadas en otra parte del programa.

La estructura general de una computadora hipotética simplificada se muestra en la siguiente figura:

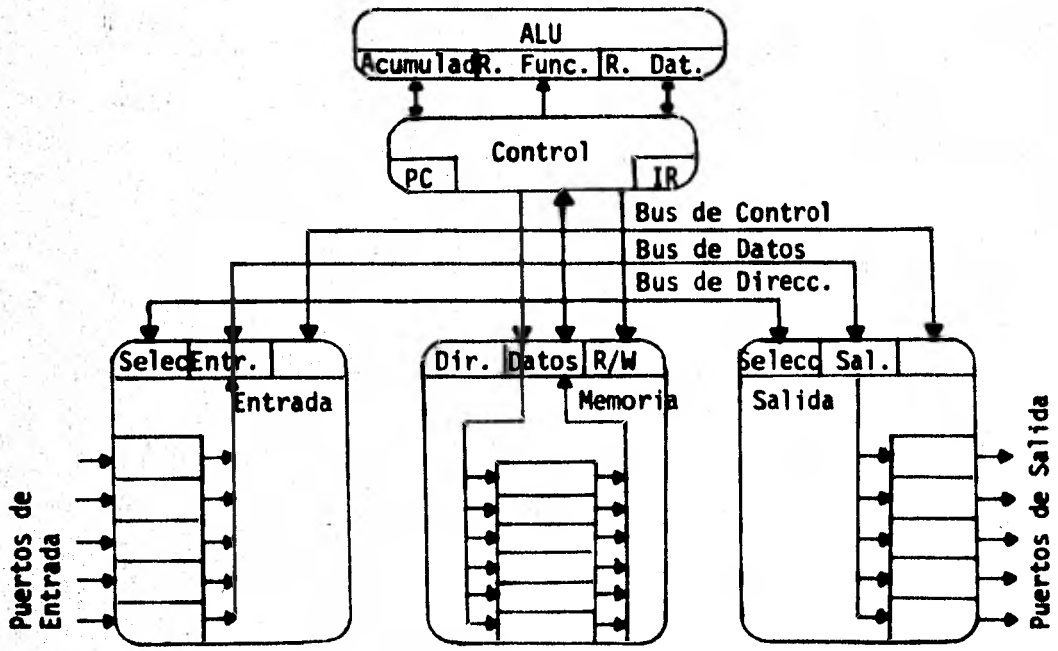


Figura 2.5. Registros que forman una computadora simple.

La comunicación entre los elementos de una microcomputadora se realiza a través de un conjunto de lineas conocidas como Bus (que proviene de la palabra latina "Omnibus" que significa para todos). Todos los microprocesadores tienen 3 Buses: Bus de Direcciones, Bus de Datos y Bus de Control (Figura 2.5).

Cuando el CPU quiere leer la información de un registro de memoria dado, manda ésta dirección al registro de direcciones y transfiere su contenido a través del Bus de direcciones. Después de que ésta dirección se decodifica el registro seleccionado transfiere su contenido al registro de datos y lo envia al CPU usando el Bus de Datos.

Para escribir se aplica el mismo principio: El contenido del registro de datos se almacena en la dirección indicada por el registro de direcciones.

El bus de direcciones es unidireccional y transfiere las señales generadas por el CPU a los otros elementos del sistema.

El bus de datos es bidireccional y es el conjunto de lineas que permiten llevar los datos del microprocesador a los elementos del sistema y de éstos al microprocesador.

Las inter-relaciones entre los 3 buses de una microcomputadora se pueden representar más facilmente usando un diagrama de tiempo (que se vera posteriormente para el caso del MC 6800).

2.2.- Microprocesador MC 6800.

a). Características.

El microprocesador MC 6800 es un microprocesador de 8 - bits en el Bus de Datos y 16 bits en el de direcciones, fa-- bricado con tecnología MOS de canal N.

La comunicación entre los elementos de la computadora - es a través de los buses de Datos, Direcciones y Control.

Los microprocesadores al igual que las computadoras ope ran en forma síncrona. La transferencia de datos entre regis tros internos y entre registros de elementos distintos de la computadora esta sincronizada por señales de reloj y señales de control derivadas de éste (Figura 2.6).

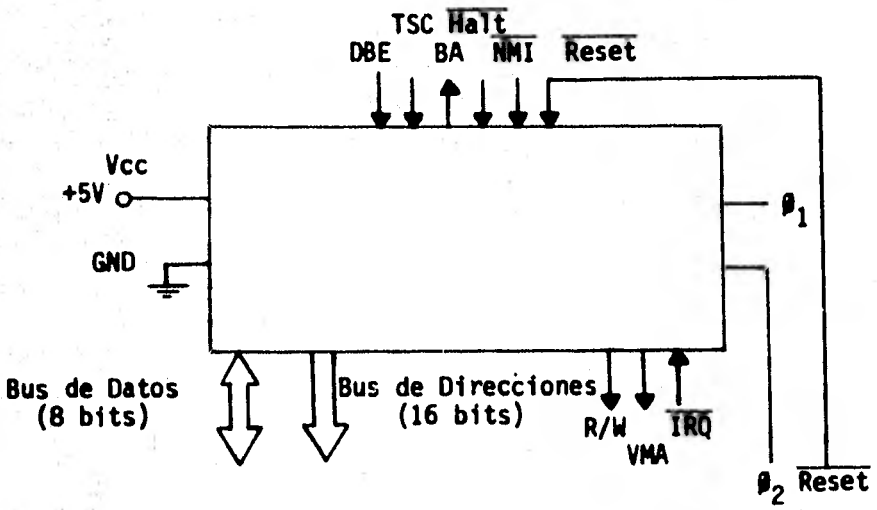


Figura 2.6. Señales de entrada y salida del Microprocesador MC 6800.

b). Registros del Microprocesador.

El Microprocesador MC 6800 tiene 3 registros de 8 bits y 3 registros de 16 bits. Los acumuladores A (ACCA) y B (ACCB) son registros de 8 bits que usa el microprocesador para guardar operandos y resultados de la Unidad Aritmética y Lógica (ALU). El registro de condición (CC) de 8 bits, tiene 6 que se modifican según los resultados de una operación de la Unidad Aritmética y Lógica (Figura 2.7).

A continuación se describen los bits del registro de condición (CC).

1.- Bit C (Carry-Borrow). Se enciende (C=1) para indicar que hay un acarreo del bit más significativo del acumulador como resultado de una operación. Cuando esto no ocurre C = 0. Las instrucciones de rotación y corrimiento también afectan éste bit.

2.- Bit V (Overflow). Se enciende (V = 1) cuando se tiene un resultado que sobrepasa la capacidad del registro durante una operación aritmética con complemento a dos. Esto indica que se ha excedido el número máximo posible (+127) o el número mínimo (-128) que se puede representar en notación de complemento a dos. En caso contrario V = 0.

3.- Bit Z (Zero). Se enciende (Z = 1) cuando el resultado de una operación artmética anterior es cero. Si el resultado es distinto de cero Z = 0.

4.- Bit N (Negative). Se enciende (N = 1) cuando el bit más significativo del resultado anterior es 1, lo que indica que el número que se tiene es negativo (en complemento a dos).

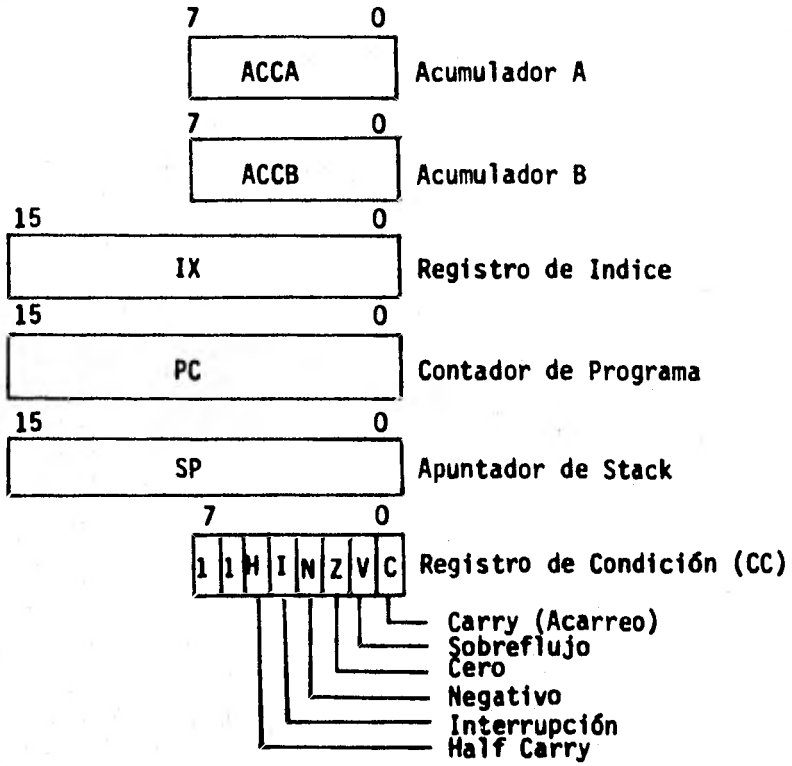


Figura 2.7. Registros del Microprocesador.

Cuando N = 0 el bit más significativo del número que se tiene es cero, lo que indica que el número es cero o positivo.

5.- Bit I (Interrupt Mask). Se usa para habilitar o deshabilitar interrupciones enmascarables. Cuando I = 1 las interrupciones enmascarables se deshabilitan.

Su nivel es 1 después que ocurre una interrupción enmascarable o se da la instrucción SWI (Software Interrupt) o SEI (Set interrupt mask bit). Regresa a I = 0 después de ejecutar la instrucción CLI o RTI.

6.- Bit H (Half carry). Se enciende (H = 1) si se tiene un acarreo del bit 3 al bit 4 durante la ejecución de alguna de las instrucciones ABA, ADC o ADD. Si durante éstas operaciones no hay acarreo de las posiciones 3 a 4 se apaga el bit H.

Los bits 7 y 8 del registro de Condición no se usan y - son siempre unos.

El registro Contador de Programa (PC) de 16 bits, guarda la dirección de la instrucción que se ejecuta y después - de transferir su contenido al bus de direcciones se incrementa en forma automática. Cuando se tiene un salto en el pro-- grama se transfiere la nueva dirección del programa al PC.

Apuntador de Stack (SP). Este registro de 16 bits guarda la dirección de la siguiente localidad de memoria de un - Stack situado en cualquier parte de la memoria RAM del sistema. La memoria del stack esta organizada como una memoria - LIFO (Last in first-out) donde el último dato que se guarda es el primero que sale. Cuando se guarda un byte de información en el stack, se hace en la dirección que contiene el - apuntador de stack y se decrementa en uno el apuntador. El - apuntador de stack se incrementa en uno antes de recuperar - información del stack.

Registro de Indice (X). Es un registro de dos bytes (16 bits) que se usa como base de un apuntador cuando se usa direccionamiento indiciado. En las instrucciones que usan di-- reccionamiento indiciado se tiene un byte adicional que es-- pecifica el desplazamiento de la base.

Las señales de entrada y salida del microprocesador pueden dividirse en 3 grupos: líneas de interfase, líneas de - control del Bus y líneas de control del procesador. Las lí-- neas de interfase son el bus de datos y el bus de direccio-- nes. El bus de direcciones esta formado por 16 líneas, cada

línea puede ser un 1 lógico o un 0 lógico, por lo cual un microprocesador con P (16) líneas de direcciones puede direccionar 2^P localidades de memoria. El microprocesador MC 6800 puede direccionar 2^{16}= 65,536 localidades de memoria. Este conjunto de localidades se conoce como el espacio de memoria.

Para representar la dirección de una localidad en el espacio de memoria se usa notación hexadecimal, la dirección de memoria más baja es la 0000 y la dirección de memoria más alta es la FFFF.

La unidad para expresar el tamaño de la memoria se conoce como Kilopalabra que es igual a 2^{10} ó 1024 palabras. En éste caso donde la palabra es de 8 bits la unidad es un Kilobyte que corresponde a 1024 bytes y la memoria que puede direccionar el microprocesador es de 64K.

Los circuitos de salida (drivers) del bus de direcciones tienen capacidad para manejar una carga TTL (Transistor-transistor Logic). Cuando las salidas estan apagadas se comportan como un circuito abierto, ésto permite que se pueda tener acceso directo a memoria (DMA). Para que las líneas de dirección pasen al tercer estado (Figura 2.8) es necesario poner TSC en su estado alto.

El bus de datos esta formado por 8 líneas bidireccionales que definen el tamaño de la palabra del microprocesador. A través de estas el microprocesador transfiere los datos a la memoria y dispositivos periféricos. Para que las líneas del bus de datos se encuentren en el tercer estado (Figura 2.8), es necesario poner DBE (Data Bus Enable) en su estado bajo, (que deshabilita el bus de datos).

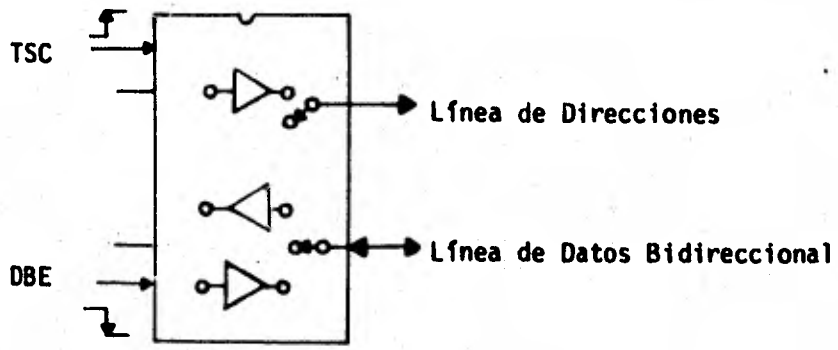


Figura 2.8. Muestra de una línea del Bus de Direcciones y una línea del bus de Datos en la condición de tercer estado (tristate).

Señales de Reloj.

El microprocesador tiene dos señales de reloj: ϕ_1 y ϕ_2 que controlan la transferencia de datos entre registros in-- ternos.

Por lo general ϕ_2 esta unido con DBE (Data Bus Enable), por lo cual controla los buffers de salida del bus de datos y la transferencia de éstos al microprocesador, que ocurre - en la parte activa (alta) de ϕ_2.

Si se desea aumentar el tiempo de escritura del micro-- procesador es necesario disminuir el tiempo de $\overline{DBE}$, que no - debe ser menor de un tiempo mínimo ($t_{\overline{DBE}}$) y debe ocurrir durante la parte alta de ϕ_1 (Figura 2.9). La frecuencia permitida del reloj puede variar de 0.1 MHz a 1.0 MHz.

La línea de salida de R/W indica a los dispositivos de entrada/salida y memorias si el microprocesador esta en un -

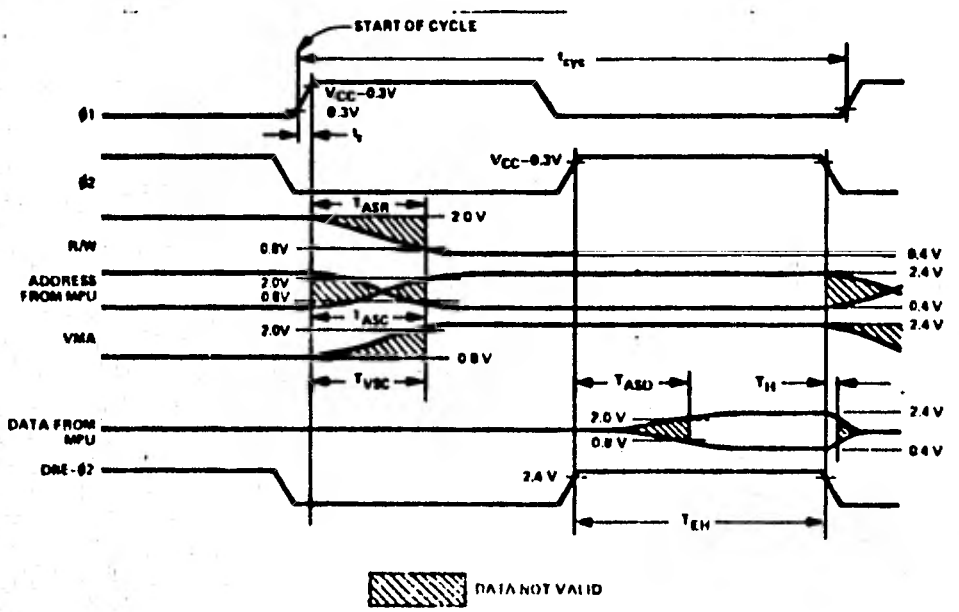


Figura 2.9. Diagrama de tiempo de escritura.

estado de lectura (alto) o de escritura (bajo). El estado - normal de ésta línea es alto (lectura), lo que impide escrituras erróneas en los dispositivos periféricos (cuando no - hay información presente en el bus de direcciones).

La señal de salida de R/W pasa al estado de alta impe-- dancia (OFF) si TSC (Three State Control) pasa a su estado - alto o el microprocesador esta detenido.

Dirección de Memoria Válida (Valid Memory Adress "VMA"). Indica a los dispositivos en el bus que el CPU quiere tener acceso a memoria ROM, RAM o a un dispositivo de entrada/salida. Esta señal es activa en alto. Cuando VMA = 0 el ROM, RAM y dispositivos de entrada/salida quedan deshabilitados e ignoran la información que exista en ese momento en el Bus de Direcciones.

La línea de Interrupt Request (IRQ) es una señal de entrada al microprocesador, que permite que éste sea interrumpido por los periféricos. Antes de dar servicio a la interrupción el microprocesador termina la ejecución de la instrucción que realiza en el momento en que es interrumpido y checa el estado del bit I en el registro de Condición (Condition Code Register), si éste no esta encendido inicia la secuencia de interrupción, encendiendo la bandera I que deshabilita una nueva interrupción y guarda en el Stack el registro de Indice (IX), el Contador de Programas (PC), los dos acumuladores y el registro de Concidión (CC). Al final del ciclo aparece en el bus de direcciones la dirección del vector de interrupciones $\overline{IRQ}$ (FFF8,FFF9) que guarda la dirección de memoria donde se encuentra la rutina de servicio de la interrupción (Figura 2.13). Para que la interrupción sea atendida la línea de $\overline{Halt}$ debe estar en alto.

El microprocesador tiene seis líneas de control que son: 1).- Reset. Cuando se detecta un frente de onda positivo en ésta entrada del microprocesador se reinicia una secuencia de reset. Aparece en el bus de direcciones la dirección FFFE seguida de FFFF, se carga en el PC el contenido de estas localidades de memoria (Figura 2.13), que contienen la dirección inicial de la subrutina de reset y se enciende el bit I que deshabilita las interrupciones enmascarables ($\overline{IRQ}$). Para habilitar las interrupciones debe apagarse el bit I.

Después que se enciende la fuente y ésta alcanza su valor de Vcc es necesario que la línea de $\overline{Reset}$ se mantenga baja durante un mínimo de 8 ciclos de reloj, éste tiempo permi

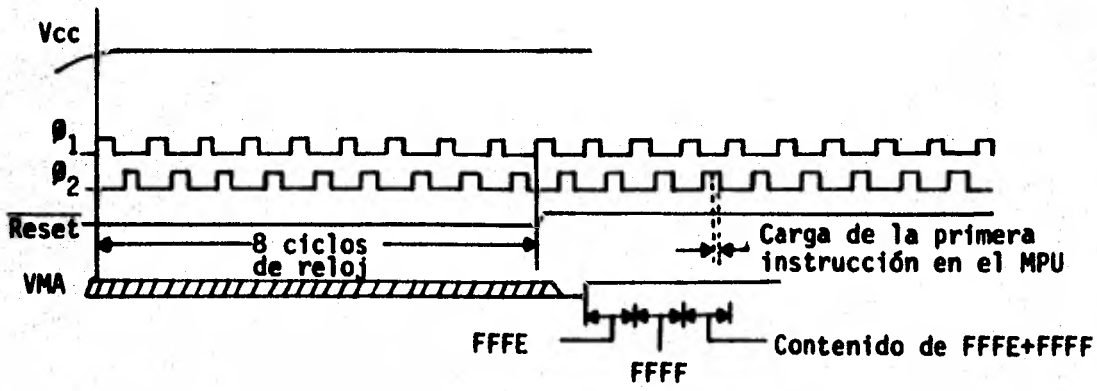


Figura 2.10 Inicialización del microprocesador después del Reset.

te que el microprocesador se estabilice. El estado de las - líneas de salida del microprocesador en este momento es: VMA = bajo, Bus de Datos = alta impedancia, R/W = alto (lectura), y el Bus de Direcciones tiene a su salida la dirección FFFE (Figura 2.10).

2).- NMI (Non-Maskable Interrupt). La secuencia de las operaciones que siguen a una interrupción no enmascarable se inicia con la aparición de un frente de onda negativo en la entrada NMI del microprocesador. Esta interrupción es parecida a IRQ, el procesador termina la instrucción que ejecuta antes de atender la interrupción. El estado del bit I en el - registro de Condición no tiene ningún efecto sobre NMI. Los registros internos del microprocesador se guardan en el - Stack y el contenido de las direcciones de memoria FFFC y -

FFFD (Figura 2.17) se carga en el registro PC, después de lo cual el microprocesador inicia la ejecución del programa de servicio de la interrupción no enmascarable, empezando con - la instrucción direccionada por el PC.

3).- $\overline{\text{Halt}}$. Cuando la línea de entrada de $\overline{\text{Halt}}$ esta en su estado bajo se detiene toda actividad interna del microprocesador. Este cuenta con una señal (Bus Available) que indica el estado en que se encuentra. Cuando BA es baja el microprocesador ejecuta su programa de control. Cuando BA es alta se - detiene toda la actividad del microprocesador, las líneas - del bus de direcciones, bus de datos y R/W se encuentran en su estado de alta impedancia y VMA pasa a su estado bajo.

Si ocurre una interrupción $\overline{\text{NMI}}$ o $\overline{\text{IRQ}}$ cuando el microprocesador esta detenido, ésta sera atendida después de que el microprocesador es sacado de tal estado. Si se da un $\overline{\text{Reset}}$ - cuando el microprocesador esta detenido, el estado de las - líneas del microprocesador sera: VMA = bajo, BA = bajo, bus de datos = alta impedancia, R/W = alto (lectura) y el bus de direcciones tiene la dirección FFFE. Cuando el estado de la línea $\overline{\text{Halt}}$ varia a su estado alto el microprocesador ejecuta la rutina de reset, cuya dirección se encuentra en las localidades FFFE FFFF. Cuando la línea de $\overline{\text{Halt}}$ pasa a su estado bajo el microprocesador termina la ejecución de la última - instrucción, la transición de $\overline{\text{Halt}}$ debe ocurrir un tiempo - t_{pcs} antes del frente de onda negativo del reloj ϕ_1 en el - último ciclo de una instrucción (Figura 2.11). Si la transición ocurre un tiempo menor a t_{pcs} el microprocesador ejecuta

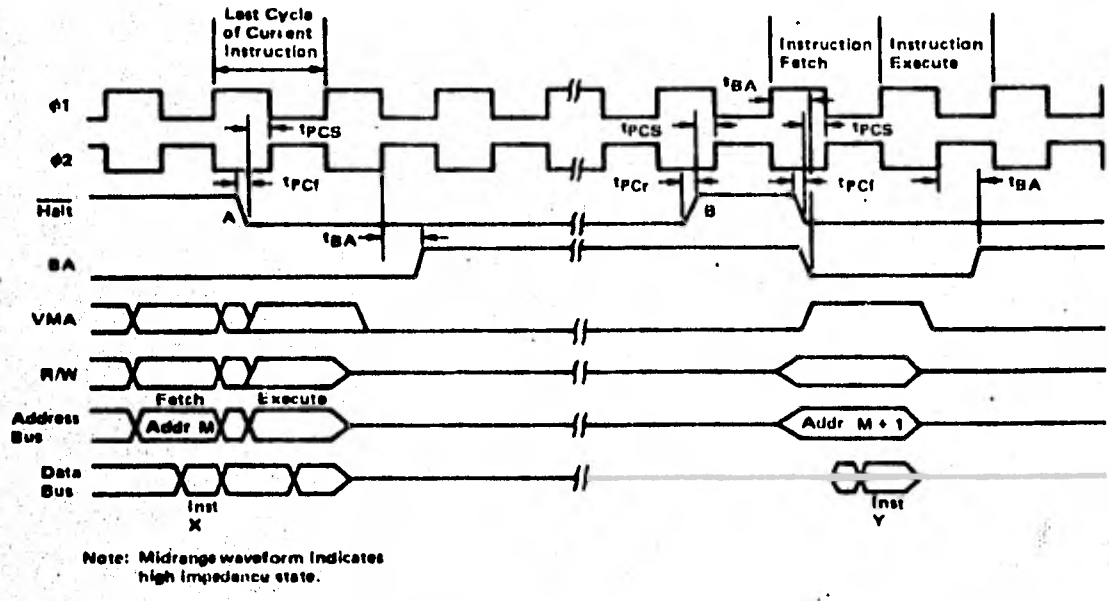


Figura 2.11. Diagrama de tiempo de $\overline{\text{Halt}}$.

la siguiente instrucción y se detiene. El tiempo que tarda - BA en pasar a su estado alto después que se ejecuta la última instrucción es t_{BA} (Figura 2.11).

$\overline{\text{Halt}}$ permite la ejecución de programas instrucción por instrucción, para esto es necesario llevar la línea a su estado alto durante un tiempo igual a 1 ciclo del microprocesador (Figura 2.11).

4).- TSC (Three-State Control). Cuando ésta línea de entrada al microprocesador esta en un 1 lógico, el bus de direcciones y R/W pasan a su estado de alta impedancia. Para impedir falsas lecturas o escrituras VMA y BA pasan a su estado bajo.

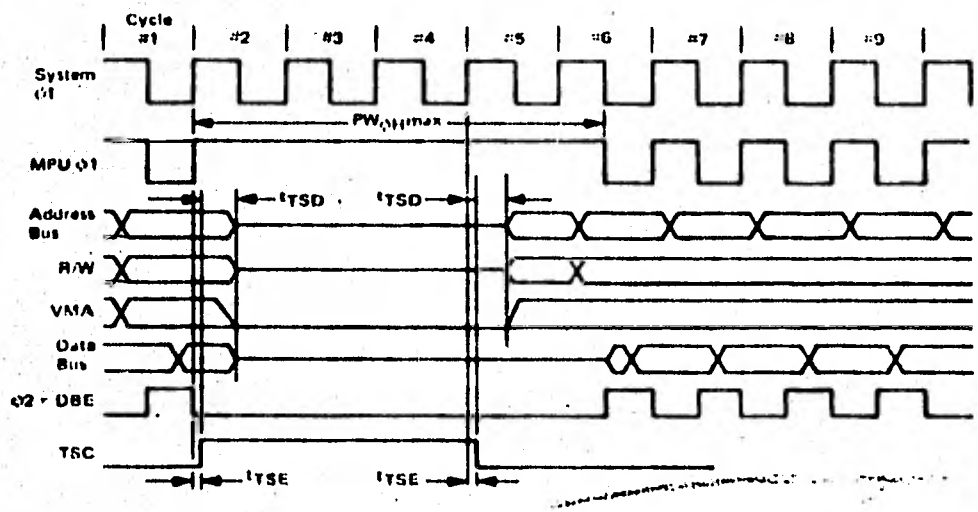


Figura 2.12. Diagrama de tiempo de TSC.

Mientras TSC es alto los relojes $\emptyset_1$ y $\emptyset_2$ deben ser mantenidos alto y bajo respectivamente para retrazar la ejecución del programa. Como el microprocesador es un dispositivo dinámico, el tiempo máximo que puede ser detenido el reloj del microprocesador sin que halla destrucción de datos dentro de éste es $PW_{\emptyset H}$ (Figura 2.12). TSC se puede usar para accesos directos a memoria, durante los ciclos #3 y #4, el ciclo #5 se usa para sincronización.

El tiempo de transición de TSC es t_{TSE} (three state enable), una vez alcanzado tal estado el bus de direcciones y R/W pasan al tercer estado después de un tiempo t_{TSD} (three state delay).

5).- BA (Bus Available). Es una señal de salida del micropro

cesador que normalmente esta en un estado bajo; cuando pasa a su estado alto indica que el microprocesador esta detenido, lo que ocurre cuando el estado de la línea $\overline{\text{Halt}}$ es bajo o el procesador esta en el estado de wait. El estado de BA cuando TSC esta en un uno lógico es bajo.

6).- DBE (Data Bus Enable). Es una señal de control del bus de datos del microprocesador. Esta entrada es compatible con TTL y en operación normal es controlada por el reloj $\emptyset_2$. Los drivers del bus de datos se deshabilitan durante los ciclos de lectura. En accesos directos a memoria el estado de DBE debe ser bajo.

Vector		Descripción
MS	LS	
FFFE	FFFF	$\overline{\text{Reset}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	$\overline{\text{SWI}}$
FFF8	FFF9	$\overline{\text{IRQ}}$

Figura 2.13. Localidades de memoria de los vectores de interrupción.

El microprocesador MC 6800 tiene 72 instrucciones básicas, que pueden agruparse en: Instrucciones aritméticas binarias y decimales, lógicas, corrimientos, giros, carga y almacenamiento, saltos condicionales o incondicionales, interrupciones y manipulación del Stack.

c).- Modos de Direccionamiento.

El microprocesador MC 6800 tiene 7 modos de direccionamiento, el ensamblador examina el operador y el operando para determinar el modo de direccionamiento dado. Estos son:

1). Direccionamiento del Acumulador. Estas son instrucciones de 1 byte. Este modo de direccionamiento se especifica escribiendo en el campo del operando únicamente el carácter A o B que corresponda al acumulador.

2). Direccionamiento implicado (Inherente). En éste caso el operador indica los registros que tienen el operando o donde se guarda el resultado. Por ejemplo, el operador ABA necesita dos operandos que se encuentran en el acumulador A y B del microprocesador y determina que el resultado de la ejecución se guarde en A. Toda la información necesaria para el direccionamiento se encuentra en el operador y no se necesita operando.

3). Direccionamiento Inmediato. Son instrucciones de dos o tres bytes con un operando de ocho o dieciseis bits. Para operaciones con el acumulador se requiere un operando de 8 bits que esta en el segundo byte de la instrucción.Cuando el operando es de 16 bits como en el caso de las instrucciones CPX, LDS y LDX; el segundo byte de la instrucción tiene la -

parte más significativa del operando y el tercer byte la parte menos significativa del operando.

CODIGO DE OPERACION	OPERANDO INM. (DATO)

CODIGO DE OPERACION	OPERANDO INM. PARTE MAS S.	OPERANDO INM. PARTE MENOS S.

4).. Direccionamiento Directo. Son instrucciones de dos bytes. El primer byte de la instrucción contiene el código de operación y el segundo la dirección del operando. El usuario puede direccionar las localidades de memoria 0-255 y lograr un ahorro significativo de tiempo guardando datos en esas localidades.

CODIGO DE OPERACION	DIRECCION 0 - 255

5). Direccionamiento Extendido. En éste modo de direcciona-- miento el operando se encuentra en el segundo y tercer byte de la instrucción. El segundo byte tiene los 8 bits más significativos de la dirección y el tercer byte los 8 bits me-- nos significativos de la dirección. Estas instrucciones son de 3 bytes, y la dirección que se tiene en el operando es - una dirección absoluta. El direccionamiento extendido es semejante al direccionamiento directo, pero en este caso la dirección que se encuentra en el operando es de 16 bits.

CODIGO DE OPERACION	DIRECCION PARTE MAS S.	DIRECCION PARTE MENOS S.

6). Direccionamiento Relativo. En éste direccionamiento se - suma al PC la dirección que se encuentra en el segundo byte de la instrucción mas dos.

El rango de la dirección varia de:

(PC + 2) - 128 D (PC + 2) + 127

donde:

PC = dirección del primer byte de la instrucción de ramificación (Branch).

D = dirección de destino de la instrucción de ramificación.

Cuando se quiere transferir el control fuera del rango de las instrucciones de ramificación puede hacerse usando - las instrucciones JMP (salto incondicional) o JSR (salto a - subrutina), que no usan el modo de direccionamiento relativo.

CODIGO DE OPERACION	DIRECCION RELATIVA

7). Direccionamiento Indiciado. En éste modo de direcciona-- miento se suma la dirección que se encuentra en el segundo - byte de la instrucción a los 8 bits menos significativos del registro de fndice. Si existe acarreo se suma a los bits más significativos. Estas instrucciones son de dos bytes. El registro de fndice no se modifica ya que la dirección número que se tiene se guarda en un registro de direcciones temporalmente, sin modificar el registro de fndice.

Solo son válidos operandos que varien de 0 - FF (hexadecimal), el valor del operando se suma al contenido del registro de fndice teniendose la siguiente dirección.

D = Valor numérico + X

donde:

X = contenido del registro de índice.

D = dirección numérica.

CODIGO DE OPERACION	DIRECCION (OPERANDO)

d).- Familia de Componentes del Sistema M 6800.

La familia de partes del sistema M 6800 se diseño teniendo como objetivo un número mínimo de componentes (Figura 2.14). La familia incluye un Microprocesador (MC 6800), memorias RAM (MCM 6810A), ROM (EPROM S6834) y circuitos de Entrada/Salida (ACIA MC 6850 y PIA MC 6820). Las componentes pueden ensamblarse en forma de bloques para construir una microcomputadora.

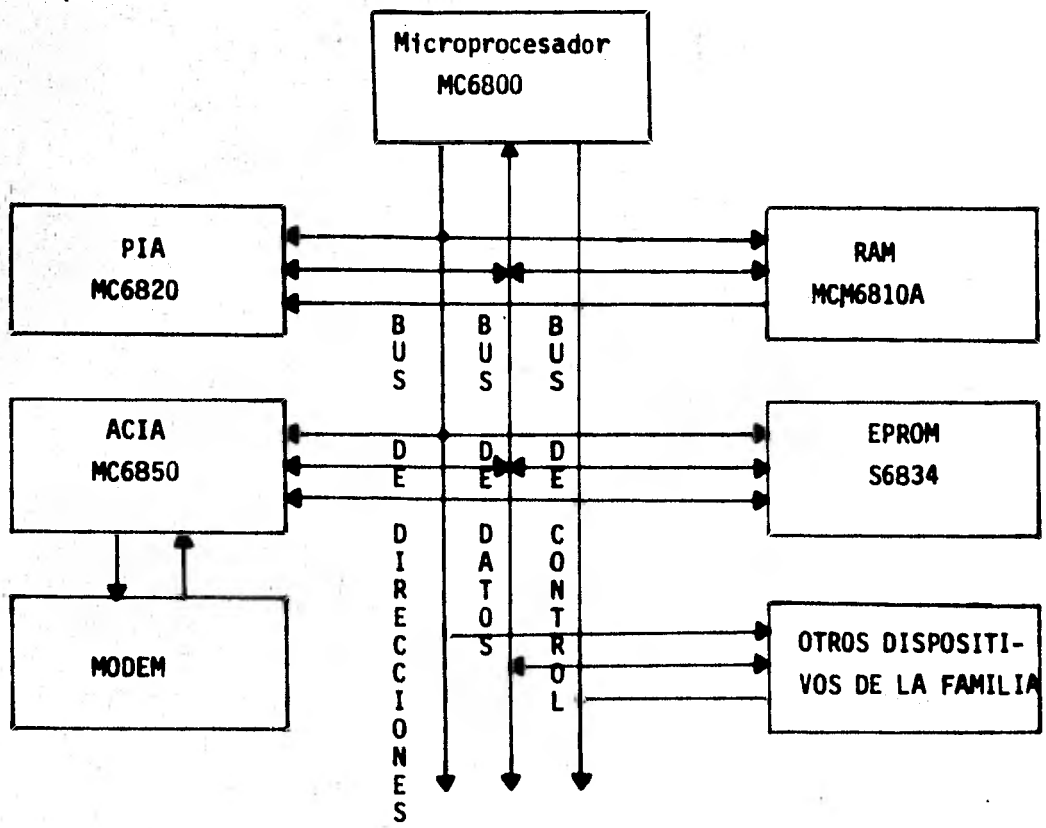


Figura 2.14. Diagrama de Bloques de la familia M 6800.

2.3.- Memoria (RAM, ROM).

a). Memoria RAM MCM 6810A. Esta memoria de acceso aleatorio es de tipo estático y no necesita de reloj o refrescamiento. Cuenta con 128 localidades (bytes) de 8 bits cada una.

En su fabricación se usa tecnología MOS de canal N con compuerta de silicio. Para su funcionamiento solo se necesita una fuente de voltaje. Es compatible con TTL y DTL, por lo cual no es necesaria ninguna interfase adicional para conectarse al microprocesador MC 6800.

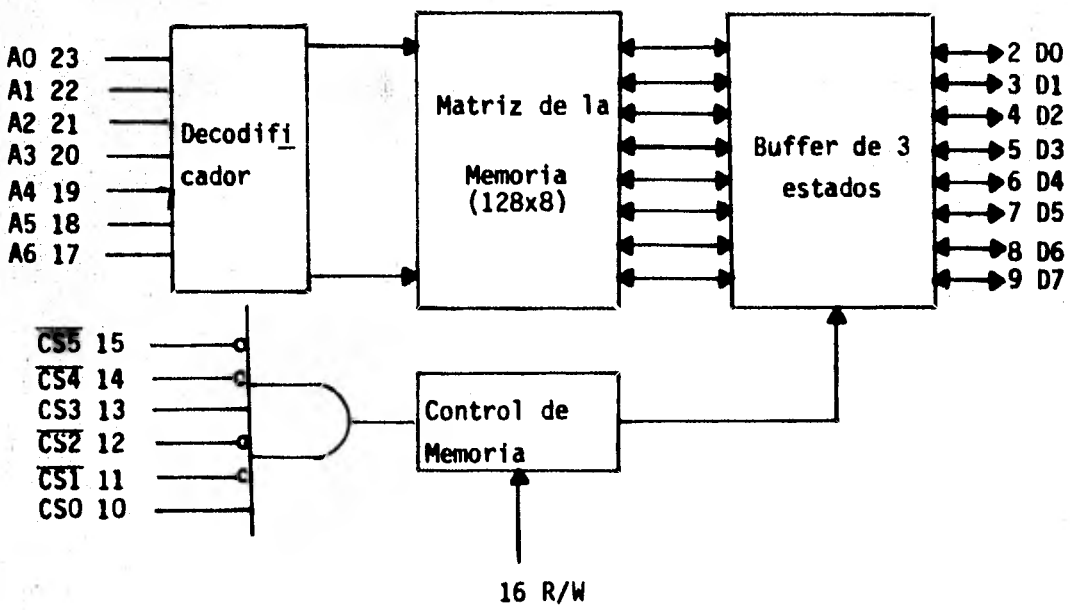


Figura 2.15. Diagrama de bloques de la memoria RAM MCM 6810A

La memoria RAM MCM 6810A tiene 7 líneas (A0 - A6) que permiten seleccionar una dirección particular en éste RAM.

El RAM cuenta con 6 líneas adicionales (Chip Select Inputs) que escogen un RAM particular entre varios circuitos -

integrados de la micro. Cuatro de éstas líneas ($\overline{CS1}$, $\overline{CS2}$, $\overline{CS4}$, $\overline{CS5}$) son activas en nivel cero y dos son activas en nivel 1 (CSO, CS3). En sistemas de tamaño pequeño y medio la decodificación que se tiene con éstas líneas es suficiente para distinguir todos los dispositivos del sistema.

R/W.- Esta línea de entrada se usa para controlar la dirección del flujo de datos. Cuando el estado de R/W es alto (ciclo de lectura del MPU), se encienden los drivers del bus de datos y puede leerse la localidad de memoria seleccionada. Cuando el estado de R/W es bajo el microprocesador escribe en la localidad de memoria seleccionada.

El tiempo de acceso máximo a memoria es de 1.0 us para el MCM 6810L y de 575 ns para el MCM 6810-1.

b). Memoria EPROM S6834.

La capacidad de almacenamiento de ésta memoria es de 512 bytes. Esta es una memoria estática que se puede borrar exponiendo la ventana transparente del circuito integrado a una fuente de luz ultravioleta durante un tiempo de 7 a 10 minutos.

Después de que el EPROM ha sido borrado el nivel de cada uno de los bits del S6834 es cero. Los datos se guardan programando selectivamente un nivel alto en cada uno de los bits de la localidad deseada. La línea de R/W se usa para seleccionar el modo de operación deseado. Cuando su estado es bajo se habilita la escritura del EPROM. Cuando se programa el EPROM se escribe un byte del código fuente y después de que se da la dirección del S6834 se aplica un pulso de

programación (V_p = -50 Vdc). El voltaje de programación es-- cribe los datos en el arreglo de la memoria (Figura 2.16). Este proceso se repite hasta que todos los 512 bytes del código fuente se han transferido al EPROM S6834. El tiempo de programación es menos de un minuto.

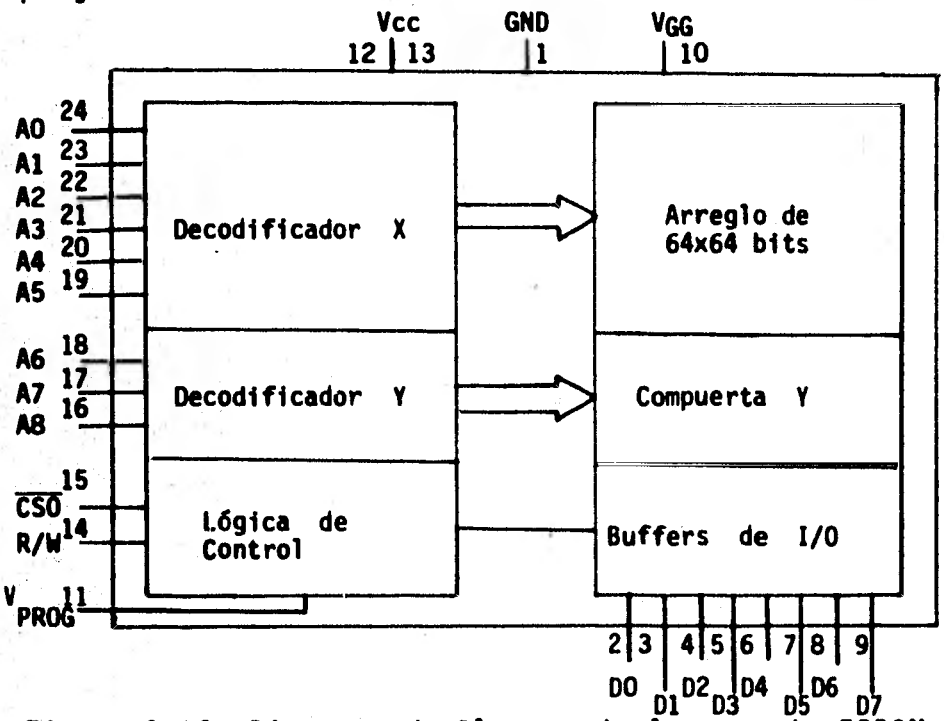


Figura 2.16. Diagrama de Bloques de la memoria EPROM S6834.

La cantidad de energía que se necesita para asegurar - que la memoria mantendra los datos puede definirse como una función del número de pulsos del programa (N) multiplicado - por el ancho del pulso del programa (t_{pw}) (N x t_{pw} 60 mseg). Por lo cual si el ancho del pulso es de 3 ms, se necesitan 20 pulsos de programa como mínimo; si t_{pw} es de 5 ms se necesitan 12 pulsos de programa como mínimo.

Cuando el EPROM se lee, la línea de R/W esta en su estado alto y la línea V_{prog} se conecta a Vcc.

El tiempo de acceso al S6834 es de 575 ms y su configuración es igual a la del PROM S6830 de 1K x 8 bits. Sus entradas y salidas son compatibles con TTL.

Algunas aplicaciones del EPROM son: Microprogramación, traducción de código, generación de caracteres, decodificación, etc.

2.4.- Interfases de entrada y salida (PIA, ACIA).

a). PIA (Peripheral Interface Adapter, MC6820). Esta interfase permite conectar equipo exterior al MC 6800 usando dos buses de 8 bits bidireccionales y 4 líneas de control.

Las líneas de interfase entre el PIA y el MPU son:

Bus de Datos Bidireccional (D0 - D7). Estas líneas permiten la transferencia de datos entre el MPU y el PIA. Los drivers del bus de datos son dispositivos que se encuentran generalmente en su estado de alta impedancia (OFF), excepto cuando el MPU realiza una operación de lectura del PIA (R/W en estado alto).

E (Enable). Esta señal de reloj que recibe el PIA se usa como referencia de tiempo de las otras señales. Normalmente se obtiene del reloj $\emptyset_2$ del MPU.

R/W. Controla la dirección de transferencia de datos entre el MPU y el PIA. Cuando su estado es bajo se habilitan los Buffers de entrada del PIA y el MPU realiza la transferencia de datos. Si el estado de R/W es alto y el PIA esta habilitado se transfieren datos de éste al bus de datos.

$\overline{\text{Reset}}$. Cuando esta línea pasa a su estado activo (bajo), los bits de todos los registros del PIA pasan a un estado lógico cero (bajo). PA0 - PA7, PB0 - PB7, CA2 y CB2 quedan establecidas como líneas de entrada y se deshabilitan las interrupciones. Después de un reset se configura el PIA en la forma deseada usando un programa de reinicio.

CS0, CS1, y $\overline{\text{CS2}}$ (Chip Selects). Estas entradas se usan para seleccionar el PIA. Para que esto ocurra CS0 y CS1 -

deben estar en su estado alto y $\overline{CS2}$ en su estado bajo. El PIA no se selecciona si alguna de éstas líneas esta en su estado inactivo.

RS0 y RS1 (Register Select). Estas dos líneas, junto con el bit 2 del registro de control A(B) permiten seleccionar uno de los registros del PIA (Figura 2.17).

$\overline{IRQA}$ e $\overline{IRQB}$ ($\overline{Interrupt\ Request}$). Cuando una de éstas líneas pasa a su estado activo (bajo) interrumpe al MPU. Cada línea tiene asociadas 2 banderas internas que pueden llevar a la línea IRQ a su estado activo. Estas banderas permiten que el MPU sea interrumpido por un dispositivo externo. Las banderas se apagan cuando el MPU lee el registro de datos correspondiente.

Líneas de Interfase Periférica.

El PIA permite recibir o transmitir información en paralelo, a través de dos Buses de datos periféricos de 8 bits bidireccionales y 4 líneas de control.

El PIA tiene 6 registros (Figura 2.17): Dos registros de Control (CRA, CRB), dos registros de recepción y transmisión de datos (ORA, ORB), y dos registros para controlar la dirección de los datos transmitidos en cada línea periférica (DDRA, DDRB).

PA0 - PA7. Cada una de éstas líneas puede programarse como entrada o salida. Para esto escribimos un "1" en el bit del registro de Dirección de Datos correspondiente a las líneas de salida, y un "0" para que sea una línea de entrada.

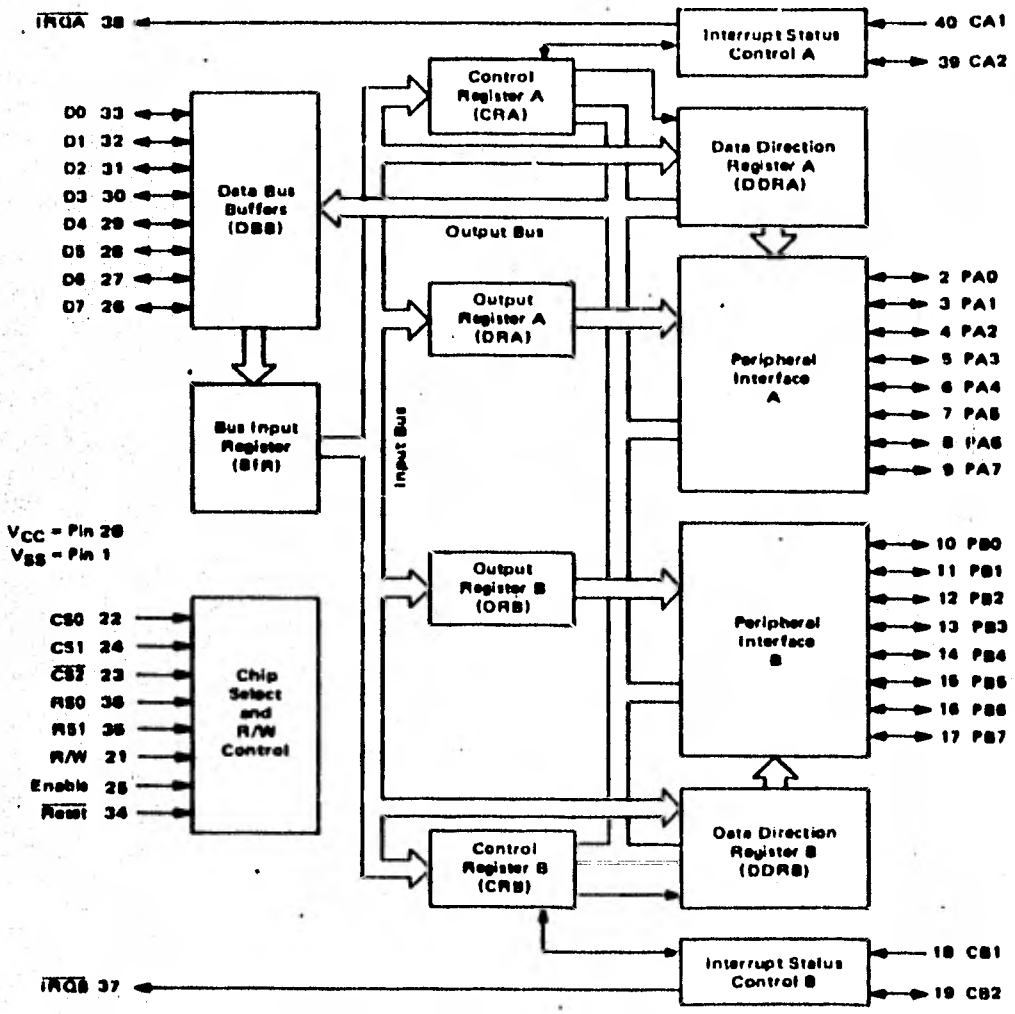


Figura 2.17. Diagrama de Bloques del PIA.

Cuando las líneas son programadas como entradas la resistencia interna es la de una carga TTL normal.

PB0 - PB7. Estas líneas pueden programarse como líneas de entrada o salida de una forma semejante a PA0 - PA7. Sin embargo cuando las líneas se programan como salidas, tienen la capacidad de poder pasar al tercer estado (alta impedancia). Son compatibles con lógica TTL.

CA1, CB1. Estas son solo líneas de entrada, que encienden las banderas de interrupción de los registros de control. El registro de control permite programar las transiciones activas.

CA2, CB2. Estas líneas se pueden programar como líneas de entrada de interrupciones o como líneas de salida para control. Son compatibles con lógica TTL. Su funcionamiento se programa con el registro de control correspondiente. Cuando CB2 se programa como salida puede usarse como fuente de corriente.

Controles Internos.

Los 6 registros del PIA ocupan 4 localidades de memoria. La selección de estos registros es controlada por las líneas de entrada RS0 y RS1 junto con el bit 2 del registro de control (tabla 1) que permite distinguir entre el registro de datos DDRA(B) y ORA(B).

RS1	RS0	Bit del Registro de Control CRA-2	CRB-2	Localidad Seleccionada
0	0	1	X	Registro periférico A
0	0	0	X	Registro de Dir. de Dat. A
0	1	X	X	Registro de Control A
1	0	X	1	Registro periférico B
1	0	X	0	Registro de Dir. de Dat. B
1	1	X	X	Registro de Control B

Tabla 1. Direccionamiento Interno del PIA.

Registro de Dirección de Datos (DDRA y DDRB). Estos 2 - registros permiten que el MPU controle la dirección de datos a través de cada línea periférica. Un "0" en un bit del re-- gistro de Dirección de Datos configura la línea periférica - como una línea de entrada, un "1" la configura como salida.

Registros de entrada y salida de datos (ORA, ORB). Es-- tos son registros de lectura y escritura que permiten el intercambio de información entre el MPU y los dispositivos periféricos.

Registros de Control (CRA y CRB). Estos registros permiten que el MPU controle la operación de las 4 líneas de control CA1, CA2, CB1 y CB2; habilite las líneas de interrup- - ción y registre el estado de las banderas de interrupción - (tabla 2). Los bits 0-5 son bits de lectura y escritura. Los bits 6 y 7 son bits solo de lectura que se modifican por interrupciones externas que se tienen en las líneas de control CA1, CA2, CB1 o CB2.

El bit 2 en cada uno de los registros de control (CRA-2 y CRB-2) permite seleccionar entre ORA(B) y DDRA(B), cuando se aplican las señales apropiadas en RS0 y RS1.

Los bits 6 y 7 ce cada uno de los registros de control (tabla 2) son bits que se encienden por transiciones activas de una de las líneas de control periférico declarada como - entrada. El MPU no puede encender ninguno de estos bits. Se apagan después de una operación de lectura de ORA (ORB).

7	6	5	4	3	2	1	0
IRQA1	IRQA2	Control	CA2		Acceso DDRA	Control	CA1

REGISTRO DE CONTROL A (CRA)

7	6	5	4	3	2	1	0
IRQB1	IRQB2	Control	CB2		Acceso DDRB	Control	CB1

Tabla 2. REGISTRO DE CONTROL B (CRB)

Control de las líneas de entrada CA1 y CB1. Los bits - "0" y "1" de los registros de control se usan para controlar las interrupciones de las líneas de entrada CA1 y CB1. El - bit "0" del registro de control A y B se usa para habilitar las señales de interrupción del MPU $\overline{IRQA}$ e $\overline{IRQB}$ respectiva-- mente. CRA-1 y CRB-1 determinan que tipo de transición activa la señal de interrupción de una de las líneas CA1 y CB1 - (tabla 3).

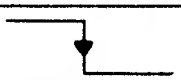
CRA-1 (CRB-1)	CRA-0 (CRB-0)	Entrada de interrupc. CA1(CB1)	Bandera de interrupc. CRA-7(CRB-7)	Solicitud de interrupc. del MPU $\overline{IRQA}$ ($\overline{IRQB}$)
0	0		Nivel uno en ↓ de CA1(CB1)	Inhibido
0	1		Nivel uno en ↓ de CA1(CB1)	Nivel cero cuando bit 7 es uno.
1	0		Nivel uno en ↑ de CA1(CB1)	Inhibido
1	1		Nivel uno en ↑ de CA1(CB1)	Nivel cero cuando bit 7 es uno.

Tabla 3. Control de Interrupciones de CA1 y CB1.

Control de las líneas de entrada CA2 y CB2. Los bits 3, 4 y 5 de los 2 registros de control se usan para controlar las líneas CA2 y CB2. Estos bits determinan si la línea de - entrada puede interrumpir o si se tiene una señal de control.

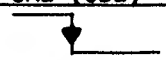
CRA-5	CRA-4	CRA-3	Entrada de interrupc. CA2 (CB2)	Bandera de interrupc. CRA-6(CRB-6)	Solicitud de interrupc. del MPU $\overline{IRQA}$($\overline{IRQB}$)
0	0	0		Nivel uno en ↓de CA2(CB2)	Inhibido
0	0	1		Nivel uno en ↓de CA2(CB2)	Nivel cero cuando CRA-6 (CRB-6) es 1
0	1	0		Nivel uno en ↑de CA2(CB2)	Inhibido
0	1	1		Nivel uno en ↑de CA2(CB2)	Nivel cero cuando CRA-6 (CRB-6) es 1

Tabla 4. Control de Interrupciones de CA2 y CB2.

Cuando el bit CRA-5 (CRB-5) es "0" la línea CA2 (CB2) es una línea de entrada que puede interumpir a CA1 (CB1). Cuando el bit CRA-5 (CRB-5) es "1", la línea CA2 (CB2) es una señal de salida que permite controlar la transferencia de datos a los dispositivos externos.

b). ACIA (Asynchronous Communication Interface Adapter, MC 6850). Esta interfase proporciona el formato de datos y control de errores para que el MC 6800 pueda recibir y transmitir datos asincrónos en serie. El ACIA aparece como dos localidades de memoria direccionables en el Bus del sistema. Internamente esta formada por dos registros de escritura y dos de lectura (Figura 2.18). La configuración del ACIA se programa durante la inicialización del sistema. Para preparar esta configuración es necesario dar un Reset, lo cual se consigue encendiendo los bits "0" y "1" del registro de Control, posteriormente se programan los bits "5" y "6" del mismo registro para definir el estado de $\overline{RTS}$. El reset impide transiciones de salida erróneas y una vez dado se puede programar

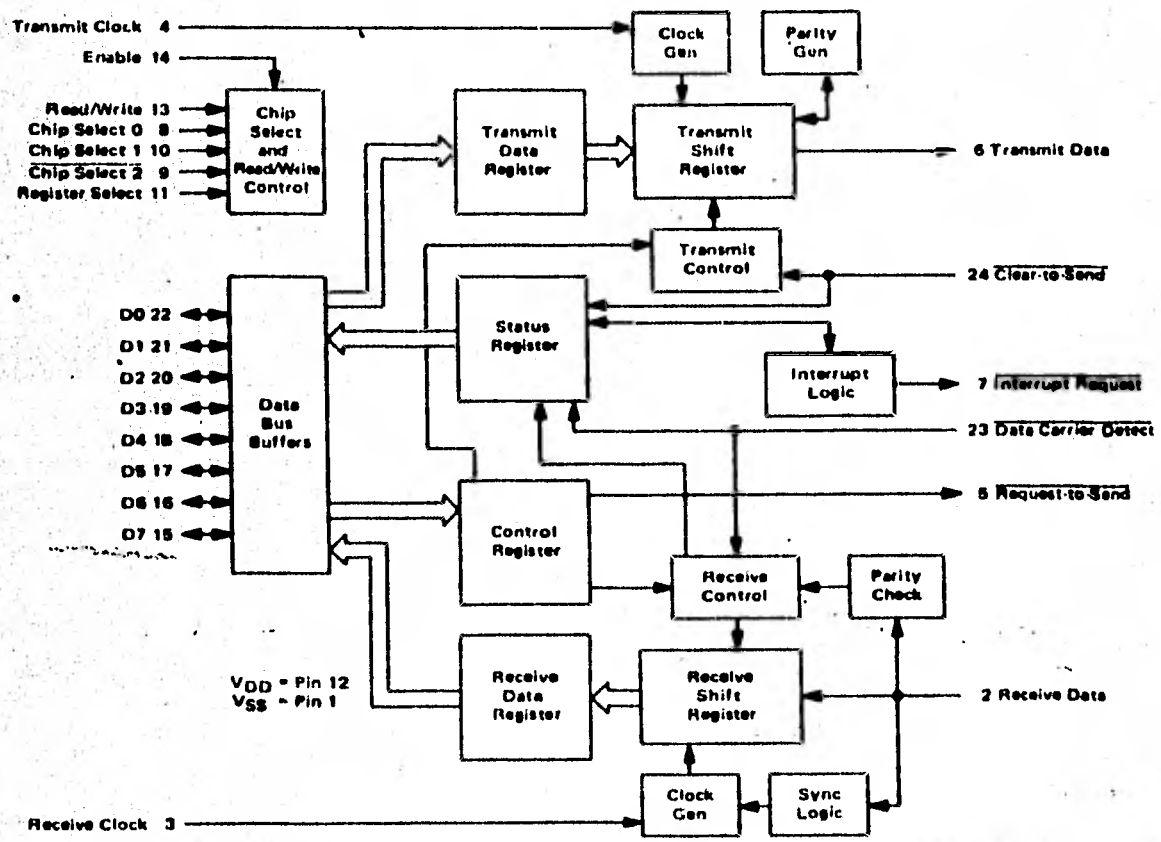


Figura 2.18. Diagrama de Bloques del ACIA.

el registro de control para escoger la frecuencia del reloj, longitud de palabra, el número de bits de stop, paridad (par, impar, ninguna), etc.

Líneas de Interfase entre el ACIA y el MPU.

D0 - D7. Las 8 líneas de datos bidireccionales permiten la transferencia de datos entre el ACIA y el MPU. Los drivers de salida de estas líneas son dispositivos de 3 estados que se encuentran en estado de alta impedancia (OFF), excepto - cuando el MPU realiza una lectura del ACIA.

E (Enable). Esta es una línea de entrada de alta impe-- dancia compatible con TTL. Habilita los Buffers de entrada/ salida del 8us de datos. Normalmente es una señal que se toma del reloj $\emptyset_2$ del MC 6800.

R/W (Read Write). Es una línea de entrada de alta impedancia, compatible con TTL. Se usa para controlar la direc-- ción del flujo de datos a través de la interfase de entrada/ salida del 8us de datos. Cuando el nivel de R/W es alto (ciclo de lectura del MPU) se encienden los crivers de salida y se lee el registro seleccionado. Cuando su nivel es bajo se apagan los drivers de salida y el MPU escribe en el registro seleccionado. La línea de R/W permite seleccionar los registros de escritura o lectura del ACIA (tabla 5).

RS	R/W	Registro
0	0	Control
0	1	Estado
1	1	Recepción de Datos
1	0	Transmisión de Datos

Tabla 5 Direccionamiento Interno del ACIA.

RS (Register Select). Esta es una línea de entrada compatible con TTL. Un nivel alto en esta línea permite seleccionar los registros de Transmisión/recepción de datos y un nivel bajo selecciona los registros de Control/estado (tabla 5). Junto con esta señal se usa la línea de R/W para seleccionar el registro de lectura o escritura de cada pareja de registros.

CS0, CS1, $\overline{CS2}$ (Chip Select). Estas 3 líneas de entrada son de alta impedancia y compatibles con TTL. Se usan para direccionar el ACIA; que ocurre cuando el nivel de CS0, CS1 es alto y el de $\overline{CS2}$ es bajo.

$\overline{IRQ}$ ($\overline{Interrupt\ Request}$). Es una línea de salida compatible con TTL; es activa en nivel bajo y se usa para interrumpir al MPU. Permanece en estado activo mientras la causa de interrupción esta presente y el bit de interrupción (7) del registro de estado esta encendido.

La sección de transmisión del ACIA genera una interrupción si se selecciona (CR5.$\overline{CR6}$) y el estado del bit 2 del registro de estado TDRE (Transmit Data Register Empty) es alto. La interrupción termina cuando se escriben datos en el Registro de Transmisión de Datos. Las interrupciones del transmisor pueden enmascararse y deshabilitarse usando los bits CR5 y CR6 del registro de control. La pérdida de $\overline{CTS}$ ($\overline{Clear\ to\ Send}$) inhibe el bit TDRE del registro de estado.

La sección de recepción del ACIA genera una interrupción cuando el bit 7 del Registo de Control se enciende y el estado del bit "0" (RDRF) del Registro de Estado es alto.

La interrupción termina cuando se lee el registro de Recep-- ción de Datos. Las interrupciones producidas por la condi- - ción de error por pérdida de uno o más caracteres de la cadena de datos (Overrun) o pérdida de señal portadora termina - cuando se lee el registro de estado, o se da un Reset al - ACIA.

Entradas de Reloj.

Se tienen 2 entradas de reloj compatibles con TTL que - determinan la frecuencia de transmisión de los datos recibidos y transmitidos.

T x CLK (Transmit Clock). El reloj recibido determina - la frecuencia de los datos transmitidos. La transmisión de - datos se inicia en la transición negativa del reloj.

R x CLK (Receive Clock). El reloj recibido se usa para la sincronización de los datos. El receptor muestrea los datos en la transición positiva del reloj.

Líneas de entrada y salida en serie.

Recepción de Datos (R x Data). Esta es una línea de entrada de alta impedancia compatible con TTL. A través de ésta se reciben los datos que tienen un formato serial. La frecuencia del reloj puede dividirse por Software entre 16 ó 64 veces la original. La detección de datos se inicia en el - frente positivo de la transición de Marca o espacio del bit de Start. Mientras se recibe un carácter se checa su paridad, y en caso de error aparece una indicación en el registro de Estado. En una secuencia de recepción se lee el registro de estado para determinar si se ha recibido un caráter. Cuando

el registro de recepción de datos esta lleno, se coloca el - carácter en el Bus de 8 bits del ACIA. Cuando se recibe un - comando de lectura, el receptor quita el bit de paridad y - transfiere los datos al MPU. El receptor tiene un Buffer doble, asi puede leerse un carácter del registro de datos mientras se recibe otro en el Registro de Corrimiento. La secuencia continua hasta que se reciben todos los carácteres.

Transmisión de Datos (T x Data). Una secuencia de transmisión consiste en leer el registro de estado del ACIA como resultado de una interrupción o de una secuencia de encuesta. Si la lectura del registro de estado indica que el registro de Transmisión esta vacío puede escribirse un carácter, el - cual se transmite a un registro de corrimiento donde se serializa y se transmite precedido por un bit de Start (inicio) seguido por uno ó dos bits de stop. Opcionalmente puede agregarse la paridad del carácter entre el último bit del dato y el primer bit de stop. Desués de escribir el primer carácter en el registro de Datos puede leerse el registro de estado - que indica que el registro de transmisión esta vacío, por lo que puede cargarse otro carácter para su transmisión, aún - cuando el primer carácter esta aún en proceso de transmisión. Esto es posible debido al doble buffereado. Cuando termina - la transmisión del primer carácter se transfiere automáticamente el segundo carácter al registro de corrimiento. La secuencia continua hasta que se transmiten todos los caracteres.

Controles del Modem.

El ACIA tiene 3 funciones que permiten el control limitado de un periférico o modem.

1. $\overline{CTS}$ ($\overline{Clear\ to\ Send}$). Esta es una línea de entrada de alta impedancia compatible con TTL. Proporciona control automático para terminar un enlace de comunicación inhibiendo el bit TDRE del registro de estado. La dirección de esta señal es - del Modem al MPU e indica si el Modem o equipo de comunica-- ción de datos esta listo para transmitir.

2. $\overline{RTS}$ ($\overline{Request\ to\ Send}$). Esta salida permite al MPU controlar un equipo periférico o modem. Refleja el estado de los - bits CR5 y CR6 del registro de control. La dirección de esta señal es del MPU al Modem o equipo periférico y se usa para condicionar al modem para transmisión de datos. Una transi-- ción de OFF a ON indica al modem que puede iniciar la transmisión. Una transición de ON a OFF indica al modem que termine la transmisión de datos.

3. $\overline{DCD}$ ($\overline{Data\ Carrier\ Detect}$). Esta es una línea de alta impedancia compatible con TTL. Proporciona control automático en un enlace de comunicación. Cuando el nivel es alto inhibe e inicializa la sección de recepción del ACIA. Una transición de nivel bajo a nivel alto inicia una interrupción del MPU - que indica pérdida de portadora. La dirección de esta señal es del modem al MPU. Cuando el modem recibe una señal del - nivel apropiado, el estado de la línea esta activo (ON). Cuando el modem no recibe señal o su nivel no es apropiado para demodulación, el estado de la línea es inactivo (OFF).

Registros del ACIA.

Registro de Transmisión de Datos (TDR). Cuando se direc ciona el ACIA durante una transición negativa de E (Enable) y RS·$\overline{R/W}$. La escritura de datos hace que el bit 1 de (TDRE) pase a su estado bajo y la transferencia del dato hace que - TDRE pase a su estado alto que equivale a un registro de - transmisión vacío.

Registro de Recepción de Datos (RDR). Después de que el registro de corrimiento recibe un carácter completo lo trans fiere automáticamente al Registro de Recepción de Datos. Esto enciende el bit RDRF (Receive Data Register Full) del registro de estado. Este bit se apaga cuando en un ciclo de - lectura se selecciona el Registro de Recepción de datos del ACIA que ocurre cuando RS y R/W estan en su estado alto.

Número de la línea del Bus de Datos	RS·$\overline{R/W}$ Registro de Transmisión de Datos (solo escr.)	RS·R/W Registro de Recepción de Datos (solo lect.)	$\overline{RS}$·$\overline{R/W}$ Registro de Control (solo escr.)	$\overline{RS}$·R/W Registro de Estado (solo lect.)
0	Bit de dat. 0	Bit de dat. 0	Selector de división 1 (CR0)	Registro de dat. de rec. lleno(RDRF)
1	Bit de dat. 1	Bit de dat. 1	Selector de división 2 (CR1)	Registro de Trans. de dat. lleno(TDRE)
2	Bit de dat. 2	Bit de dat. 2	Selector de palabra 1 (CR2)	Detector de portadora (DCD)
3	Bit de dat. 3	Bit de dat. 3	Selector de palabra 2	Clear to send ($\overline{CTS}$)
4	Bit de dat. 4	Bit de dat. 4	Selector de palabra 3	Framing Error (FE)
5	Bit de dat. 5	Bit de dat. 5	Control de transmisión 1	Receiver Over run (OVRN)
6	Bit de dat. 6	Bit de dat. 6	Control de transmisión 2	Parity Error (PE)
7	Bit de dat. 7	Bit de dat. 7	Habilita interrupc. recibidas (CR7)	Interrupt Request(IRQ)

Tabla 6. Registros del ACIA.

Cuando el bit RDRF esta encendido se inhibe la transferencia de datos del registro de corrimiento al RDR cuyo contenido - se conserva.

Registro de Control. Este registro se selecciona cuando el nivel de RS y R/W es bajo. Controla el funcionamiento del receptor, transmisor, interrupciones y salida $\overline{RTS}$.

Bits CR0 y CR1 (Counter Divide Select Bits). Estos bits determinan la cantidad entre la que hay que dividir la fre-- cuencia original del transmisor y receptor del ACIA. Se usan también para dar un reset al ACIA, que limpia el registro de estado (Excepto los bits $\overline{CTS}$ y $\overline{DCD}$) e inicializa el receptor y transmisor.

El reset al ACIA se da encendiendo los bits CR0 y CR1 - (tabla 7). Después de esto puede seleccionarse la cantidad - entre la cual se divide la frecuencia del reloj.

CR1	CR0	Función
0	0	÷ 1
0	1	÷ 16
1	0	÷ 64
1	1	Reset

Tabla 7. Bits para seleccionar la frecuencia del Reloj.

Bits CR2, CR3 y CR4 (Bits para seleccionar la palabra). Estos bits se usan para seleccionar la longitud de la pala-- bra, paridad y el número de bits de stop. Su formato es el - siguiente.

CR4	CR3	CR2	Función
0	0	0	7 bits + paridad par + 2 bits de stop
0	0	1	7 bits + paridad impar + 2 bits de stop
0	1	0	7 bits + paridad par + 1 bit de stop
0	1	1	7 bits + paridad impar + 1 bit de stop
1	0	0	8 bits + 2 bits de stop
1	0	1	8 bits + 1 bit de stop
1	1	0	8 bits + paridad par + 1 bit de stop
1	1	1	8 bits + paridad impar + 1 bit de stop

Bits CR5 y CR6 (Control de Transmisión). Estos bits permiten controlar las interrupciones del Registro de Transmisión de datos cuando se encuentra vacío, la salida de $\overline{RTS}$ ($\overline{Request\ to\ Send}$) y la transmisión de un nivel de Break (espacio). Su formato es el siguiente:

CR6	CR5	Función
0	0	RTS = bajo, transmisión de interrupc. deshabilitadas
0	1	RTS = bajo, transmisión de interrupc. habilitadas
1	0	RTS = alto, transmisión de interrupc. deshabilitadas
1	1	RTS = bajo, transmite un nivel de Break en la salida de transmisión de datos. Transmisión de interrupciones deshabilitadas.

Bit CR7 (Bit para habilitar interrupciones del Receptor). Cuando el bit 7 del registro de control se enciende, se habilitan las interrupciones: Registro de recepción de Datos lleno, Overrun y una transición de nivel bajo a alto que se presenta en la línea $\overline{DCD}$ ($\overline{Data\ Carrier\ Detect}$).

Registro de Estado. El MPU conoce el estado del ACIA después de leer este registro, el cual se selecciona cuando RS es bajo y R/W es alto.

Bit 0 (RDRF, Receive Data Register Full). Cuando el bit se enciende indica que el dato recibido se ha transferido al registro de Recepción de Datos. El bit se apaga cuando el MPU lee el RDR, se da un reset al ACIA o el estado de $\overline{DCD}$ es alto.

Bit 1 (TDRE, Transmit Data Register Empty). Cuando el bit se enciende indica que el contenido del registro de transmisión de datos se ha enviado y que puede escribirse un nuevo dato. Cuando esta apagado indica que el registro de

transmisión esta lleno.

Bit 2 ($\overline{DCD}$, $\overline{Data\ Carrier\ Detect}$). Cuando el bit se enciende indica que no hay portadora en la entrada $\overline{DCD}$. Cuando esto ocurre y el bit 7 del registro de control esta encendido se genera una interrupción. El bit se apaga cuando se lee el registro de Estado y luego el registro de datos, o se da un reset al ACIA. Cuando el bit 7 del registro de control esta apagado, el bit 2 ($\overline{DCD}$) sigue la entrada ($\overline{DCD}$).

Bit 3 ($\overline{CTS}$, $\overline{Clear\ to\ Send}$). Cuando el bit se enciende se inhibe el bit 1 del registro de estado. El bit apagado indica que hay un $\overline{CTS}$ del modem.

Bit 4 (FE, Framing Error). Indica que el carácter recibido no esta limitado correctamente por los bits de start y stop. Esto indica un error de sincronización, una transmisión defectuosa o una condición de Break. El bit se enciende y se apaga durante la transferencia de datos.

Bit 5 (OVRN, Receiver Overrun). Indica que se perdieron uno o más carácteres de la cadena de datos. Esto significa que algunos caracteres recibidos no fueron leidos del registro RDR. El bit se apaga cuando se lee el registro de datos o se da un reset al ACIA.

Bit 6 (PE, Parity Error). Indica que el número de unos del carácter no concuerda con la paridad par o impar seleccionada. Se define paridad impar cuando el número total de unos es impar. El bit permanece encendido mientras esta el carácter en el RDR.

Si no se selecciona paridad se inhibe el transmisor generador de paridad y el receptor checador de paridad.

Bit 7 (IRQ, Interrupt Request). Este bit indica el estado de la salida $\overline{IRQ}$. Se enciende cuando el bit IRQ esta en su estado bajo. Se apaga cuando se lee el registro de Recepción de Datos o se escribe en el registro de Transmisión de Datos.

CAPITULO III

MONITOR

Un monitor es un programa que se encuentra en memoria - ROM o PROM y que permite inicializar el sistema para que - acepte comandos especiffcos de la terminal y ejecuta estos - comandos. Cuando se enciende la microcomputadora el monitor indica que esta en el programa de control e imprime un asterisco solicitando información. El usuario puede contestar - con una letra de un conjunto de caracteres; cada una de és-- tas especifica una función del monitor y tiene un significado mnemónico.

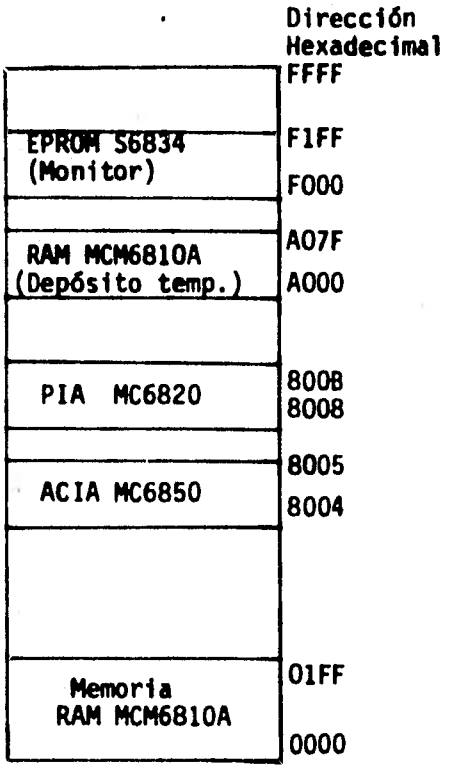


Figura 3.1. Mapa de la Memoria

Un monitor proporciona un ejemplo de programación de la computadora que se tiene y además es un recurso adicional de software que ayuda a cargar, ejecutar y corregir los propios programas.

A continuación se estudia un monitor que permite la comunicación entre el MPU y un teletipo o terminal. El programa monitor se encuentra en un EPROM S6834 de 512x8 bits y usa las direcciones de memoria F000 a F1FF (que se traslapan con FE00 a FFFF).

El monitor utilizado en esta microcomputadora necesita un ACIA (Asynchronous Communications Interface Adapter) como interfase paralelo-serie para comunicarse con una terminal externa (pantalla o teletipo). El registro de estado y el registro de control del ACIA ocupan la localidad de memoria 8004 y el registro de datos la 8005 (Figura 3.1).

El monitor necesita también memoria RAM (MC 6810) como depósito temporal de información. En éste caso se han asignado para tal propósito las localidades de memoria A000 - A07F (Figura 3.1).

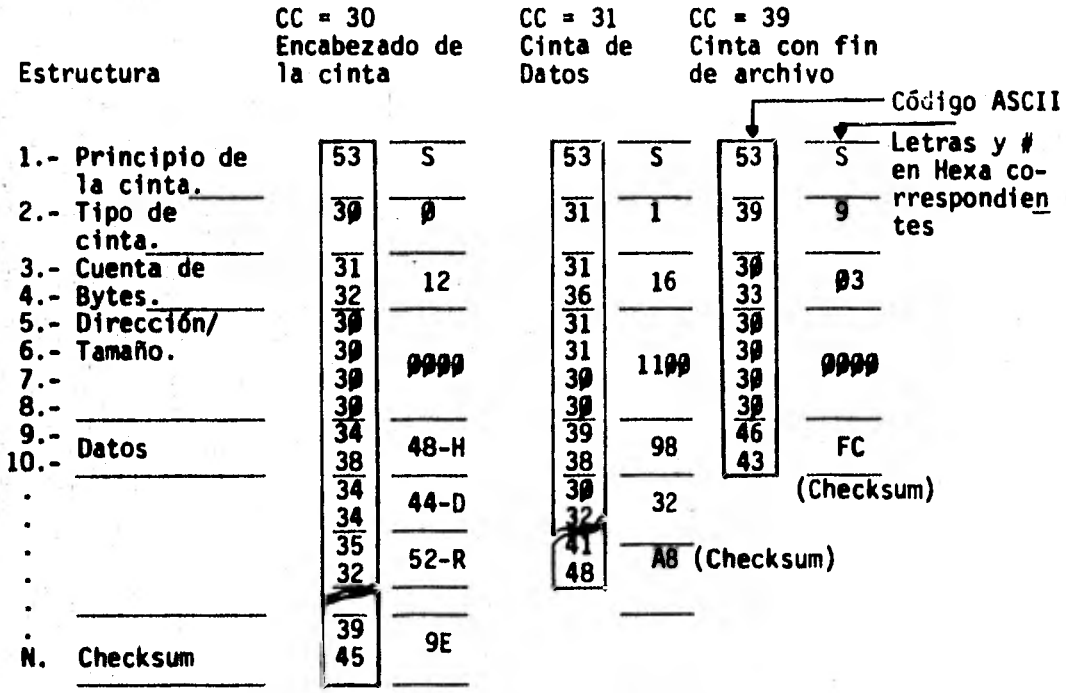
a). Operación del Monitor.

Función de Reset.- Es necesario oprimir el botón de "Reset" cuando se enciende la computadora o el monitor pierde el control del programa. Esto permite que el monitor recupere el control del programa. La terminal responde con CR, LF y *, que indica que el programa monitor esta listo para recibir un comando.

El monitor permite que el usuario realice las siguien-- tes funciones: L, M, P, R y G que son llamadas desde la consola tecleando la letra correspondiente.

A continuación se describe cada una de éstas funciones:

L.- Función para cargar en memoria la información contenida en cinta con formato binario (Figura 3.2).



El checksum es el complemento a uno de la suma de bytes de 8 bits. Suma a partir de la "Cuenta de Bytes" despreciando en cada paso el acarreo después del bit Más Significativo.

Figura 3.2. Formato de una cinta de Papel.

Cuando se carga una cinta se ignoran los caracteres que se encuentran antes del principio de cinta (Start of record, S).

Para terminar la función L es necesario teclear S9, que indica al monitor el fin de archivo, o dar un reset. Si en - el momento que se lee la cinta se detecta un error de Checksum se detiene la lectura de la cinta y se imprime un signo de interrogación. El operador tiene ahora las siguientes opciones:

1).- Presionar el botón de Reset y abortar la carga de la información en memoria. El MPU regresa al programa de control y la terminal imprime CR, LF y * .

2).- Leer la cinta nuevamente desde el principio.

3).- Ignorar el error de Checksum y teclear L de nuevo. Esta última opción no es recomendable, ya que no es posible determinar en que parte de memoria se carga el dato.

Ejemplo:

```
*L
S1130000020202020286AA02023F86FC881CCC7B
S10400108635
S9
*
```

M.- Función para examinar y cambiar el contenido de una localidad de memoria.

Para cambiar el contenido de una localidad de memoria - se sigue la siguiente secuencia.

1.- Examinar el contenido de la localidad de memoria seleccionada (teclear la dirección).

2.- Si es necesario, cambiar el contenido de la localidad de memoria.

3.- Regresar el contenido a la localidad de memoria (cerrar la localidad de memoria).

El monitor imprime el contenido de una localidad abierta para su exámen. Una localidad cerrada es aquella cuyo contenido no esta disponible para su exámen.

Al principio de la función M el monitor se encuentra en su programa de control y el último dato impreso es un asterisco. Después de esto se escribe el carácter M para abrir una localidad de memoria, el monitor agrega enseguida un espacio.

Posteriormente se escriben 4 carácteres con formato hexadecimal que indican la localidad de memoria que se abrira, el monitor imprime esta dirección en la siguiente línea, junto con su contenido en hexadecimal. El contenido de esta localidad abierta puede cambiarse dando un espacio seguido de dos carácteres hexadecimales (nuevo contenido). El monitor abre la siguiente localidad de memoria. Para terminar la secuencia y cerrar una localidad sin ningún cambio debe darse un espacio seguido de un CR.

En el ejemplo de abajo, se cambia el contenido de la localidad 1000 por FC y se abre la localidad de memoria 1001 que contiene un 0A.

```
*M 1000
1000 A5 FC
1001 0A
```

P.- Función para imprimir el contenido de memoria y perforar una cinta con esta información.

La cinta tiene el formato que se muestra en la Figura - 3.2. Para perforar una cinta es necesario dar la dirección - inicial del programa usando la función M. La dirección ini-- cial se escribe en las localidades de memoria A03F, A040 (BEGA), y la dirección final en las localidades A041 y A042 - (ENDA).

La dirección inicial es la del primer byte que se perfora y la dirección final es la dirección del último. Todas - las direcciones y datos estan en hexadecimal.

Ejemplo:

```
* M A03F

* A03F FF 00
* A040 0C 00    (BEGA)
* A041 EA 00
* A042 0F 10    (ENDA)
* A043 A0 (SP)(CR)
* P
S1130000020202020202038EAA02023F86FC881CCC7B
S10400108635
* S9
*
```

Después de dar el carácter P el monitor escribe el contenido de memoria y perfora la cinta con esta información. Terminada esta operación imprime un asterisco y regresa al - programa de control.

R.- Función para mostrar el contenido de los registros del - Microprocesador.

Después de que se muestra el contenido de los registros del Microprocesador, se puede cambiar su contenido usando la función M como se ve en el siguiente ejemplo:

Ejemplo:

```
    CC  B  A   X   PC    S
* R C1 0B 20 0016 000D A035
* M A035
* A035 01   SP (Stack Pointer)
* A036 C1   CC (Condition Codes Register)
* A037 0B   B  (Accumulator B)
* A038 20   A  (Accumulator A)
* A039 00   XH (Index Register, Higher order)
* A03A 16   XL (Index Register, Lower order)
* A03B 00   PCH(Program Counter, High)
* A03C 0D   PCL(Program Counter, Low)
*
```

El exámen de las localidades de memoria termina dando un espacio y CR.

El apuntador de Stack apunta a la primera localidad del Stack que no se usa, la siguiente localidad contiene el registro de códigos de condición (CC), seguida por el acumulador B, el acumulador A, el registro de fndice (X) y el contador de programa (PC). Estos registros se guardan en el Stack del usuario.

G.- Función para ejecutar el programa del usuario.

Esta función inicia la ejecución del programa a partir de la dirección que se encuentra en el contador de programa en el Stack, por lo cual antes de usarla es necesario cargar la dirección inicial del programa en el contador de programa del Stack que esta en las direcciones A045 (MS byte) y A046 (LS byte).

El MC 6800 ejecuta el programa del usuario hasta que:

1).- El Microprocesador encuentra una instrucción de Wait, después de lo cual espera una interrupción enmascarable o no enmascarable.

2).- El Microprocesador encuentra una instrucción SWI (Software Interrupt). El Microprocesador guarda el contenido de los registros en el Stack y va al programa de control. El monitor imprime el contenido de los registros que se guardaban en el Stack.

3).- Se da un "Reset" al Microprocesador que hace que el monitor vuelva al programa de control.

b) Atención de Programas de Servicio de Interrupciones.

$\overline{IRQ}$.- Interrupciones Enmascarables.

Cuando la línea $\overline{IRQ}$ pasa a su estado activo y las interrupciones enmascarables se encuentran habilitadas (I = 0), el Microprocesador va a la rutina de servicio cuya dirección se encuentra en las localidades A03D (MS byte) y A03E (LS byte).

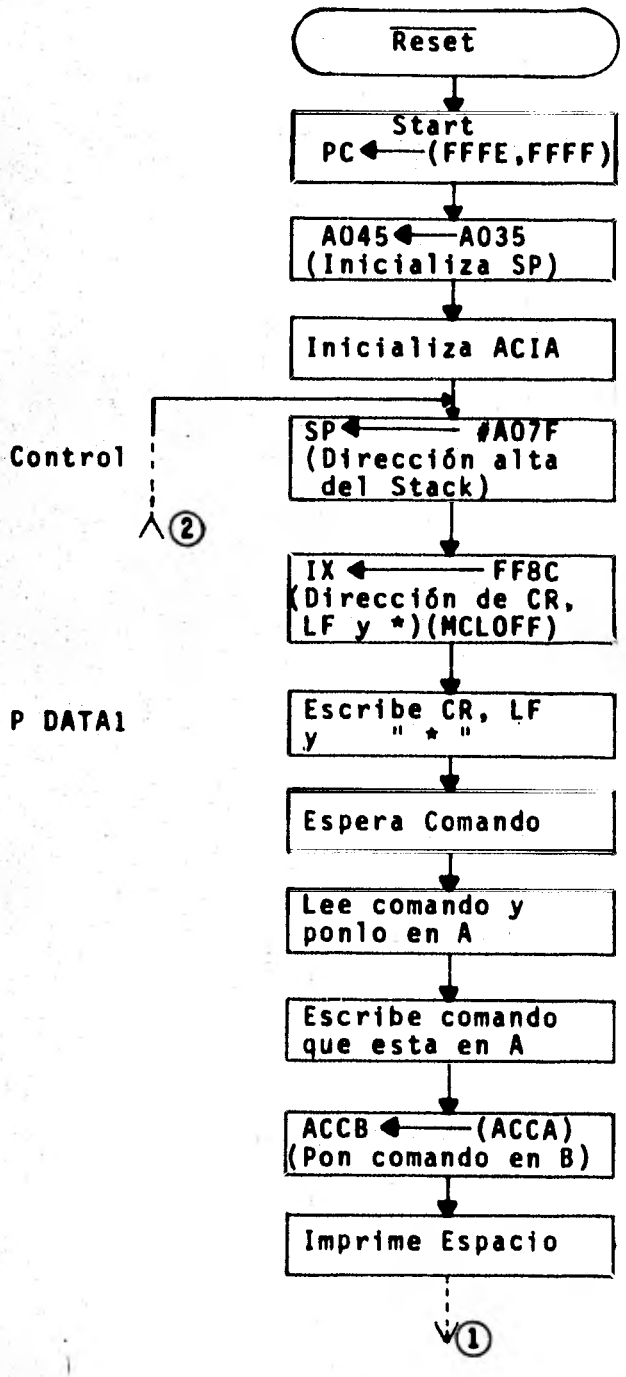
El contenido de estas localidades se transfiere al registro de índice y usando direccionamiento indiciado se va a tales localidades.

$\overline{NMI}$.- Interrupciones no Enmascarables.

Cuando se presenta esta interrupción se guardan en el Stack los registros del usuario. El monitor busca la dirección de la rutina de servicio en las localidades A043 (MS byte) y A044 (LS byte).

El monitor solo acepta caracteres hexadecimaels y los caracteres de control L, M, P, R, G, espacio y CR. Si se recibe un carácter distinto, el monitor lo ignora y pregunta de nuevo por el comando.

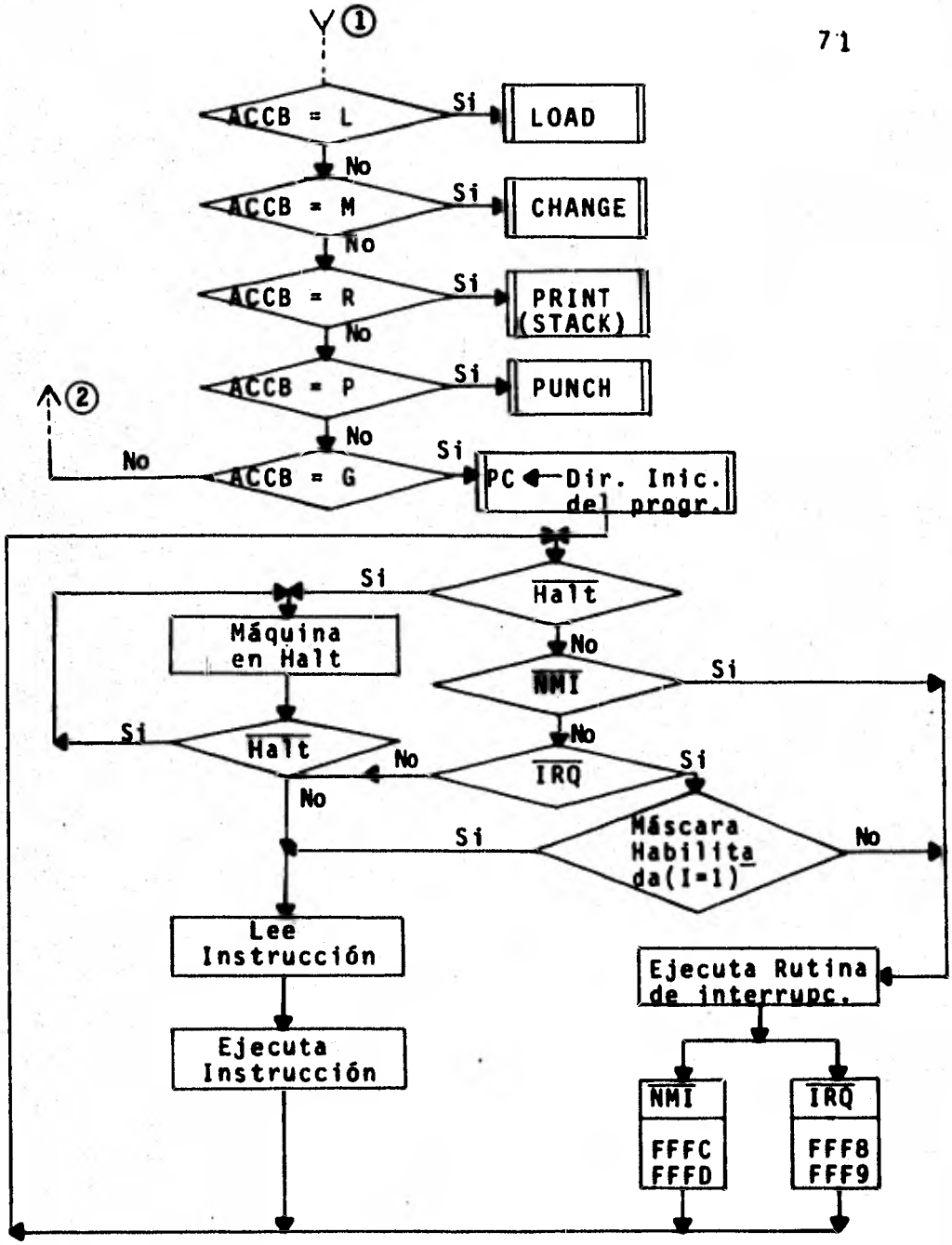
c) Diagrama de Bloques del Monitor



Secuencia que inicializa el sistema después de que se enciende o se da un reset.

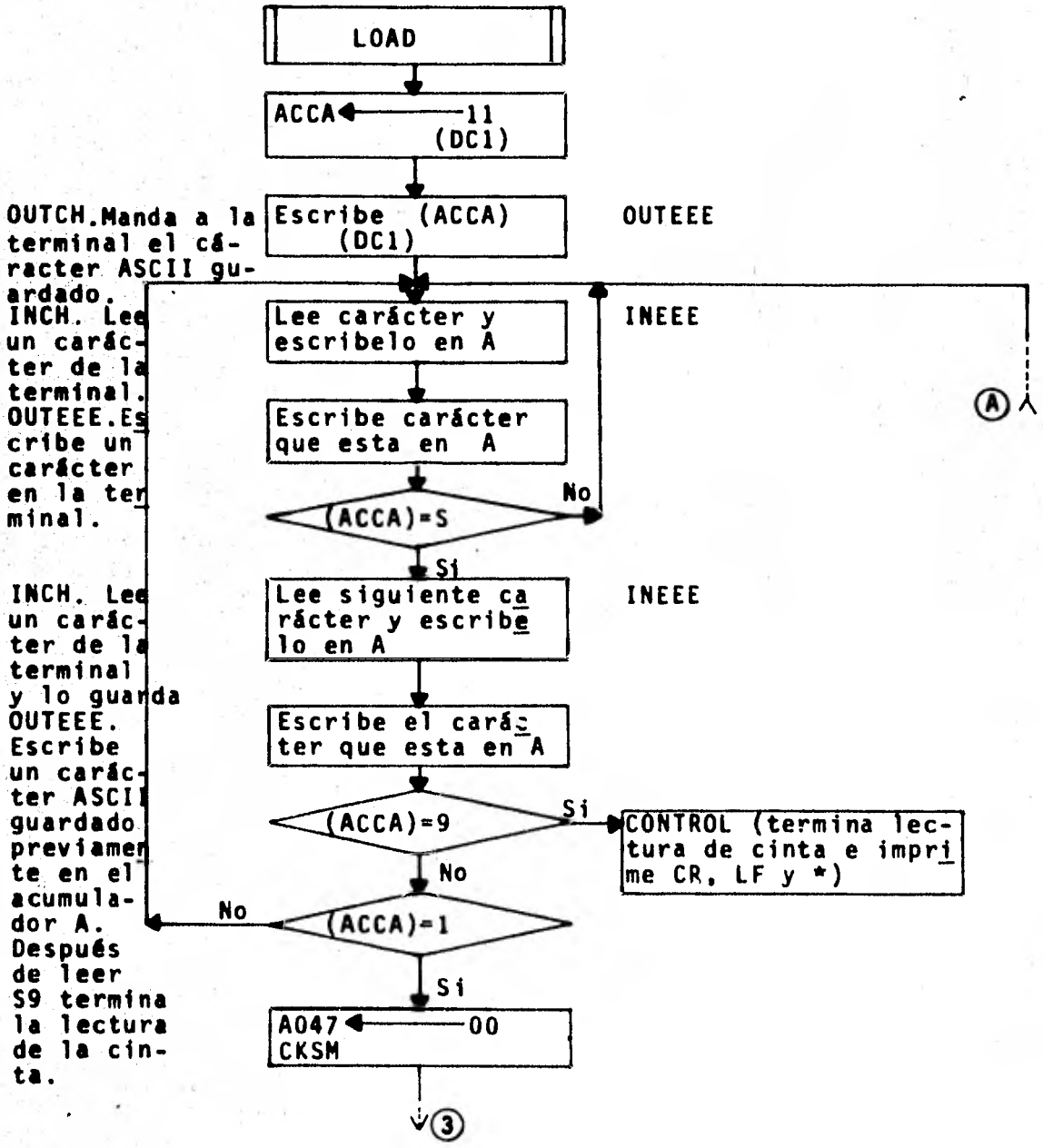
El monitor imprime un *, que indica que esta listo para recibir un comando.

Escribe una cadena de texto hasta encontrar un carácter de fin de archivo (04).


①
ACCB = L
Si
LOAD
No
ACCB = M
Si
CHANGE
No
ACCB = R
Si
PRINT (STACK)
No
ACCB = P
Si
PUNCH
No
②
No
ACCB = G
Si
PC ← Dir. Inic. del progr.
Halt
Si
Máquina en Halt
No
NMI
Si
Halt
Si
No
IRQ
Si
No
Máscara Habilitada(I=1)
Si
No
Lee Instrucción
Ejecuta Instrucción
Ejecuta Rutina de interrupc.
NMI
FFFC
FFFD
IRQ
FFF8
FFF9

c). Diagrama de Flujo de las Funciones del Monitor

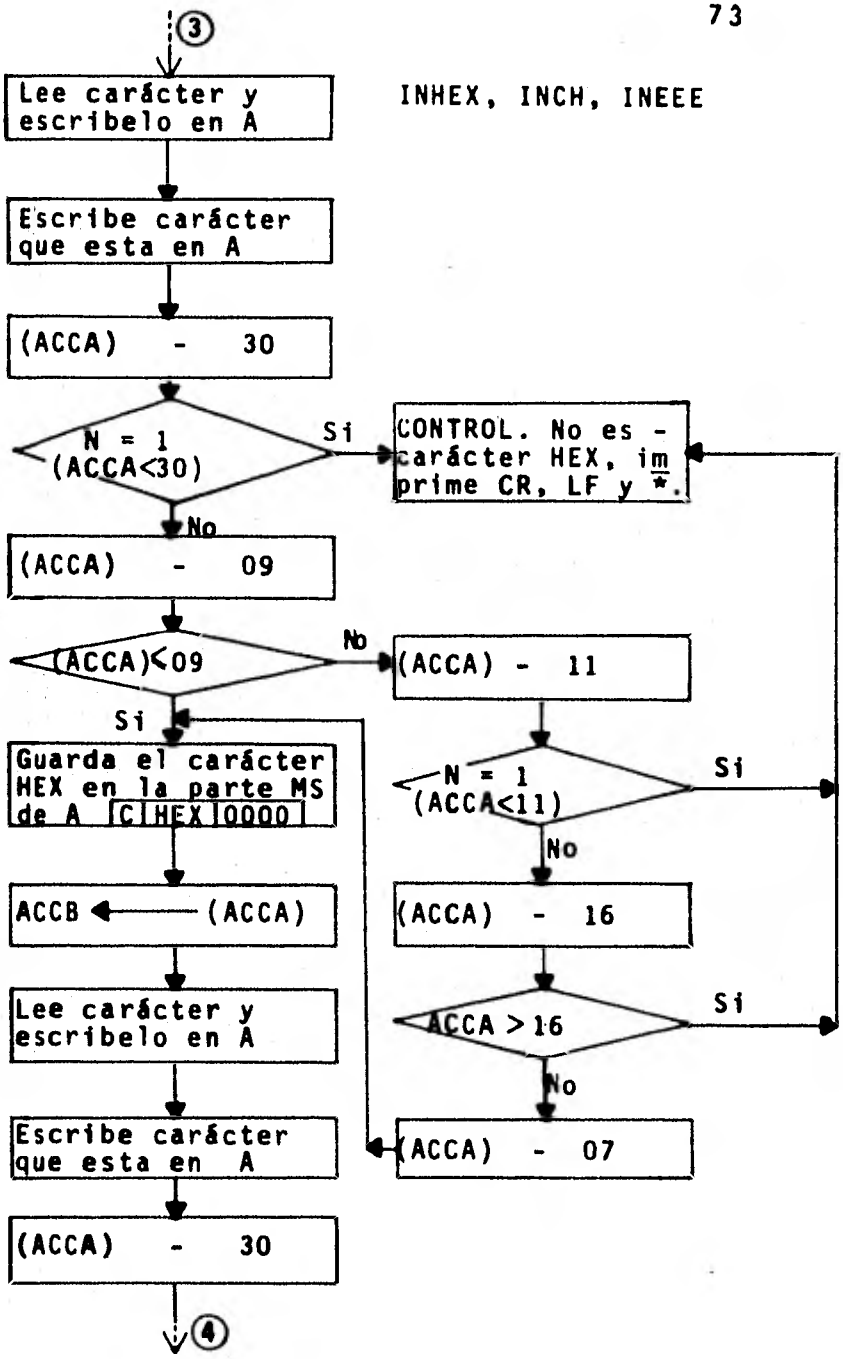
L.- Este comando permite cargar en memoria la información - contenida en la cinta.

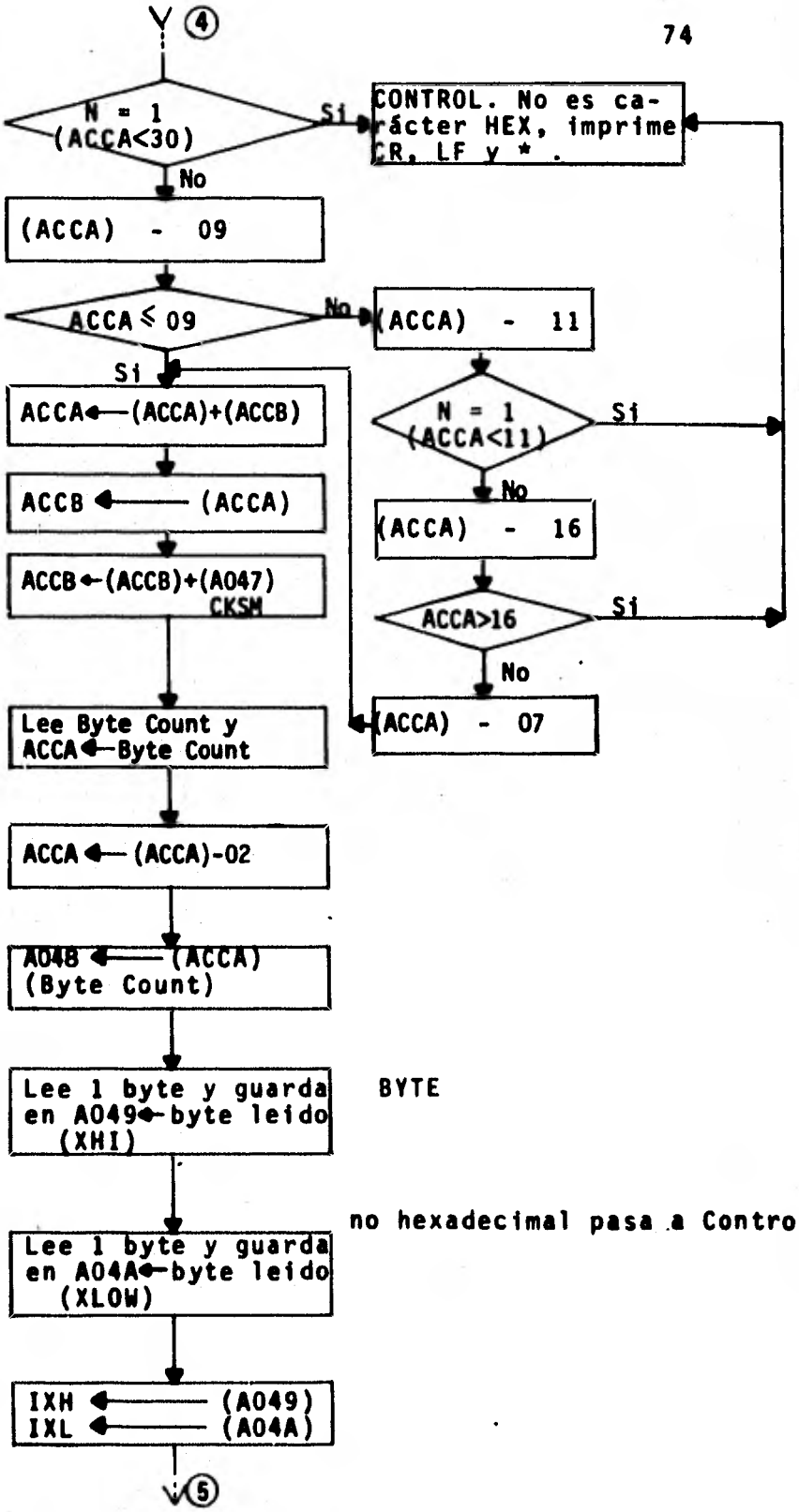


BYTE. Lee dos carácteres - hexadecimales de la termi - nal. Si no es carácter hexa decimal pasa a Control.

INHEX. Lee un carácter hexa- decimal de la terminal. Si - no recibe un - carácter hexa- decimal pasa a Control.

INHEX, INCH, INEEE





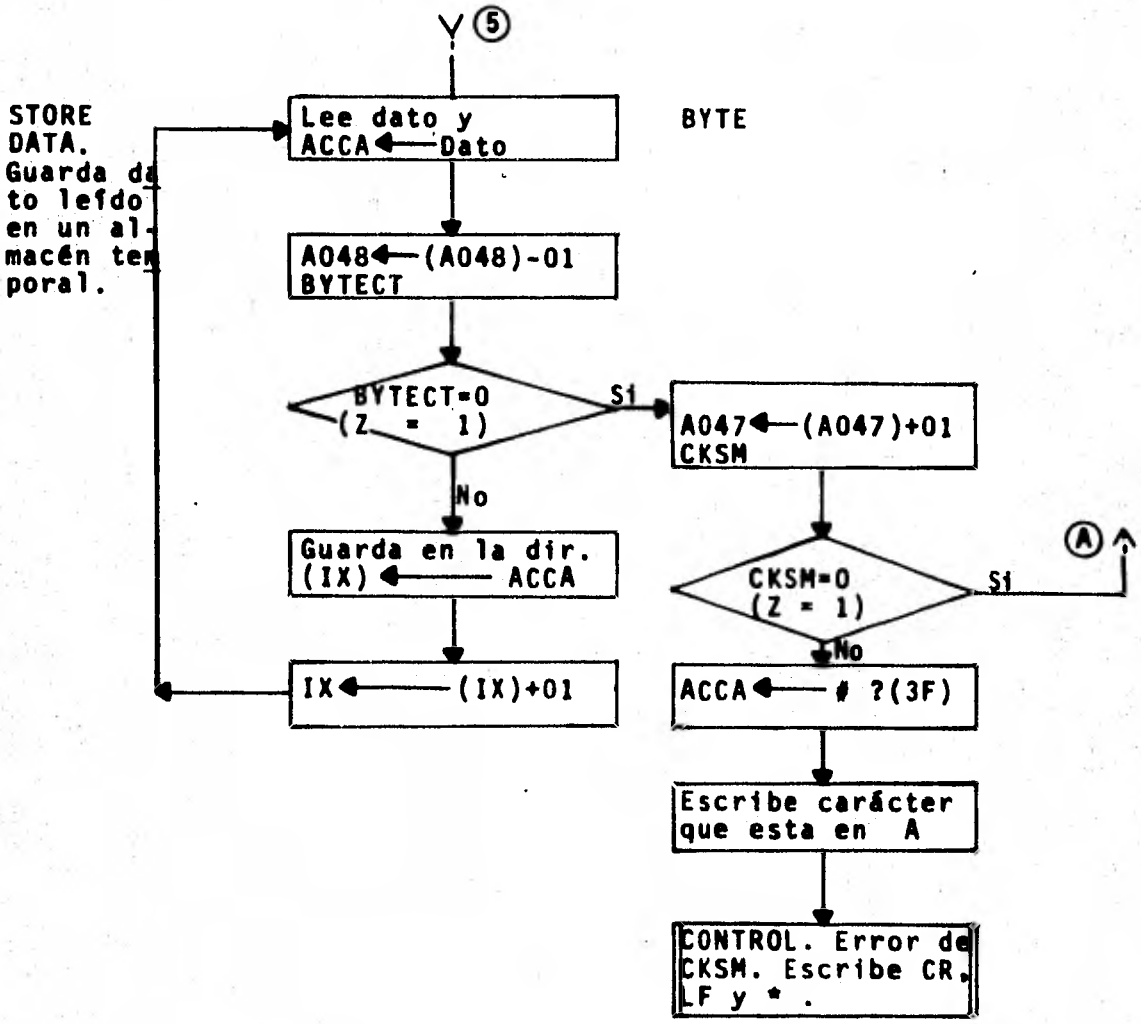
BYTE. Lee dos carácteres - hexadecimales de la termi-- nal. Si no re cibe ninguno pasa a Control

BUILD ADRESS. Lee 4 números hexadecimales de la termi-- nal. Si recibe 1 carácter

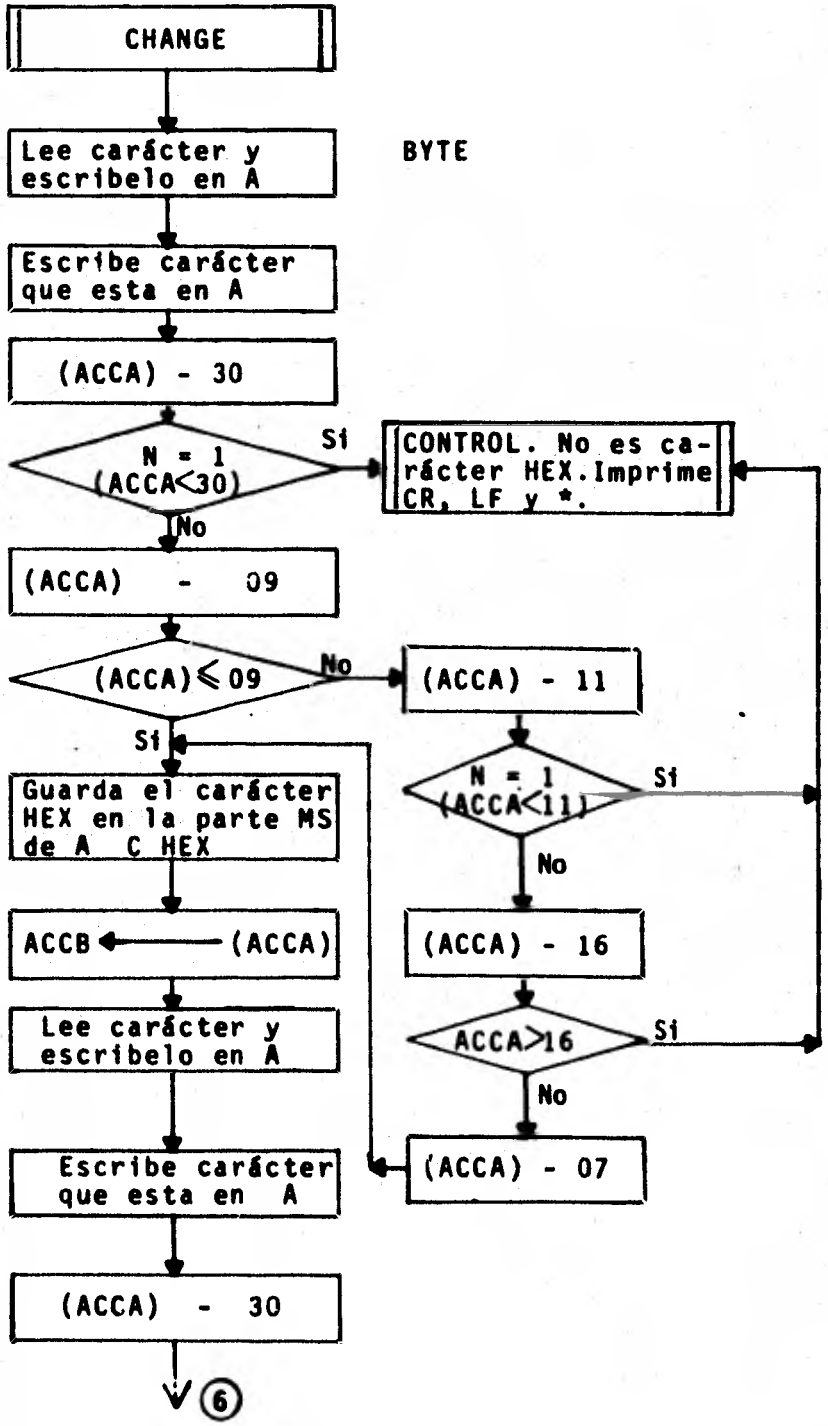
BYTE. Lee dos carácteres - hexadecimales de la terminal Si no recibe - ninguno pasa a Control.

BYTE

no hexadecimal pasa a Control


STORE
DATA.
Guarda da
to leído
en un al-
macén tem
poral.
Lee dato y
ACCA ← Dato
BYTE
A048 ← (A048)-01
BYTECT
BYTECT=0
(Z = 1)
Si
No
A047 ← (A047)+01
CKSM
Guarda en la dir.
(IX) ← ACCA
CKSM=0
(Z = 1)
Si
A
No
IX ← (IX)+01
ACCA ← # ?(3F)
Escribe carácter
que esta en A
CONTROL. Error de
CKSM. Escribe CR,
LF y * .

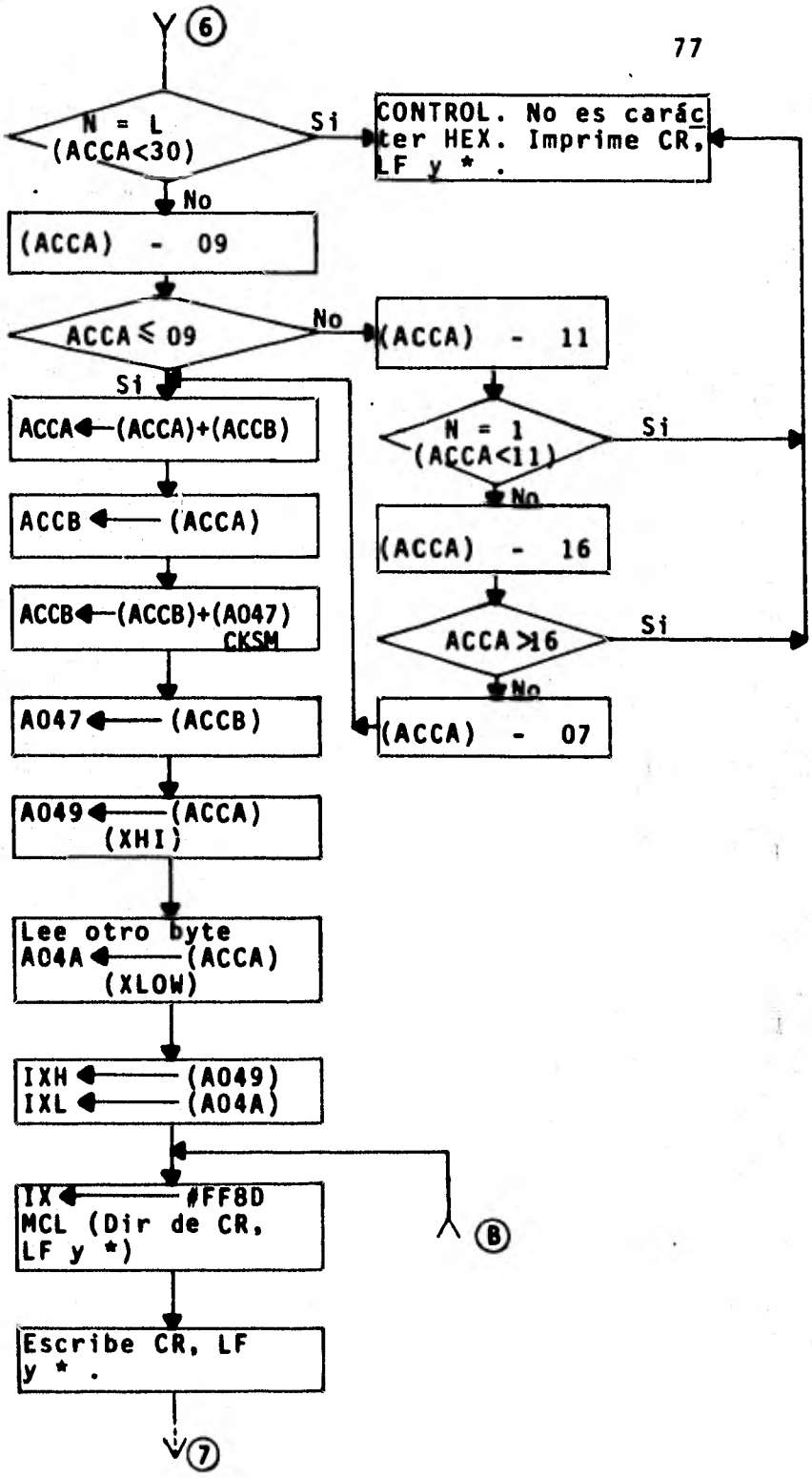
M.- Permite cambiar el contenido de una localidad de memoria



BUILD ADRESS. Lee 4 números hexadecimales de la terminal. Si recibe un carácter no - hexadecimal - pasa a Control

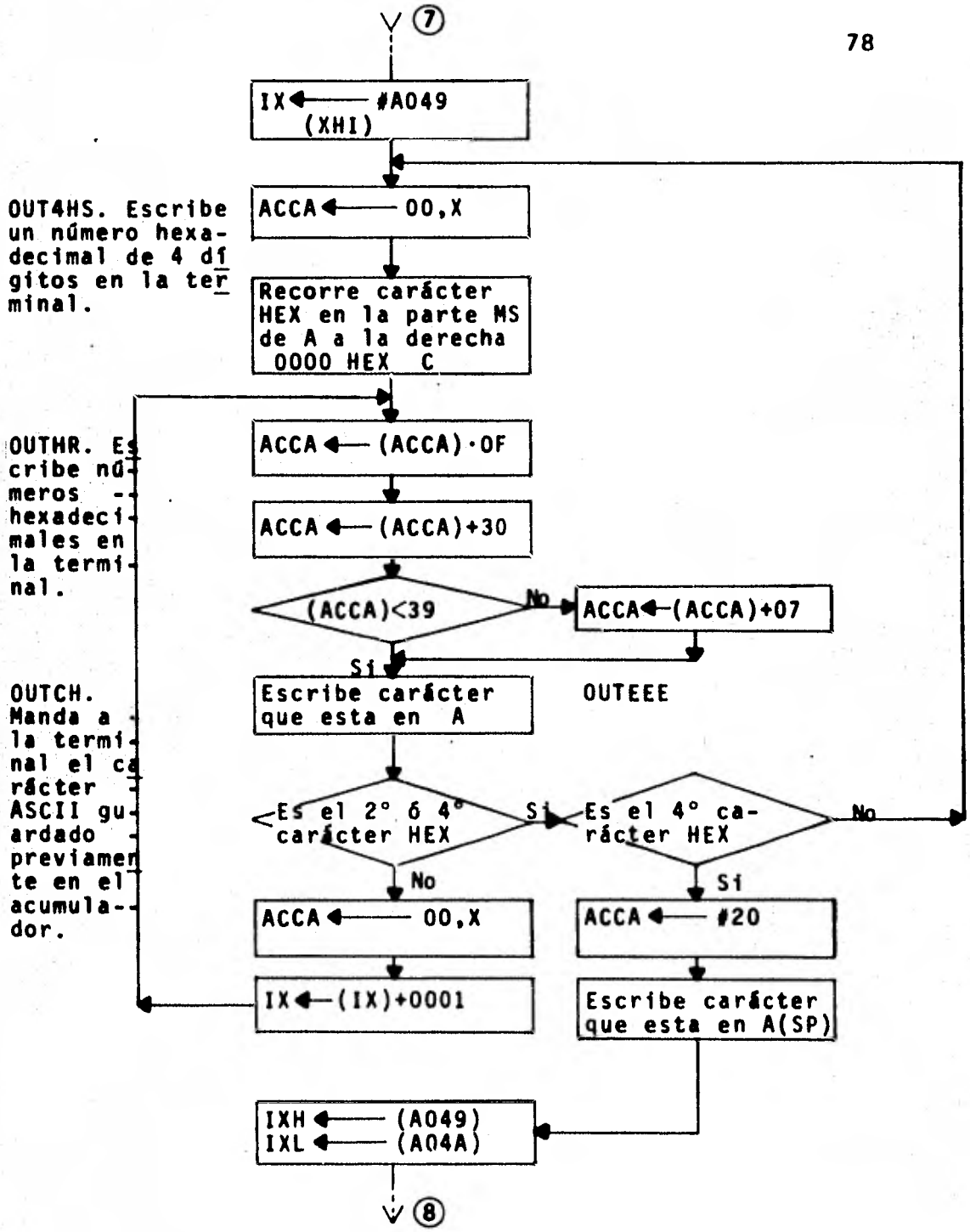
BYTE

INHEX. Lee un carácter hexadecimal de la terminal. Si - no pasa a Cont. OUTEEE. Escribe un carácter - ASCII guardado previamente en el acumulador A

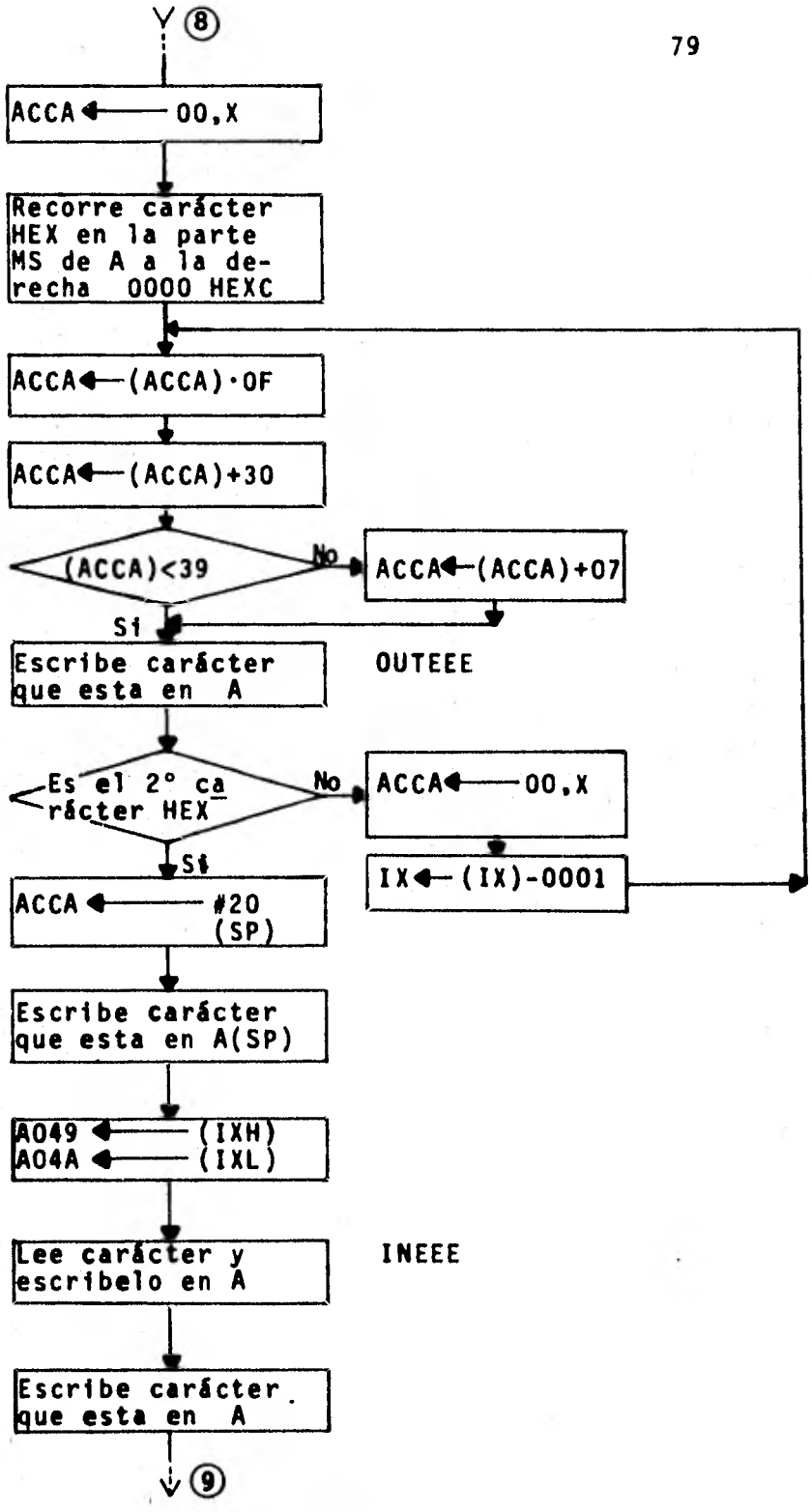


CHA51. Manda un CR, LF y * a la terminal.

PDATA 1. Escribe una cadena de texto hasta encontrar un carácter de fin de archivo.


⑦
IX ← #A049
(XHI)
OUT4HS. Escribe un número hexadecimal de 4 dígitos en la terminal.
ACCA ← 00,X
Recorre carácter HEX en la parte MS de A a la derecha
0000 HEX C
OUTHR. Escribe números hexadecimales en la terminal.
ACCA ← (ACCA)·0F
ACCA ← (ACCA)+30
(ACCA)<39
No
ACCA ← (ACCA)+07
Si
OUTCH. Manda a la terminal el carácter ASCII guardado previamente en el acumulador.
Escribe carácter que esta en A
OUTEEE
Es el 2° ó 4° carácter HEX
Si
Es el 4° carácter HEX
No
No
Si
ACCA ← 00,X
ACCA ← #20
IX ← (IX)+0001
Escribe carácter que esta en A(SP)
IXH ← (A049)
IXL ← (A04A)
⑧

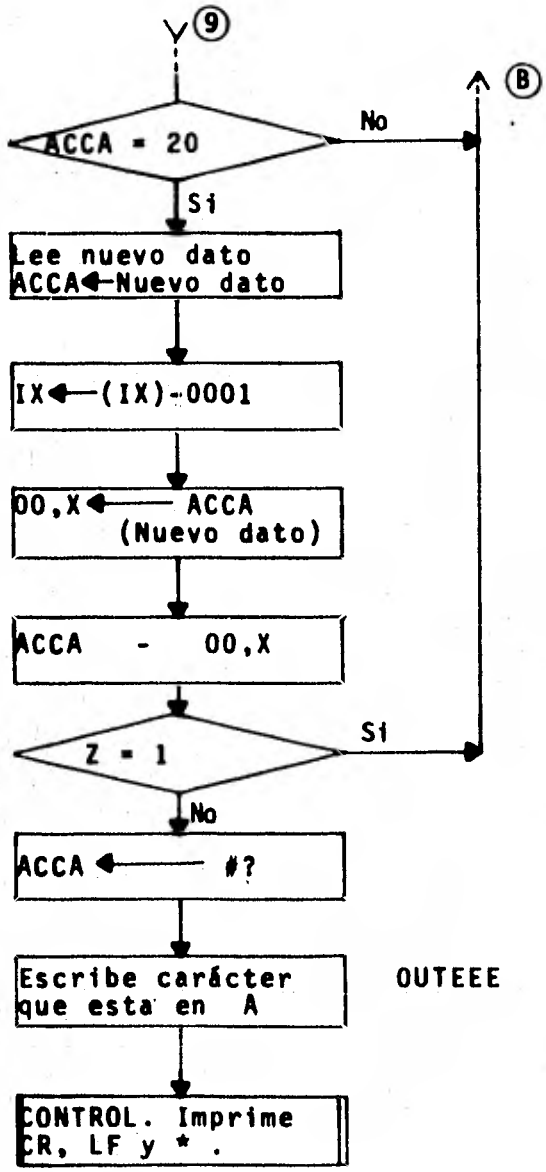
OUT2H. Escribe 2 carácteres - hexadecimales en la terminal En IX se guarda la direc- - ción inicial.



OUTCH. Manda a la terminal un carácter ASCII guardado pre-- viamente en el acumulador.

INCH. Lee un - carácter de la terminal y lo guarda en el - acumulador.

OUTEEE. Escribe en la terminal un carácter - ASCII guardado previamente en el acumulador.

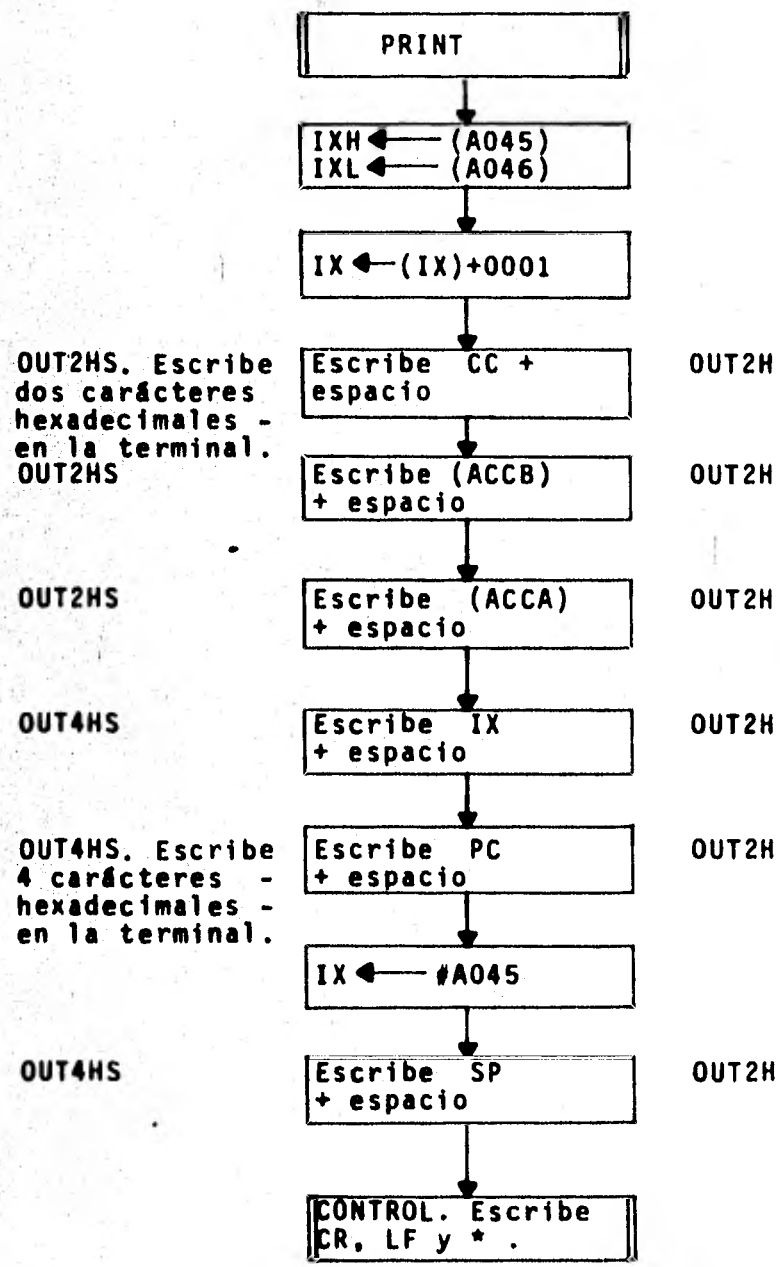


BYTE. Lee dos carácteres - hexadecimales de la termi-- nal. Si no re cibe ninguno pasa a Con- - trol.

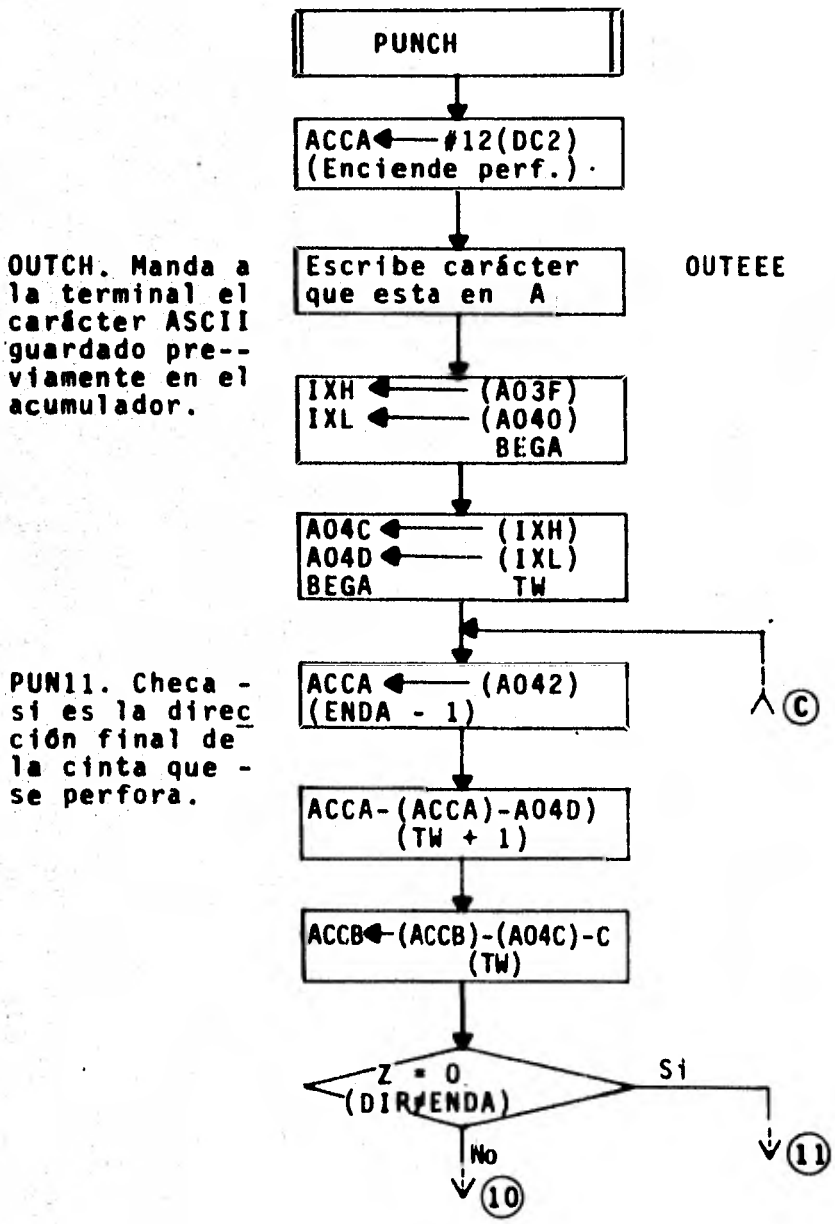
OUTCH. Manda a la terminal el carácter ASCII guardado pre-- viamente en el acumulador.

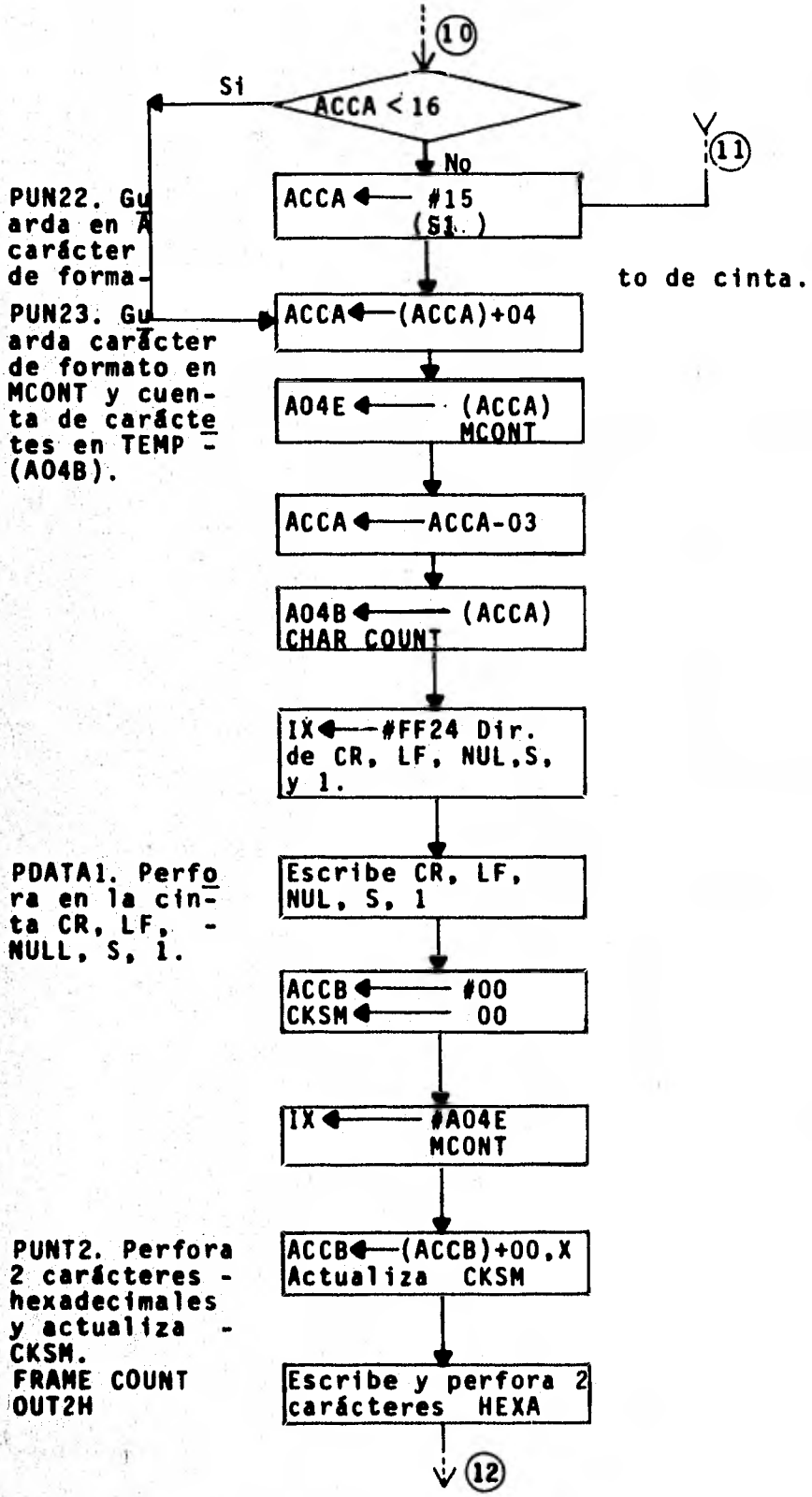
OUTEEE

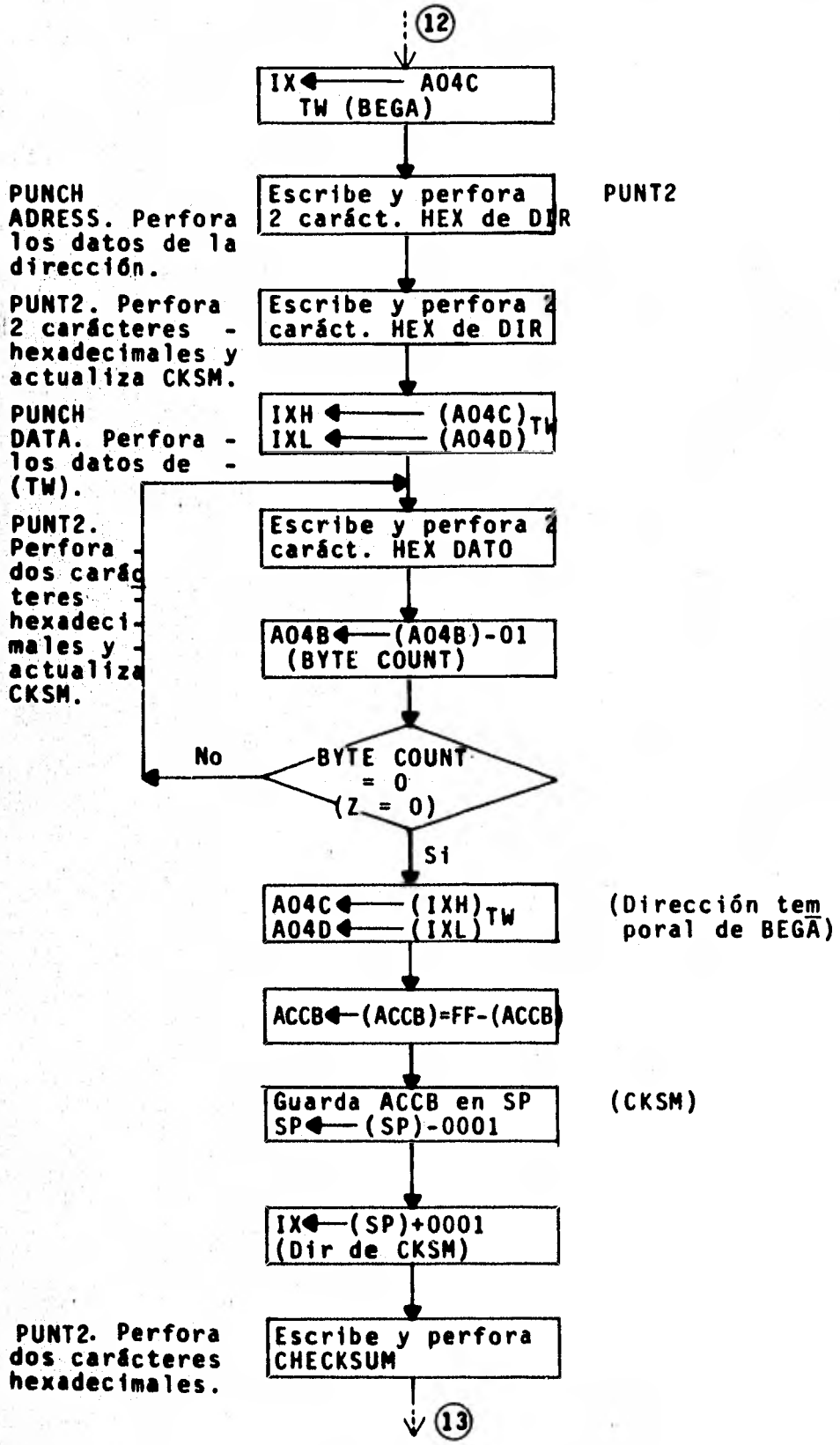
R.- Permite que el monitor muestre el contenido de los registros del microprocesador.

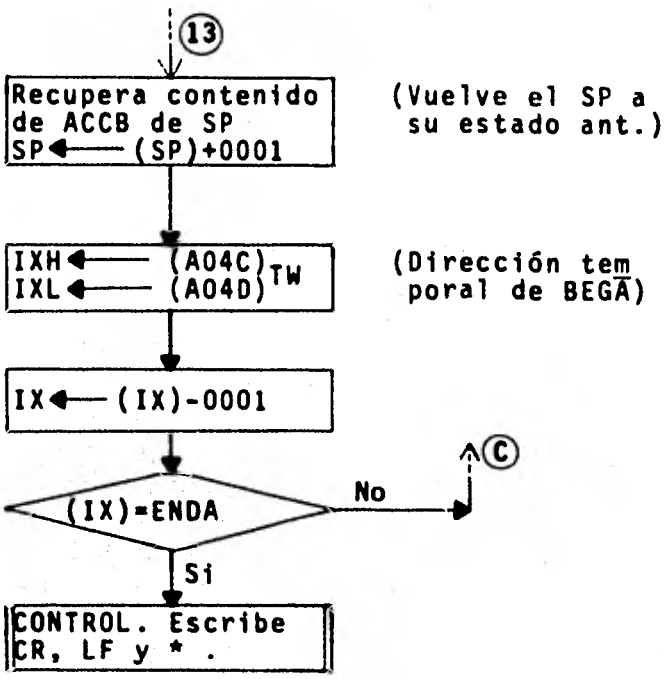


P.- Permite perforar una cinta con formato binario e impri-- mir el contenido de las localidades de memoria seleccionadas. No perfora en la cinta ningún carácter de fin de archivo; - éste carácter (S9) debe ser perforado por el usuario. La dirección inicial del programa se guarda en la localidad de - memoria A03F, A040 y la dirección final en A041, A042.



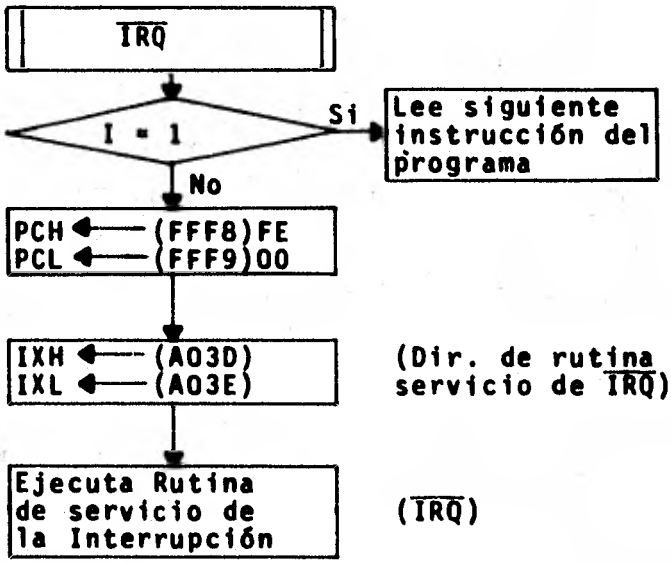

10
Si
ACCA < 16
No
11
ACCA ← #15 (51.)
PUN22. Guarda en A carácter de forma-
to de cinta.
PUN23. Guarda carácter de formato en MCONT y cuenta de caracteres en TEMP - (A04B).
ACCA ← (ACCA)+04
A04E ← (ACCA) MCONT
ACCA ← ACCA-03
A04B ← (ACCA) CHAR COUNT
IX ← #FF24 Dir. de CR, LF, NUL, S, y 1.
PDATA1. Perfora en la cinta CR, LF, - NULL, S, 1.
Escribe CR, LF, NUL, S, 1
ACCB ← #00
CKSM ← 00
IX ← #A04E MCONT
PUNT2. Perfora 2 carácteres - hexadecimales y actualiza - CKSM.
FRAME COUNT OUT2H
ACCB ← (ACCB)+00,X Actualiza CKSM
Escribe y perfora 2 carácteres HEXA
12


12
IX ← A04C
TW (BEGA)
PUNCH ADRESS. Perfora los datos de la dirección.
Escribe y perfora 2 carácter. HEX de DIR
PUNT2
PUNT2. Perfora 2 carácteres hexadecimales y actualiza CKSM.
Escribe y perfora 2 carácter. HEX de DIR
PUNCH DATA. Perfora los datos de (TW).
IXH ← (A04C)TW
IXL ← (A04D)TW
PUNT2. Perfora dos carácteres hexadecimales y actualiza CKSM.
Escribe y perfora 2 carácter. HEX DATO
A04B ← (A04B)-01
(BYTE COUNT)
No
BYTE COUNT = 0
(Z = 0)
Si
A04C ← (IXH)TW
A04D ← (IXL)TW
(Dirección temporal de BEGA)
ACCB ← (ACCB)=FF-(ACCB)
Guarda ACCB en SP
SP ← (SP)-0001
(CKSM)
IX ← (SP)+0001
(Dir de CKSM)
PUNT2. Perfora dos carácteres hexadecimales.
Escribe y perfora CHECKSUM
13

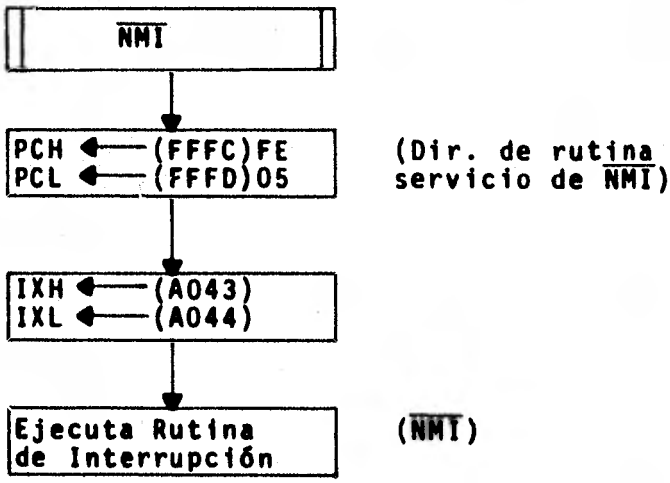

13
Recupera contenido de ACCB de SP
SP ← (SP)+0001
(Vuelve el SP a su estado ant.)
IXH ← (A04C)TW
IXL ← (A04D)TW
(Dirección temporal de BEGA)
IX ← (IX)-0001
(IX)=ENDA
No
C
Si
CONTROL. Escribe CR, LF y * .

INTERRUPCIONES.

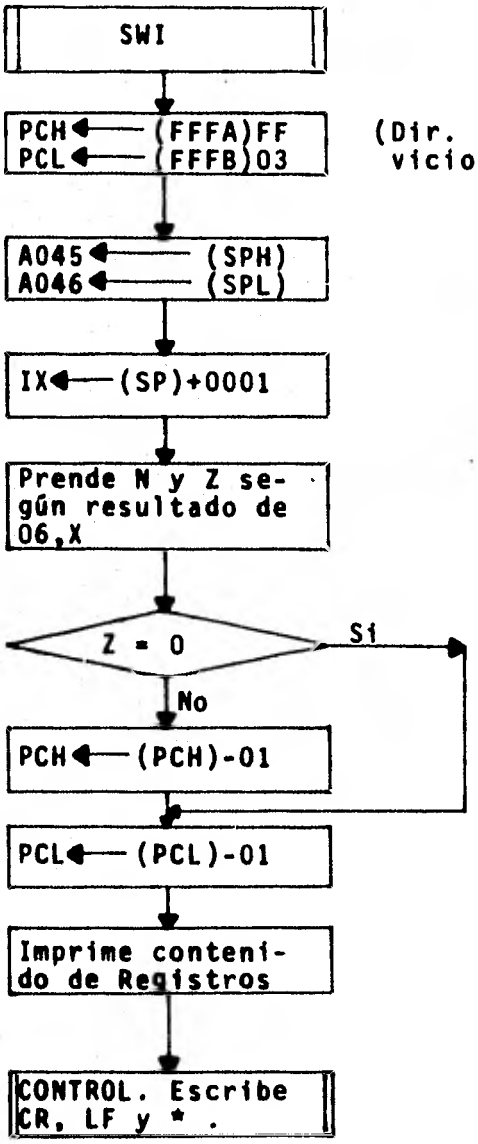
$\overline{IRQ}$.- Esta función permite que el usuario atienda una rutina de interrupción no enmascarable. Si I = 0.



$\overline{NMI}$.- Cuando ocurre esta interrupción todos los registros se guardan en el Stack del usuario.



SWI.- Durante la ejecución de ésta instrucción se guarda en el Stack el contenido de los registros del MPU. El valor del PC es el valor de la dirección de SWI. El bit I se enciende y el MPU no atiende interrupciones ($\overline{IRQ}$).



PRINT. Subrutina que imprime el contenido de los registros del microprocesador.

d). Listado del Monitor.

```
                    *       MONITOR
                    *
                    *       L LOAD
                    *       G EJECUTA EL PROGRAMA
                    *       M CAMBIA EL CONTENIDO DE MEMORIA
                    *       P IMPRIME/PERFORA CINTA
                    *       R MUESTRA EL CONTENIDO DE REGISTROS
                              CC  B  A  X  P  S

      8004          ACIACR  EQU   $8004
      8004          ACIASR  EQU   ACIACR
      8005          ACIADR  EQU   ACIACR+1

                            ORG   $FE00

                    *       I/O SECUENCIA DE INTERRUPCION IRQ
FE00 FE A0 3D       IO      LDX   IOV
FE03 6E 00                  JMP   X

                    *       SECUENCIA DE LA INTERRUPCION NMI
FE05 FE A0 43       POWDWN  LDX   NIO       :OBTIENE VECT NMI
FE08 6E 00                  JMP   X

      FE0A          LOAD    EQU   .
FE0A 86 11                  LDA A #21
FE0C 8D 62                  BSR   OUTCH     :ESCRIBE CARACTER

FE0E 8D 63          LOAD3   BSR   INCH
FE10 81 53                  CMP A #'S'
FE12 26 FA                  BNE   LOAD3     :1° CARACT. NO ES S
FE14 8D 5D                  BSR   INCH      :LEE CARACTER
FE16 81 39                  CMP A #'9'
FE18 27 25                  BEQ   LOAD21
FE1A 81 31                  CMP A #'1'
FE1C 26 F0                  BNE   LOAD3
FE1E 7F A0 47               CLR   CKSM      :HAZ CHECKSUM CERO
FE21 8D 2D                  BSR   BYTE      :LEE UN BYTE
FE23 80 02                  SUB A #2
FE25 B7 A0 48               STA A BYTECT    :NUMERO DE BYTES
                    * FORMA LA DIRECCION
FE28 8D 18                  BSR   BADDR
                    * GUARDA EL DATO
FE2A 8D 24          LOAD11  BSR   BYTE
FE2C 7A A0 48               DEC   BYTECT
FE2F 27 05                  BEQ   LOAD15    :CONTADOR DE BYTE 0
FE31 A7 00                  STA A X         :GUARDA EL DATO
FE33 08                     INX
FE34 20 F4                  BRA   LOAD11
```

```
FE36 7C A0 47      LOAD15   INC     CKSM
FE39 27 D3                  BEQ     LOAD3
FE3B 86 3F         LOAD19   LDA A   #'?'       :ESCRIBE CAR. DE INT.
FE3D 8D 31                  BSR     OUTCH
     FE3F          LOAD21   EQU     .
FE3F 7E FE DB      C1       JMP     CONTROL

                   * FORMA LA DIRECCION DE MEMORIA
FE42 8D 0C         BADDR    BSR     BYTE      :LEE 2 BYTES
FE44 B7 A0 49               STA A   XHI
FE47 8D 07                  BSR     BYTE
FE49 B7 A0 4A               STA A   XLOW
FE4C FE A0 49               LDX     XHI       :(X) DIR. FORMADA

                   * LEE UN BYTE
FE50 8D 53         BYTE     BSR     INHEX     :LEE CARACT. HEX
FE52 48                     ASL A
FE53 48                     ASL A
FE54 48                     ASL A
FE55 48                     ASL A
FE56 16                     TAB
FE57 8D 4C                  BSR     INHEX
FE59 1B                     ABA
FE5A 16                     TAB
FE5B FB A0 47               ADD B   CKSM
FE5E F7 A0 47               STA A   CKSM
FE61 39                     RTS

FE62 44            OUTHL    LSR A             :GUARDA CARACTER
FE63 44                     LSR A             :HEX EN PARTE MS
FE64 44                     LSR A             :DE A
FE65 44                     LSR A

FE66 84 0F         OUTHR    AND A   #$F       :ESCRIBE EL DIGITO
FE68 8B 30                  ADD A   #$30      :BCD HEXADECIMAL
FE6A 81 39                  CMP A   #$39      :QUE ESTA A LA DER.
FE6C 23 02                  BLS     OUTCH
FE6E 8B 07                  ADD A   #$7

                   * ESCRIBE UN CARACTER
FE70 7E FF A5      OUTCH    JMP     OUTEEE
FE73 7E FF 95      INCH     JMP     INEEE

                   * ESCRIBE DATO CUYA DIRECCION SE ENCUENTRA
                   * EN EL REGISTRO X
FE76 8D FB         PDATA2   BSR     OUTCH
FE78 08                     INX
FE79 A6 00         PDATA1   LDA A   00,X
FE7b 81 04                  CMP A   #4
```

```
FE7D 26 F7              BNE     PDATA2    :TERMINA CUANDO
FE7F 39                 RTS               :ENCUENTRES EOT

                      * CAMBIA EL CONTENIDO DE MEMORIA
                        (M AAAA DD NN)
FE80 8D C0            CHANGE  BSR     BADDR     :FORMA LA DIRECCION
FE82 CE FF 8D         CHA51   LDX     #MCL
FE85 8D F2                    BSR     PDATA1    :ESCRIBE CR, LF y *
FE87 CE A0 49                 LDX     #XHI
FE8A 8D 37                    BSR     OUT4HS    :ESCRIBE LA DIRECC.
FE8C FE A0 49                 LDX     XHI
FE8F 8D 34                    BSR     OUT2HS    :ESCRIBE DATO ANTIG.
FE91 FF A0 49                 STX     XHI       :GUARDA DIR. DEL DATO
FE94 8D DD                    BSR     INCH      :LEE UN CARACTER
FE96 81 20                    CMP A   #$20
FE98 26 E8                    BNE     CHA51     :NO SE DIO ESPACIO
FE9A 8D B4                    BSR     BYTE      :LEE NUEVO DATO
FE9C 09                       DEX
FE9D A7 00                    STA A   0,X       :CAMBIA CONTENIDO
FE9F A1 00                    CMP A   0,X
FEA1 27 DF                    BEQ     CHA51     :¿SE MODIFICO?
FEA3 20 96                    BRA     LOAD19    :NO CAMBIO

                      * LEE UN CARACTER HEXADECIMAL
FEA5 8D CC            INHEX   BSR     INCH
FEA7 80 30                    SUB A   #$30
FEA9 2B 94                    BMI     C1        :NO ES CARACT. HEX
FEAB 81 09                    CMP A   #$09
FEAD 2F 0A                    BLE     IN1HG
FEAF 81 11                    CMP A   #$11
FEB1 2B 8C                    BMI     C1        :NO ES CARACT. HEX
FEB3 81 16                    CMP A   #$16
FEB5 2E 88                    BGT     C1        :NO ES CARACT. HEX
FEB7 80 07                    SUB A   #7
FEB9 39               IN1HG   RTS

FE8A A6 00            OUT2H   LDA A   0,X       :ESCRIBE 2 CARACT.HEX
FEBC 8D A4            OUT2HA  BSR     OUTHL     :ESCR. CARACT. HEX IZQ
FEBE A6 00                    LDA A   0,X
FEC0 08                       INX
FEC1 20 A3                    BRA     OUTHR     :ESCRIBE CARACTER
                                                :DERECHO Y R
FEC3 8D F5            OUT4HS  BSR     OUT2H     :ESCR.4 CARACT. HEX
FEC5 8D F3            OUT2HS  BSR     OUT2H     :ESCR. 2 CARACT. HEX
FEC7 86 20            OUTS    LDA A   #$20      :ESPACIO
FEC9 20 A5                    BRA     OUTCH     :(BSR y RTS)

                      * SECUENCIA QUE SE SIGUE CUANDO SE
                      * ENCIENDE LA MICROCOMPUTADORA
      FECB            START   EQU     .
FECB 8E A0 35                 LDS     #$A035
FECE BF A0 45                 STS     SP        :INICIALIZA APUNTA
                                                :DOR DE STACK
```

```
                        * INICIALIZACION DEL ACIA
FED1 86 03                      LDA A   #$3
FED3 B7 80 04                   STA A   ACIACR      :RESET
FED6 86 12                      LDA A   #$12        :DIV POR 64,8+2 BITS
FED8 B7 80 04                   STA A   ACIACR      :DE STOP SIN PARIDAD
FEDB 8E A0 7F           CONTRL  LDS     #STACK      :GUARDA APUNTADOR
                                                    :DE STACK
FEDE CE FF 8C                   LDX     #MCLOFF     :DIR DE CR, LF y *
FEE1 8D 96                      BSR     PDATA1      :ESCR. CR, LF y *

FEE3 8D 8E                      BSR     INCH        :LEE COMANDO
FEE5 16                         TAB                 :PON COMANDO EN B
FEE6 8D DF                      BSR     OUTS        :IMPRIME ESPACIO
FEE8 C1 4C                      CMP B   #'L'
FEEA 26 03                      BNE     +5
FEEC 7E FE 0A                   JMP     LOAD
FEEF C1 4D                      CMP B   #'M'
FEF1 27 8D                      BEQ     CHANGE
FEF3 01 52                      CMP B   #'R'
FEF5 27 18                      BEQ     PRINT       :ESCR. REGISTROS DEL
FEF7 C1 50                      CMP B   #'P'        :MPU
FEF9 27 32                      BEQ     PUNCH       :ESCR. Y PERF. CINTA
FEFB C1 47                      CMP B   #'G'
FEFD 26 DC                      BNE     CONTRL
FEFF BE A0 45                   LDS     SP          :GUARDA EN EL PC LA
                                                    :DIR INICIAL DEL PROGR

                        * ATENCION DE UNA INTERRUPCION POR SOFTWARE
FF03 BF A0 45                   STS     SP          :GUARDA APUNTADOR DE SP
                        * DECREMENTA CONTADOR DE PROGRAMA (PC)
FF06 30                         TSX
FF07 6D 06                      TST     6,X
FF09 26 02                      BNE     +4
FF0B 6A 05                      DEC     5,X
FF0D 6A 06                      DEC     6,X

                        * IMPRIME CONTENIDO DEL STACK
FF0F FE A0 45           PRINT   LDX     SP
FF12 08                         INX
FF13 8D B0                      BSR     OUT2HS      :ESCRIBE CC
FF15 8D AE                      BSR     OUT2HS      :ESCRIBE ACCB
FF17 8D AC                      BSR     OUT2HS      :ESCRIBE ACCA
FF19 8D AB                      BSR     OUT4HS      :ESCRIBE IX
FF1B 8D A6                      BSR     OUT4HS      :ESCRIBE PC
FF1D CE A0 45                   LDX     #SP
FF20 8D A1                      BSR     OUT4HS      :ESCRIBE SP
FF22 20 B7              C2      BRA     CONTRL
```

```
                        * PUNCH
                        * PERFORA DESDE LA DIRECCION INICIAL (BEGA)
                        * HASTA LA DIRECCION FINAL (ENDA)

FF24 0D                 MTAPE1  BYTE    $D,$A,0,0,0,0,'S','1',4
FF25 0A                                                 :FORMATO DE LA
FF26 00                                                 :CINTA PERFORADA
FF27 00                                                 :(CR,LF,NUL,S,1,
FF28 00                                                 : EOT)
FF29 00
FF2A 53
FF2B 31
FF2C 04

                        PUNCH
FF2D 86 12                      LDA A   #$12            :ENCIENDE PERF. DE TTY
FF2F BD FE 70                   JSR     OUTCH           :ESCRIBE UN CARACT.

FF32 FE A0 3F                   LDX     BEGA
FF35 FF A0 4C                   STX     TW              :DIR DE INICIO TEMPORAL
FF38 B6 A0 42           PUN11   LDA A   ENDA+1
FF3B B0 A0 4D                   SUB A   TW+1
FF3E F6 A0 41                   LDA B   ENDA
FF41 F2 A0 4C                   SBC B   TW
FF44 26 04                      BNE     PUN22
FF46 81 10                      CMP A   #16
FF48 25 02                      BCX     PUN23
FF4A 86 0F              PUN22   LDA A   #15
FF4C 8B 04              PUN23   ADD A   #4
FF4E B7 A0 4E                   STA A   MCONT           :GUARDA MCONT TEMPORALM.
FF51 80 03                      SUB A   #3
FF53 B7 A0 4B                   STA A   TEMP            :GUARDA CUENTA DE CARACT
                        * PERFORA CR, LF, NULL, S, 1
FF56 CE FF 24                   LDX     #MTAPE1
FF59 BD FE 79                   JSR     PDATA1
FF5C 5F                         CLR B                   :HAZ CERO EL VALOR DE
                        * PERFORA CONTENIDO             :CHECKSUM
                        * DE MCONT (FRAME COUNT)
FF5D CE A0 4E                   LDX     #MCONT
FF60 8D 25                      BSR     PUNT2           :PERFORA 2 CARACT. HEX
                        * PERFORA EL VALOR DE LA DIRECCION
FF62 CE A0 4C                   LDX     #TW             :DIRECCION DONDE SE
FF65 8D 20                      BSR     PUNT2           :GUARDA EL VALOR EN
FF67 8D 1E                      BSR     PUNT2           :FORMA TEMPORAL
                        * PERFORA LOS DATOS
FF69 FE A0 4C                   LDX     TW
FF6C 8D 19              PUN32   BSR     PUNT2           :PERFORA 1 BYTE
FF6E 7A A0 4B                   DEC     TEMP            :DECR CUENTA DE CARACT.
FF71 26 F9                      BNE     PUN32
FF73 A0 4C                      STX     TW
FF76 53                         COM B
```

```
FF77 37                 PSH B
FF78 30                 TSX
FF79 8D 0C              BSR     PUNT2       :PERFORA VAL. DE CHECKSUM
FF7B 33                 PUL B               :RECUPERA STACK
FF7C FE A0 4C           LDX     TW
FF7F 09                 DEX
FF80 BC A0 41           CPX     ENDA
FF83 26 B3              BNE     PUN11
FF85 20 9B              BRA     C2          :PASA A CONTRL

                        * PERFORA 2 CARACTERES HEX, ACTUALIZA CHECKSUM
FF87 EB 00              PUNT2   ADD B   0,X         :ACTUALIZA CHECKSUM
FF89 7E FE BA           JMP     OUT2H       :ESCRIBE DOS CARACTE-
                                            :RES HEX Y RTS
FF8C 13                 MCLOFF  BYTE    $13         :APAGA LECTORA
FF8D 0D                 MCL     BYTE    $D,$A,$14,0,0,0,'*',4 :PERFORA
FF8E 0A                                             :CR, LF y *
FF8F 14
FF90 00
FF91 00
FF92 00
FF93 2A
FF94 04

                        * LEE UN CARACTER Y GUARDALO EN ACCA
FF95 86 01              INEEE   LDA A   #$1         :BIT RDRF
FF97 B5 80 04           INE1    BIT A   ACIASR      :ESPERA QUE DR ES-
FF9A 27 FB              BEQ     INE1        :TE LLENO
FF9C B6 80 05           LDA A   ACIADR      :LEE EL CARACTER
FF9F 84 7F              AND A   #$7F        :IGNORA LA PARIDAD
FFA1 81 7F              CMP A   #$7F
FFA3 27 F0              BEQ     INEEE       :IGNORA CARACTER
                                            :DE RUBOUT (DEL)
                        * ESCRIBE UN CARACTER DE ACCA
FFA5 37                 OUTEEE  PSH B
FFA6 C6 02              LDA B   #$2         :BIT TDRE
FFA8 F5 80 04           OUT1    BIT B   ACIASR      :ESPERA TRANSMISION
FFAB 27 FB              BEQ     OUT1        :DE DATO
FFAD B7 80 05           STA A   ACIADR      :ESCRIBE UN CARACTER
FFB0 33                 PUL B
FFB1 39                 RTS

                        * VECTORES DE INTERRUPCION
                        ORG     $FFF8
FFF8 FE 00              WORD    IO          :IRQ
FFFA FF 03              WORD    SFE         :SWI
FFFC FE 05              WORD    POWDWN      :NMI
FFFE FE CB              WORD    START       :RESTART
```

```
                    * DIRECCIONES DE ALMACENAMIENTO TEMPORALES

                            ORG     $A03D
A03D                IRQ     RMB     2       :APUNTADOR DE IRQ
A03F                BEGA    RMB     2       :DIR INICIAL DE PNCH
A041                ENDA    RMB     2       :DIR FINAL DE PNCH
A043                NMI     RMB     2       :APUNTADOR DE INT NMI
A045                SP      RMB     1       :S - PARTE MS (HIGH)
A046                        RMB     1       :S - PARTE LS (LOW)
A047                CKSM    RMB     1       :CHECKSUM
A048                BYTECT  RMB     1       :CUENTA DE BYTECT
A049                XHI     RMB     1       :PARTE MS DE REG X
A04A                XLOW    RMB     1       :PARTE LS DE REG X
A04B                TEMP    RMB     1       :CUENTA DE CARACTERES
A04C                TW      RMB     2       :ALMACEN TEMPORAL
A04E                MCONT   RMB     1       :ALMACEN TEMPORAL
A04F                XTEMP   RMB     2       :ALMACEN TEMP DE REG X
A051                        RMB     46      :TAMAÑO DEL STACK
A07F                STACK   RMB     1       :APUNTADOR DE STACK
                                            :EN LA DIRECCION SU-
                                            :PERIOR DE MEMORIA
                            END
```

CAPITULO IV

CONSIDERACIONES DE DISEÑO

4.1.- Direccionamiento de Periféricos y Memoria.

Una vez que se conocen las caracterfsticas de la familia M6800. El siguiente paso es conectar al microprocesador: los relojes, los buffers para el bus de direcciones, datos y control, la memoria solo de lectura (ROM), la memoria de acceso aleatorio (RAM), la interfase de comunicación en serie (ACIA) y la interfase de comunicación en paralelo (PIA).

Como parte inicial para el alambrado del sistema se realizo el mapa de memoria del sistema completo (Figura 4.1), que permite asignar las direcciones de memoria de los distintos dispositivos.

DISPOSITIVO	LINEAS DE DIRECCION (A_0 - A_{15})																DIR
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RAM 1	0	-	-	-	-	-	0	0	0	X	X	X	X	X	X	X	0000 - 007F
RAM 2	0	-	-	-	-	-	0	0	1	X	X	X	X	X	X	X	0080 - 00FF
RAM 3	0	-	-	-	-	-	0	1	0	X	X	X	X	X	X	X	0100 - 017F
RAM 4	0	-	-	-	-	-	0	1	1	X	X	X	X	X	X	X	0180 - 01FF
RAM 5	1	0	1	0	-	-	-	-	-	X	X	X	X	X	X	X	A000 - A07F
ROM	1	1	1	1	-	-	-	X	X	X	X	X	X	X	X	X	F000 - F1FF
ACIA	1	-	0	-	-	-	-	-	-	-	-	-	-	1	-	X	8004 - 8005
PIA	1	-	0	-	-	-	-	-	-	-	-	-	1	0	X	X	8008 - 800B

X = Conectado - = No Conectado

Figura 4.1.- MAPA DE LA MEMORIA DEL SISTEMA.

En el direccionamiento directo, la dirección del operando esta contenida en el segundo byte de la instrucción. Este direccionamiento permite que el usuario use los 256 bytes mas bajos de la memoria con un tiempo de ejecución menor, lo que hace que se emplee este método de direccionamiento siempre que es posible. Esta es una primera condición para la asignación de las localidades bajas de memoria al RAM.

El paso siguiente es examinar alguna restricción posible para la asignación de las direcciones de memoria ROM.

En una aplicación típica un dispositivo periférico puede interrumpir al microprocesador. Una secuencia de interrupción se inicia aplicando la señal de control apropiada a cualquiera de las tres interrupciones de Hardware; $\overline{\text{RESET}}$ ($\overline{\text{RES}}$) Interrupciones no enmascarables ($\overline{\text{NMI}}$) Interrupciones enmascarables ($\overline{\text{IRQ}}$), o bien usando la interrupción por software (SWI).

En caso de presentarse uns de estas interrupciones, se presenta en el bus de Direcciones la dirección del vector de interrupción correspondiente que tiene la dirección de la rutina de servicio de la interrupción. Estos vectores se encuentran a partir de la dirección FFF8 a FFFF. Esto establece la segunda condición sobre el sistema: la memoria ROM debe estar en las localidades altas. La dirección F1F8, que se encuentra en memoria ROM es equivalente a la del vector de interrupciones FFF8.

Una vez que la memoria baja y alta han sido asignadas, se asigna la memoria intermedia a la sección de entrada/sal.

Cuando se han asignado las localidades de memoria a los diferentes dispositivos (Fig. 4.1) se conectan el bus de Datos, el bus de Direcciones y el bus de Control de la tarjeta del CPU (Fig. 4.2), tarjeta de memoria RAM (Fig. 4.3), Tarjeta de interfase Serie (Fig. 4.4) y tarjeta de interfase en paralelo con convertidores A/D y D/A (Fig. 4.5).

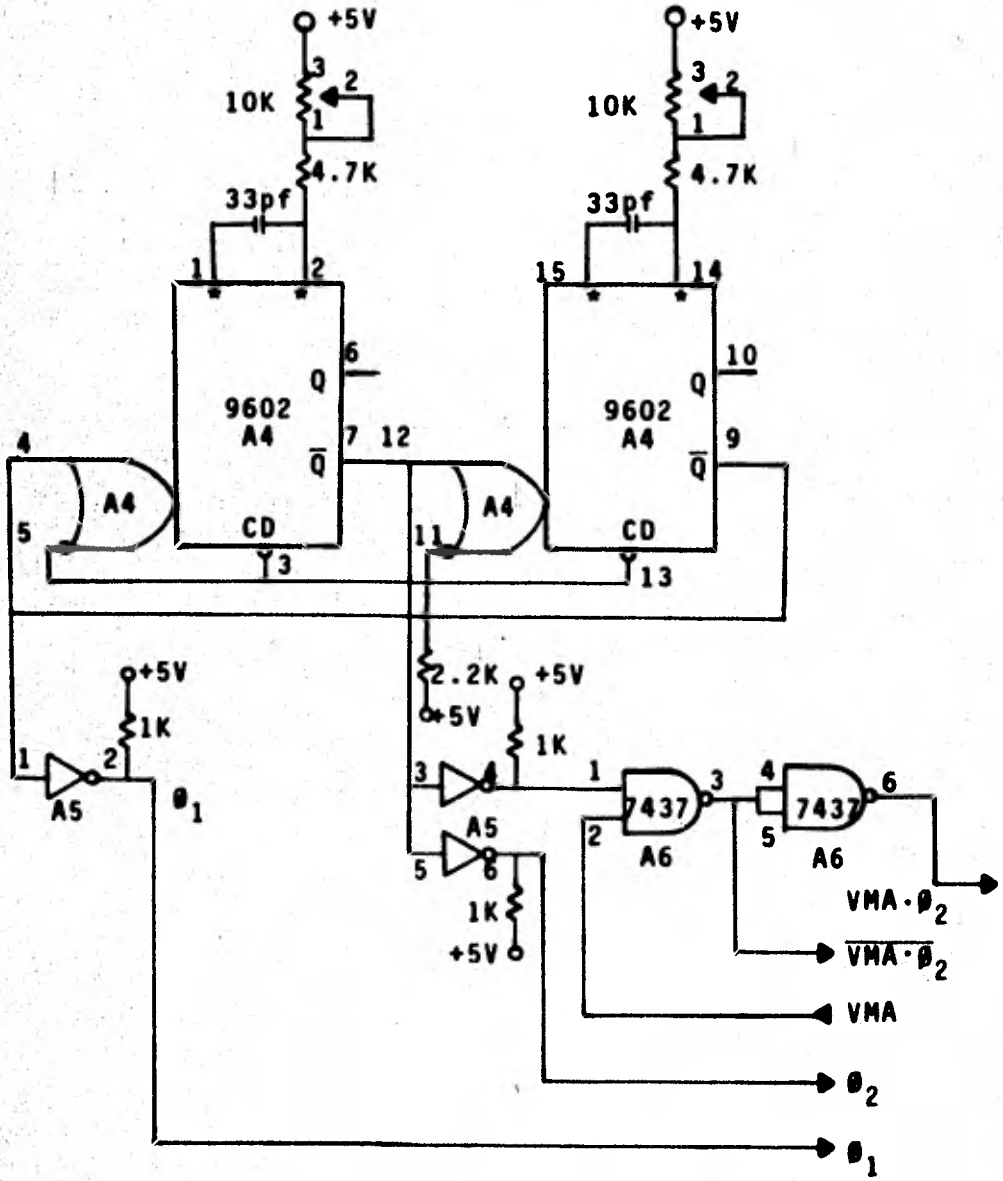
Cada dispositivo obtiene su dirección a través del alambrado del bus de Direcciones. Para los RAM se conectan A_0 a A_6 al bus de Direcciones, para el ROM se conectan A_0 a A_8 a las lineas del bus de direcciones correspondientes. Para el PIA se conectan $\overline{RS}_0$ y $\overline{RS}_1$ a las líneas A_0 y A_1 y en el ACIA RS se conecta a la línea A_0.

Como se vé en la Fig. 4.1, las lineas del bus de direcciones mas significativas permiten discriminar un tipo de dispositivo de otro. Por ejemplo cuando se direcciona la memoria RAM, el ROM y los dispositivos de I/O se deshabilitan, esto se consigue conectando la línea A_{15} a varias entradas CS (Chip selects) positivas o $\overline{CS}$ (Chip selects) negativas.

Para escoger un dispositivo entre otro de su mismo tipo se usan las lineas intermedias del bus de direcciones (en el caso de las memorias RAM, A_7, A_8, A_9).

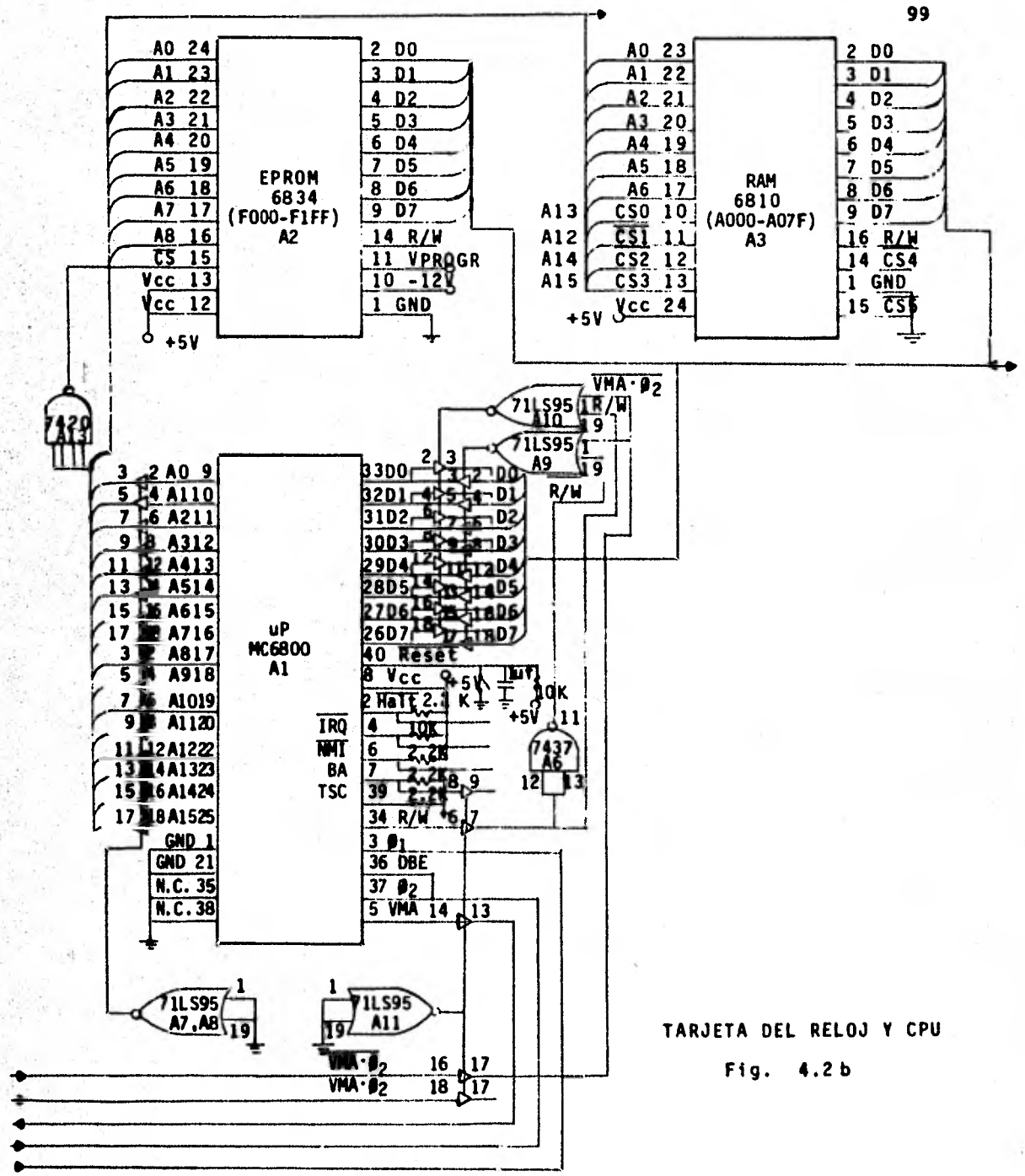
Las entradas CS que no se usan se conectan a +5V o tierra para reducir el ruido al mínimo.

Cuando se encuentran alambrados el bus de Datos y de Direcciones se conectan las lineas del Bus de Control, que son: $\overline{IRQ}$, $\overline{RESET}$, $\emptyset_2$, R/W y VMA.



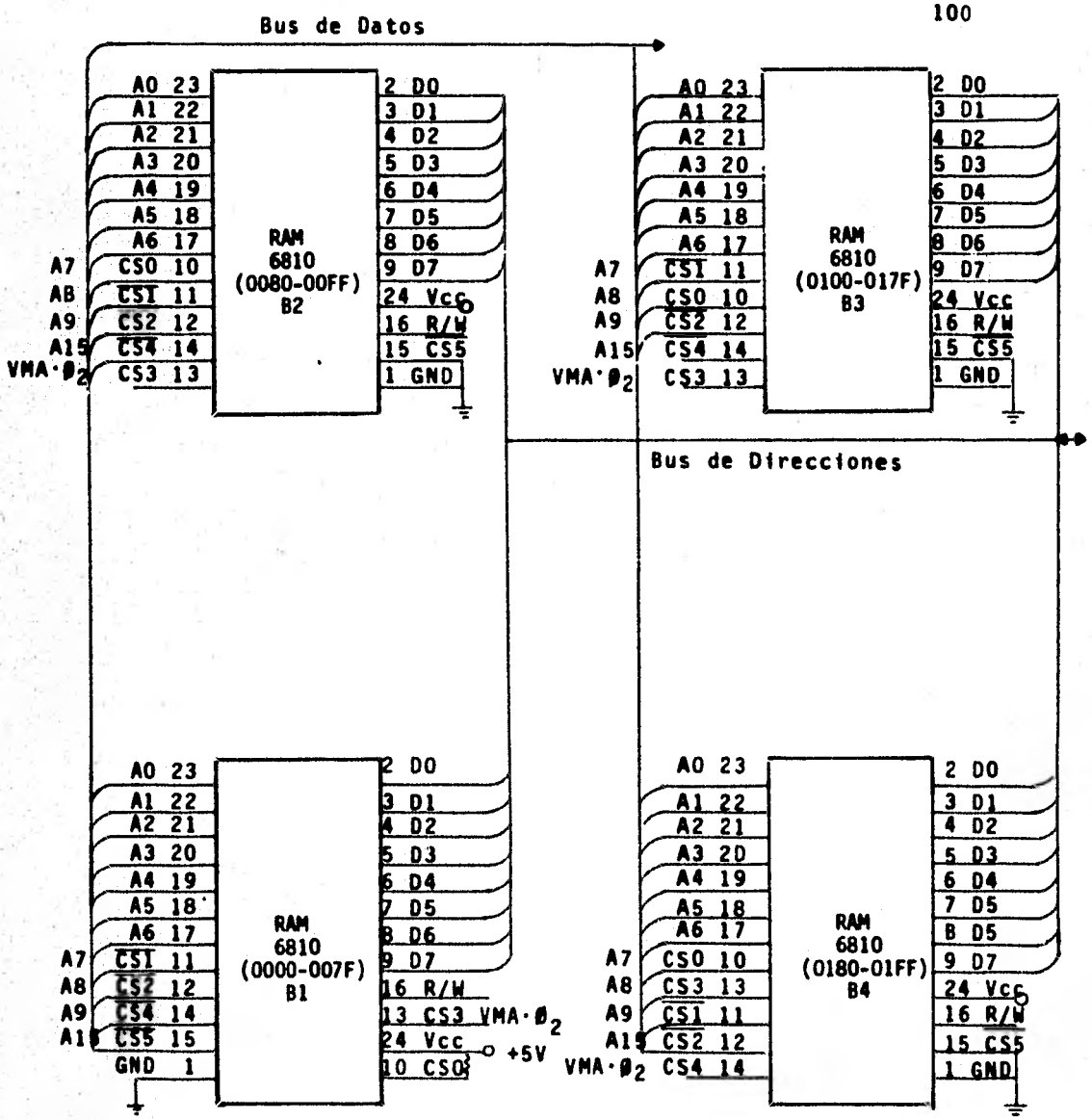
A5 = 7406
* = Reloj externo

Figura 4.2 a. Reloj del Microprocesador

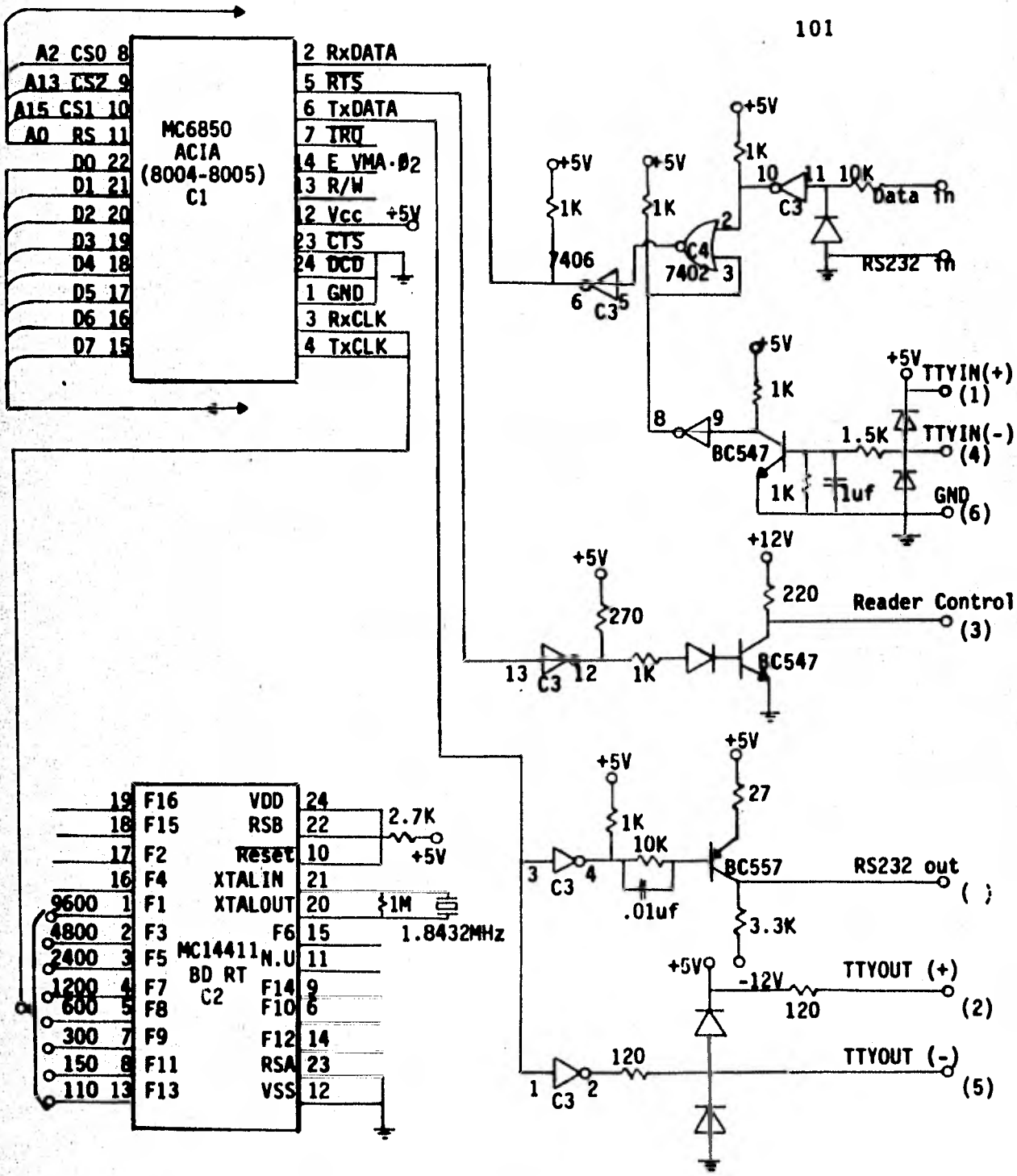


TARJETA DEL RELOJ Y CPU

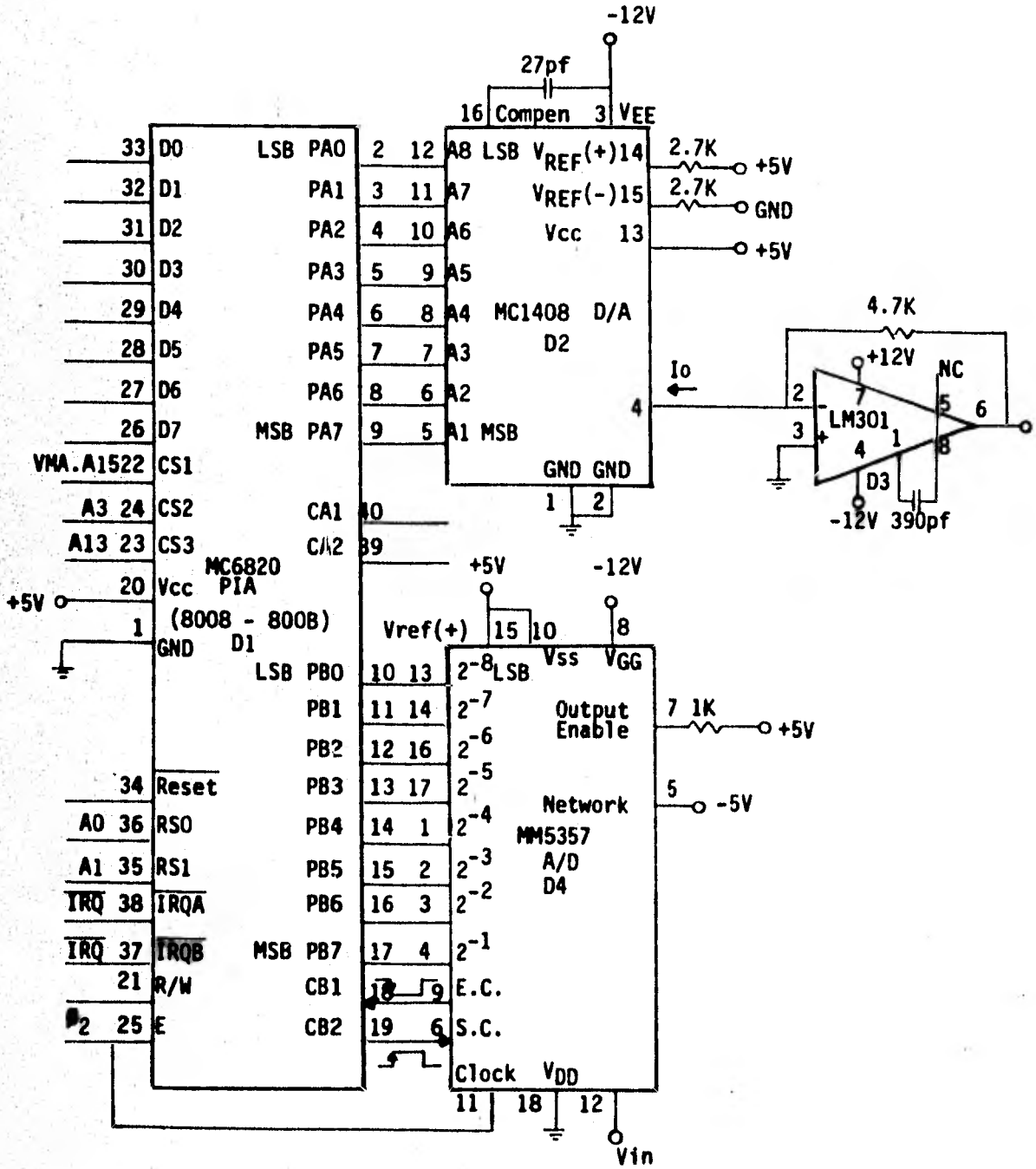
Fig. 4.2 b



TARJETA DE MEMORIA RAM Fig. 4.3



TARJETA DE INTERFASE SERIE Fig. 4.4



TARJETA DE INTERFASE EN PARALELO, A/D Y D/A. Fig. 4.5

La señal de $\overline{\text{RESET}}$ se genera externamente y se conecta - al microprocesador y al PIA.

Las líneas de $\overline{\text{IRQ}}$ se alambran todas juntas y se conec-- tan a $\overline{\text{IRQ}}$ del procesador.

La señal de reloj $\emptyset_2$ se usa para restringir la transferencia de datos, que ocurre en la parte alta de la señal de reloj y se conecta a la entrada E (Enable) del PIA; en oca-- siones se realiza un AND lógico con la línea VMA, para habilitar a los dispositivos periféricos (ACIA) y memorias RAM - cuando haya una dirección válida (Figuras 4.2 - 4.5).

La línea de R/W se conecta a la entrada de las memorias RAM e interfases de entrada/salida para controlar la direc-- ción del flujo de datos.

A partir de la información del mapa de memoria (Figura 4.1) se alambran las distintas tarjetas de la microcomputadora.

El conjunto de tarjetas se conectan a un conector alambrado en paralelo, cuya configuración se muestra en la Figura 4.6.

Lado del alambrado		Lado de Componentes	
A	+5	1	GND
B	A_0	2	$\overline{RESET}$
C	A_1	3	$VMA \cdot \emptyset_2$
D	A_2	4	$\overline{VMA \cdot \emptyset_2}$
E	A_3	5	-
F	A_4	6	VMA
H	A_5	7	-
J	A_6	8	-
K	A_7	9	-
L	A_8	10	-
M	A_9	11	+12V
N	A_{10}	12	-12V
P	A_{11}	13	D_0
R	A_{12}	14	D_1
S	A_{13}	15	D_2
T	A_{14}	16	D_3
U	A_{15}	17	D_4
V	$\overline{HALT}$	18	D_5
W	$\overline{IRQ}$	19	D_6
X	$\overline{NMI}$	20	D_7
Y	R/W	21	B.A.
Z	+5V	22	GND

- = Desconectado (No se usa)

Figura 4.6. CONECTOR

4.2.- Reloj

Las señales de reloj que requiere el microprocesador se generan con un Multivibrador Monoestable 9602 (Fig. 4.2), - que proporciona un pulso de salida cuya duración y precisión es función de los componentes de tiempo externos (R_x, C_x).

La inmunidad al ruido de la línea de V_{cc} y tierra es - buena. Sus entradas son compatibles con lógica TTL y la capacidad de carga (fan-out) a la salida es grande.

El disparo (trigger) del multivibrador es independiente del frente de onda que entra (positivo o negativo).

Cuando se inicia un ciclo el capacitor externo C_x se - descarga y se carga rápidamente. Si el tiempo del ciclo de - entrada es menor que el de salida, la salida del 9602 es continua.

Como se muestra en el diagrama lógico (Fig. 4.2) se necesita una resitencia R_x y un capacitor C_x externos. El valor de R_x puede variar de 5.0K a 50K cuando opera entre - 0° y 75°C.

El valor de C_x puede variar de 0 a 10^3 pf.

El ancho del pulso del reloj t, esta dado por:

$$t = 0.31R_xC_x[1 + \frac{1}{R_x}]$$

Donde R_x esta en Kn, C_x esta en pf y t esta en ns.

La fase del reloj ϕ_1 es el complemento de la fase de - reloj ϕ_2.

4.3.- Buffers.

Un Buffer es un amplificador que se usa para incrementar la capacidad de corriente de una línea del microprocesador y permite aislar distintas partes del mismo. Cuando el número de dispositivos conectados al microprocesador esta dentro de las limitaciones de carga básica de éste (sistema mínimo) no es necesario conectar buffers. Sin embargo cuando se tienen varios circuitos de memoria MOS conectados en paralelo, las capacitancias de entrada de cada memoria se suman, teniendose una carga dinámica. Para eliminar los problemas de carga es necesario agregar amplificadores (Buffers).

Los buffers DM 71LS95 empleados (Figura 4.7) tienen 8 amplificadores en cada circuito integrado, con dos líneas de entrada cada uno. Emplean tecnología TTL Schottky de baja potencia. Una de las 2 entradas a cada buffer se usa como línea de control, que permite poner la línea de salida en un estado de alta impedancia, la otra línea pasa el dato a través del buffer.

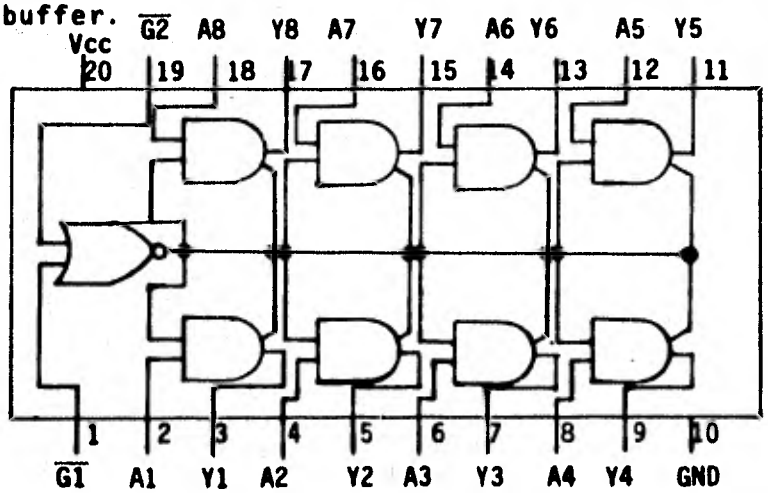


Figura 4.7. Diagrama de Conexión del Buffer DM 71LS95.

El retrazo típico de la señal en el buffer es de 13ns y la potencia típica que disipa es de 80 mw. A continuación se presenta la tabla de verdad.

Entradas			Salida
$\overline{G}_1$	$\overline{G}_2$	A	Y
H	X	X	Z
X	H	X	Z
L	L	H	H
L	L	L	L

donde:

H = Nivel alto

X = Cualquier nivel

L = Nivel bajo

Z = Estado de alta impedancia.

4.4.- Problemas encontrados y forma en que fueron resueltos.

Inicialmente se alambro un sistema prototipo usando la tecnica de alambrado conocida como "Wire Wrapp". Los problemas que se encontraron fueron en los niveles superiores e inferiores del reloj, que impedia la puesta en marcha del sistema; esto se soluciono usando el multivibrador monoestable 9602 que tiene una gran capacidad (Fan-out) de carga a la salida. Después se modifico el monitor para cambiar a una localidad de memoria más alta el área de variables; los errores existentes en el monitor solo pudieron detectarse con el uso del analizador de estados lógicos (Secc. 4.5). Resueltos

los problemas anteriores fue posible poner en funcionamiento un sistema mínimo.

Una vez que funciono el sistema mínimo se construyeron 4 tarjetas formadas básicamente por uno de los elementos de una Microcomputadora. Estas se describen a continuación:

Tarjeta del Reloj y CPU (A).- Tiene un Microprocesador MC - 6800 (A1), un circuito para generar el reloj del microprocesador (A4,A5,A6, Fig. 4.2a), un EPROM S6834 (A2) que contiene el programa Monitor, Memoria RAM MC6810 (A3) que sirve como almacén temporal de variables del monitor y Buffers DM - 71LS95 (A7, A8, A9, A10, A11).

Tarjeta de Memoria RAM (B).- Esta formada por 4 circuitos integrados de memoria MC 6810 (B1, B2, B3, B4) que ocupan las localidades de memoria 0000-01FF. La tarjeta tiene disponible espacio para 8 RAMs adicionales. Antes de agregar esta - memoria es necesario decodificar las localidades correspondientes.

Tarjeta de Interfase Serie (C).- Tiene el circuito integrado ACIA MC 6850 que permite al microprocesador comunicarse a - través de la interfase RS232 con una terminal, y a través de la interfase de 20 mA con un teletipo (Figura 4.4). La frecuencia de transmisión y recepción la da el circuito de reloj integrado MC14411 (C2).

Tarjeta de Interfase en Paralelo, A/D y D/A (D).- Esta formada por el PIA MC6820 (D1), un convertidor D/A MC1408 (D2), - un convertidor de corriente a voltaje LM 301 (D3) y un convertidor A/D MM5357 (D4) cuyas características se describen

en el Capítulo V.

Durante la construcción de éstas 4 tarjetas se tuvieron problemas con la decodificación del EPROM S6834. Fue necesario agregar una compuerta NAND 7420 de 4 entradas (Figura 4.2b) que permitio discriminar a éste EPROM de las demás componentes y al mismo tiempo redujo el área de memoria que tiene una imagen del EPROM.

Construir una microcomputadora del conjunto de circuitos integrados que la forman implica la realización de un gran esfuerzo para conectar el reloj y memorias al CPU para que el sistema sea probado.

4.5.- Analizador de Estados Lógicos.

Ya que muchos de los problemas de Hardware de un microprocesador envuelven las relaciones entre muchas señales durante periodos de reloj sucesivos, con frecuencia resulta difícil encontrar la causa de un problema viendo solo una o dos de las señales en el osciloscopio. Una herramienta muy útil, diseñada para corregir y detectar errores es el analizador de Estados Lógicos. El analizador de Estados Lógicos 1600A de Hewlett Packard tiene 18 puntas de prueba lógicas en 3 conectores con 6 puntas cada uno, que pueden conectarse al bus de Direcciones, Datos y Control para conocer la información que se encuentra en los registros y memorias del Microprocesador en la forma de 1's y 0's. Tiene también una punta de prueba para una señal de reloj.

El 1600A permite analizar sistemas digitales que dependen de secuencias de estados lógicos para su operación.

El 1600A tiene una pantalla donde se muestra la información de los estados lógicos en uno de dos formatos posibles:
1) Tabular.- Los datos del sistema se muestran en forma de 1's y 0's en dos tablas de 16 palabras de 16 bits. La tabla a la izquierda de la pantalla (A) es una muestra de los datos de la memoria A adquiridos del circuito de prueba a través de los conectores de entrada de datos del 1600A.

La tabla a la derecha de la pantalla es una muestra de los datos de la memoria B que fueron transferidos de la memoria A al oprimir el botón: guarda A - B (Store A - B), o bien pueden ser los datos de entrada a un Analizador 1607A conectado al 1600A a través de un cable de interfase, o puede ser el OR exclusivo de los datos de la memoria A y B'.
2) Mapa.- En este formato el 1600A muestra las palabras digitales como puntos sobre la pantalla, la posición de estos identifica una palabra en forma única. En este modo el 1600A puede mostrar hasta 65,535 palabras.

Cuando el Analizador de Estados Lógicos realiza un ciclo de adquisición de datos, cada palabra de entrada se almacena temporalmente antes de ser cargada en memoria. Los datos de entrada que son almacenados temporalmente se comparan con la palabra de entrada (trigger word).

Cuando una palabra de entrada reune las condiciones seleccionadas con el panel (trigger word y qualifier) el 1600A realiza un algoritmo de adquisición de datos.

Para comparar la palabra seleccionada con el panel se presionan simultáneamente el reset del 1600A y el reset del

microprocesador. Después de disparar con ambos resets y habiendo seleccionado como palabra de datos FFFE se encontro - la secuencia de Restart que se tiene en el monitor. En el modo de mapeo se puede ver como el Monitor hace la inicialización del puerto de entrada/salida (ACIA). De esta forma pudo conocerse si el acceso del microprocesador era a la localidad de memoria correcta y si esta se encontraba habilitada.

Usando el modo de mapeo puede verse también el contenide de la memoria RAM cuando corre un programa, lo que permite determinar si éste sigue la secuencia correcta. Esto ayuda a su corrección.

CAPITULO V

APLICACION DE LA MICROCOMPUTADORA COMO INSTRUMENTO DE MEDIDA

5.1.- Medición de voltajes.

Dentro de un microprocesador todas las señales se representan por valores digitales. Muchas de las señales que se quieren analizar con un microprocesador son señales analógicas que varian continuamente. El proceso de convertir una señal analógica a una señal digital se llama conversión A/D.

Como un sencillo ejemplo del uso de los microprocesadores, discutiremos enseguida como se implemento la medición de voltajes usando el convertidor A/D MM 5357 y el puerto de interfase PIA MC6820 (Figura 4.5).

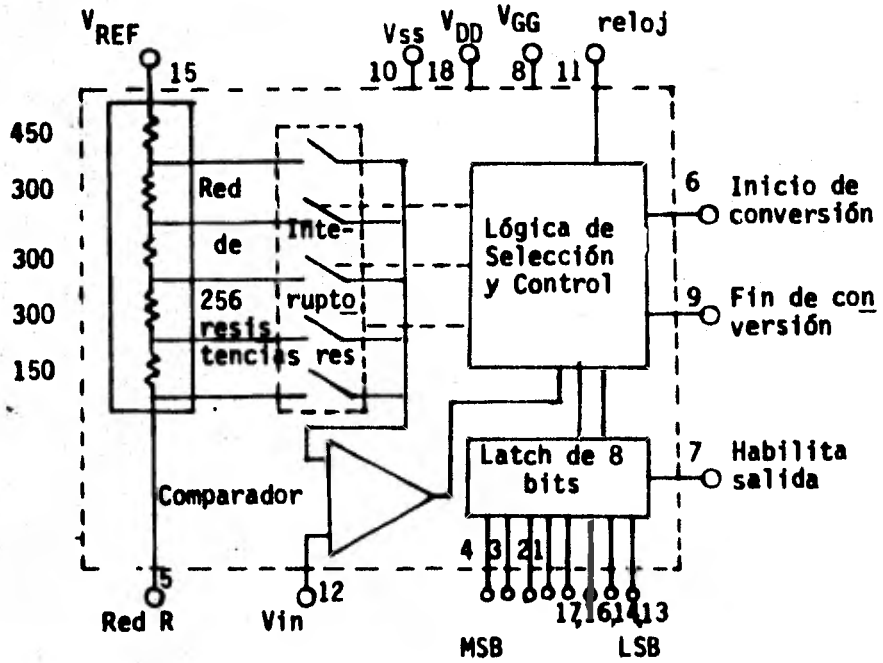


Figura 5.1. Diagrama de Bloques de convertidor A/D MM5357.

El convertidor A/D MM5357 es un convertidor de 8 bits, construido con tecnología MOS de canal P. Tiene un comparador de alta impedancia de entrada (>100MΩ), 256 resistencias en serie, interruptores (switches) analógicos, lógica de selección y control (Figura 5.1) y adicionalmente la información que resulta se guarda (latches) durante un tiempo posterior a la conversión para su transferencia al Microprocesador.

El convertidor realiza la conversión utilizando la técnica de aproximaciones sucesivas, donde el voltaje de entrada analógico desconocido se compara con el voltaje que existe en el punto de unión de las resistencias usando interruptores (switches) analógicos.

La conversión se inicia mandando un pulso positivo a la pata 6 (Start conversion). La señal de entrada se compara con un voltaje de referencia, cuya salida sigue una de las ramas del árbol de la figura 5.2.

Durante cada ciclo sucesivo se suman 1/2, 1/4, 1/8,... del voltaje de referencia y se comparan con la señal analógica a la entrada del convertidor. Si después de un ciclo de conversión la suma de voltajes de referencia es mayor que el valor de entrada, entonces el valor en que se habia incrementado la suma de voltajes de referencia se descarta y se prueba un incremento más pequeño; el proceso continua hasta que se tiene el valor V_{ref}/N más cercano al valor de entrada. N define una derivación especifica de la red de resistencias.

Cuando la conversión termina la sección de control -

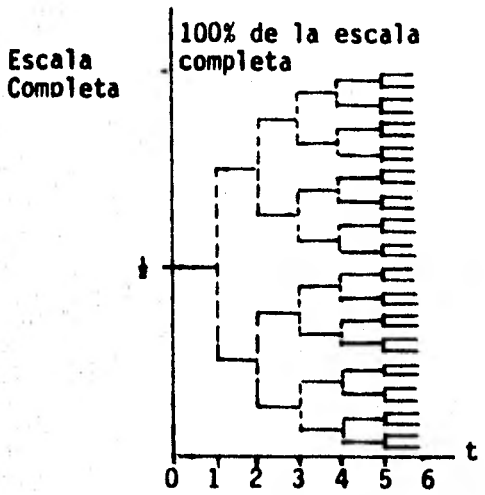


Figura 5.2. Secuencia de conversión por aproximaciones Sucesivas.

lógico guarda a la salida del convertidor (latch) una pala-- bra binaria que corresponde a la señal lógica que se tiene a la entrada y genera un pulso negativo (Figura 5.3) que indica el fin de la conversión.

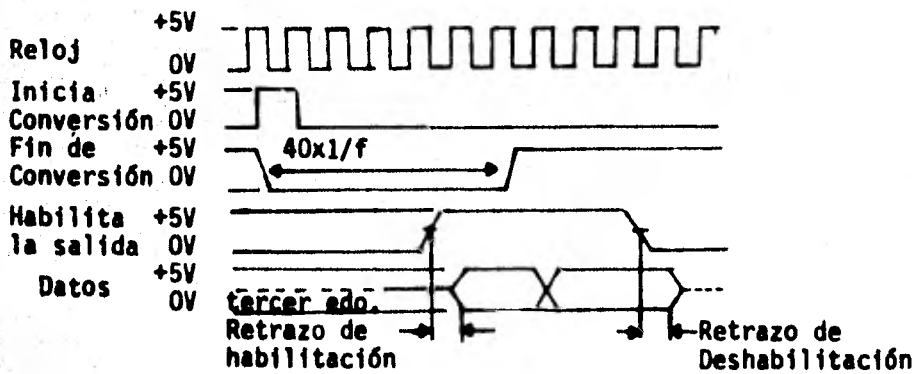


Figura 5.3. Diagrama de Tiempo.

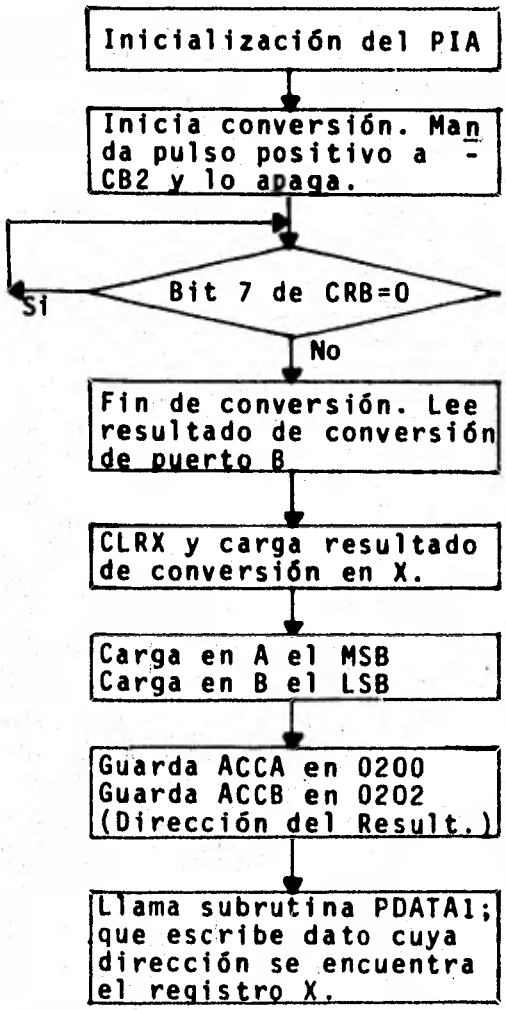
Las características del convertidor A/D son:

- Resolución 8 bits
- Linealidad ± LSB
- Velocidad de conversión 40 us
- Impedancia de entrada 100 M
- Voltajes de alimentación +5.0V, -12V, GND
- Frecuencia del reloj 5.0KHz a 2.0MHz
- Compatible con TTL

El voltaje de referencia determina el rango del voltaje de entrada. Si Vref se conecta a +5V el voltaje de entrada puede variar de -5 a +5 volts.

El programa para la medición de voltajes inicializa el puerto del microprocesador (PIA), posteriormente manda un pulso positivo a CB2 que inicia el proceso de conversión del A/D y espera que este disponible el valor correspondiente al voltaje de entrada checando la línea de entrada CB1; cuando esto sucede lee de una tabla de memoria, que contiene el valor del voltaje correspondiente y escribe tal valor en el teletipo o terminal.

Enseguida se presenta el diagrama de flujo del programa y un listado del mismo:



La dirección de los registros del PIA es:

8008	Registro de datos de A (DDRA).
8009	Registro de control de A (CRA).
800A	Registro de datos de B (DDRB).
800B	Registro de control de B (CRB).

Programa de conversión A/D para la medición de voltajes.

```
      * Inicialización del PIA

0100 4F           CLR A
0101 B7 80 09     STA A  CRA  :Limpia registro de control A
0104 B7 80 0B     STA A  CRB  :Limpia registro de control B
0107 B7 80 0A     STA A  DDRB :Establece PB0-PB7 como entradas
010A 86 F6        LDA A  #F6  :Selecciona el Reg. de Recepción
010C B7 80 0B     STA A  CRB  :y transmisión de datos y fija
                              :el modo de control para el lado
                              :B
010F 86 FF        LDA A  #FF
0111 B7 80 08     STA A  DDRA :Establece PA0-PA7 como salidas
0114 86 04        LDA A  #04  :Selecciona el Reg. de Recepción
0116 B7 80 09     STA A  CRA  :y transmisión de datos y fija
                              :el modo de control para el lado
                              :A

      * Inicia Conversión. Manda pulso
      * positivo a CB2 y lo apaga.

0119 B6 80 0B     LDA A  CRB  :Inicia conversión de A/D
011C 8A 08        ORA    #08
011E B7 80 0B     STA A  CRB  :Manda pulso positivo a CB2.
0121 84 F7        AND A  #F7
0123 B7 80 0B     STA A  CRB  :Apaga CB2

      * Lee bit 7 de CRB que indica fin de conversión.

0126 B6 80 0B S1  LDA A  CRB  :Lee bit 7 de CRB
0129 84 80        AND A  #80
012B 27 F8        BEQ    S1   :Espera hasta que termina
                              :conversión (Bit 7 = 0).

      * Lee resultado de conversión
      * del puerto B y carga registro X

012D B6 80 0A     LDA A  DDRB :Lee puerto B
0130 5F           CLR B       :B   (00)
0131 F7 02 10     STA B  0210 :Parte MS de resultado de conv.
0134 B7 02 11     STA A  0211 :Resultado de conversión A/D
0137 FE 02 10     LDX    0210 :Resultado de conversión A/D
013A A6 00        LDA A  00,X :Carga en A resultado de conv.
013C 46           ROR A       :Deja en A solo el
013D 46           ROR A       :MSB del resultado
013E 46           ROR A       :de la conversión
013F 46           ROR A       :A/D
0140 84 0F        AND A  #0F
0142 8B 30        ADD A  #30  :Convierte el núm. HEX en (A) a ASCII
0144 E6 00        LDA B  00,X :Carga en B el result. de conv.
0146 C4 0F        AND B  #0F  :Deja solo la parte menos signif.
0148 CB 30        ADD B  #30  :Conv. el núm. HEX en (B) a ASCII
```

```
014A B7 02 01     STA A 0201   :Resultado, parte más signif.
014D F7 02 03     STA B 0203   :Resultado, parte menos signif.
0150 CE 02 00     LDX   #0200 :Dirección del resultado
0153 BD E0 7E     JSR   PDATA1:Escribe dato de dir 00,X
0156 7E E0 E3     JMP   CONTROL

      * Area de Resultados y datos

0200 2B           +            :Signo (+) [2D signo (-)]
0201 00           MS           :Parte más significativa
0202 2E           .            :Punto
0203 00           LS           :Parte menos significativa
0204 20           Space        :Espacio
0205 20           Space        :Espacio
0206 56           V            :Carácter ASCII
0207 4F           O            :       "
0208 4C           L            :       "
0209 54           T            :       "
020A 53           S            :       "
020B 04           EOT          :Fin de texto.
```

Para correr el programa se substituye en el PC la dirección inicial de éste y se da el comando G:

```
* M A048
*A048 00 01
*A049 38 00
*A04A D7 (CR)
*G 5.0 VOLTS
```

5.2.- Sistema de adquisición de datos.

Un sistema de adquisición de datos tiene varios canales de entrada que permiten controlar y medir distintos paráme-- tros. En el presente caso se estudia un sólo canal de entrada. Un canal con mayor número de entradas puede implementarse fácilmente alambrando varios PIAs con una configuración como la mostrada en la Fig. 4.5 en distintas localidades de memoria.

El canal de entrada esta formado por el convertidor A/D MM5357 y el sistema completo: por el PIA y el convertidor - D/A MC1408 de 8 bits.

El MC 1408 consiste de un amplificador de corriente, un circuito en escalera R-2R y 8 interruptores (switches) de - corriente de alta velocidad Fig. 5.4.

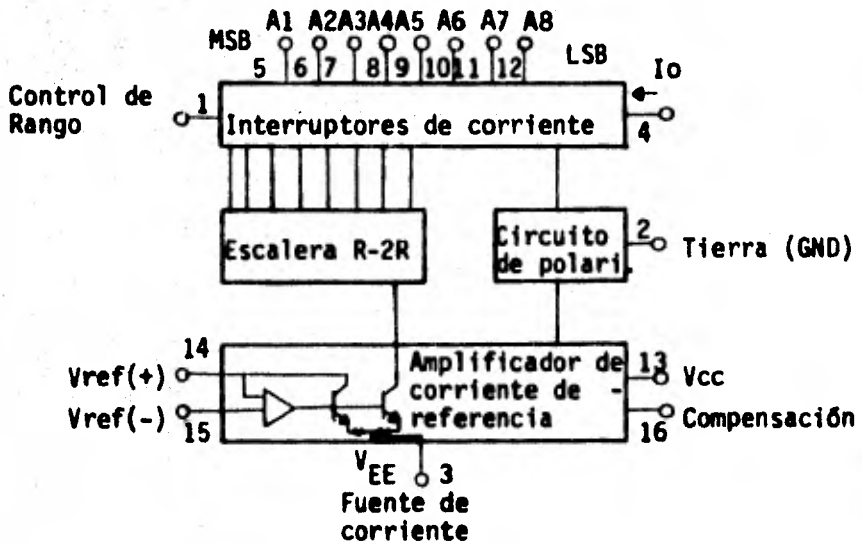


Diagrama de bloques del convertidor D/A MC1408L-8

Fig. 5.4

Las características del MC1408L-8 son;

. Precisión relativa $\pm$ 0.19%

. Entradas no inversoras compatibles con lógica MTTL y CMOS

. Voltajes de alimentación +5.0V, -5.0V y -15.0V

El circuito R-2R divide la corriente del amplificador - de referencia en componentes binarias. A la salida del con-- vertidor se tiene una corriente que es un producto lineal de una palabra digital de 8 bits y un voltaje de entrada analógico:

$$I_o = \frac{Vref}{R_{14}}\left[\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256}\right]$$

donde: $A_N = 1$ si A_N es un nivel alto.

y $A_N = 0$ si A_N es un nivel bajo.

Valores típicos son: $R_{14} = R_{15} = 1K$; $V_{ref} = +2.0V$; $C = 15pf$.

La configuración mostrada en la fig. 4.5 es para un voltaje de referencia positivo. Para manejar señales bipolares es necesario conectar R_{14} a un voltaje de referencia positivo igual al valor de entrada en la pata 15.

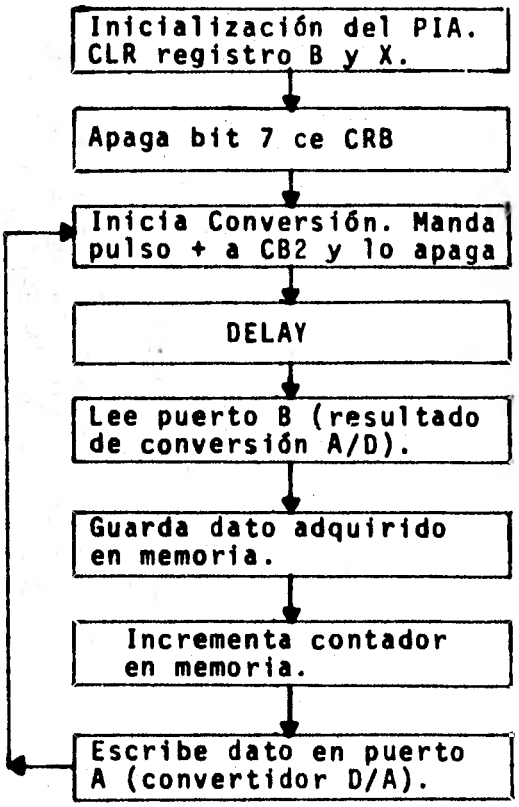
El rango de voltajes en la pata 4 esta restringido a un rango de -0.6 a 0.5 volts y la corriente máxima a 4.2mA. Para obtener un voltaje a la salida, se usa un amplificador operacional LM301, que convierte de corriente a voltaje.

El programa de adquisición de datos inicializa el PIA, apaga el bit 7 de CRB, inicia la conversión A/D mandando un pulso positivo a la pata CB2, espera que la conversión termine para leer el resultado del puerto B, guarda el dato adquirido en memoria, incrementa un contador y escribe el dato en

el puerto de datos A.

El programa permite almacenar los valores digitales correspondientes a la señal analógica estudiada.

Enseguida esta el diagrama de flujo y programa para el sistema de adquisición de datos.



Programa para guardar en memoria los valores correspondientes a una señal y mostrar esta señal a la salida de un convertidor D/A.

```
        * Inicialización del PIA

0100 4F          CLR A
0101 B7 80 09    STA A  CRA  :Limpia registro de control A
0104 B7 80 0B    STA A  CRB  :Limpia registro de control B
0107 B7 80 0A    STA A  DDRB :Establece PB0-PB7 como entradas
```

```
010A 86 F6        LDA A  #F6  :Selecciona el Reg. de recep-
010C B7 80 0B     STA A  CRB  :ción y transmisión de datos y
                              :fija el modo de control para
                              :el lado B
010F 86 FF        LDA A  #FF
0111 B7 80 08     STA A  DDRA :Establece PA0-PA7 como salidas
0114 86 04        LDA A  #04  :Selecciona el Reg. de recep-
0116 B7 80 09     STA A  CRA  :ción y transmisión de datos y
                              :fija el modo de control para
                              :el lado A
      *Limpia registro X, reg. B y apaga bit 7 de CRB

0119 CE 00 00     LDX    #0000
011C 5F           CLR B
011D B6 80 0B S2  LDA A  CRB  :
0120 84 7F        AND A  #7F  : Apaga bit 7 de CRB
0122 B7 80 0B     STA A  CRB  :

      * Inicia conversión, manda pulso + a CB2 y lo apaga

0125 B6 80 0B     LDA A  CRB  :Inicia conversión de A/D
0128 8A 08        ORA    #08  :Manda pulso positivo a CB2
012A B7 80 0B     STA A  CRB  :
012D 84 F7        AND A  #F7  :Apaga CB2
012F B7 80 0B     STA A  CRB  :

      * Lee puerto B, escribe en memoria y guarda en puerto A

0132 BD 01 60     JSR    DELAY
0135 B6 80 0A     LDA A  DDRB :Lee puerto B
0138 FE 02 00     LDX    CONT :Carga X con el contador en mem.
013B A7 00        STA A  00,X :Guarda ACCA en memoria,X
013D B7 80 08     STA A  DDRA :Guarda ACCA en puerto A
0140 5C           INC B       :Incrementa contador
0141 F7 02 01     STA B  0201 :Guarda contador en memoria
0144 7E 01 1D     JMP    S2

      * Delay

0160 CE 00 1F     LDX    #001F:Duración del Delay
0163 09       S3  DEX         :Decrementa X
0164 26 FD        BNE    S3
0166 39           RTS         :Regresa de la subrutina
```

Para correr el programa substituimos la dirección ini-- cial de éste en el PC y damos el comando G:

```
* M A048
*A048 00 01
*A049 56 00
*A04A D7 (CR)
* G
```

Después de dar el comando anterior el programa queda en un loop que guarda la información del convertidor en memoria y la muestra de nuevo a la salida del convertidor D/A.

CAPITULO VI

CONCLUSIONES.

El diseño de una microcomputadora requiere de un amplio conocimiento de sistemas lógicos digitales. Una forma de iniciar la construcción de una microcomputadora es realizando el diagrama de la memoria, lo que genera información que permite una fácil asignación de las localidades de memoria.

La arquitectura modular de la microcomputadora permite implementar fácilmente cambios en el Hardware y corregir errores del mismo, lo cual es conveniente para la enseñanza.

Como una microcomputadora depende de secuencias de estados lógicos para su funcionamiento, es de gran utilidad estudiar uno de estos sistemas con un Analizador de Estados Lógicos. Este permite encontrar y corregir errores que aparecen durante la construcción y funcionamiento de cualquier sistema digital.

Cuando ya se tiene un sistema modular las fallas en el sistema pueden también detectarse empleando el método de substitución de partes, usando la técnica de aproximaciones sucesivas es posible determinar cual es el circuito integrado que debe substituirse, empleando solo unos cuantos pasos.

El campo de los microprocesadores es un campo muy dinámico que evoluciona constantemente.

Para el diseño de una microcomputadora o de un sistema basado en un microprocesador, es importante que:

1.- El diseñador tenga un buen conocimiento de las componentes que forman el sistema.

2.- El diseñador tenga un buen conocimiento del Software del microprocesador.

3.- Se cuente desde un principio con el conjunto completo de componentes que se necesitaran para la construcción del sistema.

B I B L I O G R A F I A

(1) American Microsystems.
AMI6800 Microprocessors.
American Microsystems, Inc., febrero de 1976.

(2) Garland, Harry.
Introduction to Microprocessor System Design.
McGraw-Hill Book Company, 1979.

(3) Guerrero Guadarrama, Héctor.
Experiencias Sobre Microcomputadoras.
Tesis, Facultad de Ingenieria, 1979.
Universidad Nacional Autónoma de México.

(4) Hewlett Packard.
Operating and Service Manual 1600A Logic State Analyser.
Hewlett Packard, Agosto de 1979.

(5) Lilen, H.
Del Microprocesador al Microordenador.
Marcombo, S.A., Barcelona 1978.

(6) Malvino, Albert Paul.
Digital Computer Electronics.
McGraw-Hill Book Company, 1977.

(7) Millman, Jacob; Halkias, Christos C.
Integrated Electronics: Analog and Digital Circuits and Systems.
McGraw-Hill Kogakusha, 1972.

(8) Motorola Semiconductor.
Introduction to Microprocessors.
Motorola Inc., 1975.

B I B L I O G R A F I A

(9) Motorola Semiconductor.
M6800 Microcomputer System Design Data.
Motorola Inc., 1976.

(10) Motorola Semiconductor.
M6800 Microprocessor Programming Manual.
Motorola Inc., 1976.

(11) M. Vazquz, Ray.
Design Worksheet can generate least-part system, best adressing.
Electronics Book Series
Applying Microprocessors.
McGraw-Hill Inc., 1976.

(12) National
Linear Data Book.
National, Junio de 1976.

(13) Ogdin, Carol Anne.
Microcomputer Design.
Prentice-Hall, 1978.

(14) Peatman, John 8.
Microcomputer-based design.
McGraw-Hill Inc., 1977.

(15) Taub, Herbert; Schilling, Donald.
Digital Integrated Electronics.
McGraw-Hill Kogakusha Ltd., 1977.

(16) Texas Instruments Incorporates.
Designing with TTL Integrated Circuits.
McGraw-Hill Kogakusha Ltd., 1971.

APENDICE

CONJUNTO DE INSTRUCCIONES DEL MICROPROCESADOR MC6800

Instruction	Mnemonic	Addressing Mode: Implied OP	MC	PB	Immediate OP	MC	PB	Direct OP	MC	PB	Extended OP	MC	PB	Indexed OP	MC	PB	Relative OP	MC	PB	Boolean/Arith Operation	Condition Reg 5	4	3	2	1	0
																						H	I	N	Z	V	C
Load accumulator	LDAA				86	2	2	96	3	2	B6	4	3	A6	5	2				M → A	●	●	↕	↕	R	●
	LDAB				C6	2	2	D6	3	2	F6	4	3	E6	5	2				M → B	●	●	↕	↕	R	●
Load stack pointer	LDS				8E	3	3	9E	4	2	BE	5	3	AE	6	2				M → SP_H, (M + 1) → SP_L	●	●	↕	↕	R	●
Load index register	LDX				CE	3	3	DE	4	2	FE	5	3	EE	6	2				M → X_H, (M + 1) → X_L	●	●	↕	↕	R	●
Store accumulator	STAA							97	4	2	B7	5	3	A7	6	2				A → M	●	●	↕	↕	R	●
	STAB							D7	4	2	F7	5	3	E7	6	2				B → M	●	●	↕	↕	R	●
Store stack pointer	STS							9F	5	2	BF	6	3	AF	7	2				SP_H → M, SP_L → (M + 1)	●	●	↕	↕	R	●
Store index register	STX							DF	5	2	FF	6	3	EF	7	2				X_H → M, X_L → (M + 1)	●	●	↕	↕	R	●
Transfer accumulators	TAB	16	2	1																A → B	●	●	↕	↕	R	●
	TBA	17	2	1																B → A	●	●	↕	↕	R	●
Transfer Acc. to cond. reg.	TAP	06	2	1																A → CCR	Note 12					
Transfer cond. reg. to Acc.	TPA	07	2	1																CCR → A	●	●	●	●	●	●
Transfer stack ptr to index	TSX	30	4	1																SP + 1 → X	●	●	●	●	●	●
Transfer index to stack ptr	TXS	35	4	1																X − 1 → SP	●	●	●	●	●	●
Pull data	PULA	32	4	1																SP + 1 → SP, M_{SP} → A	●	●	●	●	●	●
	PULB	33	4	1																SP + 1 → SP, M_{SP} → B	●	●	●	●	●	●
Push data	PSHA	36	4	1																A → M_{SP}, SP − 1 → SP	●	●	●	●	●	●
	PSHB	37	4	1																B → M_{SP}, SP − 1 → SP	●	●	●	●	●	●
Add accumulators	ABA	1B	2	1																A + B → A	↕	●	↕	↕	↕	↕
Add	ADDA				8B	2	2	9B	3	2	BB	4	3	AB	5	2				A + M → A	↕	●	↕	↕	↕	↕
	ADDB				CB	2	2	DB	3	2	FB	4	3	EB	5	2				B + M → B	↕	●	↕	↕	↕	↕
Add with carry	ADCA				89	2	2	99	3	2	B9	4	3	A9	5	2				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB				C9	2	2	D9	3	2	F9	4	3	E9	5	2				B + M + C → B	↕	●	↕	↕	↕	↕
Subtract accumulators	SBA	10	2	1																A − B → A	●	●	↕	↕	↕	↕
Subtract	SUBA				80	2	2	90	3	2	B0	4	3	A0	5	2				A − M → A	●	●	↕	↕	↕	↕
	SUBB				C0	2	2	D0	3	2	F0	4	3	E0	5	2				B − M → B	●	●	↕	↕	↕	↕
Subtract with carry	SBCA				82	2	2	92	3	2	B2	4	3	A2	5	2				A − M − C → A	●	●	↕	↕	↕	↕
	SBCB				C2	2	2	D2	3	2	F2	4	3	E2	5	2				B − M − C → B	●	●	↕	↕	↕	↕
Increment	INCA	4C	2	1																A + 1 → A	●	●	↕	↕	↕	●
	INCB	5C	2	1																B + 1 → B	●	●	↕	↕	↕	●
	INC										7C	6	3	6C	7	2				M + 1 → M	●	●	↕	↕	↕	●
Increment stack pointer	INS	31	4	1																SP + 1 → SP	●	●	●	●	●	●
Increment index reg.	INX	08	4	1																X + 1 → X	●	●	●	↕	●	●
Decrement	DECA	4A	2	1																A − 1 → A	●	●	↕	↕	↕	●
	DECB	5A	2	1																B − 1 → B	●	●	↕	↕	↕	●
	DEC										7A	6	3	6A	7	2				M − 1 → M	●	●	↕	↕	↕	●
Decrement stack pointer	DES	34	4	1																SP − 1 → SP	●	●	●	●	●	●
Decrement index register	DEX	09	4	1																X − 1 → X	●	●	●	↕	●	●
Complement (1's)	COMA	43	2	1																A → A	●	●	↕	↕	R	S
	COMB	53	2	1																B → B	●	●	↕	↕	R	S
	COM										73	6	3	63	7	2				M → M	●	●	↕	↕	R	S
Complement (2's)	NEGA	40	2	1																00 − A → A	●	●	↕	↕	↕	↕
	NEGB	50	2	1																00 − B → B	●	●	↕	↕	↕	↕
	NEG										70	6	3	60	7	2				00 − M → M	●	●	↕	↕	↕	↕
Decimal adjust accumulator	DAA	19	2	1																	●	●	↕	↕	↕	↕

OP = Operation Code MC = Number of MPU Cycles PB = Number of Program Bytes

Instruction	Mnemonic	Addressing Mode: Implied OP	MC	PB	Immediate OP	MC	PB	Direct OP	MC	PB	Extended OP	MC	PB	Indexed OP	MC	PB	Relative OP	MC	PB	Boolean/Arith Operation	Condition Reg 5	H	I	N	Z	V	C
Logical and	ANDA				84	2	2	94	3	2	B4	4	3	A4	5	2				A • M → A	•	•		↕	↕	R	•
	ANDB				C4	2	2	D4	3	2	F4	4	3	E4	5	2				B • M → B	•	•		↕	↕	R	•
Inclusive or	ORAA				8A	2	2	9A	3	2	BA	4	3	AA	5	2				A + M → A	•	•		↕	↕	R	•
	ORAB				CA	2	2	DA	3	2	FA	4	3	EA	5	2				B + M → B	•	•		↕	↕	R	•
Exclusive or	EORA				88	2	2	98	3	2	B8	4	3	A8	5	2				A ⊕ M → A	•	•		↕	↕	R	•
	EORB				C8	2	2	D8	3	2	F8	4	3	E8	5	2				B ⊕ M → B	•	•		↕	↕	R	•
Shift left arithmetic	ASLA	48	2	1																A, B, M ← C b7 b0	•	•		↕	↕	↕	↕
	ASLB	58	2	1																							
	ASL										78	6	3	68	7	2											
Shift right arithmetic	ASRA	47	2	1																A, B, M → b7 b0 C	•	•		↕	↕	↕	↕
	ASRB	57	2	1																							
	ASR										77	6	3	67	7	2											
Shift right logical	LSRA	44	2	1																A, B, M → 0 b7 b0 C	•	•		R	↕	↕	↕
	LSRB	54	2	1																							
	LSR										74	6	3	64	7	2											
Rotate left	ROLA	49	2	1																A, B, M → C b7 b0	•	•		↕	↕	↕	↕
	ROLB	59	2	1																							
	ROL										79	6	3	69	7	2											
Rotate right	RORA	46	2	1																A, B, M → C b7 b0	•	•		↕	↕	↕	↕
	RORB	56	2	1																							
	ROR										76	6	3	66	7	2											
Compare accumulators	CBA	11	2	1																A − B	•	•		↕	↕	↕	↕
Compare	CMPA				81	2	2	91	3	2	B1	4	3	A1	5	2				A − M	•	•		↕	↕	↕	↕
	CMPB				C1	2	2	D1	3	2	F1	4	3	E1	5	2				B − M	•	•		↕	↕	↕	↕
Compare index register	CPX				8C	3	3	9C	4	2	BC	5	3	AC	6	2				XH − M, XL − (M+1)	•	•		↕	↕	↕	•
Test (zero or minus)	TSTA	4D	2	1																A − 00	•	•		↕	↕	R	R
	TSTB	5D	2	1																B − 00	•	•		↕	↕	R	R
	TST										7D	6	3	6D	7	2				M − 00	•	•		↕	↕	R	R
Bit test	BITA				85	2	2	95	3	2	B5	4	3	A5	5	2				A • M	•	•		↕	↕	R	•
	BITB				C5	2	2	D5	3	2	F5	4	3	E5	5	2				B • M	•	•		↕	↕	R	•
Branch	BRA																20	4	2	TEST	•	•		•	•	•	•
Branch if carry clear	BCC																24	4	2	C = 0	•	•		•	•	•	•
Branch if carry set	BCS																25	4	2	C = 1	•	•		•	•	•	•
Branch if overflow clear	BVC																28	4	2	V = 0	•	•		•	•	•	•
Branch if overflow set	BVS																29	4	2	V = 1	•	•		•	•	•	•
Branch if equal to zero	BEQ																27	4	2	Z = 1	•	•		•	•	•	•
Branch if greater or equal to zero	BGE																2C	4	2	N ⊕ V = 0	•	•		•	•	•	•
Branch if greater than zero	BGT																2E	4	2	Z + (N ⊕ V) = 0	•	•		•	•	•	•
Branch if less than zero	BLT																2D	4	2	N ⊕ V = 1	•	•		•	•	•	•
Branch if less than or equal to zero	BLE																2F	4	2	Z + (N ⊕ V) = 1	•	•		•	•	•	•
Branch if not equal to zero	BNE																26	4	2	Z = 0	•	•		•	•	•	•
Branch if minus	BMI																2B	4	2	N = 1	•	•		•	•	•	•
Branch if plus	BPL																2A	4	2	N = 0	•	•		•	•	•	•
Branch if higher	BHI																22	4	2	C + Z = 0	•	•		•	•	•	•
Branch if lower or same	BLS																23	4	2	C + Z = 1	•	•		•	•	•	•

OP = Operation Code MC = Number of MPU Cycles PB = Number of Program Bytes

Instruction	Mnemonic	Addressing Modes: Inherent OP MC PB	Direct OP MC PB	Immediate OP MC PB	Extended OP MC PB	Indexed OP MC PB	Relative OP MC PB	Boolean/Arith Operation	Condition Reg. 5 4 3 2 1 0 — H I N Z V C
Branch to subroutine	BSR						8D 8 2		
Jump to subroutine	JSR				BD 9 3	AD 8 2		See Special Operations	
Jump	JMP				7E 3 3	6E 4 2			
Return from subroutine	RTS	39 5 1							
Return from interrupt	RTI	3B 10 1							Note 10
Software interrupt	SWI	3F 12 1							S
Wait for interrupt	WAI	3E 9 1							1
No operation	NOP	02 2 1						PC + 1 → PC	
Clear	CLRA	4F 2 1						00 → A	R S R R
	CLRB	5F 2 1						00 → B	R S R R
	CLR				7F 6 3	6F 7 2		00 → M	R S R R
Clear carry	CLC	0C 2 1						0 → C	R
Clear interrupt mask	CLI	0E 2 1						0 → I	R
Clear overflow	CLV	0A 2 1						0 → V	R
Set carry	SEC	0D 2 1						1 → C	S
Set interrupt mask	SEI	0F 2 1						1 → I	S
Set overflow	SEV	0B 2 1						1 → V	S

CONDITION CODE SYMBOLS:

- **H** Half-carry from bit 3;
- **I** Interrupt mask
- **N** Negative (sign bit)
- **Z** Zero (byte)
- **V** Overflow, 2's complement
- **C** Carry from bit 7
- **R** Reset Always
- **S** Set Always
- **↕** Test and set if true, cleared otherwise
- **•** Not Affected

LEGEND:

- **OP** Operation Code (Hexadecimal);
- **MC** Number of MPU Cycles;
- **PB** Number of Program Bytes;
- **+** Arithmetic Plus;
- **−** Arithmetic Minus;
- **·** Boolean AND;
- **Msp** Contents of memory location pointed to by Stack Pointer;
- **+** Boolean Inclusive OR;
- **⊕** Boolean Exclusive OR;
- **M̄** Complement of M;
- **→** Transfer Into;
- **0** Bit = Zero;
- **00** Byte = Zero;

Note – Accumulator addressing mode instructions are included in the IMPLIED addressing.

CONDITION CODE REGISTER NOTES:

(Bit set if test is true and cleared otherwise)

1	(Bit V)	Test: Result = 10000000?
2	(Bit C)	Test: Result = 00000000?
3	(Bit C)	Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4	(Bit V)	Test: Operand = 10000000 prior to execution?
5	(Bit V)	Test: Operand = 01111111 prior to execution?
6	(Bit V)	Test: Set equal to result of N ⊕ C after shift has occurred.
7	(Bit N)	Test: Sign bit of most significant (MS) byte = 1?
8	(Bit V)	Test: 2's complement overflow from subtraction of MS bytes?
9	(Bit N)	Test: Result less than zero? (Bit 15 = 1)
10	(All)	Load Condition Code Register from Stack. (See Special Operations)
11	(Bit I)	Set when interrupt occurs, if previously set, a Non-Maskable Interrupt is required to exit the wait state.
12	(ALL)	Set according to the contents of accumulator A