UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

Una Aplicación de la Computación a la Instrumentación Programable: Analizador de Memorias Digitales.

Tesis Profesional

QUE PARA OBTENER EL TITULO DE:

F I S I C O

P R E S E N T A :

ROBERTO SERNA HERRERA

MEXICO, D. F. 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

Con el auge de la computación, la instrumentación programable ha tenido grandes adelantos. Se han desarrollado instrumentos de control, medición y prueba [A.3] [A.6] [A.8] [A.14] para muy variados fines que van desde la industria hasta el campo de la investigación pasando por la docencia. Las computadoras personales o microcomputadoras han tenido gran influencia sobre este tipo de instrumentos, muchos de ellos se construyen en tarjetas que simplemente se insertan en las ranuras de expansión de una microcomputadora (Tarjetas de conversión A/D y D/A, tarjetas para control digital, programadores de PLAs y EPROMs , etc.) quien las comanda mediante el software adecuado. Esto los hace versátiles, aunque para una aplicación en donde se requiera dedicar por completo el instrumento a una tarea específica, no es práctico, porque se están desperdiciando recursos, por lo que en muchos casos estos instrumentos se diseñan de tal forma que sean independientes de las computadoras pero que tengan forma de interconectarse con ellas (Analizadores de Estados Lógicos, Controles Digitales, Detectores de Fallas, etc). Hablando de instrumentos que dependen directamente de una computadora podemos ubicarnos en de un contexto de programación, es decir que mediante software pueden ser configurados para realizar sus tareas bajo diferentes condiciones y procesos. Si nos fijamos ahora en los instrumentos independientes de las computadoras entonces podemos hablar de una "programación" permanente del aparato vía hardware. Dentro de este contexto se puede hacer una clasificación de instrumentos atendiendo al modo de configuración: Programados y Programables. Los primeros son los que no requieren ningún software por parte del usuario para operar (este se encuentra fijo en su hardware) y los segundos necesitan de programación externa para su funcionamiento. Esta división, con todo el adelanto en electrónica, ya no es tajante y la frontera se va diluyendo.

La ventaja que tiene el uso de instrumentos programados es la facilidad de operación. Pueden tenerse en cuenta dos tipos de aplicaciones: específica y general. Digamos que para una aplicación específica no son necesarios conocimientos especiales de hardware o software, por ejemplo en el control de calidad de una línea de producción, donde un operario simplemente vigila que un instrumento "programado" indique si el artículo cumple con las normas de calidad requeridas, no así en un laboratorio donde se

requiere un instrumento que pueda ser "programado" y usado con varios objetivos.

En este trabajo, se diseñó y construyó un instrumento de prueba para Circuitos Integrados (C.I.) de la familia TTL [A.14] que se encuentra en la intersección de la clasificación hecha arriba es decir, puede operar en dos versiones: la programable y la programada. En adelante haré referencia al instrumento diseñado con la siglas PLC (Probador Lógico de Circuitos).

La idea original de este **PLC** fué de la Maestra Gertrudiz Kurtz de Delara [A.17], quién desarrolló un **PLC** que realizaba pruebas de C.I. mediante una conmutación manual de las conexiones de las terminales del Dispositivo en Prueba (en adelante DEP). Todavía bajo su dirección, en 1985, se empezó a trabajar sobre la versión automatizada de este **PLC**, es decir una conmutación programada de las señales en las entradas al DEP, así también como de la lectura de sus salidas.

Para efecto de este trabajo únicamente se consideraron como DEPs a las memorias digitales estáticas (NMOS) compatibles con la familia TTL (RAM, ROM y EPROM). Su extensión a otras aplicaciones requiere solamente la modificación del software. Visto como un controlador digital, tiene la capacidad de poder manejar 6 puertos de entradas/salidas digitales cada uno con 8 líneas (para ver la capacidad eléctrica de estas consultar el apéndice C) y usar el reloj del sistema como un temporizador con una frecuencia máxima de 1 MHz. Con el programa adecuado y los dispositivos de acoplamiento necesarios, este **PLC** puede controlar algunos experimentos.

El **PLC** no es algo nuevo dado que en la actualidad se cuenta con instrumentos de control y prueba bastante complejos pero a su vez demasiado costosos, como lo son los "Analizadores de Estados Lógicos" (Logic State Analizer) que pueden presentar en una pequeña pantalla, parecida a la de un osciloscopio, el estado lógico de todas las señales en las terminales de un DEP, pero los bits de prueba que deben ser presentados en las terminales de entradas no son generados por este tipo de aparatos, por lo tanto una prueba requiere de alambrar la lógica necesaria para la generación de ellos. La aplicación principal de estos instrumentos básicamente es en el análisis de microprocesadores [A.12], dado que por medio de estos se puede hacer el rastreo de un programa mediante su firma (ruta de ejecución). La desventaja de la mayoría de los instrumentos de prueba es su alto costo y el hecho de que sólo pueden ser usados con ese fin, a diferencia del instrumento desarrollado aquí que tiene más de una aplicación.

Para fines didácticos, el **PLC** puede ser usado en las materias relacionadas con electrónica digital o arquitectura de computadoras como Circuitos Digitales y Máquinas Digitales que imparte el Laboratorio de Cibernética de la Facultad de Ciencias, dado que su arquitectura modular sigue los principios del diseño digital basado en microprocesadores, en particular

aquí se usó el microprocesador de Motorola MC6800 y, como dispositivos periféricos, toda la familia de éste (PIAs, ACIAs, Buffers, etc), sin que ésto tenga una justificación más allá que la disponibilidad de dichos dispositivos dentro laboratorio antes mencionado. La construcción puede ser llevada a cabo usando alguna otra familia de microprocesadores. Todo el hardware fue realizado usando la técnica de Wire-Wrap que consiste en alambrar sin soldar, esto se usa esencialmente en el desarrollo de prototipos en electrónica. Cada módulo fue separado en una tarjeta, en el capítulo 2 hablaré con detalle de cada una de ellas así como del diseño en sí. Las pruebas de cada una de las tarjetas se efectuaron usando el equipo de desarrollo EXORcisar 1A de Motorola, basado en el microprocesador MC6809, que es completamente compatible con el microprocesador MC6800. El software se realizó usando el lenguaje ensamblador del MC6800 depurado con el mismo equipo de desarrollo.

CAPITULO 1
DESCRIPCION

En este capítulo haré la descripción de las características del hardware del PLC, aunque los detalles ténicos de cada una de las tarjetas los discutiré en capitulo 2. Se mencionan aqui los pasos típicos a seguir al realizar la prueba de un circuito integrado con el PLC diseñado y se hacen algunas consideraciones acerca de las familias lógicas de circuitos integrados.

1.1 Descripción Operativa del Instrumento.

La idea básica para probar un circuito de la familia TTL es determinar un patrón de niveles lógicos sobre las entradas y verificar si las salidas presentan los niveles lógicos esperados de acuerdo a las especificaciones técnicas del DEP [A.14]. Este proceso involucra una serie de bits (ceros o unos lógicos) aplicados a las entradas del DEP, el cual responde con una serie de bits en sus salidas. A estas series de bits les daré los nombres de vectores de entrada y vectores de respuesta (o vectores de salida), respectivamente. Una forma de llevar a cabo esto sería elaborar el montaje adecuado sobre un proto-board (tarjeta de pruebas) y comprobar cada uno de los vectores de respuesta usando un multímetro o mediante un arreglo de LEDs. El PLC lleva a cabo estas comparaciones en forma automatizada, compara el vector de respuesta con la respuesta esperada y determina si el DEP se encuentra operando correctamente. Tratándose de dispositivos de mediana escala de integración (MSI) la tarea de escribir los vectores de entrada y sus correspondientes vectores de respuesta puede resultar una tarea de unos cuantos minutos pero en algunos casos muy engorrosa. La situación se complica aún más cuando el DEP es un dispositivo de alta y muy alta escala de integración (LSI o VLSI: Memorias, decodificadores, PLAs, etc.) porque el número de vectores de entrada y vectores de respuesta crece radicalmente cuando se trata de hacer una prueba exhaustiva, de otra forma el usuario deberá conformarse con realizar la prueba usando solamente unos cuantos vectores de entrada tomados al azar.

El diseño del PLC se hizo en base a las necesidades planteadas en el párrafo anterior, es decir, que en forma automática (programada) presente al DEP los vectores de entrada y los compare con los vectores de respuesta esperada. Para hacerlo versátil se pensó en una estructura modular. El PLC puede operar en dos versiones: una la programable y otra la programada.

La secuencia básica para realizar una prueba deberá considerar los siguientes pasos:

Empezando preferentemente con el **PLC** apagado y contando con las especificaciones técnicas del DEP. (Por lo menos tener perfectamente localizadas las terminales correspondientes a Vcc y GND dado que estas señales deben proporcionarse externamente.):

1o. Colocar el DEP en la basa correspondiente del intrumento.

2o. Conectar Vcc y GND a las terminales del DEP.

3o. Encender el **PLC**.

4o. Una vez presentado el menú principal, escoger la opción de realizar una prueba.

5o. Con el menú de pruebas de C.I., elegir la opción correspondiente al DEP. Una mala elección provoca un mensaje de error y si la opción es válida pero no correspondiente al DEP, éste resultará defectuoso. Es conveniente, por lo tanto, si como respuesta se obtiene que el DEP no opera correctamente, realizar una segunda prueba para corroborar.

6o. Para desmontar el DEP, es conveniente apagar el **PLC** ya que el no hacerlo puede dañarlo si este resultó en buen estado durante la prueba.

1.2 Algunas consideraciones acerca de los Circuitos Integrados Digitales.

La mayoría de los sistemas digitales modernos utilizan circuitos integrados digitales debido a que producen un incremento en la confiabilidad de su operación y gran reducción en peso y tamaño. Así pues, existe un gran avance en cuanto al desarrollo de técnicas para la fabricación de los C.I.s, que van desde la baja escala de integración (SSI) hasta la muy alta escala de integración (VLSI).

Las diversas familias lógicas [L.1][L.4][L.10] caen dentro de dos amplias categorías basadas en el dispositivo principal que se usa para su fabricación. Las familias bipolares, TTL y ECL, utilizan el transistor bipolar (NPN,PNP) como elemento principal del circuito. Las familias de semi-conductores de óxido metálico (MOS) utilizan los transistores de efecto de campo (MOSFET) como elemento principal del circuito. PMOS, NMOS y CMOS son todas familias MOS.

Hoy la familia lógica TTL domina las áreas de aplicación que requieren componentes SSI y componentes MSI. Las familias MOS se adaptan mejor a las aplicaciones en donde se requiere una alta y

CAPITULO 2
HARDWARE

Habiendo hablado de las características del **PLC**, es el momento de hacer una descripción detallada del hardware. Para cada uno de las tarjetas que lo componen se hace una discusión acerca de la lógica de control y selección así como la correspondiente etapa de acoplamiento. En los apéndices (A, B y C) se da una descripción de los principales circuitos usados. Para una consulta especializada de las características de alguno de ellos en la pag. **/06**, se da la lista de los manuales usados.

2.1 Descripción del Hardware.

Como ya se ha mencionado en el capítulo anterior, el **PLC** está basado en el microprocesador de Motorola MC6800 (MPU); este es un microprocesador síncrono de 8 bits compatible con la familia TTL (apéndice A), sólo requiere alimentación de +5 volts y no necesita dispositivos externos para ser interconectado cuando ellos no sobrepasan el factor de carga (que en este caso es igual a una carga TTL). La filosofía de diseño que se siguió fué el de una arquitectura modular que permite su fácil crecimiento y que sus partes puedan ser usadas con fines didácticos haciendo modificaciones sobre ellas o creando tarjetas nuevas, esto también con la finalidad de su posible utilización con objetivos distintos de las pruebas de C.I. con el software correspondiente.

Con la visión de crecimiento es necesario que cada tarjeta del **PLC** tenga una etapa de acoplamiento que permita mantener adecuadamente las señales de los canales, porque como se mencionó, la familia de este MPU tiene un factor de carga de uno. Para ello se usaron los buffers 8T26 que son bidireccionales para el canal de datos y algunas líneas de control y los 8T97 que son unidireccionales para el canal de direcciones. Los primeros cuentan con dos líneas de habilitación y los segundos con una. Es así que cada una de las tarjetas del **PLC** cuenta con 3 integrados 8T97 y 2 integrados 8T26, que mantienen adecuadamente las señales en el canal de datos, canal de control y canal de diracciones del sistema. La lógica de habilitación de las etapas de acoplamiento cambia un poco en cada tarjeta dependiendo de cual es la tarea que realiza aunque en casi todos se hace a través de las líneas R/W (Read/Write), BA (Bus Available), VMA (Valid

memory address) y PHI2 (señal de reloj, apéndice A).

El diagrama a bloques del sistema completo se muestra en la fig. 2.1. Como se puede observar el centro del PLC es el MPU que divide sus líneas en tres grupos: canal de datos, canal de direcciones y canal de control. Estos tres grupos llegan a todo el sistema pero, como se mencionó antes, lo hacen a través de una etapa de acoplamiento.

Cada tarjeta en el sistema cuenta con una etapa de selección, esto es, se tiene una lógica que se encarga de habilitar la tarjeta cuando ésta es direccionada por el MPU para tareas específicas. Las tarjetas que tienen contacto con el exterior son la tarjeta del ACIA y la tarjeta de los PIAs. La tarjeta del ACIA se encarga de la interacción máquina-usuario, a través de ésta el **PLC** se comunica con la terminal vía el protocolo RS-232C de comunicación asíncrona serial . Es posible por lo tanto conectar cualquier dispositivo que maneje este protocolo, aunque para ésto harían falta los programas correspondientes. La tarjeta que contiene a los PIAs es otro medio de entrada-salida del **PLC**, es en esta parte donde se llevarán a efecto las pruebas de los circuitos integrados.

El **PLC** ésta constituido por 5 tarjetas que se nombran de acuerdo a la función que desempeñan: Tarjeta del MPU, Tarjeta de RAM , Tarjeta de ROM, Tarjeta del ACIA y Tarjeta de los PIAs.

Un sistema elemental basado en el microprocesador [L.5] tiene las siguientes partes:

MPU: Es quien tiene el control del sistema y es aquí donde se llevan a cabo las operaciones lógico-aritméticas y de control.

Memoria RAM: (Random Access Memory) Es la parte de la memoria del sistema en donde se almacenan temporalmente los datos de la pila del sistema, las condiciones transitorias de los programas, datos en general, etc. Esta memoria es volátil.

Memoria ROM:(Read Only Memory) Es la parte de la memoria donde se encuentran en forma permanente los programas de control del sistema. Esta memoria no es volátil.

Dispositivos de entrada/salida (I/O): A través de éstos el sistema puede comunicarse con el exterior. Es el medio por el cual se le dan instrucciones o condiciones al sistema para que realice tareas específicas y/o a través del cual se obtienen los resultados del proceso.

En la fig. 2.1 se esquematiza a bloques la configuración básica del sistema basado en microprocesador para el **PLC** diseñado en este trabajo quedando con las siguientes características: 3 Kbytes de memoria RAM, 16 Kbytes de memoria EPROM, puerto de comunicación asíncrona a través de un ACIA (MC6850) y adicionalmente tres bases de pruebas de 16, 24 y 40

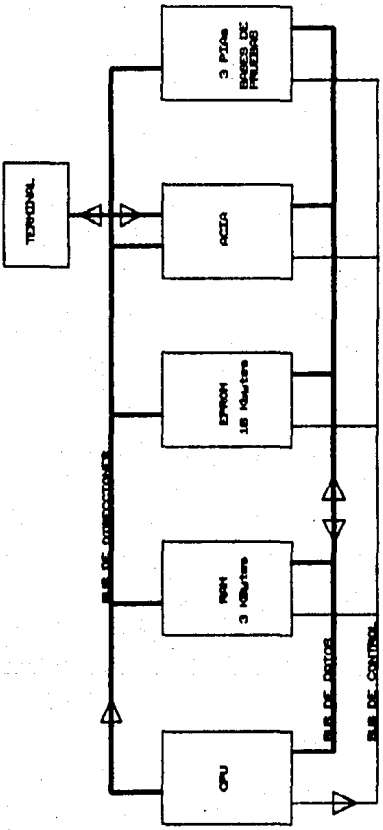


Fig. 2.1 Diagrama de bloques del PCL.

terminales conectadas a 3 PIAs (MC6821) y todo con la capacidad de expansión.

El mapa de memoria (Fig. 2.2) tiene en su parte más baja la memoria RAM, en la parte alta a la memoria ROM y en el centro los dipositivos de entrada salida, de tal forma que cualesquiera RAM o ROM pueden ser expandidas hacia el centro en posteriores modificaciones del PLC.

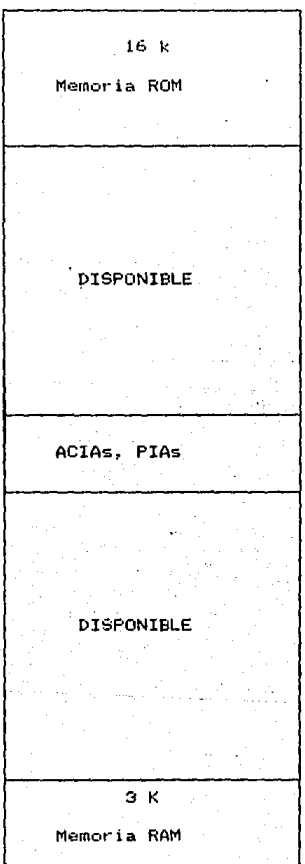


Fig. 2.2 Mapa de memoria del sistema.

El hecho de que este sistema se haya desarrollado alrededor de la familia 6800 no limita el alcance de la idea global, porque las tarjetas pueden ser conectadas, por ejemplo, al canal de una microcomputadora PC compatible mediante una lógica adicional de direccionamiento y de control ya que las líneas de datos, direcciones y control no pueden ser conectadas directamente. Esto puede realizarse en el futuro como una extensión de éste trabajo.

2.2 Tarjeta del Microprocesador.

En esta tarjeta se encuentra el microprocesador MC6800 (Ver Fig. 2.3 y Diag. 2.1). Los bloques que la constituyen son : El circuito de reloj (Diag. 2.2), el circuito de reset (Diag. 2.3), la lógica de control del MPU , los buffers de datos y los buffers de direcciones. Hay cuatro clases de señales que controlan la operación del MPU. El primer par corresponde a las señales proporcionadas por el circuito de reloj (PHI1 y PHI2). El segundo par de señales, HALT y BA, son usadas para detener la ejecución de un programa y para indicar que el canal de direcciones y el canal de datos se encuentran libres para ser usados, por ejemplo, en un DMA (Acceso Directo a Memoria). El tercer grupo de líneas controlan las señales de interrupción que hacen que el MPU responda a llamadas del exterior. En órden de prioridad tenemos: RESET, NMI (Non Mascarable Interrup) e IRQ (Interrupt Request). La señal de entrada RESET es usada para iniciar al MPU al prender el sistema o al querer reestablecerlo. Durante la secuencia de RESET todas las líneas de direcciones son forzadas a ir a un estado lógico alto. El contenido de las dos últimas localidades de la memoria ROM $FFFF y $FFFE, son cargadas dentro del Contador de Programa (PC), ésta dirección es el inicio de la rutina de atención al RESET. Durante la ejecución de esta rutina el bit de la máscara de interrupción del registro de control del MPU es encendido y debe ser borrado por una instrucción dentro del programa antes de que el MPU pueda ser interrumpido por IRQ. El circuito de RESET propociona al MPU una señal en estado bajo por un tiempo mínimo de 8 ciclos de reloj, asegurando con ésto una correcta iniciación, durante este tiempo las señales VMA y BA permanecen en bajo, el canal de datos en estado de alta impedancia, R/W en modo lectura y el canal de datos tiene la dirección del Reset $FFFE. La línea de Reset del MPU también es usada para iniciar el sistema en cualquier momento durante su operación, simplemente presionando el botón etiquetado con RESET (Diag. 2.3). El pulso de reset es completamente asincrono. Las otras dos interrupciones que puede manejar el MC6800 son el NMI e IRQ. En el diseño de este PLC no se usaron. Para una descripción de ellas ver el apéndice A. En las tarjetas de expansión estas señales pueden ser usadas dado que no fueron inhibidas dentro del diseño.

Otro bloque de esta tarjeta es el circuito de reloj. Las entradas PHI1 y PHI2 requieren pulsos cuya amplitud van desde Vss + 0.3 Volts hasta Vcc-0.3 Volts, con una frecuencia de 1 MHz. Se usaron un par de circuitos multivibradores monoestables cruzados usando los flip-flops 74LS123 y los transistores de un arreglo

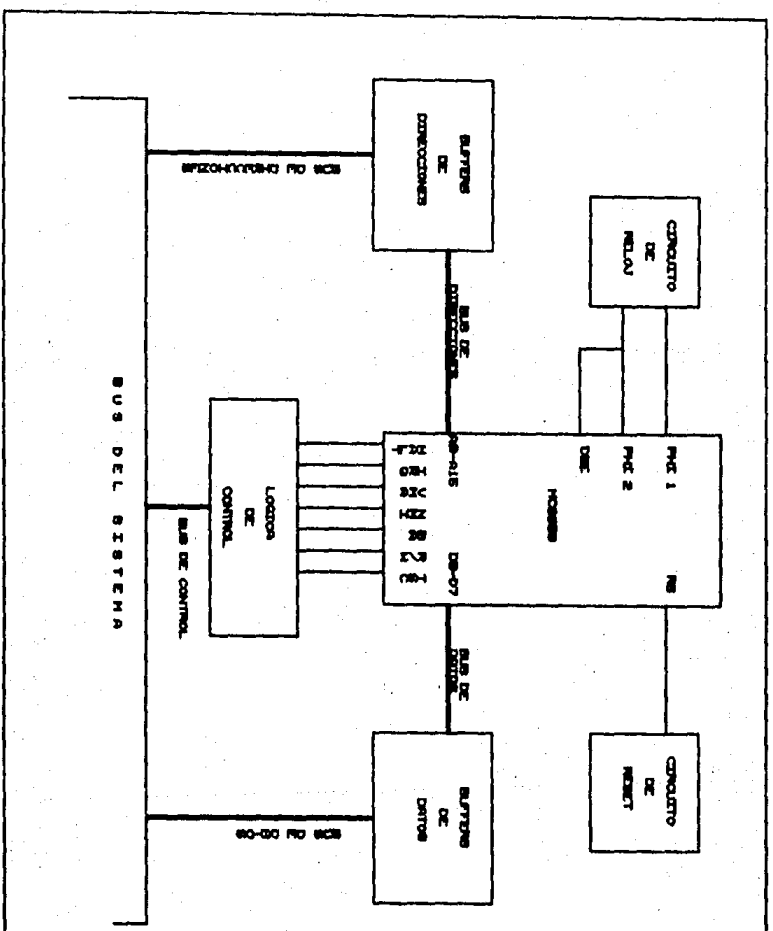
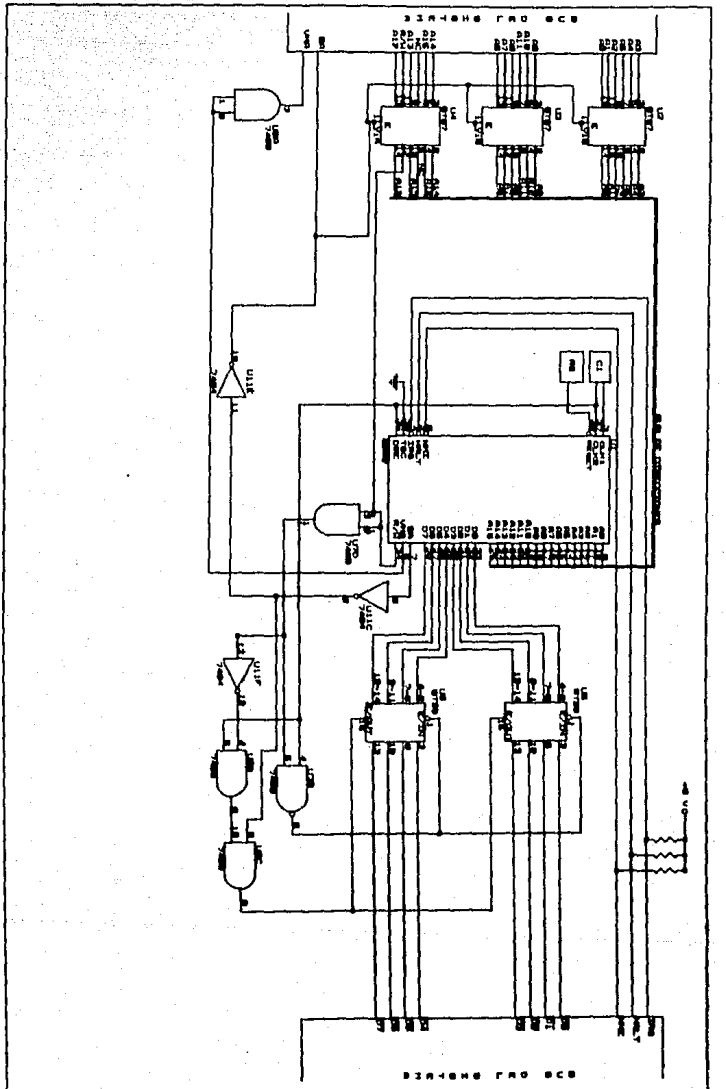
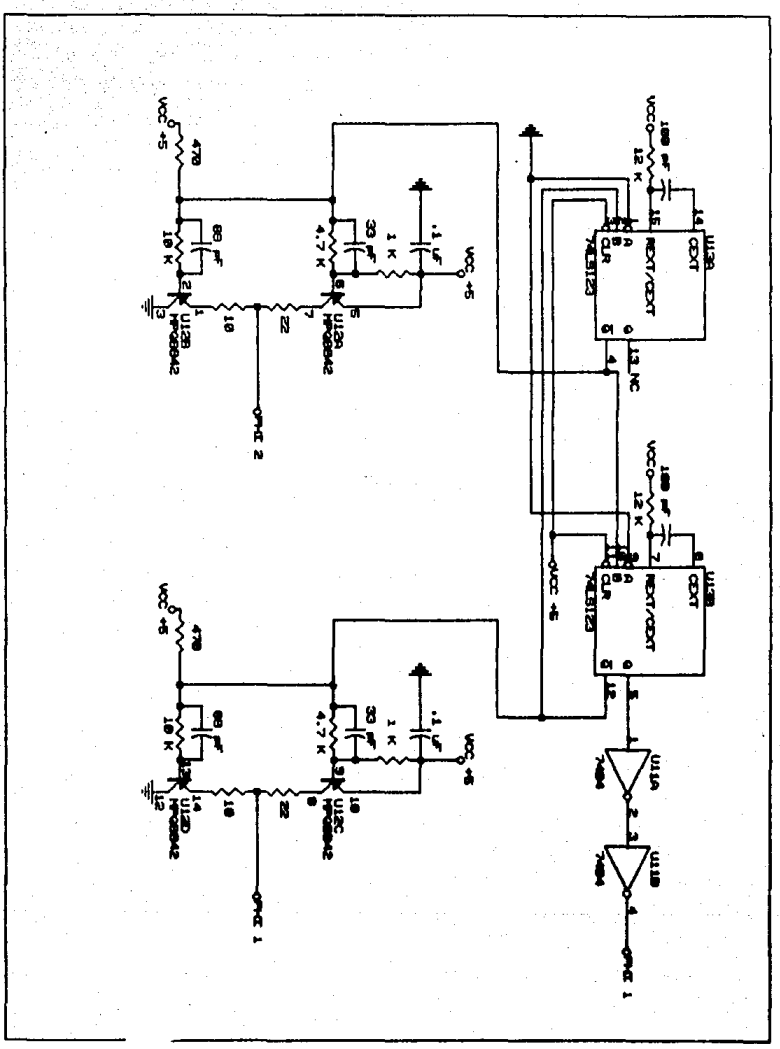


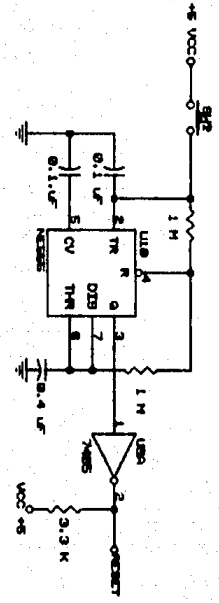
Fig. 2.3 Diagrama a bloques de la Tarjeta del MPU.



Diag. 2.1 Circuito de la Tarjeta del Microprocesador.



Diag. 2.2 Circuito del Reloj.



Diag. 2.3 Circuito del RESET.

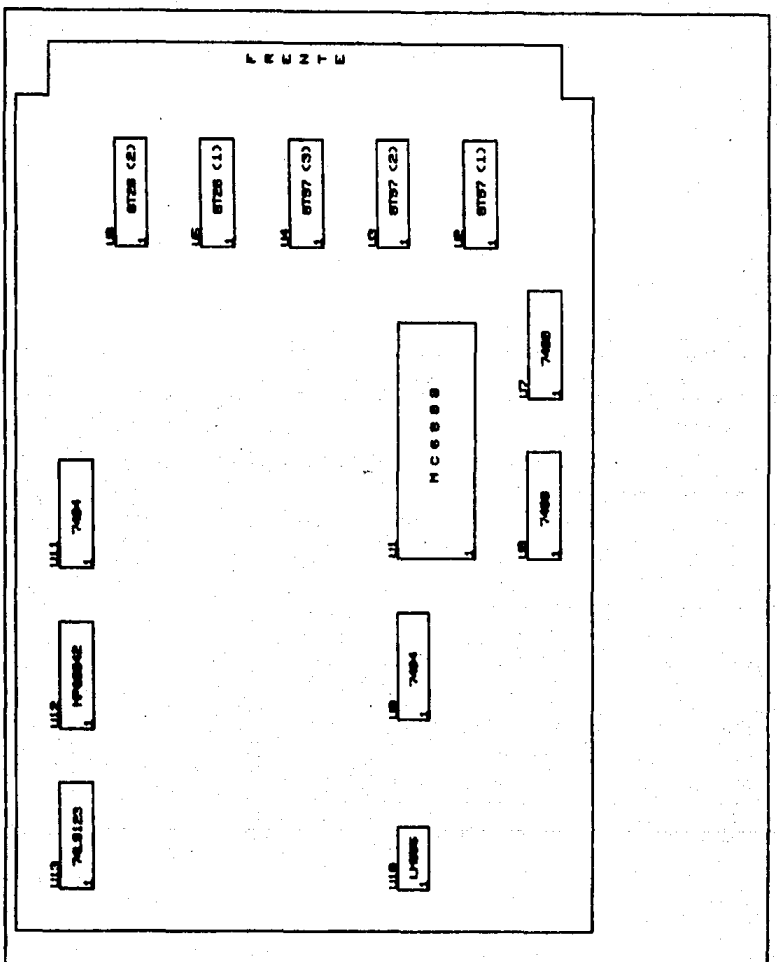


Fig. 2.4 Disposicion fisica de los componentes de la tarjeta del MPU.

2.3 TARJETA DE MEMORIA RAM.

Todo sistema basado en microprocesador requiere de un banco de memoria RAM (Diag. 2.4). La memoria RAM usada en el diseño de este **PLC** es estática y se usa esencialmente para el almacenamiento temporal de programas y datos. En esta versión del **PLC**, la memoria RAM se usa únicamente para lo segundo, dado que los programas se encuentran permanentemente grabados en otro tipo de memoria de la que hablaré adelante. Solamente se requiere almacenar temporalmente datos ya sean del sistema cuando éste se encuentra ejecutando los programas (pila del sistema) o los leídos de algún DEP. Los circuitos usados para construir el banco de memoria RAM fueron los MC2102 que son memorias RAM estáticas de 1 K x 1 bits. Se usaron éstas porque eran de las que se disponía en el momento de hacer la implementación . Estos circuitos consumen alrededor de 33 mA, debido a ésto se deben de reconsiderar en futuras expansiones o modificaciones. Finalmente el **PLC** quedó con 3 KBytes de memoria RAM, pero con la posibilidad de crecer a través de otra tarjeta. Se usaron 24 C.I. para armar los 3 KBytes, usando 8 para cada uno, los cuales son habilitados usando un selector de línea 3 a 8, el 74LS138. La habilitación del selector de línea y por lo tanto del bloque de memoria se hace a través de la función lógica

$$VMA \cdot PHI2 \cdot \overline{(A13 + A14 + A15)}$$

A las líneas de entradas codificadas (A,B Y C) del selector de línea (Diag. 2.4) llegan las líneas de direcciones A10, A11 y A12 que direccionarán cada KByte de la memoria RAM (ver el mapa de memoria Fig. 2.2), usando las correspondientes líneas de salida del selector de línea Y0, Y1 y Y2 de acuerdo a la siguiente tabla:

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X
2	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X
3	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X

KByte

Todos los C.I. de memoria reciben la señal R/W del MPU para efectuar la lectura o escritura de datos. Las terminales 11 y 12 de las memorias 2102, que son el data input y data output respectivamente van unidos dado que una operación de lectura nunca se realizará al mismo tiempo que una de escritura y viceversa. Al igual que las otras tarjetas, ésta también tiene una etapa de acoplamiento para el canal de datos y otra para el canal de direcciones. El canal de datos se habilitará con la función lógica

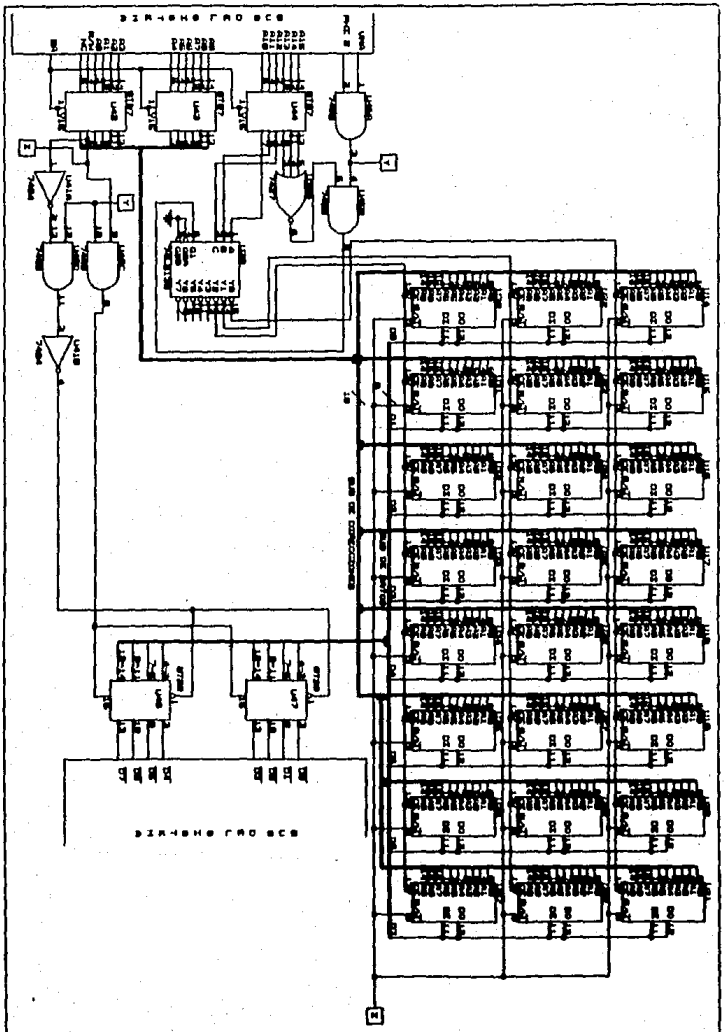
$$VMA \cdot PHI2 \cdot R/W$$ para entrada

esta señal va a la terminal 1 de los C.I. 8T26,

y

$$VMA \cdot PHI2 \cdot \overline{R/W}$$ para salida

esta señal va a la terminal 15 de los C.I. 8T26, y el canal de direcciones se habilita con la línea de control BA conectada a las terminales 1 y 15 de los C.I. 8T97. La disposición física de los elementos de esta tarjeta se muestra en la figura 2.5.



Diag. 2.4 Circuito de la Tarjeta de RAM.

2.4 Tarjeta de Memoria EPROM.

En la mayoría de los sistemas basados en microprocesador, la memoria ROM (Read Only Memory, Memoria Unicamente de Lectura), contiene los programas de control y generalmente son unos cuantos KBytes. Sin embargo en el diseño de este **PLC** se dipuso de 16 KBytes de memoria EPROM (Diag. 2.5), con la posibilidad de crecer, dejando fijos en ellos los programas y las tablas de pruebas. Se usaron 8 C.I. MC2716 que son memorias EPROMs de 2 048 x 8 Bits (Erasable Programmable ROM) borrables y programables eléctricamente , no-volátiles. Con estas memorias se tiene la posibilidad de borrar la información almacenada en ellas, usando una lámpara de luz ultravioleta y reprogramar usando un instrumento conocido como quemador o programador de EPROMs. La parte más alta de la memoria EPROM es usada para el programa monitor que es el que se encarga del control del **PLC** y los programas que se encargan de la interacción con el usuario. El bloque de memoria correspondiente a EPROM comprende de la dirección $C000 a la $FFFF (fig. 2.2). Aquí el único problema que se presenta es el hecho de que se requiere el programador de EPROMs para que el usuario pueda grabar sus propios programas de prueba.

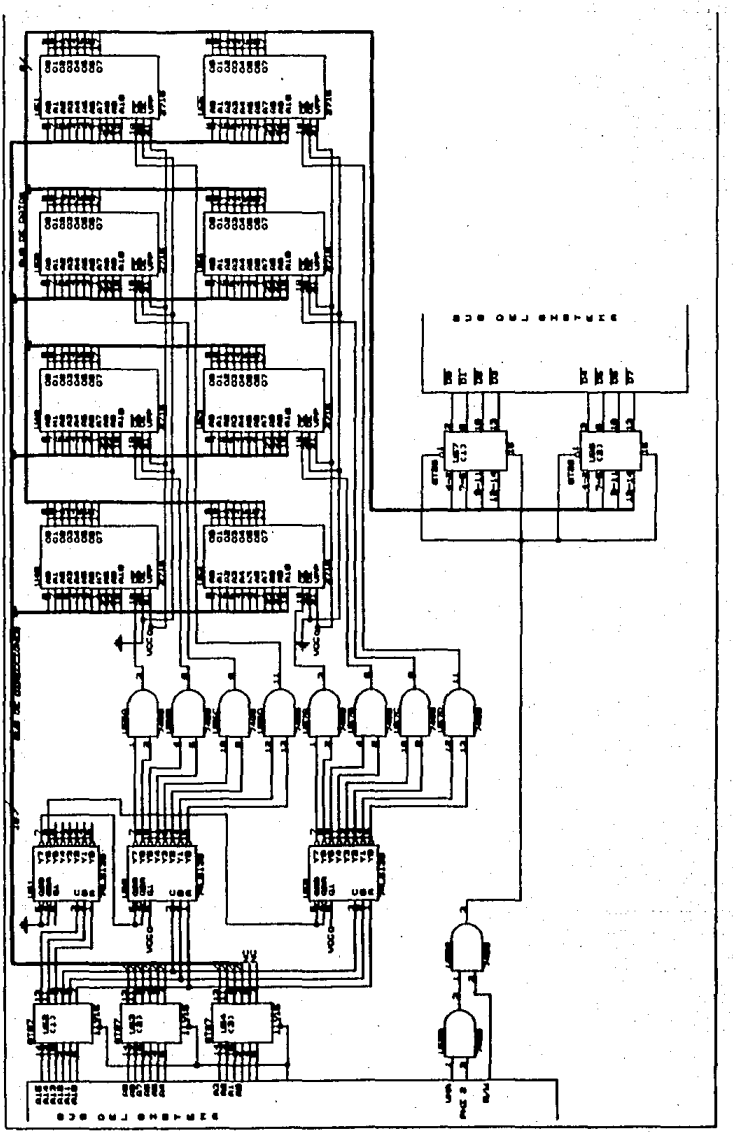
Al igual que en todas las tarjetas, también se cuenta con dos etapas de acoplamiento, una para el canal de datos y otra para el canal de direcciones. Aunque la lógica de selección varía un poco. Para el canal de datos sólo se considera la salida ya que a esta tarjeta no se puede escribir información, entonces la habilitación del canal de datos se hace a través de la función lógica

$$VMA \cdot PHI2 \cdot R/W$$

que va a las terminales 1 y 15 de los C.I. 8T26. La habilitación del canal de direcciones se sigue haciendo a través de BA.

La decodificación de direcciones se hace en forma total para dejar abierta la posibilidad de mapear nuevos dispositivos en memoria fuera de estas zonas sin tener conflicto por repeticiones. Las líneas de direcciones desde A0 hasta A10 llegan después de haber pasado por la correspondiente etapa de acoplamiento a todos los EPROM 2716. La decodificación se hace cuando las líneas A11 hasta la A15 se encuentran en estado alto, y las líneas de direcciones restantes seleccionan la localidad particular de los EPROMs, logrando con ésto una decodificación total. La lógica usada para esto permite seleccionar bloques de 1 KByte y se realizó con 3 selectores de línea 3 a 1 (74LS138). Si la entrada de habilitación está inactiva, todas las salidas de este circuito están inactivas y si la entrada de habilitación está activa solamente una salida está activa y ésta corresponde al número codificado en BCD que se presente en las entradas A,B y C. El circuito U12 (74LS138) tiene en sus entradas codificadas a las líneas A15,A14 y A13 de tal forma que éste sirve para seleccionar 8 bloques de 8 KBytes, con sus líneas de salida, ésto cubre los 64 Kbytes que el MPU es capaz de direccionar

directamente. De sus líneas de salida se tomaron Y7 y Y6, es decir, la parte más alta de la memoria para zona de EPROM; estas líneas se usan a su vez para habilitar otros dos C.I. 74LS138, los cuales tienen en sus líneas de entrada codificadas las líneas de direcciones A13, A12 y A11, logrando seleccionar así, con cada línea de salida de los dos selectores de línea, bloques de 1 Kbyte cada una, o sea, los 16 Kbytes que conforman toda la zona de EPROM. Cada pareja de líneas de salida de ellos , llegan a una compuerta AND cuya salida se usa para seleccionar bloques de 2 KBytes, es decir, a cada uno de los C.I. MC2716 . Las líneas de salida de las compuertas ANDs llegan directamente a la terminal 18 (Chip Select) y la señal de Vpp terminal 21 (Voltaje de Programación), se encuentra directamente conectada a Vcc. La línea OE (Output Enable) del C.I. 2716 va conectada directamente a tierra para tener permanentemente habilitadas las salidas. La disposición física de los componentes de esta tarjeta se muestra en la figura 2.6.



Diag. 2.5 Circuito de la Tarjeta de EPROM.

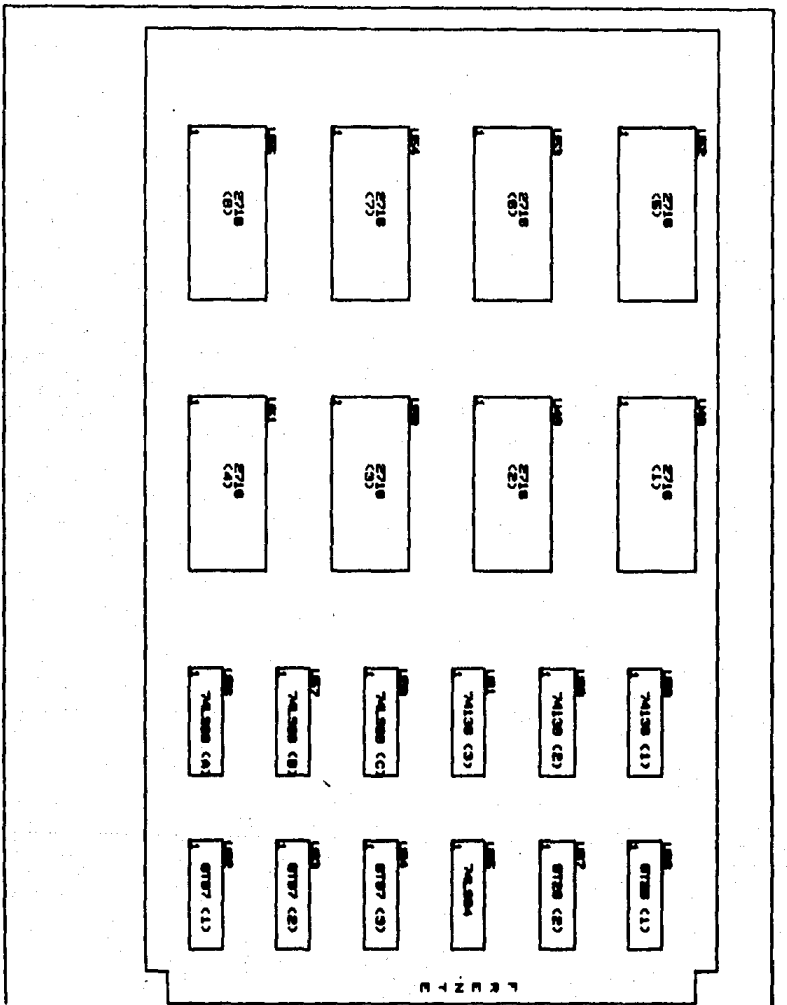


Fig. 2.6 Disposición física de los componentes de la tarjeta de EPROM.

2.5 Tarjeta del ACIA. (Comunicación Asincrona)

Este PLC necesita que el usuario eliga de un menú el tipo de C.I. a probar y, además se requiere mostrar información al usuario. Es necesario, por lo tanto, un medio de comunicación usuario-máquina. El ACIA o Adaptador de Interconexión de Comunicación Asincrona, permite el intercambio de información entre el PLC y el usuario (Diag. 2.6). Es un medio de presentación y captura que puede usar distintos mecanismos para ello. El que se usó aquí fué una terminal VISTAR, con puerto de comunicación serie RS-232C, éste es un protocolo de comunicación que usan la mayoría de las computadoras para la transferencia de información ya sea a otras máquinas o a dispositivos periféricos. Para mayor detalle en el funcionamiento del ACIA ver el apéndice B. La decodificación de direcciones para la habilitación del ACIA se hace en forma total a través de compuertas NOR usando todas las lineas de direcciones excepto A0 y A15 que se usan directamente sobre las entradas de él, A0 va a la terminal 11 RS (Register Select) por medio del cual se selecciona el registro del ACIA a usar, y A15 va a la terminal 8 (CS0, Chip Select 0), el CS1 esta conectado a Vcc y el CS2 recibe la linea que resulta de la decodificación de todas las líneas de direcciones restantes. Por la naturaleza de la comunicación serie es necesario un reloj para la transmisión y otro para la recepción. En este diseño se usó el mismo reloj para los dos fines. La parte de reloj fué implementada usando un temporizador NE555 en una configuración aestable con resistencias variables que permiten su ajuste (R1, R2). La línea de R/W llega directamente al ACIA. La naturaleza asíncrona del ACIA reside en que puede mandar o recibir información en cualquier instante y la transferencia sólo está condicionada a que el PLC tenga el programa adecuado que maneje este flujo de información ya sea por búsqueda o por interrupciones. La técnica usada aquí es la de búsqueda. El ACIA cuenta también con varias líneas de Hand-shake pero en este diseño no fueron usadas: DCD (Data Carrier Detect), CTS (Clear-to-Send) y RTS (Request-to-Send); debido a que hay un programa que se encarga de administrar el flujo de información. Así pues, al exterior sólo salen tres líneas: TxDATA (Transmit Data), RxDATA (Receive Data) y GND (Tierra). Las dos primeras líneas se conectan hacia el exterior mediante unos convertidores de niveles TTL a niveles de RS-232: estos integrados son el MC1489 y el MC1488. La diferencia consiste en que los niveles TTL son de 0 a 5 volts y los niveles RS-232 van de -12 a 12 Volts. El 1489 convierte TTL en RS-232 y el 1488 en forma inversa. Se usa un conector DB-25 hembra conectando el TxDATA a la terminal 3 y el RxDATA a la terminal 2, la tierra va a las terminales 1 y 7.

La velocidad de transmisión se fijó en 1200 bauds, con el siguiente formato de palabra : 1 bit de start, 7 bits de datos, dos de Stop y un bit de paridad

start	d1	d2	d3	d4	d5	d6	d7	d8	stop	stop

Fig. 2.10 Formato de Transmisión de un BYTE.

Los datos llegan en paralelo del MPU al ACIA , esta internamente los convierte a un formato en serie para su transmisión (apéndice B), y el ACIA entrega en paralelo al MPU la información que recibe en serie. Eventualmente ocurren errores de transmisión ya que debido a la implementación del reloj, éste es poco preciso. Lo ideal aquí hubiera sido usar un cristal como oscilador pero por la falta de éste se tuvo que optar por un oscilador basado en el temporizador NE555. Generalmente los errores ocurren en la transmisión de información a desplegar en la ,terminal y no en la recepción. La disposición física de los componentes se muestra en la figura 2.7.

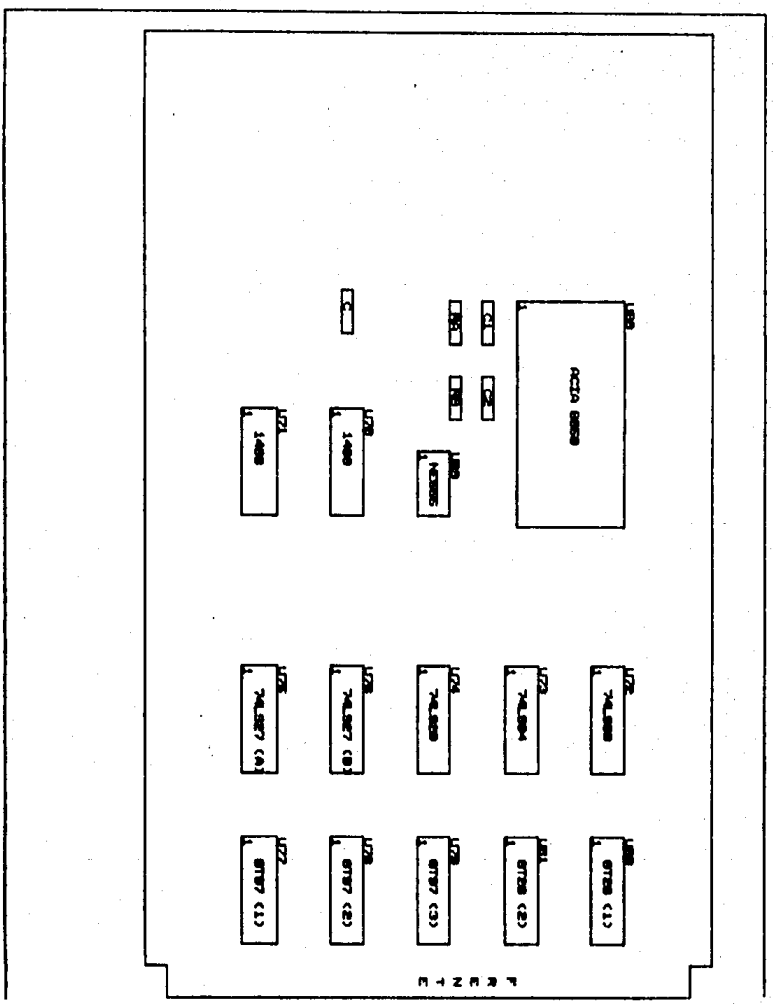


Fig. 2.7 Disposición física de los componentes de la tarjeta de ACIA.

datos son transferidos o recogidos de sus puertos a través del registro de salida de datos . De esta manera, para la transferencia de datos a través de el PIA, sólo es necesario escribir las palabras apropiadas en los registros adecuados. Para más detalles acerca del funcionamiento y programación de el PIA ver el apéndice C.

Esta tarjeta cuenta también con dos etapas de acoplamiento. La habilitación para el acoplamiento del canal de datos se hace en la misma forma que en la tarjeta de memoria RAM. El canal de direcciones se encuentra permanentemente habilitado al dejar las líneas de habilitación de sus buffers en nivel lógico bajo. La decodificación de direcciones se hace nuevamente en forma total para evitar conflictos con futuras tarjetas.

Los PIAs son seleccionadas a través de tres líneas de selección (Chip Select) CS0, CS1 y CS2. Al CS0 de las tres PIAs llega directamente la línea de dirección A15, con ésto, los PIAs se seleccionan cuando A15 esta en nivel alto. Al CS1, también de las tres PIAs, llega la decodificación de direcciones de la función

$$A3 \cdot A7 \cdot A8 \cdot A9 \cdot A10 \cdot A11 \cdot A12 \cdot A13 \cdot A14.$$

La línea de habilitación CS2 se obtiene de un selector de línea 3 a 8 (74LS138), a cuyas entradas codificadas llegan A4, A5 y A6, y es habilitado por A2 cuando ésta se encuentra en estado bajo. A la entrada CS2 de el PIA 1 llegan la salida Y1 del decodificador, a el PIA 2 llega Y2 y a el PIA 3 llega Y4.

Con todo lo anterior los PIAs se seleccionan de acuerdo a la tabla:

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	X	X
2	1	0	0	0	0	0	0	0	0	0	1	0	0	0	X	X
3	1	0	0	0	0	0	0	0	0	1	0	0	0	0	X	X
PIA																

X = No importa

A0 y A1 no importan para la selección del chip, éstas sirven para la selección de los registros internos de los PIAs. De acuerdo a ésto tenemos a los PIAs en las direcciones: PIA 1 en $8010, PIA 2 en $8020 y PIA 3 en $8040.

Del canal de control los PIAs reciben directamente en sus entradas de habilitación a PHI2, R/W y a la señal del RESET.

Las líneas de control de interrupciones CA1 y CA2 se conectaron a Vcc, de esta manera las interrupciones se encuentran deshabilitadas para los PIAs y, en consecuencia, las líneas CA2, CB2, IRQA e IRQB se encuentran desconectadas.

Las líneas de cada uno de los puertos de los PIAs se encuentran distribuidas en cada una de las bases de la siguiente

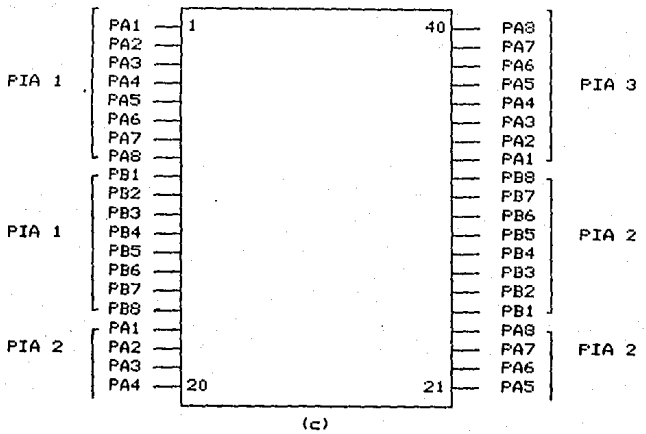


Fig. 2.8 Asignación de los puertos de los PIAs a las bases de pruebas.

La disposición física de los componentes de esta tarjeta se muestra en la figura 2.9.

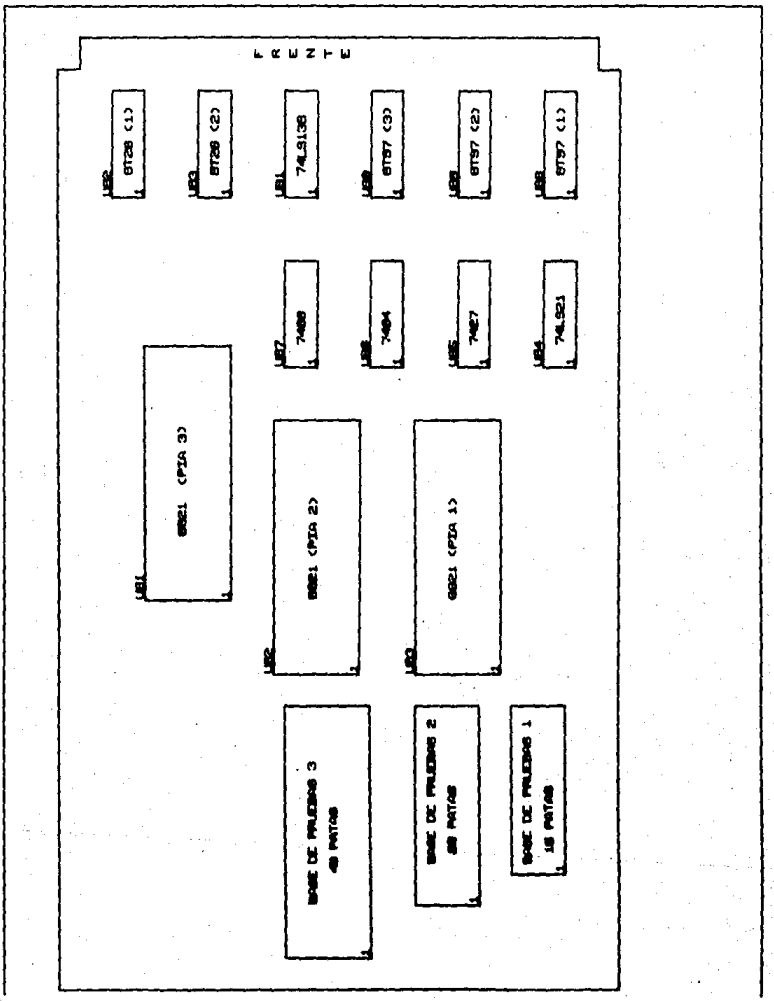


Fig. 2.9 Disposición física de los componentes de la tarjeta de los PIAs

CAPITULO 3
SOFTWARE

El software es una componente importante del **PCL**, ya que es quién se encarga de presentar la secuencia de vectores de entrada para realizar una prueba, así como de la comparación de los vectores de respuesta con los vectores de respuesta esperada y decide si el DEP está operando adecuadamente. Es responsabilidad del programador la correcta introducción de estos. La secuencia básica para introducir un programa de prueba es: Desarrollar el programa en el lenguaje ensamblador del MC6800 (puede usarse cualquier emulador de él), localizarlo dentro del mapa de memoria del **PLC** y pasarlo a una memoria EPROM MC2716. A continuación se dan algunos ejemplos de estos programas y se describe el que se encarga de controlarlos: El Programa Monitor.

3.1 Rutina de Reestablecimiento. (RESET)

Esta rutina se encarga de iniciar todos los componentes del **PLC** (MPU, ACIA y PIAs), así como de la administración del software de prueba. Es el procedimiento por medio del cual el **PLC** espera y ejecuta las tareas escogidas por el usuario, según el modo de operación del **PLC** (programada o programable).

Al encender el **PLC** o al presionar el botón de RESET, el MPU presenta en el primer ciclo de máquina la dirección $FFFF y $FFFE en el siguiente. Es en estas direcciones donde se encuentra, dentro de la memoria EPROM, el vector de RESET. Con lo anterior se almacena dentro del registro interno PC (Program Counter) la dirección de la rutina de atención al RESET.

Toda llamada a inicio (presionando el botón de RESET), hace uso de la rutina de Reestablecimiento cuyo diagrama de flujo se presenta en la figura 3.1. Primeramente se carga en el registro de Apuntador de Pila (Stack Pointer) la dirección inicial de la memoria RAM asignada a la pila del sistema, con esto se asegura que cuando el programa monitor haga llamadas a subrutinas o uso de la pila del sistema esta no va a interferir con la memoria RAM del usuario. Enseguida es necesario iniciar el protocolo de comunicación con la terminal para la interacción **PLC**-usuario lo cual se hace llamando a la subrutina INACI, es aquí donde se inicia el ACIA, escribiendo a sus registros de control la información necesaria para la correcta transferencia de información de y hacia el exterior. Esto se hace mediante un

inicio maestro al ACIA y escribiendo a su registro de control el dato $11 que corresponde a una selección de palabra de transmisión y recepción con las siguientes características: 8 bits de datos, 2 bits de stop, sin paridad y un modo de división de reloj entre 16. Antes de poder entrar a la secuencia de interacción todavía hace falta iniciar los PIAs, rutina INPIA, que son las que se encargan de traducir los vectores de prueba de ceros y unos a niveles de voltaje aplicables a las entradas del DEP. Inicialmente todos los puertos de los PIAs se programan como entradas para prevenir cualquier conflicto con dispositivos que pudieran estar puestos sobre las bases de pruebas. Con todo lo anterior el control se pasa al programa monitor.

3.2 Programa Monitor.

Una vez establecido el protocolo de comunicación es necesario mostrar información para que el usuario pueda iniciar una sesión de prueba. Todo mensaje que se muestra en la terminal hace uso de la rutina PRMEN cuyo parámetro de entrada es un vector en el registro de índice X que indica el inicio del texto a mostrar. Inicialmente se muestra información acerca del modo de operación actual del PLC así como la versión del programa monitor en uso. A continuación se muestra el primer menú cuyas opciones consisten en:

1a. Indicación de estado del sistema
2a. Iniciar una prueba.

Con la primera opción se tiene la capacidad de presentar las condiciones de la prueba inmediata anterior antes de iniciar una prueba nueva. La segunda opción presenta el menú de los circuitos integrados factibles a ser probados por la existencia dentro de los EPROMs, de los programas correspondientes. La elección de cada uno de ellos se hace mediante el número correspondiente usando la rutina LEECOM. Como ya se mencionó anteriormente este menú puede ser ampliado por el usuario.

El programa monitor entra a un ciclo permanente: Menú-Pruebas-Menú, del cual sólo puede ser sacado por una llamada a RESET, dado que hasta este punto la tarea del PLC es únicamente la de probar circuitos. Mediante el software adecuado este hardware puede ser utilizado como un controlador digital.

3.3 Rutinas del Programa Monitor.

3.3.1 PRMEN(X;): (PResenta MENsajes Fig. 3.2) Como ya se mencionó arriba, esta rutina tiene como finalidad la de mostrar en la terminal una cadena de caracteres ASCII que estan en memoria apuntados por la dirección cargada en el registro de índice X y cuyo final está determinado por el caracter ASCII "@". Mediante la rutina EBYTE , se manda inicialmente un retorno de carro (CR, $0A) y otro cada vez que dentro del texto aparace el caracter "|" el cual indica un fin de linea. Cada linea nueva se inicia con un espacio en blanco. Se escriben todos los caracteres en la pantalla localizados mediante un

direccionamiento indicado para lo cual se va incrementando el registro de índice X.

3.3.2 EBYTE(B;): (Escribe **BYTE** Fig.3.3a) Esta rutina escribe un byte a la pantalla, el cual es pasado como parámetro en el acumulador B. El ACC A es almacenado en la pila para no perder la información contenida en ese momento en él, dado que el acumulador es usado para otras tareas en esta rutina. Dentro de él se carga el estado del ACIA y si el registro de control de ésta indica que el registro de transmisión se encuentra vacio entonces se inicia la transmisión del byte almacenado en el ACC B. Si no es así entonces se espera a que dicha condición se satisfaga. Si ocurre algún error dentro de la transmisión entonces el dato se vuelve a mandar. Una vez completada la transmisión del byte en forma exitosa se recupera de la pila, el contenido del ACC A.

3.3.3 LEECOM(;): (**LEE COM**ando Fig. 3.4) No requiere parámetros y sirve para leer de la terminal usada un byte correspondiente a la elección de comando hecha por el usuario. Primeramente se lee un byte de la terminal usando la rutina LBYTE (descrita abajo). Si el byte leído corresponde al caracter "1" entonces se muestra el menú de circuitos mediante la rutina MENUCI (descrita abajo), si el caracter es "2" entonces se despliega el estado del sistema usando la rutina ESTADO. Si no es ninguno de los anteriores entonces el caracter leído es no válido por lo que se imprime un mensaje de error y se vuelve a mostrar el menú inicial hasta que se tenga una elección válida ("1" o "2").

3.3.4 LBYTE(;B): (Lee **BYTE** Fig.3.5a) Lee un byte de la terminal y lo guarda en el ACC B, que es un parámetro de salida de esta rutina, para su posterior uso dentro de otras rutinas. Inicialmente se comprueba si el registro de recepción del ACIA se encuentra vacío para poder comenzar con la captura, si es así, entonces de dicho registro se toma la información para almacenarla en el ACC B. Si no, entonces se espera a que la condición de registro de recepción vacío sea verdadera. Después se verifica si el dato ha sido recibido completo, si no espera a que sea terminado de recibir. Una vez recibido completamente se elimina el bit más significativo de él, dado que para el código ASCII (no extendido de 7 bits) no es necesario.

3.3.5 MENUCI(;): (**MENU** de **CI**rcuitos Fig. 3.5b) Con esta rutina se presenta el menú de circuitos integrados que pueden ser probados dentro de la versión en operación del **PLC.**

Se despliega el texto correspondiente al menú de circuitos usando la rutina PRMEN con X=MENUIN. En la pantalla de la terminal aparece lo siguiente:

```
MENU DE CIRCUITOS INTEGRADOS
QUE PUEDEN SER PROBADOS:
#    NOMENCLATURA      TIPO DE DISPOSITIVO
0    2102              RAM 1 024 X 1 BITS
1    6810              RAM 256 X 1 BITS
2    2716              EPROM 2 048 X 8 BITS
3    74112             FLIP-FLOP J-K

MEMORIA A PROBAR ? NUMERO = _
```

La opción se toma mediante la rutina OPCION descrita a continuación.

3.3.6 ESTADO(X;): (**ESTADO** Fig. 3.5c) Se despliega información de la prueba inmediata anterior, si la hay. La información se encuentra a partir de la dirección contenida por el Registro de Indice que se pasa como parámetro.

3.3.7 OPCION(;): (**OPCION** Fig. 3.6) Esta rutina toma de la terminal el número de circuito que aparece dentro del manú de circuitos y que haya sido seleccionado por el usuario. Si la elección hecha corresponde a una opción inválida se envia a la pantalla un mensaje de error y se pide nuevamente el dato. Si la elección es válida entonces se procede a ejecutar la rutina correspondiente. Todas las rutinas están numeradas de acuerdo al orden en el que aparecen dentro del menú de circuitos (CERO,UNO,DOS,TRES descritas abajo) para que cualquier ampliación futura hecha por el usuario sea fácil de integrar al contexto.

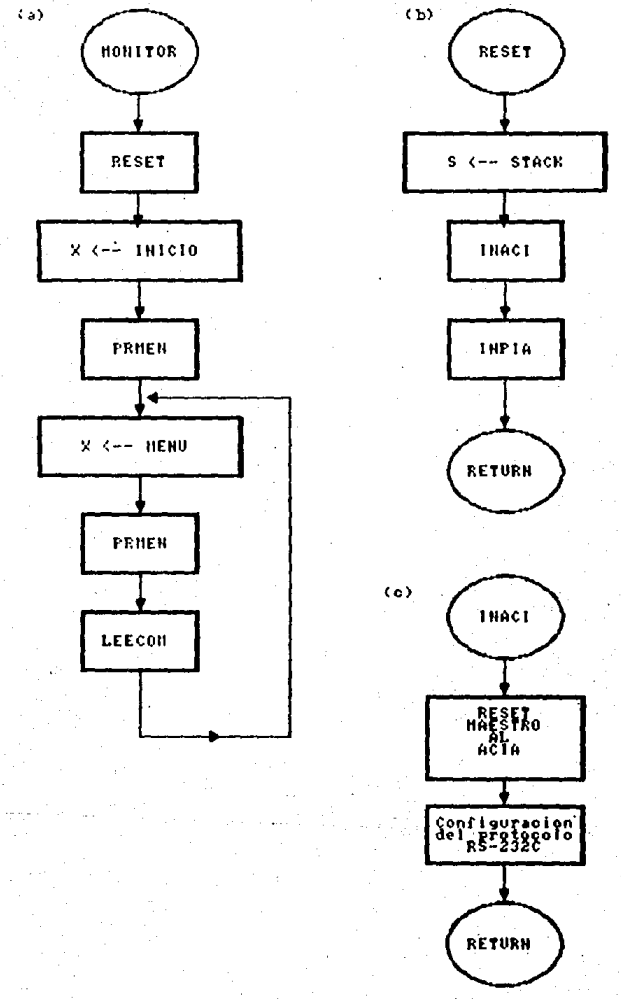


Fig. 3.1 (a) Programa Monitor
(b) Rutina de Atencion al RESET
(c) Rutina INACI, inicia el ACIA

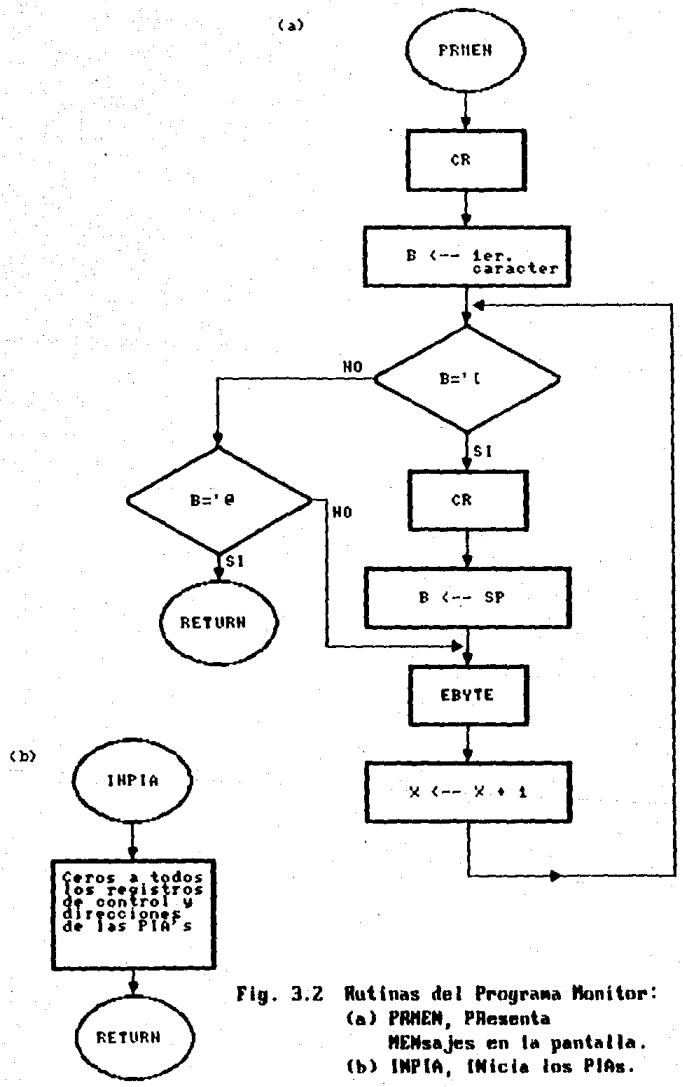
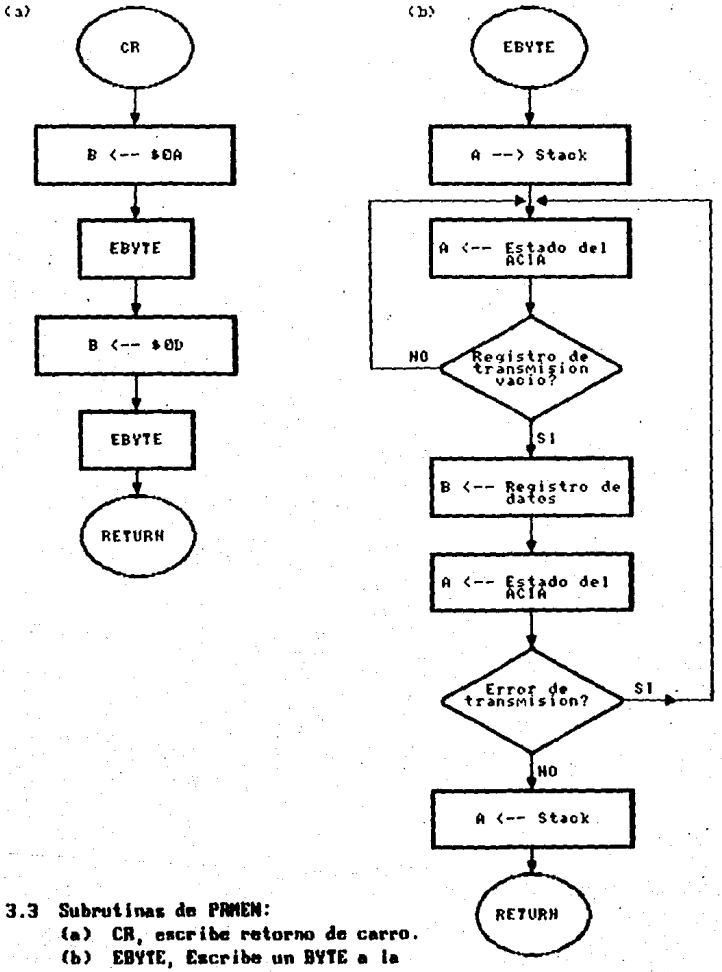


Fig. 3.2 Rutinas del Programa Monitor:
(a) PRMEN, PResenta MENsajes en la pantalla.
(b) INPIA, INicia los PIAs.



3.3 Subrutinas de PRMEN:

(a) CR, escribe retorno de carro.

(b) EBYTE, Escribe un BYTE a la pantalla.

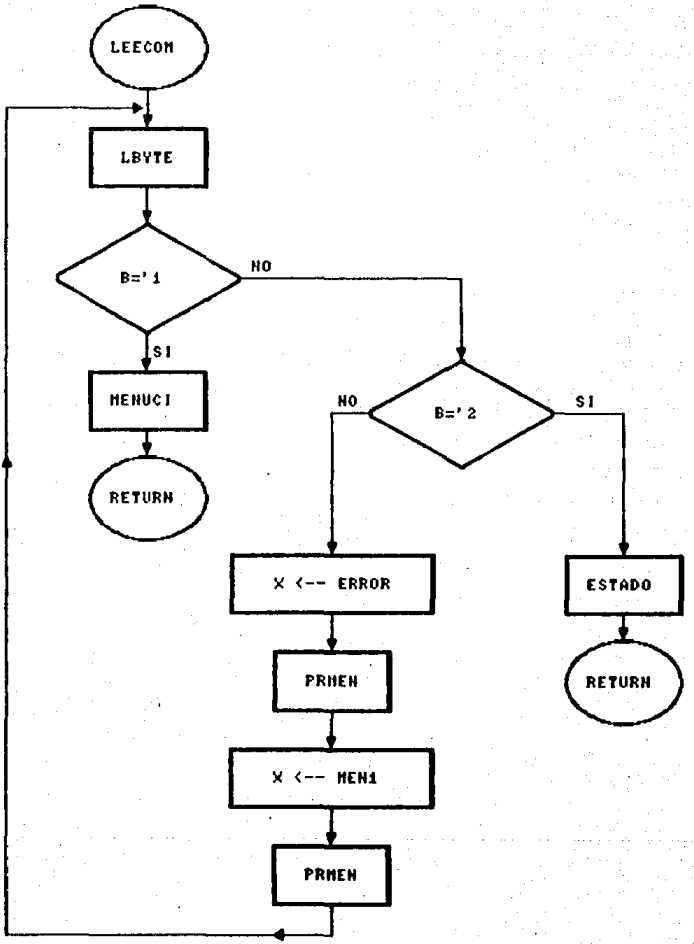


Fig. 3.4 Rutina LEECOM, LEE Comando del teclado.

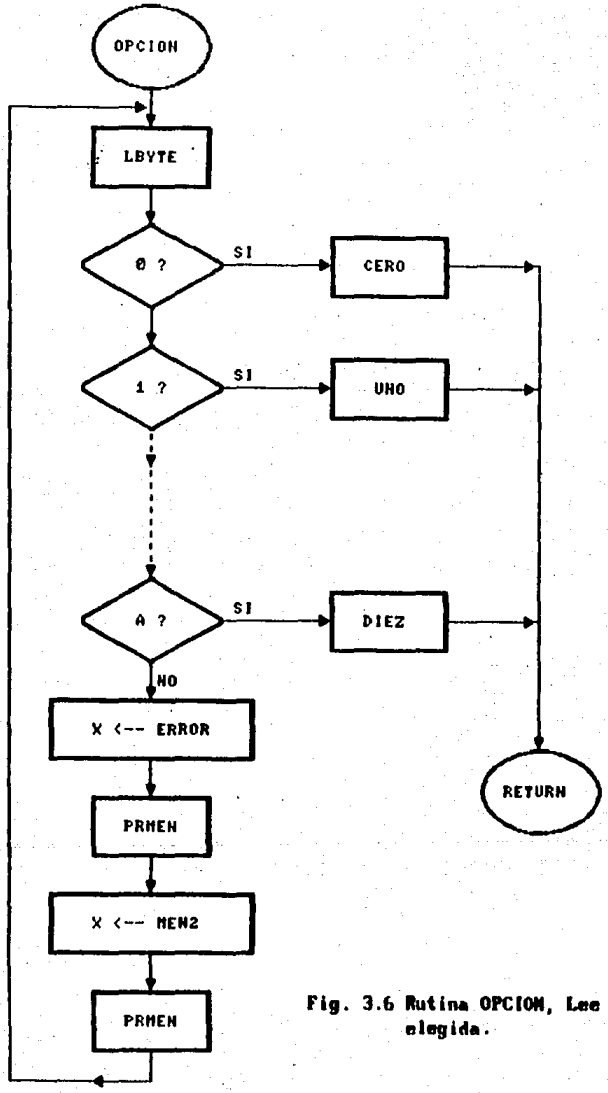


Fig. 3.6 Rutina OPCION, Lee la OPCION elegida.

Con el ACC A se direccionan bloques de memoria de 128 Bits, poniendo $04 en este registro y aplicándolo al puerto B de el PIA 1, se tiene la dirección 00 0XXX XXXX con un 1 en el DATA IN. Con un $04 en el ACC B aplicado al puerto A de misma PIA, se tiene la dirección 00 0000 0000 y la condición de escritura (R/W en bajo). Bajo estas características se llama a la rutina ESCR1 que sirve para acceder a todas las localidades de memoria dentro de un bloque. Para cambiar de bloque se suma $20 al ACC A y esto se repite hasta que se completa un ciclo, es decir, hasta que ACC A vuelve a tener un $04. Escritos los 1s en el DEP se procede a leer los vectores respuesta y compararlos con los vectores de respuesta esperada. En este caso sólo se tiene que verificar un bit. El proceso de direccionamiento es el mismo pero aquí la señal de R/W debe estar en alto para una condición de lectura, esto implica que el puerto A de el PIA 1 debe comenzar con un $00 y para cada bloque de memoria se llama a la rutina LEE121 que verifica si el contenido de cada localidad de memoria accedida corresponde a un 1. Dado que la rutina anterior usa al AöCC A es necesario, cada vez que se entra a ella, almacenarlo en la pila. Con el registro de índice X se lleva un control de la ocurrencia de errores.

Todo lo anterior se repite pero ahora escribiendo y verificando para los 0s con las subrutinas ESC021 y LEE021 respectivamente. Cada vez que ocurre un error el registro X se incrementa llegando al final con el número total de errores.

ESC121(A;): (**ESC**ribe **1**s a la **2**102 Fig. 3.9b) Se escriben 1s a todas las localidades de memoria dentro del bloque delimitado por el ACC A. Se limpia el ACC B y este se usa para crear la nueva dirección escribiéndolo sobre el puerto A de el PIA 1, pero para cada nuevo vector se verifica que la condición de escritura (R/W en bajo) se siga cumpliendo quitando aquellos que no lo hagan. El ciclo se completa cuando el ACC B vuelve a ser cero.

LEE121(A;V): (**LEE** **1**s de la **2**102 Fig. 3.9c) Se leen todas las localidades de memoria dentro del bloque delimitado por el ACC A. El procedimiento de lectura es igual al de escritura excepto por que aquí se verifica la condición de lectura (R/W en alto) y además se verifica que el dato leído (vector de respuesta) sea un 0, si no es así se prende el bit de estado V y se incrementa el contador de errores dentro del registro de índice X.

ESC021(A;): (**ESC**ribe **0**s a la **2**102 Fig. 3.10a) Funciona igual que la rutina ESC121 excepto porque aquí la terminal correspondiente al DATA IN se matiene en 0.

LEE021(A;): (**LEE** **0**s de la **2**102 Fig. 3.10b) Su operación es la misma que la rutina LEE121 pero aquí la verificación de cada lectura se hace sobre los 0s.

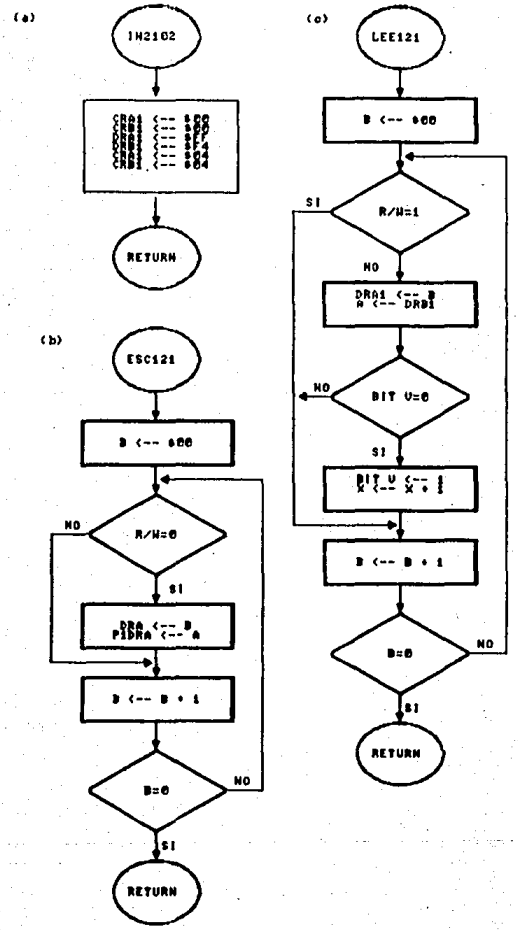


Fig. 3.9 Subrutinas de CERO:
(a) IN2102, inicia los PIAs.
(b) ESC121, Se ESCriben unos.
(c) LEE121, LEE unos.

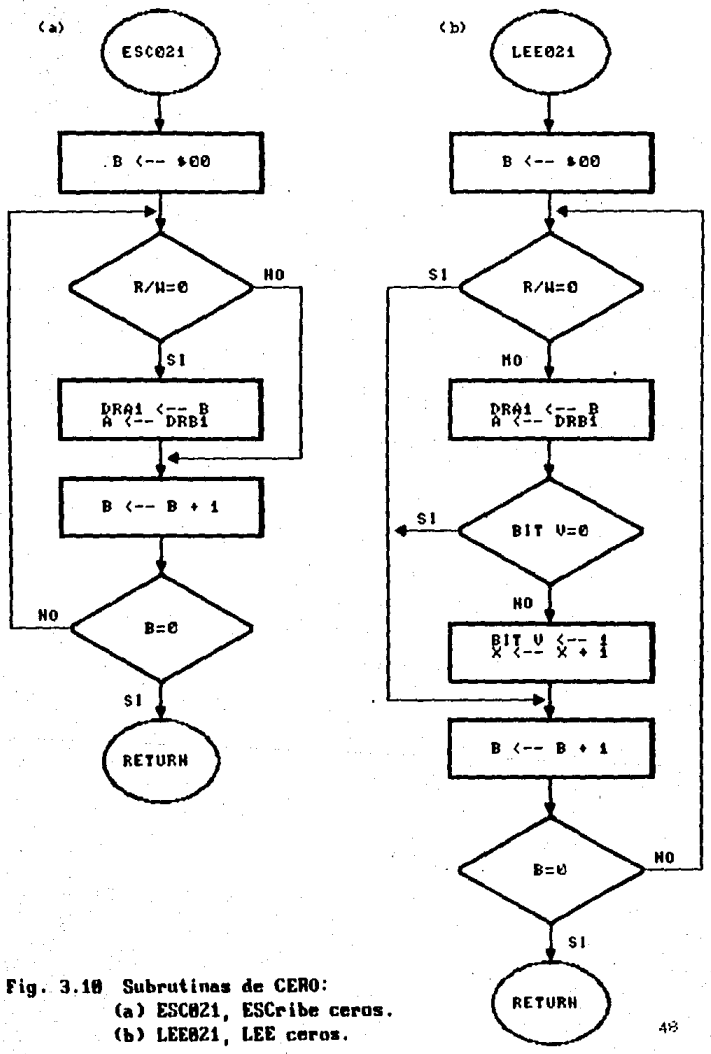


Fig. 3.18 Subrutinas de CERO:
(a) ESC021, ESCribe ceros.
(b) LEE021, LEE ceros.

3.4.2 Prueba del C.I. MC6810.

UNO(;): (Fig. 3.12a) Con esta rutina se prueban las memorias estáticas MC6810 de 256 X 8 Bits. Se sigue la misma filosofía que en el caso anterior: crear una serie de vectores de prueba con sus correspondientes vectores de respuesta esperada. La asignación de funciones de sus terminales de acuerdo al fabricante son:

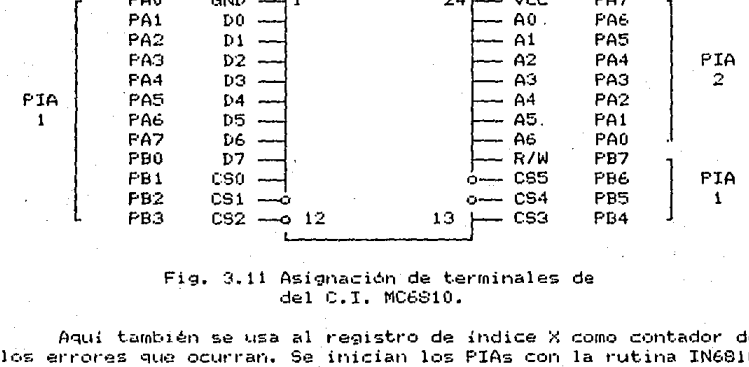


Fig. 3.11 Asignación de terminales de del C.I. MC6810.

Aquí también se usa al registro de índice X como contador de los errores que ocurran. Se inician los PIAs con la rutina IN6810 (Fig. 3.12b) programando como entradas para los PIAs las líneas correspondientes a PA0 de el PIA 1 y PA7 de el PIA 2, dejando las restantes como salidas.

Usando la rutina ESC168 se escriben $FF a todas las localidades de memoria (esto es 1s) y con la rutina LEE168 se leen las mismas y se comparan con los vectores de respuesta esperada. Si estos no coinciden con los primeros, entonces se incrementa el contador de errores. La misma secuencia se sigue para los 0s. En esta rutina se tienen dos cuentas de errores una para los 1s y otra para los 0s.

ESC168(;): (**ESC**ribe 1s a la **6810** Fig. 3.12c) Se escriben $FFs a todas las localidades de la memoria. Las condiciones para poder iniciar una secuencia de escritura son que el circuito se encuentre seleccionado poniendo CS0 y CS3 en alto y CS1, CS2, CS4 y CS5 en bajo dado que estas últimas son verificadas bajas, así también como R/W. Se inicia el ACC A con un $80 dado que el bit más significativo de este dato no interesa porque corresponde a la línea de Vcc, se manda este dato al puerto A de el PIA 1, con esto se están escribiendo los 1s. Se incrementa el ACC A y se repite la tarea hasta que el ACC A llega a $00.

LEE168(;): (**LEE** 1s de la **6810** Fig. 3.13) Se verifica que todas las localidades de memoria tengan los 1s correspondientes.

Aquí es necesario cambiar la configuración inicial de los PIAs dado que las mismas líneas de datos se usan para leer la información, por lo tanto se requiere que ahora estas líneas sean entradas para el PIA. Se escribe un $00 al registro de direcciones A (DRA) para tener las entradas correspondientes a D0 hasta D6, un $FE al registro de direcciones B para D7 y un $FE en el registro de direcciones A de el PIA 2, para Vcc. Para poner las condiciones de lectura se requiere que el circuito este seleccionado de la misma manera que en la rutina anterior, pero aquí la línea de R/W debe estar en alto. Listo el chip para lectura, es necesario tomar el dato en dos partes dado que los bits D0 hasta D6 van a quedar en un registro y D7 en otro. Dejando fijas las condiciones de lectura, primero se lee en el ACC A la parte D0-D6 y se le hace un desplazamiento hacia la izquierda para ponerlo en la parte menos significativa. Después se verifica que todos estos bits sean 1s, si no es así entonces se marca un error. Se lee el último bit D7 en el mismo acumulador y se hace nuevamente un corrimiento a la izquierda con acarreo, con esto el bit deseado (D7) se lleva al bit C del resgitro de código de condición y se pregunta por su estado: si es uno se pasa a leer la siguiente localidad y si no, ha ocurrido un error. Esto se repite para todas las localidades de memoria las cuales se direccionan usando el ACC B comenzando en $80 hasta $00 que corresponden a las 256 localidades del circuito.

ESC068(;): (**ESC**ribe **0**s en la **68**10 Fig. 3.14a) Se escriben 0s a todas las localidades del DEP siguiendo la misma secuencia que en la rutina ESC168 descrita antes.

LEE068(;): (**LEE 0**s de la **68**10 Fig. 3.14b) Se leen los 0s escritos en el DEP siguiendo la secuencia descrita en la rutina LEE168 descrita arriba.

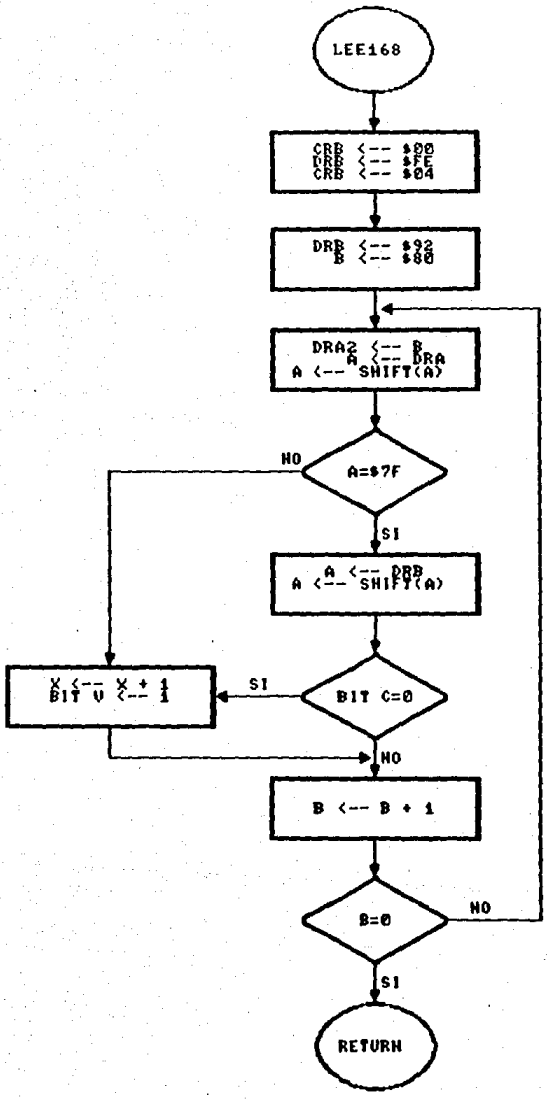


Fig. 3.13 Subrutina de UNO: LEE168, LEE unos del C.I. MC6810.

3.4.3 Prueba del C.I. MC2716.

DOS(;): (Fig. 3.16a) Se prueban los EPROMs MC2716 de 2,048 X 8 Bits. En este tipo de memorias los datos están grabados sin que por medios normales de programación puedan ser modificados. Sólo con un programador (quemador) de EPROMs podrían ser cambiados. Por lo anterior la forma de probarlos es únicamente verificar si el circuito tiene solamente 1s con lo cual podríamos decir que no ha sido programado. Otra opción es el mostrar toda la información grabada y si se tiene un listado de lo que debe de tener, hacer una comparación visual o teclearla para ponerla en la memoria RAM del **PLC** y que éste se encargue de realizar las comparaciones. En esta rutina sólo se implementó el caso en el que se muestra la información en forma total a pantalla o en su defecto a impresora a través de la terminal.

La disposición física de las terminales en este circuito integrado es:

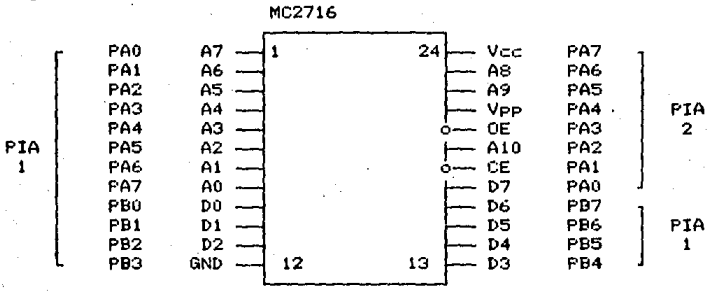


Fig. 3.15 Asignación de las terminales del C.I. MC2716.

La configuración de los PIAs se hace a través de la rutina IN2716 (Fig. 3.16b). Todas las líneas del puerto A de el PIA 1 son programadas como salidas, todo el puerto B como entradas, PA0, PA4 y PA7 de el PIA 2 como entradas y las líneas restantes de este mismo puerto como salidas.

Para la prueba de este circuito fue necesario construir dos tablas (TBL1 y TBL2) a partir de las cuales se generan las direcciones de cada localidad de memoria y otra (TABLA) que es la que se genera con los datos leídos. La dirección inicial de TABLA se carga en la localidad CONT, el ACC B se limpia para hacer el direccionamiento, que hasta aquí todavía es parcial, dado que sólo se está escribiendo información a las líneas de direcciones A0, A1,...,A7. Falta escribir la parte alta del direccionamiento (A8, A9, y A10), lo cual se hace dentro de la rutina DIRLEE. Se lee el dato contenido por la localidad direccionada una vez que la dirección se ha completado dentro de la rutina antes citada. El ACC B se incrementa con $50 y toda la secuencia se repite

hasta que el ACC B alcanza $40. Todo lo anterior se repite pero ahora cargando en el ACC B un $04 que corresponde al segundo KByte de memoria y el proceso es detenido cuando el ACC B es igual a $44.

DIRECC(;): (**DIR**recciona y **LEE** Fig. 3.17) Con este procedimiento se completa la dirección que ha quedado inconclusa en la rutina principal correspondiente a la prueba del EPROM 2716 y se lee el dato de dicha localidad. Primeramente se guardan los acumuladores en la pila, dado que tienen información que se usa en la rutina principal. En la localidad TEMPO se almacena el apuntador de la tabla TBL1 que contiene máscaras para la construcción de la dirección de cada localidad del DEP. Los primeros dos elementos de esta tabla son cargados dentro del registro X y se verifica si se han sacado 16 elementos de la tabla TBL1, si es así, entonces se recuperan los valores de los acumuladores de la pila y ha terminado, si no se almacena el apuntador de la tabla TBL2 en la localidad TEMPO+2 y los dos primeros elementos de esta tabla se cargan en el registro X y mientras no se hayan sacado los 16 elementos de dicha tabla se realizan las siguientes tareas: En el ACC A se guarda el contenido de la dirección apuntada por X (1er. elemento de la tabla TBL2), en el ACC B se tiene a esta altura el primer elemento de la tabla TBL1. Se suman lógicamente los acumuladores y en el ACC A se tiene el vector de prueba para el puerto A de el PIA 1, se escribe a él y se llama a la rutina LEE para leer el dato de la DEP. Como se puede observar tenemos dos ciclos iterados, con cada elemento de la tabla TBL1 se barren todos los elementos de la tabla TBL2 para la construcción de las direcciones del EPROM en prueba. Después de leer el dato (que es almacenado en TABLA). Se incrementa CONT (apuntador de TABLA) y TEMPO+3 (apuntador de TBL2). Cuando se han accedido todos los elementos de la tabla TBL2 se incrementa TEMPO+1 (apuntador de la tabla TBL1) hasta que también se han accedido todos los elementos de la tabla TBL1. Con todo esto se tiene en memoria RAM una tabla a partir de la dirección TABLA todos los datos de todas las localidades del EPROM.

LEE(;X): (**LEE Fig. 3.16c**) Se lee el contenido de las localidades del EPROM, una vez que han sido direccionadas por la rutina anterior. Dado que los bits del dato en el DEP no se encuentran en forma secuencial en los puertos de los PIAs es necesario leerlos por partes: Primero se leen los bits D1 al D7, se recorren a la izquierda y usando la operación lógica AND con la máscara $78, se dejan en el ACC A los bits D7,D6,D5,D4 en las posiciones 6,5,4 y 3 de este acumulador. En forma análoga pero usando la máscara $07 se dejan los bits D3,D2,D1 en las posiciones 2,1,0. Después los bits D7 a D1 se dejan en las posiciones 6 a 7 del ACC A. Unicamente falta el bit D8 que se obtiene en el ACC B. Haciendo un AND de este ACC con $80 se eliminan los restantes y finalmente se suman ACC A + ACC B dejando en el ACC A el byte resultante que corresponde al dato leído y ordenado. Se termina almacenando el byte en la localidad DATO y recuperando de la pila a los acumuladores.

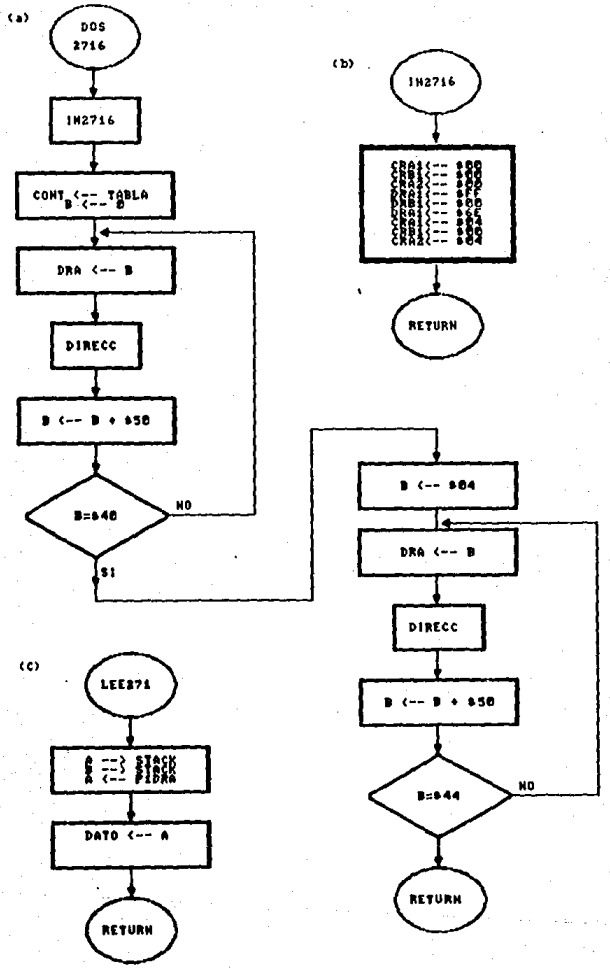


Fig. 3.16 Rutina de Prueba DOS:
(a) DOS, prueba el C.I. MC2716.
(b) IN2716, INicia los PIAs.
(c) LEE2716, LEEn Bytes.

3.4.4 Prueba del C.I. 74112.

TRES(;): (Fig. 3.19a) Se prueba el integrado 74112 que es un flip-flop doble tipo J-K cuya asignación de terminales se muestra a continuación:

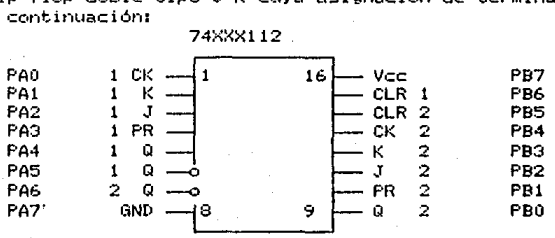


Fig. 3.18 Asignación de las terminales del C.I. 74112.

La rutina IN74112 (Fig. 3.19b) inicia los PIAs como sigue: PA4, PA5, PA6, PA7 y PB0 como entradas a el PIA 1 y todas las demás terminales como salidas.

La tabla de verdad para este circuito es:

	ENTRADAS					SALIDAS	
	PR	CLR	CLK	J	K	Q	¬Q
1	L	H	X	X	X	H	L
2	H	L	X	X	X	L	H
3	L	L	X	X	X	H*	H*
4	H	H	T	L	L	Qo	¬Qo
5	H	H	T	H	L	H	L
6	H	H	T	L	H	L	H
7	H	H	T	H	H	TOGGLE	
8	H	H	H	X	X	Qo	¬Qo

T = Transición

Fig. 3.1 Tabla de verdad del 74XX112

Aquí también es necesario el uso de una tabla de máscaras y datos que se construye de acuerdo al siguiente arreglo siguiando la correspondencia de la numeración de extrema izquierda:

	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	
	Vcc	CL1	CL2	CK2	K2	J2	PR2	Q2	
	X	1	1	1	0	0	1	X	72
4	X	1	1	0	0	0	1	X	62
	X	X	X	X	X	X	X	1	01 m
	X	1	1	1	1	1	1	X	7E
5	X	1	1	0	1	1	1	X	6E
	X	X	X	X	X	X	X	0	00 m
	X	1	1	1	0	1	1	X	76
6	X	1	1	0	0	1	1	X	66
	X	X	X	X	X	X	X	1	01 m
	X	1	1	1	1	0	1	X	7A
7	X	1	1	0	1	0	1	X	6A
	X	X	X	X	X	X	X	0	00 m
	X	1	1	X	X	X	0	X	60
8	X	1	1	1	X	X	1	X	72
	X	X	X	X	X	X	X	1	01 m
	X	1	1	X	X	X	0	X	60
1	X	X	X	X	X	X	X	1	01 m
	X	0	0	X	X	X	1	X	02
2	X	X	X	X	X	X	X	0	00 m
	X	0	0	X	X	X	0	X	00
3	X	X	X	X	X	X	X	1	01 m

Fig. 3.3 Tabla para el puerto B del PIA 1

Los valores de la extrema derecha corresponden a los vectores de prueba necesarios para cada combinación de las entradas. El subindice "m" indica que ese valor corresponde a la

máscara para el vector de respuesta esperada. Como se puede observar hay tercias (dos vectores de prueba con un vector de respuesta y un vector de prueba con un vector de respuesta). El uso de las tercias se debe a que para obtener la respuesta deseada se requiere una transición negativa de reloj (1 a 0), por lo tanto se escribe el primer vector de prueba seguido del segundo provocando la transición requerida. La respuesta obtenida se mantiene en las salidas hasta que no ocurra otra transición, pudiendo con esto compararlo usando el tercer vector (máscara). Para el caso de los pares no se requiere la transición de reloj.

Dentro de la rutina se creó una tabla conteniendo todos los elementos de las dos tablas de arriba. EL registro X se asigna como apuntador de la la tabla TBL, se llama a la subrutina FUNES que sirve para presentar un vector de prueba para luego con la subrutina CAPT leer el vector de respuesta y compararlo con el vector de respuesta esperada usando las máscaras de la tabla TBL. Si ocurre algún error se regresa en el ACC A el valor $55. Cuando ocurre esto termina la ejecución de la rutina. Al sacar los doce primeros elementos de la tabla (los pares), se procede en forma similar a lo anterior, a sacar el resto (tercias) para lo cual es necesario presentar dos vectores de prueba a través de la subrutina FUNES.

FUNES(;): (**FUN**ción **ES**cribe Fig. 3.20a) Se escribe el vector de prueba a los puertos de el PIA 1. Como ya se mencionó arriba el registro X es el apuntador de la tabla, usando un modo de direccionamiento indicado se van sacando los elementos de la tabla a los acumuladores A y B escribiendolos posteriormente en los registros de datos correspondientes de el PIA 1.

CAPT(;A): (**CAPT**ura Fig. 3.20b) Se lee el vector de respuesta y usando la máscara correspondiente se compara con el vector de respuesta esperada poniendo en su párametro de salida (ACC A) el valor $55 si la comparación no es satisfactoria, esto indica el error en la rutina TRES.

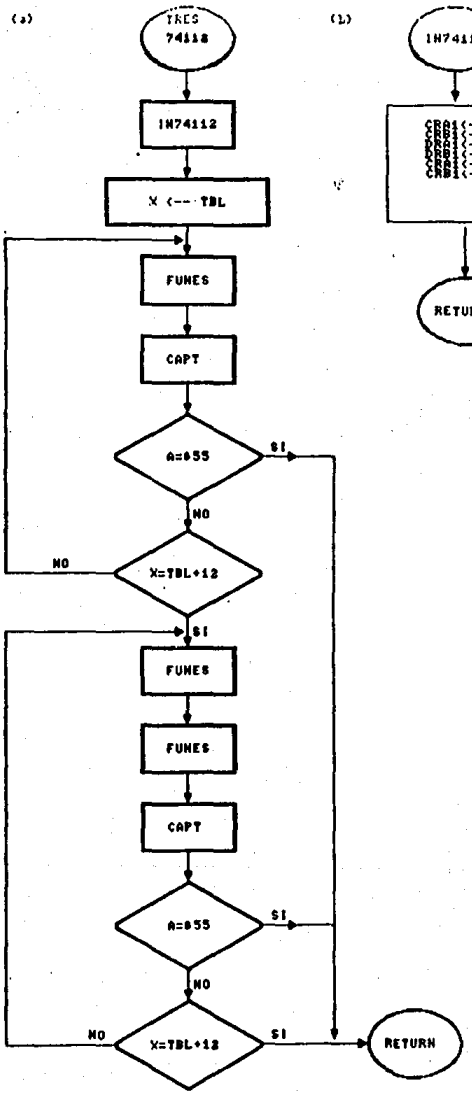


Fig. 3.19 **Rutina de Prueba TRES:**
(a) TRES, prueba el C.I. 74112.
(b) IN74112, INician los PIAs.

```
                        ********************************************
                        *
                        * Programa monitor del instrumento de pruebas
                        * ULTIMA CORRECCION:  24 NOV 89
                        * Ver 1.1
                        *
                        ********************************************


                0900    STACK   EQU     $0900     Stack del sistema
                8000    ACCR    EQU     $8000     Registro de Control del ACIA
                8001    ACDR    EQU     $8001     Registro de datos del ACIA
                        *
                        * PIA 1
                        *
                8010    P1DRA   EQU     $8010     Registro de Direcciones puerto A de la  PIA 1
                8011    P1CRA   EQU     $8011     Registro de Control puerto A de la PIA 1
                8012    P1DRB   EQU     $8012
                8013    P1CRB   EQU     $8013
                        *
                        * PIA 2
                        *
                8020    P2DRA   EQU     $8020
                8021    P2CRA   EQU     $8021
                8022    P2DRB   EQU     $8022
                8023    P2CRB   EQU     $8023
                        *
                        * PIA 3
                        *
                8040    P3DRA   EQU     $8040
                8041    P3CRA   EQU     $8041
                8042    P3DRB   EQU     $8042
                8043    P3CRB   EQU     $8043

0100                    ORG     $0100     LOCALIDADES RESERVADAS EN RAM
                        *
0100            NUMERO  RMB     3         Usada en el programa monitor
0103            ERROR0  RMB     2
0105            ERROR1  RMB     2         Usadas en la rutina RT0 (2102)
0107            ERR2    RMB     2
0109            ERR1    RMB     2         Usadas en la rutina RT1 (5810)
010B            INIC    RMB     2
010D            TEMPO   RMB     2
010F            DATO    RMB     2
0111            CONT    RMB     2
0113            TABLA   RMB     2048      Usadas en la rutina RT2 (2716)
0913            MALO    RMB     5
0918            DATO74  RMB     2         Usadas en la rutina RT3 (74112)
                        *

0C00                    ORG     $0C00     ORIGEN PARA LOS MENSAJES DEL PROGRAMA MONITOR
0C00 50 52 4F 47 INICIO FCC     'PROGRAMA MONITOR INSBUG VER. 1.0('
0C04 52 41 4D 41
```

```
                0CD3 45 4E 20 53
                0CD7 45 52 20 50
                0CDB 52 4F 42 41
                0CDF 44 4F 53 2E
                0CE3 5B
  65            0CE4 20 23 20 20        FCC   ' 0  NOMENCLATURA    TIPO DE DISPOSITIVO['
                0CE8 20 4E 4F 4D
                0CEC 45 4E 43 4C
                0CF0 41 54 55 52
                0CF4 41 20 20 20
                0CF8 20 20 54 49
                0CFC 50 4F 20 44
                0D00 45 20 44 49
                0D04 53 50 4F 53
                0D08 49 54 49 56
                0D0C 4F 5B
  66            0D0E 20 30 20 20        FCC   ' 0  2102          RAM 1K X 1 BITS['
                0D12 20 32 31 30
                0D16 32 20 20 20
                0D1A 20 20 20 20
                0D1E 20 20 20 20
                0D22 20 20 52 41
                0D26 4D 20 31 4B
                0D2A 20 58 20 31
                0D2E 20 42 49 54
                0D32 53 5B
  67            0D34 20 31 20 20        FCC   ' 1  6810          RAM 256 X 8 BITS['
                0D38 20 36 38 31
                0D3C 30 20 20 20
                0D40 20 20 20 20
                0D44 20 20 20 20
                0D48 20 20 52 41
                0D4C 4D 20 32 35
                0D50 36 20 58 20
                0D54 38 20 42 49
                0D58 54 53 5B
  68            0D5B 20 32 20 20        FCC   ' 2  2716          EPROM 2K X 8 BITS['
                0D5F 20 32 37 31
                0D63 36 20 20 20
                0D67 20 20 20 20
                0D6B 20 20 20 20
                0D6F 20 20 45 50
                0D73 52 4F 4D 20
                0D77 32 4B 20 58
                0D7B 20 38 20 42
                0D7F 49 54 53 5B
  69            0D83 20 33 20 20        FCC   ' 3  74112         FLIP-FLOP J-K['
                0D87 20 37 34 31
                0D8B 31 32 20 20
                0D8F 20 20 20 20
                0D93 20 20 20 20
                0D97 20 20 46 4C
                0D9B 49 50 20 46
                0D9F 4C 4F 50 20
```

```
PROGRAMA INSEJG VER. 1.1          11-26-89 TSC ASSEMBLER  PAGE    4

      0DA3 4A 2D 4B 5B
 70   0DA7 20 42 20 20          FCC   ' B   NO APARECEIC'
      0DAB 20 4E 4F 20
      0DAF 41 50 41 52
      0DB3 45 43 45 5B
      0DB7 5B
 71   0DB8 4D 45 4D 4F  MEN2    FCC   'MEMORIA A PROBAR ? NUMERO = @'
      0DBC 52 49 41 20
      0DC0 41 20 50 52
      0DC4 4F 42 41 52
      0DC8 20 3F 20 4E
      0DCC 55 4D 45 52
      0DD0 4F 20 3D 20
      0DD4 40
 72   0DD5 45 53 54 41  ESTAMN  FCC   'ESTADO DEL SISTEMA.@'
      0DD9 44 4F 20 44
      0DDD 45 4C 20 53
      0DE1 49 53 54 45
      0DE5 4D 41 2E 40
 73   0DE9 53 45 20 52  PRUEMN  FCC   'SE REALIZA LA PRUEBA.@'
      0DED 45 41 4C 49
      0DF1 5A 41 20 4C
      0DF5 41 20 50 52
      0DF9 55 45 42 41
      0DFD 2E 40
 74   0DFF              ESTAD   RMB   6A
 75   0E3F 5B 43 49 52   MENE99  FCC   '[CIRCUITO DANADO..******[@'
      0E43 43 55 49 54
      0E47 4F 20 44 41
      0E4B 4E 41 44 4F
      0E4F 2E 2E 2A 2A
      0E53 2A 2A 2A 2A
      0E57 5B 40
 76   0E59 5B 43 49 52   BUENO   FCC   '[CIRCUITO EN BUEN FUNCIONAMIENTO.  [@'
      0E5D 43 55 49 54
      0E61 4F 20 45 4E
      0E65 20 42 55 45
      0E69 4E 20 46 55
      0E6D 4E 43 49 4F
      0E71 4E 41 4D 49
      0E75 45 4E 54 4F
      0E79 2E 20 20 5B
      0E7D 40
 77
 78                     ***************************
 79                     * INACI( ) *
 80                     * Se inicializa el ACIA
 81                     ***************************
 82   0E7E 86   03      INAC1   LDAA   #$03     Master Reset
 83   0E80 B7   8000            STAA   ACCR
 84   0E83 86   11              LDAA   #$11     %0001 0001 8 bits de datos, 2 de stop, no paridad, entre 16
 85   0E85 B7   8000            STAA   ACCR
 86   0E88 39                   RTS
 87                     *
```

```
PROGRAMA INSBUG VER. 1.1          11-26-89 TSC ASSEMBLER  PAGE    5

                        ****************************
                        * INPIA CLR PICRA
                        * Se inicializan las PIA's
                        ****************************
 0EB9 7F  8011  INPIA   CLR   PICRA
 0EBC 7F  8010          CLR   PIDRA
 0EBF 7F  8013          CLR   PICRB
 0EC2 7F  8012          CLR   PIDRB
 0EC5 7F  8021          CLR   P2CRA
 0EC8 7F  8020          CLR   P2DRA
 0ECB 7F  8023          CLR   P2CRB
 0ECE 7F  8022          CLR   P2DRB
 0ED1 7F  8041          CLR   P3CRA
 0ED4 7F  8040          CLR   P3DRA
 0ED7 7F  8043          CLR   P3CRB
 0EDA 7F  8042          CLR   P3DRB
 0EDD 39                RTS
                        *
                        ****************************
                        * RESET( : )
                        * Se inician ACIA y PIA's al encender o presionar el boton
                        * de RESET o al inicio de sesion
                        ****************************
 0EDE 10CE 0F00 RESET   LDS   #STACK    Se inicializa el apuntador de stack
 0EE2 BD  0E7E          JSR   INACI     Se inicializa el ACIA
 0EE5 BD  0EB9          JSR   INPIA     Se inicializa la PIA
 0EE8 7E  0083          JMP   CONTROL   Pasa el control al programa monitor
                        *
                        ****************************
                        * LBYTE( :B)
                        * Se lee un byte de la terminal en el acumulador A
                        ****************************
 0EEB B6  8000  LBYTE   LDAA  ACCR      A (-- Estado del ACIA
 0EEE 8A  01            ORAA  #$01      Registro de recepcion vacio ?
 0EF0 27  F9            BEQ   LBYTE     Si es asi, ve a LBYTE
 0EF2 F6  8001          LDAB  ACDR      Si no, Lee el dato del registro de recepcion
 0EF5 B6  8000  LBYB    LDAA  ACCR
 0EF8 81  02            CMPA  #$02      Dato leido completamente ?
 0EFA 26  F9            BNE   LBYB      Si no es asi, espera completarlo
 0EFC F7  8001          STAB  ACDA      Si es asi, B --) Registro de Transmision
 0EFF C4  7F            ANDB  #$7F      Se quita el ultimo bit
 0F01 39                RTS
                        *
                        ****************************
                        * EBYTE(B: )
                        * Se manda un byte a la terminal
                        ****************************
 0F02 34  02    EBYTE   PSHA            A --) Stack
 0F04 B6  8000  EBYB    LDAA  ACCR      A (-- Estado del ACIA
 0F07 81  02            CMPA  #$02      Se checa si el ACIA se encuentra ocupada
 0F09 26  F9            BNE   EBYB      Si es asi, espera
 0F0B F7  8001          STAB  ACDA      Si no es asi, manda B a la terminal
 0F0E B6  8000          LDAA  ACCR
 0F11 84  10            ANDA  #$10      Se verifica la condicion del ACIA
```

```
PROGRAMA INSBUG VER. 1.1          11-26-83 TSC ASSEMBLER  PAGE    6

  142  0EE3 26  EF          BNE   EBY0      Si ocurrio algun error de transmision se vuelve a envia
  143  0EE5 35  02          PULA            A (-- Stack
  144  0EE7 39              RTS
  145                       *
  146                       ************************
  147                       * CR( : )
  148                       * Escribe un LF seguido por un CR
  149                       ************************
  150  0EE8 C6  0A    CR     LDAB  #$2A
  151 )0EEA BD  0ED2          JSR   EBYTE     Se manda un CR a la terminal
  152  0EED C6  0D           LDAB  #$0D
  153 )0EEF BD  0ED2          JSR   EBYTE     Se manda un LF a la terminal
  154  0EF2 39              RTS
  155                       *
  156                       *
  157                       ************************
  158                       * PRMEN(X) )
  159                       * Se escribe una cadena de caracteres ASCII a la terminal
  160                       * el inicio de la cadena esta en el registro X
  161                       * y el fin se marca con el caracter '@'
  162                       ************************
  163 )0EF3 BD  0EE8    PRMEN  JSR   CR        Manda a la terminal CR+LF
  164  0EF6 E6  84      PRM1   LDAB  0,X       X=Inicio de la cadena, B (-- 1er. caracter
  165  0EF8 C1  5B            CMPB  #'[
  166  0EFA 26  07            BNE   PRM2      Sigue imprimiendo sobre la misma linea
  167 )0EFC BD  0EE8          JSR   CR        Si no, manda una linea nueva ( CR+LF)
  168  0EFF C6  20           LDAB  #$20       Para mandar un espacio en blanco
  169  0F01 20  04           BRA   PRM3
  170  0F03 C1  40    PRM2   CMPB  #'@       Fin de la cadena ?
  171  0F05 27  07           BEQ   PRM0       Si es asi termina
  172 )0F07 BD  0ED2  PRM3   JSR   EBYTE      Si no es asi, manda el siguiente caracter
  173  0F0A 30  01           INX              X (-- X + 1 Apunta al siguiente caracter
  174  0F0C 20  E8           BRA   PRM1
  175  0F0E 39      PRM0   RTS
  176                       *
  177                       ************************
  178                       * LEECOM( : )
  179                       * Se lee el numero de la opcion elegida
  180                       * y se ejecuta , se manda
  181                       * un mensaje si la opcion no es valida.
  182                       ************************
  183 )0F0F BD  0EDB  LEECOM JSR   LBYTE     Lee un caracter de la terminal
  184  0F12 C1  31           CMPB  #'1
  185  0F14 27  12           BEQ   LEE1       Si es la opcion 1 despliega menu de circuitos
  186  0F16 C1  32           CMPB  #'2
  187  0F18 27  12           BEQ   LEE2       Si es la opcion 2 despliega el estado del sistema
  188  0F1A BE  0C9A         LDX   #ERROR
  189 )0F1D BD  0EF3          JSR   PRMEN      Si no es ninguno de los dos manda un mensaje de error
  190  0F20 BE  0C8D         LDX   #MEN1
  191 )0F23 BD  0EF3          JSR   PRMEN      Se despliega el menu de opciones
  192  0F26 20  E7           BRA   LEECOM     Vuelve a leer el caracter
  193  0F28 BD  12BF  LEE1   JSR   MENUC1     Despliega el menu de circuitos integrados
  194  0F2B 39              RTS
  195  0F2C BD  1208  LEE2   JSR   ESTADO     Despliega el estado del sistema
```

```
PROGRAMA INSBUG VER. 1.1          11-26-99 TSC ASSEMBLER  PAGE   7

 196  0F2F 39                    RTS
 197                      *
 198                      ********************************
 199                      * OPCION( : )
 200                      * Se lee la opcion del chip a probar
 201                      * y se ejecuta la rutina correspondiente
 202                      ********************************
 203 )0F30 BD  0EE8    OPCION  JSR   LBYTE     Lee un caracter de la terminal
 204  0F33 C1  30              CMPB  #'0       Se busca cual es fue el circuito seleccionado
 205  0F35 27  3A              BEQ   CERO      por el usuario y se ejecuta la rutina correspondiente
 206  0F37 C1  31              CMPB  #'1
 207  0F39 27  3B              BEQ   UNO
 208  0F3B C1  32              CMPB  #'2
 209  0F3D 27  3C              BEQ   DOS
 210  0F3F C1  33              CMPB  #'3
 211  0F41 27  3D              BEQ   TRES
 212  0F43 C1  34              CMPB  #'4
 213  0F45 27  3E              BEQ   CUATRO
 214  0F47 C1  35              CMPB  #'5
 215  0F49 27  3F              BEQ   CINCO
 216  0F4B C1  36              CMPB  #'6
 217  0F4D 27  40              BEQ   SEIS
 218  0F4F C1  37              CMPB  #'7
 219  0F51 27  41              BEQ   SIETE
 220  0F53 C1  38              CMPB  #'8
 221  0F55 27  42              BEQ   OCHO
 222  0F57 C1  39              CMPB  #'9
 223  0F59 27  43              BEQ   NUEVE
 224  0F5B C1  41              CMPB  #'A
 225  0F5D 27  44              BEQ   DIEZ
 226  0F5F C1  42              CMPB  #'B
 227  0F61 27  44              BEQ   NOESTA    Si no esta se manda el control al prosrama monitor
 228  0F63 BE  0C9A            LDX   #ERROR    Si no fue un caracter valido entonces se mada un mensaje de error
 229 )0F66 BD  0EF3            JSR   PRMEN
 230  0F69 BE  0088            LDX   @MEN2
 231 )0F6C BD  0EF3            JSR   PRMEN     y se manda el mensaje para esperar la eleccion
 232  0F6F 20  BF              BRA   OPCION
 233                      *
 234                      * EN ESTE BLOQUE SE LLAMAN A LAS RUTINAS DE PRUEBA
 235                      *
 236  0F71 BD  1012    CERO    JSR   RT0
 237  0F74 20  38              BRA   OPC2
 238  0F76 BD  1108    UNO     JSR   RT1
 239  0F79 20  28              BRA   OPC2
 240  0F7B BD  110A    DOS     JSR   RT2
 241  0F7E 20  26              BRA   OPC2
 242  0F80 BD  1289    TRES    JSR   RT3
 243  0F83 20  21              BRA   OPC2
 244  0F85 BD  1281    CUATRO  JSR   RT4
 245  0F88 20  1C              BRA   OPC2
 246  0F8A BD  1282    CINCO   JSR   RT5
 247  0F8D 20  17              BRA   OPC2
 248  0F8F BD  1283    SEIS    JSR   RT6
 249  0F92 20  12              BRA   OPC2
```

```
PROGRAMA INSBUG VER. 1.1          11-26-89 TSC ASSEMBLER  PAGE    9

 304
 305                    ********************************
 306                    * LEE121( ;V ) Se prende V si hay error
 307                    * Se leen todas las localidades de la memoria y se checan
 308                    * que tengan UNO's, en caso contrario se incrementa el contador
 309                    * de errores en el registro indice X
 310                    ********************************
 311  0FD2 5F           LEE121  CLRB
 312  0FD3 C5  04       LEE11   BITB   #$04     Se Checa PB3 en 1 para R\W en alto (Lectura)
 313  0FD5 27  0E               BEQ    LEE12
 314  0FD7 F7  0010             STAB   PIDRA     Se direcciona una localidad
 315  0FDA B6  0012             LDAA   PIDRB     Se lee el contenido de tal localidad
 316  0FDD 84  08               ANDA   #$08      Se checa su contenido, bit V (--0
 317  0FDF 26  04               BNE    LEE12     Si no hay error continuamos leyendo
 318  0FE1 1A  02               SEV              Se prende el bit de OVERFLOW
 319  0FE3 30  01               INX              Se cuenta el error
 320  0FE5 5C           LEE12   INCB             Se pasa a la siguiente localidad
 321  0FE6 26  EB               BNE    LEE11
 322  0FE8 39                   RTS
 323
 324                    ********************************
 325                    * ESC021 ( ; )
 326                    * Se escriben CERO's
 327                    ********************************
 328  0FE9 5F           ESC021  CLRB
 329  0FEA C5  04       ESCR00  BITB   #$04     Se verifica PB3 en 0 para R\W bajo (escritura)
 330  0FEC 26  06               BNE    ESCR01
 331  0FEE F7  0010             STAB   PIDRA     Se direcciona una localidad
 332  0FF1 B7  0012             STAA   PIDRB     Se escribe CERO
 333  0FF4 5C           ESCR01  INCB             Pasa a la siguiente localidad
 334  0FF5 26  F3               BNE    ESCR00
 335  0FF7 39                   RTS
 336
 337                    ********************************
 338                    * LEE021( ;V ) Se prende V si hay error
 339                    * Se leen todas las localidades de la memoria y se checan
 340                    * que tengan UNO's, en caso contrario se incrementa el contador
 341                    * de errores en el registro indice X
 342                    ********************************
 343  0FF8 5F           LEE021  CLRB
 344  0FF9 C5  04       LEE01   BITB   #$04     Se Checa PB3 en 1 para R\W en alto (Lectura)
 345  0FFB 27  0E               BEQ    LEE02
 346  0FFD F7  0010             STAB   PIDRA     Se direcciona una localidad
 347  1000 B6  0012             LDAA   PIDRB     Se lee el contenido de tal localidad
 348  1003 84  08               ANDA   #$08      Se checa su contenido, bit V (--0
 349  1005 27  04               BEQ    LEE02     Si no hay error continuamos leyendo
 350  1007 1A  02               SEV              Se prende el bit de OVERFLOW
 351  1009 30  01               INX              Se cuenta el error
 352  100B 5C           LEE02   INCB             Se pasa a la siguiente localidad
 353  100C 26  EB               BNE    LEE01
 354  100E 39                   RTS
 355
 356
 357                    *  CUERPO PRINCIPAL DE LA RUTINA 2102
```

```
PROGRAMA INSBUG VER. 1.1          11-26-89 TSC ASSEMBLER  PAGE  10

                        ********************************************
                        
1002F BD  0FAA          JSR   IN2102   Se inicia la PIA
                        
                        * Se escriben UNO's a todas las localidades
                        
1012 86  04     RT0    LDAA  #$04
1014 B7  8012   SIG    STAA  P1DRB     Se direccionan localidades de memoria
1017 BD  0FC3          JSR   ESC121    Se escriben UNO's
101A 8B  20            ADDA  #$20      Se pasa a la siguiente bloque de localidades
101C 81  04            CMPA  #$04
101E 26  F4            BNE   SIG       Hasta barrer todos los bloques
                        
                        * Se verifica que esten almacenados los UNO's
                        
1020 CE  0000          LDX   #$0       El contador de errores en 0
1023 86  04            LDAA  #$04
1025 B7  8012   SIG1   STAA  P1DRB     Se direccionan localidades de memoria
1028 34  02            PSHA
102A BD  0F02          JSR   LEE121    Se leen y compara el contenido de las localidades
102D 35  02            PULA
102F 8B  20            ADDA  #$20      Se direcciona el siguiente bloque
1031 81  04            CMPA  #$04
1033 26  F0            BNE   SIG1      Hasta checar todas las localidades
1035 BF  0105          STX   ERROR1
                        
                        * Se escriben CERO's a todas las localidades
                        
1038 86  00            LDAA  #$00
103A B7  8012   SIG2   STAA  P1DRB     Se direccionan localidades de memoria
103D BD  0FE9          JSR   ESC021    Se escriben CERO's
1040 8B  20            ADDA  #$20      Se pasa al siguiente bloque de direcciones
1042 81  00            CMPA  #$00
1044 26  F4            BNE   SIG2      Hasta checar todas las direcciones
                        
                        * Se verifican los CEROS en las localidades
                        
1046 CE  0000          LDX   #$0
1049 86  00            LDAA  #$00
104B B7  8012   SIG3   STAA  P1DRB     Se direccionan localidades de memoria
104E 34  02            PSHA
1050 BD  0FF8          JSR   LEE021
1053 35  02            PULA
1055 8B  20            ADDA  #$20      Se direcciona el siguiente bloque de direcciones
1057 81  00            CMPA  #$00
1059 26  F0            BNE   SIG3      Hasta checar todas las direcciones
105B BF  0103          STX   ERROR0
105E 39                RTS
                        *
                        ***** FIN DE LA RUTINA DE PRUEBA PARA 2102
                        *
                        ***** RT1
                        ********************************
                        *
```

```
PROGRAMA INSBUG VER. 1.1          11-26-89 TSC ASSEMBLER  PAGE   11

                        * Subrrutinas del programa monitor para realizar pruebas
                        * de la memoria MCS810
                        *
                        *
                        *
                        ************************************

                        ****************************
                        * IN6810( ; )
                        * Se inician las PIA's para realizar las pruebas
                        ****************************
1059 7F  8011   IN6810  CLR   P1CRA     se inician solo los puertos a usar
1062 7F  8013           CLR   P1CRB
1065 7F  8021           CLR   P2CRA
1068 86  FE             LDAA  #%FE
106A B7  8010           STAA  P1DRA     Bit PA0 PIA 1 como entrada, resto salidas
106D 86  FF             LDAA  #$FF
106F B7  8012           STAA  P1DRB     Puerto B de la PIA 1 como salidas
1072 86  7F             LDAA  #$7F
1074 B7  8020           STAA  P2DRA     Bit PA7 PIA 2 como entada resto como salidas
1077 86  04             LDAA  #$04
1079 B7  8011           STAA  P1CRA     Seleccion del OUTPUT REGISTER
107C B7  8013           STAA  P1CRB
107F B7  8021           STAA  P2CRA
1082 39                 RTS

                        ************************************
                        * ESC168( ; )
                        * Se escriben bytes $FF a la memoria
                        ************************************
1083 86  13     ESC168  LDAA  #%00010011
1085 B7  8012           STAA  P1DRB     Seleccion de la memoria a traves de los CHIP SELECT's
1088 86  FE             LDAA  #$FE
108A B7  8010           STAA  P1DRA     Se escriben $FF's a cada localidad
108D 86  80             LDAA  #$80
108F B7  8020   ESC1    STAA  P2DRA     Direccionamiento de una localidad de memoria
1092 4C                 INCA
1093 26  FA             BNE   ESC1      Hasta barrer 128 localidades
1095 39                 RTS

                        ************************************
                        * LEE168( ;V)
                        * Se leen las 128 localidades de la memoria y si hay error
                        * se prende el BIT de OVERFLOW
                        ************************************
1096 7F  8013   LEE168  CLR   P1CRB     Seleccion del DIRECCION DATA REGISTER
1099 86  FE             LDAA  #$FE
109B B7  8012           STAA  P1DRB     bit PA0 PIA1 como entrada
109E 86  04             LDAA  #$04
10A0 B7  8013           STAA  P1CRB     Seleccion del OUTPUT REGISTER
10A3 86  92             LDAA  #%10010010
10A5 B7  8012           STAA  P1DRB     R/W en alto para lectura
10A8 C6  80             LDAB  #$80
10AA F7  8020   LE168   STAB  P2DRA     Se direcciona una localidad de memoria
```

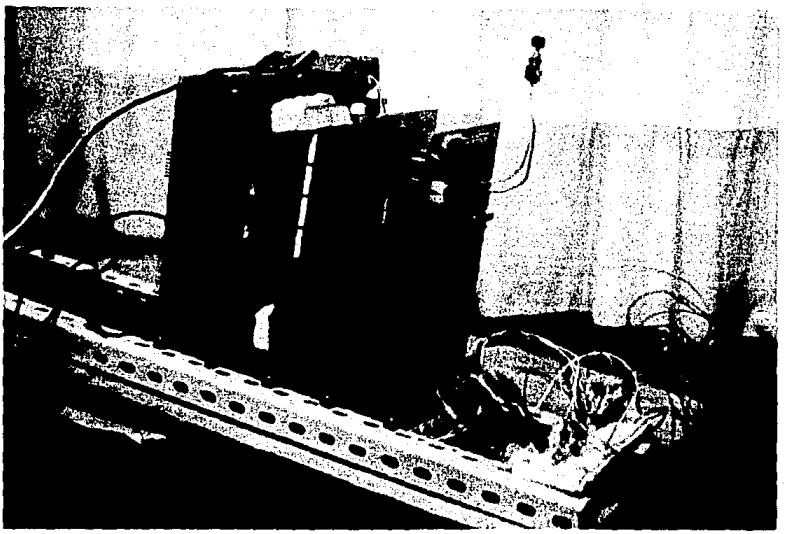
```
PROGRAMA INSBUG VER. 1.1          11-26-89 TSC ASSEMBLER  PAGE   13

 520  1105 20  F5            BRA    COS0
 521  1107 39          FIN    RTS
 522
 523                      *  CUERPO PRINCIPAL DEL RUTINA DE PRUEBA PARA LA 6810
 524                      ********************************************************
 525
 526  1108 8E  0000    RT1    LDX    #$0
 527  110B BD  105F           JSR    IN6810   Se inicializan las PIA's
 528  110E BD  1083           JSR    ESC16B
 529 )1111 BD  1096           JSR    LEE16B
 530  1114 BF  0109           STX    ERR1
 531  1117 8E  0000           LDX    #$0
 532  111A BD  105F           JSR    IN6810
 533 )1110 BD  10C7           JSR    ESC06B
 534 )1120 BD  10D9           JSR    LEE06S
 535  1123 BF  0107           STX    ERR0
 536  1126 39                 RTS
 537                      ****** FIN DE LA RUTINA  DE PRUEBA PARA 6810
 538                      ******* RT2
 539                      ***********************************
 540                      *
 541                      * Subrrutinas del programa monitor para realizar pruebas
 542                      * de la memoria M2716
 543                      *
 544                      *
 545                      *
 546                      ***********************************
 547
 548  1127 00 08 04 0C    TBL1   FCC    $0,$8,$4,$C,$2,$A,$6,$E,$1,$9,$5,$D,$3,$B,$7,$F
      112B 02 0A 06 0E
      112F 01 09 05 0D
      1133 03 0B 07 0F
 549  1137 00 80 40 C0    TBL2   FCC    $0,$80,$40,$C0,$20,$A0,$60,$E0,$10,$90,$50,$D0,$30,$B0,$70,$F0
      113B 20 A0 60 E0
      113F 10 90 50 D0
      1143 30 B0 70 F0
 550
 551                      ***********************************
 552                      * IN2716 (1)
 553                      * Se inician las PIA's
 554                      ***********************************
 555
 556  1147 7F  8011    IN2716 CLR    P1CRA    Seleccion del DATA DIRECCITION REGISTER
 557  114A 7F  8013           CLR    P1CRB
 558  114D 7F  8021           CLR    P2CRA
 559  1150 86  FF             LDAA   #$FF
 560  1152 B7  8010           STAA   P1DRA    Puerto A PIA 1 como salidas
 561  1155 4F                 CLRA
 562  1156 B7  8012           STAA   P1DRB    Puerto B PIA 1 como entradas
 563  1159 8E  6E             LDAA   #$6E
 564  115B B7  8020           STAA   P2DRA    puerto A PIA 2 como entradas y salidas
 565  115E 86  04             LDAA   #$04
 566  1160 B7  8011           STAA   P1CRA    Seleccion del OUTPUT REGISTER
 567  1163 B7  8013           STAA   P1CRB
```

```
PROGRAMA INSBUS VER. 1.1          11-26-89 TSC ASSEMELER   PAGE   15

  622  11CE 20   DA              BRA    DIR1
  623  11D0 7C   010E    DIR2    INC    TEMPO+1
  624  11D3 20   C5              BRA    DIR0
  625  11D5 35   04      DIR3    PULB
  626  11D7 35   02              PULA
  627  11D9 39                   RTS
  628
  629                           *  CUERPO PRINCIPAL DE LA RUTINA RT2
  630                           ***********************************************
  631  11DA BD   1147    RT2     JSR    IN2716    Se inician las PIA's
  632  11DD 108E 0100            LDY    #INIC
  633  11E1 8E   0113            LDX    #TABLA
  634  11E4 BF   0111            STX    CONT
  635  11E7 5F                   CLRB
  636  11E8 F7   0020    LL0     STAB   P2DRA     Se direcciona un bloque de memoria
  637 )11EB BD   1150            JSR    DIRECC
  638  11EE CB   50              ADDB   #$50      Nuevo bloque de memoria
  639  11F0 C1   40              CMPB   #$40
  640  11F2 26   F4              BNE    LL0
  641
  642                           * Segundo K'bits
  643
  644  11F4 C6   04              LDAB   #$04
  645  11F6 F7   0020    MM0     STAB   P2DRA
  646 )11F9 BD   1150            JSR    DIRECC
  647  11FC CB   50              ADDB   #$50
  648  11FE C1   44              CMPB   #$44
  649  1200 26   F4              BNE    MM0
  650  1202 39                   RTS
  651                           *
  652                           ***** FIN DE LA RUTINA DE PRUEBA PARA 2716
  653
  654                           ******* RT3
  655                           ***********************************
  656                           *
  657                           * Subrrutinas del programa monitor para realizar pruebas
  658                           * de los Flip-Flop's 74112 (J-K DUAL)
  659                           *
  660                           *
  661                           ***********************************************
  662                           * Tabla para los renglones 1,2 y 3 de tabla de verdad
  663  1203 00 60 10 01  TBL     FCC    $0,$60,$10,$1,$1,$2,$60,$0,$0,$0,$70,$1
       1207 01 02 60 00
       120B 00 00 70 01
  664                           * Tabla para los renglones 4,5,6,7 y 8 de la tabla de verdad
  665  120F 09 72 08 62          FCC    $09,$72,$08,$62,$10,$01,$0F,$7E,$0E,$6E,$10,$0
       1213 10 01 0F 7E
       1217 0E 6E 10 00
  666  121B 00 76 0C 66          FCC    $0,$76,$0C,$66,$10,$1,$2B,$7A,$0A,$EA,$60,$0
       121F 10 01 0B 7A
       1223 0A EA 60 00
  667  1227 00 60 09 72          FCC    $0,$60,$09,$72,$10,$1
       122B 10 01
  668
```

```
PROGRAMA INSEJO VER. 1.1          11-26-89 TSC ASSEMBLER  PAGE   17

                        *ERRR ( ; )
                        * Se ha detectado un error y se prende una bandera
                        ***************************************
1286  1283 86  55       ERRR    LDAA   #$55
727   1285 B7  0013             STAA   XVALD
728   1288 39                   RTS
729
730                     *************************************
731                     * CUERPO PRINCIPAL DE LA RUTINA PARA PRUEBA DE 74112
732                     *************************************
733
734  )1289 BD  1220     RT3     JSR    IN7411   Se inician las PIA's
735                     * Se probaran los renglones 1,2,3 de la tabla de verdad
736   128C BE  1283             LDX    #TBL     Se apunta al inicio de la tabla de datos
737  )128F BD  1246     PR0     JSR    FUNES    Se leen y escriben a puertos  dos datos de la tabla
738  )1292 BD  1255             JSR    CAPT     Se leen los puertos de las PIA's y se comparan con las mascaras
739   1295 81  55               CMPA   #$55
740   1297 27  17               BEQ    FIN74
741   1299 BC  1287             CMPX   #TBL+12   Si no han ocurrido errores se checa el final de Tabla
742   129C 26  F1               BNE    PR0
743  )129E BD  1246     PR01    JSR    FUNES    Se leen y escriben a los puertos dos datos de la tabla
744  )12A1 BD  1246             JSR    FUNES    IDEM
745  )12A4 BD  1255             JSR    CAPT     Se leen los puertos y se comparan con las mascaras
746   12A7 81  55               CMPA   #$55
747   12A9 27  05               BEQ    FIN74
748   12AB BC  1288             CMPX   #TBL+42   Se pregunta por el final de la tabla
749   12AE 26  E7               BNE    PR01
750   12B0 39           FIN74   RTS
751                     *
752                     ******* FIN DE LA RUTINA RT3
753                     *
754
755   12B1 39           RT4     RTS
756   12B2 39           RT5     RTS
757   12B3 39           RT6     RTS
758   12B4 39           RT7     RTS
759   12B5 39           RT8     RTS
760   12B6 39           RT9     RTS
761   12B7 39           RTA     RTS
762                     *
763                     *
764                     *
765                     ***************************************
766                     *
767                     * Fin del bloque de rutinas de prueba.
768                     *
769                     ***************************************
770                     *
771                     *
772                     *
773                     ***************************
774                     * ESTADO( ; )
775                     * Se despliega el estado del sistema
776                     ***************************
```

```
PROGRAMA INSBUG VER. 1.1          11-26-89 TSC ASSEMBLER   PAGE   18

  777   12B8 8E   0000    ESTADO  LDX   #ESTAMN
  778   12BB BD   0EF3            JSR   PRMEN
  779   12BE 39                   RTS
  780                     *
  781                     ********************************
  782                     * MENUCI( ; )
  783                     * despliega menu de C.I.
  784                     ********************************
  785   12BF 8E   0CAC    MENUCI  LDX   #MENUIN
  786   12C2 BD   0EF3            JSR   PRMEN
  787   12C5 BD   0F38            JSR   OPCION
  788   12C8 39                   RTS
  789                     *
  790   3000                      ORG   $8000
  791                     ********************************
  792                     * Programa principal
  793                     * CONTROL
  794                     ********************************
  795   8003 BD   0E7E            JSR   INACI
  796   8006 8E   0C80    CONTROL LDX   #INICIO
  797   8009 BD   0EF3            JSR   PRMEN
  798   800C 8E   0C35    CONT0   LDX   #MENU
  799   800F BD   0EF3            JSR   PRMEN
  800   8012 BD   0F0F            JSR   LEECOM
  801   8015 3F                   SWI


PROGRAMA INSBUG VER. 1.1          11-26-89 TSC ASSEMBLER   PAGE   19


  803                             END

0 ERROR(S) DETECTED


PROGRAMA INSBUG VER. 1.1          11-26-89 TSC ASSEMBLER   PAGE   20


SYMBOL TABLE:

ACCR   0800  ACDR   0801  BUENO  0E59  CAPT   1255  CAPT1  1282
CERO   0F71  CINCO  0F8A  CON1   10BB  CONT   0111  CONT0  0C0C
CONTRO 0C03  COS0   10FC  CR     0EEB  CUATRO 0F85  DATO   010F
DATO74 0318  DIEZ   0FA3  DIR0   115A  DIR1   11AA  DIR2   11D0
DIR3   11D5  DIRECC 1150  DOS    0F7B  EBYT0  0ED4  EBYTE  0ED2
EP0    1101  ER1   1EC0  ERR0   0107  ERR1   0109  ERR0R  0C9A
ERRO#0 2103  ERR0R1 2105  ERRR   1283  ESC01  1002  ESC021 0FE9
ESC056 12C7  ESC1   103F  ESC121 2FC3  ESC163 1033  ESCR00 0FEA
ESCR21 0FF4  ESCR10 0FC4  ESCR11 0FCE  ESTAD  0DFF  ESTADO 1208
ESTAMN 0305  FIN    1127  FIN1   10C5  FIN74  1200  FUNES  1246
IN2102 0F59  IN2716 1167  IN6810 105F  IN7411 1220  INACI  0E7E
```

INIC	0100	INICIO	0000	INPIA	0E83	LBYO	0EC5	LBYTE	0E8B
LED	10ED	LEE00	1000	LEE01	0FF9	LEE02	1008	LEE021	0FF8
LEE055	1003	LEE1	0F38	LEE11	0FD3	LEE12	0FE5	LEE121	0FD2
LEE105	1056	LEE2	0F3C	LEE27	116A	LEECOM	0F0F	LL0	11E0
MAL0	0917	MEN1	0C80	MEN2	0D65	MENERR	0E3F	MENU	0C35
MENUCI	121F	MENUIN	0CAC	MM0	11F6	NOESTA	0FA7	NUEVE	0F9E
NUMERO	2103	OCHO	0F99	OPC1	0FA6	OPCION	0F30	P1CRA	8011
P1CRB	8013	P1DRA	8010	P1DRB	8012	P2CRA	8021	P2CRB	8023
P2DRA	8020	P2DRB	8022	P3DRA	8041	P3CRB	8043	P3DRA	8040
P3DRB	8042	PR0	129F	PR01	129E	PRM0	0F0E	PRM1	0EF6
PRM2	0F0C	PRM3	0F07	PRMEN	0EF3	PRUEMA	0DE9	RESET	0EAE
RT0	1010	RT1	1185	RT2	11DA	RT3	1289	RT4	12B1
RT5	1252	RT6	12B3	RT7	12B4	RT8	1285	RT9	12B6
RTA	1297	SALERR	127F	SEIS	0F8F	SIETE	0F94	SIG	1014
SIG1	1025	SIG2	103A	SIG3	104B	STACK	0900	TABLA	0113
TBL	1283	TBL1	1127	TBL2	1137	TEMPO	0100	TRES	0F88
UNO	0F76								



FOTOGRAFIA DEL PROTOTIPO DESARROLLADO.

CONCLUSIONES

El prototipo del **PLC** construido, ha demostrado ser una herramienta útil en la detección de circuitos integrados dañados, esto se refleja significativamente en el tiempo invertido para detectarlos, sobre todo tratándose de circuitos de *memorias*. En este trabajo se desarrollaron tanto el hardware como el software para el análisis de memorias digitales sin que esto limite el alcance total de lo realizado, es decir que las pruebas de otros circuitos integrados pueden llevarse a cabo realizando únicamente el software necesario, el como hacerlo puede verse en el capitulo 3. La facilidad para crear los programas de pruebas para cualquier DEP lo hace versátil, y visto como un sistema entrada-salida puede usarse como un controlador digital, escribiendo un programa semejante al de prueba. A diferencia de la mayoria de los diseños de probadores de circuitos, el discutido aqui no requiere de un complejo manual de operación, el único requisito para quién desee crear su propio programa de prueba es el que conozca el lenguaje ensamblador del microprocesador MC6800. Para salvar esta limitante en futuras modificaciones he pensado que la creación de un pequeño lenguaje de programación para un **PLC** (con la misma filosofía de diseño seguida aquí) que pueda interconectarse a una Computadora Personal, lo cual seria de gran utilidad, dado que se podrían usar los recursos de ésta para la creación, depuración y compilación de los programas de pruebas. Dentro de la literatura relacionada con este tema se encuentran ya varios diseños de este tipo por ejemplo el presentado por Steven Ciarcia [13]. Ese instrumento está basado en el microcontrolador de Intel 8031 y puede trabajar en forma independiente pero también puede conectarse a una PC aunque tiene algunas desventajas en cuanto a su construcción, dado que algunos de los integrados que usa no son fáciles de conseguir en el mercado nacional, cosa que no ocurre con el instrumento diseñado en este trabajo.

Aún cuando la filosofía de diseño y los objetivos de uso son distintos, una de las principales ventajas encontradas comparando al **PLC** con un analizador de estados lógicos es que éste último requiere de un temporizador del sistema en prueba para la captura de cada uno de los estados del DEP, tomando en cuenta también (como se mencionó en el capitulo 1) el hecho de que el analizador sólo muestra las salidas o sea los vectores de respuesta sin que a través de él puedan generarse los patrones de prueba. Las

limitaciones principales respecto al analizador de estados lógicos es la manera de despliegue de los resultados de las pruebas, los opciones de disparo que tiene el analizador cuando se trata de analizar un sistema completo []. La diferencia escencial entre ambos es el precio.

Sin embargo, el **PLC** aún se encuentra en la etapa de desarrollo y a lo largo de su diseño han surgido ideas que pueden servir para modificar y/o cambiar la idea original. A continuación listo una serie de esas ideas:

1o. Usando el puerto libre de la PIA 3 y agregando la lógica necesaria se puede hacer la conmutación automática de las terminales de Alimentación y Tierra en los DEPs.

2o. Introducción de un programa que sea capaz de reconocer el tipo de DEP [A.14].

3o. Adecuación de la lógica para su interconexión con una PC. Esto implicaría también la introducción del programa adecuado, o en su defecto la elaboración de una tarjeta propia para PC con circuitos que sean directamente compatibles con ella, es decir, de la familia del MPU 8086/88.

Los detalles para hacer lo anterior se salen de los objetivos de este trabajo pero en algún instante dentro del diseño ya fueron considerados.

Hice mención al inicio de este trabajo que el **PLC** puede ser usado en la docencia. Podemos analizar esto desde dos puntos de vista: 1o. Su uso como un controlador de experimentos, lo cual requiere de algunos dispositivos de interconexión hacia el mundo analógico (convertidores A/D y D/A) y 2o. El estudio de su arquitectura dentro de las materias que imparte el Laboratorio de Cibernética o cualquier materia que tenga que ver con electrónica digital, el diseño modular permite que el estudio de cada una de sus partes por separado pueda ser muy didáctico ya que cualquier sistema digital contará con una o varias de ellas. Si se revisa el programa de estudio de la materia de Máquinas Digitales con Laboratorio veremos que cada modulo correspondería a un tema. Dado que el MC6800 es un microprocesador muy sencillo el estudio de él permite que el alumno pueda tener una idea clara de lo que es la arquitectura de una microcomputadora, así como su software, pero sobre todo, una idea sólida de como usar los microprocesadores en el diseño digital y dentro de la instrumentación.

el canal de datos. Son bidireccionales y tres estados con la capacidad de manejar directamente hasta una carga TTL con una capacidad de 130 pF. Estas señales transfieren información desde y hasta los dispositivos externos y memoria. Este canal permanece en estado de alta impedancia mientras la señal DBE (Data Bus Enable) se encuentre en estado bajo.

Habilitación de canal de datos (Data Bus Enable DBE) Esta señal sirve para habilitar el estado de alta impedancia en el canal de datos, es verificada alta. En operación normal puede ser manejada por la Fase de reloj 2 (PHI2). Cuando se requiere que un dispositivo externo tenga control del canal de datos DBE debe permanecer en estado bajo.

Canal Disponible (Bus available BA) Esta señal se encuentra normalmente en estado bajo, cuando llega a estar en alto indica que el MPU se ha detenido y que el canal de direcciones se encuentra disponible. Esto ocurre cuando la señal de HALT en el MPU se encuentra en estado bajo teniéndolo en un estado de espera como resultado de la ejecución de la instrucción WAIT. El MPU sale de éste estado ante la aparición de cualquier interrupción. Si TSC está en estado alto, BA será baja.

Lectura/Escritura (Read/Write R/W): Esta señal de salida llega a los dispositivos periféricos y a la memoria indicándoles si el MPU se encuentra en un ciclo de escritura (estado bajo) o en un ciclo de lectura (estado alto). El estado normal de esta señal es alto o sea de lectura. El control de estado de alta impedancia (TSC) en alto hace que la señal R/W vaya al estado de alta impedancia.

Reset: Esta señal se usa para iniciar o reestablecer las condiciones iniciales del MPU. Es verificada baja por lo tanto una transición al estado bajo inicia la secuencia de RESET. El contenido de las dos últimas localidades de memoria ROM ($FFFF y $FFFE) se cargan dentro del Contador de Programa (Programa Counter PC) que es el apuntador a la rutina de atención al RESET. Durante la ejecución de esta rutina, se prende el bit de la máscara de interrupción y se debe borrar antes de que el MPU pueda ser interrumpido por el IRQ. Mientras el RESET se encuentre en estado bajo, las señales de control del MPU se encuentran como sigue: VMA (Valid Memory Address en bajo), BA en bajo, canal de Datos en estado de alta impedancia, R/W en estado alto (Lectura) y el canal de direcciones contiene la dirección $FFFE. Durante 8 ciclos de reloj, al menos, la señal de RESET permanece en estado bajo, manteniendo así al VMA también bajo, de esta manera los dispositivos periféricos no pueden realizar ninguna tarea durante éste tiempo. La señal de RESET es completamente asíncrona respecto al reloj del sistema.

Requerimiento por Interrupción (IRQ): Esta señal de entrada sensa cuando un dispositivo periférico requiere ser atendido por el MPU. Este espera hasta haber completado el ciclo de instrucción en ejecución y si al terminarla, el Bit del código de condición correspondiente a la máscara de interrupción se

HALT: Esta es una línea de entrada al MPU que mientras se encuentra en estado bajo, lo mantiene en un estado de inactividad. Este responde a un HALT poniendo la línea de BA en estado alto indicando con esto a los dispositivos periféricos, el estado actual de él. Si BA se encuentra en estado bajo, el MPU está en proceso de ejecutar el programa de control, y si está en bajo, el MPU se encuentra en estado de HALT (Inactivividad).

Registros del MPU.

Contador de Programa (Program Counter): Este es un registro de 16 bits que contiene la dirección de la siguiente instrucción que debe de ejecutar el programa.

Apuntador de pila (Stack Pointer): Es un registro de 16 bits que contiene la siguiente dirección disponible en memoria para la pila del sistema.

Registro de Indice (Index Register): Es de 16 bits y es usado para almacenar temporalmente localidades de memoria o direcciones para ser usados con el modo de direccionamiento indexado.

Acumuladores: Este MPU cuenta con dos acumuladores de 8 bits para uso general en la programación. Aquí se guardan operandos y operadores para el ALU (Unidad Lógico Aritmética).

Registro de Código de Condición (Condition Code Register): El registro de código de condición indica las condiciones resultado de una operación en el ALU: Acarreo (C) bit 1, Overflow (V) bit 2, Cero (Z) bit 3, Negativo (N) bit 4, Interrupción (I) bit 5 y Medio Acarreo (H) bit 4.

APENDICE B

EL ADAPTADOR DE INTERCONEXION PARA COMUNICACION ASINCRONA (ACIA MC6850)

El ACIA proporciona el formato y el control para la interfase de comunicación asíncrona de datos en sistemas con organización de canal como el MPU 6800.

El canal del 6850 incluye líneas de selección, habilitación, lectura/escritura, interrupciones y lógica de interfase de canal para permitir la transferencia de datos sobre un canal de 8 bits. El dato paralelo del sistema es convertido a formato serie para su transmisión por la interfase con la configuración de palabra adecuada . La configuración del ACIA es programada vía el canal de datos durante la iniciación del sistema. Un registro programable de control permite tener varias longitudes de palabra en la transmisión, división del reloj, control de transmisión, recepción e interrupciones. Para la operación de periféricos y modems, el ACIA tiene tres líneas de control que permiten ser conectados directamente.

Características:

- 8 y 9 bits de transmisión
- Paridad par o impar opcional
- Registro de control programable
- Modos opcionales de división de reloj
- Más de 1 Mbps de transmisión
- Borrado de un bit de inicio falso
- Funciones de control Modem/Periférico
- Doble etapa de buffers
- Operación con uno o dos bits de alto.

Operación del dispositivo.

Para el conexionamiento del canal, el ACIA aparece como dos direcciones de memoria para el MPU. Internamente hay cuatro registros: Dos de ellos son sólo de lectura y los otros dos sólo de escritura. Los primeros son el Registro de Estado y el Receptor de Datos (Status and Receive Data) y los últimos son el registro de Control y el de Transmisión (Control and Transmit). El modulo de comunicación serie consiste de una entrada serie y

siendo leído para determinar cuando otro caracter está disponible sobre el RDR. El receptor también se provee de doble buffer, así otro caracter puede leerse del registro de datos mientras otro caracter está siendo recibido en el registro de corrimiento. La secuencia continúa hasta que todos los caracteres han sido recibidos.

Funciones de Entrada/Salida.

Señales de Interconección con el MPU. El ACIA se interconecta con el MPU 6800 directamente a través de un canal de datos de 8 bits, tres líneas de selección de Chip, una línea de selección de registros, una línea de requerimiento por interrupción, una línea de Lectura/Escritura y una línea de habilitación. Todas estas líneas permiten al MPU tener un control total sobre el ACIA.

Líneas de Datos Bidireccionales (D0-D7): Estas líneas permiten la tranferencia de datos entre el ACIA y el MPU. Los manejadores del canal de datos son dispositivos de tres estados que manejan el estado de alta impedancia excepto cuando el MPU realiza una operación de lectura al ACIA.

Habilitación (Enable E): La señal de habilitación, E, es una señal de entrada, compatible TTL y de alta impedancia que habilita los buffers del canal de datos y los relojes de datos. Esta señal generalmente se toma de la fase 2 del reloj del MPU (PHI 2).

Lectura/Escritura (R/W): Es una línea de tres estados y compatible TTL, se usa para indicar el sentido del flujo de los datos del MPU. Cuando R/W se encuentra en alto (ciclo de lectura del MPU), los manejadores de las salidas se encuentran prendidos y los registros de lectura seleccionados. Cuando está en bajo, los manejadores de las salidas están apagados y el MPU puede seleccionar los registros de escritura. Por lo anterior esta línea se usa para seleccionar los registros de sólo lectura o sólo escritura según sea el caso.

Selección de Chip (Chip Select CS0, CS1 y CS2): Las tres líneas son de tres estados y compatibles con TTL. se usan para seleccionar al ACIA y ésto sucede cuando CS0 y CS1 pasan a su estado alto y CS2 a su estado bajo dado que ésta última es verificada baja. La transferencia de información se hace estando seleccionada el ACIA bajo el control de R/W, E y la selección de registros.

Selector de Registros (Register Select RS): Es una línea de tres estados, compatible con TTL. Estando en alto se seleccionan los registros de Recepción/Transmisión de datos y en bajo los registros de Control/Estado. Usada conjuntamente con R/W sirve para seleccionar los correspondientes registros de lectura o escritura.

Requerimiento por interrupción (Interrupt Request IRQ): Es

una línea de colector abierto verificada baja que se usa para interrumpir al MPU. Esta permanecerá en estado bajo mientras la la causa de la interrupción esté presente y la apropiada habilitación de interrupción dentro del ACIA se encuentre encendida. El bit de estado IRQ, en estado alto, indica que la salida IRQ se encuentra en el estado activo.

Las interrupciones pueden ocurrir tanto en la etapa de recepción como en la de transmisión del ACIA. La etapa de transmisión provoca una interrupción cuando la habilitación de interrupción de transmisor es seleccionada (CR5•CR6) y el estado del bit de estado del registro de transmisión de datos vacío (TDRE) se encuentra en estado alto. El estado del bit TDRE (Transmitter Data Register) indica el estado actual de registro de transmisión de datos excepto cuando es inhibido y esto sucede cuando la señal Clear-To-Send (CTS) está en alto o cuando el ACIA se encuentra en la condición de RESET. La interrupción se limpia cuando se escribe o se lee al registro de datos. La interrupción es enmascarable deshabilitando la interrupción del transmisor via CR5 o CR6 o por la pérdida de CTS la cual inhibe el bit de estado TDRE. La sección del receptor causa una interrupción cuando la habilitación de interrupción del receptor se encuentra habilitada y el bit de estado de registro lleno del receptor de datos (RDRF Receive Data Register Full) está prendido, cuando ocurre una sobre-corrida o durante la pérdida de un Data Carried Detect (DCD). La interrupción se limpia durante una lectura al registro de estado después de que condición de error a ocurrido, leyendo el registro de recepción de datos o durante la condición de RESET. La interrupción del receptor es enmascarable limpiando la habilitación de interrupción del receptor.

Entradas de Reloj.

Reloj de Transmisión (TxCLK): El reloj de trasmisión es una señal de entrada para sincronizar la transmisión de los datos. El transmisor incia la secuencia de transmisión de un dato en la transición negativa del reloj.

Reloj de Recepción (RxCLK): El Reloj de recepción es una entrada para la sincronización en la recepción de datos (en el modo de división ÷ 1, el reloj y los datos deben de ser sincronizados externamante). El receptor muestrea los datos en la la transición positiva del reloj.

Líneas de Entrada/Salida Serie.

Recepción de Datos (RxData): La línea de recepción de datos es de alta impedancia y compatible con TTL, a través de la cual los datos son recibidos en el formato serie. Se usa un reloj de sincronización interna cuando las razones de división son ÷ 16 o ÷ 64.

Transmisión de Datos (TxData): La salida de datos en el formato serie hacia el exterior se lleva a cabo a través de esta línea.

Control de Periféricos.

Limpia-para-enviar (Clear-To-Send CTS): Es una línea de alta impedancia y compatible TTL, proporciona control automático del fin de la transmisión de una comunicación encadenada vía modem, se activa en bajo por la inhibición del bit de estado TDRE.

Requerimiento-para-Enviar (Request-To-Send RTS): Esta salida habilita al MPU para controlar al periférico o modem vía el canal de datos. La salida RTS corresponde al estado de los bits de control CR5 y CR6. Cuando CR6=0 o ambos CR5=CR6=1, la salida RTS se activa. Esta salida puede usarse para el Data Terminal Ready (DTR).

Detección de Transporte de Datos (Data Carrier Detect DCD): Es de alta impedancia y compatible con TTL, dá control automático sobre el fin de recepción de una comunicación encadenada por medio de una salida de detección de transporte de datos del modem. El DCD inhibe e inicia la sección de recepción del ACIA cuando está en alto. Una transición al estado bajo de esta señal provoca una interrupción al MPU para indicar la pérdida del portador cuando la habilitación de interrupción de receptor se encuentra encendida. El RxCLK debe de estar corriendo adecuadamente para una operación apropiada de DCD.

Registros Internos del ACIA.

Registro de Transmisión de datos (Transmit Data Register TDR): Los datos se escriben en el TDR durante la transición negativa del Enable (E) cuando el ACIA ha sido direccionada con RS en alto y R/W en bajo. La escritura de datos en el TDR causa que el bit TDRE del registro de estado, vaya a estado bajo. Los datos pueden, entonces, ser transmitidos. Si el transmisor está desocupado y ningún caracter está siendo transmitido, entonces la tranferencia tendrá lugar durante el tiempo de 1-bit de la transición de un comando de escritura. Si una caracter está siendo transmitido, el nuevo caracter comenzará a ser transmitido hasta que la transmisión del primero se haya completado. La transferencia de datos causa que el bit TDRE del registro de estado indique vacío.

Registro de Recepción de Datos (Receive Data Register RDR): Los datos son transferidos automáticamente al RDR desde el receptor que convierte el formato serie (un registro de corrimiento) una vez que el caracter fue recibido completamente. Esto causa que el bit de estado RDRF vaya estado alto. Los datos pueden ser leídos a través del canal de datos mediante la selección y direccionamiento del RDR del ACIA con RS y R/W en alto cuando el ACIA se encuentra habilitada. El ciclo de lectura no destructiva causa qu el bit de estado RDRF se limpia indicando con esto que el registro de recepción de datos (RDR) se encuentra vacio reteniendo el último dato en el RDR. Cuando el registro de recepción de datos está lleno, la transferencia automática de datos del registro de corrimiento de recepción al registro de

datos es inhibida y el RDR contiene el remanente válido con su correspondiente estado cargado en el registro de estado.

Registro de Control (Control Register CR): El registro de Control del ACIA es de 8 bits solamente de lectura. Es seleccionado cuando RS y R/W están el estado bajo. Este registro controla las funciones del receptor, transmisor, habilitaciones de interrupciones y la salida de requerimiento-para-enviar (RTS) de los periféricos o modem.

-Bits de Selección de División de Contador (CR0 y CR1)- Estos determinan las razones de división a ser ultilizadas tanto en receptor como en el transmisor. Adicionalmente, estos bits son usados para dar un reset maestro al ACIA el cual limpia el registro de estado (excepto para condiciones externas sobre el DCD y el CTS) e inicia al receptor y al transmisor. El reset maestro no afecta a los otros bits del registro de control. Hay que notar que después del encendido, de una falla de alimentación o de un reset, estos bits deben ser puestos en alto para restablecer el ACIA. Después de un reset, la división del reloj debe de ser nuevamente seleccionada. La selección de las razones de división se hacen de acuerdo a la tabla B.1.

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

Tabla B.1

-Bits de Selección de Palabra (CR2, CR3 y CR4)- Estos bits seleccionan el formato de la palabra para la transmisión y recepción de acuerdo a la tabla B.2

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

Tabla B.2

Los cambios hechos al formato de la palabra no se registran en ningún buffer, por lo tanto cualquier cambio se hace efectivo inmediatamente.

-Bits de Control de Transmisor (CR5 y CR6)- Sirven para controlar las interrupciones de la condición de registro de transmisión de datos vacio, la salida de requerimiento-para-envio (RTS) y la transmisión de nivel de Break (espacio).

CR4	CR2	Función
0	0	RTS=BAJO, interrupción deshabilitada
0	1	RTS=BAJO, interrupción habilitada
1	0	RTS=ALTO, interrupción deshabilitada
1	1	RTS=BAJO, Transmite un nivel de break en la salida de transmisión de datos, interrupción deshabilitada

Tabla B.3

-Bit de Habilitación de Interrupción (CR7)- Las siguientes interrupciones son habilitadas con el bit CR7 en alto: el RDRF, sobre-corrida o la transición de bajo a alto en la señal del DCD.

Registro de Estado (Status Register ST)1 La información acerca del estado del ACIA esta disponible para el MPU leyendo este registro. Es solamente de lectura y se selecciona cuando RS

caracter válido antes de la sobre-corrida no haya sido leído. El bit RDRF permanece encendido hasta que el OVRUN se limpia. El caracter de sincronización se mantiene durante la condición de sobre-corrida. La indicación de la sobre-corrida es limpiada después de la lectura del dato del RDR o por un reset maestro.

-Bit 6, Error de Paridad (Parity Error PE)- La bandera de error de paridad indica que el número de altos (unos) en el caracter no corresponde a la paridad preseleccionada, par o impar. Paridad Impar se define como un número impar de unos en el caracter. La indicación de un error de paridad aparecerá de acuerdo a la longitud del dato en el RDR. Si no se selecciona paridad, entonces tanto el generador de paridad del transmisor como el verificador de paridad del receptor quedarán inhibidos.

-Bit 7, Requerimiento por interrupción (Interrup Request IRQ)- Este bit indica el estado de la salida IRQ. Cualquier condición de interrupción con su correspondiente habilitación será reflejada en este bit. Cuando la salida IRQ se encuentra en bajo el bit de IRQ estará en alto la interrupción o el estado de un requerimiento de servicio. El bit IRQ se limpia por una operación de lectura al RDR o una operación de escritura al TDR.

APENDICE C

EL ADAPTADOR DE INTERFASE PERIFERICO MC6821 (PIA).

Este dispositivo permite interconectar al MPU con otros dispositivos periféricos que manejen hasta dos canales de 8 bits y cuatro líneas de control. La configuración de una PIA se hace dentro de la rutina de iniciación (RESET), pero puede ser modificada en cualquier punto del programa de control. Cada una de las líneas de datos puede ser programada como salida o como entrada.

Características:

- Canal de datos de 8 bits para comunicación con el MPU.
- Dos canales de datos de 8 bits cada uno para conectarse a periféricos.
- Dos registros de control programables.
- Dos registros de dirección de datos programables.
- Cuatro líneas individuales para control de interrupciones.
- Lógica de Hand-shake para la operación entrada/salida de los dispositivos periféricos.
- Líneas de periféricos de alta impedancia con manejo directo de transistor.
- Interrupciones controladas por programa con la capacidad de deshabilitación
- Capacidad para manejar CMOS en el puerto periférico A.
- Capacidad para manejar lógica TTL en los puertos A y B.
- Compatible con TTL
- Operación estática

Descripción de las señales de el PIA:

Canal de Datos Bidireccionales (D0-D7): Este conjunto de líneas permite transferir datos de el PIA al MPU. Todas estas líneas tienen estado de alta impedancia. La línea de R/W esta en alto mientras es seleccionada para una operación de lectura.

Habilitación (Enable E): El pulso de habilitación E, es la única señal de tiempo que el PIA requiere. Cualquier otra referencia a tiempos es para las transiciones de bajada y subida de E.

Lectura/Escritura (R/W): Esta señal se genera en el canal de

control del MPU para controlar la dirección del flujo de de información. Esta línea se encuentra en alto mientras el PIA es seleccionada para una operación de lectura.

RESET: Es activa baja y se usa para reestablecer las condiciones iniciales de el PIA. Esto puede ocurrir en el encendido o en cualquier parte de la ejecución del programa de control. Puede ser tomada directamente de la señal de RESET del MPU para evitar lecturas o escrituras falsas a el PIA.

Selección de Chip (Chip Select CS0, CS1 y CS2): Estas tres señales de entrada se usan para seleccionar a el PIA. CS0 y CS1 son verificadas altas y CS2 es verificada baja. La transferencia de datos se hace una vez seleccionada el PIA y bajo el control de la líneas E y R/W. Las líneas de selección deben permanecer estables durante el pulso de habilitación. Cuando cualquiera de éstas señales pasa a su estado inactivo el PIA no es seleccionada.

Selectores de Registros (Register Select RS0 y RS1): Las dos líneas sirven para seleccionar los registro internos de el PIA. Estas son usadas conjuntamente con uno de los bits del registro de control. También estas señales deben ser estables durante el período activo de E.

Requerimiento por interrupción (Iterrupt Request IRQA e IRQB): Estas señales son activas bajas y de colector abierto, por lo que requieren de una resistencia conectada a Vcc. Los registros de control tienen unos bits que se usan como banderas y que pueden causar que las señales IRQA e IRQB vayan a estado bajo. Estas interrupciones pueden ser inhibidas por medio del registro de control.

La atención a un requerimiento por interrupción puede realizarse por medio de software o bien por Hardware.

Las banderas de interrupciones son llevadas a cero por una acción de lectura por el MPU sobre el registro correspondiente. Después de que han sido limpiados, estos bits sólo pueden ser encendidos cuando el PIA no se encuentre seleccionada durante un pulso de E.

Datos de Periféricos Sección A (Peripheral Data PA0-PA7): Cada una de estas líneas puede ser programada para actuar como salida o como entrada. La programación se hace escribiendo unos o ceros en el Registro de Dirección de Datos (DDR). El "1" corresponde a una salida y un "0" a una entrada. Durante una operación de lectura del MPU, los valores lógicos existentes en la líneas programadas como entradas, aparecen directamente sobre el canal de datos del MPU. En este modo de operación, estas líneas son capaces de manejar hasta 1.5 cargas TTL.

El dato en el Registro de Salida "A" (Output Register) aparecerá en líneas de éste que fueron programadas como salidas. Un dato escrito sobre el Registro de Salida "A" puede ser leido

x = no importa

		BIT 2 CR		
RS1	RS0	CRA 2	CRB 2	REGISTRO SELECCIONADO
0	0	1	x	Reg Periférico A
0	0	0	x	Reg de Direcciones A
0	1	x	x	Reg de Control A
1	0	x	1	Reg Periférico B
1	0	x	0	Reg de Direcciones B
1	1	x	x	Reg de Control B

Tabla C.1

Registros de Control (CRA y CRB): Los dos registros de control de el PIA permiten al MPU controlar la operación de las cuatro líneas de control CA1, CA2, CB1 y CB2. Los Bits 0 al 5 de éstos registros pueden ser escritos o leídos por el MPU cuando el PIA se encuentra seleccionada y selecionado también el registro de control mediante las líneas RS0 y RS1. Los Bits 6 y 7 de los dos registros sólo son de lectura y se modifican cuando hay requerimientos por interrupción sobre las líneas de control antes mencionadas.

-Bit de Control de Acceso al Registro de Dirección de Datos (CRA-2 y CRB-2): El Bit 2 de cada uno de los registros de control determinan cuando se está seleccionando, sobre la misma localidad, al Registro de Dirección de Datos o al Registro Periférico de Salida. Un "1" en éste bit selecciona al Registro Periférico de Datos y un "0" al Registro de Dirección de Datos.

-Banderas de Interrupción (CRA-6, CRA-7, CRB-6 y CRB-7): Las cuatro banderas de interrupción son prendidas en la transición activa sobre las cuatro líneas de control del Periférico, cuando estas líneas son programadas como entradas. Estos bits no pueden ser encendidos directamente a través del canal de datos del MPU y se limpian mediante una lectura al registro.

-Control de las líneas de control de Periféricos CA2 y CB2 (CRA-3, CRA-4, CRA5, CRB3, CRB4 y CRB5): Los bits 3,4 y 5 de ambos registros de control son usados para realizar el control de las líneas de control de Periféricos. Estos bits determinan si estas líneas de control actuarán como salidas o como entradas.

-Control de las líneas de interrupciones de entrada CA1 y CB1 (CRA-0, CRB-0, CRA-1 y CRB): Los bits CRA-0 y CRB-0 son usados para habilitar las señales de interrupción del MPU (IRQA e IRQB) respectivamente. Los bits CRA-1 y CRB-1 determinan la transición activa de las señales de interrupción de entrada CA1 y CB1.

Bibliografía.

[L.1] "Digital Computer Fundamentals"
Bartee, Thomas C., Ed. McGrawHill 1981.

[L.2] "Basic Electronics For Scientists"
Brophy, James J., Ed. McGraw-Hill 1977.

[L.3] "Circuitos Integrados Lineales y Amplificadores Operacionales"
Coughlin, Robert F., Ed. Prentice-Hall 1987.

[L.4] "An Engineering Approach To Digital Design"
Fletcher, William I., Ed. Prentice-Hall 1980.

[L.5] "Sistema Basados en Microprocesador"
Gault, James W., Ed. McGraw-Hill 1983.

[L.6] "Modern Logic Design"
Green, David

[L.7] "Microprocessors And Interfacing"
Hall, Douglas V., Ed. McGraw-Hill 1986.

[L.8] "Teoría de Conmutación y Diseño Lógico"
Hill, Frederik J. Ed. Limusa 1974.

[L.9] "Diseño Digital"
Mano, Morris M., Ed. Prentice-Hall 1987.

[L.10] "Lógica Digital y Diseño de Computadores"
Mano, Morris M., Ed. Prentice-Hall 1979.

[L.11] "Fundamentos de Electrónica"
Norman, E., Ed. C.E.C.S.A 1978.

[L.12] "Microcomputer Based Design"
Peatmean, John B., Ed. McGraw-Hill.

[L.13] "Electrónica Digital"
Strangio, C.E., Ed. InterAmericana 1984.

[L.14] "Sistemas Digitales Principios y Aplicaciones"
Tocci, Ronald J., Ed. Prentice-Hall 1987.

Manuales.

[M.1] "Memory Data Book"
National Semiconductor 1980

[M.2] "The TTL Data Book"
Texas Instruments Incorporation 1978

[M.3] "MC6800 Microproccesor Aplications Manual"
Motorola Corporation Products 1975

[M.4] "MC6800 EXORciser User's Guide"
Motorola Corporation Products 1975

[M.5] "MC6809 EXORciser User's Guide"
Motorola Corporation Products 1975

[M.6] "MC6800 Programming Refernce Manual"
Motorola Corporation Products 1979

[M.7] "Macro Assembler Reference Manual 6800,6801,6805 y 6809"
Motorola Corporation Products 1979

[M.8] "AMI6800 Hardware Refernce Manual"
American Microsystem, Inc. 1976

[M.9] "Disk Operating System Flex Ver. 1.0 User's Guide"
Southwest Technical Product's Coorporation 1978

[M.10] "MP-R Programmer User's Guide"
Southwest Technical Product's Coorporation 1978

[M.11] "Televideo Operating Instrucctions Model 912"
Televideo N.C. 1979

[M.12] "Linear Interface Integrated Circuits"
Motorola Corporation Products 1982

[M.13] "Motorola Microprocessors Data Manual"
Motorola Corporation Products 1981

[M.14] "6800 ROM Monitor Version 1.0"
Southwest Technical Product's Coorporation 1978

[M.15] "M6809 MPU Module User's Guide"
Motorola Corporation Products 1979

[M.16] "M6809DB Debug Module User's Guide"
Motorola Corporation Products 1979

[M.17] "MEK6802D3 Microcomputer Unit"
Motorola Corporation Products 1979

[M.18] "Assembly Instructions, Motorola MC6800 Microcomputer System Design Evaluation Kit".
Motorola Corporation Products 1975.

Lista de Figuras por Capitulo.

Capitulo 2.

Capitulo 3.

Lista de Diagramas.

Capítulo 2.