UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Diseño y construcción de una tarjeta emuladora de EPROM´s.

Tesis Profesional
que para obtener el título de

Ingeniero en Electrónica

Presenta:

Miriam Cuevas Cuevas.

asesor
M.C. Enrique Guzmán Ramírez

Acatlima, Huajuapan de León, Oax. Junio de 1999.

Tesis presentada ante los siguientes sinodales:

M.C. Felipe Santiago Espinosa.

M.C. Jacob Vázquez Sanjuan.

M.C. Enrique Guzmán Ramírez.

A MIS PADRES

Gracias porque me han enseñado a ser mujer. Me han enseñado que ante todos los problemas y adversidades teniéndolo todo para perder, el darse por vencida nunca es la solución.

Me han enseñado a arriesgar lo poco que se tiene en pos de conseguir algo mejor, dándome ejemplo de no pecar de soberbia si triunfo, y educando mi capacidad de afrontar frustraciones y derrotas sin quejas ni ira al ser vencida.

Me han enseñado que en esta vida triunfa el que transciende, fracase o no. Aquel que logra avanzar poco a poco, pero sin aportar nada a los demás es un derrotado.

Me han corregido suave y fuerte en mis momentos de desorientación, siempre estando a mi lado cuando los he necesitado, en los momentos de felicidad para alentarme y en mis momentos de tristeza para consolarme y aconsejarme.

Gracias por legarme una personalidad de servicio y entrega, pues han dejado sus diversiones por darme hasta lo que no tienen.

Me han enseñado a tener sangre fría en los momentos de crisis y cautela y honor en los momentos grandes. Me han respetado mi individualidad y más aun, me han enseñado a no cometer sus errores invitándome a seguir su camino de aciertos.

Gracias a Dios por tener unos padres como ustedes, un hogar, una familia...

Los Ama... Miriam

A MI FAMILIA

Que han sido un ejemplo para mi.

Laura y Marco Antonio me han enseñado la unión y el amor de una familia. Mis pequeños sobrinos Wendy, Evelin y Marco Antonio (los ñoños) que son la alegría de la casa.

Oscar y Carmen me han enseñado la superación y fortaleza para alcanzar las metas propuestas.

René que a pesar de su lejanía siempre está con nosotros

Mi abuela por su gran sabiduría.

A OSCAR

Que es el amor de mi vida, gracias por su amor, apoyo y comprensión.

Miriam.

Agradecimientos

Agradezco a todas las personas que a lo largo de mi vida me han brindado su amistad limpia y sincera.

A todos mis maestros y académicos que durante mi carrera me brindaron sus conocimientos, así como su ayuda para aclarar mis dudas.

Gracias al Ing. Raúl Placencia, M.C. Miguel Matilla Martín y M.C. Felipe de Jesús Rivera López por el apoyo incondicional, por guiarme en esta lucha de superación y más aun por confiar en mi.

Agradezco al M.C. José Ignacio Castillo, M.C. Felipe Santiago Espinosa y al M.C Jacob Vásquez, por el apoyo y ayuda en la realización y conclusión de esta tesis

A mis amigos que siempre me apoyaron en la realización de esta tesis

ÍNDICE

PROLOGO I

CAPÍTULO 1: INTRODUCCIÓN

1.1 *Antecedentes* *1-1*
1.2 *Objetivos de la tesis* *1-1*

CAPÍTULO 2: GENERALIDADES DE LAS MEMORIAS 2-4

2.1 *Características* *2-5*
2.2 *Tipos de memorias* *2-6*
2.3 *Clasificación* *2-10*
2.4 *Operación de la memoria* *2-11*

CAPÍTULO 3: TARJETAS EMULADORAS DE EPROM'S 3-14

3.1 *Descripción de las tarjetas emuladoras* *3-14*
3.2 *Estructuras de las tarjetas emuladoras* *3-15*
3.3 *Funciones de las tarjetas emuladoras* *3-17*
3.4 *Ventajas y desventajas* *3-18*
3.5 *Aplicaciones* *3-18*

CAPÍTULO 4: DESCRIPCIÓN DE DISEÑO DEL HARDWARE Y SOFTWARE DE LA TARJETA 4-19

4.1 *Características generales* *4-19*
4.2 *Diagrama a bloques de la tarjeta* *4-20*
4.3 *El circuito MC68HC11* *4-20*
4.3.1 Modos de operación 4-22
4.3.2 Memoria interna 4-23
4.4 *Organización de la memoria* *4-24*
4.5 *Bus de control* *4-25*
4.6 *Bus de direcciones* *4-27*
4.7 *Bus de datos* *4-28*
4.8 *Alimentación* *4-28*
4.9 *Comunicación serial* *4-29*
4.10 *Decodificador de direcciones* *4-30*
4.11 *Descripción del software* *4-31*
4.11.1 Comunicación con la PC 4-31
4.11.2 Inicialización 4-31
4.12 *Comandos* *4-35*
4.13 *Subsistemas utilizados* *4-36*

4.14 *Diagrama de flujo* 4-37

CAPÍTULO 5: FUNCIONAMIENTO 5-47

5.1 *Requerimientos* 5-47
5.2 *Configuración* 5-48
5.3 *Operación* 5-50
5.3.1 Tarjeta emuladora 4-50
5.3.2 Tarjeta de evaluación 5-52

CAPÍTULO 6: CONCLUSIONES 5-56

APÉNDICE A DIAGRAMA ESQUEMÁTICO Y DISEÑO DE LA TARJETA A-60
APÉNDICE B MANUAL DEL USUARIO B-66
APÉNDICE C PROGRAMA EN FORMATO OPAL C-72
APÉNDICE D LISTADO DEL PROGRAMA D-74

GLOSARIO TÉCNICO G-96

BIBLIOGRAFÍA 102

PRÓLOGO

El desarrollo de este trabajo de tesis se realiza en términos de la importancia que han adquirido los microcontroladores en los sistemas modernos.

La tarjeta emuladora de EPROM's se presenta como una herramienta de doble función; como *tarjeta emuladora* es una herramienta para la realización de sistemas que necesiten de una memoria EPROM, generalmente el primer programa de prueba no es el correcto, el tiempo consumido en el proceso de borrado y reprogramación es relativamente mayor a utilizar el emulador y como *tarjeta de evaluación* nos permite el estudio teórico y práctico del microcontrolador MC68HC11E1 de Motorola.

La información presentada en este trabajo, puede guiar a otros compañeros estudiantes en el proceso de diseño de "hardware" y "software" aplicado a sistemas más innovadores y complejos, donde un microprocesador ó microcontrolador represente al dispositivo principal para su diseño, principalmente en sistemas de control. Contribuyendo de esta forma con una herramienta para el estudio de uno entre tantos microcontroladores.

M.C.C.

CAPÍTULO 1

1.1 ANTECEDENTES

Los Sistemas Digitales basados con microprocesadores se han vuelto cada vez más populares debido a la realización de múltiples aplicaciones con ellos. Para facilitar el diseño y construcción de dichos sistemas se requiere hacer uso de herramientas como: Tarjetas de Evaluación - la cual es un ensamble de circuito impreso y contiene los elementos básicos de un circuito genérico del propio microprocesador o microcontrolador-, y la tarjetas Emuladoras - las cuales permiten la simulación, verificación, desarrollo y programación de prototipos -.

Este tipo de herramientas son útiles hoy en día en todo laboratorio de electrónica porque cuando se usa una de estas tarjetas, la mayoría del diseño del circuito, fabricación de tarjetas y pruebas, ya han sido hechas. Sólo se necesitan agregar los componentes para la aplicación en específico y programar el circuito de control.

Comparado a otros métodos en diseño y construcción, estas tarjetas pueden reducir el tiempo de implementación de un sistema basado en microprocesadores o microcontroladores, ya que es fácil modificar, optimizar y corregir su software.

1.2 OBJETIVOS DE LA TESIS

El objetivo de este trabajo fue realizar una TARJETA EMULADORA DE EPROM's con las siguientes características:

1. Utilizar el microcontrolador MC68HC11E1.
2. Reducir el tamaño del sistema como sea posible.
3. Los circuitos integrados para el funcionamiento de la tarjeta deben ser comerciales.
4. El sistema debe ser capaz de retener la información aun después de retirar la fuente de alimentación.
5. Flexibilidad en la habilitación de los dispositivos para ser colocados en cualquier parte del mapa de memoria.
6. Comunicación serial con una computadora personal.
7. La tarjeta debe emular EPROM's de 2 Kbytes hasta 32 Kbytes

8. Tener un RESET por hardware y software en caso de fallas ó para reinicializar el sistema.
9. Flexibilidad en la alimentación de la Tarjeta de 5 volts a 35 volts.

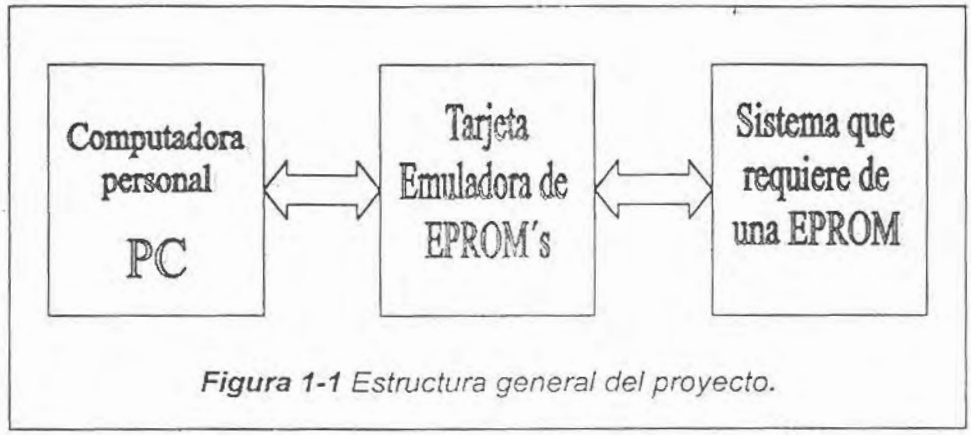


Figura 1-1 *Estructura general del proyecto.*

Con forme a este contexto se plantea el desarrollo de la presente tesis, la tarjeta Emuladora de memorias del tipo "*Erasable Programmable Read Only Memory*, Memoria Programable y Borrable de Solo Lectura " (**EPROM**) de 8 bits de datos, además es una Tarjeta de Evaluación para el microcontrolador MC68HC11E1.

Está tarjeta Emuladora de EPROMs trabaja en conjunto con una computadora personal a través de un puerto serie, la interface entre la PC y la tarjeta es mediante un programa de comunicaciones que permite interactuar con el usuario; cuenta con una conexión para cable plano y éste se conecta al lugar destinado para el EPROM del sistema mínimo o huésped.

La tarjeta podrá Emular EPROM's de las siguientes capacidades: 2716 (2Kb), 2732 (4Kb), 2764 (8Kb), 27128 (16Kb) y 27256 (32Kb).

Así como podrá emplearse para desarrollar diversas aplicaciones, convirtiéndose de esta forma en una herramienta de enseñanza capaz de responder a las demandas de competitividad y productividad requeridas en el marco académico, podrá ser utilizada en diferentes cursos de la carrera de Ingeniería en Electrónica y Computación que se imparten es esta casa de estudios. En donde los alumnos podrán poner en práctica los conocimientos

adquiridos y fomentar de esta forma su creatividad para que experimenten y desarrollen prototipos de una manera sencilla.

Este documento presenta el diseño de la tarjeta, la elaboración de los programas y la documentación necesaria para su reproducción.

La información se distribuye de la manera siguiente:

En el capitulo 2, se describen los diferentes dispositivos de memorias, además de sus características y operación.

En el capitulo 3, se explican algunas de las diferentes estructuras de Tarjetas Emuladoras existentes, así como la representación de sus diagramas a bloques, las ventajas y desventajas de las diferentes estructuras.

En el capitulo 4, se describen las características generales del sistema, el microcontrolador MC68HC11E1, las memorias internas y externas, etc.

Se explica el mapa de memoria del sistema, puertos, los diversos buses, la comunicación serial, etc. y colocando por último el diagrama de flujo de la Tarjeta Emuladora de EPROMs presentada.

En el capitulo 5, se presentan los requerimientos para el funcionamiento de la Tarjeta, la forma de configurar los diferentes modos de funcionamiento del sistema, presentando además sus diagramas a bloques.

En el capítulo 6, se muestran las conclusiones del sistema, describiendo las pruebas realizadas y proponiendo mejoras a la tarjeta Emuladora.

CAPÍTULO 2

MEMORIAS

Cuando una señal de entrada se aplica a muchos dispositivos o circuitos, la salida cambia de alguna manera como respuesta a la entrada y, cuando se retira la señal de entrada, la salida retorna a su estado original. Estos circuitos no exhiben la propiedad de una memoria, ya que sus salidas regresan al estado normal. En los circuitos digitales, ciertos tipos de dispositivos y circuitos tienen memoria. Cuando una entrada se aplica a tal circuito, la salida cambia de estado, pero se mantiene en éste aun después de que se retire la entrada. Esta propiedad de retención de su respuesta a una entrada momentánea se denomina **memoria**. En la figura 2.1 se realiza la comparacion de una operación que no es de memoria con una que si es.

Los dispositivos y circuitos de memoria desempeñan un papel importante en los sistemas digitales debido a que ofrecen medios para almacenar números binarios temporales o permanentes, con la capacidad de cambiar la información almacenada en cualquier instante. Como observaremos, los diversos elementos de la memoria incluyen los tipos magnéticos y aquellos que utilizan circuitos electrónicos, denominados biestables (flip-flops)

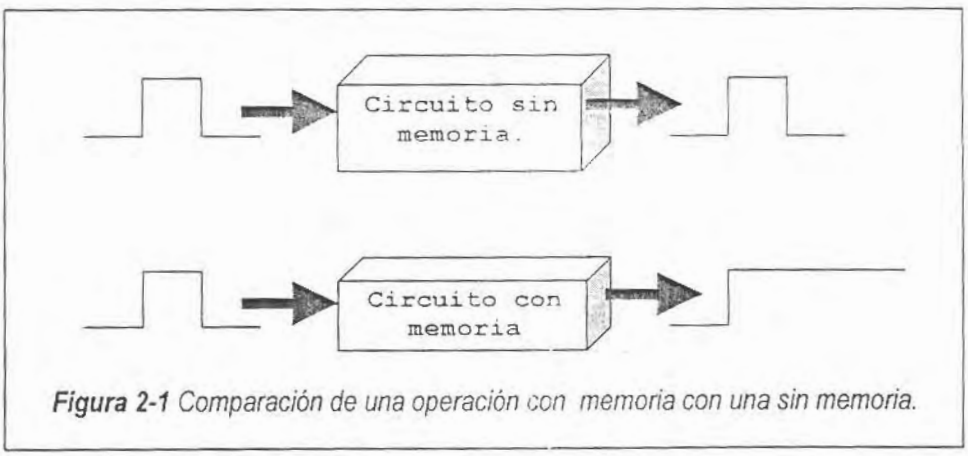


Figura 2-1 *Comparación de una operación con memoria con una sin memoria.*

Los fabricantes de circuitos ofrecen al mercado unidades de memorias de distintas tecnologías, capacidad y características. Estas memorias se presentan en el mercado en pastillas o chips de semiconductores, de forma que llegan al usuario bajo el mismo formato que cualquier otro circuito integrado, con una diferencia notable en el caso de aquellas que presentan una ventana, convenientemente protegidas por un tipo de resina transparente.

2.1 CARACTERÍSTICAS

Cada tecnología para la fabricación de memorias aporta características que le son propias y que el usuario debe evaluar antes de decidir que tipo le conviene para su aplicación concreta. Las características que definen en mayor medida las propiedades funcionales de una memoria son las siguientes:

TIEMPO DE ACCESO

Es el indicativo de la rapidez con que la memoria es capaz de ser leída o escrita. Puede venir expresada en función de un tiempo de acceso máximo, o bien de un tiempo de acceso medio.

TIEMPO DE ESCRITURA

Se le conoce al tiempo transcurrido desde la aplicación de la dirección en la que se pretende escribir una determinada información hasta el momento en que realmente la información ha quedado almacenada en la memoria.

TIEMPO DE LECTURA

Es el tiempo transcurrido desde la aplicación de la dirección en la que se pretende leer su contenido hasta el momento en que la información realmente está disponible en la salida de la unidad de memoria.

TIEMPO DE CICLO DE OPERACIÓN

El tiempo total que la memoria precisa para efectuar una operación de lectura o escritura y quedar en disposición de efectuar una nueva operación. Dicho tiempo es, la suma del tiempo propio de realización de la referida operación más el tiempo de acceso.

CAPACIDAD DE MEMORIA

Esta expresada por el número de bits que la memoria es capaz de almacenar. Un bit es una celda elemental capaz de almacenar un "1" o "0" lógicos.

La capacidad de una memoria también se puede medir en bytes o palabras. Un "byte" es, por definición, igual a ocho bits, y una palabra puede tener 16, 24, 32, o 64 bits, según el fabricante del equipo.

VELOCIDAD DE TRANSFERENCIA

Es la velocidad máxima con la que la memoria es capaz de aceptar información, ya sea para escritura o lectura. La unidad de medida es el bit/segundo.

2.2 TIPOS DE MEMORIAS

Dentro de los tipos más comunes de memorias, se encuentran las siguientes:

MEMORIA DE SÓLO LECTURA (ROM)

La ROM (Read Only Memory), es una memoria cuya información ha sido pre-grabada de manera permanente por el fabricante del circuito. Es no volátil por lo que su información no desaparece aún cuando se suspenda su suministro de energía. Durante su operación normal, únicamente puede ser accesada para lecturas.

Las memorias ROM son comúnmente utilizadas para almacenar instrucciones o constantes numéricas, es decir información que no cambiará durante la vida de un producto. Por ejemplo, las PCs usan una ROM para soportar el llamado BIOS (sistema básico de entrada/salida), que es el encargado de inicializar el sistema. También se utilizan para guardar programas en equipos controlados por microprocesadores tales como: cajas registradoras electrónicas, instrumentos y sistemas de seguridad, etc.

MEMORIA PROGRAMABLE DE SÓLO LECTURA (PROM)

La PROM (Programmable Read Only Memory), son memorias del tipo ROM pero con la diferencia de que pueden ser grabadas por los propios usuarios (el fabricante las suministra sin información en su interior) a través de un *programador de PROMs*.

La PROM se programa al hacer que se fundan fusibles de Nichrome o de óxido de silicio, por lo que una vez programada no se puede borrar. A veces se emplean PROMs durante la producción de equipos si se espera que su programación no cambie o si resulta inconveniente montar unidades ROM.

MEMORIA PROGRAMABLE Y BORRABLE DE SÓLO LECTURA (EPROM)

Una EPROM (Erasable Programmable Read Only Memory) es una memoria PROM que puede ser borrada y vuelta a grabar. Suele construirse mediante la tecnología del nitruro metálico sobre silicio (MNOS).

La EPROM se programa con un dispositivo llamado programador de EPROM y su contenido se borra exponiéndola a la luz ultravioleta (UV), la cual se aplica a través de la ventana que se encuentra sobre el encapsulado del circuito (ver figura 2.2).

El proceso de borrado requiere exponer al EPROM a los rayos UV durante unos 20 minutos o menos, según sea el tipo de EPROM. No existe alguna forma de borrar sólo algunas celdas; la luz UV borra todas las celdas al mismo tiempo [DDPP, Pág. 577- 579] [SD, Pág. 656].

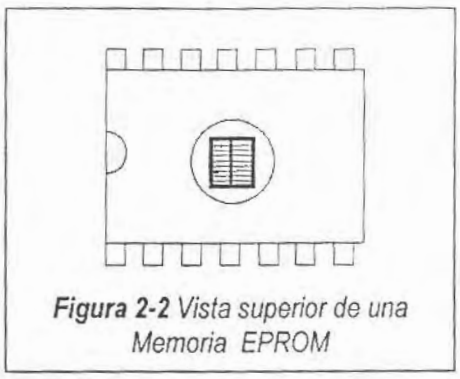


***Figura 2-2** Vista superior de una Memoria EPROM*

Una EPROM tiene un *transistor* MOS de compuerta flotante en cada posición de bit. La compuerta flotante está rodeada por material aislante de alta impedancia. Para programar una EPROM, el programador aplica un alto voltaje al otro extremo de la compuerta, en cada posición de bit donde se almacenará un "0" (cero lógico), lo que ocasiona una ruptura en el material aislante y permite que se acumule una carga negativa. Cuando se suprime el alto voltaje, la carga negativa permanece. Durante las subsecuentes operaciones de lectura, la carga negativa evita que el transistor MOS se active cuando se selecciona.

El material aislante a la compuerta flotante se hace ligeramente conductor si se expone a la luz ultravioleta de cierta longitud de onda.

En la serie 27XXX de números de piezas para las EPROM se incluyen las siguientes: 2704 (512 x 8); 2708 (1K x 8); 2716 (2K x 8); 2732 (4K x 8); 2764 (8K x 8); 27128 (16K x 8); 27256 (32K x 8); 27512 (64K x 8) y 271024 (128K x 8). Cada una de esas piezas tiene terminales para dirección, 8 conexiones para datos, una entrada de selección (/CE) y una terminal de habilitación de salida (/OE).

Los datos sólo aparecen en las conexiones de salida hasta después de hacer un 0 lógico en las terminales de selección y habilitación. Si alguna de ellas no es cero, las conexiones de salidas de datos permanecen en su estado de alta impedancia. Se debe tener en cuenta que la terminal Vpp debe colocarse en 1 lógico para que la EPROM lea los datos. En algunos casos la terminal Vpp está en la misma posición que la terminal de escritura (/WE) para una SRAM

MEMORIA ROM PROGRAMABLE, ELÉCTRICAMENTE BORRABLE (EEPROM)

Estas memorias son borrables y gravables, pero con la salvedad de que el borrado y la grabación puede hacerse por procedimientos eléctricos dentro del circuito de utilización, esto facilita enormemente su manipulación. Las siglas EEPROM provienen de la denominación: Electrical Erasable Programmable Read Only Memory. La memoria EEPROM se utiliza para almacenar información que puede ajustar a un sistema, por ejemplo la tarjeta de vídeo en la computadora.

MEMORIA DE SOLO LECTURA, PROGRAMABLE (REPROM)

Las memorias REPROM, son memorias PROM que pueden ser borradas y vueltas a grabar.

Debido a que las operaciones de borrado y reprogramación no son sencillas, estas memorias se usan, en su mayor parte, para ser leídas únicamente, con la considerable ventaja de que, si es preciso, pueden efectuarse en ellas operaciones de borrado. Debido a esta peculiaridad funcional las memorias REPROM se denominan también RMM, o sea memorias utilizadas para ser leídas. Esta denominación corresponde a la simplificación de las palabras inglesas *Read Mostly Memory* [MS].

MEMORIA DE ACCESO ALEATORIO (RAM)

La información contenida en este tipo de memorias puede ser "leída" y "modificada" en cualquier momento, es fácilmente programada, borrada y reprogramada por el usuario. La información almacenada en una RAM es temporal o volátil, es decir que pueden almacenar datos mientras se aplica energía al circuito.

El termino acceso "aleatorio" significa, que se puede acceder a cualquier byte de la memoria sin pasar por los bytes precedentes, por lo tanto, el tiempo invertido para la

lectura/escritura de un dato es el mismo sin importar su dirección. En contraste, en un dispositivo de almacenamiento secuencial, por ejemplo un manejador de cinta, el tiempo de acceso depende de la localización de la información deseada.

Por las características de las memorias RAMs, son comúnmente utilizadas en las computadoras como medio de almacenamiento temporal para programas y datos. El contenido de muchas de las localidades de dirección de la RAM será leído y escrito a medida que la computadora ejecuta un programa. Esto requiere que la RAM tenga ciclos de escritura y lectura rápidos para que no reduzca la velocidad de operación de la computadora.

Hay dos tipos básicos de memorias RAM: (a) las RAM estáticas o **SRAM** (*Static* RAM) y (b) las RAM dinámicas o **DRAM** (*Dynamic* RAM). Cada una tiene beneficios e inconvenientes particulares.

RAM ESTÁTICA Y DINÁMICA

La ***RAM estática*** utiliza como su celda de almacenamiento básica un multivibrador S-R (de colocación y recolocación); es decir son en esencia flip-flops, mientras que la ***RAM dinámica*** utiliza un capacitor MOS como su elemento de almacenamiento, los datos almacenados desaparecerán gradualmente debido a la descarga del capacitor, de manera que se necesitan refrescar en forma periódica los datos (o sea cargar los capacitores).

El multivibrador es un dispositivo biestable y permanece en uno de sus estados en forma indefinida hasta que sobre él actúa directamente una señal de datos. El capacitor de la *RAM dinámica*, por el otro lado, es un dispositivo cuasiestable y también tiene dos estados: cargado y descargado. Sin embargo, debido a la corriente de fuga en su estado cargado, este estado del dispositivo se deberá refrescar en forma periódica (ciento de veces por segundo) para no perder los datos almacenados. Esta función de refresco suele implicar circuitos externos que no requieren los dispositivos estáticos. La compensación es una mayor densidad en los dispositivos dinámicos debido a la sencillez de la celda de almacenamiento.

Para aplicaciones donde los factores de velocidad y reducción en la complejidad son más importantes que las consideraciones de espacio y consumo de potencia, las ***RAM estáticas*** siguen siendo la mejor opción, son más rápidas que las RAM dinámicas y no

requieren ninguna operación de refresco. Son más fáciles de diseñar, pero no pueden competir con la mayor capacidad y menor requerimiento de potencia de las ***RAM dinámicas***.

Las **RAM dinámicas** son más simples y baratas que las estáticas. Ambos tipos son volátiles, lo que significa que pueden perder su contenido cuando se desconecta la alimentación. En el lenguaje común, el término RAM es sinónimo de memoria principal, la memoria disponible para programas, por lo que este tipo de memorias son muy comunes en las computadoras y en otros dispositivos, tales como las impresoras.

2.3 CLASIFICACIÓN DE LAS MEMORIAS

Según su tecnología, las memorias se clasifican en:

BIPOLARES

Típicamente, las memorias RAM modernas se fabrican utilizando tecnología TTL (basada en transistores bipolares) o MOS (basada en transistores de efecto de campo). También existen chips RAM fabricados con tecnología BIMOS, que es un híbrido de las dos tecnologías tradicionales.

La tecnología bipolar requiere un área relativamente grande del chip para cada compuerta lógica. Puesto que necesitan muchas compuertas para construir una SRAM, es obvio que el espacio disponible se agota rápidamente.

El desarrollo de la tecnología MOS es, en gran parte, el principal responsable de los avances en el campo de la electrónica de alto nivel y, particularmente en el área de las memorias RAM. Las variantes de esta tecnología de uso más extendido son las CMOS (MOS complementaria), la NMOS(MOS de canal N) y la HMOS (MOS de alta velocidad).

La ventaja de la tecnología CMOS sobre la TTL, es que la primera disipa menos potencia y puede operar en un rango mucho más amplio de voltajes de alimentación. La tecnología NMOS se usa para producir memorias que son rápidas, disipan poca potencia y pueden albergar muchos componentes por chip. La tecnología HMOS, que es una variante de la NMOS, se utiliza, principalmente, en los microprocesadores de alta velocidad y baja potencia.

A pesar de las ventajas obvias, todos los dispositivos MOS sufren de la debilidad clave: son extremadamente sensibles a la electricidad estática (ESD).

CCD: Es una tecnología constitutiva de dispositivos de cargas acopladas, muy utilizados en memorias **[MS].**

TECNOLOGIAS ESPECIALES

Se refieren a las memorias de burbuja magnética, y otras no tan extendidas.

2.4 OPERACIÓN DE LA MEMORIA

Cada tipo de memoria es diferente en su operación interna, pero ciertos principios básicos de operación son los mismos para todos los sistemas de memoria. Las terminales comunes para todos los dispositivos de memoria son: CONEXIONES PARA LAS DIRECCIONES, CONEXIONES PARA DATOS, CONEXIONES DE SELECCIÓN y, cuando menos, una conexión de CONTROL utilizada para seleccionar una operación de *lectura o escritura*. En la figura 2.3 se muestran las líneas de conexiones:

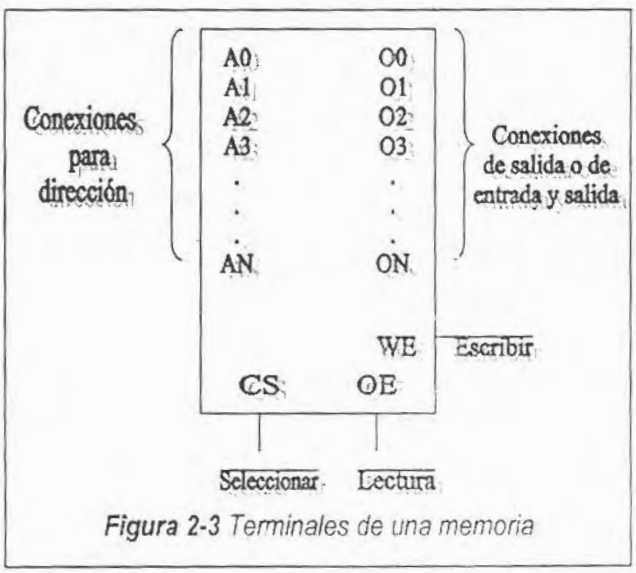


Figura 2-3 Terminales de una memoria

CONEXIONES PARA LAS DIRECCIONES:

Los dispositivos de memoria tienen entradas de dirección que seleccionan una localidad de la memoria dentro del dispositivo. El número de terminales de dirección en un dispositivo de memoria, se determina por el número de localidades de la memoria. Un dispositivo de memoria de 1K tiene 10 terminales (A_0 - A_9); así que se requieren 10 terminales

de dirección para seleccionar cualquiera de sus 1,024 localidades. Si un dispositivo de memoria tiene 11 terminales de dirección (A_0 - A_{10}) tiene 2,048 (2K) localidades internas en la memoria. Por lo tanto, el número de localidades en la memoria se puede extrapolar con el número de terminales de dirección.

CONEXIONES PARA DATOS:

Estas conexiones de datos son los puntos en los cuales se da entrada a los datos para almacenarlos o extraerlos para su lectura. Generalmente hay ocho conexiones de E/S de datos, por lo que solo se almacenan 8 bits de datos en cada localidad de memoria. En la actualidad, casi todos los dispositivos de memoria son de ocho bits de ancho, existen otros de 16 bits, de 4 bits o de apenas 1 bit de ancho.

CONEXIONES DE SELECCIÓN:

Cada dispositivo de memoria tiene una entrada, a veces más de una, que selecciona o habilita la memoria. A esta entrada se suele llamar *selección del integrado* (/CS), *habilitación de integrado* (/CE) o *selección* (/S).

CONEXIONES DE CONTROL:

La entrada de control que se encuentra con más frecuencia en una ROM es la habilitación de salida (/OE) o conexión (/G) de la compuerta, que permite el paso de datos de salida desde las terminales de salida de datos en la ROM. Si tanto (/OE) como la entrada de selección están activas, entonces se habilita la salida; si /OE está inactiva, se deshabilita la salida ya que está en alta impedancia. La conexión /OE habilita o deshabilita un conjunto de acopladores de tres estados, ubicados dentro de la memoria y que deben estar activos para leer los datos.

En una memoria RAM se tienen una o dos entradas de control. Si existe una sola, se le conoce como R/ϖ . Esta terminal selecciona una operación de lectura (R/ϖ =1), y escritura (R/ϖ = 0) sólo si la entrada de selección (/CS) esta activa. Si la RAM tiene dos entradas de control, suelen estar etiquetadas /WE (o /W) y /OE (o /G). /WE (habilitar escritura) debe estar activa para efectuar una escritura en la memoria y /OE debe estar activa para leer en la memoria. Cuando se cuenta con estos dos controles, **no** deben estar activos ambos al mismo tiempo. Si ambas entradas de control están inactivas (1 lógico), entonces no se escriben ni se leen los datos y las conexiones para datos están en su estado de alta impedancia.

Un ciclo de lectura típicamente requiere de las siguientes funciones:

1. Habilitar la memoria.
2. Seleccionar la localidad en que se leerá el dato.
3. Habilitar la operación de lectura (/OE).
4. Leer el dato de las líneas dispuestas para ello.

Mientras que un ciclo de escritura necesita las siguientes acciones:

1. Habilitar la memoria.
2. Seleccionar la localidad a escribir el dato.
3. Habilitar la operación de escritura (/W).
4. Escribir el dato en la localidad direccionada.

CAPÍTULO 3

TARJETAS EMULADORAS DE EPROMS

En la presente década, con la continua aparición de nuevos microprocesadores y microcontroladores, se está haciendo mas útil el emplear ' una tarjeta emuladora de EPROM's durante el desarrollo de aplicaciones especificas, porque con ellas podemos realizar la emulación del programa para el funcionamiento de esos sistemas. Así, la corrección o modificación del software o parte de él es más fácil.

3.1 DESCRIPCIÓN DE LAS TARJETAS EMULADORAS

La finalidad de todo emulador es reducir el tiempo de programación de la EPROM, con lo cual se reduce el tiempo invertido para la obtención del primer prototipo de alguna aplicación especifica.

Al desarrollar sistemas digitales, generalmente se requiere de un circuito integrado EPROM de cierta capacidad en la inicialización del sistema. En la programación de este dispositivos requerimos de:

- Una PC.
- El programa o archivo en formato binario o hexadecimal a ser grabado.
- Borrador de EPROMs (fuente de luz ultravioleta).
- Programador de EPROMs que puede contener una tarjeta interfaz a través del bus de expansión de la PC o usar el puerto paralelo/serie de esta.
- Y el software para manejar dicho programador

SI SE CUENTA CON TODO LO NECESARIO, EL PROCESO ES EL SIGUIENTE:

1. Configurar el programador para el tipo de EPROM a programar.
2. Cargar en la memoria de la PC el archivo que será descargado a la EPROM.
3. Borrar el Circuito Integrado EPROM en caso de que contenga datos. El tiempo promedio de este proceso varía de 20 minutos o menos. [DDPP, Pag. 577- 579] [SD, Pág. 656] [MCI, pág. 290]. Se pueden tomar las siguientes consideraciones en la variación del tiempo de borrado:
   a) Fabricante

b) Vida de la memoria
c) Cantidad de información almacenada

4. Colocar la memoria en el programador
5. Verificar si el borrado fue satisfactorio, de lo contrario se debe repetir el proceso desde el punto 3.
6. Programar la EPROM
7. Quitar la EPROM del programador y conectarlo al sistema en desarrollo o prototipo.

Una vez conectada la EPROM, si el comportamiento del sistema no es el esperado o se desea optimar o agregar más código al programa de la EPROM, se debe repetir todo el proceso.

Con todo lo anterior y el continuo manejo de la EPROM, se le puede causar algún daño al dispositivo, como doblarse o romperse alguna terminal o podría ser mal alimentado; teniendo que reemplazar el circuito integrado por otro y de esta forma se incrementa el costo del diseño.

Para el estudiante que empieza a realizar sus prácticas con memorias EPROMs esto no le es conveniente, además el proceso puede resultarle complicado; consumiendo mas tiempo del deseado en la realización. Con las tarjetas emuladoras se evita este tedioso proceso y se minimiza el tiempo invertido en el desarrollo del sistema o prototipo, porque no se necesita emplear el circuito integrado EPROM y el programador, solamente la PC y la tarjeta emuladora.

3.2 ESTRUCTURAS DE LAS TARJETAS EMULADORAS

Existen diversas estructuras para conformar una Tarjeta Emuladora de EPROMs, se tienen aquellas que se conectan a un espacio del bus de expansión de una Computadora personal (PC) maestra.

Un ejemplo de esta estructura la encontramos en el articulo presentado por Luis Esteban Curiel León con titulo: *"Tarjeta Emuladora de EPROMs"* en la revista Polibits (año IV, vol. 1, número 9). Este circuito está construido con tecnologia TTL, es totalmente dependiente de la PC y mediante un programa especial controla las funciones de la tarjeta.

Dicha tarjeta emula las funciones de una memoria de lectura y su restricción para el huésped es, tener tecnología TTL.

Mostrando en la siguiente figura su diagrama a bloques.

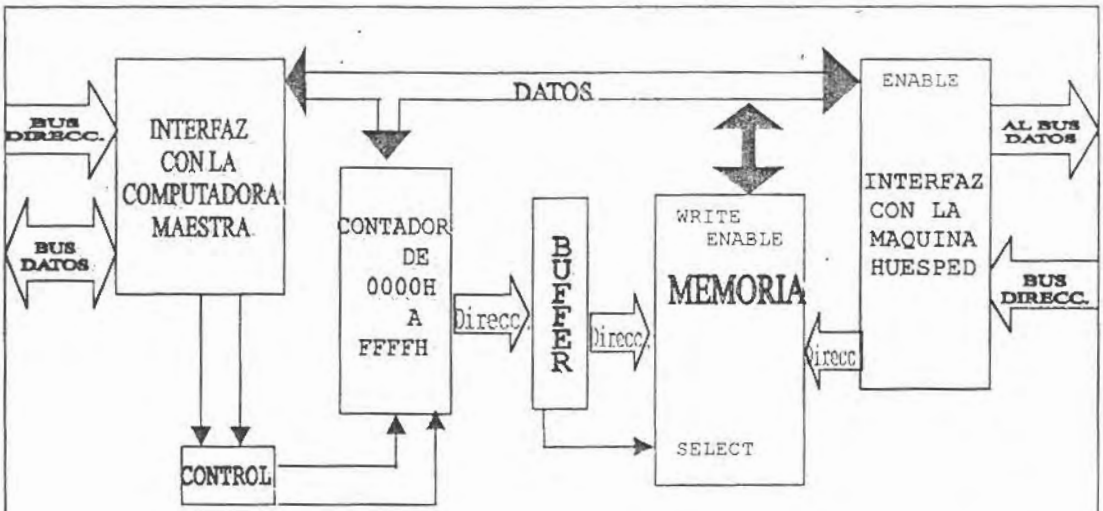


Figura 3-1 *Estructura de una tarjeta usando el Bus de expansión de la PC.*

También existen tarjetas en que la transferencia de datos se realiza mediante el puerto paralelo, al igual que la anterior utiliza tecnología TTL, contiene dos registros de direcciones, un registro de datos y un multiplexor que reduce a dos el número de líneas de datos que van hacia la PC. Diseñado por C. F. Urban en la revista ELEKTOR, mostrando a continuación su diagrama a bloques:

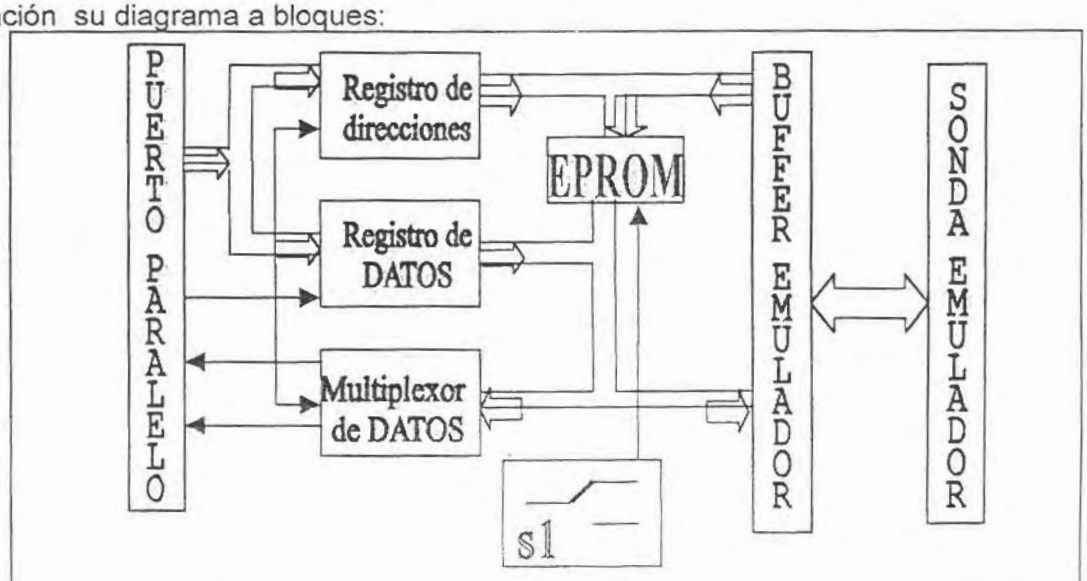


Figura 3-2 *Estructura de una tarjeta usando el puerto paralelo de la PC.*

Por ultimo tenemos las tarjetas Emuladoras que contienen un microprocesador o microcontrolador para su funcionamiento y emplean el puerto serie para la comunicación con una PC, sin necesidad de utilizar el bus de expansión de la PC. Construida con tecnología CMOS. En la figura 3-3 se expone el diagrama a bloques de una "Tarjeta Emuladora de EPROM" con el microprocesador MC68HC05E0 presentado por Peter Topping publicado en las Notas de aplicación de MOTOROLA.

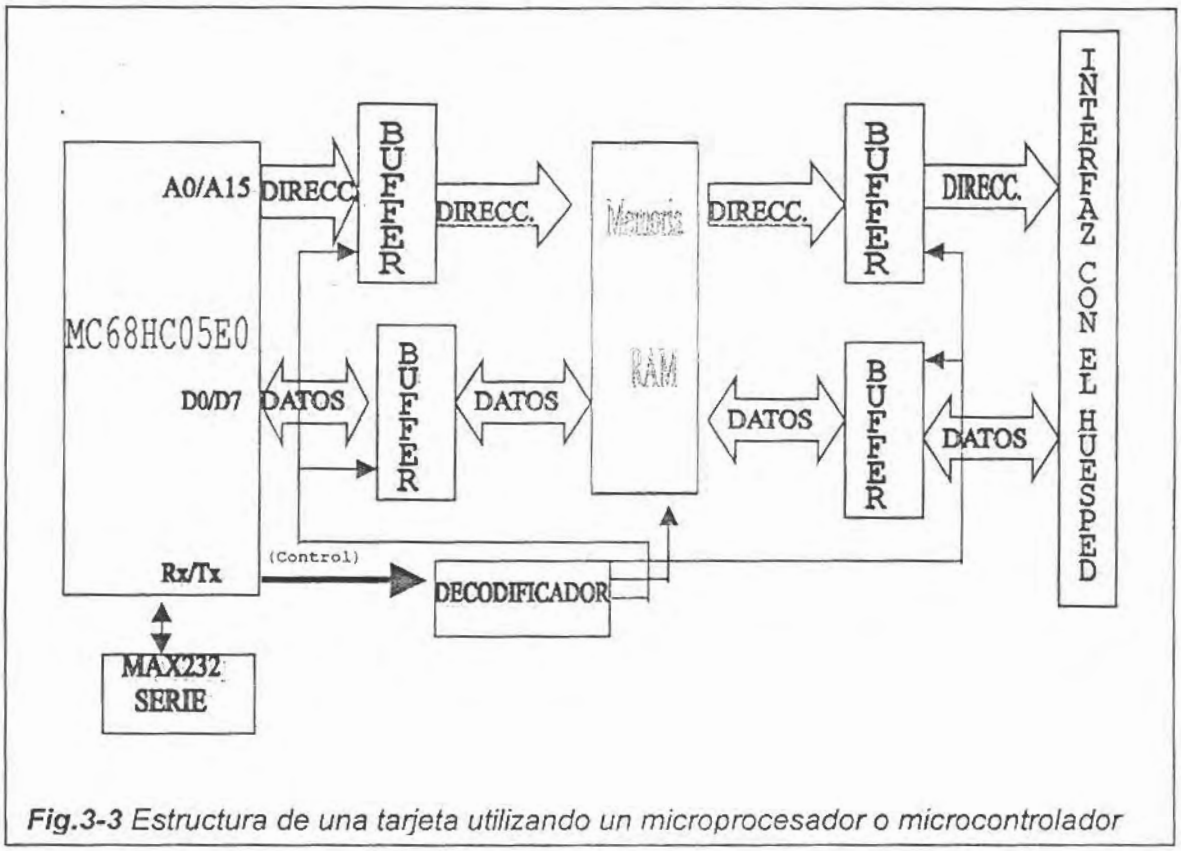


Fig.3-3 Estructura de una tarjeta utilizando un microprocesador o microcontrolador

3.3 FUNCIONES DE LAS TARJETAS EMULADORAS

- Emular memorias de diferentes capacidades
- Tener la opción de utilizarla como programadoras
- Ser de tipo universal, es decir, aplicables a cualquier sistema.
- Algunas tarjetas emuladoras de EPROMs, además, funcionan como emuladoras de algunos microcontroladores.
- La tarjeta emuladora presentada es también una tarjeta de evaluación para el microcontrolador MC68HC11E1.

3.4 VENTAJAS Y DESVENTAJAS DE LAS TARJETAS EMULADORAS

VENTAJAS

- Facilita y ahorran tiempo de diseño.
- Optimiza el programa fuente del usuario.
- Es más fácil modificar un programa.
- Fácil transporte, porque regularme son pequeñas. Así, se puede llevar del laboratorio a un cubículo u otro lugar.
- Algunas estructuras de tarjetas contienen una pila, la cual retiene la información almacenada en la memoria RAM al desconectar la fuente de alimentación.
- Existen estructuras con diversas aplicaciones, además de ser emuladoras.
- Relativamente menos costosas que un programador de este tipo de memorias.

DESVENTAJAS

- Dependiendo de su diseño, a veces dependen necesariamente de una PC.
- Si se usa el puerto paralelo, no se podrá conectar la impresora, si se tiene solo un puerto.
- Las limitaciones de las tarjetas dependen del tipo de estructura. La tarjeta presentada esta limitada para emulaciones de memorias EPROMs de 32Kb hacia abajo.

3.5 APLICACIONES

Cuando se quiere diseñar y construir sistemas digitales con un microprocesador ó microcontrolador, es indispensable que dichos sistemas consten de los siguientes módulos: procesador central, que controla a todo el sistema; dispositivos periféricos, que permitan la entrada o salida de datos; **memoria**, para almacenar datos o instrucciones útiles al sistema; el bus, o canal de comunicación de datos entre los distintos módulos. Dentro de las memorias que utilizan estos sistemas, está entre otras, la memoria EPROM.

En particular, el presente trabajo se plantea como una herramienta para el laboratorio de electrónica, para el desarrollo de prototipos y la realización de prácticas con cualquier microcontrolador, así como el aprendizaje del MC68HC11E1.

CAPÍTULO 4

DISEÑO DEL HARDWARE Y SOFTWARE

El sistema de la TARJETA EMULADORA de memorias EPROM's se basa en un microcontrolador MC68HC11E1 de MOTOROLA, en conjunto con las memorias externas RAM estática de 8 y 32 Kbytes y la memoria EPROM de 8 Kbytes. Esta tarjeta es capaz de comunicarse con una computadora personal (PC) mediante el puerto serie, y cuenta con una conexión para cable plano, cuyo extremo contrario tiene una base que simula la EPROM; la cual se conecta en el lugar destinado por el huésped.

La memoria EPROM contiene el Sistema Básico de Entrada Salida (BIOS); para este proyecto la dirección de inicio es $E000. El presente capítulo describe entre otros conceptos los registros que intervienen en el arranque (inicialización) del sistema, además de señalar las funciones que intervienen después de que el RESET se ha ejecutado; cuando esto ocurre la dirección a la que apunta es $FFFEH, en esta dirección es donde se le indica a que dirección debe saltar e iniciar nuevamente con el sistema.

4.1 CARACTERÍSTICAS GENERALES DE LA TARJETA EMULADORA DE EPROMs

Este sistema posee un microcontrolador que se encargará de realizar la aplicación del programa residente en su sección de memoria EPROM, así como de la lectura de señales externas.

Las características más importantes de la Tarjeta Emuladora son:

1. Interfaz para la comunicación serial (SCI)
2. Microcontrolador MC68HC11E1 con frecuencia de trabajo de 2 MHz
3. Decodificador de direcciones programables GAL22V10A
4. 8 líneas de datos disponibles al usuario
5. 16 líneas de direcciones disponibles al usuario
6. Puertos A, B, C, D y E
7. Reset Manual y por Software
8. Memoria EPROM de 8 Kbytes
9. Memoria RAM estática de 8 Kbytes
10. Memoria RAM estática de 32 Kbytes

11. Interfaz para emular una memoria EPROM de 2Kb, 4Kb, 8Kb, 16Kb y 32Kbytes
12. Pila de 3.6 volts
13. Alimentación con 5 volts regulados ó de 7 a 35 volts.

4.2 DIAGRAMA A BLOQUES

A continuación se presenta el diagrama a Bloques de la Tarjeta Emuladora de EPROMs

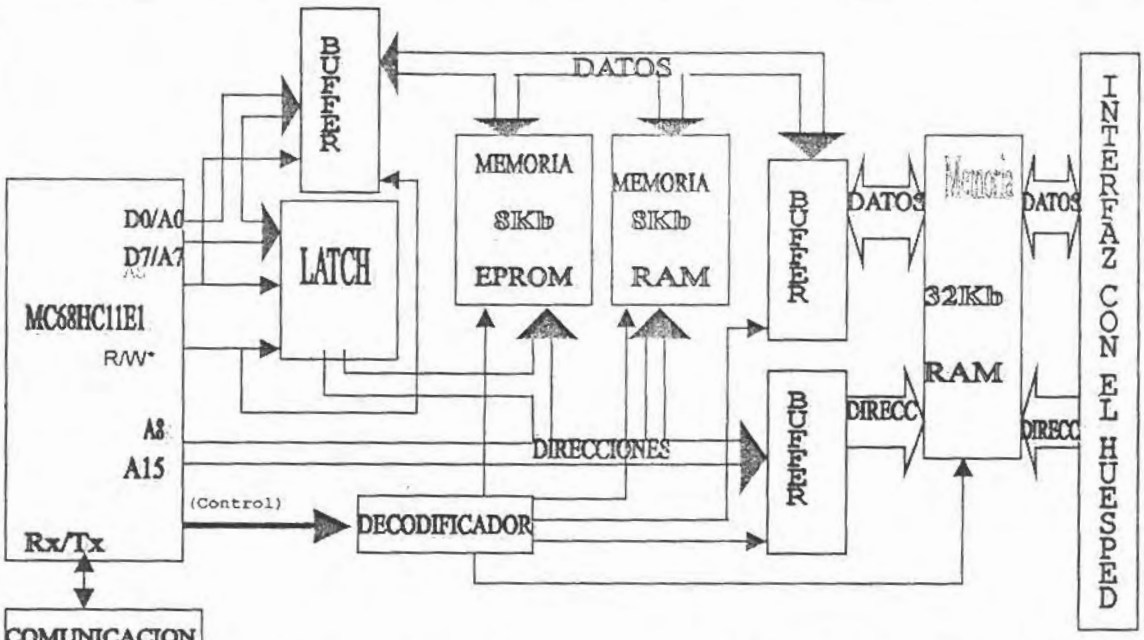


Figura 4-1 Diagrama a Bloques de la Tarjeta Emuladora de EPROM's

4.3 EL CIRCUITO MC68HC11

El corazón de la tarjeta **EMULADORA DE EPROMs**, es el microcontrolador **MC68HC11E1.** El método de construcción del circuito MC68HC11 emplea tecnología de semiconductor de óxido de metal complementario de alta densidad (***high-density complementary metal-oxide semiconductor***, -HCMOS-). Debido a esta particularidad, el circuito, es rápido, pequeño, consume poca potencia y tiene una alta tolerancia a señales ruidosas. Estas características son muy importantes para el diseño y construcción de cualquier prototipo. Existen distintas versiones del microcontrolador MC68HC11, las variantes básicas incluyen distintos tamaños y tipos de memoria interna **[MHC11, pág. 1-5].**

El diseño de esta tarjeta se basa especificamente en el circuito MC68HC11E1 el cual, a consideración de la compañía Motorola, representa de manera adecuada a la familia de unidades microcontroladoras (MCU's).

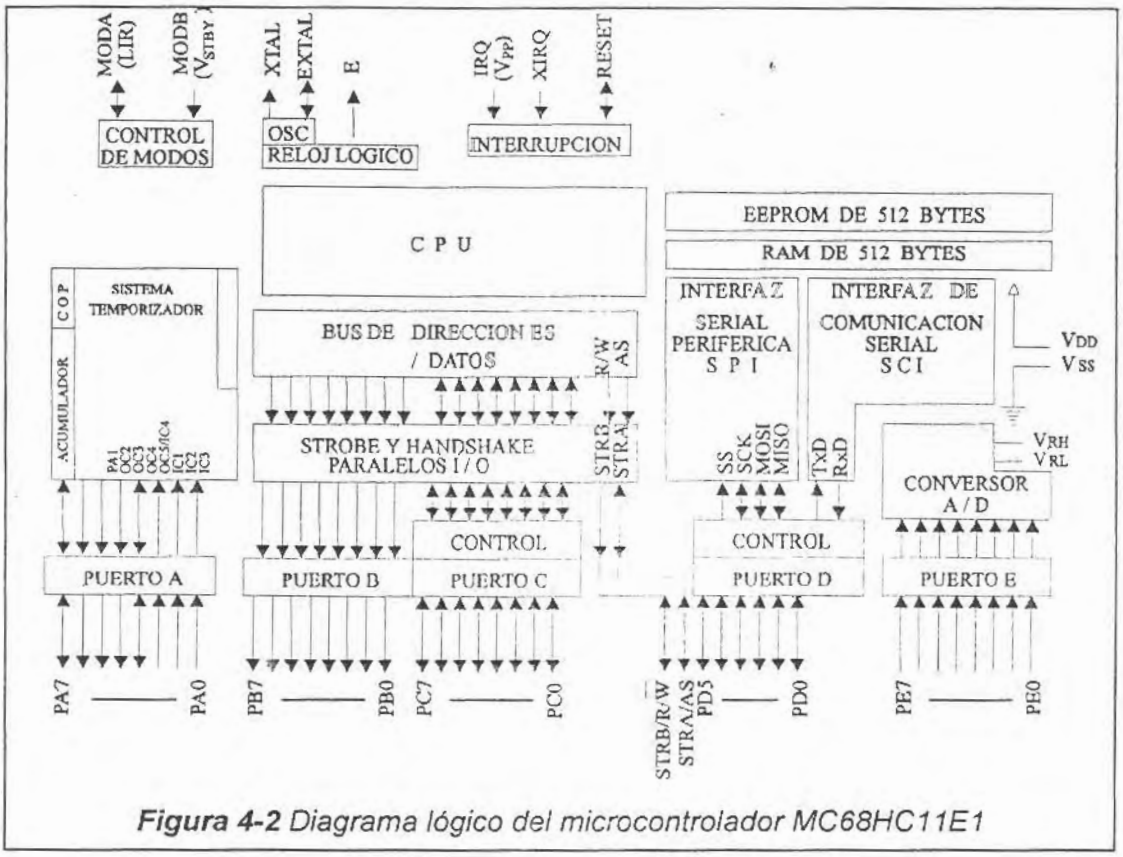


Figura 4-2 *Diagrama lógico del microcontrolador MC68HC11E1*

El 68HC11E1 de la familia Motorola, es un microcontrolador de 8 bits en su bus de datos, 16 bits en su bus de direcciones, por lo que puede direccionar hasta 65536 bytes (64 Kbytes); con un conjunto de instrucciones que es similar a los más antiguos miembros de la familia 68xx (6801, 6805, 6809). Dependiendo del modelo, el 68HC11 tiene internamente los siguientes dispositivos: una unidad central de proceso de 8 bits, EEPROM o OTPROM, RAM, Entrada/Salida digital, temporizadores (timers), generador PWM, Convertidor A/D y canales de comunicación sincrónica y asincrónica (SPI y SCI). La corriente típica que maneja es menor que 10mA.

Típicamente el bus de datos y direcciones están multiplexados. El temporizador se basa en un único contador de 16 bits y con un preescalador programable para bajarlo si es requerido. Cuenta con un convertidor A-D que es típicamente de 8 canales y 8 bits de

resolución. La comunicación serie asíncrona SCI, tiene un formato de datos de 1 bit inicio, 8 o 9 bits de datos, y un bit de parada.

Las versiones del MC68HC11 con EEPROM incluyen una "Bomba de carga" en el circuito; el cual proporciona el alto voltaje necesario para el procedimiento de programación directa. Esta memoria no volátil se emplea para el almacenamiento de información de calibración, diagnóstico, registro de datos y seguridad.

4.3.1 MODOS DE OPERACIÓN

El circuito MC68HC11 tiene cuatro modos de funcionamiento: como un solo circuito integrado (***single-chip operating mode***), multiplexado expandido (***expanded multiplexed operating mode***), especial para cargado de programas (***special bootstrap operating mode***) y modo para pruebas (***special test operating mode***). De acuerdo con el modo en que se configure el circuito, podrá emplear de manera distinta sus recursos internos. Los dos últimos modos se disponen principalmente para realizar pruebas en la fábrica.

Modo Mono-Integrado (***Single-Chip***). En este caso el circuito se comporta como un microcontrolador monolítico, sin buses externos de direcciones ó datos. Se pueden emplear todos los elementos periféricos en su totalidad. La cantidad de información contenida y procesada está limitada a la capacidad natural de memoria interna que posea.

Modo Multiplexado Expandido. Bajo este modo el microcontrolador puede accesar un espacio de memoria de hasta 64 Kilobytes. Este espacio incluye - a discreción del programador - la memoria interna. Las líneas de direcciones y datos requeridas para accesar la memoria externa se configuran a partir de terminales que correspondían a periféricos designados a otras labores. Las líneas de datos y las 8 líneas bajas del bus de direcciones se encuentran multiplexadas, por lo que es necesario emplear un registro externo para su demultiplexación.

En la siguiente tabla se muestra la manera de como seleccionar el modo de operación deseado, a partir de las terminales MODA y MODB (observe la figura 4.2) antes de aplicar una señal de inicialización (*reset*).

MODB	MODA	Modo seleccionado
1	0	Un solo circuito integrado
1	1	Multiplexado expandido
0	0	Especial para cargado de programas
0	1	Modo para pruebas

Tabla 4.1. *Selección de Modos*

Como se puede observar, la terminal MODA selecciona entre los modos **Mono-Integrado** (***Single-Chip)*** y **Multiplexado Expandido**; y MODB selecciona entre la variación normal y la variación especial del modo operativo escogido.

Debido a que el sistema de la tarjeta **Emuladora de EPROMs** contiene una EPROM y dos RAMs externas se utiliza al microcontrolador en modo **Multiplexado Expandido** (figura apéndice A).

El microcontrolador MC68HC11 es un dispositivo HCMOS por lo que las líneas que no se utilizan es conveniente conectarlas a un nivel lógico alto; con la finalidad de evitar oscilaciones en las entradas, además de evitar corriente de alimentación excesiva que dañaría al microcontrolador.

4.3.2 MEMORIA INTERNA

En el modo de un solo circuito integrado no se tienen direcciones externas. En el modo de multiplexado expandido las localidades de memoria son básicamente las mismas que en el modo de un solo circuito integrado, sin embargo las localidades entre las áreas sombreadas y las marcadas para ROM son para direcciones de memorias externas y para dispositivos de entrada/salida (ver figura 4.3).

Existen 64 registros que permiten efectuar el mapeo de la memoria RAM, EPROM o EEPROM interna del microcontrolador, el control de los dispositivos periféricos y en general el control de la operación del microcontrolador. Estos registros pueden ser relocalizados en espacios de memoria en bloques de 4 Kbytes **[MHC11, pág. 3-6].** Los 512 bytes de

memoria RAM pueden ser localizados durante la inicialización cuando se está escribiendo en el registro INIT. **[MCT, págs. 201, 202, 382].**

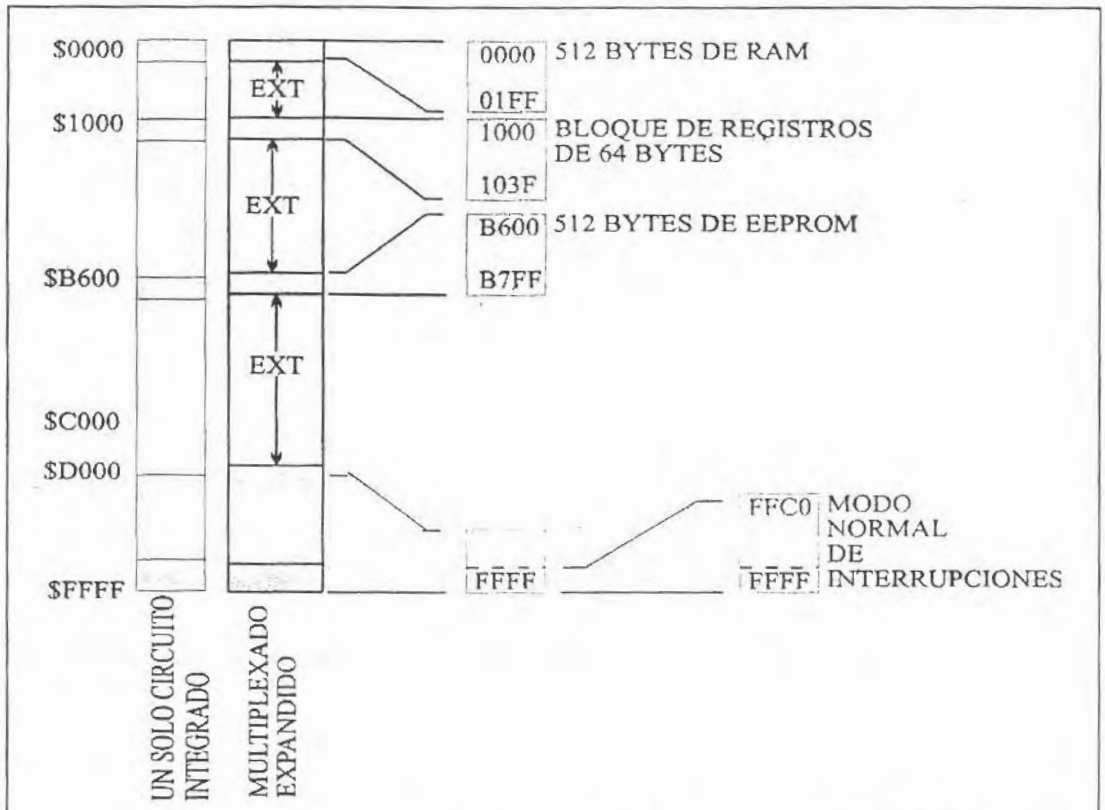


***Figura 4.3.** - Modelo del Mapa de Memoria del Microcontrolador MC68HC11*

4.4 ORGANIZACIÓN DE LA MEMORIA

Como el microcontrolador se opera en **Modo Multiplexado Expandido**, el bloque de memoria de la **Tarjeta Emuladora de EPROMs** esta compuesta por un EPROM y dos RAMs externas.

El circuito integrado AMD27C64 correspondiente a la memoria **EPROM** que es el BIOS del sistema, es decir contiene el programa que sirve como Sistema Operativo ó Monitor; el cual tiene la finalidad de inicializar los diferentes periféricos y esta comprendida entre las direcciones $E000H hasta $FFFFH correspondiente a 8 Kbytes. El circuito integrado MS6264L de MOSEL, que corresponde a la memoria **RAM** ubicada entre las direcciones $C000H hasta $DFFFH, con un total de 8 Kbytes; nos sirve para almacenar datos o programas objeto del usuario. También nos permite guardar valores de parámetros y variables intermedias, los cuales pueden variar a lo largo de la ejecución de un programa, y

por ultimo, una **RAM** de 32 Kbytes comprendida entre las direcciones $4000H hasta $BFFF, que sirve para guardar el programa del huésped o sistema mínimo.

En la siguiente figura se representa el mapa de memoria que muestra las direcciones bases a que corresponde cada uno de los elementos fundamentales del sistema.

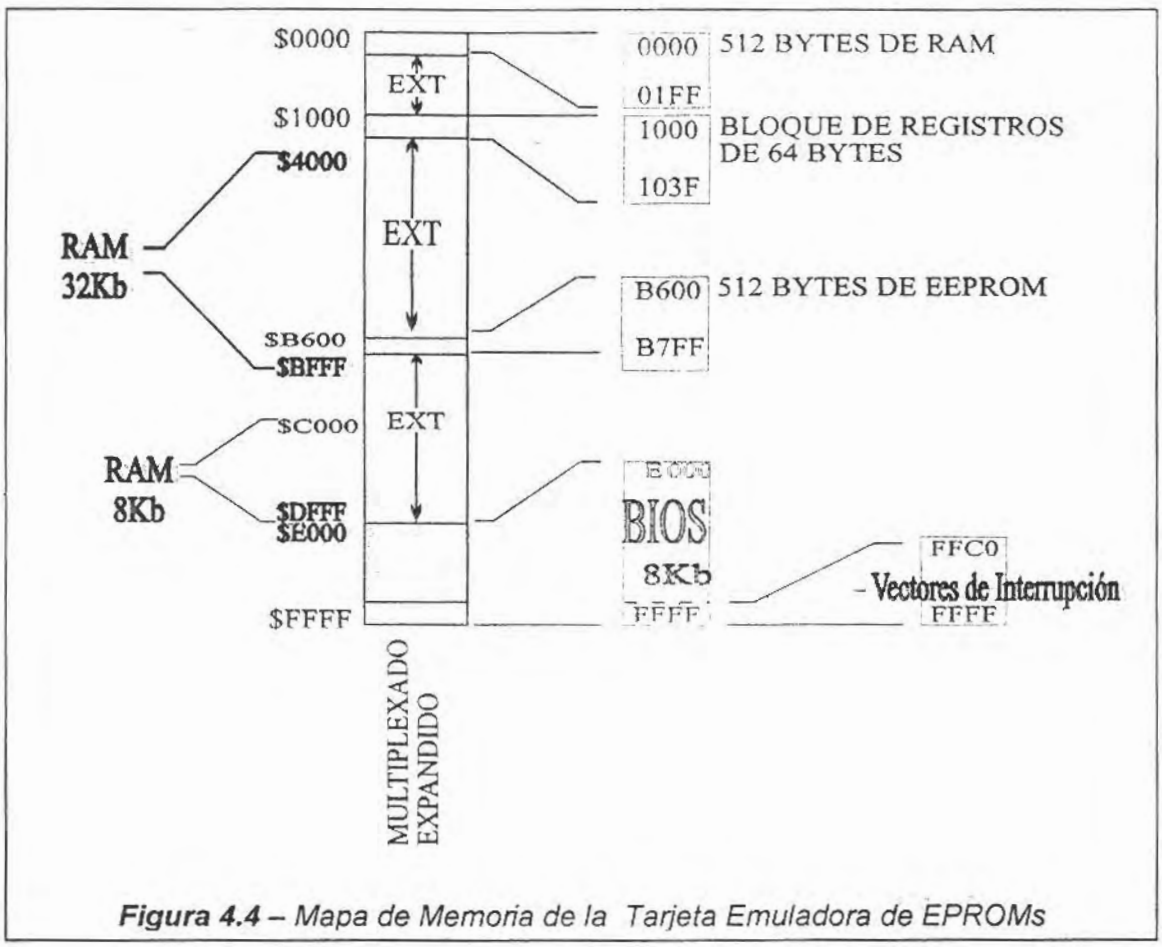


***Figura 4.4** – Mapa de Memoria de la Tarjeta Emuladora de EPROMs*

4.5 BUS DE CONTROL

Estas líneas tienen como finalidad sincronizar la Unidad Central de Procesamiento con los demás dispositivos que se encuentran en el sistema. Entre las líneas de control se tiene: el reloj (E), la señal de lectura/escritura, las interrupciones, etc. Además se consideran señales de control las generadas por el decodificador de direcciones.

SEÑAL DE RELOJ E (*ENABLE CLOCK*)

Esta terminal es el correspondiente al control del tiempo de referencia del microcontrolador. La principal función de la señal de reloj E es sincronizar el funcionamiento

de todos los dispositivos en el sistema, es decir se usa junto con otras señales para habilitar dispositivos externos para manejar datos dentro del CPU durante la segunda mitad de un ciclo de lectura.

La frecuencia de esta señal es de 2 MHz, debido a que es la cuarta parte de la frecuencia de trabajo del cristal externo. Para fines de este sistema se utiliza un cristal de 8 MHz.

SEÑAL DE CONTROL STRB (*STROBE B*) / R/W* (*READ/ WRITE*)

La terminal STRB / (R/W*) esta asociada con el puerto B y su comportamiento depende del modo en que el 68HC11 se encuentra funcionando. STRB es una linea de "strobe" o de validación de datos, que se utiliza durante la comunicación entre controladores a través de los puertos paralelo en el modo sencillo (Mono-Integrado).

Esta terminal **R/W*** es una línea de salida usada para controlar la dirección de transferencia en el bus de datos externo del microcontrolador en el modo de operación de multiplexado expandido; es decir, cuando se encuentra en el modo multiplexado expandido R/W*[1] [2] es la terminal que selecciona la lectura o escritura e indica la dirección del flujo de datos. Se debe tener un nivel lógico alto para los ciclos de lectura y un nivel lógico bajo para los ciclos de escritura.

Esta señal en combinación con la señal de reloj controla la lectura y escritura en las memorias

SEÑAL DE CONTROL AS (*ADDRESS STROBE*) / STRA (*STROBE A*)

En modo sencillo o circuito único, STRA sirve también como una línea de "strobe" al igual que STROB. En modo expandido, esta terminal actúa como una señal de control; es usada para habilitar el registro externo (latch octal), mediante el cual se demultiplexan las direcciones de orden más bajo del puerto C, dejándolas pasar mientras ocurra un nivel lógico alto, la información es atrapada durante un nivel lógico bajo.

SEÑAL RESET*

La terminal denominada **RESET*** (*Reset*) es una señal de control bidireccional activa en bajo usada como entrada para inicializar el funcionamiento del microcontrolador. El

[1] ' * ' significa negación

RESET manual se activa al presionar un botón generando un nivel lógico bajo (Observe S1 del diagrama en el apéndice A2). Como salida, indica que se ha detectado un defecto interno durante el funcionamiento, mediante un "perro guardián" (watchdogs) disponibles.

Para mejor comprensión se presenta el diagrama de tiempo de las señales de control.

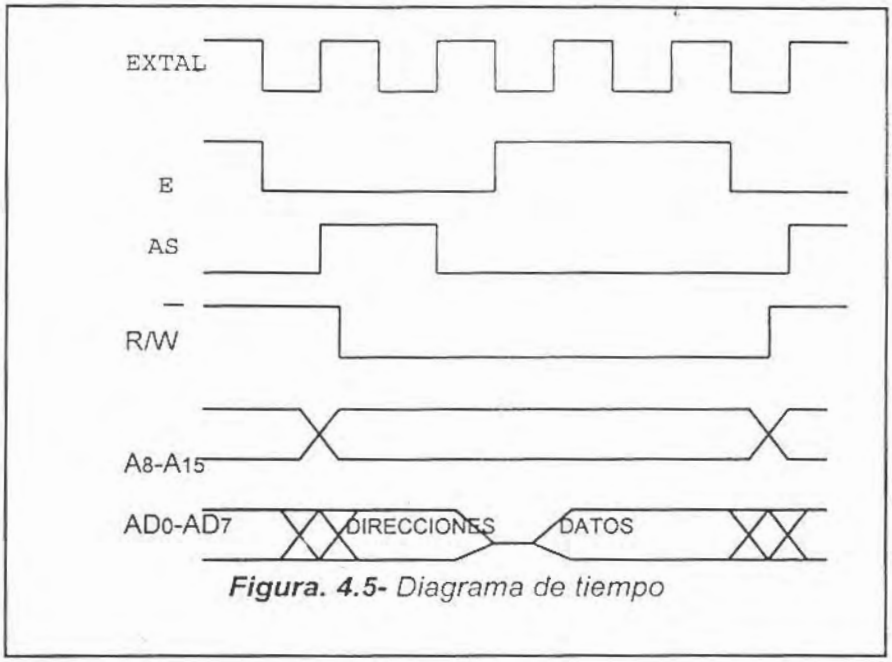


Figura. 4.5- *Diagrama de tiempo*

Otras señales de control son las señales codificadas por la GAL22V10A, para seleccionar los diferentes dispositivos de memoria.

4.6 BUS DE DIRECCIONES

El microcontrolador MC68HC11 proporciona un bus de 16 lineas de direcciones las cuales están disponibles en el puerto B y C. Estas lineas se usan para accesar a cualquier localidad de los 64 Kbytes del mapa de memoria.

El comportamiento de las ocho terminales del puerto B depende del modo en que el 68HC11 está operando:

PB[0] - PB[7] en modo sencillo (modo de único chip) son ENTRADAS/SALIDAS de propósito general.

PB[0] - PB[7] modo multiplexado extendido, el Puerto B es usado como el byte de orden más alto del *bus de direcciones*.

El Puerto C puede funcionar en una variedad de modos. En el modo sencillo (modo **single-chip**) cada terminal es capaz de ser una entrada o una salida. Cuando opera en el modo multiplexado extendido el Puerto C funciona como un *bus multiplexado de direcciones/datos* y corresponden a los 8 bits menos significativos.

Como los datos y las direcciones de orden más bajo están multiplexados en el tiempo, se emplea un circuito "*latch*" 74HC573, para separar las direcciones de los datos.

Por consiguiente durante la **primera mitad** de cada *ciclo de bus*, se obtiene señales de direcciones (A0 – A7) y durante la **segunda mitad** de cada *ciclo de bus*, el bus se convierte en un bus bidireccional de datos (D0 – D7), véase la figura 4.5.

4.7 BUS DE DATOS

El bus de datos contiene 8 líneas bidireccionables, que se encargan de proporcionar la comunicación del CPU hacia los diferentes dispositivos del sistema. Las señales de datos (D0 - D7) se comunican en paralelo con los dispositivos correspondientes del sistema.

El bus de datos se conecta directamente al "*buffer*" 74HC245 con la finalidad de mantener la corriente necesaria de los dispositivos a los que llegaran los datos (como el EPROM, RAM, etc.).

También se utiliza el *buffer* para controlar la dirección de los datos, es decir si salen o entran al microcontrolador, a las memorias o al huésped, esto dependerá de la habilitación, la señal de control que se le aplique al dispositivo correspondiente, las cuales son generadas y controladas por la GAL22V10A.

4.8 ALIMENTACIÓN

La energía es suministrada al microcontrolador por las terminales VDD y VSS donde VDD es la entrada de voltaje positivo y VSS es tierra. El sistema trabaja con 5 volts regulados ó de 7 a 35 volts.

Se alimenta a la memoria RAM de 32 Kbytes de la tarjeta Emuladora con una pila de respaldo para retener la información sin necesidad de la fuente de alimentación.

4.9 COMUNICACIÓN SERIAL

Este sistema de comunicación serie (*Serial Communication System - SCI -*) se usa para realizar la comunicación entre una computadora personal (PC) y la tarjeta *EMULADORA DE EPROMs*, El subsistema emplea el formato estándar NRZ (***Non Return to Zero, no retorno a cero***) que incluye un bit de arranque, datos de ocho. ó nueve bits, y un bit de alto (o parada) y permite varias selecciones de velocidad de comunicación. En el sistema SCI se transmite y se recibe información de manera separada, teniendo implementadas estas funciones independientes, sin embargo, emplean algunos de los parámetros de la comunicación en forma común. Se puede mencionar que la arquitectura de este sistema incluye:

- Un formato de comunicación estándar NRZ (***Mark/space***)
- Método de detección de error que incluye detección de ruido ó interferencia
- Operación ***full-duplex***
- Programación en software para distintas velocidades de transmisión / recepción
- Selección de longitud de palabra por software (palabras de ocho ó nueve bits)
- Habilitación para la transmisión y recepción de bits por separado

Este sistema se encuentra trabajando con: 1 bit de inicio, 8 bits de datos y 1 bit de parada. El transmisor y el receptor del SCI son funcionalmente independientes, pero usan el mismo formato de datos y velocidad.

La interface entre el MC68HC11 y la computadora personal a través del puerto serie se realiza mediante el dispositivo MAX232N y capacitores de 10 μF, para efectuar el cambio de los niveles lógicos del RS-232 que es típicamente de ±12 volts a niveles lógicos utilizables por el microcontrolador (0 a 5 volts).

Para realizar la comunicación del sistema con la PC, se utiliza un cable en cuyos extremos tiene conectores tipo DB-9 hembra y macho. La siguiente tabla muestra las terminales y las señales del DB-9.

Señal	Descripción	DB-9
RxD	Recepción de Datos	2
TxD	Transmisión de Datos	3
DCD	Detector de Acarreo	1
DSR	Datos preparados	6
DTR	Terminal Datos Preparados	4
RTS	Preparado para Enviar	7
CTS	Línea de Control	8
GND	Tierra	5

Tabla 4.2- *Terminales de conexión entre la Tarjeta Emuladora y la PC*

4.10 DECODIFICADOR DE DIRECCIONES

El decodificador de direcciones esta constituido por el dispositivo GAL22V10A, el cual a partir de las señales de entrada que son: las líneas de direcciones A13, A14 y A15, así como las líneas de control AS, E, R/W* provenientes del microcontrolador; se realizan las ecuaciones necesarias para la habilitación de cada elemento del sistema.

Apreciando en la siguiente tabla las ecuaciones de decodificación:

SALIDAS	ENTRADAS
/CS_1 =	A15 * A14 * A13 * E
/CS_2 =	A15 * A14 * /A13 * E
/CS_3 =	A14 * /A15 * E + /A14 * A15 * E
/CS_4 =	/AS * E
/CS_5 =	A14 * /A15 * E + /A14 * A15 * E
/RD =	RW * E
/WR =	/ RW * E

TABLA 4.3. *- Ecuaciones de codificación.* **Nota:** "/" significa negación y "*" es igual a AND

4.11 DESCRIPCIÓN DEL SOFTWARE

El BIOS de este sistema inicia en la dirección $E000H hasta $FFFF, ocupando los 8 Kbytes de memoria EPROM externa. Contiene el programa de arranque y las funciones que realiza después de que el RESET se ha ejecutado.

4.11.1 COMUNICACIÓN CON LA PC

Para poder establecer la comunicación del sistema con la computadora personal a través del puerto serie, es necesario tener un programa de *comunicaciones*. Este programa de comunicaciones puede ser cualquiera, teniendo y configurando los siguientes parámetros:

a) Velocidad de Transmisión 9600 bauds
b) 8 bits de datos, 1 bit de inicio, 1 bit de paro y sin paridad.

Si por algún motivo se cambia la velocidad en el programa de inicialización (de la tarjeta) también se deberá modificar la velocidad de transmisión del programa de comunicaciones.

4.11.2 INICIALIZACIÓN

El BIOS del sistema se encarga de inicializar la tabla de vectores de interrupción, registros, comunicación serie, etc.

Para llevar a cabo la interface de comunicación serial (SCI) del sistema con una computadora personal, se ocupan las terminales PD0/RxD y PD1/TxD del puerto D, como líneas de Recepción y Transmisión respectivamente, las cuales son configuradas y controladas por los registros de control: BAUD, SCCR1, SCCR2, SCSR y SCDR.

En este sistema se opera a 9600 Bauds por segundo, 8 bits de datos, 1 bit de inicio, 1 bit de parada y sin paridad, por lo que se lleva a cabo el siguiente procedimiento de *configuración*

1-- Seleccionar la razón bauds para la transmisión. Escribir la selección en el registro BAUD (SCP0-1, SCR0-2).

2- Seleccionar la longitud de la palabra y la forma de despertado del receptor. Escribir al registro SCCR1 (M, WAKE).

3- Habilitar interrupciones de transmisión y recepción, y escribirlo en el registro SCCR2 (TIE, TCIE, RIE, ILIE, TE, RE, RWU).

A continuación se escribe la función principal de cada registro, posteriormente se presenta el protocolo de inicialización, finalizando con el diagrama de flujo.

El ***registro de control baud*** (BAUD), que se encuentra en la dirección 102BH, es usado para seleccionar la velocidad (bauds) para operaciones SCI; contiene 2 bits de control para prueba de fábrica y 5 bits para seleccionar el baud. La selección del baud se hace a partir de una frecuencia dada por el cristal con el que sé esta trabajando. Cabe aclarar que el microcontrolador MC68HC11 divide entre 16 la frecuencia de reloj o entre 64 la frecuencia del cristal.

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
TCLR		SCP1	SCP0	RCKB	SCR2	SCR1	SCR0

SCP1 y **SCP0**.- Selectores del Preescalador.- *Seleccionan la velocidad de transmisión serial en bauds.*

Como el sistema (tarjeta Emuladora) utiliza un cristal externo de 8 MHz, entonces la frecuencia de trabajo (E) es de 2 MHz; la selección de velocidad se obtiene colocando:

SCP1	SCP0	Frecuencia del cristal: 8 MHz
1	1	9600 BAUD
		2 MHz (reloj E)

SCR2 - SCR1 - SCR0 - *SCI Baud Rate Selects,* Selectores de la Velocidad de Transmisión.

Seleccionan la velocidad de transmisión y recepción de información. El ajuste de los bits del preescalador **(SCP1** y **SCP0)** determina la velocidad de transmisión mayor que se puede lograr. **[MHC11, páginas 9-5 a 9-8]**

SCR2	SCR1	SCR0	Tasa de Velocidad de Transmisión
0	0	0	9600 bauds

El primer ***registro de control de comunicación serial*** (SCCR1), el cual se puede accesar en la dirección 102CH, determina la longitud de palabra y el método para el despertado del receptor. Contiene 3 bits asociado con el formato opcional de 9 bits de datos y otro bit (WAKE) usado para seleccionar uno de los dos métodos para despertar al receptor.

El segundo ***registro de control de comunicación serial*** (SCCR2), que se localiza en la dirección 102DH, contiene los controles principales del SCI, debido a que contiene los bits de control para habilitar o deshabilitar las funciones individuales de la Recepción y Transmisión **[MHC11, pág. 9-9].**

El ***registro de estado de la comunicación serial*** (SCSR), localizado en la dirección 102EH. Contiene 2 banderas de estado de transmisión y 5 banderas de estado relacionadas con el receptor. La transmisión genera banderas para TDRE y TC. La recepción genera las banderas para RDRF, OR, IDLE, NF(bandera de ruido) y un indicador de error (FE) **[MHC11, pág. 9-9].**

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
TDRE	TC	RDRF	IDLE	OR	NF	FE	

TDRE - Bandera que indica que el Registro de Transmisión de Datos está Vacío.

Este bit está ajustado a "1" si ya se puede escribir otro dato al registro **SCDR** para continuar la transmisión de caracteres. Si es igual a "0", entonces el registro de transmisión de datos está ocupado. Se ajusta a "0" con la lectura del registro **SCSR** seguido de una escritura al registro **SCDR**.

TC - Bandera que indica que se terminó la transmisión de un carácter.

Se ajusta a "1" si el circuito transmisor está vacío. Se pone a "0" con la lectura del registro **SCSR** seguido de la escritura del registro **SCDR**.

RDRF - Bandera que indica que el Registro de Datos del receptor está Lleno.

Se ajusta a un valor de "1" si el carácter recibido está listo para ser leído del registro

SCDR. Este bit se pone a "0" con la lectura del registro SCSR seguido de la lectura del registro SCDR.

IDLE - Bandera de la Detección de Línea Vacía.

El ***registro de datos de la comunicación serial*** (SCDR) localizado en la dirección 102FH, es de dos registros separados, TDR y RDR. TDR es un registro de paso de datos de transmisión de solo escritura y RDR es un registro de paso de datos de recepción de solo lectura.

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0

Si en la programación se lee el registro SCDR, se esta accesando al registro RDR, o si en la programación se escribe al registro SCDR sé accesa al registro TDR

De lo expuesto anteriormente se presenta a continuación el código mnemónico necesario para inicializar la comunicación serial.

```
REGST   EQU  $1000
BAUD    EQU  $REGST+$2B
SCCR1   EQU  $REGST+$2C
SCCR2   EQU  $REGST+$2D
SCSR    EQU  $REGST+$2E
SCDR    EQU  $REGST+$2F

INISCI  EQU  *
        LDDA  #$30
        STAA  BAUD      *9600 bps
        LDDA  #$00
        STAB  SCCR1     *1bit inicio, 8bits datos, 1bit paro, sin paridad
        LDDA  #$0C
        STAA  SCCR2     *Habilita TDR y RDR
        RTS
```

4.12 COMANDOS

Es necesario un programa de comunicaciones para poder establecer la comunicación a través del puerto serie de la computadora y nuestro sistema. Ejecutando el programa de comunicaciones se estará listo para enviar y/o recibir datos desde/hacia el sistema.

Los comandos tecleados en código ASCII desde la 'PC, son enviados vía puerto serial, para ser analizados y buscados en el programa del BIOS, si es encontrado se ejecuta la subrutina. En el apéndice D se anexa el listado del programa.

El menú presenta las siguientes opciones

LIMPIA.- Limpia la memoria RAM de 32 Kbytes

LOAD [CAPACIDAD DE LA MEMORIA] Carga un archivo .s19 a memoria RAM de 32Kbytes

LOAD [DIRECCIÓN C000] Carga archivo .s19 a memoria RAM de 8 Kbytes

CARGA [CAPACIDAD DE LA MEMORIA]. Carga un archivo .HEX a memoria RAM de 32Kbytes

LLENA <DIR INICIO> <DIR FINAL> [DATO]. Llena la memoria RAM con un dato

MUESTRA <DIR INICIO> <DIR FINAL>. Muestra el contenido de la memoria

CONTROL W (CTL W) Detiene la ejecución y continua con cualquier tecla

CONTROL X (CTL X) Aborta la ejecución que se encuentre realizando

RETORNO DE CARRO (CR). Repite el último comando

[?] Presenta el menú

- Después de la presentación del menú aparece el prompt indicando la espera de algún comando desde la PC.

- Analiza el comando introducido desde la PC. La finalidad de esta rutina es de eliminar los espacios antes de encontrar el primer carácter, convertirlo a mayúscula y terminar cuando se encuentre el retorno de carro.

- Se busca en la tabla de comandos. El comando introducido se compara con la lista de comandos hasta que sea igual, en caso contrario termina y envía un mensaje de error.

- Si el comando es encontrado, se ejecuta la subrutina correspondiente a ese comando y cuando se termina la operación regresa el control al programa principal y espera un nuevo comando.

4.13 SUBSISTEMAS UTILIZADOS

Algunos de los subsistemas o subrutinas empleadas para la elaboración del programa del BIOS de la Tarjeta Emuladora de EPROMs, fueron guiadas por el programa monitor *buffalo* de Motorola.

Estas subrutinas son: por ejemplo, para leer o escribir un dato en el sistema SCI, o convertir un carácter ASCII a un número hexadecimal. A continuación se presentan algunas de las subrutinas con su respectiva descripción.

INIVECT: Inicializa la tabla de vectores.

INISCI: Inicializa la interface serie de comunicaciones (SCI) a 9600 bauds por segundo, 1 bit de inicio, 8 bits de datos, 1 bit de parada y sin paridad.

ENVSCI: Envía el carácter ASCII contenido en el acumulador A hacia SCI.

ENVMSG: Envía una cadena de caracteres ASCII hacia SCI, hasta encontrar el fin de texto ($04).

LEECARNB: Lee desde SCI hasta encontrar un carácter distinto de espacio en blanco, coma o tabulador.

CHECABRT: Verifica si existe un aborto (CTL X) o pausa (CTL W) desde SCI.

LEESCI: Lee un carácter desde SCI, si el carácter es recibido, lo regresa en el acumulador A, en caso contrario regresa un cero en el acumulador A.

MINMAY: Convierte a mayúscula el carácter contenido en el acumulador A.

HEXBIN: Convierte el carácter ASCII del acumulador en Binario

RETARDO: Envía un retardo de 7.6 milisegundos para la descarga de S-records.

ENVCRLF: Envía salto de línea y retorno de carro.

4.14 DIAGRAMA DE FLUJO DE LA TARJETA EMULADORA

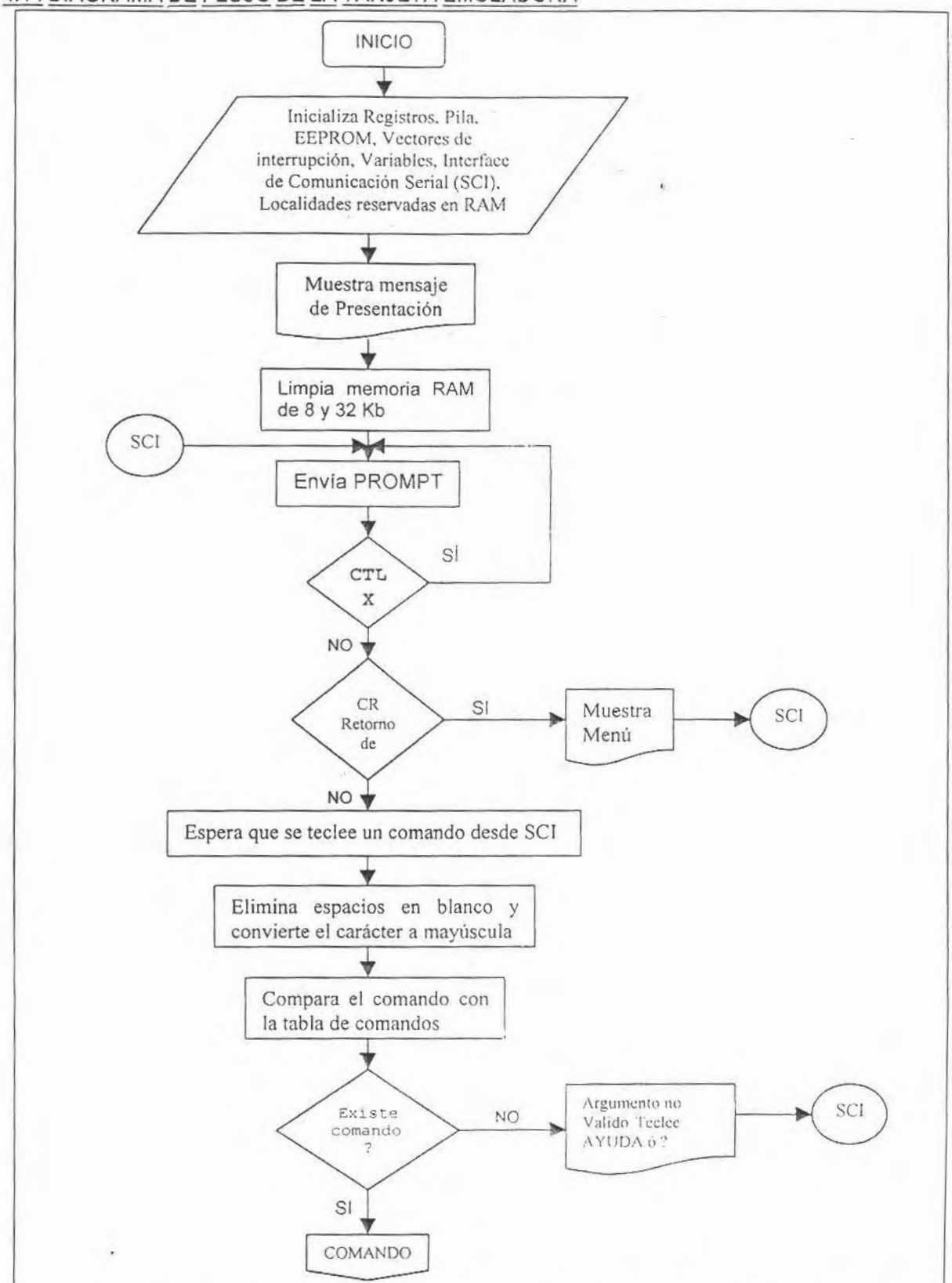
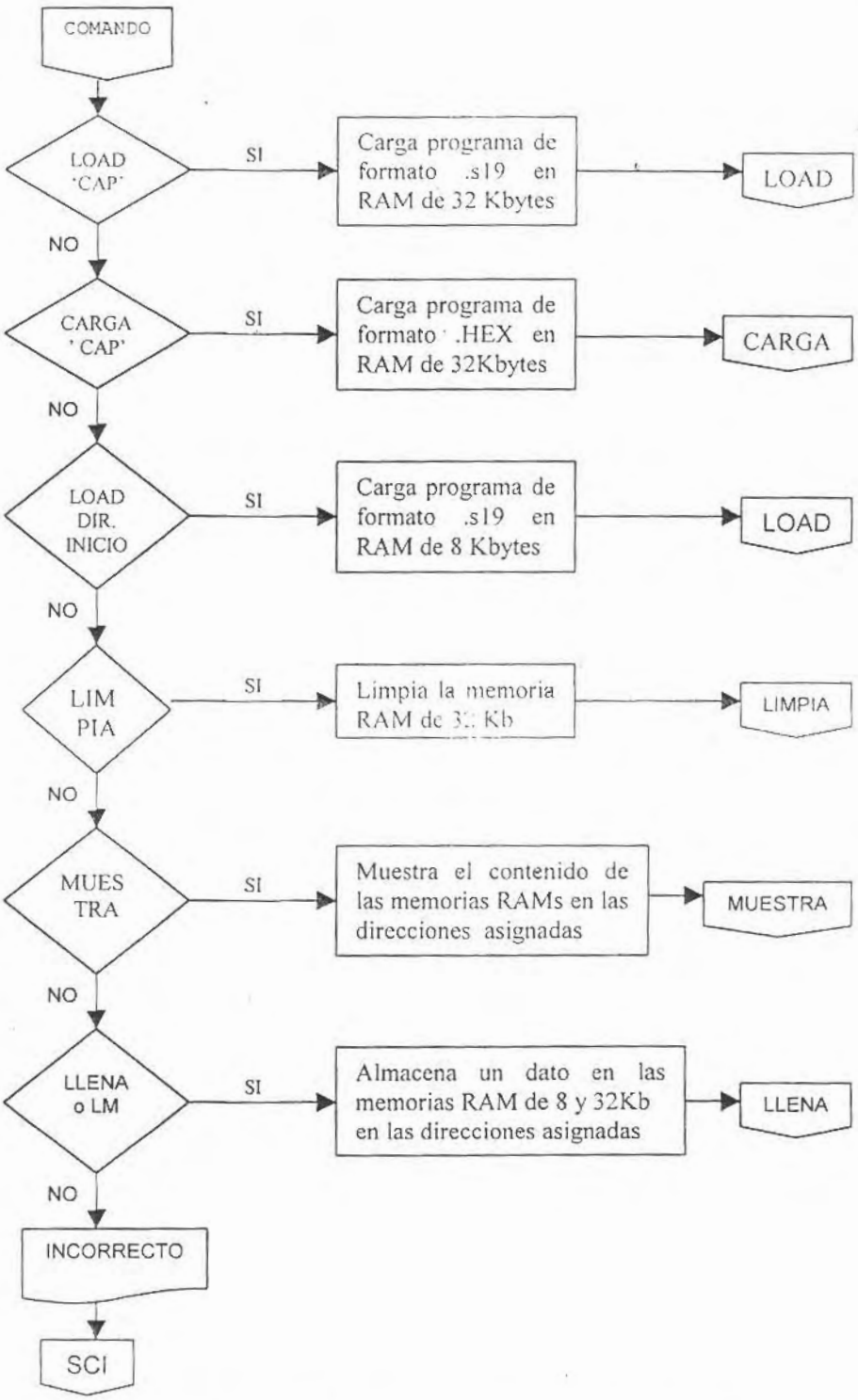


Figura 4-6 Diagrama de flujo principal de la tarjeta emuladora de EPROM's



Continuación del diagrama de flujo principal

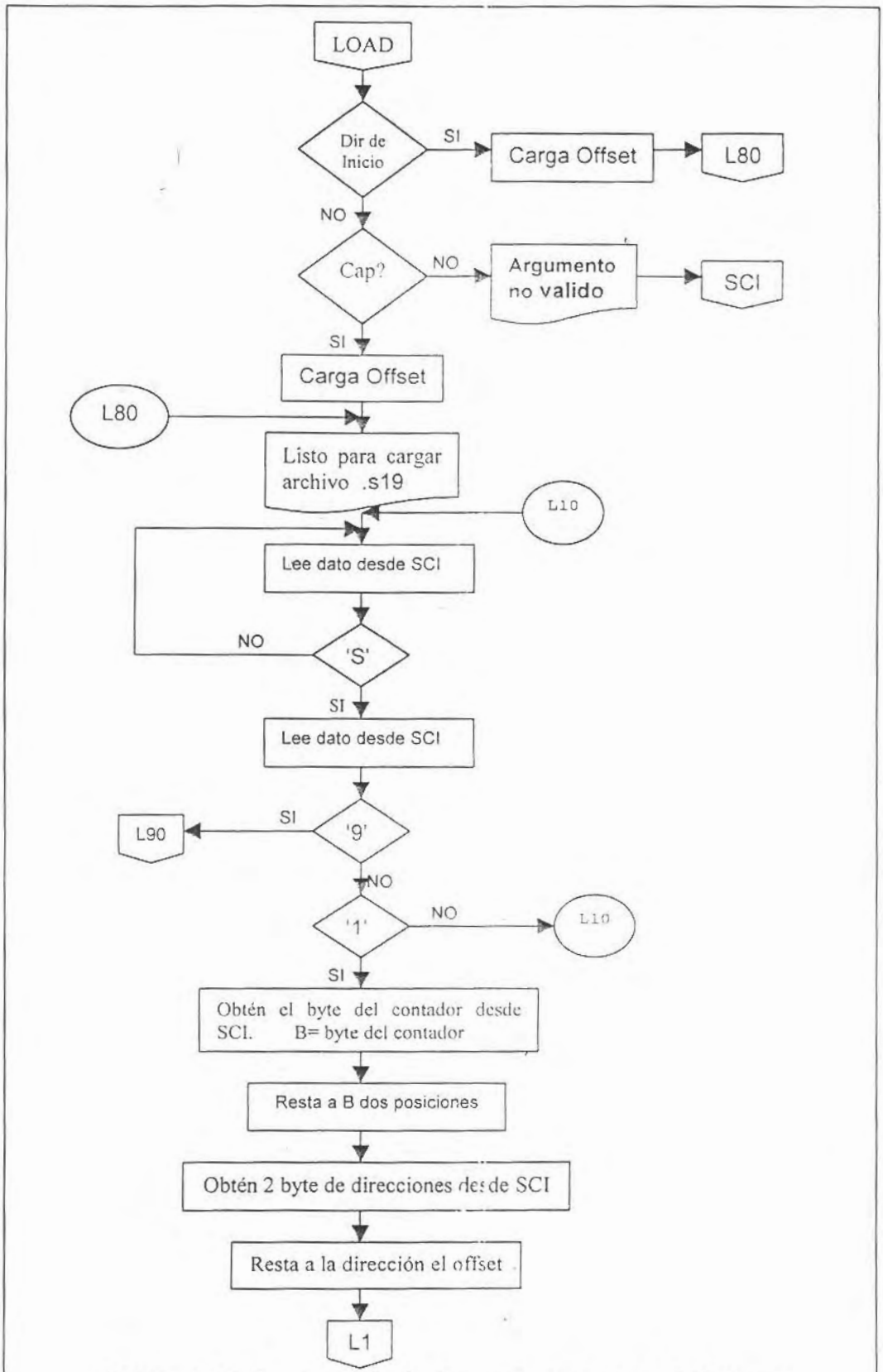
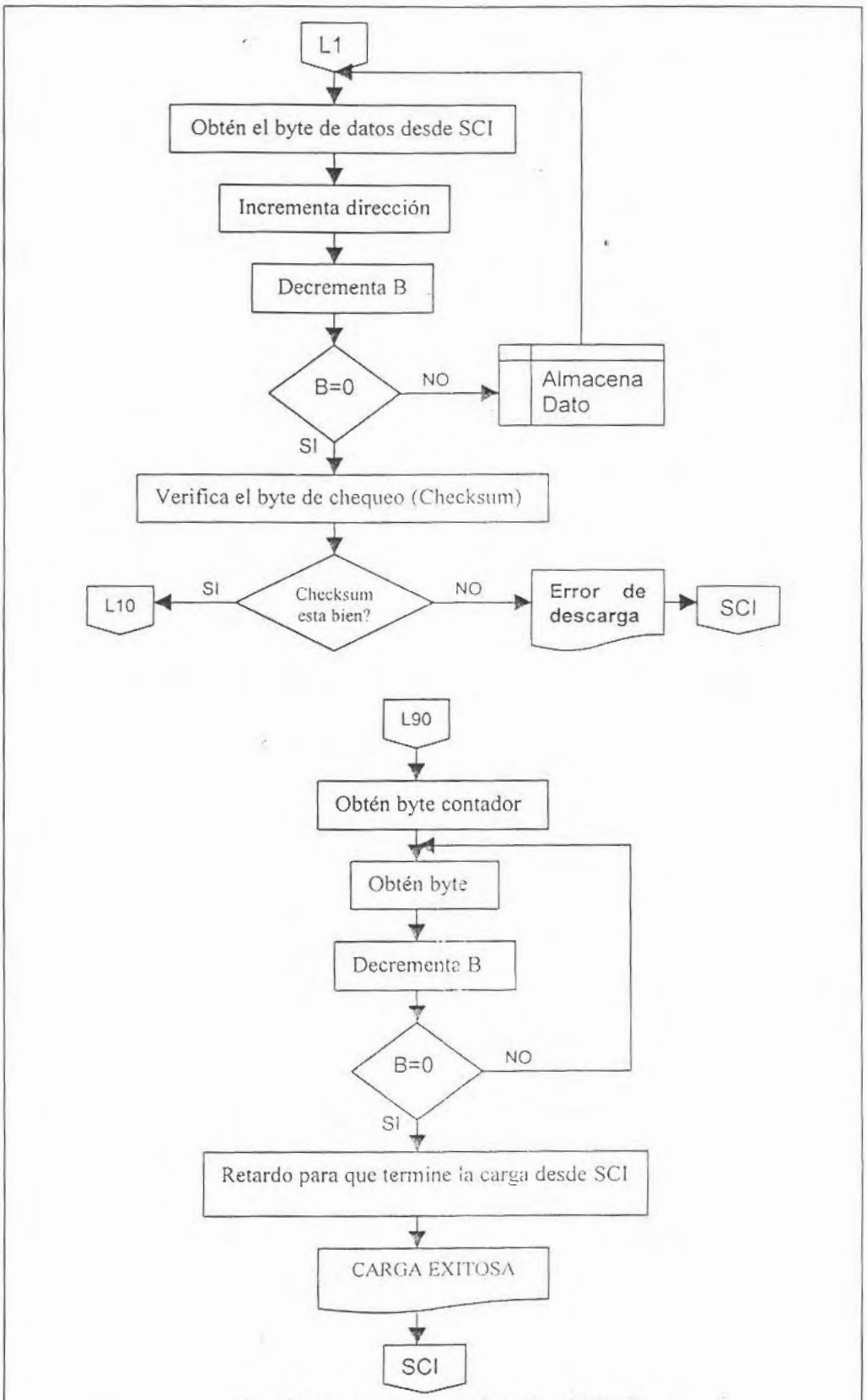


Figura 4-7 Subrutina LOAD: descarga un archivo de formato MOTOROLA



Continuación de la subrutina LOAD

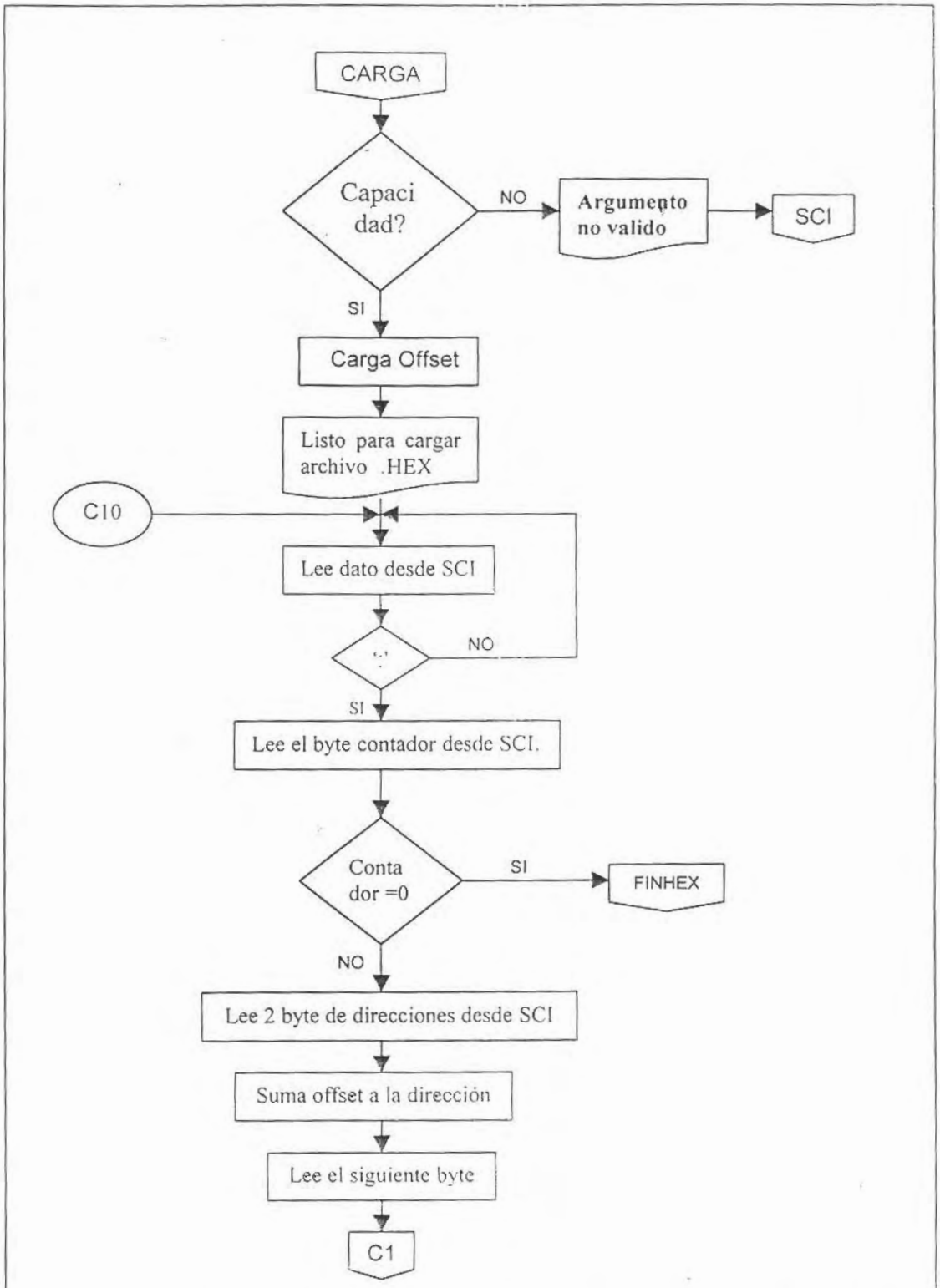
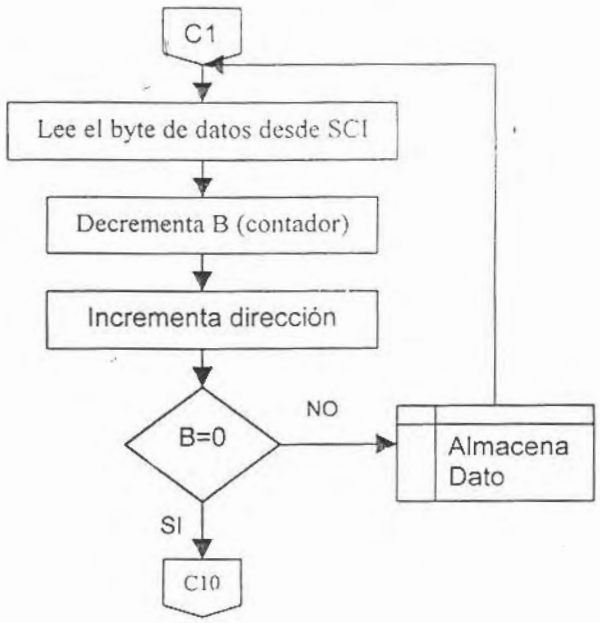


Figura 4-8 Subrutina CARGA: descarga un archivo de formato INTEL



Continuación de la subrutina CARGA

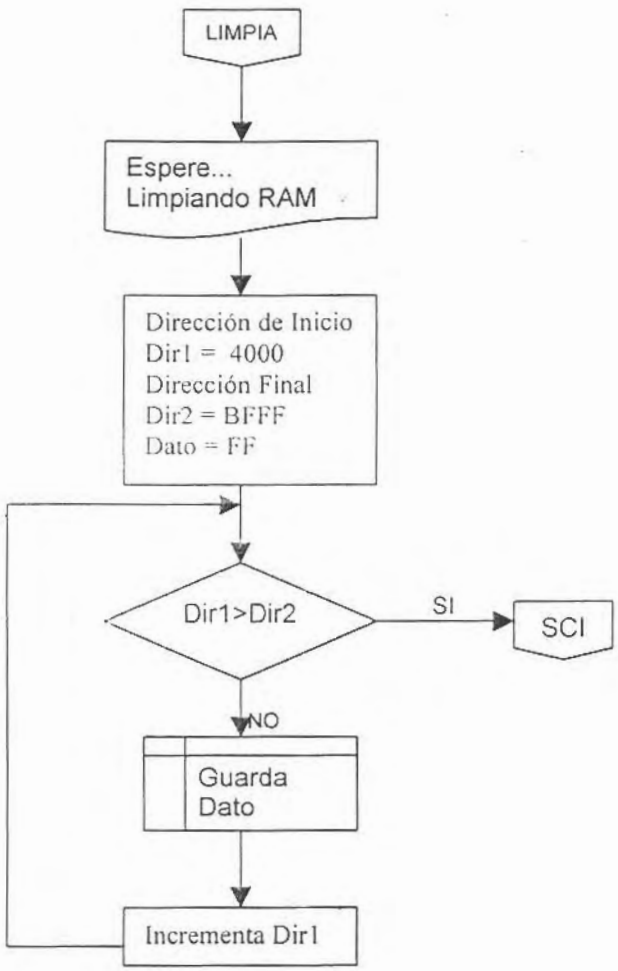


Figura 4-9 Subrutina LIMPIA: limpia la memoria RAM de 32Kbytes

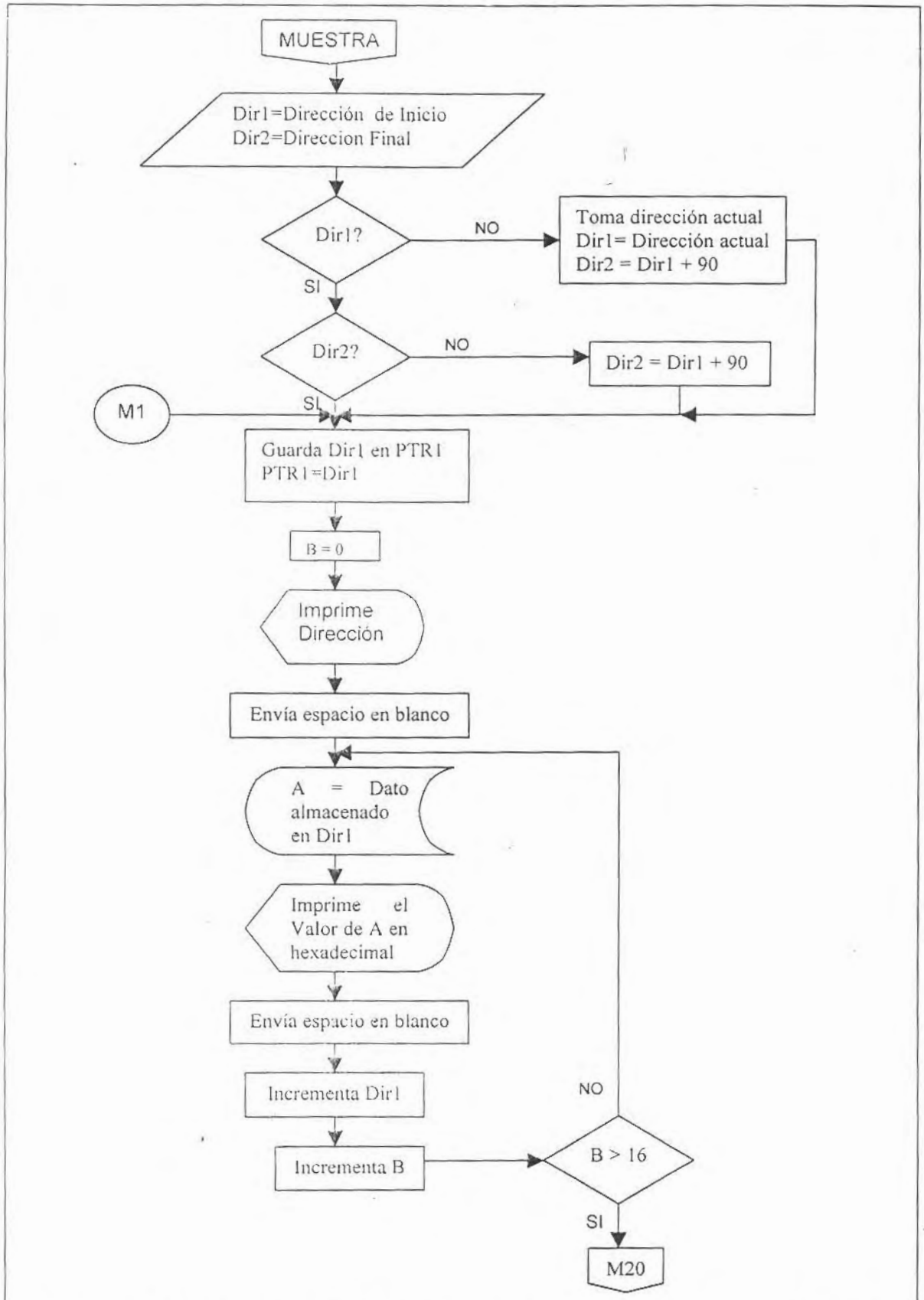
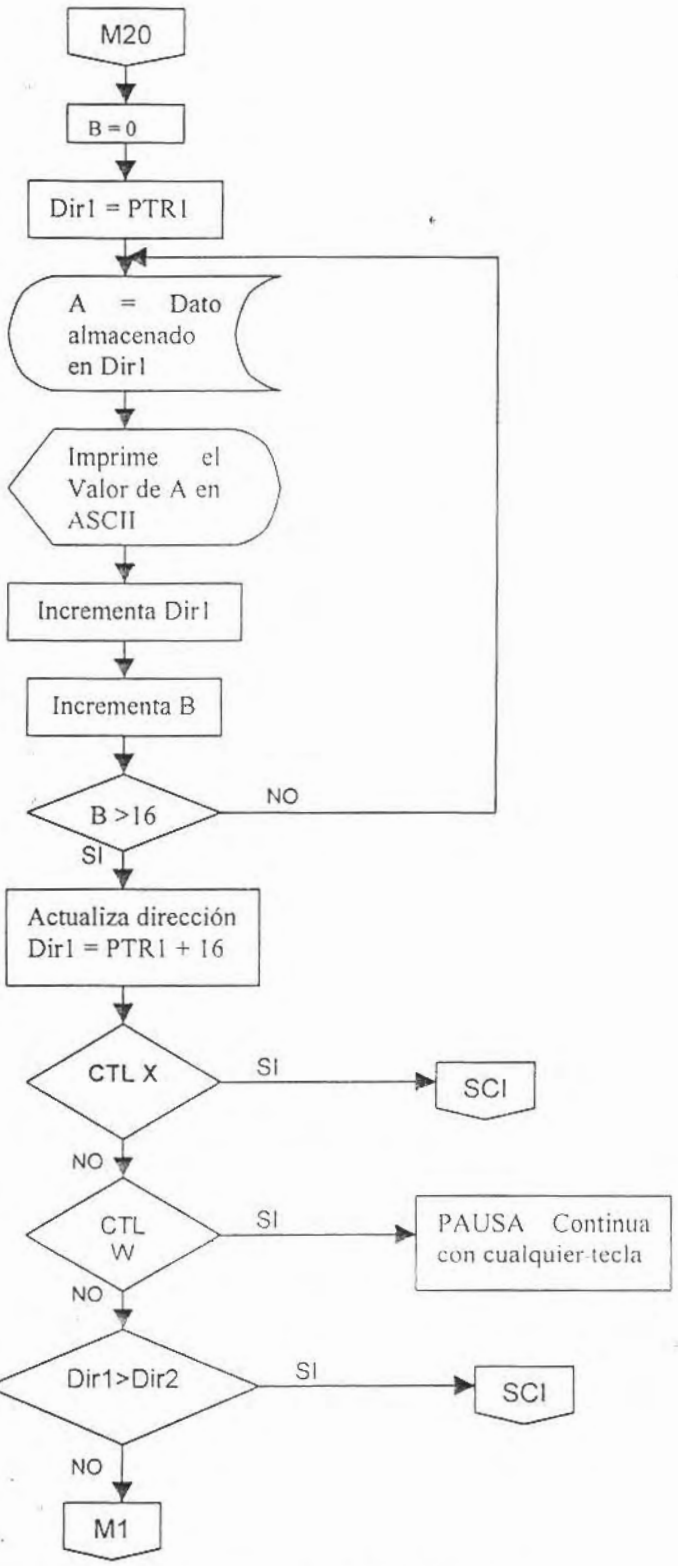


Figura 4-10 Subrutina MUESTRA: muestra el dato contenido en una dirección



Continuación de la subrutina MUESTRA

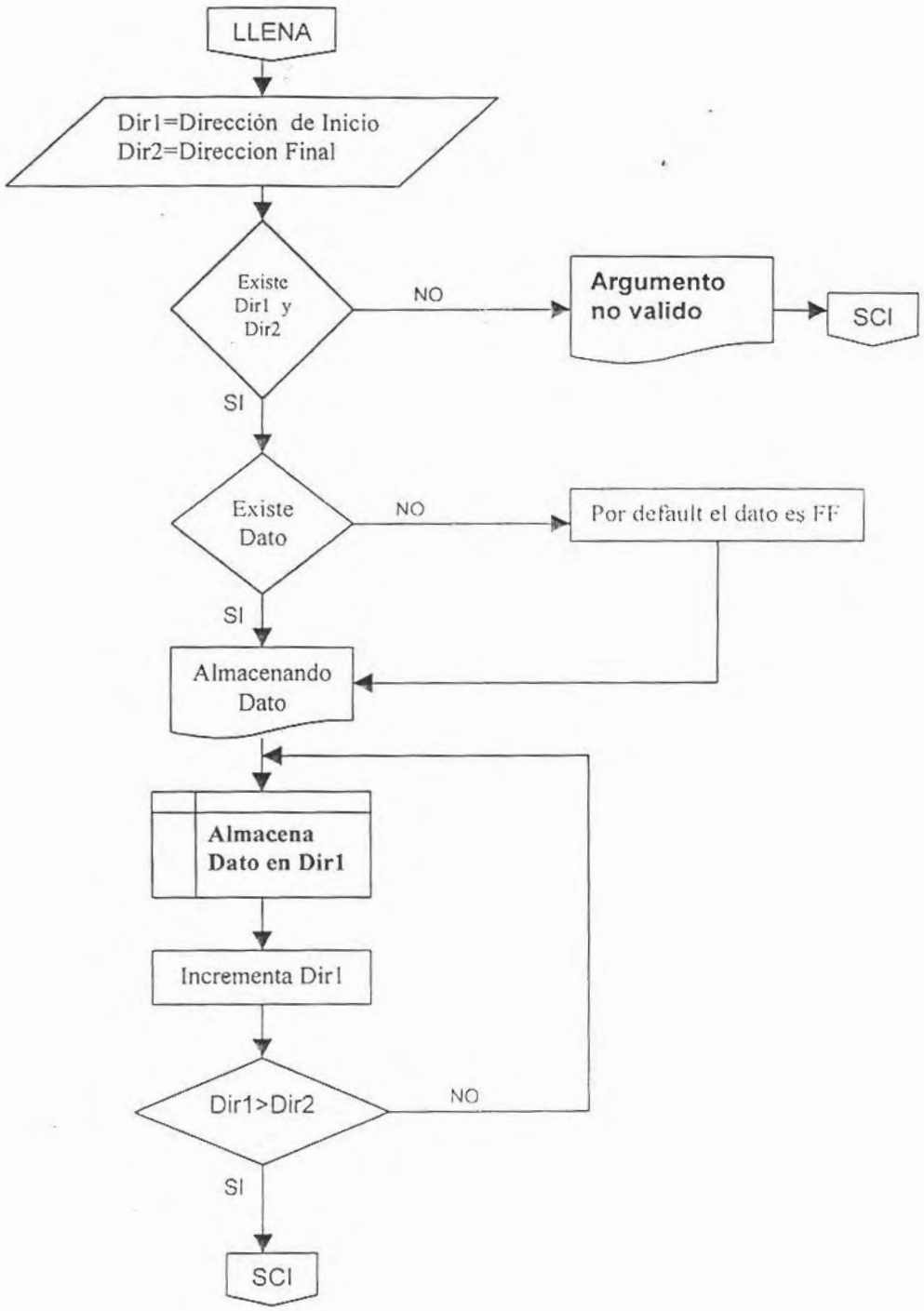


Figura 4-11 Subrutina LLENA: almacena un dato en la memoria

CAPÍTULO 5

FUNCIONAMIENTO

En este capitulo se describen los funcionamientos de la Tarjeta Emuladora de EPROMs: **TARJETA EMULADORA Y TARJETA DE EVALUACIÓN.** También se presenta la forma de configurar al hardware para el adecuado funcionamiento.

El funcionamiento de la tarjeta emuladora es simple, con una computadora personal (PC). Utilizando un editor de texto o un procesador de palabras, se escribe el programa del usuario o código fuente en *lenguaje ensamblador*, ejecutando un programa de traducción llamado Ensamblador se procesa el archivo fuente. El archivo ensamblado se transfiere con formato hexadecimal de Intel o Motorola hacia la memoria RAM de la tarjeta emuladora de EPROM's, posteriormente los datos son leídos por el huésped o sistema mínimo, observe la siguiente figura.

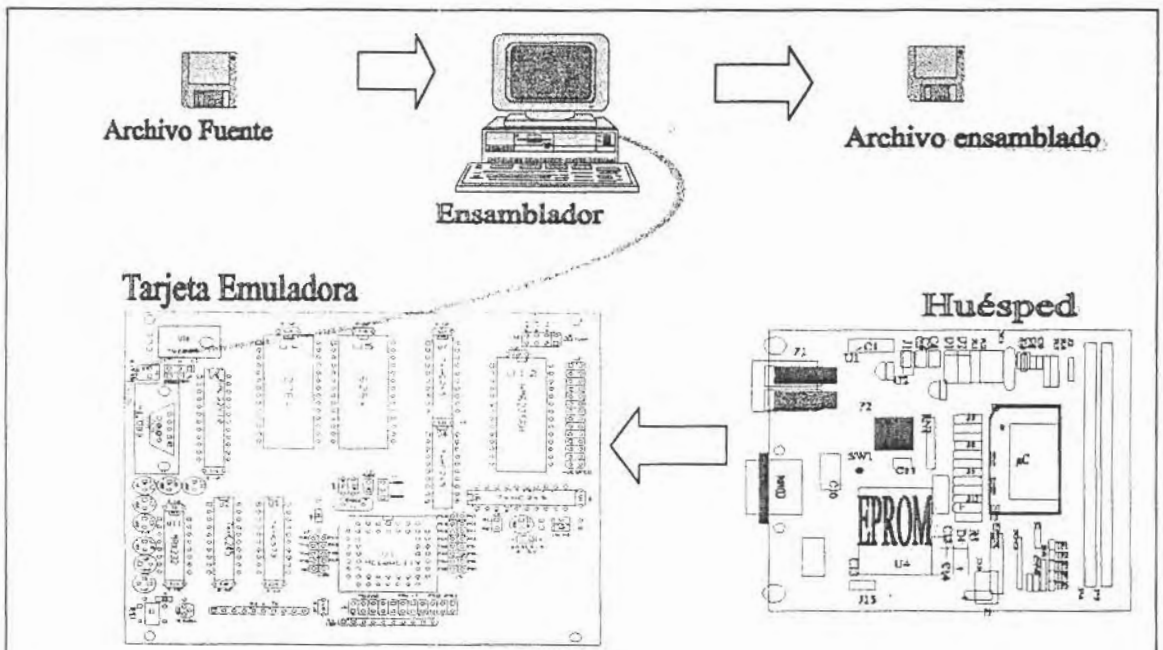


Figura 5-1 *Conexión de la Tarjeta Emuladora*

5.1 REQUERIMIENTOS

A) Se requiere de una computadora personal (PC) con microprocesador 286 o superior.

B) Tener un programa de comunicaciones; por ejemplo el programa proporcionado por el fabricante del microcontrolador MC68HC11E1 llamado IASM11 o el programa HYPERTERMINAL de WINDOWS.

Configurar el protocolo de comunicación con la siguiente información:

- ❖ Velocidad de 9600 bauds
- ❖ 1 bit de inicio, 8 de datos, 1 bit de paro
- ❖ Sin paridad

C) Fuente de voltaje.

D) Programa del usuario.

E) Para realizar la comunicación de la PC con el sistema se necesita un cable en cuyos extremos contienen conectores tipo DB9 macho y hembra.

F) La conexión del sistema (tarjeta emuladora de EPROM's) con el huésped o sistema mínimo se realiza mediante un cable plano, el cuál tiene en sus extremos un cabezal hembra y una base.

5.2 CONFIGURACIÓN

La configuración de los modos de funcionamiento del sistema se realiza por medio del hardware, es decir físicamente se colocan o se quitan los conectores, como a continuación se explica:

ALIMENTACIÓN DE LA TARJETA EMULADORA

La alimentación de la Tarjeta se realiza de dos formas mediante el conector J4 (ver figura 5-2)

a) Con 5 volts regulados y,

b) El dispositivo MC7805 permite alimentar a la Tarjeta con un voltaje dentro del rango: de 7 a 35 volts

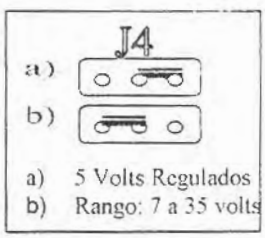


Figura 5-2 *Alimentación*

El conector J9 proporciona 5 volts para alimentar en modo Emulador al Huésped (ver apéndice A3).

FUNCIONAMIENTO

TARJETA EMULADORA

En modo EMULADOR se selecciona la capacidad de la memoria a Emular mediante las tres primeras posiciones del switch S2 (Como se explica en SELECCIÓN DE LA MEMORIA EPROM), la cuarta posición del switch debe estar activado permitiendo así la habilitación de la memoria RAM (U12).

Es importante **no colocarse el cable plano** al cabezal macho (J2); además de tener la configuración física del sistema en Modo Multiplexado Expandido (J3).

SELECCIÓN DE LA MEMORIA EPROM

La tarjeta tiene la capacidad de emular EPROMs de las siguiente capacidades: 2Kb, 4Kb, 8 Kb, 16 Kb y 32Kb. Esto se realiza por medio de las tres primeras posiciones del switch S2, la selección se realiza de acuerdo a la siguiente tabla:

	2716(2Kb)	2732(4Kb)	2764 (8Kb)	27128 (16Kb)	27256 (32Kb)
A11	OFF	ON	ON	ON	ON
A13	OFF	OFF	OFF	ON	ON
A14	OFF	OFF	OFF	OFF	ON

***Tabla5.1** Selección de la memoria EPROM*

TARJETA DE EVALUACIÓN

MODO SENCILLO O MODO MULTIPLEXADO EXPANDIDO

Al poner el conector J3 el funcionamiento de la tarjeta será en modo sencillo y el modo multiplexado expandido se obtiene cuando MODB del microcontrolador se encuentra a un voltaje alto (véase tabla 4.1 del capítulo 4), es decir se retira el conector J3, observe la siguiente figura.

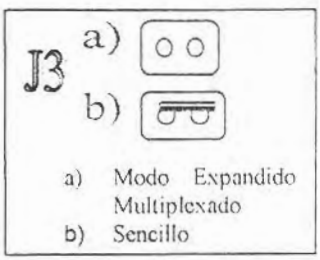


***Figura 5-3** Selección del modo sencillo o Expandido multiplexado*

5.3 OPERACIÓN

Se explica el funcionamiento de los Modos de Operación del sistema; así como sus respectivos diagramas a bloques. Como se comentó en el capítulo 4, los 8 bits de direcciones menos significativos y el bus de datos provenientes del microcontrolador se encuentran multiplexado en el tiempo para la demultiplexación se requiere del dispositivo 74HC573 (LATCH), los 16 bits de direcciones se controlan con dos buffers MC74HC245, estos buffers son bidireccionables de tres estados a 8 bits y dos más controlan el flujo de datos. El control de las señales de habilitación y dirección son enviadas por el dispositivo GAL22V10, generando las señales de control CS_1, CS_2, CS_3, CS_4, CS_5, RD, WR; sus ecuaciones se pueden observar en la tabla 4.3, capítulo 4.

5.3.1 TARJETA EMULADORA

La tarjeta Emuladora de EPROMs se puede transportar sin perder los datos contenidos en la memoria RAM de 32Kbytes, gracias a una batería de 3.5 Volts; se retienen los datos sin necesidad de una fuente externa.

Verificando que el conector J3 no se encuentre colocado, además de seleccionar correctamente la memoria a Emular, controlada por las tres primeras posiciones del switch S2.

Recuerde no conectar el cable plano que comunica a la tarjeta emuladora con el huésped, con la ayuda de la figura 5-4 se explica el funcionamiento del emulador (descarga de un archivo a la memoria RAM de 32 Kbytes).

En el primer ciclo de reloj (E) el flujo de información contiene los 8 bits menos significativos de direcciones obtenidos en la salida del dispositivo 74HC573 (LATCH) y los 8 bits más significativos. El LATCH se habilita con un nivel alto proveniente de la señal de control *AS*. Así las terminales de dirección de la RAM (o cualquier otro dispositivo) verán el byte bajo y el byte alto de direcciones en un ciclo entero de reloj.

Durante la segunda mitad del ciclo de reloj (E) los datos fluyen a través del buffer U2 de B a A (de izquierda a derecha) mandando el flujo de información al siguiente buffer (U6). En este ciclo la señal de control *AS* genera un nivel bajo, habilitando la compuerta (G) del

buffer U2 y como se esta realizando una escritura la señal de control *R/W** coloca un nivel lógico bajo habilitando al DIR de los buffers.

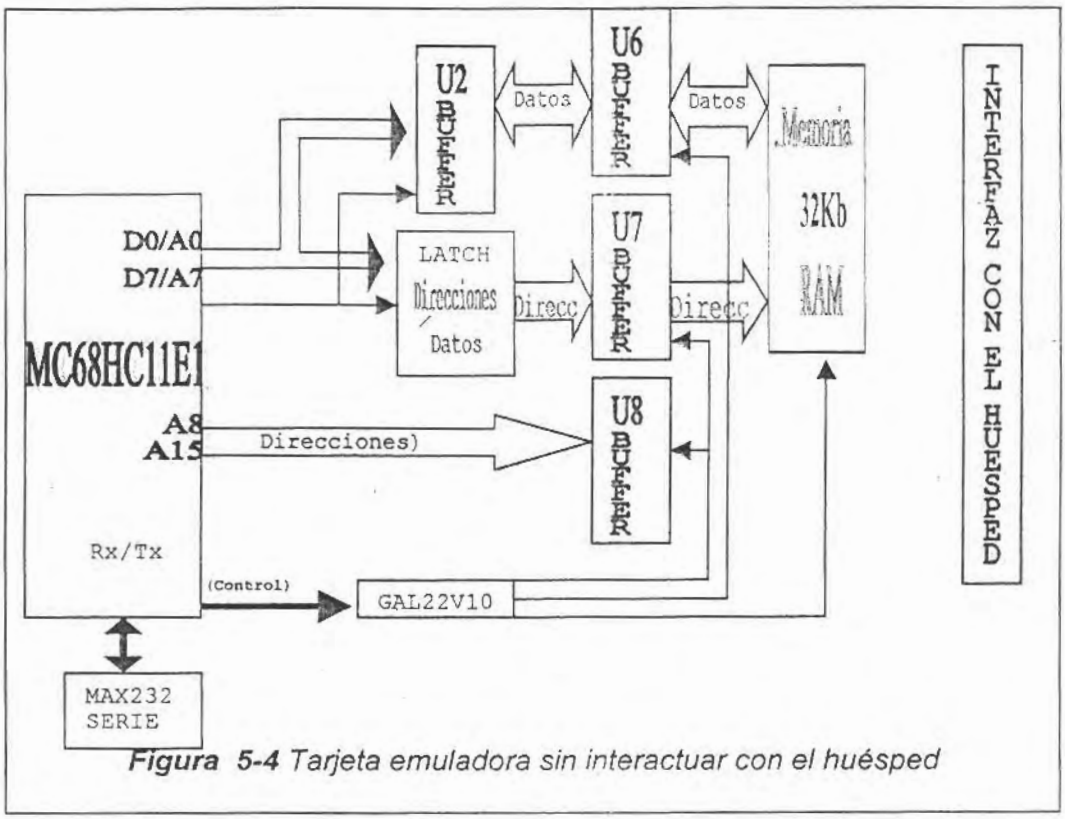


***Figura 5-4** Tarjeta emuladora sin interactuar con el huésped*

El buffer U6, funciona de manera similar al buffer U2, pero es habilitado a través de señal de control *CS_5*.

El flujo de direcciones fluye de A a B en los buffers U7 y U8, los cuales se habilitan a través de la señal de control CS_4. La función de los buffers es evitar conflictos con las direcciones del huésped y las direcciones del MC68HC11.

El flujo de datos y direcciones llegan a la memoria RAM de 32 Kb (U12) que fungirá como la EPROM a Emular, mientras que en el huésped verá a esta memoria como una ROM.

Las señales de habilitación en la memoria RAM son generadas por el dispositivo GAL22V10, las cuales se tratan de la siguiente forma:

- La habilitación de salida (/OE), se activa por la señal RD generada por la GAL22V10A o la señal del huésped obtenida del cabezal doble J2.
- La habilitación de escritura (/WE), se genera con la señal de control WR
- El **/CS** se habilita de la señal de control CS_3 representada por la ecuación: /CS_3= /A15 * A14 * E + (A15 * /A14 * E) ó de la señal CS_HUES proveniente del huésped (J2).

Después de haber descargado el programa compilado del usuario a la memoria RAM se procede a colocar el cable plano que es la interfaz de la tarjeta con el huésped, donde un extremo se conecta al cabezal doble (J2) y el otro tiene una base que simulara al EPROM correspondiente se colocara en el lugar destinado en el huésped.

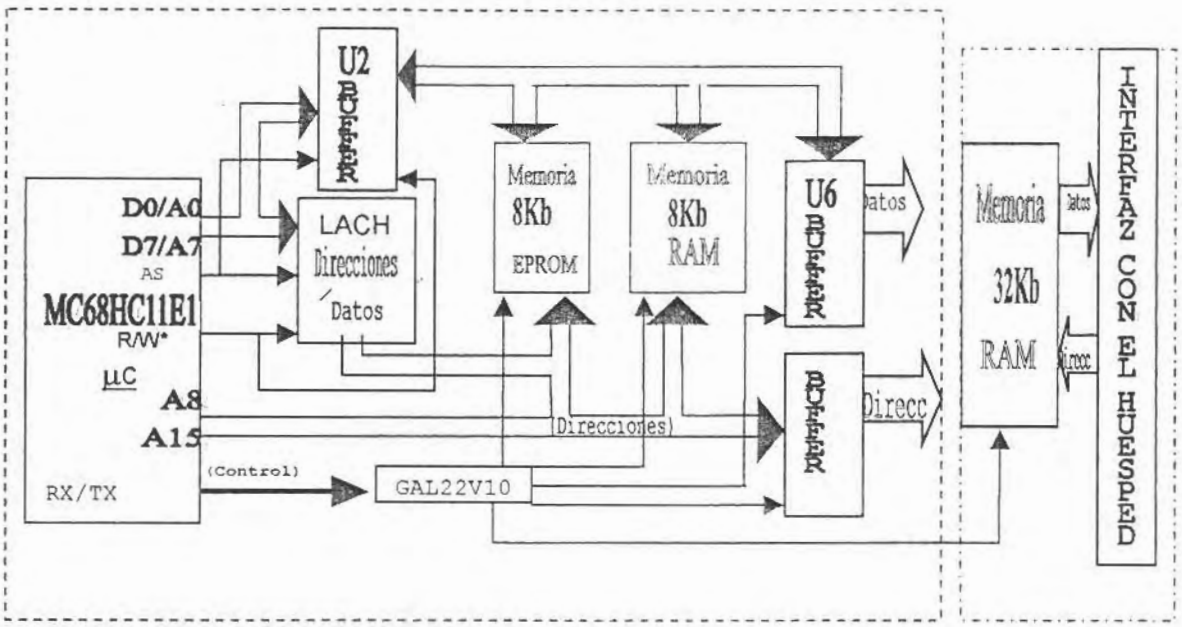


Figura 5-5 *Conexión del cable plano al huésped*

Se alimenta al huésped o sistema mínimo, el cual direcciona y habilitada a la memoria RAM de 32Kbytes, obteniendo así los datos de la memoria. De esta forma el huésped o sistema mínimo lee la memoria RAM como si fuera la memoria EPROM.

5.3.2 TARJETA DE EVALUACIÓN. MODO SENCILLO Ó MODO MONO INTEGRADO.

Una vez colocado el conector J3 (ver apéndice A3), el sistema trabaja en modo sencillo. El MC68HC11E1 no tiene buses externos de datos ó direcciones, está limitada por la capacidad de memoria interna, es decir en este modo no contamos con memorias externas. Contiene en su mapa de memoria 512 bytes en RAM, 512 bytes en EEPROM; así

la cantidad de información esta sujeta a la capacidad de memoria disponible. Este modo emplea todos los elementos periféricos en su totalidad.

Se tienen 5 puertos disponibles:

a) Puerto A, 3 de sus líneas se configuran como entrada, 1 línea bidireccional y 4 líneas de salida. Estas líneas sirven para las diversas funciones del temporizador interno en los modos captura y comparación. Los datos del puerto A son leídos desde y escritos hacia el registro PORTA, su dirección es $1000H.

b) En este modo las líneas del puerto B se configuran totalmente como salidas son de propósito general. La dirección de este registro (PORTB) es $1004.

c) En este modo las líneas del puerto C se configuran como Entrada/Salida de propósito general. La dirección $1003 corresponde a este puerto PORTC.

d) Puerto D, tiene 4 líneas bidireccionales de Entrada/Salida de propósito general, pueden ser individualmente configuradas como entrada o como salida. La dirección para accesar este puerto es la $1008H. Si la interfaz serial periférica (SPI) se activa, las terminales PD2/MISO, PD3/MOSI, PD4/SCK y PD5/SS son líneas dedicadas a las funciones de SPI. Las líneas PD0/RxD y PD1/TxD son comunes con la entrada y la salida del puerto serie asíncrono o SCI.

e) El puerto E, permite usar sus líneas como entradas digitales de propósito general y también son las entradas del convertidor A/D. La dirección para leer este puerto es $100AH.

MODO MULTIPLEXADO EXPANDIDO

Retirando manualmente el conector J3, el sistema funciona en modo Multiplexado Expandido. Este modo contiene las memorias externas: RAM de 8 y 32 Kbytes (U5 y U12 respectivamente) las cuales están disponibles para almacenar información del usuario y la memoria EPROM, la función de esta última es inicializar al sistema. Son habilitadas por las señales de control CS_2, CS_3 y CS_1 respectivamente.

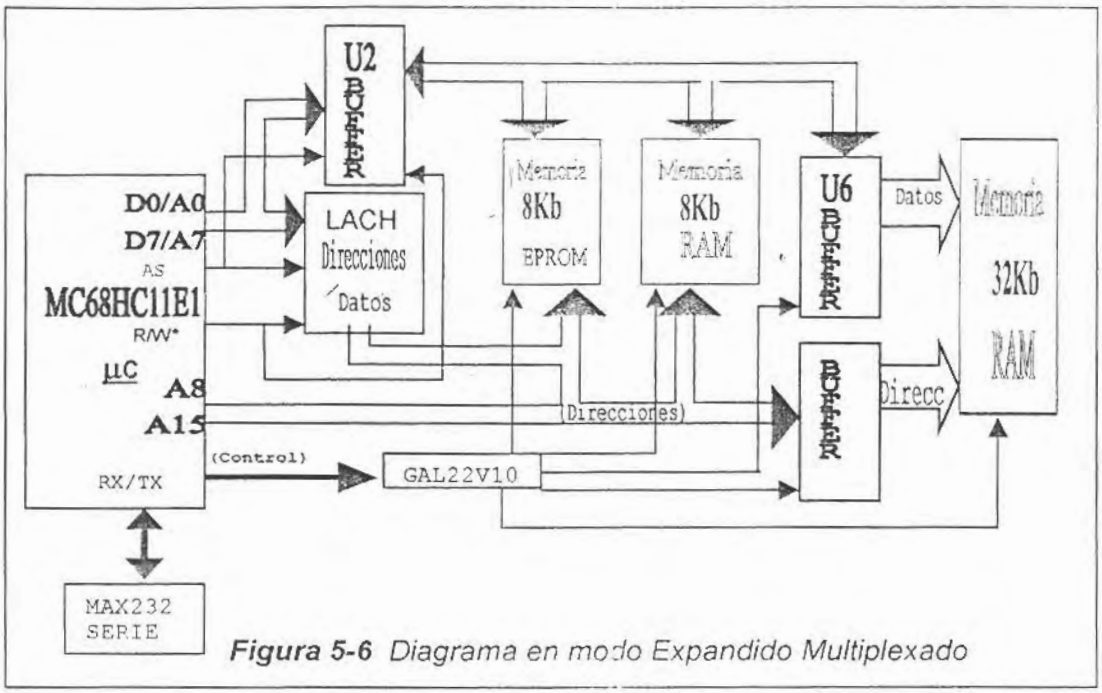


Figura 5-6 *Diagrama en modo Expandido Multiplexado*

En el modo Multiplexado Expandido el puerto B es remplazado por el byte de direcciones altas ($A_8 - A_{15}$); el puerto C es remplazado por el byte de direcciones bajas (A_0–A_7) y por el byte de datos. Estos bytes (dirección/datos) del puerto C no pueden aparecer en el bus al mismo tiempo porque se encuentran multiplexados en el tiempo, se deben demultiplexar.

Al inicio de un ciclo de reloj **E** (durante la primera mitad del ciclo, E se encuentra en bajo) el MCU coloca el byte más significativo de direcciones en el puerto B y los pines del puerto C envía las direcciones del byte menos significativo al mismo tiempo, pero esta es retenida hasta que el reloj suba. Generándose durante esta primera mitad del ciclo un nivel alto en la señal de control AS, la cual habilita el LATCH, dejando pasar el flujo y obtener en la salida del 74HC573 el byte bajo de direcciones. Para mejor entendimiento observe el diagrama de tiempo (figura 4.4, capítulo 4).

La señal STROBE (AS) esta a un nivel alto cuando el puerto C tiene un byte de direcciones y un nivel bajo para el byte de datos.

En la segunda mitad del ciclo de reloj (cuando E esta en alto), el MCU puede realizar una lectura o escritura de (enviar o recibir) datos. Si es un ciclo de lectura, la señal R/W*

permanece en un nivel alto habilitando al control de dirección (DIR) del buffer U2 de A a B (derecha a izquierda). En este ciclo la señal de control AS proporciona un nivel bajo a G (habilitación), dejando pasar el flujo de datos provenientes de alguna memoria externa.

Si es un ciclo de escritura (las entradas de datos en la RAM vienen del bus), el MCU coloca un dato en las líneas del puerto C, también coloca R/W* en bajo. El flujo de datos pasa a través del buffer U2, su habilitación en la compuerta (G) es dada por un nivel bajo de la señal de control AS y como se esta realizando una escritura hacia las memorias, el DIR del buffer se activa con un nivel bajo generado por la señal de control R/W*. Las memorias son habilitadas por las señales generadas en el dispositivo GAL22V10. Este dispositivo también habilita a los buffers U6, U7 y U8.

Los puertos disponibles para el usuario son: A, D y E descritos anteriormente.

CAPÍTULO 6

CONCLUSIONES

La construcción de este sistema abarca los objetivos propuestos. A continuación se presentan las características más importantes, ventajas y mejoras del sistema. También se mencionan las pruebas realizadas en la construcción de la Tarjeta.

- Al usar un microcontrolador se está abriendo a un sin número de "usos" para la tarjeta, ya que tiene "doble función": Puede trabajar como una TARJETA EMULADORA DE EPROM's y como una TARJETA DE EVALUACIÓN para el microcontrolador MC68HC11E1 que puede trabajar en cualquiera de sus dos modos.

- Al implementar el sistema en un circuito impreso de doble cara y utilizando la técnica "throug-hole" para comunicar ambas caras, el espacio se reduce hasta 12.2 por 11 centímetros.

- La tarjeta contiene 11 circuitos integrados que se encuentran disponibles en el mercado.

- La tarjeta emuladora puede transportarse sin perder la información en la memoria RAM de 32Kbytes debido a que cuenta con una bateria de respaldo.

- La dirección de inicio de la memoria RAM de 32 Kb puede ser de $0000H o desde la dirección $4000. Si se inicia en la dirección $0000 se debe mover la memoria RAM interna y los registros configurando el registro INIT por software y por Hardware se cambia de posición del conector J1.

- La flexibilidad de un dispositivo PLD, permite cambios futuros de direcciones en la habilitación de los dispositivos empleados.

- Las direcciones bases de habilitación de las memorias RAM son $4000, $C000, las cuales se pueden cambiar dentro del rango del mapa de memoria (Capítulo 4, figura 4.4)

- Cuando el sistema no responde por alguna falla en el microcontrolador, por parámetros erróneos u otra causa, se puede reinicializar el sistema con el RESET manual.

- Cuenta con regulador de voltaje de 5 volts, y también se puede alimentar al sistema con 5 volts regulados de una fuente de alimentación de bajo ruido

PRUEBAS REALIZADAS EN LA CONSTRUCCIÓN DE LA TARJETA

Dentro de las pruebas realizadas a la tarjeta prototipo se encuentran las siguientes:

- Se comprobó la continuidad de las pistas, es decir se verificó que las pistas no estuvieran rotas y realmente llegaran a su destino.

- Se probó que las pistas no estuvieran en corto circuito con otras.

- También se verifico la polaridad, es decir entre las conexiones de alimentación y tierra de los circuitos integrados.

- Después de soldar las bases, se verificó nuevamente la continuidad de las pistas, además de comprobar la existencia de 5 volts de alimentación.

- Las pruebas de la primera fase se realizaron en modo sencillo (observe el diagrama del apéndice A3).

Después de la exitosa inspección para localizar líneas o pistas en cortocircuito con otras, continuidad de las mismas y la polaridad correcta, se procedió a:

- Colocar el microcontrolador, la fase de comunicación (MAX232) y alimentando del sistema, además de la configuración en modo sencillo; verificando que el microcontrolador nos diera una frecuencia de trabajo de 2 MHz, esto físicamente se realizo mediante la ayuda de un Osciloscopio monitoreando la señal de reloj E. También se observó la alimentación de la señal RESET, que fuera mayor a los 4 volts.

- Continuamos con la parte de comunicaciones; esto es, conectar la computadora personal con la tarjeta, cabe mencionar que en modo sencillo el microcontrolador tiene instalado desde fábrica el programa monitor *Buffalo*, ubicado en su área de ROM.

  Esta fase fue ***exitosa*** por lo que la tarjeta funciona bien en modo sencillo.

Se continuó con la siguiente fase, configurando al microcontrolador en su Modo multiplexado expandido. Se realizó el BIOS de la tarjeta y se implementó al EPROM, con ello verificamos que la comunicación se estableciera y que efectivamente se leía la EPROM.

- Después se realizó una prueba cargando un programa a las memorias RAMs, con diferentes direcciones bases. El BIOS tiene la opción de cargar un dato dando una dirección de inicio y una dirección final, y también la opción de leerlos comprobando el contenido de las memorias.

- Se realizo un pequeño sistema con Les para verificar que realmente la memoria contenía los datos almacenados.

EN EL SOFTWARE DE LA TARJETA EMULADORA PODEMOS CONCLUIR:

1. El BIOS que se coloco en el área de ROM del mapa de memoria, tiene la capacidad de manejar el protocolo de comunicación serial.
2. Se puede descargar un programa de formato Motorola o formato Intel a las memorias RAMs
3. Ejecutar comandos desde la PC.
4. El usuario tiene la libertar de programar su propio BIOS, y direccionar los dispositivos según su aplicación.
5. La interfaz entre el BIOS y los programas del usuario se logran mediante las llamadas al sistema, las cuales son subrutinas localizadas en determinadas direcciones dentro del BIOS.
6. La tarjeta funciona correctamente sin perdidas de información en la transmisión serial siempre y cuando no se exceda la velocidad de transmisión (9600 bps), de lo contrario puede haber perdidas u omisiones de caracteres.

7. Cualquier comando proveniente desde la PC, puede ser cancelado mediante la tecla [CTL X]. El comando " muestra " tiene la opción de hacer una pausa con [CTL W] y continuar con cualquier tecla.

MEJORAS QUE SE PUEDEN REALIZAR EN ESTE TRABAJO.

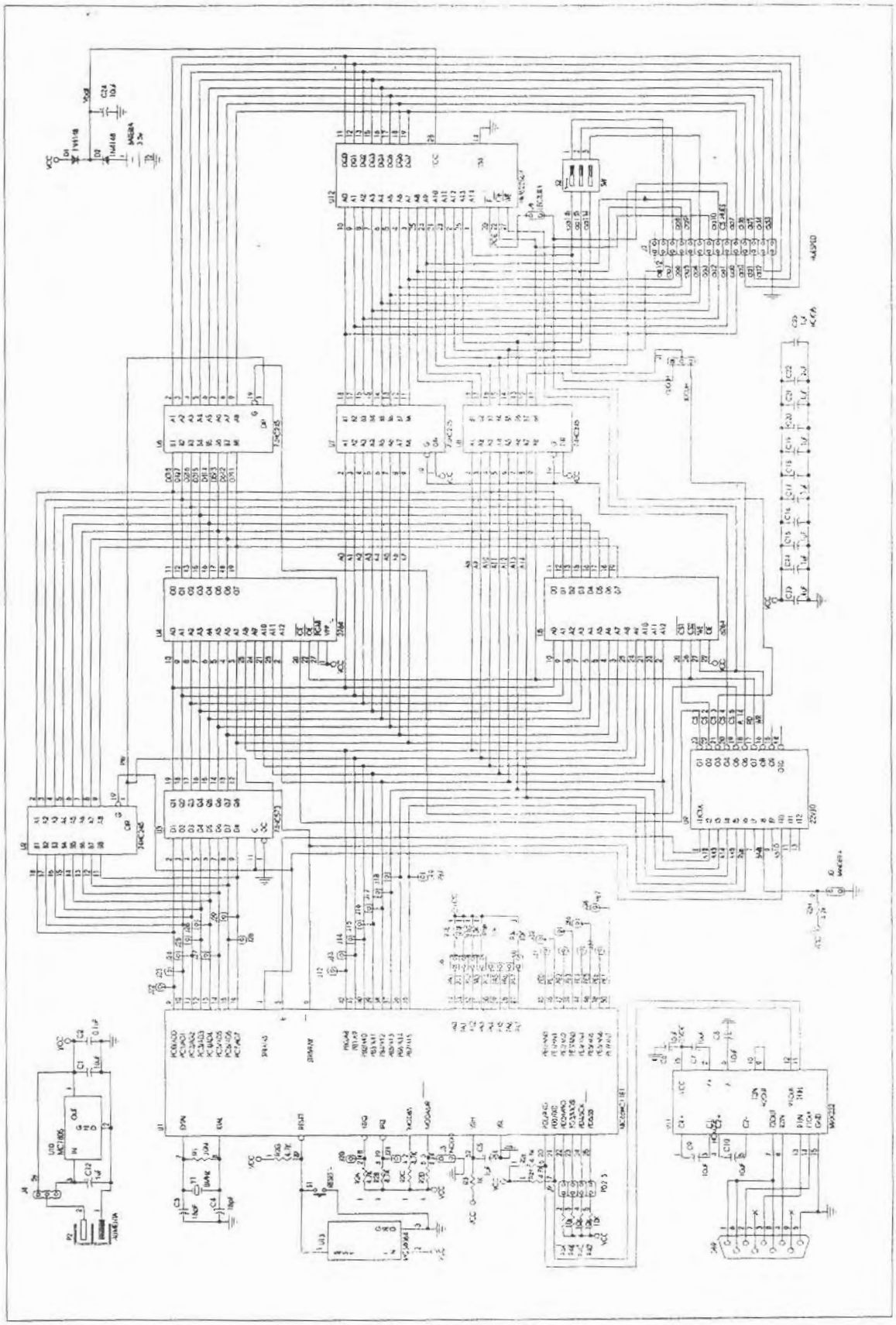
- Cambiar la pila por una pila de botón, esto nos permite una fijación de la misma y ahorra espacio.
- Para aumentar la frecuencia de trabajo se puede cambiar el cristal a un valor mayor.
- Cambiar los conectores por un switch para facilitar su uso.
- Implementar nuevos comandos, como modificar un dato en memoria, mover un bloque de memoria a otra dirección, ejecutar un programa, etc.
- Agregar unos buffers para aislar la memoria RAM del sistema al que se conecte.
- Convertirla en una tarjeta lectora de memorias EPROM's

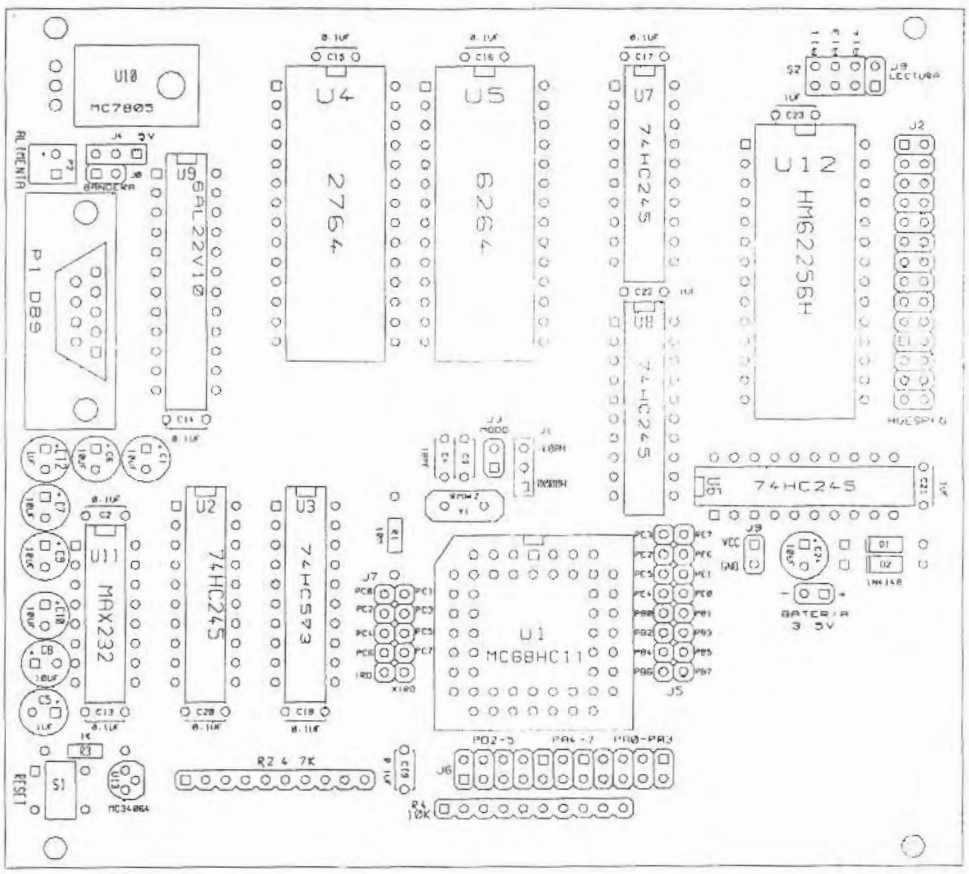
APÉNDICE A

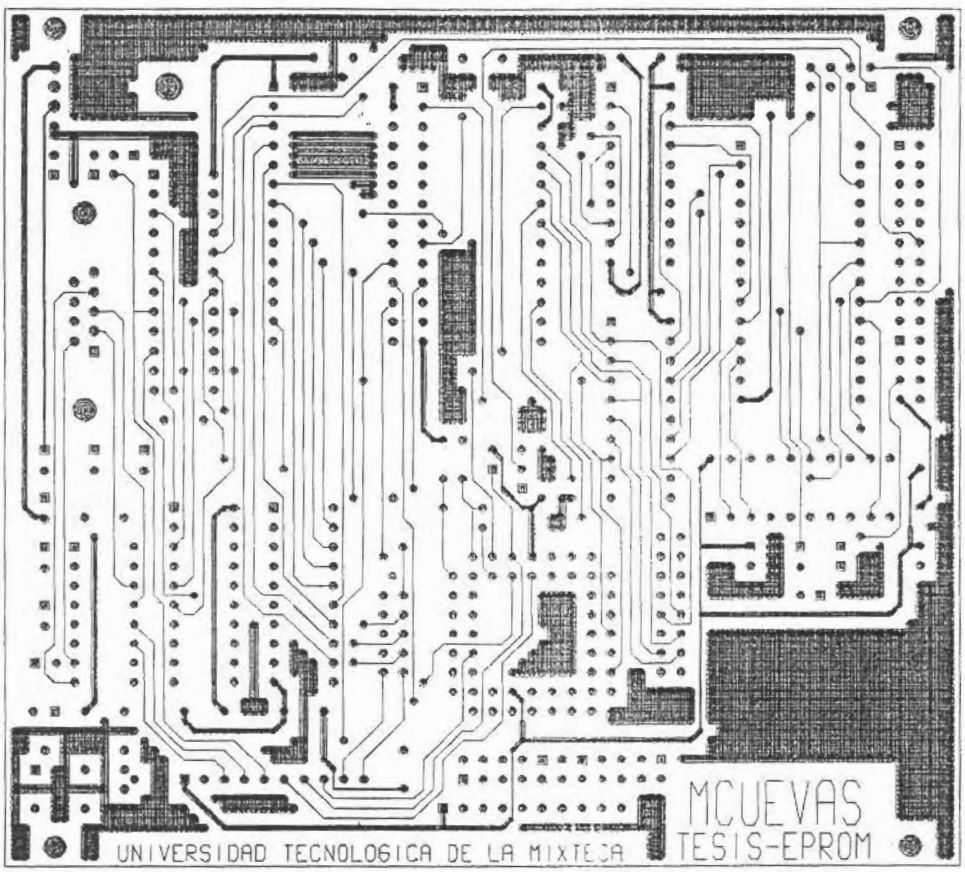
Diagrama esquemático y diseño de la tarjeta.

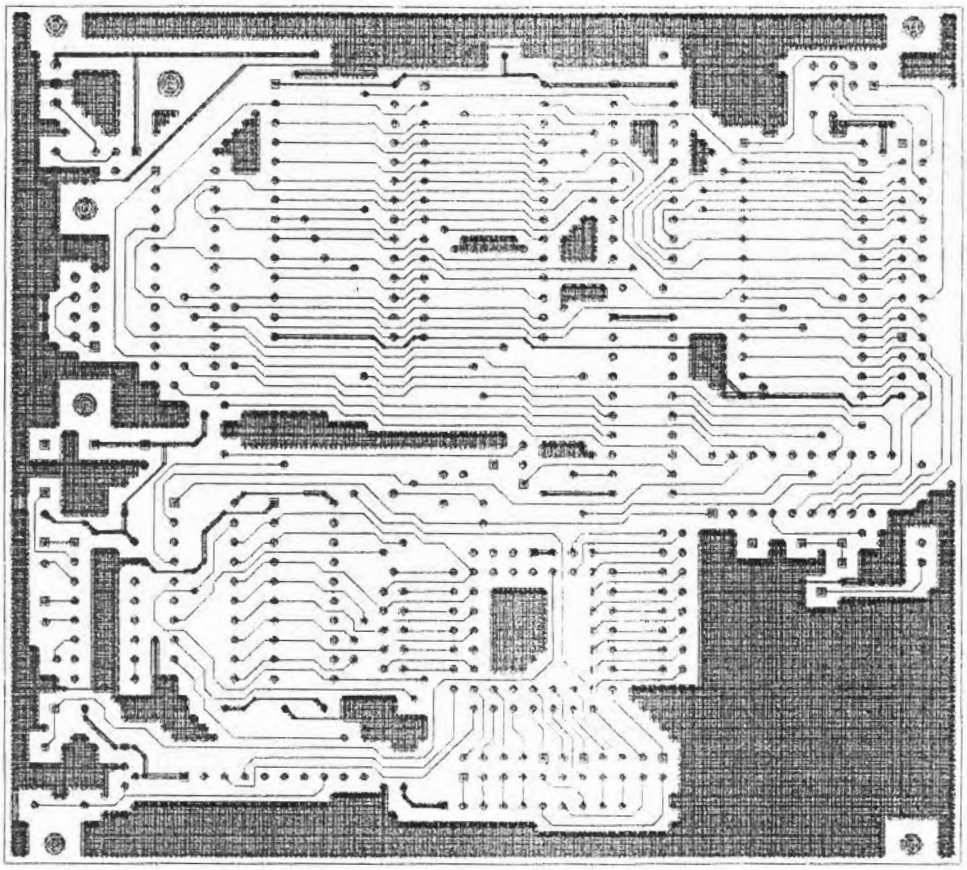
LISTA DE MATERIAL DEL SISTEMA

REFERENCIA	UNIDAD	DESCRIPCIÓN
U1	MC68HC11E1	Microcontrolador
U2, U6 - U8	74HC245	Buffer
U3	74HC573	Latch
U4	27C64	Memoria EPROM de 8 Kbytes
U5	6264	Memoria RAM de 8 Kbytes
U9	GAL22V10	Arreglo programable
U10	MC7805	Regulador de voltaje
U11	MAX232	Driver / Receiver – RS232
U12	HM62256	Memoria RAM de 32 Kbytes
U13	MC34064	Reset
C1, C6-C10, C24	10 μF	Capacitor Electrolitico @ 63 Volts
C3,C4	22 ρF	Capacitor Cerámico
C5, C12	1 μF	Capacitor Electrolitico @ 25 Volts
C2, C13 - C23	.1 μF	Capacitor de Tantalio
R1	10MΩ	Resistor a ¼ de WATT
R2	SIP 4.7KΩ 10 pines	Arreglo de resistores de 10 pines
R3	1KΩ	Resistor a ¼ de WATT
R4	SIP 10KΩ 10 pines	Arreglo de resistores de 10 pines
Y1	8 MHz	Cristal de cuarzo
D1, D2	1N4148	Diodo de Si,Fast Switching
P1	DB9	Conector DB9 para cto. Impreso
P2	Molex	MOLEX DOBLE
S1		Mini Push Button
S2		Switch de 8 pines
J0,J1,J3,J4,J8		Jumpers
J2,		Cabezal doble
Batería		Batería de 3.6 Volts




U10
MC7805
U4
2764
U5
6264
U7
74HC245
U12
HM62256H
74HC245
74HC245
U9
GAL22V10
P1 DB9
U11
MAX232
U2
74HC245
U3
74HC573
U1
MC68HC11
MC34064
RESET
R2 4.7K
R4 10K
BATERIA 3.5V


MCUEVAS
UNIVERSIDAD TECNOLOGICA DE LA MIXTECA
TESIS-EPROM



APÉNDICE B

Manual del usuario

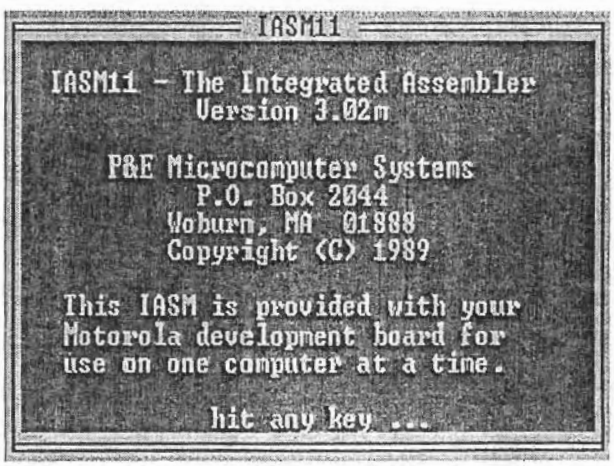
MANUAL DE LA TARJETA EMULADORA DE EPROM's

REQUERIMIENTOS

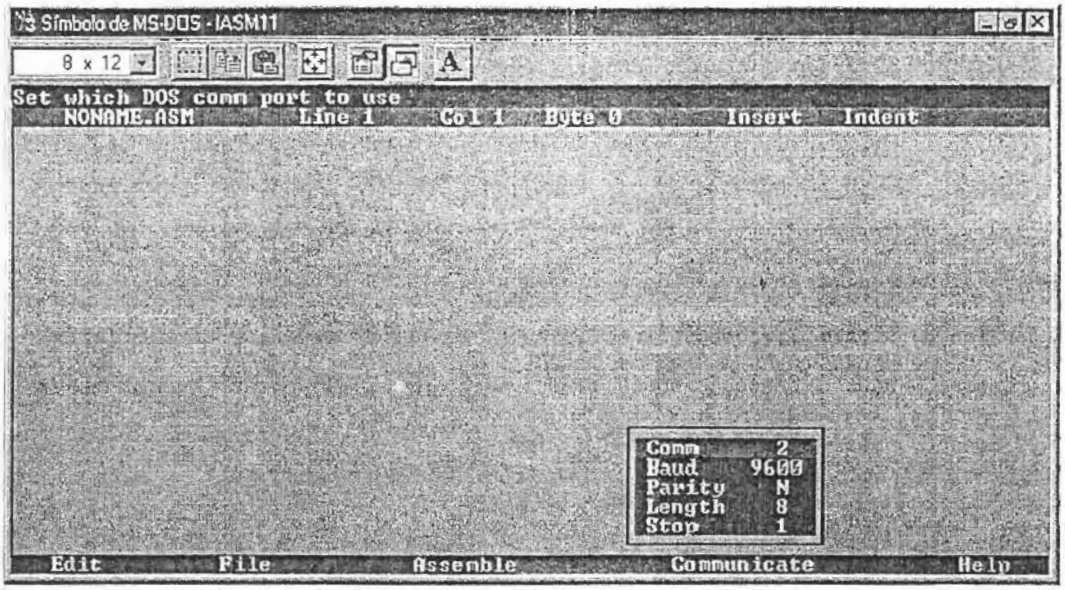
1) Se requiere de una computadora personal (PC) con microprocesador 286 o superior.
2) Un programa de comunicaciones.
3) Fuente de alimentación.
4) Cable con conectores tipo DB9 macho y hembra.
5) La conexión del sistema con el huésped es mediante un cable plano, el cuál tiene en sus extremos un cabezal hembra y una base.

CONEXIONES Y EJECUCIÓN

1. Inicializar la PC.
2. Conectar el cable a la Tarjeta Emuladora y al puerto serie (Com1 o Com2) de la PC.
3. Ejecutar el programa de comunicaciones. Por ejemplo el programa de comunicaciones de Motorola IASM11, como se muestra a continuación:



4. Seleccionar los atributos correctos y el puerto de comunicación serial. En IASM11 de Motorola presione la tecla F10 que corresponde al menú, a continuación teclee C



5. Seleccionar el modo de operación de la Tarjeta Emuladora

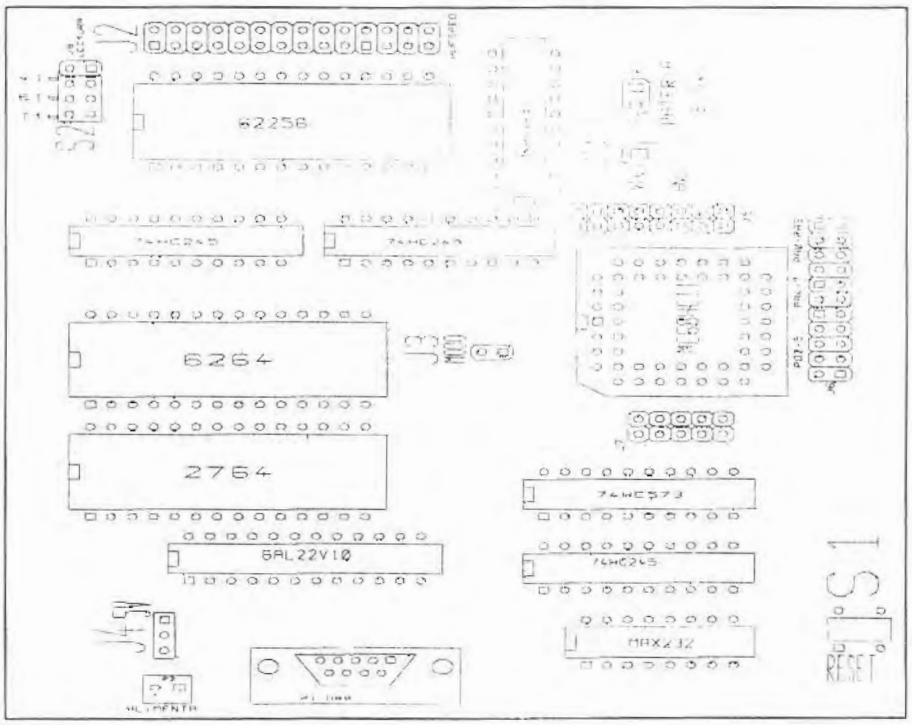


Diagrama 1 *Mascarilla de algunos componentes*

TARJETA EMULADORA

En el **modo EMULADOR** se retira el saltador J3, observe la colocación del jumper en el diagrama anterior. Es importante **no colocarse el cable plano** al cabezal macho (J2).

TARJETA DE EVALUACIÓN

Colocando el conector J3 (ver diagrama 1), se obtiene el **modo sencillo**. En este modo se trabaja únicamente con el microcontrolador, las memorias internas, registros, etc.

Retirando este conector la tarjeta Emuladora de EPROMs estará operando en **modo multiplexado expandido** y modo Emulador. Para mejor comprensión apóyese en el capítulo 4.

6. Con la tecla F7 realizamos la comunicación y F8 nos permite ampliar la ventana de comunicación serial.

7. Alimentar al sistema. Con el jumper J4 (ver diagrama 1), se selecciona la alimentación entre 5Volts regulados o dentro del rango de 7 a 35 volts. Cuando se alimenta la tarjeta con 5 Volts es necesario colocar el jumper en los dos últimos pines del jumper J4 (de izquierda a derecha) y si la alimentación es dentro del rango de 7 a 35 volts el jumper se coloca en los dos primeros pines.

8. El sistema inicia con un mensaje de presentación de la tarjeta, muestra las capacidades de las memorias a Emular y limpia las memorias RAMs. Después de la presentación aparece el prompt indicando la espera de algún comando desde la PC.

9. Los comandos tecleados en código ASCII desde la PC, son enviados vía puerto serial, para ser analizados y buscados en el programa del BIOS, si es encontrado se ejecuta la subrutina según el comando tecleado, en caso contrario termina y envía un mensaje de error.

10. Si no ha introducido ningún comando y presiona ENTER (CR), aparece el menú ó teclee el comando AYUDA ó ?. El menú presenta las siguientes opciones:

LIMPIA. Limpia la memoria RAM de 32 Kbytes

LOAD [CAPACIDAD DE LA MEMORIA] Carga un archivo .s19 a memoria RAM de 32Kbytes

LOAD [C000] Carga archivo .s19 a memoria RAM de 8 Kbytes

CARGA [CAPACIDAD DE LA MEMORIA]. Carga un archivo .HEX a memoria RAM de 32Kbytes

LLENA <DIR INICIO> <DIR FINAL> [DATO]. Llena la memoria RAM con un dato

MUESTRA <DIR INICIO> <DIR FINAL>. Muestra el contenido de la memoria

CONTROL W (CTL W) Detiene la ejecución y continua con cualquier tecla

CONTROL X (CTL X) Aborta la ejecución que se encuentre realizando

RETORNO DE CARRO (CR). Repite el último comando

[?] Presenta el menú

11. Después de la presentación del menú aparece el prompt indicando la espera de algún comando desde la PC.

12. Comandos para descargar una archivo a la memoria RAM de 32 Kbytes

El comando LOAD [CAPACIDAD DE LA MEMORIA] Carga un archivo de formato MOTOROLA (*.s19) a la memoria RAM de 32Kbytes

El comando CARGA [CAPACIDAD DE LA MEMORIA]. Carga un archivo de formato INTEL (*.HEX) a la memoria RAM de 32Kbytes.

Por ejemplo:

>CARGA 16Kb (ENTER)

mensaje: **Listo para cargar un archivo .HEX**

Presionando la tecla **F6** el programa de comunicaciones estará listo para recibir el nombre del archivo y su extensión.

FILE TO DOWNLOAD: display (nombre del archivo)**.hex (ENTER)**

Cuando termina la descarga del programa del usuario a la memoria RAM, se observa el mensaje: **CARGA EXITOSA.**

Colocando el cable plano en el cabezal doole **J2** de la tarjeta y el otro extremo (base) se coloca en el lugar destinado para emular la memoria EPROM.

Seleccione la capacidad de la memoria a Emular, esto se realiza por medio de las tres primeras posiciones del switch **S2**, de acuerdo a la siguiente tabla:

	2716(2Kb)	2732(4Kb)	2764 (8Kb)	27128 (16Kb)	27256 (32Kb)
A11	OFF	ON	ON	**ON**	ON
A13	OFF	OFF	OFF	**ON**	ON
A14	OFF	OFF	OFF	**OFF**	ON

13. Se procede a alimentar al huésped para llevar a cabo la emulación.

El sistema mínimo o huésped se puede conectar a **J9** (ver diagrama 1). J9 proporciona 5 volts.

14. Cuando el sistema funge como TARJETA DE EVALUACION. Se tiene disponible la memoria RAM de 8Kbytes.

El comando LOAD [C000] Carga un archivo de formato MOTOROLA en la memoria de 8Kbytes.

Ejemplo:

LOAD C000 (ENTER)

Mensaje: **LISTO PARA CARGAR UN ARCHIVO .S19**

Teclee: **F6**

FILE TO DOWNLOAD: prueba2.s19 (ENTER)

Mensaje: **CARGA EXITOSA**

15. El comando LIMPIA llenara la memoria RAM de 32Kbytes con FF.

16. Al utilizar el comando MUESTRA <DIR INICIO> <DIR FINAL> se debe proporcionare por lo menos la dirección de inicio, mostrando los 90 primeros caracteres. Cuando se le da la dirección de inicio y dirección final mostrara el contenido en esas direcciones, se detiene la ejecución con CONTROL W continuando la ejecución con cualquier tecla y se aborta con CONTROL X.

17. Al comando LLENA <DIR INICIO> <DIR FINAL> [DATO] se le debe proporcionar la dirección de inicio y final además del dato a almacenar. Al estar almacenando el dato en memoria el sistema presenta mensaje de espera: **Almacenando Dato.**
18. El sistema cuenta con un reset manual S1(ver diagrama1), en caso de que el sistema tenga un mal funcionamiento.

APÉNDICE C

Programa en formato OPAL

PROGRAMA EN FORMATO OPAL PARA EL DISPOSITIVO GAL22V10

```
Begin header
        Decodificador de memorias para el sistema con el MC68HC11E1

                                Miriam Cuevas Cuevas
End header


Begin definition
  Device GAL22V10;

  Inputs     E=1, A13=3, A14=4, A15=5, RW=6, BAN=8, AS=10;

  Outputs /WR=16, /RD=17, /CS_5=19, /CS_4=20, /CS_3=21;
  Outputs /CS_2=22, /CS_1=23;
End definition


Begin EQUATIONS

 WR = /RW & E;

 RD = RW & E;

 CS_5 = A14 & /A15 & E + /A14 & A15 & E

 CS_4 = BAN & /AS & E

 CS_3 = A14 & /A15 & E + /A14 & A15 & E

 CS_2 = /A13 & A14 & A15 & E

 CS_1 = A13 & A14 & A15 & E

End EQUATIONS
```

APÉNDICE D

Listado del programa

```
*=============================================================
*
*             PROGRAMA    E M U L A D O R
*
*=============================================================
*-------------------------------------------------------------
* Definiciones de Registros del Microcontrolador y
* puertos empleados
*
REGST    EQU  $1000       *Inicio de registros
BAUD     EQU  REGST+$2B   *Registro BAUD de SCI
SCCR1    EQU  REGST+$2C   *Primer reg. de control de com. serial
SCCR2    EQU  REGST+$2D   *Segundo reg de control de SCI
SCSR     EQU  REGST+$2E   *Reg de estado de SCI
SCDR    EQU  REGST+$2F   *Reg de datos de SCI
BPROT    EQU  REGST+$35   *Registro del bloque de protección
COPRST   EQU  REGST+$3A   *Reg de RESET cop
RAM      EQU  $4000       *Inicio de la RAM
BIOS     EQU  $E000       *Inicio del BIOS
PROMPT   EQU  '>'
BUFFLNG  EQU  35
CTLW     EQU  $17         *Pausa
CTLX     EQU  $18         *abortar
SWI      EQU  $3F
*-------------------------------------------------------------
*-------------------------------------------------------------
* Localidades reservadas en RAM
*-------------------------------------------------------------
         ORG  $33
INISTACK   RMB  30
STACK    RMB  1
REGS     RMB  9           *contador de programa
SP       RMB  2           *Registro de pila (sp)
ENTCAR   RMB  BUFFLNG     *Entrada del carácter
COMANDO  RMB  8           *Comando
GUARDAD  RMB  2           *Entrada de una cadena de datos
STREE    RMB  2           *Inicio de direcciones del eeprom
ENDEE    RMB  2           *Fin de direcciones del eeprom
TABLA    RMB  8           *Tabla
CNTCAR   RMB  1           *Contador de caracteres
CNTCOLM  RMB  1           *Contador de columna
PTRMEM   RMB  2           *locación de memoria actual
LDOFFST  RMB  2           *offset para descargar
*-------------------------------------------------------------
*-------------------------------------------------------------
* Variables de memoria de sistema
*
PTR0     RMB  2      *inicio, leecar, incapun
PTR1     RMB  2      *inicio,muestra,limpia
PTR2     RMB  2      *muestra,limpia
PTR3     RMB  2      *Muestra
TMP1     RMB  1      *inicio,hexbin,arghex
TMP2     RMB  1      *load,llena
TMP3     RMB  1      *load
TMP4     RMB  1
*-------------------------------------------------------------
*-------------------------------------------------------------
* Tabla de los pseudovectores de interrupción
*
JSCI     RMB   3
JSPI     RMB   3
```

```
JPAIE   RMB   3
JPAO    RMB   3
JTOF    RMB   3
JTOC5   RMB   3
JTOC4   RMB   3
JTOC3   RMB   3
JTOC2   RMB   3
JTOC1   RMB   3
JTIC3   RMB   3
JTIC2   RMB   3
JTIC1   RMB   3
JRTI    RMB   3
JIRQ    RMB   3
JXIRQ   RMB   3
JSWI    RMB   3
JILLOP  RMB   3
JCOP    RMB   3
JCLM    RMB   3
*-----------------------------------------------------------------
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*
*       PROGRAMA  PRINCIPAL
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ORG  BIOS
RESET   LDS  #STACK      *Inicializa pila
        LDAA #$00
        LDX  #$0000      *Envía por default el offset de descarga
        STX  LDOFFST
        JSR  INIVECT     *Inicializa pseudovectores
        LDX  #INISTACK
        STX  SP          *por default
        JSR  INISCI      *inicializa SCI
        LDX  #MSG1       *mensaje de presentación
        JSR  ESCMSG      *Envía mensaje a SCI
*       JSR  ENVCRLF     *Envía salto de línea y retorno de carro
        JSR  LIMPIA       *limpia memoria RAM en las dir 4000-bfff
        JSR  LIMPIA8     *limpia RAM de 8Kb (C000 - DFFF)
**.....................................  ..........................
**  INICIO - Lee la entrada del usuario y lo coloca ENTCAR.
**  El primer comando es puesto en COMANDO y es buscado en
**  la tabla de comandos, si lo encuentra ejecuta la
**  rutina asignada a este comando, regresa después de ejecutarla
**.....................................................................
INICIO    LDS  #STACK    *inicializa sp
          JSR  ENVCRLF
          LDAA #PROMPT   *Envía prompt a SCI
          JSR  SALSCI
          CLRB
INICIO1   JSR  RETENCAR  *Lee un carácter desde SCI
          LDX  #ENTCAR   *Guarda información
          ABX            *suma b a X (puntero en el ENTCAR)
          CMPA #CTLX
          BEQ  INICIO    *salta si es CTL X
          CMPA #$08
          BNE  INICIO2   *salta si no es espacio en blanco
          DECB
          BLT  INICIO    *salta si ENTCAR esta vacio
          BRA  INICIO1
INICIO2   CMPA #$D
          BNE  INICIO3   *salta si no es CR
```

```
          TSTB
          BEQ  COMM0      *salta si ENTCAR esta vacio
          STAA ,X         *coloca a en el ENTCAR
          BRA  COMM0
INICIO3   STAA ,X         *coloca a en el ENTCAR
          INCB
          CMPB #BUFFLNG
          BLE  INICIO4    *salta si no es largo
          LDX  #MSG3      *"largo"
          JSR  ESCMSG
          BRA  INICIO
INICIO4   BRA INICIO1

*-----------------------------------------------------------------
*  Sintetiza la salida y ejecuta el campo de  comando
*-----------------------------------------------------------------
COMM0     EQU  *
          CLR  TMP1         *Limpia variable
          CLR  GUARDAD
          CLR  GUARDAD+1
          CLRB
          LDX  #ENTCAR      *ptrbuff[] = ENTCAR[]
          STX  PTR0
          JSR  LEECARNB     *encuentra el primer carácter
COMM1     EQU  *
          JSR  LEECAR       *Lee de ENTCAR (entcar)
          LDX  #COMANDO
          ABX
          JSR  MINMAY       *convierte a mayúscula
          STAA ,X           *coloca el comando de entrada en el entcar
          CMPA #$0D
          BEQ  BUSCOM       *salta si es CR
          JSR  BLANCO
          BEQ  BUSCOM       *salta si es espacio en blanco
          JSR  INCAPUN      *incrementa apuntador del entcar
          INCB
          CMPB #$8          *espacio en blanco
          BLE  COMM2
          LDX  #MSG3
          JSR  ESCMSG
          JMP  INICIO
COMM2     EQU  *
           TST  TMP1
          STAB CNTCAR
          JMP  INICIO
COMM3     LDX  #MSG5        *Escribe mensaje
          JSR  ESCMSG
          JMP  INICIO
*-----------------------------------------------------------------
*   Busca en la tabla de comandos.
*   COMANDO retiene el comando
*   a ser ejecutado y B = Num de caracteres del comando
*-----------------------------------------------------------------
BUSCOM   STAB CNTCAR       *tamaño del comando de entrada
         LDX  #TABLCOM     *Lee el contenido de la Tabla
         STX  PTR1         *apunta al siguiente
BUSCOM1  LDX  PTR1
         LDY  #COMANDO     *Busca comando (apunta al comando de ENTCAR)
         LDAB 0,X
         CMPB #$FF         *Fin de la Tabla
         BNE  BUSCOM2      *Salta si es diferente de cero
```

```
        LDX  #MSG6          *"comando no encontrado"
        JSR  ESCMSG
        JMP  INICIO
BUSCOM2 PSHX                *Lee la siguiente localidad de tabla
        ADDB #$3
        ABX
        STX  PTR1
        PULX
        CLRB                *Limpia
BUSCOML INCB                *ciclo de caracteres
        LDAA 1,X            *lee tabla
        CMPA 0,Y            *compara al COMANDO
        BNE  BUSCOM1        *intenta la siguiente entrada
        INX                 *mueve apuntadores
        INY
        CMPB CNTCAR
        BLT  BUSCOML        *se cicla cntcar
        LDX  PTR1
        DEX
        DEX
        LDX  0,X            *salta a las direcciones de la tabla
EXEC    JSR  0,X
        JMP  INICIO

*****************************************
*          SUBRUTINAS
*****************************************
*-------------------------------------------------------------
*  MINMAY(a) - Convierte a Mayúscula el contenido de A
*-------------------------------------------------------------
MINMAY   CMPA #'a'
         BLT  MINMAY1       *salta si < a
         CMPA #'z'
         BGT  MINMAY1       *salto si > z
         SUBA #$20          *convierte
MINMAY1  RTS

*-------------------------------------------------------------
*   HEXBIN(a) - Convierte el carácter ASCII de a en binario
*   y colocalo dentro GUARDAD.
*   Retorna el valor en tmp1 incrementando si a no es hex.
*-------------------------------------------------------------
HEXBIN  PSHA
        PSHB
        PSHX
        JSR  MINMAY     *convierte a mayúscula
        CMPA #'0'
        BLT  HEXNOT     *salta si a < $30
        CMPA #'9'
        BLE  HEXNMB     *salta si es 0-9
        CMPA #'A'
        BLT  HEXNOT     *salta si es $39> a <$41
        CMPA #'F'
        BGT  HEXNOT     *salta si a > $46
        ADDA #$9        *convierte $A-$F
HEXNMB  ANDA #$0F       *convierte a binario
        LDX  #GUARDAD
        LDAB #4
HEXSHFT ASL  1,X        *Se intercambian dos bytes hasta un
        ROL  0,X        *    bit de acarreo
        DECB
```

```
        BGT  HEXSHFT     *desplazan 4 veces
        ORAA 1,X
        STAA 1,X
        BRA  HEXRTS
HEXNOT  INC  TMP1        *indica que no es ASCII o hex
HEXRTS  PULX
        PULB
        PULA
        RTS
*-------------------------------------------------------------------
*  ARGHEX() - Construye un argumento HEX de la entrada del SCI *(ENTCAR).
*Los caracteres son convertidos a binario y colocados en GUARDAD
*hasta  no encontrar mas
*  Al salir GUARDAD retiene los últimos 4 dígitos leidos
*  CNTCAR retiene el número de dígitos leidos
*  ptrbuff apunta al primer carácter no hex y A retiene el primer
* carácter no hex
*-------------------------------------------------------------------
ARGHEX   CLR  TMP1        *indicador de no hex
         CLR  CNTCAR      *numero de digitos
         CLR  GUARDAD
         CLR  GUARDAD+1
         JSR  LEECARNB
ARGHEXLP JSR  LEECAR      *lee carácter
         JSR  HEXBIN
         TST  TMP1
         BNE  ARGHEXRTS   *salta si no es hex
         INC  CNTCAR
         JSR  INCAPUN     *mueve puntero del ENTCAR
         BRA  ARGHEXLP
ARGHEXRTS RTS
*-------------------------------------------------------------------
*    RETARDO  de 7.6 ms
*-------------------------------------------------------------------
RETARDO EQU  *            retardo de 7.6ms a E = 2MHz
        PSHX
        LDX  #$0A03
RETRD   DEX
        BNE  RETRD
        PULX
        CLR  PPROG
        RTS

*-------------------------------------------------------------------
*  LEECAR() -  Lee el carácter en ENTCAR
* y apunta a ptrbuff a A.  Regresa ptrbuff
* sin cambios
*-------------------------------------------------------------------
LEECAR   PSHX
         LDX  PTR0
         LDAA 0,X
         PULX
         RTS
*-------------------------------------------------------------------
*  INCAPUN(), DECAPUN() - Incrementa o decrementa
* ptrbuff.
*-------------------------------------------------------------------
INCAPUN  PSHX
         LDX  PTR0
         INX
         BRA  INCDEC
```

```
DECAPUN  PSHX
         LDX  PTR0
         DEX
INCDEC   STX  PTR0
         PULX
         RTS
*-------------------------------------------------------------------
*  LEECARNB() - Lee desde ENTCAR hasta encontrar un carácter
*  distinto de espacio en blanco(espacio, coma, tabulador)
*  ptrbuff regresa apuntando a el primer carácter y'
*  A retiene ese carácter.
*  LEECARNB compara A con $0D(CR), coloca códigos indicando esa comparación.
*-------------------------------------------------------------------
LEECARNB JSR  LEECAR      *lee carácter
         JSR  BLANCO
         BNE  LEECARNB1   *salta si no espacio en blanco
         JSR  INCAPUN     *mueve el apuntador
         BRA  LEECARNB    *cicla
LEECARNB1 CMPA #$0D
         RTS
*-------------------------------------------------------------------
*  BLANCO(a) - Regresa z=1
*  si a retiene un carácter de espacio en blanco
*  Regresa z=0 si no
*-------------------------------------------------------------------
BLANCO   CMPA #$2C       *coma
         BEQ  BLANCO1
         CMPA #$20       *espacio
         BEQ  BLANCO1
         CMPA #$09       *tabulador
BLANCO1  RTS

DCHEK1   RTS
*-------------------------------------------------------------------
*  CHECABRT() - si es un control x resetea el stack, regresa a INICIO.
* Si es control W entonces esta rutina se queda esperando hasta que
* otro carácter es leído.
*-------------------------------------------------------------------
CHECABRT  JSR  LEESCI
          BEQ  CHK4        *salta si no hay entrada
          CMPA #CTLW
          BNE  CHK2        *salta si no es CTL w
CHECABRT1 JSR  LEESCI
          BEQ  CHECABRT1   *salta si no hay entrada
CHK2      CMPA #CTLX
          BEQ  CHK3        *salta si es control x
CHK4      RTS
CHK3      JMP  INICIO      *abortar

*-------------------------------------------------------------------
*  INIVECT - Inicializa pseudovectores en RAM.
*  Todos los que no estén programados saltan a STOPVEC
*-------------------------------------------------------------------
INIVECT  LDX  #JSCI      *apunta al primer pseudovector en RAM
         LDY  #STOPVEC   *apunta a la rutina STOPVEC
         LDD  #$7E03     *A=JMP código B=offset
VECLOOP  CMPA 0,X
         BEQ  VECNEXT    *Si el vector ya esta en
         STAA 0,X        *coloca JMP
         STY  1,X        *a la rutina STOPVEC
VECNEXT  ABX             *Suma 3 para apuntar al sig. pseudovector
```

```
        CPX  #JCLM+3    *Bien?
        BNE  VECLOOP    *Si no,continua
        RTS
*---------------------------------------------------------------
*STOPVEC() Detiene. Se a generado una interrupción no prevista
*por nuestro BIOS
*---------------------------------------------------------------
STOPVEC LDAA #$50       *Habilita Stop
        TAP
        STOP             * estas perdido!  Apágala
        JMP  STOPVEC     * continúe por XIRQ

*////////////////////////////////////////////////////////////////
*                Comunicación con el mundo exterior.
*
*  Subrutinas Invocada: INISCI, LEESCI, y SALSCI
*
*  Datos de entrada:
*////////////////////////////////////////////////////////////////
*---------------------------------------------------------------
*   INISCI() - Inicializa el puerto serie a 9600 bauds
*  para un cristal de 8 MHz.
*---------------------------------------------------------------
INISCI  EQU  *
        LDAA #$30     *9600 BAUDS
        STAA BAUD
        LDAA #$00     *1 bit inicio,8 bits datos,1 bit paro y sin paridad
        STAA SCCR1
        LDAA #$0C     *Habilita TX y RX
        STAA SCCR2
        RTS
*---------------------------------------------------------------
*   LEESCI() - Lee desde SCI.  Retorna A = carácter o 0.
*   Esta rutina también deshabilita el cop.
*---------------------------------------------------------------
LEESCI  EQU  *
        PSHX
        LDAA #$55       * resetea COP
        STAA COPRST
        LDAA #$AA
        STAA COPRST
        LDAA SCSR       *lee el registro status
        ANDA #$20       *verifica si rdrf=1 (1=lleno)
        BEQ  LEESCI1    *salta si no hay datos
        LDAA SCDR      *Lee dato
        PSHA
        TPA            *Transfiere registro CC á A
        ANDA #%11111011      *mascara de paridad
        TAP
        PULA
LEESCI1 PULX
        RTS
*---------------------------------------------------------------
*   SALSCI() - Envía el contenido de A al puerto serie (SCI).
* CNTCOLM indica la posición de la columna en la salida.
* Este es incrementado cada vez que un carácter es enviado,
* y limpia cuando la subrutina ENVCRLF es llamada.
*---------------------------------------------------------------
SALSCI  EQU  *
        PSHA           *guarda registros
        PSHB
```

```
        PSHX
        JSR  ENVA      *escribe SCI
        PULX
        PULB
        PULA
        INC  CNTCOLM   *incrementa el contador de columna
        RTS

*-------------------------------------------------------------
*  ENVA() - Envía A a SCI.
*              CR  y LF son enviados como crlf.
*-------------------------------------------------------------
ENVA    BSR  ENVA2
        CMPA #$0D
        BNE  ENVA1
        LDAA #$0A        *si CR, envía lf
        BRA  ENVA2
ENVA1   CMPA #$0A
        BNE  ENVA3
        LDAA #$0D        *si lf, envía CR
ENVA2   LDAB SCSR        *lee  status
        BITB #$80
        BEQ  ENVA2       *ciclate hasta tdre=1
        ANDA #$7F        *mascara de paridad
        STAA SCDR        *enviar carácter
ENVA3   RTS
*-------------------------------------------------------------
*  DESPDER(), DESPIZQ(), RETA()
* Convierte A de binario a ASCII y envía. El contenido de A
* es modificado
*-------------------------------------------------------------
DESPIZQ  PSHB
        LOADB #$04
        LSRA             *cambia el dato a la derecha
        CMPB  #$04
        BNE DESPIZQ
        PULB
DESPDER  ANDA #$0F       *mascara a tope medio
        ADDA #$30        *convierte a ASCII
        CMPA #$39
        BLE  RETA        *salta si es 0-9
        ADDA #$07        *convierte a hex A-F
RETA    JSR  SALSCI      *carácter enviado
        RTS
*-------------------------------------------------------------
*  OBT1BX(x) - Convierte el byte de X a 2 caracteres ASCII
* y envíalos. Retorna X  apuntando al siguiente byte.
*-------------------------------------------------------------
OBT1BX   PSHA
        LDAA 0,X         *obtén dato en a
        PSHA             *guarda una copia
        BSR  DESPIZQ     *envía la mitad izquierda
        PULA             *restaura la copia
        BSR  DESPDER     *envía la mitad derecha
        PULA
        INX
        RTS
*-------------------------------------------------------------
*  ENV1BXE(x), ENV2BXE(x) - Envía 1 o 2 bytes de X seguidos por
* un espacio. Regresa X apuntando al siguiente byte.
*-------------------------------------------------------------
```

```
ENV2BXE  JSR  OBT1BX       *has el primer byte
ENV1BXE  JSR  OBT1BX       *has el siguiente byte
ENVESPC  LDAA #$20         *envía un espacio
         JSR  SALSCI
         RTS
*-------------------------------------------------------------
*  ENVCRLF() - Envía un retorno de carro ($0D)
* y un salto de línea (0A). regresa a = CR.
*-------------------------------------------------------------
ENVCRLF  LDAA #$0D       *CR
         JSR  SALSCI     *Envía
         LDAA #$00
         JSR  SALSCI     *Envía relleno
         LDAA #$0D
         CLR  CNTCOLM     *cero al contador de columna
         RTS
*-------------------------------------------------------------
*  ESCMSG(x) - Envía una cadena de bytes ASCII iniciando a X
* hasta fin de texto($04).
*Puede hacer una pausa por control w (continua con cualquier *carácter).
*-------------------------------------------------------------
ESCMSG   JSR  ENVCRLF
ESCMSG0 PSHA
ESCMSG1 LDAA 0,X         *lee carácter en a
        CMPA #$04
        BEQ  ESCMSG3     *salta si es $04
        JSR  SALSCI      *salida de carácter
        INX
        JSR  LEESCI
        BEQ  ESCMSG1     *salta si no es entrada
        CMPA #CTLW
        BNE  ESCMSG1     *salta si no es CTL W
ESCMSG2 JSR  LEESCI
         BEQ  ESCMSG2    *salta si es cualquier entrada
         BRA  ESCMSG1
ESCMSG3 PULA
         RTS
*-------------------------------------------------------------
* LIMPIA8()Limpia la memoria RAM de 8Kb, dato = 00.
*
* LIMPIA()Limpia la memoria RAM de 32Kbytes
* dato=FF
*-------------------------------------------------------------
LIMPIA8  EQU  *
         LDX  #$C000       *Inicio de RAM
         STX  PTR1
         LDX  #$DFFF        *Fin de RAM
         STX  PTR2
         LDAA #$00         *Por default el dato es 00
         STAA TMP2
         BRA LIMPIA1
LIMPIA   EQU  *
         JSR  ENVCRLF
         LDX  #MSG10       *Mensaje de espera
         JSR  ESCMSG
         LDX  #$4000       *Inicio de RAM
         STX  PTR1
         LDX  #$BFFF       *Fin de RAM
         STX  PTR2
         LDAA #$FF        *Por default el dato es FF
         STAA TMP2
```

```
LIMPIA1  JSR  CHECABRT     *checa para abortar
         LDX  PTR1         *direcciones de inicio
         LDAA TMP2         *dato
         STAA
         CMPA 0,X
         BNE  LIMPIABAD    *salta si no es ESCRIBE
         CPX  PTR2
         BEQ  LIMPIA2      *salir ya?
         INX
         STX  PTR1
         BRA  LIMPIA1      *ciclo
LIMPIA2  RTS

LIMPIABAD EQU  *
          LDX  #PTR1       *salida de direcciones erróneas
          JSR  ENV2BXE
         RTS
*-----------------------------------------------------------------
*  RETENCAR() - Lee la entrada hasta que el carácter es enviado.
*     Eco carácter y regresa cuando a = char.
*-----------------------------------------------------------------
RETENCAR JSR  LEESCI
         TSTA
         BEQ  RETENCAR      *salta si no es entrada
         JSR  SALSCI        *eco
         RTS
*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
*    *      *       TABLA DE COMANDOS              *         *
****  Verifica el comando y lo ejecuta si existe        ***
*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TABLCOM  EQU  *
         FCB  7
         FCC  'MUESTRA'
         FDB  MUESTRA
         FCB  5
         FCC  'CARGA'
         FDB  CARGA
         FCB  5
         FCC  'LLENA'
         FDB  LLENA
         FCB  5
         FCC  'AYUDA'
         FDB  AYUDA
         FCB  6
         FCC  'LIMPIA'
         FDB  LIMPIA
         FCB  4
         FCC  'LOAD'
         FDB  LOAD
         FCB  1
         FCC  '?'       *comando Inicial
         FDB  AYUDA
*
*** Comandos de ayuda ***
*
         FCB  2
         FCC  'LD'
         FDB  LOAD
         FCB  2
         FCC  'LM'
```

```
          FDB   LLENA
          FCB   2
          FCC   'MM'
          FDB   MUESTRA
          FCB   $FF

*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
**                    M E N S A J E S                   **
*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MSG1      FCC   '**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*'
          FCC   '***                                           ****'
          FCC   '****  EMULADOR  DE  MEMORIAS EPROMs           ****'
          FCC   '****                                                ****'
          FCC   '****                                                ****'
          FCC   '****   UNIVERSIDADTECNOLOGICADELAMIXTECA      ****'
          FCC   '****                                                ****'
          FCC   '****                                                ****'
          FCC '**** Miriam Cuevas   mcuevas@nuyoo.utm.mx       ****'
          FCC   '****                                          ****'
          FCC   '*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*'
          FCB   $0D
          FCC   ' Capacidades de memorias a Emular 2Kb,4Kb,8Kb,16Kb y 32Kb'
          FCB   $0D
          FCB   $04
MSG2      FCC   'Incorrecto'
          FCB   $04
MSG3      FCC   'Demasiado largo'
          FCB   $04
MSG4      FCC   'error RAM-'
          FCB   $04
MSG5      FCC   'Comando: ?'
          FCB   $04
MSG6      FCC   'argumento no valido'
          FCB   $0D
          FCC   ' Teclea AYUDA o ? '
          FCB   $04
MSG7      FCC   'Carga Exitosa'
          FCB   $04
MSG8      FCC   'ERROR de chksum (Chequeo)'
          FCB   $04
MSG9      FCC   'ERROR de transmisión'
          FCB   $04
MSG10     FCC   'Espere... Limpiando RAM'
          FCB   $04
MSG11     FCC   'Almacenando Dato...'
          FCB   $04
MSG12     FCC   'Listo para bajar archivo .s19'
          FCB   $04
MSG13     FCC   'Listo para bajar archivo .hex'
          FCB   $04
*-----------------------------------------------------------------------
*  BPBUSCOM() - Busca en la tabla de direcciones GUARDAD.
* Regresa b = indice de entrada
* o b = -1 si no lo es encontrado.
*-----------------------------------------------------------------------
BPBUSCOM  CLRB
BPBUSCOM1 LDX   #TABLA
          ABX
          LDX   0,X         *obtén tabla de entrada
          CPX   GUARDAD
          BNE   BPBUSCOM2   *salta si no es
```

```
          RTS
BPBUSCOM2 INCB
          INCB
          CMPB #$6
          BLE  BPBUSCOM1  *ciclo 4 veces
          LDAB #$FF
          RTS
*-----------------------------------------------------------------
*  MUESTRA <dir1> <dir2>  Muestra el contenido de la memoria
* desde <dir1> a <dir2>.
* Por default inicia en la dirección de "locación actual y el número
* de línea por default es 8.
*-----------------------------------------------------------------
MUESTRA   LDX  PTRMEM     *locación actual
          STX  PTR1       *inicio de default
          LDAB #$90
          ABX
          STX  PTR2       *fin de dirección por default
          JSR  LEECARNB
          BEQ  MUESTRA1   *salta si no hay argumentos
          JSR  ARGHEX     *lee argumento
          TST  CNTCAR
          BEQ  MUESTRAERR *salta si no hay argumento.
          JSR  BLANCO
          BEQ  DCHEK1     *salta si hay espacio en blanco
          CMPA #$0D
          BNE  MUESTRAERR *salta si es delimitador
          LDX  GUARDAD
          STX  PTR1
          LDAB #$90
          ABX
          STX  PTR2       *fin de direcciones por default
          JSR  LEECARNB
          BEQ  MUESTRA1   *salta si - argumento es 1
          JSR  ARGHEX     *lee argumento
          TST  CNTCAR
          BEQ  MUESTRAERR *salta si no hay argumento
          JSR  LEECARNB
          BNE  MUESTRAERR *salta si no es CR
          LDX  GUARDAD
          STX  PTR2
          BRA  MUESTRA1   *salta si hay 2 argumentos
MUESTRAERR  LDX  #MSG6    *"argumento incorrecto"
            JSR  ESCMSG
            RTS

MUESTRA1  LDD  PTR1
          STD  PTRMEM     *locación actual nueva
          ANDB #$F0
          STD  PTR1       *inicia MUESTRA a un limite de 16 byte

*** MUESTRA 16 bits  ***

MUESTRALP   JSR  ENVCRLF
            LDX  #PTR1
            JSR  ENV2BXE   *primera dirección
            LDX  PTR1      *direcciones base
            CLRB           *contador de ciclos
MUESTRADAT  JSR  ENV1BXE   *valor hex ciclado
            INCB
            CMPB #$10
```

```
            BLT  MUESTRADAT     cicla 16 veces
*Salida en ASCII
* 16 veces
* solo caracteres  ($7A < a < $20)

            CLRB              *contador de ciclos
MUESTRAASC  LDX  PTR1         *direcciones base
            ABX
            LDAA ,X           *valor ASCII de ciclos
            CMPA #$20
            BLO  MUESTRA3      *salta si no es imprimible
            CMPA #$7A
            BLS  MUESTRA4      *salta si es imprimible
MUESTRA3    LDAA #$20          *espacio para los no imprimibles
MUESTRA4    JSR  SALSCI        *salida del valor ASCII
            INCB
            CMPB #$10
            BLT  MUESTRAASC   *hazlo 16 veces

*      checa CTL W o CTL X
*      ptr1 = ptr1 + $10;
*mientras ptr1 <= ptr2 imprime

          JSR  CHECABRT    *checa abortar o esperar
          LDD  PTR1
          ADDD #$10        *puntero a los 16 byte de limite
          STD  PTR1        *actualiza ptr1
          CPD  PTR2
          BHI  MUESTRA5    *sal si ptr1 > ptr2
          CPD  #$00        *checa con $ffff
          BNE  MUESTRALP   *salta si no es
          LDD  PTR2
          CPD  #$FFF0
          BLO  MUESTRALP    *máximo limite para imprimir
MUESTRA5  RTS               *salir
*-------------------------------------------------------------------
*  LLENA <dir1> <dir2> [<dato>]
*  Llena el bloque de memoria desde dir1 a dir2 con
*  datos. Los Datos por default son $FF.
*-------------------------------------------------------------------
LLENA     EQU  *
          JSR  LEECARNB    *Lee carácter que no sea espacio en blanco
          JSR  ARGHEX      *Convertir a Binario
          TST  CNTCAR      *contador de carácter
          BEQ  LLENAERR    *salta si no hay argumento
          JSR  BLANCO
          BNE  LLENAERR    *salta si el argumento es incorrecto
          LDX  GUARDAD
          STX  PTR1        *direccion1
          JSR  LEECARNB    *Lee carácter no blanco
          JSR  ARGHEX
          TST  CNTCAR
          BEQ  LLENAERR    *salta si no es argumento
          JSR  BLANCO
          BEQ  DCHEK1      *salta si hay espacio en blanco
          CMPA #$0D
          BNE  LLENAERR    *salta si el argumento es incorrecto
          LDX  GUARDAD
          STX  PTR2        *direccion2
          LDAA #$FF        *Por default el dato es FF
          STAA TMP2
```

```
        JSR  LEECARNB    *Lee carácter no blanco
        BEQ  LLENA1      *salta si es el dato por default
        JSR  ARGHEX
        TST  CNTCAR
        BEQ  LLENAERR    *salta si no hay argumento
        JSR  LEECARNB
        BNE  LLENAERR    *salta si el argumento es incorrecto
        LDAA GUARDAD+1   *Guarda dato
        STAA TMP2
*almacena dato hasta (ptr1 = ptr2)
LLENA1  EQU  *
        LDX  #MSG11      *Mensaje de espera
        JSR  ESCMSG
LLENA11 JSR  CHECABRT    *checa para abortar
        LDX  PTR1        *direcciones de inicio
        LDAA TMP2        *dato
        STAA
        CMPA 0,X
        BNE  LLENABAD    *salta si no es ESCRIBE
        CPX  PTR2
        BEQ  LLENA2      *salir ya?
        INX
        STX  PTR1
        BRA  LLENA11     *ciclo
LLENA2   RTS

LLENAERR LDX  #MSG6      *"argumento incorrecto"
         JSR  ESCMSG
         RTS
LLENABAD EQU  *
        LDX  #PTR1       *salida de direcciones erróneas
        JSR  ENV2BXE
        RTS
*-------------------------------------------------------------------
*  LEEENT() --- Lee la entrada desde la terminal en Entcar
* hasta que un carácter de comando es leído (cr,lf). Si mas
* caracteres son escritos en Entcar serán retenidos,
* los caracteres extras son sobrescritos al final.
* Al salir b = número del carácter leído a=0 para salir, en otro
* caso a = siguiente comando.
*-------------------------------------------------------------------
LEEENT  CLRB
        LDAA #$0D      *retorno de carro
LEE0    LDX  #ENTCAR
        ABX
        STAA 0,X       *inicializa el entcar de entrada
        INCB
        CMPB #BUFFLNG
        BLT  LEE0
        CLRB
LEE1    JSR  RETENCAR
        CMPA #CTLX      *Control X
        BEQ  LEEQUIT
        CMPA #$08       *backspace
        BNE  LEE2
        DECB
        BGT  LEE1
        BRA  LEEENT
LEE2    LDX  #ENTCAR
        ABX
        JSR  MINMAY
```

```
        STAA 0,X        *coloca el carácter en el ENTCAR
        CMPB #BUFFLNG   *máximo tamaño del entcar
        BGE  LEE3       *salta si el entcar esta lleno
        INCB            *mueve el puntero del entcar
LEE3    JSR  ASSCHEK    *checa por un subcomando
        BNE  LEE1
        RTS
LEEQUIT CLRA            *salir
        RTS             *regresar
*-------------------------------------------------------------
*  asschek() - Hace la comparación entre lf, CR
*-------------------------------------------------------------
ASSCHEK CMPA #$0A     *salto de línea
        BEQ  ASSCHK1
        CMPA #$0D     *retorno de carro
        BEQ  ASSCHK1
ASSCHK1 RTS
*-----------------------------------------------------------------
*   AYUDA  -  Lista de comandos.
*-----------------------------------------------------------------
AYUDA    EQU  *
         LDX  #AYDMSG1
         JSR  ESCMSG     *escribe mensaje
         RTS

AYDMSG1 EQU  *
         FCC  'LIMPIA  Limpia la memoria RAM '
         FCB  $0D
FCC  ' LOAD [<capacidad de la memoria>]  Carga archivo .s19 en RAM de 32Kb'
         FCB  $0D
FCC  ' LOAD [Dir. de inicio <C000>]Carga archivo .s19 en RAM de 8Kb '
         FCB  $0D
FCC  ' CARGA [<capacidad de la memoria>] Carga archivo .HEX en RAM de 32Kb'
         FCB  $0D
         FCC  ' LLENA <dir1> <dir2> [<dato>] Llena Memoria con Dato'
         FCB  $0D
         FCC  ' MUESTRA <dir1> <dir2>   Muestra el contenido de Memoria'
         FCB  $0D
         FCC  '     [CTL W]    Pausa '
         FCB  $0D
         FCC  '     [CTL X]    Abortar'
         FCB  $0D
         FCC  '     [CR]       Repite el ultimo comando'
         FCB  $0D
         FCC  ' AYUDA o ?      Muestra el menú '
         FCB  $0D
        FCB  4
*-----------------------------------------------------------------
* LOAD(ptrbuff[]) - [capacidad de la memoria] Carga records s1/s9 a
* la memoria externa de 32Kb
* Ptrbuff[] apunta a la cadena en ENTCAR de entrada  lo cual
* es un comando para enviar records s1/s9 desde SCI
* Regresa error y la dirección si no puede escribir a
* una locación en particular.
* load C000 - Carga records s1/s9 a la memoria externa de 8Kb
*-----------------------------------------------------------------
* (ptrbuff[]) - Verifica la memoria del comando de carga.
*  Ptrbuff[] es el mismo para toda la carga.
* tmp3 es usado como un indicador de error, 0=no errores,
* 1=receiver, 2=rom error, 3=checksum error.
**********
```

```
LOAD    CLR  TMP2       * 0=load
        JSR LOAD1
        RTS

OFFST1  LDX #$5000       *offset para 2Kb
        STX  LDOFFST
        BRA SIGUE
OFFST2  LDX #$5000       *offset de 4Kb
        STX  LDOFFST
        BRA SIGUE
OFFST3  LDX #$4000       *offset de 8Kb
        STX  LDOFFST
        BRA SIGUE
OFFST4  LDX #$4000       *offset de 16Kb
        STX  LDOFFST
        BRA SIGUE
LOAD8  LDX #$0000         *offset
       STX  LDOFFST
       BRA  SIGUE


ERR LDX  #MSG6           *"argumento incorrecto"
        JSR  ESCMSG
        RTS
LOAD1   CLR  TMP3        *limpia bandera de error
        JSR  LEECARNB    *Lee carácter que no sea espacio en blanco
        JSR  ARGHEX      *Convierte a Binario
        TST  CNTCAR      *contador de carácter
        BEQ  ERR         *salta si no hay argumento
        JSR  BLANCO
        BEQ  ERR         *salta si el argumento es incorrecto
        LDX  GUARDAD
        STX  PTR1        *dirección

        CPX  #$C000      *Carga a la memoria RAM de 8Kb
        BEQ  LOAD8

        CPX  #$2KB      * F800 (2Kb)
        BEQ  OFFST1
        CPX  #$4KB      * F000 (4Kb)
        BEQ  OFFST2
        CPX  #$8KB      * E000 (8Kb)
        BEQ  OFFST3
        CPX  #$16KB     * C000 (16Kb)
        BEQ  OFFST4
        CPX  #$32KB     * 8000 (32Kb)
        BNE  LOAD3       *salta no encuentra  opción
        LDX  #$4000      *offset de 32Kb
        STX  LDOFFST
SIGUE   JSR  CAPAMEM
        CMPA #$0D
        BNE  LOAD3       *salta si no existe opción
        BRA  LOADS      *Espera si records

LOAD3     LDX  #MSG2     *Mensaje de error
          JSR  ESCMSG
LOAD4     JSR  LEECAR    *Obtén el siguiente carácter
          JSR  INCAPUN
          CMPA #$0D
          BNE  LOAD4     *salta si no es CR
          RTS
```

```
*Busca archivo .s19 y compara S1, S0, S9
LOADS     EQU  *
          JSR  CHECABRT  *verifica si hay aborto
          JSR  LEESCI    *lee  host
          TSTA
          BEQ  LOADS     *salta si no hay Entrada
          CMPA #'S'
          BNE  LOADS     *salta si no es S
LOADSS    JSR  LEESCI    *lee  host
          TSTA
          BEQ  LOADSS    *Salta si no hay entrada
          CMPA #'9'
          BEQ  LOAD9     *salta si es S9
          CMPA #'1'
          BNE  LOADS     *salta si no es S1
          CLR  TMP4      *limpia checksum

** Obtén contador de Byte e inicializa direcciones
          JSR  BYTE
          LDAB GUARDAD+1
          SUBB #$2       *b = contador de byte
          JSR  BYTE
          JSR  BYTE
          PSHB           *Guarda contador ( byte )
          LDD  GUARDAD
          SUBD LDOFFST   *suma el offset
          XGDX           *x = direc-offset
          PULB           *restaura el contador de byte
          DEX            *condición para ciclar

** Obtén y almacena el byte de dato de Entrada
LOAD11    JSR  BYTE      *Obtén el siguiente byte
          INX
          DECB           *checa el contador de byte
          BEQ  LOADCHCK    *si b=0, ir a checksum
          TST  TMP3
          BNE  LOADS     *salta si hay  error de bandera
          TST  TMP2
          BNE  LOADER    *salta si es verifica
          LDAA GUARDAD+1
          STAA
LOADER    CMPA 0,X       *verifica la locación de RAM
          BEQ  LOAD11    *salta si la RAM esta bien
          LDAA #$02
          STAA TMP3      *indica error de ROM
          STX  PTR3      *salva las direcciones de error
          BRA  LOAD11    *Termina de descargar

** Obtén y prueba el Checksum
LOADCHCK  TST  TMP3
          BNE  LOADS     *salta si hay error
          LDAA TMP4
          INCA           *has el chequeo (checksum)
          BEQ  LOADS     *salta si s1 esta bien
          LDAA #$03
          STAA TMP3      *indica error de checksum
          BRA  LOADS

*       a = '9'
LOAD9     JSR  BYTE
```

```
          LDAB GUARDAD+1  *b = contador de byte
LOAD91    JSR  BYTE
          DECB
          BNE  LOAD91     *cicla hasta fin de record
          LDAB #$64
LOAD91A   JSR  RETARDO    *retardo de 1 seg -dejar que termine el host
          DECB
          BNE  LOAD91A
          JSR  LEESCI     *limpia dispositivo comm
          LDX  #MSG7      *mensaje de descarga exitosa
          LDAA TMP3
          CMPA #$02
          BNE  LOAD92     *salta si no es error de RAM
          LDX  #PTR3
          JSR  ENV2BXE    *Error de direcciones de RAM
          BRA  LOAD95

LOAD92    CMPA #$01
          BNE  LOAD93     *salta si no hay error transmisión
          LDX  #MSG9      *"error transmisión  "
          BRA  LOAD94
LOAD93    CMPA #$03
          BNE  LOAD94     *salta si no hay error de checksum
          LDX  #MSG8      *"ERROR checksum "
LOAD94    JSR  ESCMSG
LOAD95    RTS

*-------------------------------------------------
*  byte() -  Lee 2 bytes ASCII del host y
*lo convierte  a un byte hexadecimal. Regresa
*el byte cambiado en GUARDAD y lo suma a tmp4.
*-------------------------------------------------
BYTE      PSHB
          PSHX
BYTE0     JSR  LEESCI     *lee el 1er byte
          TSTA
          BEQ  BYTE0      *Se cicla hasta una entrada
          JSR  HEXBIN
BYTE1     JSR  LEESCI     *lee el 2do byte
          TSTA
          BEQ  BYTE1      *repite hasta una entrada
          JSR  HEXBIN
          LDAA GUARDAD+1
          ADDA TMP4
          STAA TMP4       *Se lo suma al checksum
          PULX
          PULB
          RTS
CAPAMEM   EQU *
          LDX   #MSG12    *Mensaje de selección de memoria
          JSR   ESCMSG      *Envía mensaje
          RTS
CAPMEM    EQU *
          LDX   #MSG13    *Mensaje de selección de memoria
          JSR   ESCMSG      *Envía mensaje
          RTS
*************************************************************
*CARGA(ptrbuff[]) [capacidad de la memoria] Carga records ":" *archivos
.hex) a la memoria externa de 32Kbytes.
*************************************************************
CARGA     CLR  TMP2       * 0=CARGA
```

```
          JSR CARGA1
          RTS
CAROFF1 LDX #$A800        *offset para 2Kb
        STX LDOFFST
        BRA SIGUE1
CAROFF2 LDX #$A000        *offset de 4Kb
        STX  LDOFFST
        BRA SIGUE1

CAROFF4 LDX #$8000        *offset de 16Kb
        STX  LDOFFST
        BRA SIGUE1

ERR1  LDX  #MSG6          *"argumento incorrecto"
      JSR  ESCMSG
      RTS

CARGA1   EQU  *
         CLR  TMP3        *limpia bandera de error
         JSR  LEECARNB    *Lee carácter que no sea espacio en blanco
         JSR  ARGHEX      *Convertir a Binario
         TST  CNTCAR      *contador de carácter
         BEQ  ERR1        *salta si no hay argumento
         JSR  BLANCO
         BEQ  ERR1        *salta si el argumento es incorrecto
         LDX  GUARDAD
         STX  PTR1        *dirección
         CPX  #$2KB      * 2Kb
         BEQ  CAROFF1
         CPX  #$4KB      * 4Kb
         BEQ  CAROFF2
         CPX  #$8KB      * 8Kb
         BEQ  CAROFF2
         CPX  #$16KB     * 16Kb
         BEQ  CAROFF4
         CPX  #$32KB     * 32Kb
         BNE  CARGA3     *salta no encuentra  opción
         LDX #$4000       *offset de 32Kb
         STX  LDOFFST
SIGUE1   JSR  CAPMEM
         CMPA #$0D
         BNE  CARGA3      *salta si no existe opción
         BRA  CARGA10    'Espera  records
CARGA3   LDX  #MSG2     *Mensaje de error
         JSR  ESCMSG
CARGA4     JSR  LEECAR   *Obtén el siguiente carácter
           JSR  INCAPUN
           CMPA #$0D
           BNE  CARGA4    *salta si no es CR
           RTS

*Busca archivo .HEX y compara:
CARGA10    EQU  *
           JSR  CHECABRT  *verifica si hay aborto
CARGA11    JSR  LEESCI    *lee  host
           TSTA
           BEQ  CARGA10   *salta si no hay Entrada
           CMPA #':'
           BNE  CARGA10   *salta si no es :

** Obtén contador de Byte e inicializa direcciones
```

```
            JSR  BYTE
            LDAB GUARDAD+1
            CMPB #$0
            BEQ  FINHEX
            INCB
            JSR  BYTE
            JSR  BYTE
            PSHB              *Guarda contador ( byte )
            LDD  GUARDAD
            ADDD LDOFFST      *suma el offset
            XGDX              *x = dirección + offset
            PULB              *restaura el contador de byte
            DEX               *condición para ciclar
            JSR  BYTE

** Obtén y almacena el byte de dato de Entrada
CARGA20     JSR  BYTE         *Obtén el siguiente byte
            INX
            DECB              *checa el contador de byte
            BEQ  CARGA10       *si b=0, ir a checksum
            LDAA GUARDAD+1
            STAA
CARGA21     CMPA 0,X          *verifica la locación de RAM
            BEQ  CARGA20       *salta si la RAM esta bien
            LDAA #$02
            STAA TMP3         *indica error de RAM
            STX  PTR3         *salva las direcciones de error
            BRA  CARGA20       *Termina de descargar

* lee el resto de record
FINHEX      JSR  BYTE
            JSR  BYTE
            JSR  BYTE
            JSR  BYTE
            JSR  BYTE
            LDAB #$64
CARGA91A    JSR  RETARDO      *retardo de 1 seg - dejar que termine el host
            DECB
            BNE  CARGA91A
            LDX  #MSG7        *mensaje de descarga exitosa
            LDAA TMP3
            CMPA #$02
            BNE  CARGA92       *salta si no es error de RAM
            LDX  #PTR3
            JSR  ENV2BXE      *Error de direcciones de RAM
            BRA  CARGA95

CARGA92     CMPA #$01
            BNE  CARGA93       *salta si no hay error transmisión
            LDX  #MSG9        *"error transmisión "
            BRA  CARGA94
CARGA93      CMPA #$03
            BNE  CARGA94       *salta si no hay error de checksum
            LDX  #MSG8        *"ERROR checksum "
CARGA94      JSR  ESCMSG
CARGA95      RTS

*** Tabla de saltos***
        ORG     BIOS+$1F00
        JMP     INICIO
        JMP     HEXBIN
```

```
        JMP     ARGHEX
        JMP     LEECAR
        JMP     LEECARNB
        JMP     BLANCO
        JMP     INISCI
        JMP     LEESCI
        JMP     SALSCI
        JMP     RETA
        JMP     OBT1BX
        JMP     ENV1BXE
        JMP     ENV2BXE
        JMP     ESCMSG
        JMP     ESCMSG0
        JMP     RETENCAR
        JMP     INIVECT  *inicializa la tabla de vectores
         ORG    BIOS+$1FD6
*** Vectores ***
         FDB    JSCI
         FDB    JSPI
         FDB    JPAI
         FDB    JPAO
         FDB    JTOF
         FDB    JTOC5
         FDB    JTOC4
         FDB    JTOC3
         FDB    JTOC2
         FDB    JTOC1
         FDB    JTIC3
         FDB    JTIC2
         FDB    JTIC1
         FDB    JRTI
         FDB    JIRQ
         FDB    JXIRQ
         FDB    JSWI
         FDB    RESET
         FDB    RESET
         FDB    RESET
         FDB    RESET
         END
```

GLOSARIO

GLOSARIO DE TÉRMINOS TÉCNICOS

ALTA IMPEDANCIA. Se presenta cuando el circuito presenta muy alta resistencia tanto en las terminales de entrada como en las de salida. Esto hace que para fines prácticos, el circuito "desaparece" del sistema y no interactúa con los demás dispositivos.

ASÍNCRONO. No está sincronizado por un reloj.

BAUD. Término comúnmente empleado para indicar bits por segundo. En rigor, es el número de elementos de señal por segundo.

BIOS. Sistema básico de entrada/salida (*"Basic Input Output System"*). Contiene un conjunto de instrucciones que se encargan de establecer las condiciones de comunicación y control entre el microcontrolador, dispositivos de soporte y periféricos.

BIT. Abreviatura de dígito binario. Puede tener un valor de cero o uno.

BUS. Conjunto de líneas que transportan información de las mismas características, por ejemplo, el bus de direcciones, bus de datos, bus de control.

BUFFER. Dispositivo cuya función es la de aislar y proporcionar una mayor cantidad de corriente, a una señal determinada.

BYTE. Grupo de 8 bits. Estos bits tienen 255 posibilidades combinacionales, lo cual da 255 posibles bytes. Cada una de estas combinaciones representa un símbolo o carácter empleado por el computador.

CMOS *"Complementary Metal Oxide Semiconductors"*. Dispositivo con baja disipación de potencia y capacidad de operar en una amplia gama de voltaje.

CÓDIGO FUENTE. Forma legible para un ser humano. La entrada a un compilador o ensamblador.

COP (*Computer Operating Properly*, computadora Operando Apropiadamente). Sistema empleado en el circuito MCHC11 para determinar - por ausencia de atención - algún mal funcionamiento de programación en el sistema del microcontrolador.

CRISTAL. Dispositivo de cuarzo, en el cual se aprovecha su propiedad piezoeléctrica para obtener una frecuencia (oscilación) exacta. La oscilación permite la conducción intermitente de energía (corriente) produciéndose pulsos. Esta frecuencia se aplica para la temporización como reloj del sistema.

DIRECCIÓN Numero que identifica la localidad de una memoria. Cada palabra almacenada en un dispositivo de memoria tiene una dirección única. Las direcciones se especifican con un numero binario, aunque algunas veces se utilizan números octales, hexadecimales, y decimal por conveniencia.

ENSAMBLADOR. Programa que convierte un lenguaje de máquina en forma nemotécnica en un formato objeto binario para que pueda ser ejecutado por la computadora.

ESPACIO DIRECCIONABLE La cantidad de memoria que un procesador puede direccionar sin "hardware" adicional.

EPROM. (*Erasable Programmable Read Only Memory*, Memoria Programable de Solamente Lectura) Memoria construida por celdas flotantes que permiten guardar información binaria. Para llevar a este dispositivo a una condición de "borrado" es necesario exponerlo (a través de una ventana de cuarzo dispuesta para ese fin) a una dosis de luz ultravioleta. El esquema de grabación debe regirse por un algoritmo específico y de manera habitual se emplea alguna terminal sujeta a un voltaje relativamente alto (12 a 21 volts).

EXTAL (*External Clock Input*): es una entrada de reloj externa, esta entrada de reloj controlará el generador de reloj interno del microcontrolador.

GAL. *"Generic Array Logic"*. Es un circuito integrado constituido por arreglos de compuertas, las cuales son susceptibles de ser conectadas o desconectadas por ciertos puntos según lo

decidamos. Son capaces de representar en forma directa una expresión booleana expresada en suma de productos.

H. HEXADECIMAL. Símbolo que se refiere a un número hexadecimal (hex), por ejemplo, 05H tiene un valor numérico de 5, mientras que 0BH tiene un valor de 11.

HARDWARE. Conjunto físico de dispositivos (discretos como resistores, transistores, etc., e integrados como lógica TTL, HCMOS. Etc.) que conforman la arquitectura de la microcomputadora.

INPUT/OUTPUT (Entradas/Salidas). Referido principalmente a terminales ó buses relacionados con la introducción ó expulsión de información a un sistema de procesamiento de información.

JUMPER (saltador) Es un puente conductor utilizado para unir eléctricamente dos puntos de un circuito eléctrico. Regularmente es utilizado para elegir parámetros de configuración o como método rápido de corrección de diseños.

LATCH Dispositivo que sirve para almacenar temporalmente la información.

MCU Unidad Microcontroladora se le llama así en forma común a los circuitos

MICROCOMPUTADOR Un computador completo de propósito general que utiliza tecnología de microprocesadores. El computador se puede implementar con un solo circuito integrado o puede requerir varios componentes conectados entre si.

MICROPROCESADOR. Unidad central de procesamiento (CPU) de un microcomputador o un microcontrolador implementado en una sola pastilla de silicio.

MICROCONTROLADORES. Son el resultado de la evolución de los microprocesadores hacia la especialización de funciones. Es decir un microcontrolador esta compuesto por un circuito microprocesador, además de memorias y manejadores de puertos que permiten la adquisición de información y comunicaciones.

OPERACIÓN DE LECTURA La operación en la cual la palabra binaria almacenada en una localidad (dirección) especifica de la memoria es captada y después transferida a otro dispositivo.

OPERACIÓN DE ESCRITURA Operación por medio del cual se coloca una nueva palabra en cierta localidad de memoria. También se le llama operación de almacenar.

PROGRAMAS DE APLICACIÓN. Estos programas los escriben los usuarios para resolver problemas específicos, cálculos, etc.

RAM (*Random Access Memory, Memoria de Acceso Aleatorio*). Memoria que pierde la información almacenada al retirarle la alimentación.

REGISTROS. Son áreas donde se guardan datos e instrucciones para que el microcontrolador las procese.

RESET (Restablecimiento o reinicialización) Es la acción de devolver un dispositivo almacenador de información a un estado prescrito.

Al generarse esta señal de inicialización en el MC68HC11, ya sea externamente ó a partir de alguno de los subsistemas de protección -COP, Falla de Reloj, etc.- se ajustan a valores predeterminados el contador de programa (PC), algunos registros y vectores internos.

RS-232. Interfaz industrial estándar para la transferencia de datos

SCI (*Serial Communication Interface*, Interface de comunicación Serial) Subsistema del circuito MC68HC11 que permite la comunicación seria asíncrona con otros dispositivos. Consta de una línea de entrada serie y una línea de salida serie, ambas con niveles de voltaje compatibles con el estándar TTL. Emplea el esquema NRZ, se puede ajustar - por programación, ajustando valores en algunos registros -, la velocidad de transmisión, número de bits transmitidos por carácter, etc.

SEÑAL ANALÓGICA. Se refiere a los valores de voltaje o corriente manejados por el circuito o sistema son continuos, sin interrupciones o saltos preestablecidos en su valor.

SEÑAL DIGITAL. Es cuando los valores de voltaje o corriente manejados por el circuito o sistema, no son continuos, varían solamente en dos valores, que representan un nivel lógico alto (5 volts) o un nivel lógico bajo (Cero Volts); es decir presencia o ausencia de información.

SÍNCRONO. Esta sincronizado por un reloj maestro.

SOFTWARE. Programas o instrucciones que le indican al sistema físico (hardware) que realizar y como realizarlo.

THROUG-HOLE. Al realizar circuitos impresos prototipos de un sistema, generalmente el impreso es de doble cara es decir las pistas se encuentran en caras diferentes de la tarjeta. Por lo regular una señal tiene que ser conducida por la parte frontal y también por la parte trasera de dicha tarjeta. Para que la pista conduzca es necesario que los orificios que pasan de un lado hacia el otro tengan conducción eléctrica.

TIERRA, PUESTA A. Conexión de un conductor eléctrico a un gran cuerpo conductor como la tierra, que se usa como cero en la escala de potenciales eléctricos. La tierra común se refiere cuando se utiliza un nodo referente a un voltaje de referencia, es decir se pone a tierra parte de un circuito para fijar su potencial.

TTL ("*Transistor Transistor Logic*, Lógica de Transistor Transistor). La tecnología TTL se refiere a dispositivos hechos mediante conexiones entre transistores.

XTAL (*Crystal Driver*): es un manejador de cristal, donde una señal de salida de reloj manejará la entrada **EXTAL** para otro dispositivo microcontrolador MC68HC11.

UART.- Transmisor Receptor Asincrono Universal.

MANUALES

- [MHC11] MC68HC11, Manual De Referencia HC11, MOTOROLA
- Datos Técnicos M68HC11 Series E. MOTOROLA INIC. 1995
- HC11, M68HC11 E SERIES, Programming Reference Guide. MOTOROLA
- HC11, M68HC11 N SERIES, Technical Data. MOTOROLA, INIC. 1993
- M68HC11EVBU, Universal Evaluation Board User's Manual. MOTOROLA INIC. 1990, 1997.
- TTL Logic. Texas Instruments
- Programmably Logic, National Semiconductor
- SEMICONDUCTOR TECHNICAL DATA, MOTOROLA.

LIBROS

- [DDPP] Diseño Digital Principios y Prácticas, John F. Wakerly, Editorial PHH
- [DS] Sistemas Digitales, Ronald J. Tocci, Editorial PHH
- [MCI] Los microprocesadores INTEL, Arquitectura, Programación e interfaces, 3ª. Edición, Barry B. Brey, Editorial PHH
- [MCT] Microcontroller Technology, Peter Spasov, Editorial Regent/Pretice Hall, U.S.A. 1993
- DICCIONARIO DE FISICA, John Daintith, b. Sc., Editorial norma.
- Microcontroladores de 4 y 8 bits, Christian Tavernier. Editorial PARANINFO.
- Semiconductor Memories a Handbook of Design, Manufacture, and Application. BETTY PRINCE. Edit. Wiley
- Enciclopedia de la ELECTRONICA, Ingeniería y Técnica, OCEANO/CEMTRUM
- [MS] Memorias de Semiconductor, ORBIS MARCOMBO

PUBLICACIONES

CEKIT
ELECTRONICA & COMPUTADORES
Edición Internacional número 44

POLIBITS
Revista de computación CINTEC
Año IV, VOL.1, NUM.9, JULIO - SEPTIEMBRE 1992

COMPUTERCRAFT
Say You Saw It In ComputerCraft
- Octuber 1992, Pág. 68
- November 1993.
- December 1993, Pág. 27