65 2ej

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO DE UN PROGRAMADOR DE MEMORIAS PARA UNA COMPUTADORA PERSONAL (PC. COMPATIBLE)

T E S I S

QUE PARA OBTENER EL TITULO DE;

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

SERGIO ARTURO GONZALEZ JACOME

DIRECTOR: ING. RODOLFO PETERS L.

MEXICO, D. F. 1990

INDICE

Debido al auge de los microcontroladores, micropocesadores y circuitos digitales especificos, la utilización de memorias semiconductoras ha ido en aumento.

Las computadoras, en general, necesitan de memorias que inicialicen su operación y faciliten su uso. Estas memorias son necesariamente del tipo de sólo escritura y no volátiles.

En el caso de microcontroladores esta generalizado el uso de memorias denominadas reprogramables o "EPROM", lo que crea la necesidad de contar en cualquier laboratorio de desarrollo de sistemas digitales con un programador de memorias del tipo EPROM. En el mercado existen muchos tipos de memorias, sin embargo, cuando apareció la familia 27XXX, esta se transformó en un estándar. La mayoria de las aplicaciones de sistemas digitales necesitan una capacidad de memoria ROM que se ve cubierta (desde 2 hasta 64 Kbytes) con esta familia.

2. Memorias Semiconductoras

En esta parte daremos una breve introducción a las memorias semiconductoras que permitirá aclarar algunas dudas acerca del funcionamiento de las mismas sin pretender ser una explicación total y mucho menos formal de las mismas.

Nos enfocaremos principalmente a las memorias semiconductoras del tipo MOS, que son programables según el concepto conocido como compuerta flotante.

Definiremos algunos conceptos que hemos considerado básicos para entender el funcionamiento de este dispositivo.

La memoria es un dispositivo que permite almacenar la información requerida para el funcionamiento de los distintos sistemas digitales (microcontroladores, microcomputadoras, etc.). La memoria puede clasificarse desde varios puntos de vista:

Por su Modo de direccionamiento se clasifican en acceso directo y secuenciales: **Direccionamiento directo:** A cada paquete de información (palabra), le corresponde una dirección determinada y su acceso toma el mismo tiempo, independientemente de la dirección de la misma (RAM y ROM).

Direccionamiento Secuencial: Se tiene acceso a la información en forma secuencial de la misma manera en que se grabó originalmente la misma, siendo el tiempo de acceso a la misma dependiente de la posición en la que se guardó la información (Discos, Cintas, etc.). El tiempo de acceso a la información es mayor que en las anteriores.

Normalmente las memorias de acceso directo se utilizan como memoria principal de un sistema, pues permite el direccionamiento de las instrucciones y de los datos necesarios en un mínimo de tiempo. Las memorias de acceso secuencial son utilizadas como memorias de almacenamiento masivo. Normalmente se tiene que el costo por bit es más elevado para las memorias de direccionamiento directo que para las de almacenamiento secuencial.

Otra clasificación se hace basada en la posibilidad de modificar la información de la memoria (de solo lectura o de escritura-lectura) o no.

Memoria de lectura y escritura (R/W o RAM): Es aquella cuya información puede ser modificada de acuerdo a las necesidades dinámicas de un usuario.

Memoria de únicamente escritura (ROMs): son aquellas que son direccionables como las de RAM, pero su escritura se ve restringida o no es posible.

De este tipo existen dos tipos, la ROM propiamente dicha, cuya información es grabada por el fabricante, y otra que puede ser programada (escrita) por el usuario (PROM).

De estas últimas existen las que pueden borrarse y programarse muchas veces (**EPROM**). Existe otra división basada en el método de borrado, así existen las **EEPROM** que pueden ser borradas aplicando una señal eléctrica en una de sus terminales y la **UVPROM**, que para borrarse debe exponerse a una fuente de luz ultravioleta.

Es importante resaltar la más importante ventaja de la ROM: la no volatilidad de la información, esto es, que al cortar el suministro de energía, los datos en una ROM permanecen en ella y en una RAM se pierden. Esta ventaja ha ido perdiendo peso con la llegada de la memoria RAM MOS de baja potencia, pues para ella es sencillo colocar una fuente de respaldo que mantenga los valores de la RAM cuando un sistema está fuera de operación. Debido a que en este trabajo se diseñó un programador de memorias del tipo EPROM, explicaremos brevemente cómo y porqué es posible

programar este dispositivo.

2.1 Celda EPROM

El prototipo de la EPROM utiliza un transistor esquematizado en la figura 1, la cual nos muestra un MOSFET canal n del tipo ensanchamiento (es decir sin canal n) con dos compuertas (gate) de material de polisilicio.

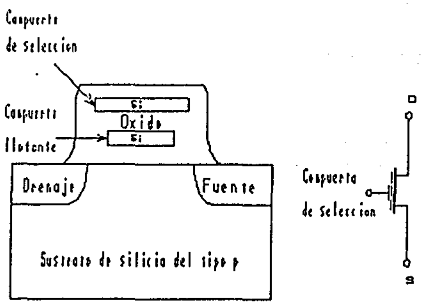


FIGURA 1

Una de las compuertas se encuentra sin conexión eléctrica con el resto de circuito (se encuentra aislada por capas de óxido de silicio), y por lo mismo es conocida como compuerta flotante ("Floating gate") y la otra es una compuerta normal de un MOSFET de ensanchamiento, conocida como compuerta de selección.

El transistor conocido como transistor de compuerta flotante tiene un símbolo como el mostrado en la figura 1, en la que se indica con una línea punteada la compuerta flotante. La celda de memoria fabricada con este transistor es conocida como celda de compuerta apilada.

Originalmente no existe carga alguna en la compuerta flotante y el transistor opera de manera regular como un MOSFET de ensanchamiento. Para programar el transistor, se aplica Vcc entre la Fuente ('Source') y el Drenaje ('Drain'), y voltaje alto (Vpp) en la compuerta de selección simultáneamente.

En este punto, como en cualquier transistor MOSFET de ensanchamiento de canal y debido a que el voltaje en la compuerta es grande, se genera un canal afilado que hace que los electrones del último adquieran una gran energía cinética que les permite llegar a la compuerta flotante donde quedan atrapados (figura 2) .

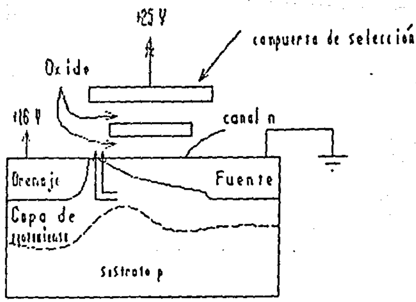


FIGURA 2

La cantidad de carga en la compuerta flotante está dada por el tiempo que se aplica el voltaje de programación (Vpp) de cierto nivel a la compuerta de selección. El voltaje de programación no puede tomar cualquier valor debido a que si es muy bajo no acelerará a los electrones y si es muy alto, puede afectar de manera destructiva al transistor. Para la familia 27, Vpp varía desde 12.5 V a 25 V con una variación de ±1 V. Transcurrido dicho tiempo la carga negativa en la compuerta flotante hace que el campo eléctrico entre la compuerta de selección y el drain se debilite hasta llegar a un punto en el cual no puede acelerar más electrones. El tiempo de programación total será entonces menor si logramos colocar la carga en la compuerta flotante en el mínimo tiempo. Con el método denominado Programación de Pulso Rápido ('Quick

Pulse Programming'), se aprovecha el margen de variación de Vpp y Vcc, para acelerar al máximo permisible los electrones.

Para reconocer que la memoria se programó (se cambió el umbral de conducción), sólo se tiene que observar su comportamiento. En la siguiente figura se observa el comportamiento de un transistor antes y después de programarse. Si se aplica un Vgs intermedio entre las dos curvas (este valor esta normalizado al nivel TTL alto, es decir 5 V), la que conduzca (no tendrá carga en la compuerta flotante que impida la formación del canal) estará sin programar, es decir con un '1' lógico y la que no lo haga (debido al campo eléctrico de la carga almacenada) estará programada con un '0' lógico.

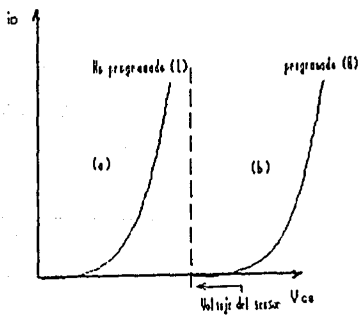


FIGURA 3

Para devolver al transistor de compuerta flotante su condición inicial ('no programado' o 'borrado'), es necesario remover la carga almacenada en la compuerta. Debido a que en la UVPROM no existe conexión física a la compuerta flotante, no puede removerse por medio de un pulso eléctrico. Para hacerlo hay que exponer al transistor a una luz ultravioleta con una longitud de onda de 2537 A^{o} durante un tiempo determinado (normalmente de 15 a 20 minutos). Esto suministra energía suficiente a los electrones atrapados en la compuerta para atravesar el óxido de silicio y regresar al substrato. Para permitir el paso de la luz, las memorias de este tipo tienen una ventana de cuarzo.

Las Memorias semiconductoras tienen un arreglo matricial para su direccionamiento. De tal manera que internamente se selecciona renglón y columna (en memorias grandes se direcciona también bloque), pero esto es totalmente invisible para el usuario, que sólo ve a la memoria como una tabla de relación entre la dirección y la información guardada

3. Método Convencional de Programación

Para la programación, hasta 1983, se había utilizado únicamente el algoritmo que utilizaba el hecho de que con un pulso de programación de 50 ms la información quedaba grabada en forma correcta, dando esto por resultado que para memorias de, por ejemplo, 32 kbytes (27256) se tuvieran tiempos de programación de hasta 24 minutos.

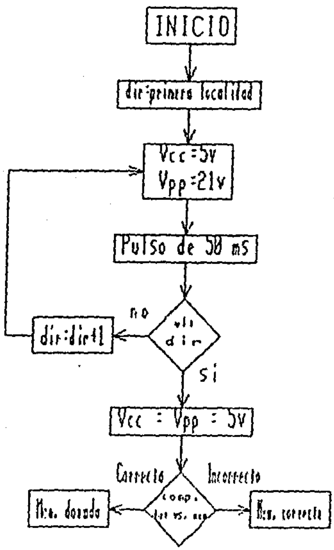
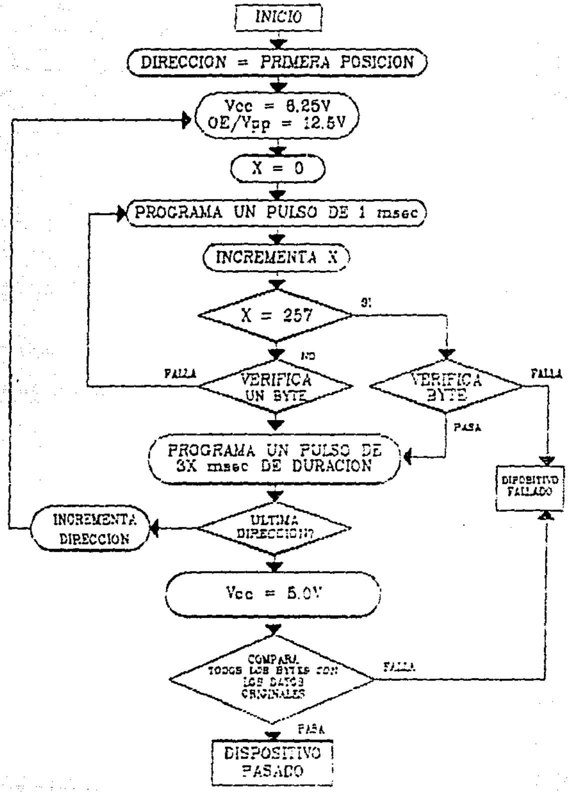


FIGURA 4

3.1 Algoritmo de Programación Inteligente (Intelligent Programming Algorithm)

Entonces apareció un método desarrollado por Intel, que permitía reducir el tiempo de programación hasta aproximadamente 6 minutos (en la misma ROM 27256).

Este método se basa en el hecho de que no se necesita aplicar un pulso de 50 ms. Lo que hacen es aplicar 2 tipos de pulsos, el primero en una ráfaga formada por pulsos de 1 ms y el segundo un pulso de reprogramación (overprogramming) de 3X ms, donde X es un contador de iteraciones, que cuenta cuantos pulsos de 1 ms fueron aplicados antes de que se verificara que los datos de la ROM son iguales a los originales. La secuencia completa de pulsos de programación y la verificación se realizan con Vcc en 6 V y Vpp en su valor nominal. En este método, se utiliza la reprogramación (overprogramming) debido a que con la secuencia de pulsos se aplica poco a poco carga en la compuerta flotante y que puede en un momento dado dar lugar a falsas lecturas (el umbral para el reconocimiento de los bits '0' y '1' lógicos es inestable), Por lo que al aplicar un pulso del triple de duración se asegura que este umbral se encuentre estable.

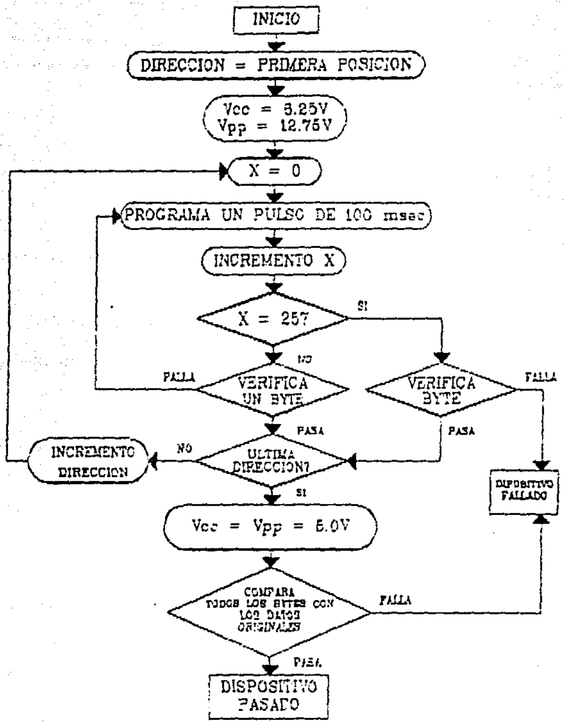


INICIO
DIRECCION = PRIMERA POSICION
Vcc = 6.25V
OE/Vpp = 12.5V
X = 0
PROGRAMA UN PULSO DE 1 msec
INCREMENTA X
X = 257
SI
NO
FALLA
VERIFICA UN BYTE
VERIFICA BYTE
FALLA
PASA
PROGRAMA UN PULSO DE 3X msec DE DURACION
DISPOSITIVO FALLADO
INCREMENTA DIRECCION
ULTIMA DIRECCION?
Vcc = 5.0V
COMPARA TODOS LOS BYTES CON LOS DATOS ORIGINALES
FALLA
PASA
DISPOSITIVO PASADO

FIGURA 5

3.2 Algoritmo de Programación de Pulso Rápido (Quick Pulse Programming Algorithm)

Este es el último y más rápido algoritmo de programación utilizado. Este aplica pulsos de 100 µs hasta un número de 25 de estos . El pulso de sobre-escritura es eliminado ya que se eleva el umbral lógico al colocar Vcc en 6.25 V, lo que permite asegurar que la carga en la compuerta es suficiente para diferenciar entre un cero y un uno. Vpp en el valor nominal más 0.25 V para proporcionar mayor energía.



INICIO
DIRECCION = PRIMERA POSICION
Vcc = 6.25V
Vpp = 12.75V
X = 0
PROGRAMA UN PULSO DE 100 msec
INCREMENTO X
X = 25?
SI
NO
FALLA
VERIFICA UN BYTE
VERIFICA BYTE
FALLA
PASA
PASA
NO
INCREMENTO DIRECCION
ULTIMA DIRECCION?
SI
DISPOSITIVO FALLADO
Vcc = Vpp = 5.0V
COMPARA TODOS LOS BYTES CON LOS DATOS ORIGINALES
FALLA
PASA
DISPOSITIVO PASADO

FIGURA 6

3.3 Método utilizado en el programador desarrollado

Se utilizó un procedimiento que permite variar poco a poco el periodo del pulso de programación hasta encontrar el tiempo mínimo requerido para programar la primera localidad, es decir se prueba con diferentes anchos de pulso hasta que se observa en modo de lectura que los datos escritos son iguales a los enviados. Se toma este ancho de pulso como patrón para los siguientes pero permitiendo en las escrituras posteriores que el ancho del mismo se vea incrementado aún más.

Esto se puede hacer debido al hecho de que los transistores forman parte del mismo integrado, es decir, los transistores están fabricados sobre la misma oblea y tienen por consiguiente, prácticamente, las mismas características. Por ello el pulso de programación de las siguientes localidades será el mismo (o casi el mismo) que el usado para programar la primera localidad. Esto nos permite minimizar los tiempos de programación sin tener que elevar los voltajes de programación y polarización a valores cercanos a los limites máximos.

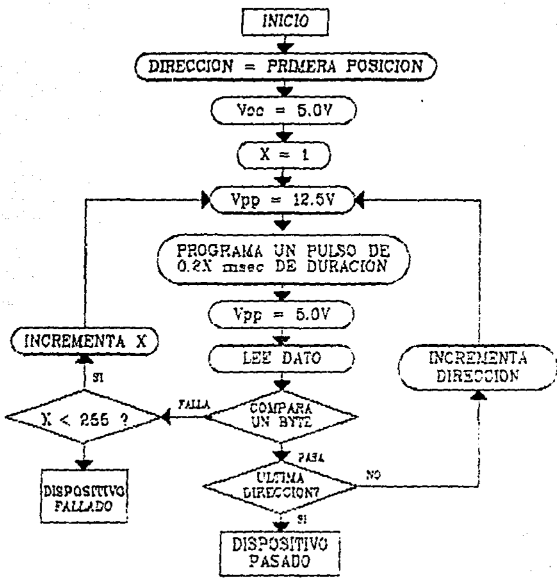

INICIO
DIRECCION = PRIMERA POSICION
Vcc = 5.0V
X = 1
Vpp = 12.5V
PROGRAMA UN PULSO DE 0.2X msec DE DURACION
Vpp = 5.0V
LEE DATO
COMPARA UN BYTE
FALLA
PASA
X < 255 ?
SI
INCREMENTA X
DISPOSITIVO FALLADO
ULTIMA DIRECCION?
NO
SI
INCREMENTA DIRECCION
DISPOSITIVO PASADO

FIGURA 7

4. Operación del Programador

En esta sección solamente daremos una visión general de como funciona el programador. En capítulos posteriores se darán mas detalles.

Por programa se varía el voltaje de programación entre los valores estandarizados (12.5, 21.0 y 25.0 V), así como la ubicación del mismo (se pueden seleccionar únicamente las terminales 1, 22 y 23), con lo cual se permite una posible expansión del número de dispositivos factibles de programar (como por ejemplo la EPROM Paginada 27513) que tengan estos voltajes en las terminales mencionadas.

Esto último con la salvedad de que la terminal OE del dispositivo sea verificada baja y se encuentre en la terminal número 20. Esto es debido a que esta terminal comanda la dirección del puerto A del 8255 para lectura de Datos (refiérase al apéndice C).

Se utilizó una sola base (de las denominadas cero esfuerzo) para colocar la memoria a programar (La mayoría de los programadores en el mercado tienen 2 bases para programar las mismas memorias, una para las de 24 terminales y la otra para las de 28), que se coloca externamente sobre la PC o en su soporte y que se encuentra conectada por medio de un cable plano a la tarjeta interna (conectada a la ranura o "slot" de la PC apéndice B).

4.1 Diseño de la Tarjeta

El Voltaje de Programación se obtiene a través de la fuente de 5 V de la ranura de la PC utilizando una fuente conmutada elevadora de voltaje (TL497 de Texas Instrument).

Esta fuente nos permite obtener los 3 voltajes de programación necesarios conmutando el juego de resistencias en su realimentación. Esta conmutación se realiza a través de 3 transistores comandados por las 3 lineas libres del Puerto C del 8255 (las otras están dedicadas al control y estado del Puerto A cuando trabaja en Modo 2) denominado Voltaje Prog. Cuando no se genera ninguna realimentación, esta fuente deja de conmutar, y a la salida de la misma tendremos un voltaje de 5 V (Refiérase al apéndice D diagrama 3 para mayores detalles).

Debido a que la terminal donde se aplica el voltaje de programación varía (terminales 1, 22 y 23) entre un tipo y otro de memoria, se maneja un arreglo de transistores que nos permite aplicar el Vpp en la terminal necesaria. El direccionamiento se realiza por medio de 3 lineas del 'Latch' denominado de Control. Si nosotros no aplicamos señal a los transistores se tiene una resistencia a tierra que mantiene esta señal en bajo, lo que nos da la facilidad de seleccionar 5 posibles voltajes (25, 21, 12.5, 5 y 0 V) en cualquiera de las terminales Vpp(1, 22 y 23 en la base de cero esfuerzo).

La familia 27 tiene un buen número de líneas en común, y se tomó esto como base para el diseño (apéndice A).

Debido a que en la familia 27 los datos tienen la misma posición con respecto a la terminal de la referencia (GND). Se utilizó la base de cero esfuerzo con estas líneas en común por lo que al colocar la memoria debe de cuidarse de que quede en la posición apropiada.

Los Datos (D0-D7) son comunes a todas las memorias, por lo que se utilizó el puerto A del 8255, aprovechando que este puede configurarse de modo bidireccional (apéndice D diagrama 1).

Las Direcciones son también comunes a todos pero se tiene el problema que en algunas memorias, en lugar de ellas existen señales de control y/o la polarización y Voltaje de programación en la base de cero esfuerzo.

La parte mas conflictiva del diseño se presentó debido a que se tienen terminales que corresponden a una dirección en una memoria y a la terminal donde se aplica el voltaje de programación en otra. Para evitar dañar la salida del puerto o del latch, se colocó a la salida de estos (en las terminales 1, 22 y 23) diodos. Así, cuando se aplica un voltaje de programación, no se daña el puerto , en el otro caso el puerto determina el comportamiento.

Otro problema que se halló durante el diseño fue que, debido a su tamaño, en la terminal 26 esta colocado el Vcc en algunas memorias y en otras A13. La salida del 8255 es insuficiente para manejar la corriente necesaria, en el primer caso y por lo mismo se utilizó un seguidor armado con un inversor y un transistor, que sigue el comportamiento de la salida del puerto, por lo que es transparente su utilización

Mientras que en la mayoria las memorias el pulso de programación se aplica en la terminal denominada CE, en las memorias 2764 y 27128, se debe aplicar un pulso de programación en la terminal 27 (PGM), y se aplica cuando la dirección, los datos y el voltaje de programación están estables. Por ello, para manejarlo ùnicamente desde el Puerto de Control, se utiliza una compuerta AND, para asegurar que el pulso en realidad existe, y no se aplique una señal de DC.

4.2 Mapa de Puertos

PUERTOS	NOMBRE EN PROGRAMAS	DIR. DISP.
Datos (D0-D7)	Pto_Datos	F78 8255 (A)
DireccionesLS (A0-A7)	Pto_Dir_H	F79 8255 (B)
DireccionesMS (A8-A15)	Latch_Dir_L	F7E LATCH
Voltaje Prog (Vpp)	Pto_Vpp	F7A 8255 (C)
Configuración	Pto_Conf	F7B 8255
Control	Latch_Dir_V	F7D LATCH

(Para mayor detalle consultar el apéndice D diagrama 2)

4.3 Base de Tiempo

Para el Manejo del Ancho de Pulsos de Programación se consideran 2 opciones.

Software - A través de ciclos

Hardware - A través de circuitos del tipo monoestable ('one shot')

Para la primera se encontró que el valor obtenido era dependiente de la máquina (XT, AT, PC turbo, etc) donde se encontrara la tarjeta. Ya que un ciclo implementado en una PC XT (4.77 MHz) tarda aproximadamente el doble que en una máquina PC turbo (8 Mhz).

Para la segunda se encontró que eran necesarios varios monoestables o un monoestable 'programable', lo que requería que la tarjeta fuera más grande y que se necesitara un número mayor de componentes electrónicos.

La solución fue utilizar el "hardware" provisto por la PC. Se programó el circuito 8253 de la PC para producir un sonido de 5 kHz (a través de su bocina), mediante la instrucción SOUND de Pascal, esta señal a su vez es muestreada por uno de los puerto del circuito 8255 de la PC (con dirección 62H) en el bit 5. Para evitar el sonido de 5 kHz se apaga la bocina mediante el bit 0 del puerto 61.

Este método nos permite variar la duración del Pulso de Programación en múltiplos de 0.2 ms hasta 51 ms de una manera sencilla (apéndice D diagrama 4).

5. Manual de Usuario

El Usuario que desee programar alguna memoria solamente debe de selecionar las opciones adecuadas en la barra de opciones y responder a las preguntas que se le harán. Deberá tener cuidado en colocar la memoria EPROM cuidando que la terminal 14 o 12 (dependiendo si es una memoria con 28 o 24 terminales) esté alineada en la parte inferior izquierda de la base. Si esto no se realiza correctamente, es probable que la misma se dañe pues se polarizará en forma inversa (Vcc en Gnd y viceversa).

Para ejecutar el programa sólo hay teclear

A>PII

y enseguida aparecerá una pantalla en la cual se tienen 2 áreas.

La primera muestra un bloque o página de 256 localidades de un byte (en total el 'buffer' de memoria tiene 256 de estas páginas, lo que da un total de 64 kbytes). A la izquierda, se nos muestra el código hexadecimal y a la derecha el caracter ASCII correspondiente. En la parte superior se le muestra en que localidad se encuentra en este 'buffer'.

La segunda es una barra, en la cual se encuentran las diferentes opciones que puede seleccionar el usuario. Para seleccionar alguna, deberá posicionarse sobre la misma utilizando las flechas, o bien con los números del 1 al 9, y después pulsar Enter. Estas opciones serán explicadas brevemente en las páginas siguientes.

5.1 Seleccion de Memoria

Permite seleccionar la memoria a programar. Esta selección configura a la base de cero esfuerzo en una forma especial para cada memoria. La tecla Esc (o Del) retrocederá en las memorias (de mayor a menor capacidad es decir de la 27512 -> 27256 -> 27128 -> 2764 -> 2732 -> 2716 ->2516), mientras que la tecla Enter (o Ins) avanzará, es decir es complementaria. Esta tarea debe realizarse siempre cuidadosamente, pues una selección errónea puede dañar su dispositivo irreparablemente o no programarlo en forma debida (o no realizar ninguna tarea).

Para ilustrar esto daremos 2 ejemplos:

Si se tiene una memoria 27512, cuyo voltaje de programación se debe aplicar en la terminal 22, y por equivocación le indicamos al programa que es una memoria. 27256, se dañará el dispositivo, pues se aplicará el voltaje de programación sobre una terminal de la memoria que no está diseñada para voltajes de esta magnitud (una línea de direcciones con límite de voltaje igual a Vcc + 1 V.

Si se tiene una memoria 27256 y se le indica al programa que se tiene una 2764, no se dañará la memoria, pues ambas tienen el Vpp asignado a la terminal 1, pero como la memoria 2764 tiene una terminal PGM que corresponde a una direccion en la 27256, no se programará correctamente (de hecho no marcará un error al llegar a la dirección 4000 pues tratará de escribir de nuevo en la dirección 0000 que tal vez ya esté programada con otra información).

5.2 Selección de Voltaje de Programación

Esta operación es también muy importante, pues si el usuario no da el valor adecuado la programación tampoco se realizará apropiadamente. Debido a que existe un gran número de fabricantes de memorias y que no existe un voltaje de programación determinado para cada tipo de las mismas, el usuario deberá investigar en el manual de memorias del fabricante cuál es el voltaje correspondiente. Si se aplica un voltaje de programación más grande del indicado el dispositivo se dañará. Si es apremiante la programación y no puede consultar el manual, puede probar con el voltaje mínimo (12.5 V) y si no se programa correctamente, aumentar gradualmente el mismo hasta que lo haga correctamente (aumentar a 21 y 25 V). Para seleccionar el voltaje de programación las teclas Esc (o Del) y Enter (o Ins) tienen la misma función que en la opción anterior (Enter 12.5 -> 21.0 -> 25.0 y Esc 25.0 -> 21.0 -> 12.5).

5.3 Lectura de un Archivo de disco

Esta opción permite la lectura de un archivo con el código hexadecimal de información. Esta opción nos permite evitar el tecleado de la información múltiples veces. Al realizar esta operación, la información en la pantalla variará para mostrar esta variación en el 'buffer'.

Al seleccionar esta opción, se le cuestionará acerca del nombre del archivo a leer. Si no existe archivo con el nombre suministrado, se le informará y se le repetirá la pregunta, pudiendo salir de esta opcion suministrando el nombre "X". Se deberá tomar en cuenta que sólamente se cargarán los primeros 64 kbytes del mismo pues esta es la capacidad del 'buffer'. Tambien se necesitará saber si desea un desplazamiento ("offset") al cargarse la memoria.

5.4 Escritura a un Archivo de disco

Esta opción nos permite copiar totalmente o en parte la información contenida en el 'buffer' de memoria. Junto con la opción anterior permite el copiado de la información de una memoria, cuando no se tiene una disponibilidad simultánea de ambas memorias (Fuente y Destino).

Al seleccionar esta opción, se le cuestionará sobre el nombre del archivo que se quiere generar, si ya existe le dará la oportunidad de cancelar la operación o de sobreescribir. Con "X" se le permite salir de esta opción. El otro dato que proporciona el usuario es el número de bytes que se quiere copiar al archivo.

5.5 Lectura de Memoria

La opción de Lectura de memoria permite copiar la información contenida en la memoria que se encuentre en la base al 'buffer' de 64 kbytes anteriormente señalado. Al seleccionar esta opción la información de la página visible en la pantalla variará para mostrar la página acabada de leer.

5.6 Escritura a Memoria

La opción de Escritura a memoria, es la parte más importante de todas, pues es la que programa en sí a la memoria. Se le preguntará al usuario que porción del buffer se grabará en la memoria, y desde que localidad en la misma.

5.7 Verificación de Borrado

La opción de verificación de borrado permite revisar antes de programar una memoria si está correctamente borrada, es decir, que en todas las localidades tengan 'FF'. Si se encuentra información diferente a 'FF' en cualquier localidad se incrementa un contador y al terminar la operación aparecerá un mensaje informando cuantas localidades con información se tienen, o en caso contrario, el mensaje indicará que la memoria esta completamente borrada. Se recomienda utilizar esta opción antes de cualquier escritura a memoria.

La opción de edición tiene en realidad dos funciones:

A la primera se tiene acceso pulsando una vez (sólo una) Enter (o Ins) cuando la opción esta marcada en la barra de opciones. Se indicará que entra a modo de inspección cambiando el color (de violeta a azul) de la localidad en la que se encuentre (si tiene monitor monocromático cambiará el tono del mismo).

Esta opción permite observar la información del 'buffer', y movernos a lo largo de los 64 kbytes de información utilizando las teclas siguientes:

Flechas: sirven para moverse a las localidades adyacentes en la pantalla (arriba, abajo, izquierda y derecha). Si nos encontramos en las localidades límites de la misma (localidad inicial o final de cada página), nos moveremos a la página anterior o siguiente según sea el caso. 'Home': Nos posiciona en el primer byte del buffer, es decir en la primera localidad de la primera página de 256 bytes (Localidad '0000').

'End' : Nos posiciona en la última página y último byte de la capacidad de la memoria seleccionada.

Así, con la memoria 27512, nos colocará en la localidad 'FFFF', en la 27256 en la '7FFF' en la 27128 en la '3FFF' en la 2764 en la '1FFF', en la 2732 en la '0FFF' en la 2716 en la '07FF'.

'PgUp': Nos permite mover a la localidad que se encuentra a 256 bytes hacia atrás de la localidad en la que nos encontremos (si estamos en la localidad '0A25', al pulsarla nos colocaremos en la '0925').

'PgDn': Nos permite mover a la localidad que se encuentra a 256 bytes hacia adelante de la localidad en la que nos encontremos (si estamos en la localidad '0A25', al pulsarla nos colocaremos en la '0B25').

Como puede verse, 'PgUp' y 'PgDn' realizan funciones complementarias.

Para salir de este modo de inspección, se deberá pulsar la tecla Esc (o Del), lo cual regresa el control a la barra de opciones.

A la segunda opción de edición se tiene acceso si, estando en la opción de inspección, se pulsa Enter (o Ins). Hecho esto cambiará el color nuevamente (de azul a blanco) para indicarle que está en modo de edición, además de aparecer el mensaje '** En Edición **' en la parte superior.

En este modo, sólo se reconocen las teclas que representan los dígitos hexadecimales ('01234546789ABC DEF') o las teclas Esc (o Del) y Enter (o Ins). Estas últimas nos permiten brincarnos a la siguiente localidad sin necesidad de haber tecleado ambos digitos hexadecimales. 'Esc' nos permite salir del modo de edición al modo de inspección . Es útil cambiar entre el modo de inspección y el de edición cuando se quieren cambiar datos en localidades no consecutivas, pues la opción de inspección nos da más movilidad.

5.9 Comparación Memoria - Buffer

La opción de comparación nos permite comparar una porción o la totalidad de la memoria EPROM con sus correspondientes localidades en el "Buffer" de la computadora.

Si existe diferencia nos indicará la primera localidad donde existió la variación en la información.

5.10 Salida

Esta opción nos permite terminar la sesión. Obviamente, la información en el buffer de memoria se pierde, por lo que deberá estar seguro que la misma no tiene importancia o ya ha sido utilizada. El control regresa al DOS.

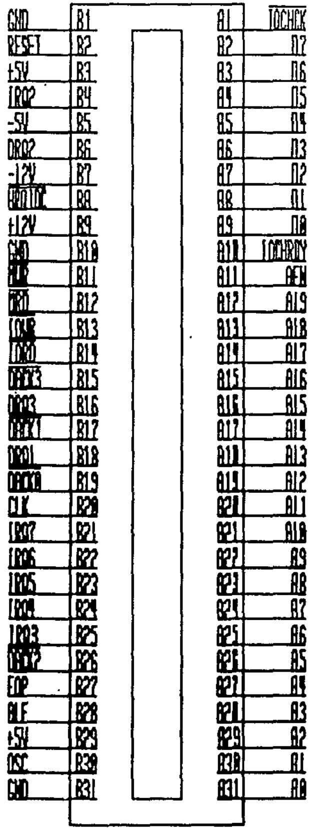
APENDICE A

Huella de la

Familia 27XXX

512	256	128	64	32	16	Base	16	32	64	128	256	512
A15	Vpp	Vpp	Vpp			1 28			Vcc	Vcc	Vcc	Vcc
A12	A12	A12	A12			2 27			PGM	PGM	A14	A14
A7	A7	A7	A7	A7	A7	3 26	Vcc	Vcc	-	A13	A13	A13
A6	A6	A6	A6	A6	A6	4 25	A8	A8	A8	A8	A8	A8
A5	A5	A5	A5	A5	A5	5 24	A9	A9	A9	A9	A9	A9
A4	A4	A4	A4	A4	A4	6 23	Vpp	A11	A11	A11	A11	A11
A3	A3	A3	A3	A3	A3	7 22	OE	OE/Vpp	OE	OE	OE	OE/Vpp
A2	A2	A2	A2	A2	A2	8 21	A10	A10	A10	A10	A10	A10
A1	A1	A1	A1	A1	A1	9 20	CE	CE	CE	CE	CE	CE
A0	A0	A0	A0	A0	A0	10 19	D7	D7	D7	D7	D7	D7
D0	D0	D0	D0	D0	D0	11 18	D6	D6	D6	D6	D6	D6
D1	D1	D1	D1	D1	D1	12 17	D5	D5	D5	D5	D5	D5
D2	D2	D2	D2	D2	D2	13 16	D4	D4	D4	D4	D4	D4
GND	GND	GND	GND	GND	GND	14 15	D3	D3	D3	D3	D3	D3

APENDICE B



BUS DE IBM PC O COMPATIBLE

Señales del Bus de la PC

Siendo el programador una tarjeta de expansión para una computadora PC a continuación se describen las señales del bus:

A0-A19

Son las direcciones de los bit 0 al 19 las cuales se utilizan para direccionar la memoria y los dispositivos de entrada salida del sistema. Estas lineas pueden ser generadas ya sea por el procesador o un controlador de DMA.

ALE

(Adress Latch Enable). Es una señal empleada para habilitar los acondicionadores de las direcciones de salida del 8088.

CLK

Reloj del sistema con una frecuencia que depende del tipo de máquina. Para las máquinas PC XT es de 4.77 MHz y el ciclo de trabajo es del 33%.

AEN(CPUI)

(CPU Inhibit). Inhibe al controlador del bus 8288 y a los acondicionadores de las direcciones para permitir que el DMA asuma el control del bus.

D0-D7

Lineas del 0 al 7 de datos que constituyen el bus de datos de 8 bits

DACK0-DACK3

(DMA Acknowledge channels). Estos canales de reconocimiento del DMA activos bajos notifican a dispositivos periféricos individuales cuando se ha otorgado un servicio de DMA.

DREQ1-DREQ3

(DMA Request chanels). Un dispositivo periférico puede obtener servicio del DMA a través de estos canales de petición de entrada.

EOP

(End of Process). Señal de salida que representa la terminación de un servicio del DMA.

HRQIOC

Reservada.

IOCHCK

(I/O Channel Check). Señal generada por una tarjeta externa en el bus de expansión que informa al CPU de una falla catastrófica dirigida al CPU por medio de la circuitería del NMI.

IOCHRDY

(I/O Channel Ready). Señal de "handshake" para una tarjeta externa en el bus de expansión empleada cuando se insertan estados de espera a los ciclos de reloj del CPU o un DMA.

IORD

(I/O Read). El CPU activa en bajo a esta señal para realizar una lectura de un dispositivo periférico del sistema.

IOWR

(I/O Write). El CPU activa en bajo a esta señal para realizar una escritura hacia un dispositivo periférico del sistema.

IREQ2-IREQ7

(Interrupt Request Channels). Estas señales de entrada son canales de petición al 8259; los niveles de prioridad de los mismos son establecidos por el diseñador del sistema.

MRD

(Memory read). Señal que el CPU habilita en bajo cuando realiza una lectura a memoria.

MWR

(Memory Write). Señal que el CPU habilita en bajo cuando el CPU realiza una escritura a memoria.

OSC

Señal cuya frecuencia es equivalente a la frecuencia del cristal de 14.318 MHz, usada exclusivamente por la circuitería del video.

RESET

Señal para restablecer o iniciar el sitema lógico.

VOLTAJES

Los voltajes que se enlistan están disponibles en el bus de expansión:

+ 5 V

- 5 V

+12 V

-12 V

GND

Nota: Las señales DACK, HRQIOC,IOCHCK,IORD,IOWR,MRD,MWR son señales activas bajas.

APENDICE C

INTERFAZ PERIFERICA PROGRAMABLE

(PROGRAMMABLE PERIPHERAL INTERFACE)

8255A/8255-5

El 8255 es un dipositivo de entrada/salida programable de uso general con microprocesadores Intel, este tiene 24 patas de entrada/salida (e/s) las cuales se pueden programar individualmente en dos grupos de 12 y usadas en tres modos mayoritarios de operación. En el primer modo (MODO 0), cada grupo de 12 patas e/s pueden ser programádos en grupos de 4 patas para ser entrada o salida. En el segundo modo (MODO 1), cada grupo se puede programar para tener 8 entradas y 8 salidas, las restantes patas se utilizan para protocolo e interrupciones. Por último el tercer modo (MODO 2) es un modo de canal (bus) bidireccional donde 8 patas son bidireccionales y 5 patas se utilizan para protocolo.

En este proyecto el modo seleccionado fue el MODO 2 por lo. que a continuación se describe el aspecto funcional para este caso de operación del circuito 8255.

Definiciones funcionales:

1.- Puerto bidireccional (Puerto A)
2.- Puerto de control y protocolo (Puerto C bits 3-7)
3.- Puerto programable en dos direcciones (Puerto B)

4.- Para poder utilizar el MODO 2 se debe escribir al puerto de control la siguiente palabra.

1	1	*	*	*	0	0	0

→ PC_{2-0}
1=entrada
0=salida

→ PUERTO B
1=entrada
0=salida

→ GRUPO B MODO
0=MODO 0
1=MODO 1

Nota:

Para obtener una salida en el puerto B solo es necesario escribir en el puerto con dirección (01).

Para controlar la dirección del puerto A se tiene el siguiente protocolo.

Definición de señales de control:

INTR (Interrupt Request pto. C pata 3). Un nivel alto en esta entrada puede ser usado para interrumpir al CPU en las operaciones de entrada o salida.

OBF (Output Buffer Ful pto. C pata 7). La salida OBF irá a un nivel bajo para indicar que el CPU ha escrito datos al puerto A.

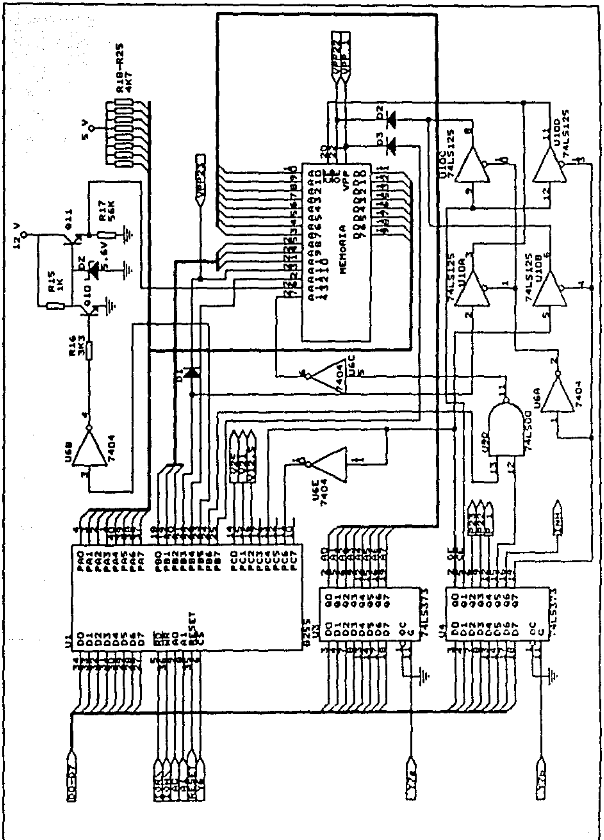
ACK (Acknowledge pto. C pata 6). Un nivel bajo en esta entrada habilita las salidas del "buffer tri-state" del puerto A y saca los datos, de otra forma las salidas del "buffer" estarán en alta impedancia.

STB (Strobe Input pto. C pata 4). Un nivel bajo en esta entrada carga los datos dentro del "latch" de entrada en el puerto A.

IBF (Input Buffer Full F/F pto. C pata 5). Un nivel alto en esta salida indica que los datos han sido cargados en el "latch" de entrada.

APENDICE D

DIAGRAMA 1.



MEMORIA
8255
74LS373
74LS125
74LS00
7404
R18-R25
4K7
R17
56K
R15
1K
R16
3K3
5 V
12 V

DIAGRAMA 2.

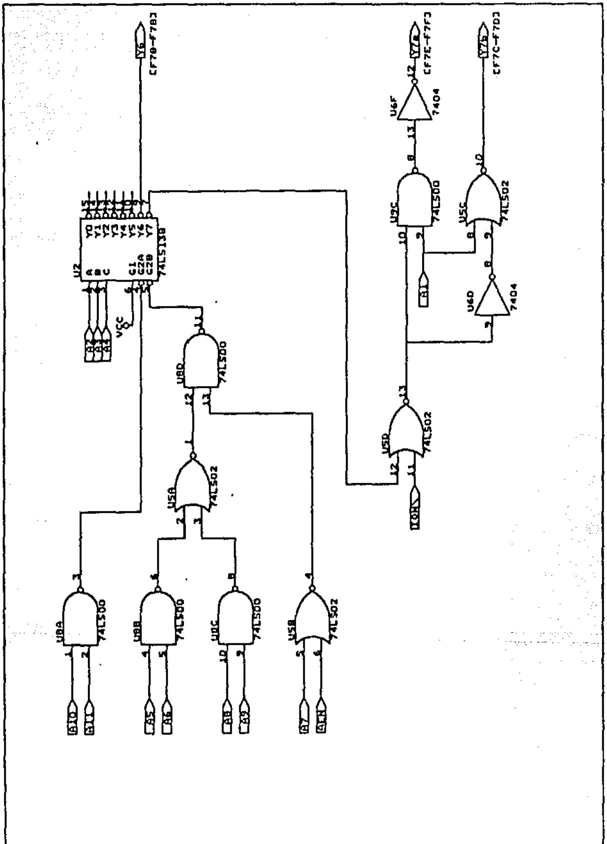
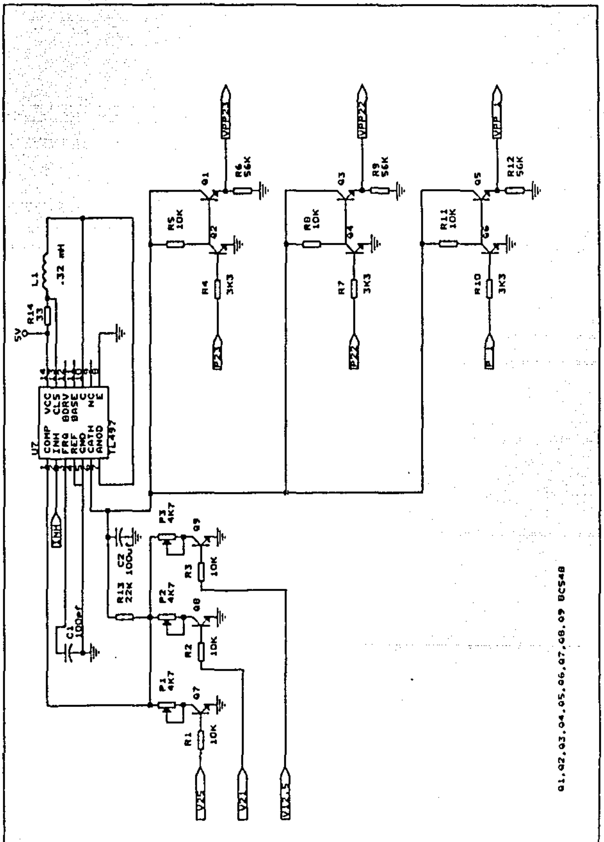
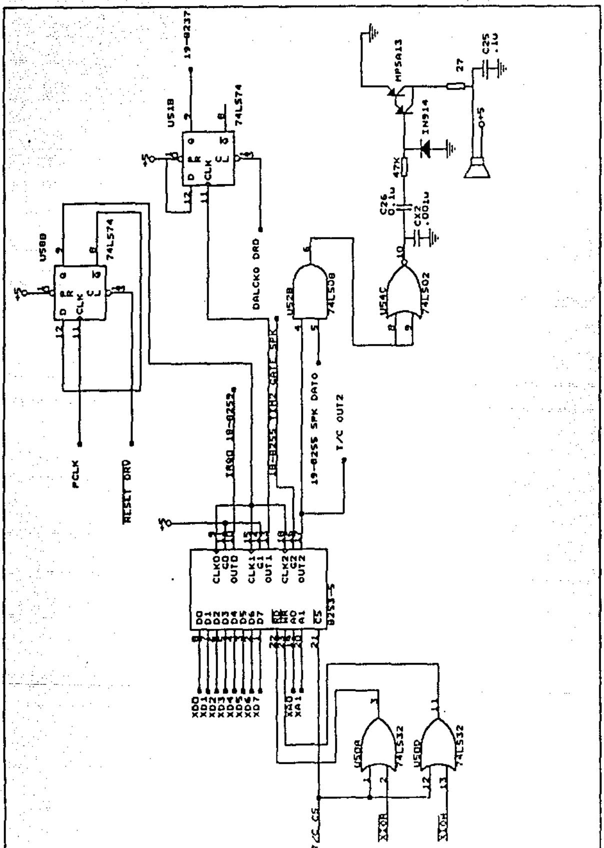


DIAGRAMA 3.



VPP21
VPP22
VPP
R6 56K
R9 56K
R12 56K
Q1
Q2
Q3
Q4
Q5
Q6
R5 10K
R8 10K
R11 10K
R4 3K3
R7 3K3
R10 3K3
P21
P22
P
L1 .32 mH
R14 33
5V
U7
COMP INH FRQ REF GND CATH ANOD
VCC CLS BDRV BASE C NC E
TL497
INH
C2 100uf
R13 22K
C1 100pf
P1 4K7
P2 4K7
P3 4K7
Q7
Q8
Q9
R1 10K
R2 10K
R3 10K
V25
V21
V12.5
Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9 BC548

DIAGRAMA 4.


U58B
74LS74
U51B
74LS74
19-8237
PCLK
RESET DRD
DALCKO DRD
U52B
74LS08
U54C
74LS02
MPSA13
IN914
47K
C26
0.1u
CX2
.001u
27
C25
.1u
+5
19-8255 SPK DATO
T/C OUT2
CLK0
G0
OUT0
CLK1
G1
OUT1
CLK2
G2
OUT2
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
CS
8253-5
XD0
XD1
XD2
XD3
XD4
XD5
XD6
XD7
XA0
XA1
U50A
74LS32
U50D
74LS32
T/C CS

APENDICE B

```
Program Memoria;
{$L PII}
{$F-}
{$R-}
Uses Crt,Dos;
Type      Apuntador=^Char;
          Str2 =String[2];
          Str4 =String[4];
const Mem64K          :Word = $FFFF;
      Llenado               =  $ff;
      Bloques128Bytes:Word  =  512;
      Home                  =  #71;
      Up                    =  #72;
      PD                    =  #73;
      Fin                   =  #79;
      Down                  =  #80;
      PU                    =  #81;
      Ins                   =  #82;
      Del                   =  #83;
      Right                 =  #77;
      Left                  =  #75;
      Enter                 =  #13;
      Esc                   =  #27;
      Negro                 =    0;
      Azul                  =    1;
      Rojo                  =    4;
      Cafe                  =    6;
      Blanco                =    7;
      Amarillo              =   14;
      Verde                 =   10;
      AzulClaro             =   11;
      Memor                        : Array[1..9] Of String[8]
                            =('2516    ',
                              '2716    ',
                              '2716-Esp',
                              '2532    ',
                              '2732    ',
                              '2764    ',
                              '27128   ',
                              '27256   ',
                              '27512   ');
      Opcion                       : Array[1..10] Of String[6]
                            =('S Memo',
                              'S Vpp ',
                              'L Arch',
                              'E Arch',
                              'L Memo',
                              'E Memo',
                              'V Borr',
                              'Editar',
                              'Compar',
                              'Salida');
      Opciones       :String='  S Memo S Vpp  L Arch E Arch L Memo E Memo V Borr Editar Compar Salida ';
      Vpp                          : Array[1..3] Of String[4]
                            =('25.0',
```

```
                              '21.0',
                              '12.5');
Var
     Paso          : Str2     ;
     Paso2         : Str4     ;
     Nombre        : String   ;
     Archivo       : File     ;
     Datos         : Apuntador;
     Entra         :Boolean   ;
     Opta          ,
     Opta2         ,
     Opta3         : Char     ;
     Llave         ,
     I             :   Byte   ;
     Indice        ,
     Kk            ,
     Com           ,
     Memo          ,
     Volt          ,
     Ayuda         ,
     V_pp_L        ,
     V_pp_E        ,
     Dir_Or        ,
     Dir_And,
     Dir_V_L       ,
     Dir_V_E       ,
     Fallas        ,
     Pto_Vpp       ,
     Pto_Conf      ,
     Termino       ,
     Pto_Datos     ,
     Pto_Dir_H     ,
     Resultado     ,
     Inicio_ROM    ,
     Principio     ,
     Seg_Datos     ,
     Retraso       ,
     Latch_Dir_L,
     Latch_Dir_V,
     Pal_prog      ,
     Pagina_de_Inicio : Word     ;
     Numero,Result         : LongInt  ;
{*****************************************************************************}
{                 Funcion para Convertir un Numero en hexadecimal             }
Function Hexa(Num:Integer):Str2;
Var
   Dig1,Dig2 :Integer;
Begin
   Dig2:=Num Mod 16;
   Dig1:=Num Shr  4;
   If Dig2>9 Then
      Dig2:=Dig2+55
   Else
      Dig2:=Dig2+48;
   If Dig1>9 Then
```

```
      Dig1:=Dig1+55
    Else
      Dig1:=Dig1+48;
    Hexa:=Chr(Dig1)+Chr(Dig2);
End;
(*****************************************************************************)
Procedure Inicializa  ;External;
Procedure Check_Blank ;
External;
Procedure EscMem ;
External;
Procedure LeeMem ;External;
Procedure Compara;External;
Procedure Pantalla;
Begin
  ClrScr;
  TextColor(Amarillo);
  GotoXY(1,1);
  Writeln('          ┌─┐  ┌─┐        Coordinación  ');
  Writeln('          ┘┌┘  ┘┌┘      Instrumentación ');
  Writeln('          ┌┘└  ┌┘└.          (UNAM)       ');
  Writeln('          └─┘ ■ └─┘ ■                    ');
  GotoXY(1,1);
  TextColor(Blanco);
  Writeln(' ┌─┐');
  Writeln(' │ │');
  Writeln(' ├─┘');
  Writeln(' ┴  ■');
  GotoXY(1,12);
  TextColor(Cafe+7);
  Writeln(' ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌┐┌┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐');
  Writeln(' ├─┘ ├┬┘ │ │ │ ┐ ├┬┘ ├─┤ │││ ├─┤ │ │ │ │ ├┬┘');
  Writeln(' │   │ │ └─┘ └─┘ │ │ │ │ │││ │ │ └─┘ └─┘ │ │');
  GotoXY(35,19);
  TextColor(AzulClaro);
  Writeln('Autores');
  GotoXY(1,21);
  Writeln('       Sergio Arturo González Jácome - Jorge Francisco Montejo Pedrero');
  Writeln('                               Roberto Garza Trujillo');
  TextColor(negro);
  TextBackGround(Negro);
  Write('  ');
  TextColor(15+BLINK);
  GotoXY(73,24);
  Write('<Enter>');
  GotoXY(1,23);
  While not Keypressed do
      i:=0;
  Readln;
End;
(*****************************************************************************)
Function Donde(X:String):Byte;
Begin
  Donde:=(Pos(X,'0123456789ABCDEF')-1);
End;
```

```
(******************************************************************************)
{                   Funcion para Convertir un Numero en decimal               }
Function Decim(Num:Str2):Byte;
Begin
   Decim:=(Donde(Num[1]) Shl 4) Or Donde(Num[2]);
End;
(******************************************************************************)
{                   Funcion para Convertir un Numero en decimal               }
Function Decimal(Num:Str4):Word;
Begin
   Decimal:=(Donde(Num[1]) Shl 12)
          Or (Donde(Num[2]) Shl  8)
          Or (Donde(Num[3]) Shl  4)
          Or Donde(Num[4]);
End;
(******************************************************************************)
{     Funcion para proteccion en despliegue de Caracteres de Control          }
Function Car(C:Byte):Char;
Begin
  If C In [0,7,8,9,10,11,12,13,255] Then
     Car:='.'
  Else
     Car:=Chr(C);
End;
(******************************************************************************)
{                              Beep para Beep                                 }
Procedure Beep;
Begin
 Sound(300);
 Delay(500);
 NoSound;
End;
(******************************************************************************)
{              Funcion para Calculo de la posicion de un Caracter (ASCII)     }
Function Xb(I:Word):Byte;
Begin
   Xb:=((I Mod 256) Mod 16)+54;
End;
(******************************************************************************)
{            Funcion para Calculo de la posicion X  de un Caracter (HEX)      }
Function Xh(I:Word):Byte;
Begin
   Xh:=(((I Mod 256) Mod 16)+1)*3;
End;
(******************************************************************************)
{            Funcion para Calculo de la posicion Y  de un Caracter (HEX)      }
Function Y(I:Word):Byte;
Begin
   Y:=((I Mod 256) Shr 4)+3;
End;
(******************************************************************************)
{      Subrutina para informar acerca de la posicion del cursor  (HEX)        }
Procedure Mess(I:Word);
var
   Dig1,Dig2 : Word;
```

```
Begin
  Dig1:=Seg(I);Dig2:=Ofs(I);
  TextColor(Amarillo);
  GotoXY(1,1);
  Write('Posición : ',Hexa(Mem[Dig1:Dig2+1])+Hexa(Mem[Dig1:Dig2]));
  GotoXY(Xh(I)-1,Y(I));
  Write(char(16));
  GotoXY(Xh(I)+2,Y(I));
  Write(char(17));
End;
(*****************************************************************************)
(                       Subrutina para llenado de la pantalla                 )
Procedure Llena(Pagina_de_Inicio:Word);
var
   I          :Word  ;
   C          :Byte  ;
   Cadena,Ren:String;
Begin
    TextColor(Magenta);
    Window(3,3,50,19);
    Cadena:='';
    Ren   :='';
    For I:=Pagina_de_Inicio to Pagina_de_Inicio+255 Do
     Begin
       C:=Mem[Seg_Datos:I];
       Ren:=Ren+Hexa(C)+' ';
       Cadena:=Cadena+Car(C);
       If I Mod 16 =15  Then
          Begin
           Write(Ren);
           Ren:='';
          End;
    End;
    Window(54,3,69,19);
    Write(Cadena,Car(C));
    Window(1,1,80,25);
End;
(*****************************************************************************)
(                             Programa Principal                              )
(*****************************************************************************)


Begin
    Pantalla;
( Direccion de Puertos )
    Pto_Datos:=$0F78;
    Pto_Dir_H:=$0F79;
    Pto_Vpp  :=$0F7A;
    Pto_Conf :=$0F7B;
    Latch_Dir_L:=$0f7E;
    Latch_Dir_V:=$0f7d;
( Inicia Variables para Direccionamiento en Pascal )
    Inicializa;
    Indice:=0;
    Pagina_de_Inicio:= 0;
```

```
{ Inicia Valores de los Menus }
    Volt      :=   1;
    Llave     :=   1;
    Memo      :=   2;
{ Inicia Colores para menu principal }
    TextColor(Negro);
    TextBackGround(Negro);
    Clrscr;
{ Reserva 64 K Bytes para los Datos }
    Getmem(Datos,Mem64K);
    Seg_Datos:=Seg(Datos^);
    TextColor(Cafe);
    GotoXY(1,2);
    Writeln('┌──────────────────────────────────────────┬───────────────┐');
    For I:=3 to 18 Do
    Writeln('│                                          │               │');
    Writeln('└──────────────────────────────────────────┴───────────────┘');
    Fillchar(Datos^,Mem64K,Llenado);
    Mem[Seg_Datos:Mem64K]:=Llenado;
{    For Indice:=0 to $FF do Mem[Seg_Datos:Indice]:=0;
    For Indice:=$100 to $1FF do Mem[Seg_Datos:Indice]:=Indice-$100; }
    Indice:=0;
    Llene(Pagina_de_Inicio);
    While Llave<>10 Do
    Begin
      If Not (Llave In [1,2,3,8]) Then
         Llene(Pagina_de_Inicio);
      If Volt=1 Then
         V_pp_E:=1
      Else
         If Volt=2 Then
            V_pp_E:=2
         Else
            V_pp_E:=4;
      V_pp_L   :=$00;
      Dir_Or   :=$00;
      Dir_And  :=$2d; { 0010 1101  0 Si Vpp 1 Si PGM 0 11 VTermi 01 No OE }
      Dir_V_E  :=$2D; { 0010 1111  Vpp en 1 OE inhabilitado CE habilitado And h}
      Dir_V_L  :=$4C; { 0100 1100  Vpp en apag OE y CE habilitado And desh}
      GotoXY(25,25);
      Write('                              ');
      GotoXY(25,25);
      TextColor(Cyan+Blink);
      Case Memo of
           1:            { 2516 }
           Begin
             Dir_Or  :=$20;
             Dir_V_E :=$1b; { 0001 1011  Vpp en 23 OE y CE inhabilitados}
             Dir_V_L :=$58; { 0001 1000  Vpp en 23 OE y CE   habilitados}
             Numero  :=$07ff;
             If (Volt In [2,3]) Then Write('Voltaje Inferior al Estandar');
           End;
           2:            { 2716 }
           Begin
             Dir_Or  :=$20;
```

```
  Dir_V_E :=$1b; { 0001 1011  Vpp en 23 OE y CE inhabilitados}
  Dir_V_L :=$58; { 0101 1000  Vpp en 23 OE y CE   habilitados}
  Numero  :=$07ff;
  If (Volt In [2,3]) Then Write('Voltaje Inferior al Estandar');
End;
3:          { 2716 Esp }
Begin
  Dir_Or :=$20;  { Vcc en A13 }
  Dir_V_E:=$1b;  { 0001 1011  Vpp en 23 OE y CE inhabilitados}
  Dir_V_L:=$5C;  { 0101 1100  Vpp en 23 OE y CE   habilitados}
  Numero :=$07ff;
  If (Volt In [2,3]) Then Write('Voltaje Inferior al Estandar');
End;
4:          { 2532 }
Begin
  Dir_Or  :=$20;
  Dir_V_E :=$00; { * }
  Dir_V_L :=$fC;
  Numero  :=$0fff;
 If Volt=3 Then Write('Voltaje Inferior al Estandar');
 If Volt=1 Then Write('Voltaje Superior al Estandar');
End;
5:          { 2732 }
Begin
  Dir_Or  :=$20;
  Dir_V_E :=$37;
  Dir_V_L :=$7C;
  Numero  :=$0FFF;
 If Volt=1 Then Write('Voltaje Inferior al Estandar');
 If Volt=3 Then Write('Voltaje Superior al Estandar');
End;
6:          { 2764 }
Begin
  Dir_Or :=$40;{ 1  }
  Dir_V_L:=$2C;{ 010 1100}
  Dir_V_E:=$2d;{ 110 1101}
  Dir_And:=$0D;{ 100 1101}
  V_pp_L :=V_pp_E;
  Numero :=$1FFF;
  If (Volt In [1,2]) Then Write('Voltaje Superior al Estandar');
End;
7:          { 27128 }
Begin
  Dir_Or :=$40;
  Dir_V_E:=$2D;
  Dir_And:=$0d;
  Dir_V_L:=$2c;
  V_pp_L :=0  { V_pp_E } ;
  Numero :=$3fff;
  If (Volt In [1,2]) Then Write('Voltaje Superior al Estandar');
End;
8:          { 27256 }
 Begin
  V_pp_L   :=$00;
  Dir_Or   :=$00;
```

```
            Dir_V_L  :=$6c;  { 0110 1100 }
            Dir_V_E  :=$2f;  { 0010 1111 }
            Dir_And  :=$2D;  { 0010 1101 }
            Numero :=$7fff;
            If (Volt In[1,2]) Then Write('Voltaje Superior al Estandar');
           End;
          9:          { 27512 }
           Begin
            V_pp_L   :=$00;
            Dir_Or   :=$00;
            Dir_V_L  :=$7c;  { 0111 1100 }
            Dir_V_E  :=$37;  { 0011 0111 }
            Dir_And  :=$35;  { 0011 0101 }
            Numero :=$ffff;
            If (Volt In [1,2]) Then Write('Voltaje Superior al Estandar');
           End;
      End;
      NoSound;
      Opta:=#0;
      TextColor(AzulClaro);
      TextBackground(2);
      GotoXY(Xh(Indice),Y(Indice));
      Write(Hexa(Mem[Seg_Datos:Pagina_de_Inicio+Indice]));
      TextBackground(Azul);
      Hexs(Pagina_de_Inicio+Indice);
      GotoXY(1,21);
      Ayuda:=Numero;
{ Presentacion y captura de opcion en menu principal }
      Writeln(Memor[Memo],' ',Vpp[Volt],' Volts ',Hexa(Ayuda Shr 8),Hexa((Ayuda) And $FF),' KBytes ');
      While Not (Opta In [Ins,Enter,Del,Esc]) Do
      Begin
          TextBackground(AzulClaro);
          TextColor(Cafe);
          GotoXY(1,23);
          Write(Opciones);
          TextColor(Rojo);
          TextBackground(LightGrey);
          GotoXY(7*(Llave-1)+3,23);
          Write(Opcion[Llave]);
          TextColor(Amarillo);
          GotoXY(7*(Llave-1)+2,23);
          Write(Chr(16));
          GotoXY(7*(Llave)+2,23);
          Write(Chr(17));
          GotoXY(79,24);
          TextBackground(Azul);
          Opta:=#0;
          While Not (Opta In [Right,Left,Up,Down,PU,PD,Ins,Enter,Del,Esc,'0'..'9']) Do
          Begin
               Opta:=ReadKey;
               If Opta=#0 Then
                  Begin
                    Opta:=ReadKey;
                    If Not (Opta In [Right,Left,Up,Down,PU,PD,Ins,Del,Enter,Esc]) Then
                       Opta:=' ';
```

```
                End;
          End;
          If (Llave>2) and (Opta in [Esc,Del]) Then Opta:=' ';
          Case Opta Of
            Right,Up,PU:
               If Llave=10 Then
                  Llave:=1
               Else
                  Inc(Llave);
            Left,Down,PD:
               If Llave=1 Then
                  Llave:=10
               Else
                  Dec(Llave);
            '0'..'9':
               Llave:=Pos(Opta,'1234567890');
          End;
      End;
{ Ejecucion de las Seleccion hecha en el menu anterior }
    TextColor(Magenta);
    TextBackground(Negro);
    Case Llave Of
         1:
          Begin
              If (Opta In[Ins,Enter]) Then
                If Nemo=9 Then
                   Nemo:=1
                Else
                   Inc(Nemo)
              Else
                If Nemo=1 Then
                   Nemo:=9
                Else
                   Dec(Nemo);
          End;
         2:
          Begin
              If (Opta In[Ins,Enter]) Then
                If Volt=3 Then
                   Volt:=1
                Else
                   Inc(Volt)
              Else
                If Volt=1 Then
                   Volt:=3
                Else
                   Dec(Volt);
          End;
         3:
          Begin
          Nombre:='';
          Entra:=true;
          While Entra do
          Begin
            TextColor(Magenta);
```

```
TextBackground(Negro);
GotoXY(20,1);
Write('¿Cuál Es el Archivo?  (Salir X) ');
Readln(Nombre);
GotoXY(20,1);
Write('                                                       ');
If (nombre<>'X') and (nombre<>'x') Then
   Begin
   {$I-}
   Assign(Archivo,Nombre);
   Reset(Archivo,1);
   {$I+}
   If IOResult=0 Then
     Begin
      Principio:=0;
      GotoXY(40,1);
      Writeln('Inicio  (0000)');
      Paso2:='    ';
      For I:=1 to 4 Do
      Begin
        GotoXY(55+I,1);
        While Not (Paso2[I] In ['0'..'9','a'..'f','A'..'F',Del,Enter]) do
              Paso2[I]:=ReadKey;
        If Paso2[I] In ['0'..'9','a'..'f','A'..'F'] Then
           Begin
             Paso2[I]:=Upcase(Paso2[I]);
             Writeln(Paso2[I]);
           End
        Else
           If Paso2[I]=Enter Then
              Paso2:=Copy('0000',1,5-I)+Copy(Paso2,1,I-1)
           Else
              If I>1 Then I:=I-1
      End;
      Principio:=Decimal(Paso2);
      Blockread(Archivo,Ptr(Seg_Datos,Principio)^,Mem64K-Principio,Resultado);
      If (Resultado<>0) then
         Blockread(Archivo,Ptr(Seg_Datos,Mem64K)^,1,Resultado);
      Close(Archivo);
      GotoXY(20,1);
      Write('                                                       ');
      Entra:=False;
     End
   Else
     Begin
      GotoXY(20,1);
      Writeln('No Existe el archivo                          ');
      delay(1000);
     End;
   End
Else
   Begin
   Entra:=False;
End;
GotoXY(20,1);
```

```
  Write('                                          ');
 End;
 Llena(Pagina_de_Inicio);
 End;
4:
 Begin
   Nombre:='';
   Entra:=true;
   While Entra do
     Begin
       TextColor(Magenta);
       TextBackground(Negro);
       GotoXY(20,1);
       Write('¿A cuál Archivo?  [Salir X] ');
       Readln(Nombre);
       For i:=1 to Length(Nombre) do
           Nombre[i]:=UpCase(Nombre[i]);
       GotoXY(20,1);
       Write('                                          ');
       If nombre[1]<>'X' Then
         Begin
           {$I-}
           Assign(Archivo,Nombre);
           Reset(Archivo,1);
           {$I+}
           If IOResult=0 Then
             Begin
               GotoXY(20,1);
               Write('¿Quiere sobreescribir? [S/N] ');
               Paso2[1]:=UpCase(ReadKey);
               If Paso2[1]='S' Then
                  entra:=False;
             End
           Else
               If nombre<>'' Then
                  Entra:=False;
           GotoXY(20,1);
           Write('                                          ');
       End;
       If (nombre[1]<>'X') and (Paso2[1]<>'N') Then
         Begin
           GotoXY(20,1);
           Write('¿Cuántos Bytes? ');
           Paso2:='    ';
           For I:=1 to 4 Do
               Begin
                 GotoXY(55+I,1);
                 While Not (Paso2[I] In ['0'..'9','a'..'f','A'..'F',Del,Enter,Ins,Esc]) do
                       Paso2[I]:=ReadKey;
                 If Paso2[I] In ['0'..'9','a'..'f','A'..'F'] Then
                    Begin
                      Paso2[I]:=Upcase(Paso2[I]);
                      Writeln(Paso2[I]);
                    End
                 Else
```

```
                    If Paso2[I] In [Enter,Ins] Then
                       Paso2:=Copy('0000',1,5-I)+Copy(Paso2,1,I-1)
                    Else
                       If I>1 Then Paso2[I]:=' ';
              End;
              Termino:=Decimal(Paso2);
              GotoXY(40,1);
              Writeln('                                  ');
              Assign(Archivo,Nombre);
              ReWrite(Archivo,1);
              BlockWrite(Archivo,Datos^,Termino,Resultado);
              blockwrite(Archivo,Ptr(Seg_Datos,Principio+TERMINO)^,1,Resultado);
              Close(Archivo);
           End;
           If nombre[1]='X' Then Entra:=False;
           GotoXY(20,1);
           Write('                                    ');
        End;
       End;
      5:
        Begin
          GotoXY(40,1);
          TextColor(Verde+blink);
          Writeln('Lectura de Memoria ');
          Termino   :=Numero;
          Principio :=     0;
          LeeMem;
          GotoXY(40,1);
          TextColor(Blanco);
          Writeln('                    ');
        End;
      6:
       Begin
        Entra    :=True;
        Fallas   :=0;
        Principio:=0;
        While (Termino-Principio>Numero) Or Entra Do
        Begin
          Entra:=False;
          GotoXY(17,1);
          Writeln('Entre (0000) y (',Hexa((Numero And $ff00) Shr 8),Hexa(Numero And $00ff),')');
          GotoXY(40,1);
          Writeln('Inicio  (0000)');
          Paso2:='    ';
          For I:=1 to 4 Do
          Begin
            GotoXY(55+I,1);
            While Not (Paso2[I] In ['0'..'9','a'..'f','A'..'F',Del,Enter]) do
                Paso2[I]:=ReadKey;
            If Paso2[I] In ['0'..'9','a'..'f','A'..'F'] Then
               Begin
                 Paso2[I]:=Upcase(Paso2[I]);
                 Writeln(Paso2[I]);
               End
            Else
```

```
      If Paso2[I]=Enter Then
         Paso2:=Copy('0000',1,5-I)+Copy(Paso2,1,I-1)
      Else
         If I>1 Then I:=I-1
  End;
  Principio:=Decimal(Paso2);
  GotoXY(40,1);
  Writeln('                                            ');
  GotoXY(40,1);
  Writeln('Termino (',Hexa(((Numero+principio) And $ff00) Shr 8),
                      Hexa((Numero+principio) And $00ff),')');
  Paso2:='    ';
  For I:=1 to 4 Do
    Begin
     GotoXY(55+I,1);
     While Not (Paso2[I] In ['0'..'9','A'..'F',Del,Enter]) do
        Paso2[I]:=Upcase(ReadKey);
     If Paso2[I]=Enter Then
          If I<>1 Then
             Paso2:=Copy('0000',1,5-I)+Copy(Paso2,1,I-1)
          Else
             Paso2:=Hexa(((Numero+principio) And $ff00) Shr 8)+Hexa((Numero+principio) And $00ff)
     Else
        If Paso2[I]<>Del Then
             Writeln(Paso2[I])
        Else
          Begin
             Paso2[I]:=' ';
             I:=I-1;
          End
  End;
  Termino:=Decimal(Paso2);
  GotoXY(17,1);
  Writeln('                                                              ');
 End;
 If Termino-Principio<>0 Then
  Begin
    GotoXY(40,1);
    Writeln('Inicio en ROM (',Hexa(((principio) And $ff00) Shr 8),
                      Hexa((principio) And $00ff),')');
    Paso2:='    ';
    For I:=1 to 4 Do
      Begin
       GotoXY(65+I,1);
       While Not (Paso2[I] In ['0'..'9','A'..'F',Del,Esc,Enter,Ins]) do
          Paso2[I]:=Upcase(ReadKey);
       If Paso2[I] In [Enter,Ins] Then
            If I<>1 Then
               Paso2:=Copy('0000',1,5-I)+Copy(Paso2,1,I-1)
            Else
               Paso2:=Hexa((principio And $ff00) Shr 8)+Hexa(principio And $00ff)
       Else
          If Not (Paso2[I] In [Del,Esc]) Then
               Writeln(Paso2[I])
          Else
```

```
              Begin
                 Paso2[I]:=' ';
                 I:=I-1;
              End
        End;
        GotoXY(17,1);
        Writeln('                                                        ');
        Inicio_ROM:=Decimal(Paso2);
        TextColor(Blanco);
        Retraso :=1;
        GotoXY(23,1);
        Sound(5000);
        GotoXY(25,1);
        TextColor(Verde+blink);
        Write('    Programando . . .    ');
        GotoXY(25,1);
        EscMem;
        If Retraso < $FF Then
           Write('                              ',hexa(Retraso))
        Else
           Write ('Fallo la Programacion');
        Delay(1000);
        GotoXY(25,1);
        Write('                              ');
        NoSound;
      End;
    End;
  7:
    Begin
     Fallas:=0;
     Principio:=0;
     Termino:=Numero;
     Check_Blank;
     If (Fallas<>0) or ((Termino=1) and (Memo=9))  Then
        Begin
        GotoXY(25,1);
        Writeln('No. de Localidades con Informacion  ',hexa((Fallas shr 8) and $FF),hexa(Fallas and $FF));
        GotoXY(60,1);
        If ((Termino=1) and (Memo=9) and (fallas=0)) then Writeln('1');
        Delay (1000);
        GotoXY(25,1);
        Writeln('                                        ');
        End
     Else
        Begin
        GotoXY(25,1);
        Writeln('Completamente Borrada');
        Delay (1000);
        GotoXY(25,1);
        Writeln('                          ');
        End;
    End;
  8:
  Begin
   Opta2:=#0;
```

```
 While Not(Opta2 In [Del,Esc]) Do
 Begin
   If Indice>255 Then
   Begin
 Pagina_de_Inicio:=Pagina_de_Inicio+Indice;
 Indice:=0;
 Llena(Pagina_de_Inicio);
End( If );
Opta2:=#0;
   TextColor(Cyan);
   GotoXY(Xh(Indice),Y(Indice));
   Write(Hexa(Mem[Seg_Datos:Pagina_de_Inicio+Indice]));
   GotoXY(Xb(Indice),Y(Indice));
   Write(Car(Mem[Seg_Datos:Pagina_de_Inicio+Indice]));
   GotoXY(80,23);
   While Not (Opta2 In [Right,Left,Up,Down,PU,PD,Ins,Del,Enter,Esc,Fin,Home]) Do
      Begin
        Opta2:=ReadKey;
        If Opta2=#0 Then Opta2:=ReadKey;
      End;
   TextColor(Magenta);
   GotoXY(Xh(Indice)-1,Y(Indice));
   Write(' ',Hexa(Mem[Seg_Datos:Pagina_de_Inicio+Indice]),' ');
   GotoXY(Xb(Indice),Y(Indice));
   Write(Car(Mem[Seg_Datos:Pagina_de_Inicio+Indice]));
   GotoXY(80,23);
   Case Opta2 Of
   Up:
     Case Indice Of
       0:
        Begin
             Indice:=$00FF;
             Pagina_de_Inicio:=Pagina_de_Inicio-$0100;
             Llena(Pagina_de_Inicio);
        End;
       1..15:
             Indice:=Indice+239;
       16..255:
             Indice:=Indice-16;
     End;
   Down:
        Case Indice Of
       255:
            Inc(Indice);
        0..239:
            Indice:=Indice+16;
       240..255:
            Indice:=Indice-239;
     End;
   Left:
         If Indice=0 Then
            Begin
             Indice:=$FF;
             Pagina_de_Inicio:=Pagina_de_Inicio-$100;
             Llena(Pagina_de_Inicio);
```

```
        End
       Else
        Dec(Indice);
Right:
      If Indice=255 Then
        Begin
         Indice:=0;
         Pagina_de_Inicio:=Pagina_de_Inicio+$100;
         Llena(Pagina_de_Inicio);
        End
       Else
        Inc(Indice);
PU:
   Begin
      Pagina_de_Inicio:=Pagina_de_Inicio+256;
      Llena(Pagina_de_Inicio);
   End;
PD:
   Begin
    Pagina_de_Inicio:=Pagina_de_Inicio-256;
    Llena(Pagina_de_Inicio);
   End;
Home:
   Begin
     Pagina_de_Inicio:=0;
     Indice       :=0;
     Llena(Pagina_de_Inicio);
   End;
Fin:
   Begin
     Pagina_de_Inicio:=Numero And $FF00;
     Indice       :=$FF;
     Llena(Pagina_de_Inicio);
   End;
Enter,Ins:
   Begin
     Opta3:=' ';
     GotoXY(40,1);
     Write('** Edicion ** ');
     Mesa(Pagina_de_Inicio+Indice);
     While Not (Opta3 In [Del,Esc]) Do
        Begin
           TextColor(Blanco);
           GotoXY(Xh(Indice),Y(Indice));
           Write(Hexa(Mem[Seg_Datos:Pagina_de_Inicio+Indice]));
           GotoXY(Xb(Indice),Y(Indice));
           Write(Car(Mem[Seg_Datos:Pagina_de_Inicio+Indice]));
           GotoXY(80,2);
           Opta3:=#0;
           Paso:=Hexa(Mem[Seg_Datos:Pagina_de_Inicio+Indice]);
           While Not (Opta3 In ['0'..'9','A'..'F',Del,Esc,Enter,Ins]) Do
              Begin
                Opta3:=Upcase(Readkey);
                GotoXY(Xh(Indice),Y(Indice));
                If Opta3 In ['0'..'9','A'..'F',Ins,Enter] Then
```

```
                    Begin
                      If Not(Opta3 In [Enter,Ins]) Then Paso[1]:=Opta3;
                      Write(Paso[1]);
                      GotoXY(Xb(Indice),Y(Indice));
                      Write(Car(Decim(Paso)));
                      GotoXY(80,2);
                    End
                  Else
                    Begin
                       If Opta3=#0 Then Opta3:=ReadKey;
                       Beep;
                    End;
                End;
              If Not (Opta3 In [Del,Esc,Ins,Enter]) Then
                Begin
                  Opta3:=' ';
                  While Not (Opta3 In ['0'..'9','A'..'F',Enter,Ins,Esc,Del]) Do
                        If Not (Opta3 In ['0'..'9','A'..'F',Enter,Ins,Esc,Del]) Then
                           Opta3:=Upcase(ReadKey)
                        Else
                            Beep;
                  If Opta3 In ['0'..'9','A'..'F'] Then
                     Paso[2]:=Opta3;
                  If Opta3 In [Esc,Del] Then
                     Begin
                       Opta3:=Enter;
                       Paso:=Hexa(Mem[Seg_Datos:Pagina_de_Inicio+Indice]);
                     End;
                  GotoXY(Xh(Indice),Y(Indice));
                  Write(Paso);
              End;
                  Mem[Seg_Datos:Pagina_de_Inicio+Indice]:=Decim(Paso);
                  TextColor(Magenta);
                  GotoXY(Xh(Indice)-1,Y(Indice));
                  Write(' ',Hexa(Mem[Seg_Datos:Pagina_de_Inicio+Indice]),' ');
                  GotoXY(Xb(Indice),Y(Indice));
                  Write(Car(Mem[Seg_Datos:Pagina_de_Inicio+Indice]));
                  GotoXY(80,2);
                  Inc(Indice);
                  If Indice>255 Then
                     Begin
                       Pagina_de_Inicio:=Pagina_de_Inicio+Indice;
                       Indice:=0;
                       Llena(Pagina_de_Inicio);
                     End;
                  Mess(Pagina_de_Inicio+Indice);
          End;
      End;
  End;
  Mess(Pagina_de_Inicio+Indice);
  TextColor(Negro);
  GotoXY(40,1);
  Write('                                        ');
  GotoXY(80,23);
End;
```

```
End;
     9:
      Begin
       Entra    :=True;
       Fallas   :=0;
       Principio:=0;
       While (Termino-Principio>Numero) Or Entra Do
       Begin
          Entra:=False;
          GotoXY(17,1);
          Writeln('Entre (0000) y (',Hexa((Numero And $ff00) Shr 8),Hexa(Numero And $00ff),')');
          GotoXY(40,1);
          Writeln('Inicio  (0000)');
          Paso2:='    ';
          For I:=1 to 4 Do
          Begin
            GotoXY(55+I,1);
            While Not (Paso2[I] In ['0'..'9','a'..'f','A'..'F',Del,Enter]) do
                Paso2[I]:=ReadKey;
            If Paso2[I] In ['0'..'9','a'..'f','A'..'F'] Then
               Begin
                Paso2[I]:=Upcase(Paso2[I]);
                Writeln(Paso2[I]);
               End
            Else
                If Paso2[I]=Enter Then
                   Paso2:=Copy('0000',1,5-I)+Copy(Paso2,1,I-1)
                Else
                   If I>1 Then I:=I-1
          End;
          Principio:=Decimal(Paso2);
          GotoXY(40,1);
          Writeln('                                        ');
          GotoXY(40,1);
          Writeln('Termino (',Hexa(((Numero+principio) And $ff00) Shr 8),
                              Hexa((Numero+principio) And $00ff),')');
          Paso2:='    ';
          For I:=1 to 4 Do
            Begin
             GotoXY(55+I,1);
             While Not (Paso2[I] In ['0'..'9','A'..'F',Del,Enter]) do
                Paso2[I]:=Upcase(ReadKey);
             If Paso2[I]=Enter Then
                  If I<>1 Then
                     Paso2:=Copy('0000',1,5-I)+Copy(Paso2,1,I-1)
                 Else
                     Paso2:=Hexa(((Numero+principio) And $ff00) Shr 8)+Hexa((Numero+principio) And $00ff)
             Else
                If Paso2[I]<>Del Then
                     Writeln(Paso2[I])
                Else
                  Begin
                     Paso2[I]:=' ';
                     I:=I-1;
                  End
```

```
            End;
            Termino:=Decimal(Paso2);
         End;
         GotoXY(17,1);
         Writeln('                                                                    ');
         Compara;
         GotoXY(25,1);
         If Fallas<>0 then
            Writeln('Falló en la localidad :',Hexa(((principio) And $ff00) Shr 8)+Hexa((principio) And $00ff),
                      ' Lee ',Hexa((termino) And $00ff))
         Else
            Writeln('Comparación exitosa');
         Delay(3000);
         GotoXY(17,1);
         Writeln('                                                                    ');
      End;
    End;
 End;
 Clrscr;
End.
(*
Data Segment Word Public
      Extrn V_pp_L:Word
      Extrn V_pp_E:Word
      Extrn Dir_Or:Word
      Extrn Dir_And:Word
      Extrn Dir_V_L:Word
      Extrn Dir_V_E:Word
      Extrn Termino:Word
      Extrn Principio:Word
      Extrn Inicio_ROM:Word
      Extrn Seg_Datos:Word
      Extrn Retraso:Word
      Extrn Fallas:Word
      Extrn Latch_Dir_V:Word
      Extrn Latch_Dir_L:Word
      Extrn Pto_Datos:Word
      Extrn Pto_Dir_H:Word
      Extrn Pto_Vpp:Word
      Extrn Pto_Conf:Word
      Extrn Pto_Conf:Word
      Extrn Memo:Word
Data EndS
Code Segment Byte Public
     Assume  Cs:Code,Ds:Data
     Public  EscMem,LeeMem,Inicialize,Check_Blank,Compara
Buzzer  equ 61h
Muestra equ 62h
FF      equ 0FFh
FB      Equ 0FBh
FD      Equ 0Fdh
cero    Equ 00
Apaga Proc
      In   Al,Buzzer      ; Verif Buzzer
      And  Al,F0
```

```
      Out  Buzzer,Al        ; Apaga Buzzer
      Ret
Apaga      EndP
Detecta Proc
      In   Al,Muestra
      Mov  Ah,Al
Ciclo2:
      In   Al,Muestra
      Cmp  Al,Ah
      Je   Ciclo2
      Mov  Bh,Ah
      Ret
Detecta    EndP
Retrasos Proc
Ciclo1:
      Mov  Ah,Bh
      In   Al,Muestra ;  1er
Primero1:
      In   Al,Muestra ;  1er
      Cmp  Al,Ah      ; Flanco
      Jne  Primero1
Primero2:
      In   Al,Muestra ;  2do
      Cmp  Al,Ah      ; Flanco
      Je   Primero2
      Loop Ciclo1
Ret
Retrasos  EndP
DatosIniciales Proc
      Mov  Es,Seg_Datos    ; Segmento Datos
      Mov  Di,Termino      ; Offset Termino
      Mov  Si,Principio
      Ret
DatosIniciales      EndP
Modo_Lee Proc
      Mov  Dx,Pto_Vpp
      Mov  Ax,V_pp_L
      Out  Dx,Al
      Mov  Dx,Latch_Dir_V
      Mov  Ax,Dir_V_L
      Or   Ax,0001
      Out  Dx,Al
      Mov  Ax,Dir_V_L
      Out  Dx,Ax
      Ret
Modo_Lee     EndP
Modo_Prog Proc
      Mov  Dx,Pto_Vpp
      Mov  Ax,V_pp_E
      Out  Dx,Al
      Mov  Dx,Latch_Dir_V
      Mov  Ax,Dir_V_E
      Out  Dx,Al
      Ret
Modo_Prog  EndP
```

```
Dir_Alta_E Proc
      Mov  Dx,Latch_Dir_V
      Mov  Ax,Dir_V_E
      Out  Dx,Al
      Mov  Ax,Inicio_ROM  ; Dirl-> Latch_Dir_L
      Mov  Dx,Pto_Dir_H
      Mov  Al,ah
      Or   Ax,Dir_Or
      Out  Dx,Al
      Ret
Dir_Alta_E EndP
Dir_Alta_L Proc
      Mov  Dx,Pto_Dir_H
      Mov  Al,ah
      Or   Ax,Dir_Or
      Out  Dx,Al
      Ret
Dir_Alta_L EndP
EscMem Proc Near
      Call Apaga
      Call Inicializa
      Call Modo_Lee
      Call DatosIniciales
      Dec  Si
Inicio:
      Inc  Si
      Mov  Al,Es:[Si]     ; Dat -> Al
      Cmp  Al,FF          ; No Programa
      Je   No_programa    ; en localidades
Programa:                 ; que tengan FF
      Mov  Cx,Retraso     ; Iniciacion de Contador
      Mov  Ax,Inicio_ROM  ; Dirl-> Latch_Dir_L
      Mov  Dx,Latch_Dir_L
      Out  Dx,Al
      Call Detecta
      Call Dir_Alta_E
        Mov  Dx,Pto_Datos   ; Datos -> Puerto Datos
        Mov  Al,Es:[Si]     ; Dat -> Al
        Out  Dx,Al
      Call Modo_Prog
      Cmp  Memo,3
       Jle  No_Ce
       Mov  Dx,Latch_Dir_V
       Mov  Ax,Dir_V_E
      Cmp  Memo,5
      Jg PGM
      And  Al,FD
      Jmp Pulso
PGM:
      Mov Ax,Dir_And
Pulso:
      Out  Dx,Al
      Out  Dx,Al
      Out  Dx,Al
No_Ce:
```

```
        Call Retrasos
        Call Modo_Lee
        Mov  Ax,Inicio_ROM  ; Dirl-> Latch_Dir_L
        Mov  Dx,Latch_Dir_L
        Out  Dx,Al
        Out  Dx,Al
        Out  Dx,Al
        Call Dir_Alta_L
        Mov  Dx,Pto_Datos
        In   Al,Dx          ; Comparo Dato escrito con
        In   Al,Dx          ; Comparo Dato escrito con
        In   Al,Dx          ; Comparo Dato escrito con
        Cmp  Al,Es:[Si]     ; original
        Je   Sigue          ; Si esta igual salgo
        Inc  Retraso        ; Incrementa el Retraso
        Cmp  Retraso,FF     ; Verifica limite de pulso
        Je   Final          ; de programacion
        Jmp  Programa       ; Regresa A Ciclo
Sigue:
        cmp  Fallas,00h
        Jne  No_Programa
        Inc   Fallas
        Shl   Retraso,1h
        Jmp   Programa
No_Programa:
        Cmp  Si,Di          ; ¿Se llego Al final?
        Je   Final          ; No Regreso
        Inc  Inicio_ROM     ; Sig Dir en ROM
        Inc  Principio      ; Sig Dir en RAM
        Jmp  Inicio
Final:
        Sti
        Ret
EscMem     EndP
LeeMem Proc Near
        Call Inicializa
        Call Modo_Lee
        Call DatosIniciales
        dec  Si
Lee_Otro:
        Inc  Si
        Mov  Ax,Si
        Mov  Dx,Latch_Dir_L ; Dir l-> Latch_Dir_L
        Out  Dx,Al
        Call Dir_Alta_L
        Mov  Dx,Pto_Datos   ; Pasa Datos a RAM
        In   Al,Dx
        In   Al,Dx          ; con el registro
        Mov  Es:[Si],Al     ; intermedio Al
        Cmp  Si,Di          ; ¿Se llego Al final
        Jne  Lee_Otro       ; No Regreso
        Ret
LeeMem EndP
Check_Blank Proc Near
        Call Inicializa
```

```
        Call Modo_Lee
        Call DatosIniciales
        Dec  SI
Checa_Otro:
        Inc  SI
        Mov  Ax,SI
        Mov  Dx,Latch_Dir_L   ; Dir L-> Latch_Dir_L
        Out  Dx,Al
        Call Dir_Alta_L
        Mov  Dx,Pto_Datos     ; Datos -> Al
        In   Al,Dx
        Cmp  Al,FF            ; Compara Datos con FF
        Je   Sigue2
        Inc  Fallas           ; Fallas+1 -> Fallas
        Jnc  Sigue2
        Mov  Ax,1
        Mov  Termino,Ax
Sigue2:
        Cmp  SI,DI            ; ¿Se llego al final?
        Jne  Checa_Otro       ; No Regreso
        Ret
Check_Blank EndP
Compara Proc Near
        Call Inicializa
        Call Modo_Lee
        Call DatosIniciales
        Dec  SI
Chec_Otro:
        Inc  SI
        Mov  Ax,SI
        Mov  Dx,Latch_Dir_L   ; Dir L-> Latch_Dir_L
        Out  Dx,Al
        Call Dir_Alta_L
        Mov  Dx,Pto_Datos     ; Datos -> Al
        In   Al,Dx
        In   Al,Dx
        Cmp  Al,Es:[SI]      ; original
        Je   Sigue3
        Mov  Fallas,01h      ; Fallo=True
        Mov  Principio,SI
        Mov  Ah,0h
        Mov  Termino,Ax
        Jmp  Fin_comp
Sigue3:
        Cmp  SI,DI            ; ¿Se llego al final?
        Jne  Chec_Otro       ; No Regreso
Fin_comp:
        Ret
Compara EndP
Inicializa Proc Near
        Cli
        Mov  Al,FB          ; Configura Puerto A como Bidireccional
        Mov  Dx,Pto_Conf
        Out  Dx,Al
        Mov  Al,Cero
```

```
      Mov  Dx,Pto_Vpp
      Out  Dx,Al
      Mov  Dx,Latch_Dir_V
      Out  Dx,Al
      Mov  Dx,Pto_Vpp
      Out  Dx,Al
      Mov  Dx,Pto_Dir_H
      Out  Dx,Al
      Mov  Dx,Pto_Datos
      Out  Dx,Al
      Mov  Dx,Latch_Dir_L
      Out  Dx,Al
      Ret
Inicializa EndP
Code       EndS
        End
*)
```

BIBLIOGRAFIA

1.- Brenner, Robert C.-IBM PC ADVANCED TROUBLESHOOTING AND REPAIR.

Howard W. Sams and Company

Estados Unidos 1988

2.- Duncan, Ray.- IBM ROM BIOS

Microsoft Press

Estados Unidos 1988

3.- Hnatek, Eugene R.-A USER'S HANDBOOK OF SEMICONDUCTOR MEMORIES.

John Wiley and Sons

Estados Unidos 1981

4.- IBM PC-XT MODEL 5160-086 COMPUTER.

Howard W. Sams and Company

Estados Unidos 1986

5.- MEMORY COMPONENTS HANDBOOK

INTEL

Estados Unidos 1988

6.- MICROPROCESSOR AND PERIPHERAL HANDBOOK VOLUME 1

MICROPROCESSOR

INTEL

Estados Unidos 1988

7.- LINEAR AND INTERFACE INTEGRATED CIRCUITS.

Motorola, Inc.

Estados Unidos 1985

8.- LINEAR INTEGRARED CIRCUITS

Texas Instrument

Estados Unidos 1977

9.- PERIPHERAL DESIGN HANDBOOK.

INTEL

Estados Unidos 1981

10.- TTL HANDBOOK

Texas Instrument

Estados Unidos 1976