CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL

INSTITUTO POLITECNICO NACIONAL

DEPARTAMENTO DE INGENIERIA ELECTRICA

SECCION DE COMPUTACION

DISEÑO Y CONSTRUCCION DE UN COPROCESADOR PARA COMUNICACIONES

Tesis que presenta el Ing. Luis Martín Rojas Cárdenas para obtener el grado de MAESTRO EN CIENCIAS en la especialidad de INGENIERIA ELECTRICA con opción en COMPUTACION.

Trabajo dirigido por el Dr. Jan Janecek Hyan.

México D.F.
junio de 1990

Becario de
CONACYT

A MIS PADRES LUIS Y OFELIA CON UN PROFUNDO RESPETO Y CARIÑO

AGRADECIMIENTOS

Mi agradecimiento muy especial al doctor Jan Janecek por sus observaciones y sugerencias en el presente trabajo.

Al CONACYT y al CINVESTAV-SECCION DE COMPUTACION por la ayuda brindada para la obtención del grado.

I N D I C E

INTRODUCCION

Desde la aparición de las primeras computadoras, la intención de comunicarlas se vislumbró como el objetivo de explotar todo el amplio horizonte de posibilidades que una red de computadoras puede proveer. Esto se logró, pero en un principio, esta tecnología estuvo reservada sólo para muy grandes compañías, universidades o para las administraciones públicas, ya que sólo ellos podían adquirir estos costosos aparatos.

En la actualidad, con el gran salto tecnológico en el campo de la física del estado sólido, la posibilidad de adquirir este tipo de máquinas aumentó a tal grado, que hoy en día cualquier empresa de medianos recursos y algunos particulares puede adquirir su propia computadora.

El uso de las computadoras personales está floreciendo en el contexto de las comunicaciones, principalmente en el área de las redes de computadoras. Las posibilidades que ofrece el uso de esta tecnología son bastas y eficientes, a un costo aceptable.

Una computadora personal del tipo PC puede efectuar diversas aplicaciones dentro del campo de las redes de computadoras, con la ayuda de dispositivos adicionales que expanden sus pobres capacidades de comunicación y, por supuesto, un conjunto de protocolos de comunicación.

Con respecto a nuestro país, la infraestructura existente dentro del campo de las redes de computadoras está asentada en la red TELEPAC y en la red telefónica. Un protocolo de acceso a estas redes es el X.25. Ahora, con respecto a las nuevas tendencias de la tecnología, dentro de esta área, TELMEX cuenta ya con una maqueta de una Red Digital de Servicios Integrados, con ayuda de la empresa INDETEL. Otra empresa que comienza a hacer pruebas de RDSI es ERICSSON. Estas pruebas están encaminadas a la comercialización de estos.

Por nuestra parte, para la realización de la tesis, tomamos en cuenta la infraestructura existente y es así que como una alternativa para aprovechar las redes de datos, decidimos diseñar y construir una tarjeta coprocesadora que permita a una

microcomputadora del tipo IBM PC/XT/AT enlazarse con una red de datos, como la TELEPAC y la RDSI.

La parte que a esta tesis corresponde, es la del diseño y construcción del nivel 1 o capa física, la realización del software de inicialización y de algunas herramientas de trabajo, tales como un cargador-monitor, herramientas que dejan lista a la tarjeta para continuar con los demás niveles del modelo OSI.

1. EL CONTEXTO DE LAS REDES.

La mezcla de la computación y las comunicaciones ha tenido una profunda influencia en la forma en la que los sistemas de cómputo son organizados. El concepto de "centro de cómputo" como una sala con una gran computadora, a la cual los usuarios llevan su trabajo para ser procesado, está siendo rápidamente obsoleto.

El antiguo modelo de una computadora atendiendo todas las necesidades computacionales de la organización, está siendo rápidamente remplazado por uno en el cual un gran número de computadoras separadas, pero interconectadas, realizan una rutina. Estos sistemas son llamados **redes de computadoras**.

Muchas compañías tienen un número sustancial de computadoras en operación, frecuentemente separadas una de otra por una gran distancia. Por ejemplo, una compañía con muchas fábricas puede tener localizada una computadora en cada una para almacenar inventarios, monitorear la productividad y hacer la nómina local. Inicialmente, cada una de estas computadoras puede estar procesando trabajo en forma aislada a las demás. En algún punto, un manejador puede decidir conectar estas para extraer y correlacionar información sobre la compañía entera.

De una manera más general, la intención de una red es la de **compartir recursos**, y la meta es hacer todos los programas, datos y equipo, disponible a cualquier abonado en la red, sin desatender la localización física de los recursos y del usuario.

Un segundo objetivo es ofrecer una **alta confiabilidad** por medio de la fuente alternativa de recursos. Por ejemplo, todos los archivos son copiados en dos o tres máquinas, así si una máquina no está disponible (por alguna falla de hardware), la otra copia puede ser utilizada.

Otra meta es el **ahorro económico**. Las computadoras pequeñas tienen mucha mejor relación precio/rendimiento que una grande. Las grandes computadoras tienen aproximadamente un factor de diez veces más rapidez que las máquinas con un simple microprocesador, sin embargo su costo es cientos de veces más. Este desbalance ha

causado que muchos diseñadores de sistemas construyan sistemas formados por computadoras personales, una por usuario, con almacenamiento de datos en uno o más servidores de archivo. Tal red es llamada una **LAN** (Local Area Network).

Pero, tal vez el objetivo más importante de una red de computadoras es que ésta pueda ser usada como un poderoso **medio de comunicación** entre personas ampliamente separados.

1.1. Aplicaciones de las redes.

Para dar una idea sobre algunos importantes usos de las redes de computadoras, veremos brevemente tres ejemplos:

a) **Acceso a programas remotos.**

b) **Acceso a bases de datos remotas.**

c) **Facilidades de comunicación.**

1.1.1. Acceso a programas remotos.

A través de una red de computadoras podemos tener acceso a programas residentes en máquinas remotas. Esto es de enorme utilidad, ya que podemos accesar programas que dependan de un sofisticado hardware o de un exótico sistema operativo, por ejemplo, una compañía que ha producido un simulador de la economía mundial, permite a sus clientes correr el programa para ver como son proyectadas las relaciones de inflación, relaciones de intereses y fluctuaciones de la moneda que pueden afectar a sus negocios.

1.1.2. Acceso a bases de datos remotas.

Una mayor área del uso de las redes de computadoras es el acceso a bases de datos remotas. Por ejemplo, una persona desde su casa puede hacer reservaciones para boletos de avión, tren, autobús, barco, hoteles, restaurants, teatros, etc. con confirmación instantánea. El banco en su casa y el periódico también caen dentro de ésta categoría.

1.1.3. Facilidades de comunicación.

Con los avances tenológicos alcanzados, los costos se han reducido enormemente y la facilidad de adquirir una computadora

está al alcance de cualquier empresa de medianos recursos y de algunos particulares. Esto abre un enorme campo dentro de las posibilidades de las comunicaciones. Ejemplos de estos son, la utilización del correo electrónico, que en el futuro tendrá la capacidad de contener voz digitalizada, imágenes fijas y posiblemente imágenes en movimiento. La oficina y la escuela como la conocemos ahora pueden desaparecer. Las tiendas también pueden desaparecer por el uso de compras por catálogo vía el correo electrónico. Estas visiones futuristas sólo son unas de las posibilidades que existen en el campo de las comunicaciones, gracias a las redes de computadoras.

1.2. Estructura de la red.

Una red de computadoras es presentada en la figura 1.1. Los cuadros son las computadoras.

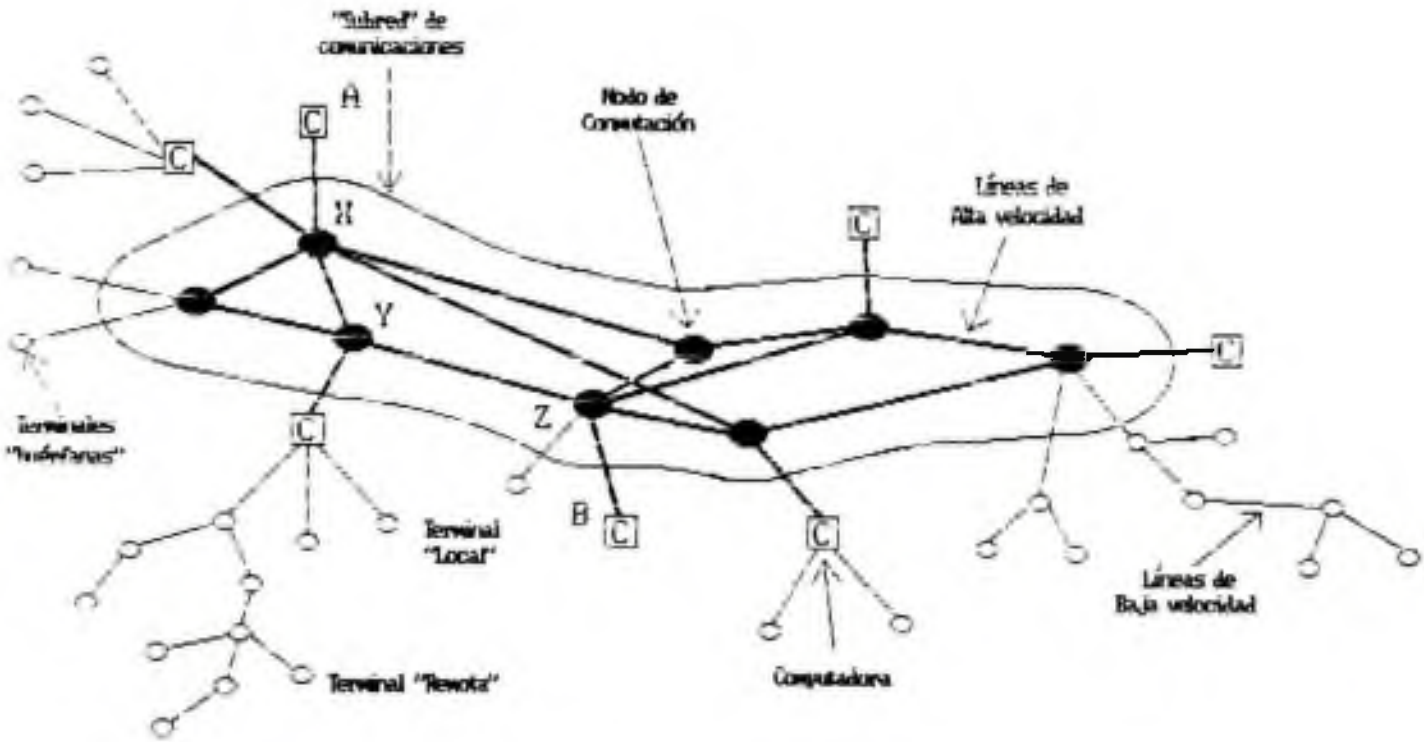


figura 1.1. Estructura de una red de computadoras.

Los circulos pequeños son las terminales, mientras que los círculos sombreados son los nodos de conmutación, los cuales

proveen las facilidades de conmutación necesarias para hacer las conexiones requeridas. Estos nodos de conmutación son sofisticadas computadoras especializadas. Algunas computadoras sirven un número de terminales y algunas de las líneas conectan terminales a sus computadoras locales. Generalmente las líneas que conectan terminales a una computadora local no necesitan transmitir datos a una alta velocidad y estas líneas son caracterizadas como líneas de baja velocidad. Las líneas de alta velocidad son necesarias para conectar los nodos de conmutación. Algunas terminales no tienen asociadas ninguna computadora local y son en cambio conectadas directamente a un nodo de conmutación (Terminales huérfanas). Similarmente algunas computadoras no tienen terminales directamente conectadas y son asociadas con la red sólo a través de su conexión con el nodo de conmutación. Finalmente, a la parte del sistema de comunicación que está compuesta sólo por nodos de conmutación es conocida como la "subred".

1.3. Tipos de redes.

Hay tres tipos básicos de redes de comunicación en servicio. Estas son:

i) Conmutación de circuitos.
ii) Conmutación de mensajes.
iii) Conmutación de paquetes.

1.3.1. Conmutación de circuitos.

En la conmutación de circuitos, los equipos terminales de datos que se están comunicando, están conectados por una ruta alambrada, dedicada a comunicar sin interrupción al par, durante toda la transmisión. Este tipo de sistema es enteramente análogo a un sistema telefónico.

1.3.2. Conmutación de mensajes.

En una comunicación por computadora, la transmisión ocurre en un relativo corto tiempo, separada por un largo intervalo de no transmisión. Típicamente una línea que conecta dos computadoras puede estar en uso un uno por ciento del tiempo disponible.

Usando conmutación de mensajes se ofrece la posibilidad de

mejorar grandemente la economía. Para ver como trabaja la conmutación de paquetes, consideremos la figura 1.1, en donde la computadora A necesita transmitir datos a la computadora B. Suponga que la trayectoria de transmisión seleccionada es **A** → **X** → **Y** → **Z** → B. Note que las líneas **X** a **Y** y **Y** a **Z** pueden ser usadas como parte de una conexión de muchas computadoras a muchas otras computadoras.

En conmutación de circuitos, una transmisión no se inicia hasta que una conexión sea establecida por completo entre los usuarios. En conmutación de mensajes no es necesario el establecimiento completo de la transmisión. En su lugar, la computadora **A** transmitirá un mensaje al nodo de conmutación **X** en donde éste será almacenado en un buffer. Este mensaje permanecerá guardado en el buffer de **X** hasta que la línea de **X** a **Y** no sea requerida por alguna otra transmisión prioritaria. Cuando la línea esté disponible, el mensaje será transmitido de **X** a **Y**. Otra vez, después de almacenado y de un posible retraso, el mensaje irá de **Y** a **Z** y después a la computadora B.

Una desventaja de éste sistema es que existe un retardo entre el tiempo de transmisión y el tiempo de recepción. En cambio, se logra que la línea esté un menor tiempo ociosa. Por ejemplo, cuando la línea de **X** a **Y** no está transmitiendo un mensaje de **A** a **B** ésta puede ser utilizada para transmitir un mensaje entre otro par de computadoras. Una red de conmutación de mensajes es referida como una red de **almacena y transmite**.

1.3.3. Red de conmutación de paquetes.

En una red de conmutación de mensajes, existe un número de características poco deseables.

i) Un mensaje puede ser muy largo o muy corto. Para acomodar los mensajes largos, grandes buffers tienen que estar disponibles. El hardware de estos buffers es desperdiciado cuando un mensaje corto es recibido.

ii) Si una línea está disponible, un nodo de conmutación no comenzará a transmitir un mensaje hasta que el mensaje entero sea recibido. Para mensajes largos y muchas retransmisiones, la longitud de éste tiempo de retardo

puede llegar a ser inconveniente.

iii) Se puede presentar el caso en el que un mensaje corto sea transmitido, pero éste no puede continuar su trayecto debido a que la línea de transmisión está ocupada por un mensaje muy largo, esto es muy inconveniente, e implantar un equilibrio entre estos dos tiempos de transmisión no es tarea fácil.

Una red de conmutación de paquetes intenta eliminar las características indeseables de la conmutación de mensajes, subdividiendo estos mensajes en paquetes. Típicamente un paquete puede tener 1024 bits de longitud. Al igual que los mensaje en conmutación de mensajes, cada paquete debe ser almacenado en buffers, en los nodos del sistema, y después transmitidos. La conmutación de paquetes es por lo tanto, un sistema de **almacena y transmite**.

En algunos sistemas de conmutación de paquetes, los diferentes paquetes de un mensaje pueden llegar al destino por diferentes rutas con diferentes retardos.

Cada unidad de transmisión, ya sea un mensaje en conmutación de mensajes o un paquete en conmutación de paquetes, debe no sólo incluir los bits de información, si no también bits adicionales referidos como encabezado de información. Este encabezado de bits debe de identificar al destino de una unidad (mensaje o paquete), así que cada nodo de conmutación conocerá como enrutar la unidad. Los bits de sincronización son también parte del encabezado y deben ser incluidos para identificar el comienzo y fin de la unidad. En conmutación de paquetes, una unidad puede viajar por diferentes rutas y por lo tanto arribar fuera de orden. Por lo tanto, estos paquetes deben ser numerados para que puedan ser ensamblados en una apropiada secuencia.

Con respecto a la conmutación de mensajes, la conmutación de paquetes tiene dos desventajas:

i) Para transmitir una cantidad de información dada por unidad de tiempo, la conmutación de paquetes requiere que los bits sean transmitidos a mayor velocidad que la requerida para la conmutación de mensajes.

ii) El hardware de conmutación necesita empaquetar,

adicionar el encabezado, desempaquetar y reensamblar en un menor tiempo que el hardware para conmutación de mensajes.

En cambio, a pesar del hecho de que la conmutación de paquetes requiere más bits de encabezado, es enteramente posible que la conmutación de paquetes transmita una cantidad de información en un menor tiempo que en conmutación de mensajes. Esta característica resulta del hecho de que un nodo de conmutación no puede comenzar a retransmitir un mensaje (relativamente largo) o un paquete (relativamente corto) hasta que el mensaje entero o paquete ha sido recibido.

1.4. Ejemplos de redes de computadoras.

En ésta sección veremos varias características de algunas de las más populares redes de computadoras.

1.4.1. TYMNET.

Tymnet es una red de computadoras de conmutación de paquetes, la cual permite a sus usuarios conectarse a las computadoras Tymnet para obtener un procesamiento interactivo y también provee facilidades de conexión para los usuarios que quieran comunicarse con otros usuarios.

Tymnet emplea una topología en malla y opera a través de los Estados Unidos y Europa usando enlace vía cable y respaldo vía satélite, lo cual asegura una operación robusta en caso de que un nodo o enlace falle. El tiempo aproximado para que un paquete transmitido de 1024 bits alcance a su destinatario es de 300 ms.

Un usuario entrando a Tymnet debe primero identificarse e informar cual es la razón por la cual usa la red, por ejemplo, para comunicarse con un destinatario fuera de la red o para el uso de las facilidades de la computadora Tymnet.

La ruta es determinada por una computadora central supervisoria. Varios "supervisores" están siempre listos en caso de falla. Cuando el usuario determina una trayectoria, ésta es establecida y el usuario siente que tiene un circuito para el solo pero en realidad, muchos usuarios pueden compartir un enlace entre dos nodos. A lo largo de la ruta determinada, un paquete es

transmitido de un nodo a otro. En cada nodo el paquete es almacenado. El nodo transmite el paquete usando el principio de **first-come-first-served** (el primero que llega es el primero que es atendido). El programa supervisorio selecciona cada ruta de transmisión para proveer el menor costo y el menor tiempo de respuesta. Como resultado del procedimiento de enrutado centralizado, los paquetes siempre arrivan en orden.

1.4.2. ARPANET.

La ARPANET es tal vez la más conocida de las redes experimentales de conmutación de paquetes en el mundo. Esta se inicio en 1969 con cuatro nodos y en la actualidad tiene más de 100. Su nombre proviene del hecho de que ésta fue desarrollada y de que es operada por la Agencia de Proyectos de Investigación Avanzada del Departamento de Defensa de los Estado Unidos(DARPA). La ARPANET es usada por investigadores para el estudio de las características de la conmutación de paquetes. Esta es usada para conectar dos o más usuarios para compartir las capacidades de la computadora central, y es usada también como medio de envío de mensajes, reportes, etc, entre los usuarios. Esta red se extiende desde Hawaii hasta Europa.

La ARPANET fue diseñada para transmitir voz y datos. Los paquetes de datos son de aproximadamente 1000 bits de longitud y los paquetes de voz son de longitud variable pero menores a 2000 bits. La ARPANET fue diseñada para proveer un costo mínimo y no más de 200 ms de tiempo de retardo independiente del tamaño del paquete. Cada paquete puede ser transmitido de fuente a destino usando diferentes nodos y por lo tanto diferentes rutas, debido a esto los paquetes pueden llegar a su destino fuera de secuencia, deben de ser almacenados y reordenados antes de ser procesados. Un mensaje es transmitido como un datagrama. Como consecuencia del almacenamiento necesario en el nodo destino antes de ser procesado, el nodo destino debe decir al fuente cuando éste puede aceptar al siguiente mensaje. Para hacer esto, el nodo destino envía al fuente, después de recibir un mensaje completo, un paquete pequeño en el cual dice que el nodo destino está listo para recibir el siguiente mensaje (**Ready-For-The-Next-Message**).

Esto es llamado el **RFNM**. Tal control de mensajes es llamado un protocolo extremo a extremo. El protocolo extremo a extremo es un algoritmo de control de flujo.

1.4.3. Red pública de datos Telepac.

A partir de la década de los setentas, la Secretaría de Comunicaciones y Transportes, a través de la Dirección General de Telecomunicaciones, comenzó a desarrollar la red pública de datos TELEPAC, con el propósito de resolver el creciente problema del procesamiento de información a grandes distancias, con las características de que fuera un servicio seguro, confiable y de alta disponibilidad.

1.4.3.1. Características.

La red Telepac utiliza la técnica de conmutación de paquetes, de acuerdo con las normas del CCITT. Esta red está formada por "nodos", los cuales se conectan por circuitos dedicados punto a punto. Los nodos son el equipo de comunicación de datos (DCE) y son los encargados de realizar el enrutamiento desde el equipo fuente al nodo destino. Los nodos utilizados por Telepac son de la serie TP4000, y también de la serie TP3000, que se emplean como convertidores de protocolos a nivel de enlace, entre los protocolos BSC y HDLC.

1.4.3.2. Topología.

La estructura de la red tiene una topología de malla, la cual se divide en dos subredes.

La red de transporte. Está compuesta por nodos conmutadores de paquetes localizados en el Distrito Federal, Monterrey, Guadalajara, Hermosillo y Villahermosa.

La red de acceso. Está constituida por equipos conectados en estrella a cada nodo de la red de transporte. Su función principal es la de portar el tráfico de datos entre el usuario y los nodos de conmutación.

1.4.3.3. Servicios.

Entre los servicios que provee la red podemos enumerar:

a) Circuitos virtuales conmutados.
b) Circuitos virtuales permanentes.
c) Grupo cerrado de abonados.
d) Comunicación por cobrar.
e) Conversión de protocolos a X.25.
f) Conexión de usuarios asíncronos (X.3, X.28 y X.29).
g) Conexión de usuarios síncronos (X.25 y otros).
h) Acceso a través de la red telefónica conmutada.
i) Acceso a través de la red Télex.

1.4.4. Red Digital de Servicios Integrados (RDSI).

La RDSI es básicamente un rediseño del sistema telefónico. La coordinación internacional para éste nuevo sistema fue tomada por el CCITT, el cual trabaja por períodos de cuatro años, con grupos de estudio, preparando las recomendaciones para las sesiones plenarias que se realizan cada cuatro años. Las recomendaciones para RDSI fueron aprobadas en 1984, con refinamientos en 1988. En la figura 1.2 se enlistan algunas de las recomendaciones.

Número	Titulo
I.120	Redes Digitales de Servicios Integrados.
I.210	Principios de servicios de telecomunicación soportados por una RDSI.
I.211	Portador de servicios soportados por una RDSI.
I.310	Principios funcionales soportados por una RDSI.
I.320	Manual de referencia del protocolo RDSI.
I.411	Configuración de las interfaces usuario/red para RDSI.
I.412	Estructura y acceso de las interfaces usuario/red para RDSI.
I.420	Interface básica usuario/red.
I.421	Relación primaria de acceso usuario/red.
I.430	Especificaciones básicas de la interface de capa 1 usuario/red.
I.431	Especificación de relación primaria de la interface de capa 1 usuario/red.
I.440	Aspectos generales del protocolo para la capa de enlace de datos usuario/red para RDSI.
I.441	Especificaciones de la capa de enlace de datos usuario/red para RDSI.
I.450	Aspectos generales de la capa 3 usuario/red para RDSI.
I.451	Especificaciones de la capa 3 usuario/red para RDSI.

fig 1.2 Algunas de las principales recomendaciones del CCITT para RDSI.

La Red Digital de Servicios Integrados se define como una red

de propósito general con conectividad total. Que puede soportar una amplia variedad de servicios (teléfono, videotex, télex, faxímil, telemetría, servicios de alarma, etc.), con un número limitado de interfaces. La evolución hacia la RDSI debe ser gradual, a partir de la actual red telefónica y debe ser compatible en todas sus etapas con los equipos y redes que estén operando, además de que deberá coexistir por muchos años con la red telefónica analógica.

Los esquemas de enlace utilizados deberán ser compatibles preferentemente con los sistemas de conmutación de circuitos a 64 Kb/s, aún cuando existen propuestas de conmutación que permitan velocidades diferentes. Deberá soportar además conmutación de paquetes, conexiones permanentes y semipermanentes.

Un elemento clave en la estabilidad de las recomendaciones de las interfaces de usuario es el análisis, de modo que se independice la evolución de los equipos de usuario de la arquitectura y tecnología de la red. Una adecuada elección de las interfaces de usuario permitirá a la RDSI adaptarse a las futuras necesidades y atraerá a los usuarios y a los fabricantes de equipo. En la figura 1.3 se plantea un esquema de la RDSI como puede ser vista por los usuarios y prestadores de servicios.

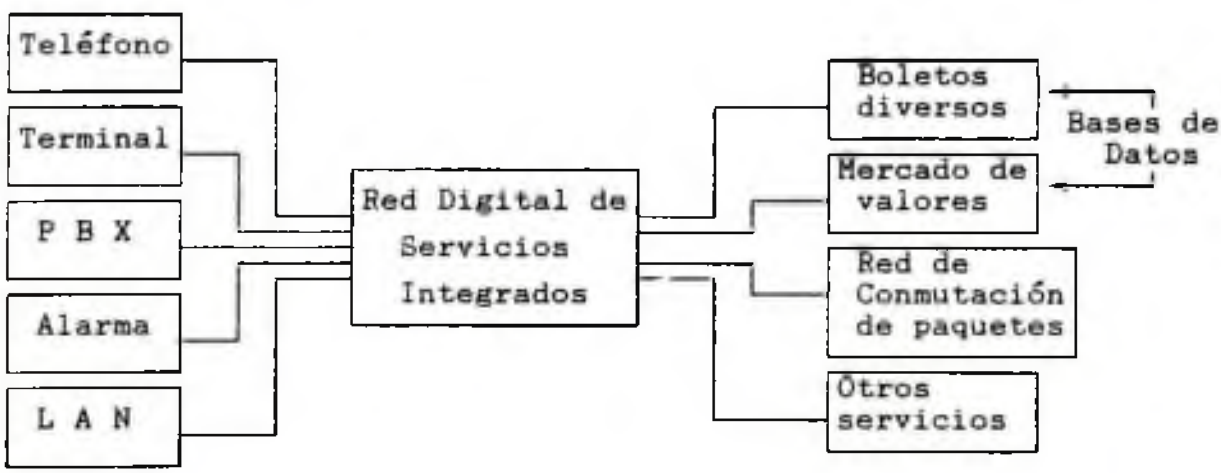
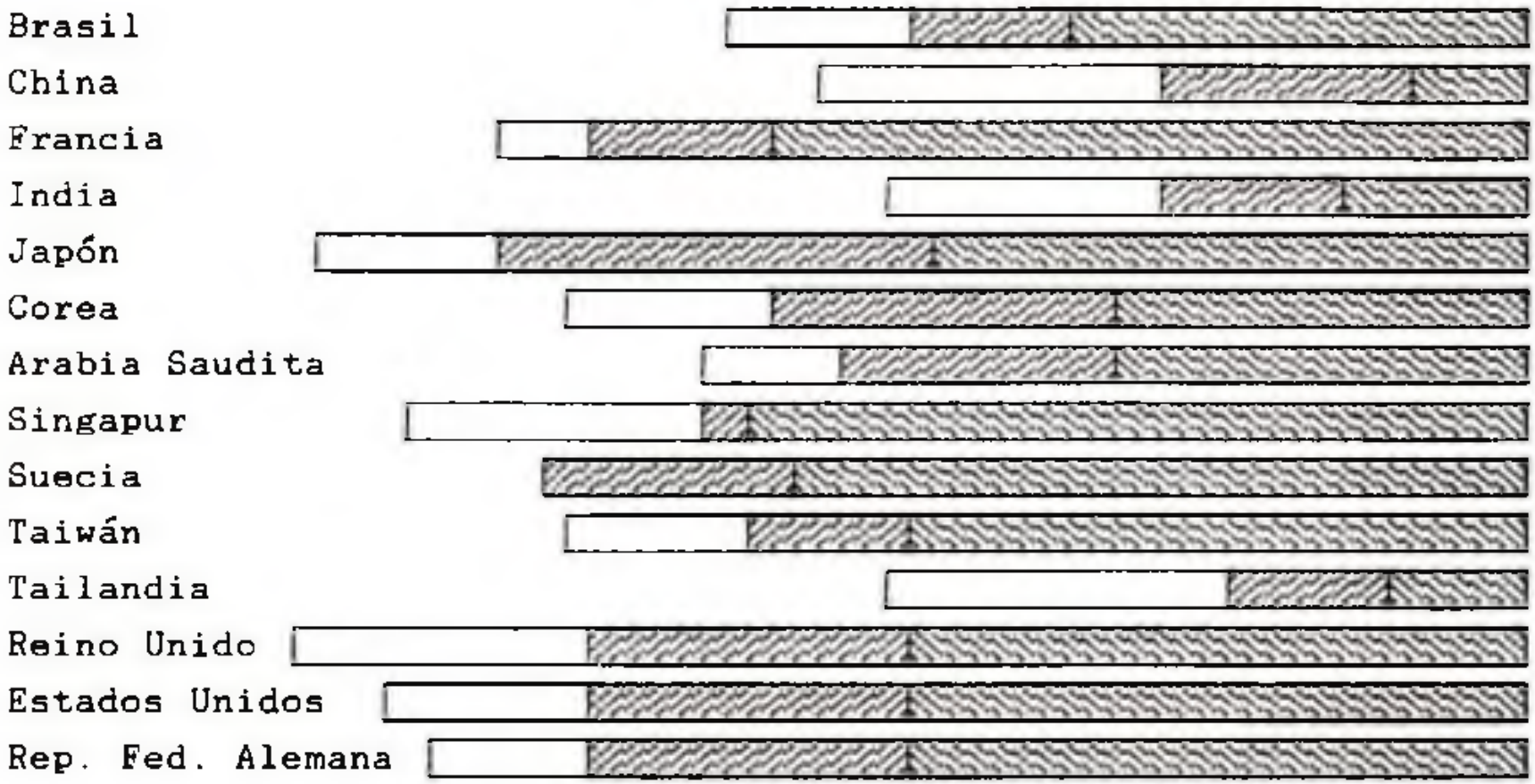


figura 1.3. Red Digital de Servicios Integrados (RDSI).

1.4.4.1. Desarrollo tecnológico de la RDSI en TELMEX.

Con el objetivo de introducir la RDSI, varios países están llevando a cabo pruebas piloto, para su futura aplicación comercial, como se muestra en la figura 1.4.



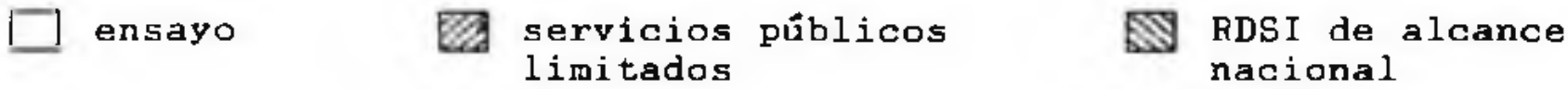


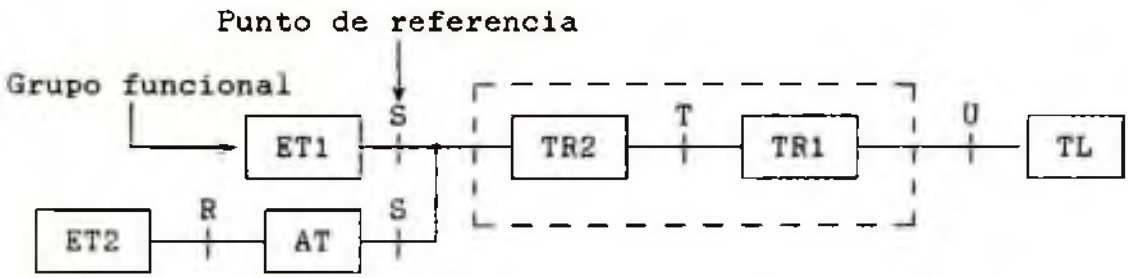
figura 1.4. Introduccíon de la RDSI en varios páises.

Para Teléfonos de México la RDSI-E (de banda estrecha con canales de hasta 64 Kb/s) reviste gran interés, por el potencial que ofrece el poder satisfacer los objetivos de diversificación, crecimiento y aumento en la calidad de los servicios, modernizando la red pública conmutada. Es por ello, que Teléfonos de México propone una prueba piloto, que permita desarrollar la infraestructura básica.

1.4.4.2. Desarrollo tecnológico.

Los estudios y desarrollos propuestos por el CID-TELMEX, están basados en las recomendaciones de la serie I, emitidas por el CCITT en el denominado Libro Rojo, y están orientados a operar

con la prueba piloto RDSI-TELMEX, en el acceso básico. Además están basados en una concepción evolutiva y formal de tal manera que permita, en base a un diseño modular y flexible, realizar los cambios que surjan en la evolución de funciones de los diferentes órganos que constituyen el diseño. El modelo de referencia que sigue TELMEX es el propuesto por el CCITT y se muestra en la figura 1.5.



TL = Terminador de Línea. Ubicado en la central, realiza funciones de nivel 1.

TR1 = Terminador de Red 1. Ubicado en el domicilio del abonado, realiza funciones de nivel 1.

TR2 = Terminador de Red 2. Realiza funciones de nivel 2 y 3. En caso de no existir, los punto S y T se juntan.

ET1 = Equipo terminal compatible con las recomendaciones de RDSI.

ET2 = Equipo terminal no RDSI. Requiere un adaptador de terminal AT.

AT = Adaptador de terminal. Hace la conversión de por ejemplo, un teléfono analógico o una terminal con interfaz RS232 a RDSI.

Figura 1.5. Grupos funcionales y puntos de referencia.

1.5. Ejemplos de algunos productos para acceso a las redes.

En ésta sección mencionaremos algunos de los productos comerciales de diferentes empresas que podemos encontrar en el mercado. Los protocolos que manejan estos productos son el X.25 o el RDSI. Es importante observar el contexto de las comunicaciones digitales, ya que esto nos puede dar un marco de referencia en el diseño y construcción de un sistema similar.

1.5.1. Tarjeta coprocesadora Am79B320ITCB.

La tarjeta coprocesadora Am79B320 de la empresa Advanced Micro Devices Inc., es una tarjeta para comunicaciones de alto rendimiento, para trabajar en el medio ambiente de una IBM PC/XT/AT. El ITCB, junto con el software AmLink3, sirve como un vehículo de desarrollo. El ITCB cumple con la recomendación I.430 del CCITT y soporta configuraciones punto a punto y punto a multipunto.

El diagrama a bloques de la tarjeta es mostrado en la figura 1.6

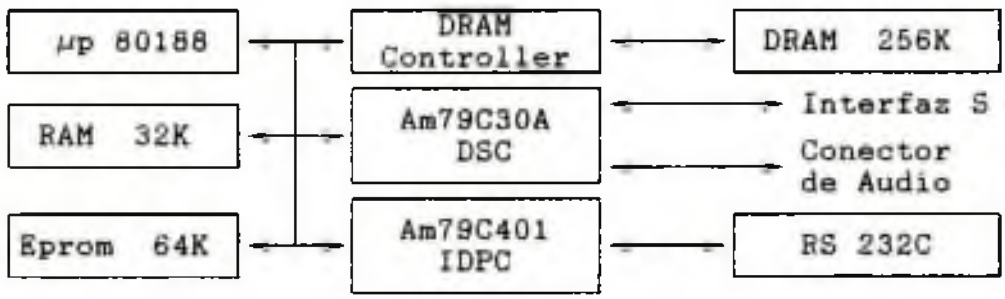


figura 1.6. Diagrama a bloques de la tarjeta coprocesadora para ISDn Am80B320ITCB.

Enlistando sus características tenemos:

-Compatible con IBM PC/XT/AT.
-Soporta la interface de acceso básico para ISDN, 2B + D.
-Puede ser complementada con el paquete de software AmLink3.
-En la tarjeta, el i80188 procesa el protocolo.
-El Am79C30A y Am79C401 provee el nivel físico para RDSI.

Paquete de software AmLink3. Este paquete provee el software necesario para el nivel de red (capa3) de el modelo OSI. AmLink3 cumple con las recomendaciones del CCITT en el protocolo X.25 para la capa 3 y el protocolo LAPB/LAPD para la capa 2. El lenguaje usado para el software de comunicaciones es el lenguaje "C". Con éste paquete, el coprocesador es compatilble con los conmutadores #5ESS de AT&T, el Northern Telecom DMS-100 o el NTT D60/70.

1.5.2. **Tarjeta para ISDN PC53.**

La PC53 ISDN es una tarjeta de comunicaciones inteligente, para enlazarse por cuatro alambres a una red RDSI conforme a la serie I de recomendaciones del CCITT. La tarjeta es de 334 mm x 114 mm y puede ser usada en una IBM PC/XT/AT o compatible. La PC53 utiliza tecnología VLSI de Intel. La tarjeta incluye un CPU i80188, 512 Kbytes de RAM Dinámica(expandible a 1MB), un controlador de protocolo para el canal B, interface "S" para RDSI, interfase telefónica y puede ser usada como tarjeta adaptadora para comunicaciones o para desarrollo de software de RDSI. El procesador i80188, con un software apropiado, puede desarrollar el procedimiento de llamada y señalización requerido para una red RDSI. Cuando es instalado con el paquete de software ISP188, el sistema es compatible con el conmutador #5ESS de AT&T. Enlistando sus características tenemos:

-Compatible con IBM PC/XT/AT.
-Soporta la interface de acceso básico para ISDN, 2B + D.
-Puede ser complementada con el paquete de software ISP188.
-En la tarjeta, el i80188 procesa el protocolo.
-El 29C53 provee el nivel físico para RDSI.
-El 82530 provee el protocolo para el canal B.
-512 Kbytes de RAM expandible a 1M byte.

Paquete de software ISP188. El paquete de software para el i80188 es específicamente diseñado para aplicaciones de RDSI utilizando dispositivos de Intel. El ISP188 soporta las recomendaciones del CCITT para la capa de enlace de datos y nivel de red (I.440, I.441 e I.450,I.451) del modelo de referecia OSI. El iATC 29C53 soporta la capa física (I.430).

2. EL COPROCO COMO UNA OPCIÓN.

En el capítulo anterior analizamos algunos tipos de redes existentes y la gama de posibilidades que ofrecen estos sistemas, en la actualidad y en el futuro cercano. Además de que mencionamos algunos de los productos que podemos encontrar en el mercado.

Por nuestra parte, la intención de esta tesis es la de entrar en este campo, desde el punto de vista de desarrollo. Para comenzar, es necesario diseñar y construir un dispositivo que sirva como base para el posterior desarrollo de capas superiores (de software) y que en conjunto nos permita accesar una red de computadoras. Bueno, este dispositivo es, por lo tanto la capa uno o nivel físico, según el modelo OSI.

Los primeros objetivos que planteamos y que no son objetivos técnicos son:

i) El dispositivo que nosotros pretendemos diseñar debe ser compatible con lo existente, además de que debe ser versátil.

ii) El factor económico es determinante, ya que muchas veces de esto depende el éxito de un producto. Por lo tanto trataremos de optimizar el diseño hasta su mínima expresión.

iii) Para que éste dispositivo pueda competir con los productos existentes en el mercado debe ser económico aunque no sea muy sofisticado.

2.1. Objetivos a seguir.

Una etapa anterior al diseño y vital es la definición de los objetivos a seguir. Para esto, nuestro sistema, al cual llamaremos **COPROCO**, tiene que considerar los aspectos mencionados anteriormente y a grandes rasgos, el coprocesador debe de:

i) Residir en forma de tarjeta dentro de la IBM PC/AT/XT[1].

ii) Ser capaz de enlazarse con el protocolo X.25.

iii) Ser capaz de enlazarse con el protocolo RDSI.

iv) Efectuar procesamiento en paralelo con la IBM PC/AT/XT.

Enseguida, se explica con más detalle cada uno de los objetivos planteados anteriormente.

i) Residencia en forma de tarjeta. Esta característica quiere decir que la circuitería a diseñar debe de montarse en una tarjeta con un conector adecuado, como para poder insertarse en una de las ranuras del bus de expansión de la IBM PC/XT/AT. Algunas de las ventajas que nos ofrece este tipo de implementación van desde el diálogo coprocesador-PC por medio de un sistema de registros direccionados como puertos o la generación de interrupciones recíprocas, hasta el manejo de memoria por medio de Accesos Directos a Memoria (DMA). Esto, regido bajo el control de un árbitro.

ii) Capacidad de enlace con el protocolo X.25. La capacidad que nosotros realizaremos en esta fase es a nivel físico y se encuentra descrita en otra norma, en la V.24. Esta trata con el tipo de codificación de las señales, con los voltajes a utilizar en estas señales. la velocidad de transmisión y otros. Para llevar a cabo esto, pensamos que el Controlador de Comunicaciones Seriales Z8530 de Zilog nos puede ayudar a realizar sólo parte de este nivel físico, ya que se necesitan de otras cosas para completarlo, como son, tipo de conector, circuitería para conversión de voltajes, etc.

iii) Capacidad de enlace con el protocolo RDSI. La capacidad que nosotros realizaremos es a nivel físico y se encuentra descrito en las recomendaciones I del CCITT. Trata con el tipo de codificación de las señales, con los voltajes a utilizar en estas señales, la velocidad de transmisión y otros. Para llevar a cabo esto, pensamos que el procesador MT8930 de MITEL nos puede ayudar en parte, ya que además de este dispositivo es necesario, tipo de conector, algún arreglo de circuitería lógica, etc.

[1] IBM PC, XT y AT son marcas registradas de International Business Machines.

iv) Capacidad para efectuar procesamiento en paralelo con la IBM PC/AT/XT. La cualidad que le da a la tarjeta la categoría de coprocesador es el procesamiento en paralelo, lo cual sólo puede ser efectuado con la utilización de un procesador extra. El procesador que utilizaremos es el 180188 de la empresa INTEL.

En la figura 2.1 mostramos la arquitectura que proponemos para el coprocesador **COPROCO**.

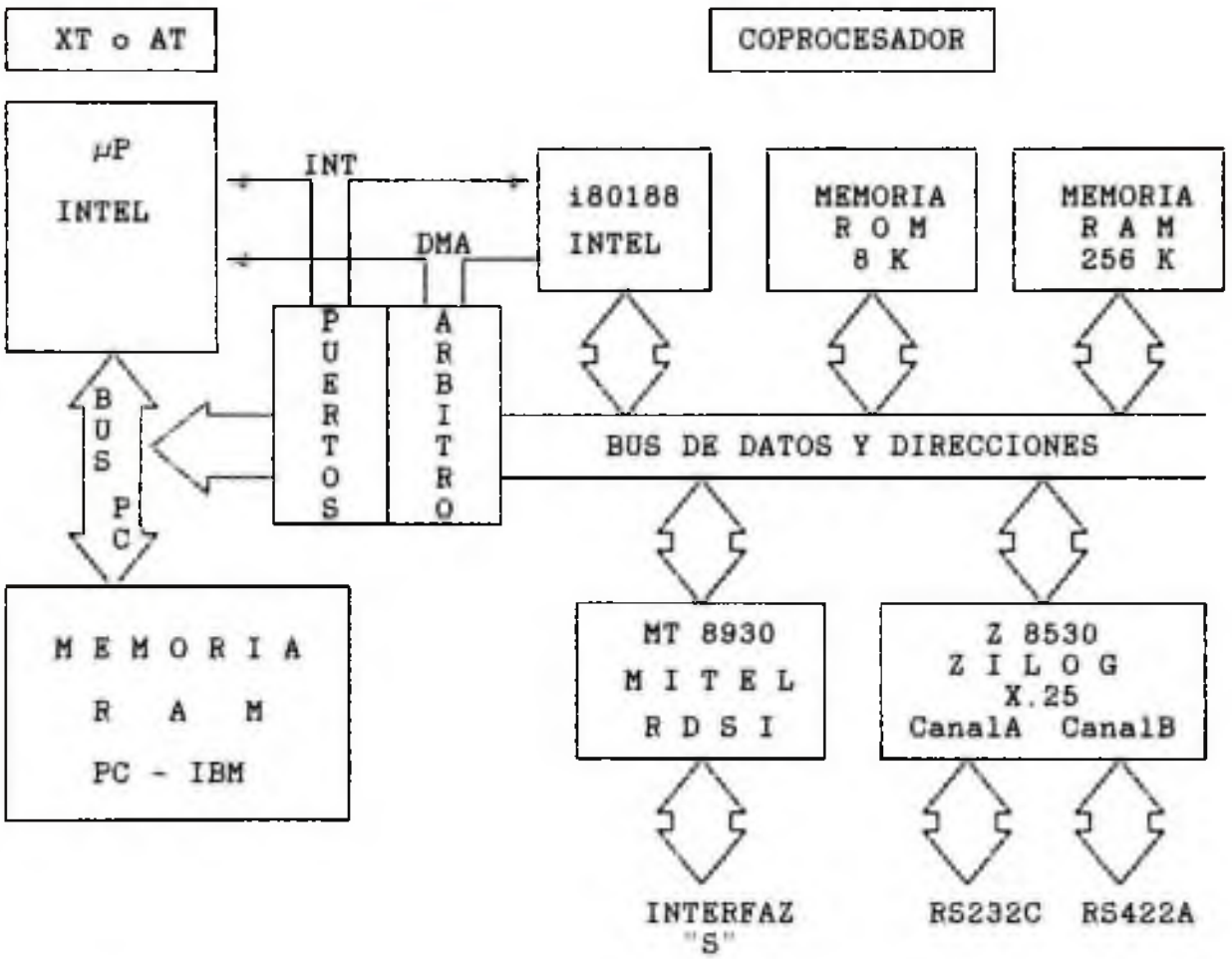


figura 2.1. Arquitectura del sistema.

3. DISEÑO DE LA CIRCUITERÍA DEL COPROCESADOR

El desarrollo de la circuitería del coprocesador está dividida en diferentes etapas de diseño, en donde cada etapa es una función específica del sistema, que incluye, el objetivo de la circuitería, el porque de la selección de los dispositivos y la explicación del desarrollo del diseño. Las etapas son numeradas en la secuencia en la cual fueron diseñadas y son las siguientes:

3.1. **Diseño del núcleo del sistema.**
3.2. **Diseño de el sistema de comunicación-interrupciones recíprocas IBM PC/XT/AT <-> Coprocesador.**
3.3. **Diseño de la circuitería de enlace con la red.**
3.4. **Diseño de la circuitería de acceso al bus de la IBM PC/XT/AT.**

A continuación se describirán detalladamente cada uno de las etapas arriba mencionadas.

3.1. Diseño del núcleo del sistema.

El núcleo del sistema es el centro del coprocesador y es la circuitería que se hará cargo del control de los procesadores de comunicaciones, así como de la comunicación con la IBM PC/XT/AT. Este está formado por los siguientes elementos:

Un microprocesador.

Este dispositivo es el elemento más importante de todo el núcleo, pues es éste el que controla a todo el sistema. Este debe de tener una capacidad de direccionamiento de 1 Mbyte de memoria.

Memoria.

El procesador contiene un sistema de 256K bytes de RAM con verificación de paridad y 8K bytes de sistema ROM.

ROM. La tarjeta tiene un elemento de memoria de 8 Kbytes con un tiempo de acceso de alrededor de 250ns. Esta memoria es adicionada para alojar rutinas de inicialización y otras rutinas primitivas.

RAM. Los dispositivos de RAM usados en el coprocesador tienen un tiempo de acceso de alrededor de 250ns. Los elementos que usa el sistema RAM son de 256 Kbit (256K x 1). El elemento usado para almacenar el bit de paridad debe de ser uno de 256 Kbits. El uso de esta memoria será utilizada como buffer de transmisión/recepción y como lugar para alojar programas.

Ciclos de refresco. Serán hechos por el controlador de DMA, mediante lecturas secuenciales.

Paridad. El bit de paridad será generado para cada byte de la RAM durante las operaciones de escritura. La señal VP será generada y destinada a una interupción no enmascarable si un error de lectura ocurre.

Mapa de memoria. El procesador que utilicemos deberá ser capaz de direccionar 1 Mbyte de memoria. Las áreas de memoria tienen que ser especificadas o mapeadas para las funciones del sistema. Estas áreas son definidas en la tabla 3.1.

Tabla 3.1. Mapa de memoria.

Dirección	Descripción	Función
00000–003FF	Vectores	Área para alojar las direcciones de las rutinas de interrupción.
00400–03FFF	Área común	Puede ser utilizada para contener programas o como buffer trasm/recp.
04000–FDFFF	No usada	
FE000–FFFFF	Inicio	Utilizada para almacenar rutinas de inicialización.

Selección de los componentes.

Una vez definidas las principales características de los componentes que formarán el núcleo, podemos pasar a la etapa de selección de los componentes.

Selección del microprocesador.

El procesador que satisface nuestros requerimientos es el microprocesador de alta integración con periféricos incluídos

i80188 de la empresa INTEL. Este procesador ofrece las siguientes ventajas:

-Periféricos incluidos, esto trae un atractivo ahorro económico.
-Ahorro en espacio, en contraste con otros, microprocesadores que requieren periféricos externos.
-No necesita de lógica de selección o en algunos casos de una muy sencilla, lo cual representa un gran ahorro desde cualquier punto de vista.

Ahora, con respecto a los demás dispositivos que un procesador necesita, como son: reloj, temporizadores/contadores, controlador de interrupciones, controlador de Accesos Directos a Memoria(DMA) y generación de estados de espera, El microprocesador i80188 de Intel no tiene la necesidad de recurrir a elementos externos de este tipo, ya que este procesador contiene:

-Un generador de reloj.
-Dos canales independientes de DMA.
-Un Controlador programable de interrupciones.
-Tres temporizadores/contadores de 16 bits.
-Lógica de selección de memoria y periféricos programable.
-Un generador programable de estados de espera.

Selección de la memoria RAM.

De los tipos de memorias existentes en el mercado, la que más nos interesó fue la DRAM MT1259-12 con tecnología MOS, esta memoria maneja 256 Kbytes por 1 bit de información con un tiempo de acceso de 120ns, suficientes para cubrir nuestras necesidades.

Selección de la Memoria ROM.

De la gran variedad de dispositivos de almacenamiento de información de este tipo, la que más se adecua a nuestras necesidades es la EPROM 2764, que nos da la opción de poderla grabar, borrarla y volverla a grabar un gran número de veces, acciones que son necesarias durante el proceso de construcción. Esta memoria tiene un tiempo de acceso de 150 ns.

Selección del sistema de verificación de memoria.

Con el fin de detectar los posibles errores de funcionamiento

en la memoria DRAM, se tiene que implementar una circuitería adicional para la verificación de ésta, utilizando el conocido método de generación del bit de paridad. El dispositivo que nos facilita el trabajo de diseñar circuitería, es el generador de paridad 74LS280 y como mencionamos anteriormente necesitamos de un elemento de almacenamiento para guardar el bit de paridad, que es provisto por una memoria MT1259-12.

Diseño.

Ya habiendo seleccionado los dispositivos necesarios que formarán el núcleo de nuestro sistema podemos comenzar a diseñar la tarjeta partiendo desde el punto en el que tenemos nuestro microprocesador al "desnudo" y al cual iremos "vistiendo" poco a poco. En primer lugar adicionaremos la ROM , en segundo lugar la RAM y por último el sistema de verificación de memoria.

3.1.1. Adición de la ROM.

Para que nuestro procesador pueda tener acceso a los datos de esta memoria, es necesario:

-Tener la dirección requerida en el bus de la ROM (A_0-A_{12}).

-Tener un bus de datos para lectura (O_0-O_7).

-Señales de control (**Chip Enable** y **Output Enable**)

Aquí notamos un pequeño problema, el procesador no puede ofrecernos direcciones y datos al mismo tiempo ya que este posee un bus multiplexado. Para vencer este problema tenemos que analizar cuales son los eventos que ocurren en el bus del procesador. En primer lugar, un cíclo de acceso de este procesador tiene una duración de cuatro pulsos de reloj. En el primer pulso de reloj T_1, los acontecimientos que suceden son:

-Aparece la señal **ALE** (Adress Latches Enable).

-Se pone la dirección requerida en el bus.

Después, hasta el tercer pulso de reloj T_3 el procesador puede leer, tal como se muestra en la figura 3.1.

Esta figura nos da una visión clara del funcionamiento del bus multiplexado del procesador i80188. Observemos como en los bits A_{19} a A_{16} y A_7 a A_0 aparece momentáneamente la dirección deseada por el procesador, por lo tanto, estas posiciones del bus

tienen que ser capturadas en algún registro, el cual tiene que ser habilitado en el momento oportuno por la señal **ALE**. El registro de retención seleccionado es el 74HC373 el cual tiene un nivel activo alto. Entonces, damos por hecho que la dirección se encuentra fija a partir del pulso T2. Se necesitaron dos latches, uno para las posiciones AD_{19}-AD_{16} y otro para las posiciones AD_7-AD_0. Después de resolver el problema del bus compartido, sólo resta conectar la salida de los registros 74HC373 al bus de direcciones de la memoria EPROM, el bus de datos de la memoria al bus de datos del procesador, la señal $\overline{RD}$ del procesador a la terminal $\overline{OE}$, esto con el fin de que sólo cuando se trate de una lectura, la salida de esta memoria se habilite.

Para seleccionar esta área de memoria no se necesita de lógica de selección ya que el procesador i80188 provee una salida programable $\overline{UCS}$ (Upper memory Chip Select) para evitarnos este trabajo. Observese el diagrama 1 del apéndice A.

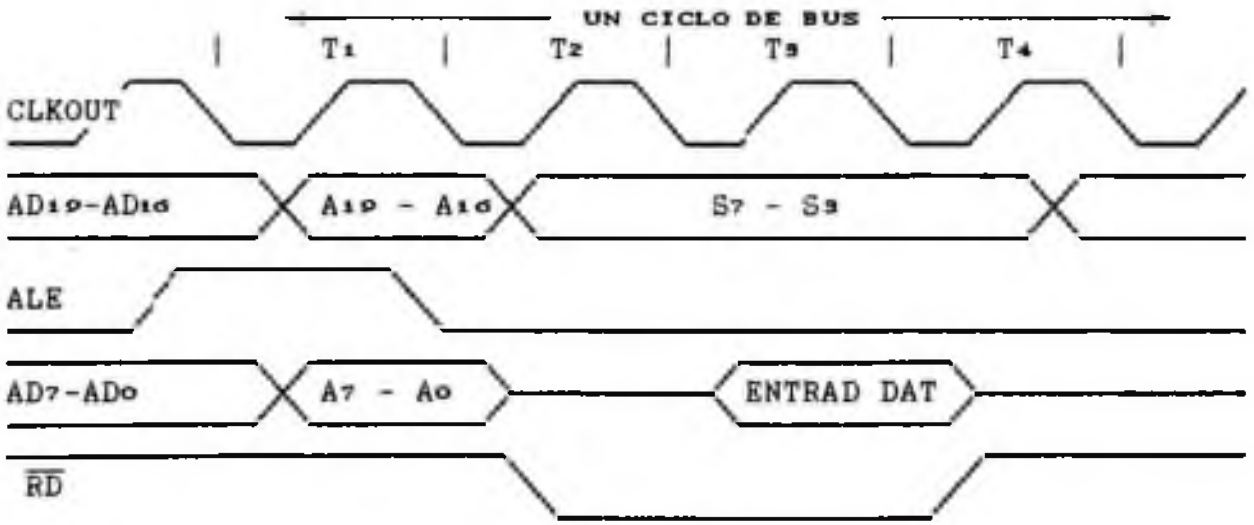


figura 3.1. Señales del bus multiplexado.

3.1.2. Adición de la RAM. Debido a la naturaleza de la memoria MT1250-12, que utilizaremos para el diseño, la adición de ésta es más complicada que la de la memoria ROM, pues hay que tomar en cuenta que:

i) Esta recibe la dirección de una celda en dos partes, primero el renglón y luego la columna. Esto se indica por medio de las señales $\overline{RAS}$ y $\overline{CAS}$.

ii) Requiere de refresco de memoria.

Esta naturaleza nos obliga a partir en dos la dirección de la celda deseada por el procesador, utilizando un multiplexor. Aquí tenemos que tomar en cuenta que el rango RAM que el procesador podrá direccionar es de 256 Kbytes (Ver tabla 1), para esto son necesarios sólo 18 de los 20 bits de direcciones (del A_0 al A_{17}). Ahora, nos conviene diseñar la circuitería de tal manera que los datos de la matriz interna de la memoria se accesen columna por renglón, esto es muy útil para realizar el refreso de memoria, ya que si el refresco de memoria consiste en la lectura de direcciones contiguas, bastará con leer un pequeño rango de memoria (512 posiciones) para que toda la memoria quede refrescada, debido a que la direccion de los renglones estará variando en cada acceso (esto se explica con más detalle en la sección 4.1.3.). Entonces la dirección del renglón estará formada por los 9 bits menos significativos (A_0 - A_8) y la dirección de la columna por los 9 bits más significativos (A_9 - A_{17}).

En resumen, en una operación de acceso a memoria, la secuencia es la siguiente:

La dirección puesta por el procesador en el bus es partida por el multiplexor, el cual mostrará a su salida la parte correspondiente al renglón(A_0 - A_8), acción controlada por la señal **MPX**. En este momento también se le indica a la memoria que capture la dirección del renglón, por medio de la señal **$\overline{RAS}$**. Una vez hecho lo anterior, la señal **MPX** tiene que cambiar de estado, indicando al multiplexor que haga visible a su salida la otra parte, la correspondiente a la columna(A_9 - A_{17}), en seguida, se le indica a la memoria por medio de **$\overline{CAS}$**, que capture la dirección de la columna. Entonces la celda direccionada está lista para ser leída o para escribir en ella, como se muestra en la figura 3.2.

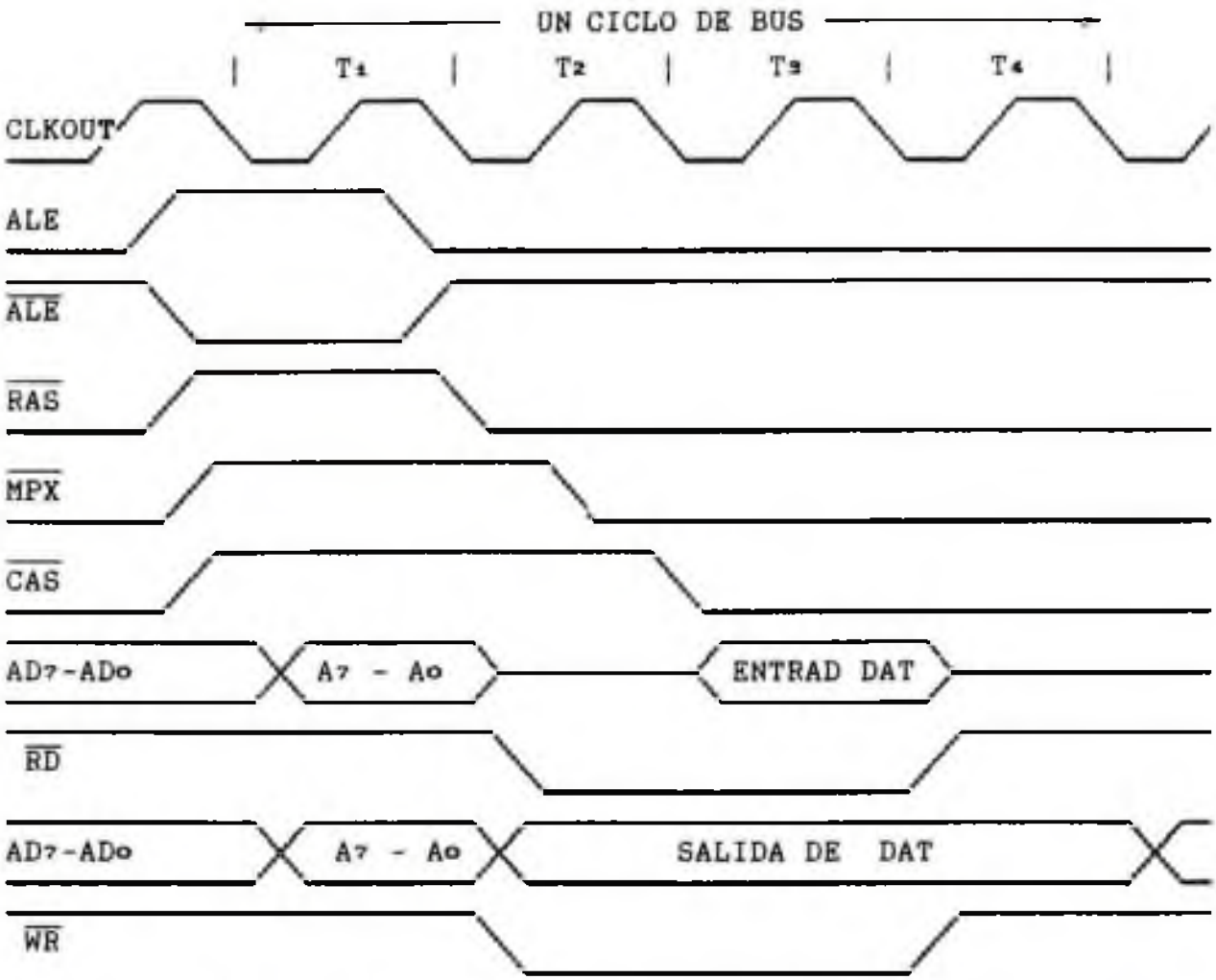


figura 3.2. Señales para el control de la RAM

Las señales **RAS** y **CAS** deben de tener un nivel activo bajo, ya que las memorias empleadas así lo requieren. La señal **MPX** también tiene un nivel activo bajo, que sirve para conmutar las salidas del multiplexor 74HC157. Se necesitaron tres multiplexores para cubrir los 18 bits de dirección.

El circuito que generará las tres señales es un simple circuito de corrimiento, que desplaza a través de sus tres flips-flops un dato constante, este dato la señal **ALE** y se encuentra colocada en la entrada D_0 del primer flip-flop.

Entonces, la secuencia de funcionamiento que queremos desarrollar tiene que iniciar con la señal **ALE** en el pulso de reloj T_1 y la secuencia debe de ser como sigue:

La señal **ALE** invertida controla la entrada PRESET de los flip-flops, que al momento de ir a su estado activo bajo, los flip-flops pone su salida a un nivel alto debido al nivel bajo registrado en su entrada PRESET, esto sucede durante el pulso T_1 del ciclo de bus. Los flip-flop se activan con un filo de subida.

La dirección se debe de encontrar en la memoria por lo menos al comienzo del pulso de reloj T_3. Para lograr esto tenemos que recurrir a un artificio, el cual consiste en poner inversores en las entradas de reloj de los flip-flops 1 y 3, esto con el fin de cambiar el estado de estos flip-flops durante los filos de bajada y no esperar hasta que haya filos de subida, de otra manera la generación de las señales sería demasiado lenta. Entonces, al inicio de un ciclo de acceso a memoria, el flip-flop$_1$ se activa con el filo de bajada del pulso T_1, cambiando éste al estado determinado por su entrada, el nivel adoptado por la salida Q_0 es por lo tanto **ALE** y ya que en ese instante está a un nivel bajo, obtenemos la señal $\overline{\textbf{RAS}}$. Durante el siguiente pulso de reloj se tienen que desarrollar las siguientes acciones: la señal **MPX** tiene que caer a nivel bajo por medio del filo de subida de T_2, esto se logra simplemente conectando el reloj CLKOUT directamente a la entrada de reloj CK_1 y conectando la entrada D_1 a la salida Q_0 que ya tiene un nivel bajo, entonces al ocurrir la transición de un "0" a un "1" en la entrada CK_1, el nivel adquirido por Q_1 es un nivel bajo, obteniendose así **MPX**. La otra acción es obtener $\overline{\textbf{CAS}}$, conectando la salida Q_1 a la entrada D_2 y esperando a que ésta se refleje en Q_2 durante el filo de bajada de T_2, de esta manera se obtiene $\overline{\textbf{CAS}}$.

Para seleccionar la memoria RAM se recurrió a las terminales de selección que provee el i80188 y a una pequeña lógica de selección. La lógica de selección solamente es la unión de dos señales, la señal $\overline{\text{MCS0}}$ y $\overline{\text{MCS1}}$ que dan como resultado la señal $\overline{\text{RAM}}$, como se puede observar en la figura 3.3. La lógica programable de selección de memoria debe de estar programada de tal manera que $\overline{\text{MCS0}}$ responda cuando se esté accesando el rango 00000H a 2FFFFH de memoria y $\overline{\text{MCS1}}$ cuando se esté accesando el rango 30000H a 4FFFFH.La señal $\overline{\text{RAM}}$ es la que habilita a la RAM dinámica para un acceso a memoria completo, ya que sólo cuando se presente esta

señal, la señal $\overline{CAS}$ se generará (última de las tres señales necesarias para accesar la RAM), acción condicionada por una compuerta "O". La utilización de esta condición trae como ventaja el hacer el ciclo de acceso a memoria completo, sólo cuando sea necesario y no en otro caso, como cuando se está refrescando la memoria. El circuito terminado es el mostrado en la figura 3.3 y en el diagrama 2 del apéndice A.

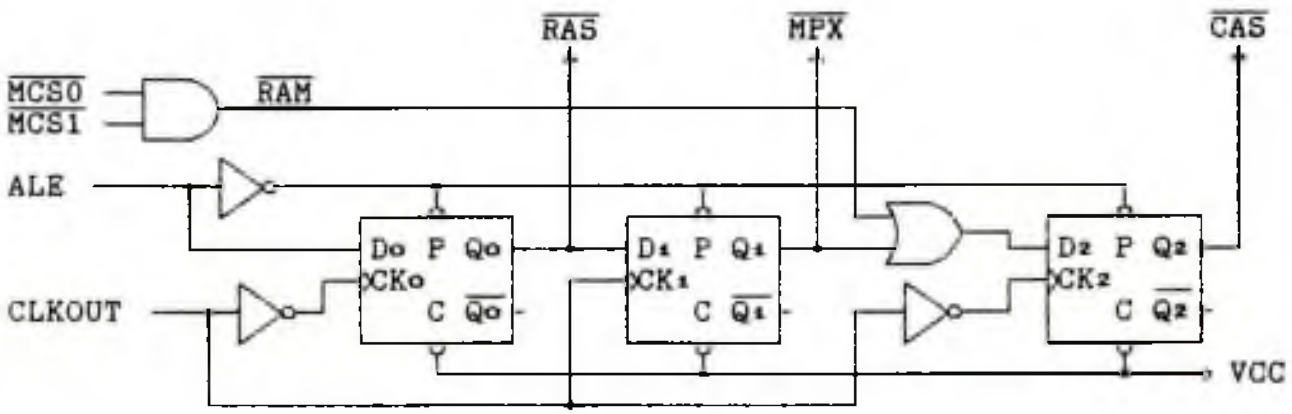


figura 3.3. Circuitería para la generación de las señales de control de la DRAM.

3.1.3. Realización del sistema de verificación de memoria.

La lectura de datos en RAM está sujeta a dos tipos de errores, errores de hardware y errores temporales. Los errores de hardware son causados por fallas permanentes en los diferentes dispositivos. Esto puede ser causado por un defecto de fabricación o por una descompostura aleatoria en el circuito integrado. Los errores temporales son errores causados por pulsos de ruido en el sistema o, en el caso de la RAM dinámica, una partícula alfa o alguna otra radiación puede causar la carga del pequeño capacitor en el cual el bit del dato es almacenado. Como nosotros adicionaremos un gran arreglo de RAM al sistema, la ocurrencia de errores por hardware o temporales se incrementa grandemente. Entonces, esto incrementa la probabilidad de que el sistema entero falle. Para prevenir o al menos reducir esta clase de fallas, adicionamos circuitería para detectar estos errores de lectura en

RAM. Hay varias formas de hacer esto.

Una opción son los códigos lineales de bloque, de los cuales el código de Hamming es uno de ellos. Este método sirve, no sólo para detectar errores, sino también para corregirlos. El problema de este código de detección de errores es que necesita una cierta cantidad de información extra para detectar/corregir los errores que se presenten en los datos. Por ejemplo, si trabajamos con datos de ocho bits, es necesario pegarle por lo menos, a estos ocho bits, una información de cuatro bits. El implementar esto en hardware traería consigo adicionar cuatro bits de información para cada palaba de 8 bits de la memoria RAM y circuitería extra de detección-corrección. Por lo tanto implementar esto en la tarjeta resulta totalmente inconveniente.

En cambio, preferimos utilizar una manera menos sofisticada de detectar errores, pero a un precio razonable.

El simple método para detección de error que utilizaremos es el del bit de paridad. Haremos esto determinando primeramente la paridad de una palabra de 8 bits cuando está siendo escrita a una localidad de memoria. Nosotros entonces generaremos un bit de paridad tal que la paridad del dato de 8 bits más el bit de paridad de este sea siempre, por ejemplo, impar. El bit de paridad generado para cada byte es escrito en una memoria separada en paralelo con el dispositivo que contiene el dato de 8 bits. Cuando el dato de 8 bits y el bit de paridad son leídos de memoria, nosotros verificamos la paridad de los 9 bits. Si la paridad de estos nueve es par, entonces algún error en el proceso de escritura/lectura ha ocurrido. Si una circuitería externa está siendo usada para generar/verificar la paridad, entonces una salida desde esta circuitería puede ser usada para decir al procesador que un error ha ocurrido, y que los datos son no válidos. El procesador puede entonces responder apropiadamente.

Una dificultad con este método de verificación de paridad es que dos errores en un byte pueden cancelarse. Un segundo problema con esta simple verificación es que esta no nos dice cual bit en la palabra está errado y pueda ser corregido.

La verificación que nosotros realizaremos la llevamos a cabo utilizando el generador de paridad 74HC280, una memoria de un bit

adicional y un flip-flop, que está destinado a retener el error detectado.

El procedimiento de verificación se desarrolla en dos partes, la primera durante el ciclo de escritura y la segunda, durante el ciclo de lectura.

Durante el ciclo de escritura se genera el bit de paridad y se almacena de la siguiente manera:

El procesador pone una dirección y un dato en el bus, el bus de datos alimenta con esta información a las ocho memorias y a las ocho de las nueve entradas del generador de paridad, el dato producido por este generador es almacenado en la novena memoria (noveno bit) y permanece ahí hasta su futura utilización, durante el ciclo de lectura. La aparición de un dato en la novena entrada está condicionada por una compuerta "Y" a aparecer sólo cuando se esté realizando una lectura.

En el ciclo de lectura se genera el bit de paridad y se compara con el almacenado anteriormente para ese mismo dato, de la siguiente manera:

El procesador pone una dirección en el bus, las memorias responden virtiendo en el bus de datos el dato solicitado por la dirección, este dato nuevamente alimenta a las ocho de las nueve entradas del generador de paridad, pero ahora la novena entrada del generador es alimentada por la novena memoria que también vierte el dato que resultó de la previa generación de la paridad para este mismo dato. Sólo durante este ciclo, este dato se hace visible a la novena entrada del generador, ya que esta se encuentra condicionada a pasar el dato sí y sólo sí la señal $\overline{WR}$ del procesador se encuentra en nivel alto.

Si existiera un error de escritura o lectura en alguna de las memorias, esto sería indicado durante el ciclo de lectura, a través de la salida EVEN = 1 del generador, pero esto es únicamente válido en el ciclo de lectura ya que el flip-flop que sujeta e indica que esto ha sucedido es activado con el filo de subida de la unión de las señales $\overline{RD}$ y $\overline{CAS}$. Esto se realiza con la ayuda de una compuerta "O". Esto se hace con el fin de que el flip-flop se active sólo cuando se trate de un acceso a RAM . La salida Q del flip-flop se conecta a la entrada de la NMI (Non

Maskable Interrupt), para que en caso de error en memoria el procesador efectúe la acción apropiada.

Por otra parte, el estado inicial de la salida Q de un flip flop es impredecible, por lo tanto para estar seguros que esta salida está a cero, es necesario de alguna manera forzarla a este estado. La solución encontrada fue la de colocar la terminal de selección **$\overline{\text{PCS0}}$** a la entrada $\overline{\text{CL}}$ del flip flop y mediante alguna instrucción residente en la ROM, accesar el área 0000H a 0080H de espacio E/S, para que este acceso se refleje en la terminal **$\overline{\text{PCS0}}$**, quedando así borrada la salida Q. Tal y como se muestra en la figura 3.4.

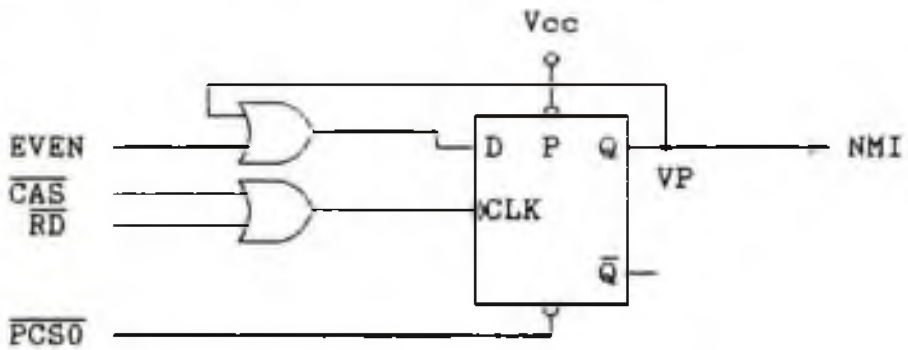


figura 3.4. Verficación de error en memoria.

El diseño final del núcleo se puede observar en el diagrama general.

3.2. Diseño del sistema de comunicación-interrupciones recíprocas IBM PC/XT/AT ↔ Coprocesador.

La principal función de este sistema es la de comunicar a la tarjeta coprocesadora con la IBM PC/XT/AT de una manera sencilla. Las funciones que debe de tener esta circuitería son:

i) Paso de datos/instrucciones con longitud de 8 bits a través de registros direccionados como puertos.
ii) Generación de interrupciones vectorizadas direccionadas como puertos.
iii) Paso de datos/instrucciones a través de una petición de DMA.

La inclusión de un sistema dedicado a estos fines hace posible la interacción entre la IBM PC/XT/AT y la tarjeta coprocesadora. Tal interacción es necesaria para las dos partes, su utilidad radica en que a través de esta circuitería, tanto la IBM PC/XT/AT como el coprocesador pueden pasarse datos, rutinas de servicio, ordenarse la ejecución de las mismas y provocarse interrupciones. En la figura 3.5 mostramos esquemáticamente las características de esta circuitería a la cual llamaremos diálogo.

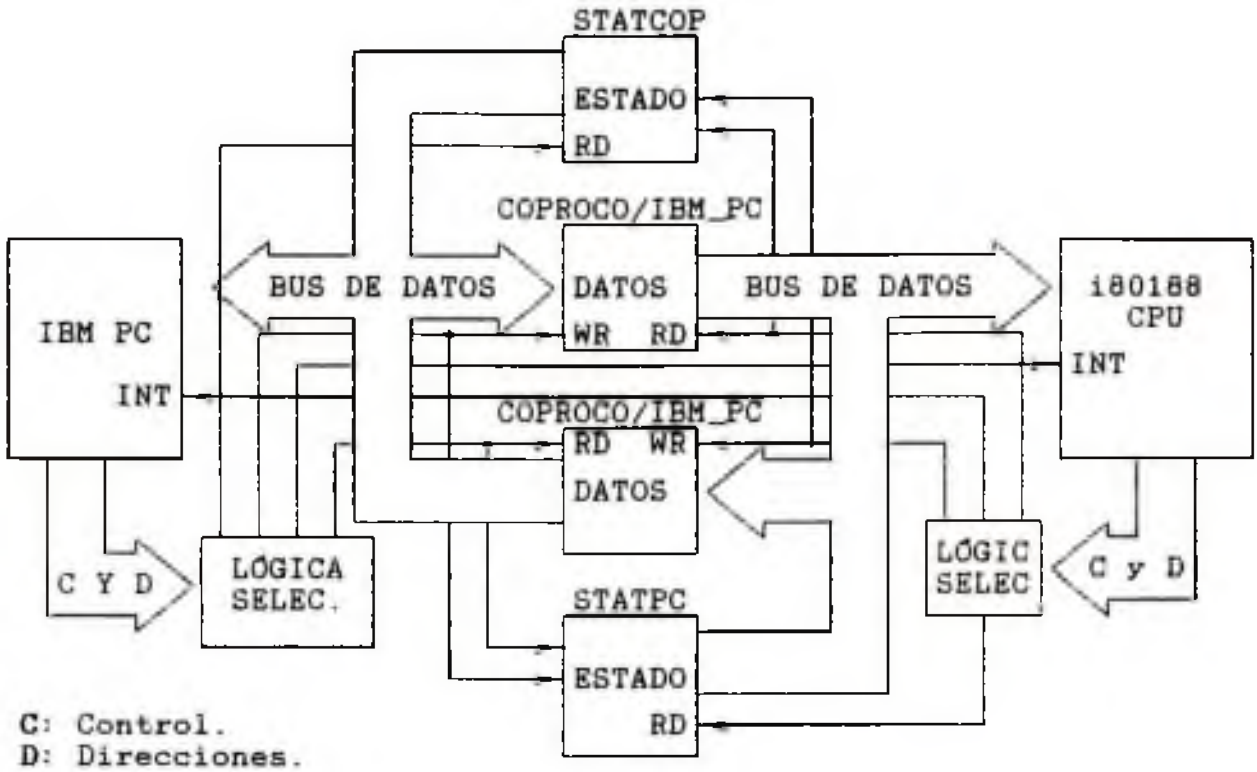


figura 3.5. Circuitería "diálogo".

El diseño de esta circuitería está dividida en dos partes, la parte que maneja la IBM PC/XT/AT y la parte que maneja el coprocesador. De estas, hay una circuitería común para las dos partes, que son los registros de datos y los registros de estado. Ya que el registro que usa la IBM PC/XT/AT para escribir, es el mismo en el que lee el i80188 y el que usa la IBM PC/XT/AT para leer, es el mismo que ocupa el i80188 para escribir. Ahora, el estado de los registros de estado depende de las acciones que ocurran en los registros de datos, por ejemplo, una escritura por parte del i80188 al registro de datos, modificará un bit del registro de estado, el cual indicará que un dato está en el registro y que la otra parte no lo ha tomado. Por el otro lado, la IBM PC/XT/AT puede detectar, como producto de esta acción, que en su registro de datos hay un dato listo para ser leído. Esto es muy importante, ya que es la única manera de saber si el dato mandado por una parte ya fue recibido por la otra, o si la otra parte nos está mandando algún dato. A grandes rasgos esta es la función de

esta circuitería, posteriormente a ésta se adicionará otra que nos permitirá hacer transferencias por DMA. En seguida se describe la secuencia de diseño para cada una de las dos partes.

3.2.1. Registro de datos direccionado por la IBM PC/XT/AT.

Por parte de la IBM PC/XT/AT se tiene que definir primeramente la dirección de espacio E/S en la cual se va a localizar esta circuitería, ya que como mencionamos al principio los registros de datos se tienen que direccionar como puertos. Recurriendo al manual de referencias técnicas de la IBM PC/XT/AT encontramos un espacio disponible en el rango de direcciones que van de la 0280h a la 02F7h. Nosotros decidimos utilizar las direcciones 0280h, 0288h, 02A0h, 02A8h, 02C0h, 02C8h, 02E0 y 02E8h para direccionar el registro de datos/instrucciones el cual llamaremos COPROCO. Sólo una dirección puede ser seleccionada por medio de un dip-switch y esto nos da la opción de conectar varias tarjetas a una misma máquina para, por ejemplo, construir un nodo de conmutación. Este registro se muestra en la figura 3.6.

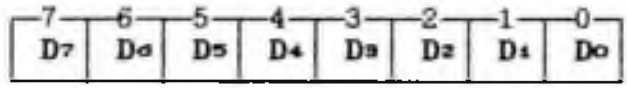


figura 3.6. Registro COPROCO.

3.2.2. Registro de estado direccionado por la IBM PC/XT/AT.

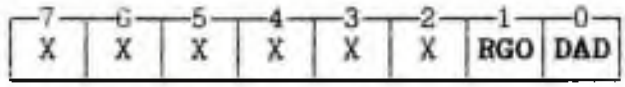
Como mencionamos anteriormente, un registro de estado es necesario, este tendrá la función de indicar el tipo de acción efectuada sobre cualquiera de los registros de datos. Las condiciones y acciones que debemos tomar en cuenta son las siguientes:

a) Cuando la IBM PC/XT/AT escriba en el registro de datos, se activará el bit 1, indicando que este registro está ocupado, a su vez, este mismo bit estará indicando al i80188 que un dato está disponible y permanecerá así hasta que el i80188 lo lea.

b) Cuando la IBM PC/XT/AT lea del registro de datos, se desactivará el bit 0, indicando que ya no hay dato

disponible. Este mismo bit indica, por su parte, a el i80188 que su registro de escritura de datos está vacío.

El funcionamiento deseado se logra implementando una sencilla circuitería con flip flops, los cuales reciben las señales **WRITE** o **READ** de la lógica de selección, variando así su estado. Este registro estará localizado en las direcciones 0281h, 0289h, 02A1h, 02A9, 02C1, 02C9, 02E1 y 02E9, será sólo lectura, lo llamaremos STATCOP. Sólo una dirección puede ser seleccionada a la vez por medio de un dip switch. Este registro se muestra en la figura 3.7.



RGO = ReGistro Ocupado.
DAD = DAto Disponible.

figura 3.7. Registro STATCOP.

3.2.3. Generación de interrupciones vectorizadas al coprocesador.

Para la generación de estas se aprovechan las direcciones 0281, 0289, 02A1, 02A9, 02C1, 02C9, 02E1 y 02E9 ya que sólo están ocupadas por un registro de sólo lectura. Entonces podemos arreglar esta circuitería de tal manera que cuando efectuemos una operación de escritura en esta dirección se provoque una interrupción en el coprocesador. El ocupar una dirección para dos funciones totalmente diferentes nos ahorra circuitería de selección. El número de interrupción debe estar alojado en el registro COPROCO con anterioridad a la generación de esta interrupción.

La circuitería necesaria para llevar a cabo esto, consiste de una lógica de selección formada por dos multiplexores 74HC138 y algunas compuertas. El registro de datos es un simple 74HC373 y el registro de estados está formado por dos flip-flops y un buffer, tal y como se muestra en el diagrama general o en el diagrama 4 del apéndice A.

3.2.4. Registro de datos direccionado por el coprocesador.

En el COPROCESADOR hay que seleccionar el rango de espacio E/S disponible, en realidad hay mucho espacio, pero sólo nos interesa el rango dentro del cual se producen las señales de selección de periféricos $\overline{PCSi}$. Nosotros tenemos disponible el rango que va de la dirección 0100h a la 017Fh, que corresponde a la terminal $\overline{PCS2}$. Nosotros decidimos utilizar la dirección 0100h para direccionar el registro de datos/instrucciones el cual llamaremos IBM_PC. Este registro se muestra en la figura 3.8.

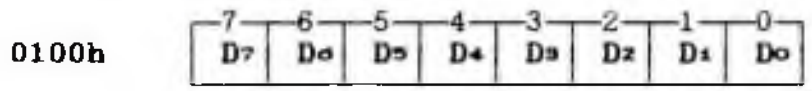


figura 3.8. Registro IBM_PC.

3.2.5. Registro de estado direccionado por el coprocesador.

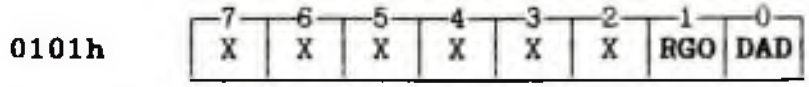
Ahora, al igual que con la IBM PC/AT/XT un registro de estado es necesario, este tendrá la función de indicar el tipo de acción efectuada sobre cualquiera de los registros de datos. Las condiciones y acciones que debemos tomar en cuenta son las siguientes:

a) Cuando el i80188 escriba en el registro de datos, se activará el bit 1, indicando que este registro está ocupado, a su vez, este mismo bit estará indicando a la IBM PC/XT/AT que un dato está disponible y permanecerá así, hasta que la IBM PC/XT/AT lo lea.

b) Cuando el i80188 lea del registro de datos, se desactivará el bit 0, indicando que ya no hay dato disponible. Este mismo bit indica, por su parte, a la IBM PC/XT/AT que su registro de escritura de datos está vacío.

El funcionamiento deseado se logra implementando una sencilla circuitería con flip flops, los cuales reciben las señales $\overline{WRITE}$ o $\overline{READ}$ de la lógica de selección, variando así su estado. El registro estará localizado en la dirección 0101h, será sólo lectura y lo llamaremos STATPC. Este registro se muestra en la

figura 3.9.



RGO = ReGistro Ocupado.
DAD = DAto Disponible.

figura 3.9. Registro STATPC.

3.2.6. Generación de interrupciones vectorizadas a la IBM PC/XT/AT.

Esto se realiza aprovechando la dirección 0101h ya que esta dirección está ocupada por un registro de sólo lectura. Entonces podemos arreglar la ciruitería de tal forma que cuando se escriba en esta dirección, se provoque una interrupción en el procesador de la tarjeta. La decisión de ocupar una sola dirección para dos funciones completamente diferentes nos ahorra circuitería de selección. El número de interrupción debe estar alojado en el registro IBM_PC con anterioridad a la generación de esta interrupción.

La circuitería necesaria para llevar a cabo esto, consiste de una lógica de selección sencilla, formada únicamente por un multiplexor 74HC138. El registro de datos es un simple 74HC373 y el registro de estados está formado por dos flip-flops y un buffer, tal y como se muestra en el diagrama general o en el diagrama 4 del apéndice A.

3.2.7. Paso de instrucciones/datos a través de peticiones de DMA.

Otra de las cualidades que debe de poseer la circuitería descrita anteriormente, es la capacidad de transferir datos de la memoria de la IBM PC/XT/AT al coprocesador, por medio de accesos directos a memoria (DMA), con las siguientes características:

a) Las transferencias deben ser realizadas por el controlador de DMA 8237A de memoria de la IBM PC/XT/AT hacia el coprocesador y viceversa. Por lo tanto este debe estar debidamente programado antes de la petición de la transferencia.

b) Las peticiones hechas a este deben ser efectuadas por el

coprocesador mediante la terminal **PCS4**. Esta terminal reflejará un 0, cuando el i80188 lea un dato en el puerto localizado en las direccines 0200H a la 027FH, esto quiere decir que, el i80188 pedirá un dato por DMA cuando accese este puerto.

c) Hay que tomar en cuenta que cuando se haga la petición de transferencia al controlador de DMA, este no podrá atendernos al instante. Por lo tanto habrá que esperar hasta que nuestra petición sea reconocida. Entonces, para lograr la temporización, hay que adicionar un árbitro.

La circuitería que hará esta función se puede implementar aprovechando los registros de datos de la circuitería "diálogo", de esta manera nos evitamos adicionar buffers de aislamiento entre los dos buses. Para realizar esto tenemos que modificar un poco la lógica de selección de las dos partes, de tal forma que:

1) El i80188 podrá seleccionar la circuitería "diálogo" en dos situaciones diferentes, cuando lea/escriba en el puerto con direcciones 0100H a 017FH (señal **PCS2**) o cuando lea/escriba en el puerto con direcciones 0200H a 027FH (señal **PCS4**). Esto se logra con una simple compuerta "Y" como se muestra en la figura 3.10.

figura 3.10.

2) La IBM PC/XT/AT podrá accesar estos registros en dos situaciones diferentes, en un transferencia normal y en una trasferencia por DMA. Esto se podrá realizar tomando en cuenta que la IBM PC/XT/AT podrá realizar lecturas habilitando la entrada **READ** y escrituras habilitado la entrada **WRITE** de cada uno de los registros correspondientes a estas funciones, por medio de la lógica de selección de la circuitería "diálogo" (**Y1** para **READ** y **Y2** para **WRITE** donde **Y12** es el producto de la lógica de selección ver figura 3.11 o diagrama general). La modificación consiste en la adición de una compuerta "Y" con la lógica de selección para

acceso normal en una entrada y la lógica de selección de DMA en la otra, como se muestra en el diagrama general o en el diagrama 4 del apéndice A. La lógica de selección de DMA se activa siempre y cuando se esté solicitando una transferencia de DMA por medio de **DRQ1**, se haya reconocido esta petición (señal **$\overline{\text{DACK1}}$**) y se genere la señal **$\overline{\text{IOW}}$/$\overline{\text{IOR}}$**, tal y como se muestra en la figura 3.11.

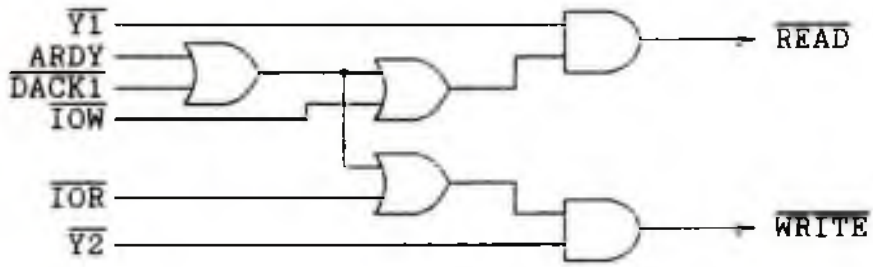
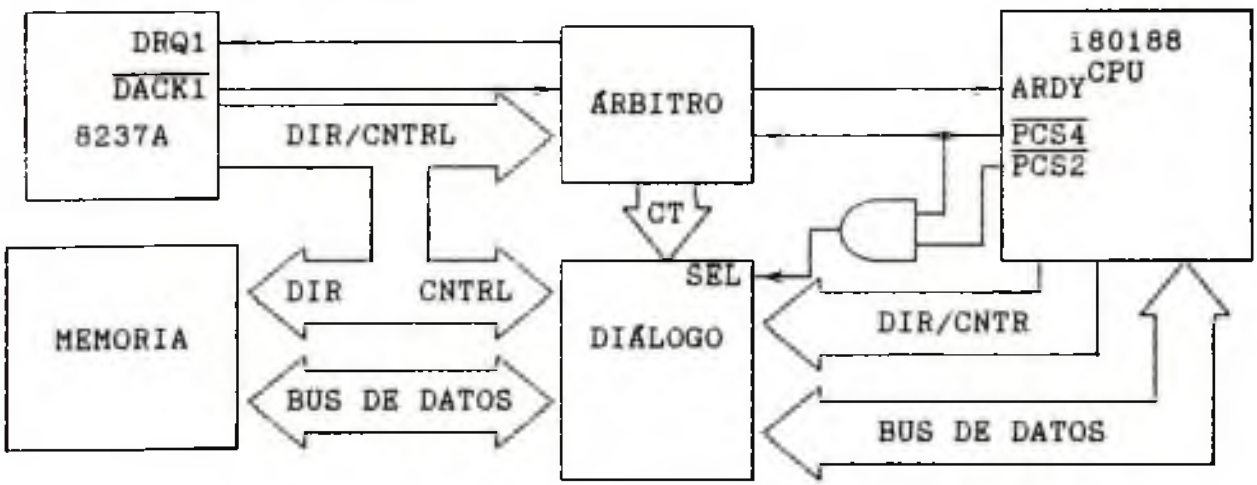


figura 3.11. Lógica de selección para transferencias por DMA.

La figura 3.12 nos muestra a bloques como en base a la circuitería "diálogo", obtenemos el paso de datos por medio de peticiones de DMA.



CT: Control.

figura 3.12. Circuitería para el paso de datos por DMA.

Una vez determinadas las dos distintas maneras en las que se seleccionarán los registros de datos, pasaremos al diseño del árbitro.

El árbitro es el intermediario entre las peticiones del i80188 y las atenciones del controlador 8237A. La secuencia de eventos que controlará son:

A) Se hace una petición de DMA mediante la señal **$\overline{\text{PCS4}}$**. En respuesta, el árbitro "congela" al i80188 llevando a nivel inactivo a la señal **ARDY** (bajo). Además hará la petición de una transferencia por DMA al canal 1 del 8237A con la señal **DRQ1** y esperará por la señal de reconocimiento **$\overline{\text{DACK1}}$**.

B) Cuando el 8237A nos indique, por medio de la señal **$\overline{\text{DACK1}}$** que la transferencia va a realizarse, el árbitro "descongela" al procesador i80188, llevando la señal **ARDY** a su nivel activo (alto). Entonces el procesador puede continuar la lectura/escritura que comenzó sin que el notara el tiempo de espera ni en que momento los datos fueron escritos/leídos en los registros.

Como podemos darnos cuenta, esta circuitería sirve para realizar una cita en la cual el 8237A por lo regular llega tarde y el i80188 debe de esperar. Estos eventos los podemos ver en el diagrama de flujo de la figura 3.13.

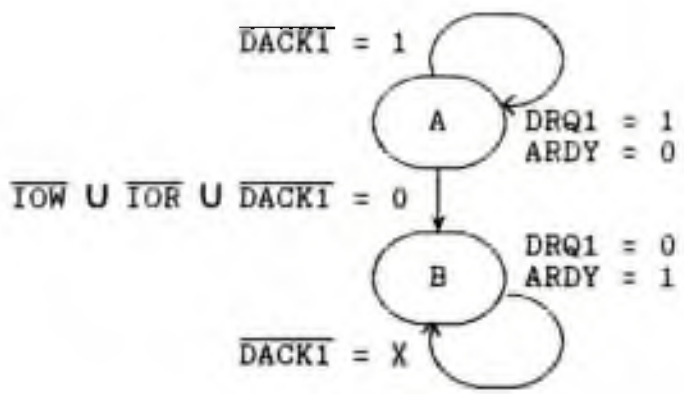


figura 3.13. Diagrama de estados del árbitro

El árbitro es una simple máquina de dos estados, la cual "existe" mientras la señal **PCS4** esté en su nivel activo. El estado inicial de esta máquina es A y es el estado de espera . La transición al siguiente estado, el B, es efectuada por el paso de la unión las señales **DACK1**, **IOW** e **IOR** a su nivel activo. Este estado es mantenido hasta que la máquina deja de "existir". La implementación de esta máquina se puede realizar en forma de circuitería con el uso de un flip flop y algunas compuertas lógicas para obtener las condiciones de transición (entradas del circuito) y los estados (salidas del circuito). La figura 3.14 nos muestra un circuito que puede realizar esto.

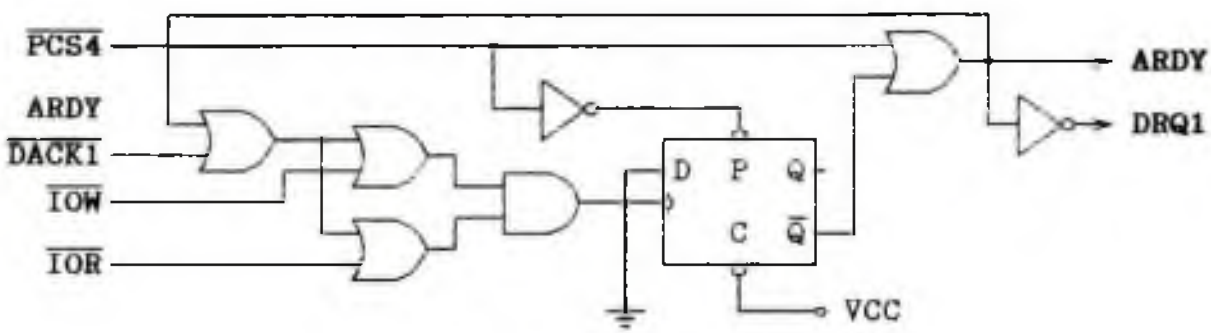
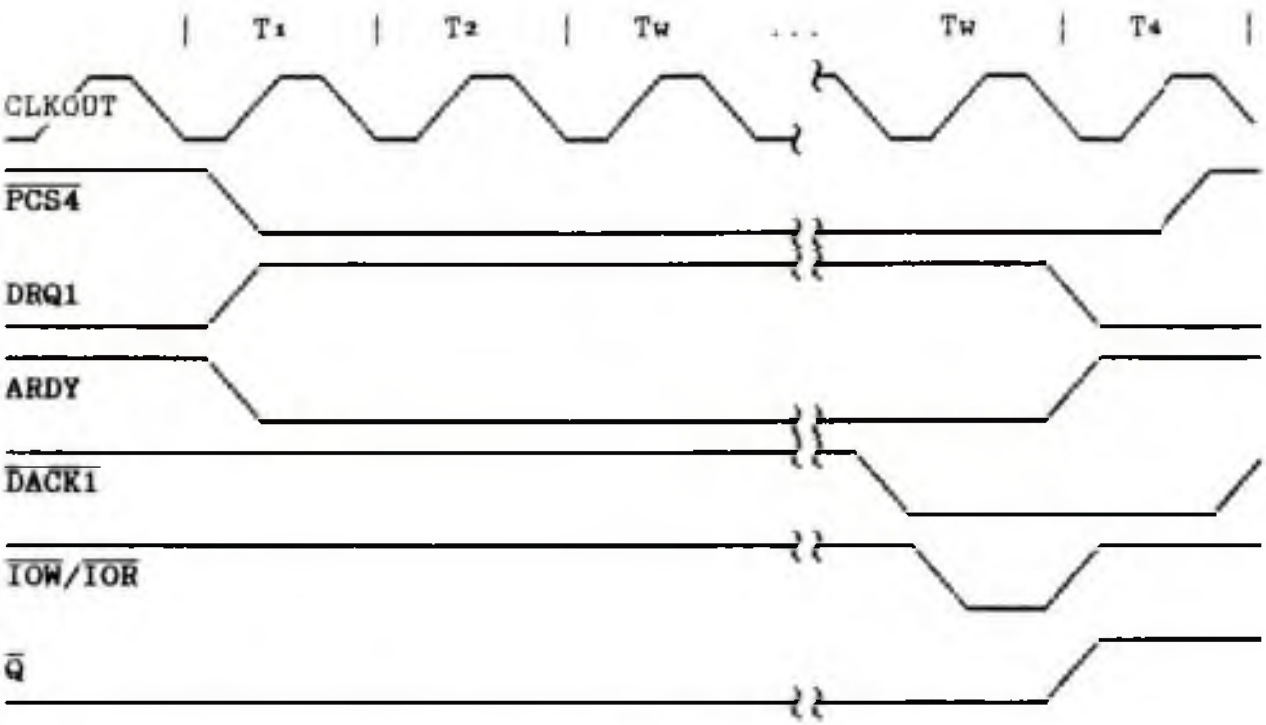


figura 3.14. Árbitro.

De la figura 3.14 podemos observar que, mientras la entrada **PRESET** del flip flop esté en "0", el flip flop estará bloqueado y su salida $\bar{Q}$ puesta a "0" . El flip flop sólo podrá operar cuando su entrada **PRESET** sea "1" y la salida $\bar{Q}$ cambiará de estado cuando haya un filo de subida a su entrada de reloj, este filo lo determina el fin de la condición mencionada anteriormente y que podemos observar en forma de lógica combinacional en la figura 3.14. Al cambiar de estado la salida $\bar{Q}$, ésta provocará el cambio de las señales **DRQ1** y **ARDY** . La inclusión de la señal **ARDY** en esta circuitería es con el objeto de que la transferencia de datos se realice sólo hacia la tarjeta que lo solicitó, ya que como recordaremos podemos tener instaladas en una IBM PC/XT/AT hasta ocho tarjetas y de esta manera se evitan colisiones. Esto se puede apreciar en el diagrama de señales de la figura 3.15.



$$ARDY = \overline{Q} \cup \overline{PCS4}$$

figura 3.15. Diagrama de temporización del árbitro

3.3. Diseño de la circuitería de enlace con la red.

La razón de ser de la tarjeta se define en esta sección, pues es aquí en donde se seleccionan y adicionan los elementos necesarios para comunicar a nuestra circuitería con dos redes muy populares, la X.25 y la RDSI (Red Digital de Servicios Integrados). Todo esto a nivel físico.

Primero, mencionaremos las características que debe de cumplir cada una de las partes y después pasaremos a la parte de adición a nuestro núcleo.

3.3.1. Enlace con la red pública X.25. Para las tres capas más bajas, el CCITT ha publicado recomendaciones que tienen que ser universalmente adoptadas por las redes públicas mundiales. Estas capas son conocidas colectivamente como X.25.

El protocolo de capa física o primer capa, llamado X.21, especifica las interfaces físicas, eléctricas y procedurales entre el ETD (Equipo Terminal de Datos) y la red. Actualmente muy pocas redes públicas soportan esta norma, porque esta requiere señales digitales(infraestructura que en la actualidad es difícil de poseer), más que analógicas. Esto será más importante en el futuro. Como una medida provisional, una interface similar a la norma RS-232C ha sido definida y esta es la V.24.

El mayor problema con RS-232C es que ésta sólo puede transmitir datos confiablemente a una distancia de 16.4 m con una máxima velocidad de 20 Kb/s. Si las líneas usadas son más largas, la relación de transmisión debe ser drásticamente reducida. Esta limitación es causada por el uso de líneas de señal abierta con una tierra común.

Una nueva norma, la RS-422A especifíca que cada señal deberá ser enviada diferencialmente sobre dos alambres adyacentes en un par trenzado. La relación de datos para esta norma es de 10 Mb/s para una distancia de 16.4 m o 100 Kb/s para una distancia de 1220 m.

Para tener una cierta flexibilidad en el coprocesador, en lo que se refiere a la transmisión de datos, utilizaremos estas dos normas, por lo tanto la tarjeta debe de poseer la siguientes

características:

a) Interface procedural. Estará provista por la norma RS-232C y RS-422A.

b) Interface eléctrica. Estará provista por las normas RS-232C y la RS-422A.

c) Interface física. Debido a que la norma RS-232C no especifica un conector, nosotros untilizaremos el conector DB25. Y para la norma RS-422A será la norma RS-449 o el conector DB37.

Desarrollo. El dispositivo que puede cumplir parcialmente con la norma RS-232C es el Controlador para Comunicaciones Seriales Z8530 de la empresa Zilog. Este dispositivo, además de proveer las características de RS-232 tiene capacidades del nivel de enlace de X.25, como son inserción y borrado de bit 0, protocolo HDLC, verificación del código redundante y formación de la trama.

Este dispositivo sólo maneja voltajes TTL, por lo tanto, para completar la norma RS-232, en lo que se refiere a los voltajes, adicionaremos a este dispositivo, unos dispositivos que los provean. Estos son el MC1488 para conversión de TTL a voltajes RS-232C y el MC1489, para conversión de voltajes RS-232C a TTL.

Ahora, para conectar este dispositivo al sistema del i80188, sólo basta conectar el bus de direcciones, de datos y control de este sistema al Z8530. El reloj utilizado es el provisto por el i80188, con un flip-flop para dividir la frecuencia de 8 a 4MHz, tal como lo indica el manual. Para el sistema de reset, fue necesario adicionar dos compuertas "Y", ya que este se efectúa bajando la señal de $\overline{WR}$ y $\overline{RD}$ al mismo tiempo.

Uno de los detalles que deben quedar bien afinados en este dispositivo, es la temporización. Por lo tanto tenemos que cumplir con los tiempos marcados por Zilog.

Con la operaciones de escritura y lectura a registros tuvimos pequeños problemas de temporización, los cuales se solucionaron fácilmente utilizando tiempos de espera por parte del procesador, ya que este resultó más rápido que nuestro dispositivo periférico.

La señal **INTACK** no es tomada del procesador i80188, debido a que utilizamos al controlador de interrupciones del i80188 en modo no cascada, por lo tanto no hay generación de esta señal. Debido a esto, la señal **INTACK** se tiene que producir artificialmente por medio de intersección de las señales **A0** y **PCS1**. Esto sucede cuando el Z8530 provoca una interrupción en el i80188 y éste en respuesta (por medio de una rutina antes establecida) lee una dirección par, claro que la dirección debe de estar dentro del rango del **PCS1**, para que de esta forma se genere la señal **INTACK**.

Esta señal presentó problemas (aún con los tiempos de espera) con respecto a la señal **RD** durante una operación de interrupción, ya que estas tienen un tiempo de separación mínimo de 250 ns, necesarios para la estabilización de la circuitería interna. Esto se puede observar mejor en la figura 3.16, tal como lo marca Zilog.

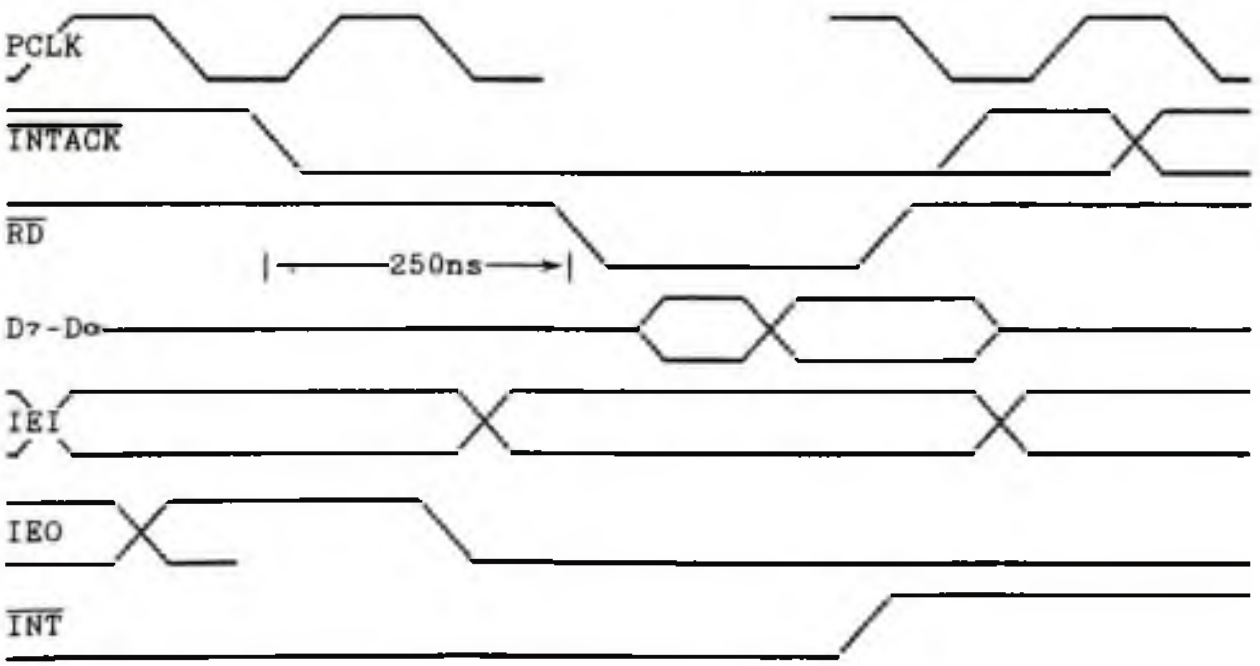


figura 3.16. Temporización de un reconocimiento de temporización

En nuestro sistema, debido a la velocidad del i80188, esta distancia se reduce a 140 ns los cuales son insuficientes para un buen funcionamiento de este dispositivo, esto lo podemos apreciar en la figura 3.17.

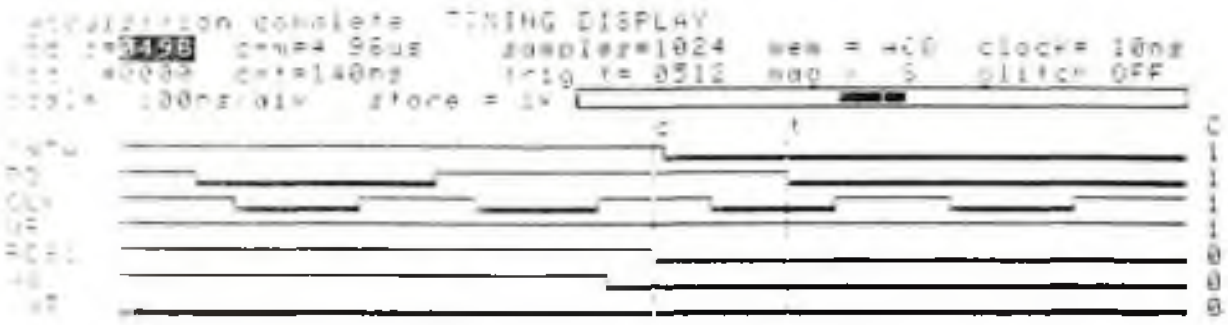


figura 3.17. Distancia incorrecta entre las señales $\overline{\text{INTA}}$ y $\overline{\text{RD}}$.

La manera en la que solucionamos este pequeño detalle, fue adicionando un retraso a la señal $\overline{\text{RD}}$, por medio de un flip-flop, el cual se encuentra temporizado por el reloj de 4MHz. El retraso consiste en prolongar la señal $\overline{\text{RD}}$ aproximadamente medio ciclo de reloj, o sea 125 ns. La circuitería que ejecutará esto es la mostrada en la figura 3.18.

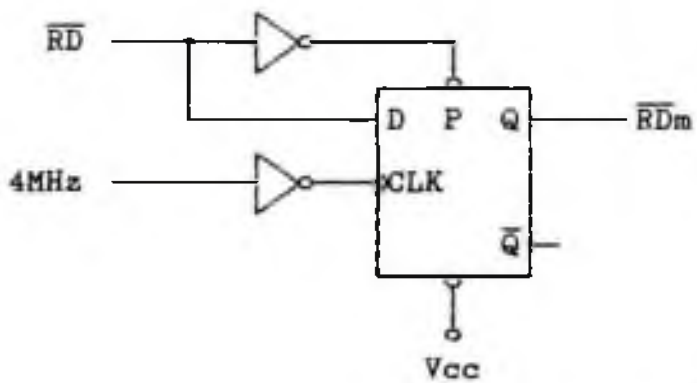


figura 3.18. Retraso de la señal $\overline{\text{RD}}$.

Como podemos observar en la figura 3.18, la señal $\overline{\text{RD}}$ invertida pone a "1" a la salida Q del flip-flop y el funcionamiento de esta circuitería se inicia con la aparición de la señal $\overline{\text{RD}}$, la cual no se hace visible en la salida Q, hasta que ha sucedido un filo de bajada del reloj de 4MHz. Cuando esto suceda, la señal $\overline{\text{RD}}$ ya ha sido retrasada y ahora la distancia

entre la señal **INTACK** y la señal **RD** es de 280 ns. Esto se puede ver claramente en la figura 3.19, la cual fue tomada con el analizador de estados lógicos GOULD K50.

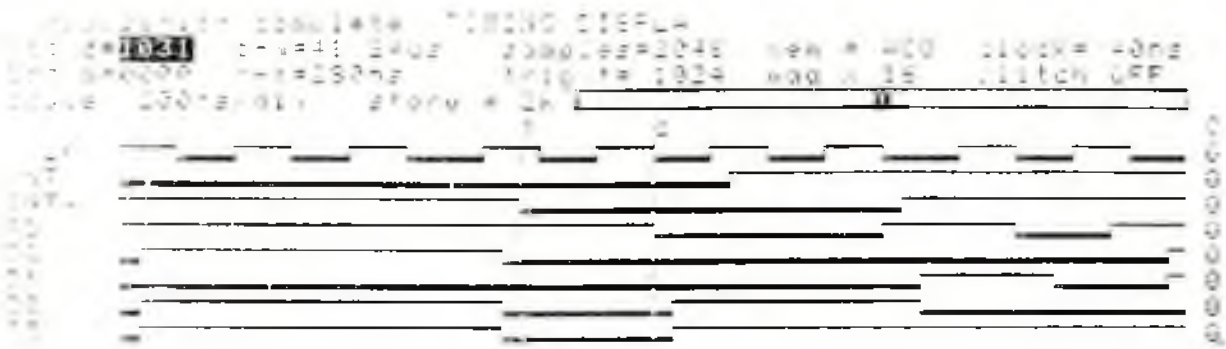


figura 3.19. Señal RD retrasada con respecto a la señal INTA.

Ahora con respecto a la adición de las interfaces RS-232C y RS-422A, esta se logra aprovechando que el Z8530 tiene dos canales. Se decidió que la interfaz RS-232C estuviera en el canal A y la interfaz RS-422A quedara en el canal B. La circuitería para el enlace a la red X.25 se puede observar en el diagrama general o en el diagrama 5 del apéndice A.

3.3.2. Enlace con la Red Digital de Servicios Integrados

La circuitería que añadiremos al coprocesador le dará la capacidad de enlazarse a nivel físico y estará acorde con las recomendaciones I del CCITT, las capacidades que se pretende que tenga este coprocesador son:

a) Operar como Equipo Terminal.
b) Capacidad de acceso al canal D.
c) Utilizar la interface S en acceso básico.

El dispositivo que nos puede proveer estas características y además parte de las del nivel de enlace, como son inserción y borrado del bit cero, formación de tramas, protocolo LAPD y otros, es el Circuito de Interface de Abonado a la Red MT8930 de la

empresa MITEL. Este dispositivo fue diseñado para implementar las recomendaciones I.430 para interface S y T para RDSI. Además de que fue hecho para adicionarse a sistemas con microprocesadores de la familia INTEL y MOTOROLA.

Para nuestro caso, la adición de este dispositivo se hizo de manera sencilla, ya que nuestro sistema está basado en un μprocesador i80188 de la empresa INTEL. La conexión de éste se puede apreciar en el diagrama general o en diagrama 6 del apéndice A.

3.4. Diseño de la circuitería de acceso al bus de la IBM PC/XT/AT

El uso de una circuitería destinada a estos fines, permite al coprocesador dejar de operar como un simple dispositivo periférico ya que le da la facilidad de usar la memoria de la IBM PC/XT/AT y operar como sistema multiprocesador. La idea principal es que cada procesador pueda operar independientemente, buscando y ejecutando instrucciones desde su propia memoria, hasta que surja, por parte del coprocesador, la necesidad de accesar la memoria de la IBM PC/XT/AT. En este sistema, el procesador que tiene el bus es conocido como "maestro".

Una cuestión que puede ocurrir en este punto es, ¿Que sucede si los dos maestros en el sistema de bus intentan usarlo al mismo tiempo?. La respuesta a esta cuestión es que el sistema de bus debe de contener una lógica, la cual en alguna forma "arbitre" la disputa y de la seguridad de que sólo un maestro a la vez, acceda a las señales de control en el bus.

3.4.1. Advertencia. Esta parte del núcleo permitiría al coprocesador realizar accesos de lectura/escritura en la memoria de la IBM PC/XT/AT y aún correr programas residentes en ésta, como si se tratara de su propia memoria. Esto, mediante peticiones del BUS al controlador de DMA 8237A en modo cascada, lo cual traería enormes ventajas, ya que el procesador i80188 podría meter y sacar todos los datos que él quisiera, de la memoria de la IBM PC/XT/AT sin necesidad de ningún acuerdo (en software) entre ellos, además de que esto se ejecutaría a una velocidad mayor que si lo hicieramos a través de la circuitería "diálogo". Por lo tanto el software se simplificaría grandemente. Claro, el hardware aumentaría considerablemente, pero valdría la pena.

Desafortunadamente la IBM PC/XT/AT no permite el acceso de su bus a otro procesador, por la sencilla razón de que el hardware está implementado de tal manera que el bus está, o en manos del procesador o en manos del controlador 8237A y no se admiten terceros. Esto es una grave desventaja, ya que en estas máquinas definitivamente no se permite una operación multiprocesador. De todas formas, esta circuitería se implementó

esperando obtener algún resultado positivo, lo cual no fue posible. Aún así, incluímos el diseño de esta circuitería para desarrollos futuros, la cual puede funcionar en máquinas con arquitectura con capacidad para operación multiprocesador, tal como una PS/2.

3.4.2. **Desarrollo**. Lo primero que hay que hacer para diseñar esta circuterría, es decidir con que rango de direcciones, el i80188 podrá accesar la memoria de la IBM PC/XT/AT. Para esto tenemos que dejar bien definido cual va a ser el mapa de memoria del i80188. El mapa se muestra en la figura 3.20.

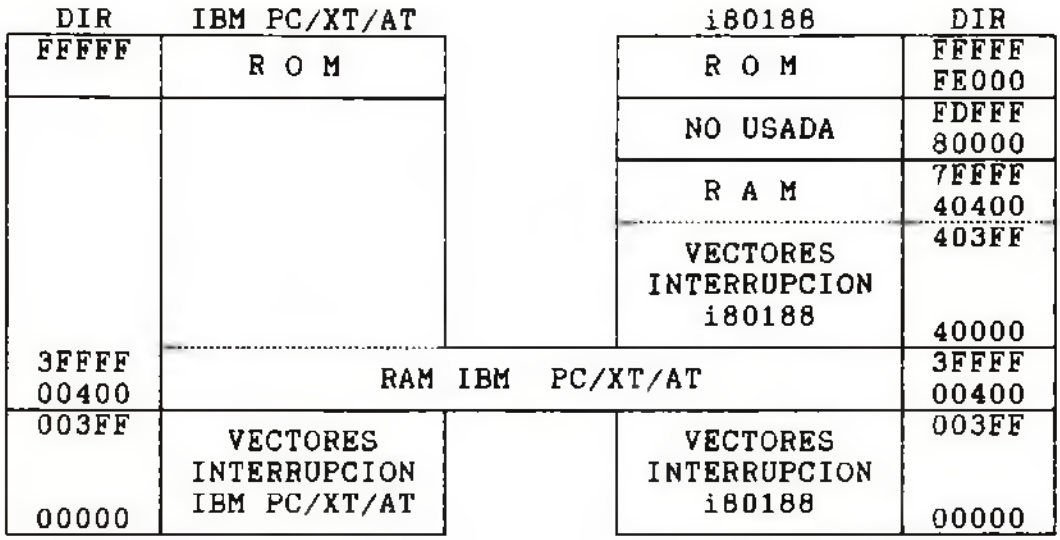


figura 3.20. Mapa de memoria de las dos máquinas en conjunto.

La parte interesante en la implementación de esta circuitería se puede observar en la figura 3.20 y es el rango de memoria comprendido entre las direcciones 00400H y la 3FFFFH, ya que cuando el procesador i80188 accese esta área, entrará en operación la circuitería de control de acceso al bus de la IBM PC/XT/AT. Entoces, los puntos que debe de cubrir esta circuitería son los siguientes:

a) Se debe de implementar una lógica de selección, ayudada

por las terminales $\overline{MCSi}$ para activar la circuitería de acceso al bus de la IBM PC/XT/AT sólo cuando se accese al área comprendida entre las direcciones 00400H a la 3FFFFH. El rango de memoria medio debe de estar programado para que sus terminales respondan en la primera mitad del Megabyte total direccionable y por lo tanto, cada $\overline{MCSi}$ podrá seleccionar 128K bytes de memoria.

B) Con respecto al área de vectores de interrupción existe un problema ya que estos necesitan una área de RAM y su lógica de selección. El problema principal no es asignarles un lugar, sino como habilitarlo, ya que las terminales de selección programables están ocupadas en toda la primera mitad del megabyte direccionable.

c) La circuitería de acceso al bus de la IBM PC/XT/AT consistirá de buffer y transceivers de aislamiento entre los dos buffers, además de un árbitro que medie las peticiones del i80188(señal $\overline{DRQ1}$) y las atenciones del controlador (señal $\overline{DACK1}$) de DMA 8237A, que es a quién

La arquitectura del sistema multibus que se desea implementar para realizar las funciones planteadas en los puntos anteriores es mostrada en la figura 3.21.

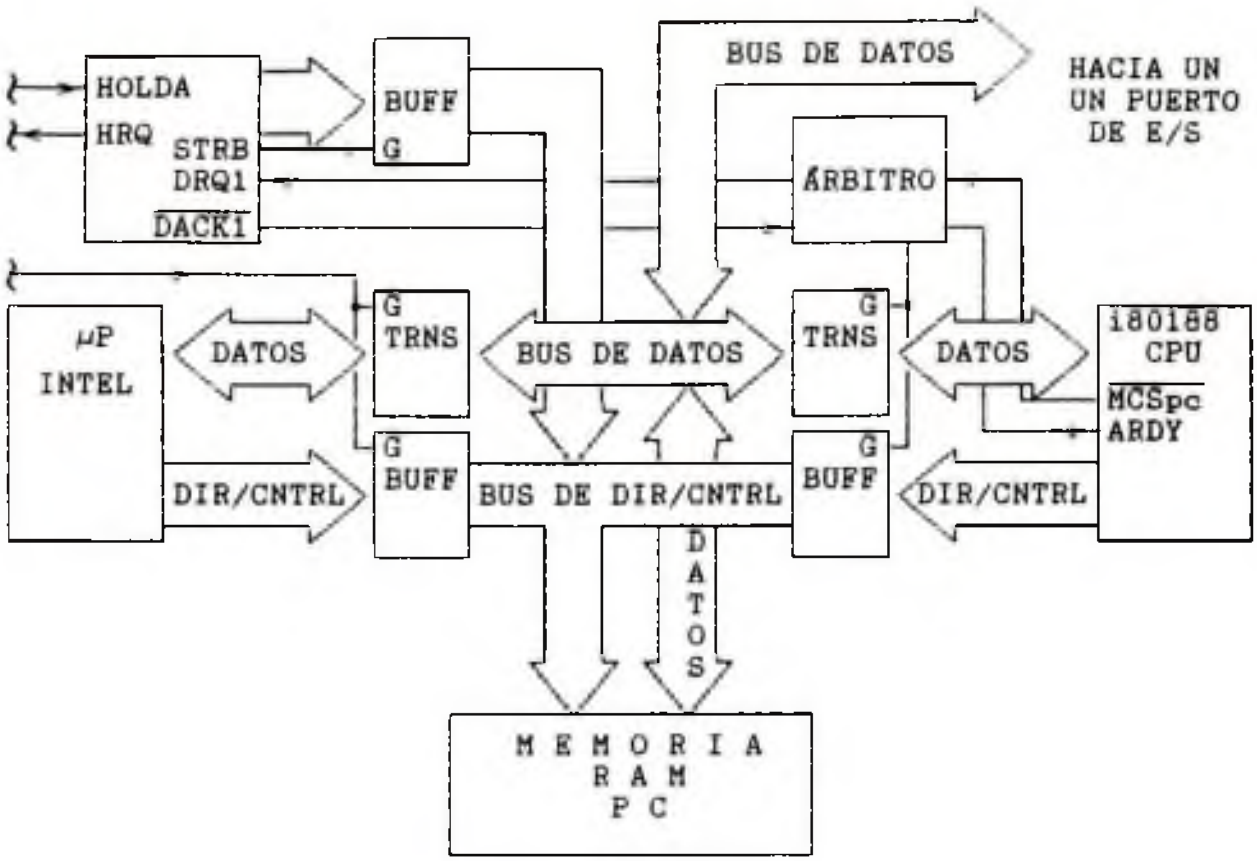


figura 3.21. Circuitería de acceso al bus de la IBM PC/XT/AT.

Siguiendo la secuencia de los objetivos, comenzaremos por definir una lógica de selección que propocione la señal de activación de la circuitería de acceso al bus de la IBM PC/XT/AT a la cual llamaremos **$\overline{\text{RAMpc}}$**. Esta se consigue suponiendo que la lógica de selección programable del i80188 está configurada para que la terminal de selección **$\overline{\text{MCS0}}$** responda en el rango de 00000H a 1FFFFH y **$\overline{\text{MCS1}}$** en el rango de 20000H a 3FFFFH. Ahora, para lograr que la señal de habilitación **$\overline{\text{RAMpc}}$** se produzca arriba de los vectores de interrupción, se necesita que una señal excluya a la señal **$\overline{\text{MCSpc}}$** durante el rango de los vectores. Esto se logra programando el registro del **$\overline{\text{LCS}}$** (Lower memory Chip Select) en el rango 00000H a 003FFH. Aquí notamos que este rango de selección se traslapa con el rango de selección de **$\overline{\text{MCS0}}$**, lo cual es una condición prohibida, pero aún así pasamos por encima de esta

advertencia arriesgandonos a una colisión. Afortunadamente en la práctica, los resultados fueron satisfactorios. El incluir una lógica de selección para evitar romper esta regla, nos hubiera salido muy caro, ya que habría que diseñar una lógica que tomara en cuenta los 20 bits de direcciones y esto es completamente inconveniente.

Otro factor que tenemos que tomar en cuenta es el refresco de memoria, el cual consistirá en la lectura de posiciones contíguas de memoria en todo el rango de direccionamiento, por medio de ciclos de DMA. Para esta circuitería esto es inconveniente, ya que en algún momento el refresco entrará en el rango de la memoria RAM de la IBM PC/XT/AT produciendo las señales $\overline{\text{MCS0}}$ y $\overline{\text{MCS1}}$ y que estas a su vez producirán la señal $\overline{\text{MCSpc}}$, produciendo así, el refresco también en esta memoria.

La única manera en la que podemos evitar parcialmente esto, fué añadiendo a la lógica de selección una señal más, la S6, que en su estado activo alto nos informa que operación de DMA se está realizando, tal como se muestra en la figura 3.22.

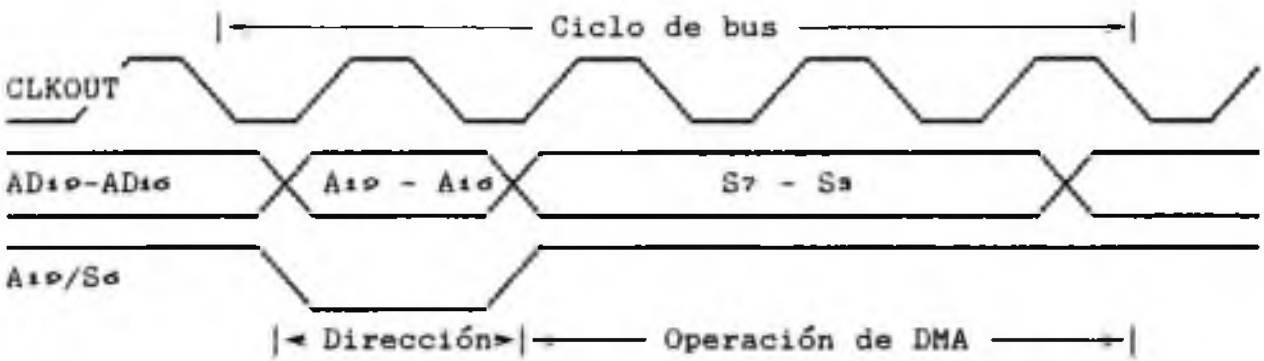


figura 3.22.

Esta señal es útil sólo en los tres últimos pulsos de reloj del ciclo de bus, debido a que está compartiendo la terminal con la señal de dirección A19. Esto quiere decir que cuando se esté realizando un refresco de memoria dentro del rango de la memoria de la IBM PC/XT/AT, aún incluyendo la señal de estado S6, se estará generando la señal $\overline{\text{MCSpc}}$ con una duración muy pequeña. Este

pequeño detalle lo resolveremos más adelante, durante el diseño del árbitro.

En base a todas las consideraciones anteriores, la lógica de selección para producir la señal $\overline{\text{RAMpc}}$ debe de tomar en cuenta que:

a) Esta sólo puede ocurrir cuando se produzca la señal $\overline{\text{MCS0}}$ o la señal $\overline{\text{MCS1}}$.

b) Y no se trate de una operación de DMA (señal S6).

c) Y además no se esté accesando el área de los vectores de interrupción (señal $\overline{\text{LCS}}$).

Por lo tanto la lógica de selección debe estar regida por la fórmula:

$$\overline{\text{MCSpc}} = ((\overline{\text{MCS0}} \wedge \overline{\text{MCS1}}) \cup S6) \cup \overline{\overline{\text{LCS}}}$$

E implementada en circuitería la podemos observar en la figura 3.23.

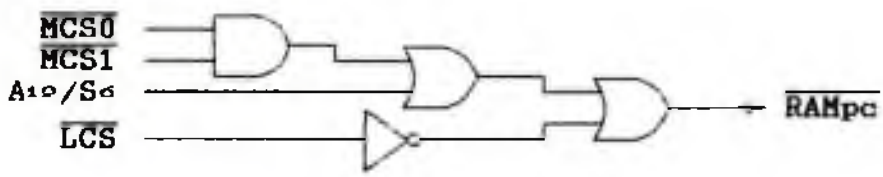


figura 3.23.

Selección del área de los vectores de interrupción. Como sabemos, esta área se encuentra localizada a partir de la dirección 00000H hasta la 003FFH y no puede ser modificada como en el caso de otros procesadores. Por lo tanto, tenemos que encontrar la manera de localizarla en algún lugar y que mejor lugar que en la RAM. Podríamos adicionar una área de memoria adicional para estos fines, pero es poco eficiente y además costoso, sobre todo, porque necesitaríamos una lógica de selección que no utlizará las terminales $\overline{\text{MCS}}$. Bueno, lo que nos queda por hacer es modificar la manera en la que se selecciona la RAM y como se acordó anteriormente, se va utilizar la terminal $\overline{\text{LCS}}$ para seleccionar el área de vectores. Por lo tanto, la RAM se va a habilitar en los siguientes casos:

a) Por medio de la señal $\overline{\text{LCS}}$. Cuando se seleccione el rango de memoria 00000H a 003FFH.

b) O cuando se genere la señal **MCS2**. Durante un acceso al rango 40000H a 5FFFFH.

c) O Además, cuando se genere la señal **MCS3**. Durante un acceso al rango 60000H a 7FFFFH.

Como podemos observar, esto trae un área de traslapamiento la cual se puede evitar con el incremento de hardware. Esto no es muy necesario, ya que se puede evitar indicando al programador que el rango de direcciones 40000H a 403FFH está reservada para los vectores de interrupción, tal y como se puede observar en la figura 3.20. Y la lógica de selección se puede apreciar en la figura 3.24.

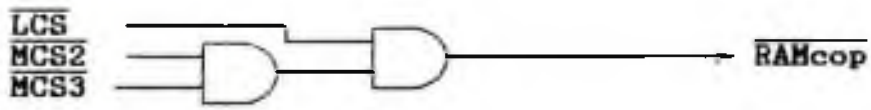


figura 3.24. Lógica de selección para la RAM.

3.4.3. Diseño del árbitro.

Este es un intermediario entre el procesador i80188 y el controlador de DMA 8237A, y es una circuitería que temporiza, por medio de un sencillo protocolo, los accesos del i80188 al bus de la IBM PC/XT/AT. Esta circuitería además debe de distinguir entre un acceso normal y un ciclo de refresco de memoria. En caso de que se trate de un ciclo de refresco debe de ignorarlo. Esta debe tener la forma de la figura 3.25.

i80188 IBM PC/XT/AT

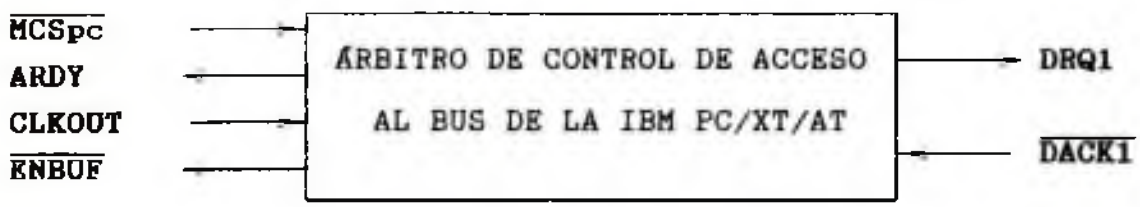


figura 3.25. Circuitería de acceso al bus de la IBM PC/XT/AT

La secuencia de operación del árbitro mostrado en la figura 3.25 es la siguiente, en donde cada inciso es un estado, los cuales se encuentran temporizados por el reloj **CLKOUT**.

A) En este estado se está en espera del nivel activo de la señal **$\overline{\text{MCSpc}}$** la cual ocurre cuando se accesa el área de memoria comprendida entre 003FFH y 3FFFFH, si esto sucede, el árbitro se activa y se produce una transición al estado B.

B) Estando el árbitro en el estado B, si la señal **$\overline{\text{MCSpc}}$** continúa en su nivel activo, entonces pasa el estado C, de lo contrario no se trató de una petición de DMA y retorna al estado A.

C) El período de la señal **$\overline{\text{MCSpc}}$** es mayor al período de un pulso de **CLKOUT**,entonces se trata de una petición de DMA al 8237A. En respuesta a esto, el árbitro debe solicitar el bus de la IBM PC/XT/AT por medio de la señal **DRQ1** con un nivel alto y además debe de congelar al procesador por medio de la señal **ARDY**, llevándola a su nivel inactivo (bajo). Hecho esto el árbitro debe esperar en el estado C hasta que 8237A le indique por medio de la señal **$\overline{\text{DACK1}}$**, con un estado activo bajo, que puede pasar al estado D.

D) En este estado se desarrollan dos eventos, se desbloquea el procesador, levantando la señal **ARDY** y se habilitan los buffers llevando a la señal **$\overline{\text{ENBUF}}$** a su nivel activo (nivel bajo), con esta señal los buffers y transceivers se habilitan para conectar el bus del i80188 con el bus de la IBM PC/XT/AT, pasando así la dirección, datos y señales de control. En este punto el árbitro se bloquea sosteniendo su estado hasta el fin del ciclo.

Cuando ocurrre el fin del ciclo, el árbitro se desactiva y vuelve a su estado inicial hasta que la señal **$\overline{\text{MCSpc}}$** suceda de nuevo. Esto lo podemos representar esquemáticamente con el siguiente diagrama de estados, de la figura 3.26.

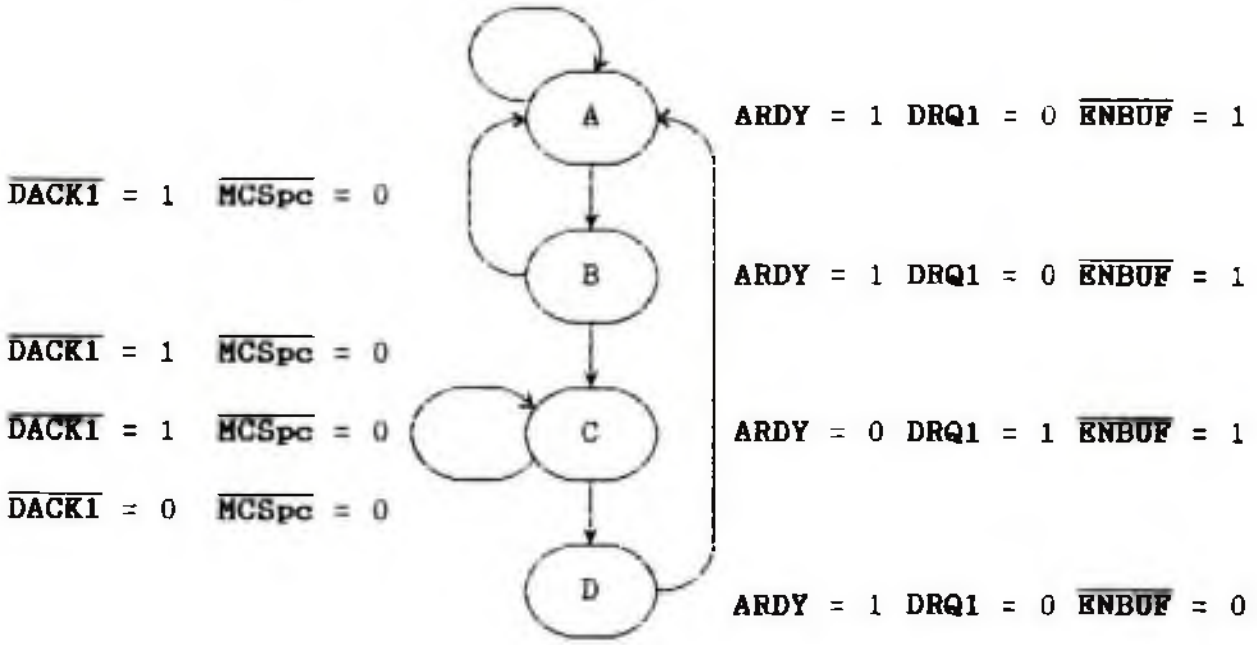


figura 26. Diagrama de estados del árbitro.

El diagrama eléctrico de este árbitro lo podemos observar en el diagrama del apéndice B y todo el sistema de acceso al bus de la IBM PC/XT/AT en el diagrama general1. En la figura 3.27 representamos la señales que debe recibir y entregar el árbitro.

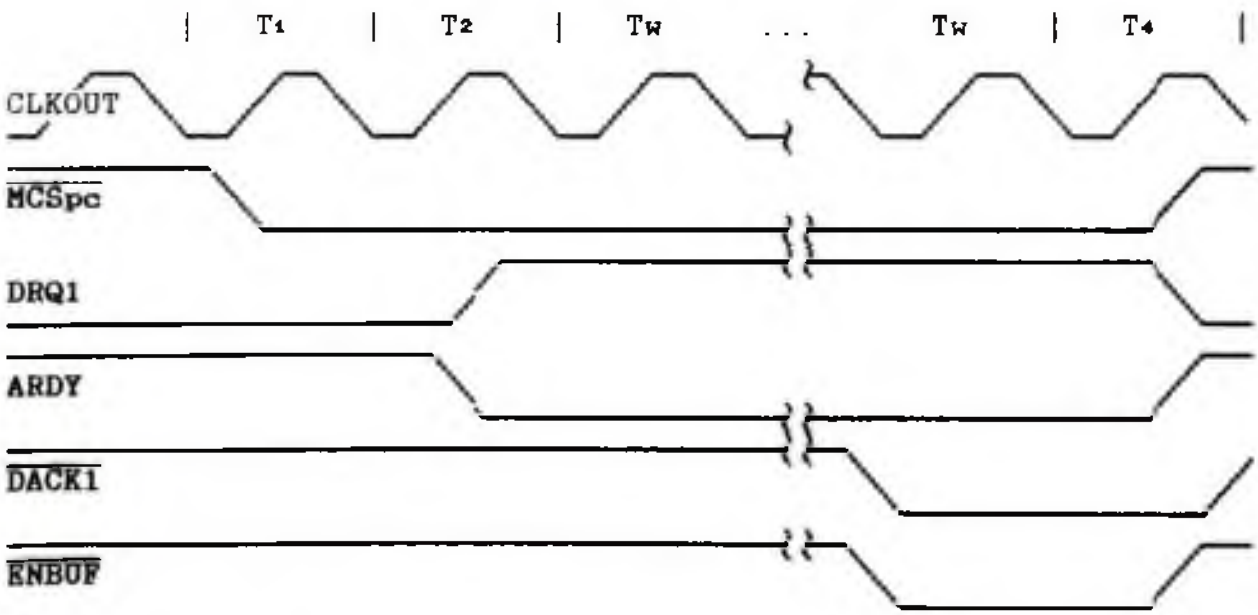


figura 3.27. Señales esperadas para el acceso al bus de la IBM PC/XT/AT.

En la figura mostrada arriba podemos observar el protocolo que debe seguir el árbitro para desarrolar un acceso al bus de la IBM PC/XT/AT. Observemos como todo comienza con la señal **$\overline{\text{MCSpc}}$** la cual al bajar, debe habilitar toda la circuitería del árbitro. Esta al ser habilitada va a sensar que esté presente la señal **$\overline{\text{MCSpc}}$**, si está presente durante el pulso de reloj T_1 el árbitro no responde, pero cambia de estado. Si durante el pulso T_2 la señal **$\overline{\text{MCSpc}}$** sube, la circutería es desabilitada y todo queda sin cambio alguno. Pero si perdura, quiere decir que no se trataba de un ciclo de refresco de memoria sino de un acceso normal. Bueno, pues en consecuencia a esto el árbitro debe responder con las señales **DRQ1** y **ARDY**, además que se debe bloquear en espera de **$\overline{\text{DACK1}}$**. Transcurrido cierto tiempo, cuando ocurra la señal **$\overline{\text{DACK1}}$**, el árbitro debe responder habilitando los buffers para que las direcciones y señales de control puedan pasar de la tarjeta a la PC y los transceivers para que los datos puedan pasar en cualquiera de los dos sentidos, el cual está determinado por la terminal del procesador **DT/$\overline{\text{R}}$** (Data Transmit/ Receive). En este punto el procesador puede continuar, esto lo puede hacer si le indicamos por medio de la señal **ARDY** que puede continuar.

Una de las cualidades de esta circuitería que hasta aquí no hemos mencionado, es que ésta puede accesar buses de datos de 8 y 16 bits, esto lo realizaremos con una circuitería que hace una conversión de un bus de 8 bits (i80188) a una máquina de 8 o 16 bits.

Para el caso de una máquina de 16 bits, esto se logra con la inclusión de alguna pequeña lógica la cual determina si el dato a transmitir tiene dirección par o dirección impar. La señal que nos puede ayudar en esto es la señal **$\overline{\text{BHE}}$** (Bus High Enable), que es la señal **A0** pero invertida. Estas dos señales van a servir para habilitar cualquiera de los dos transceivers, si se presenta la señal **A0** en su nivel bajo, se trata de una dirección par y se tiene que habilitar el transceiver para los 8 bits menos significativos de la palabra. Si se presenta la señal **$\overline{\text{BHE}}$** se trata de una dirección impar y se tiene que habilitar el transceiver de los 8 bits más significativos.

Para una máquina de 8 bits, configuramos la lógica de

selección para que por medio de un dip switch, se indique que estamos trabajando con una máquina de 8 o con una máquina de 16 bits. Y si se selecciona la opción para la máquina de 8 bits, que los dos transceivers estén siempre habilitados. La lógica de selección que nos puede ayudar en los dos casos mencionados, se muestra en la figura 3.28.

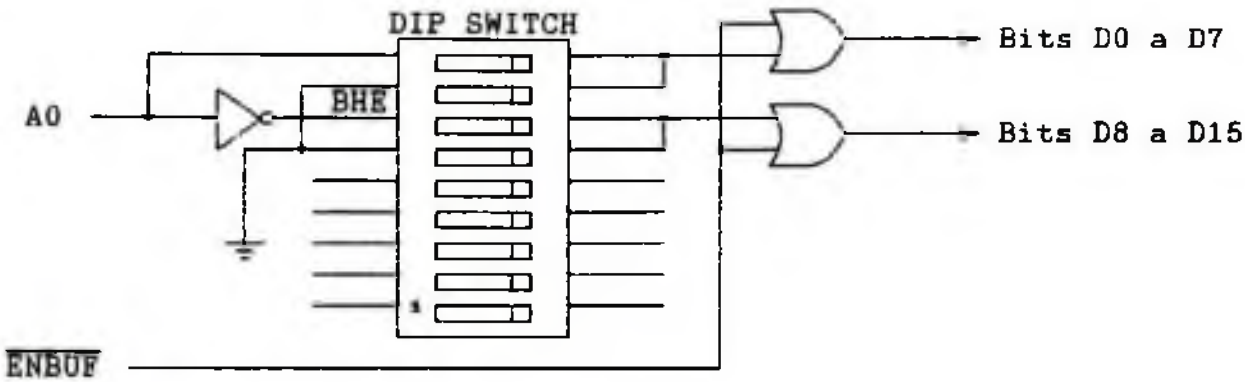


figura 28. Lógica de conversión para un bus de datos de 16 bits.

Como una prueba de que nuestro trabajo con respecto a esta sección es correcto y que la circuitería puede servir en máquinas con diferente arquitectura, incluimos algunas impresiones de las señales medidas en la tarjeta, cuando ésta tenía implementada esta circuitería. Las mediciones se hicieron utilizando el analizador de estados lógicos GOULD K50.

figura 3.29. Señales para el acceso al bus de la IBM PC/XT/AT.

En la figura 3.29 podemos observar parcialmente el protocolo de acceso al bus de la IBM PC/XT y podemos apreciar como al ser habilitados los buffers/transceivers aún en el momento correcto, sucede una colisión de datos entre los bits del bus de datos del i80188 y los bits del bus de datos de la IBM PC/XT. Obsérvese como cuando se habilitan los buffers(señal **$\overline{\text{ENBUF}}$** indicada con la marca **m** y con el cursor **c**),la tarjeta manda los bits etiquetados TR_A2 y TR_A3 (bit de dirección A2 y A3) con un valor de "0" hacia los bits de dirección A2 y A3 de la IBM PC/XT, etiquetados como PC_A2 y PC_A3, los cuales aparentemente no son recibidos, ya que en estas etiquetas se observan valores contrarios. No se trata de eso, sino de una pelea entre señales, en los cuales en los bits de dirección A1, A2 y A3 el bus de la PC siempre gana la batalla. El resultado de esta lectura es obviamente erróneo ya que se están haciendo lecturas a lugares indeseados. Esto, como ya dijimos anteriormente no tiene solución, en una IBM PC/XT/AT.

4. EL SOFTWARE DEL SISTEMA.

La circuitería de cualquier sistema basado en un microprocesador, es inútil sin un software de inicialización y de comunicación con el mundo exterior.

El software que nosotros desarrollamos para darle vida al sistema está dividido en dos partes que son, el software de inicialización y las herramientas de trabajo.

4.1. El software de inicialización. Es un programa que reside en la memoria ROM y que se inicia en la dirección física de memoria FFFF0h (Dirección a la que salta el procesador después de un RESET por hardware). Su principal función es la de definir las posiciones y funciones de las diferentes partes del sistema. Estas son:

a) Definición de las posiciones de los bloques de memoria.
b) Definición de las posiciones de los dispositivos periféricos.
c) El refresco de memoria.
d) Inicialización de los segmentos.
e) Definición de rutinas primitivas de monitoreo.

4.1.1. Definición de las posiciones de los bloques de memoria.

Se refiere a la localización que tienen los dispositivos de almacenamiento primario dentro del rango de direccionamiento de 1Mbyte. Esta posición depende del diseño del hardware y se basa en el mapa de memoria mostrado en la tabla 4.1.

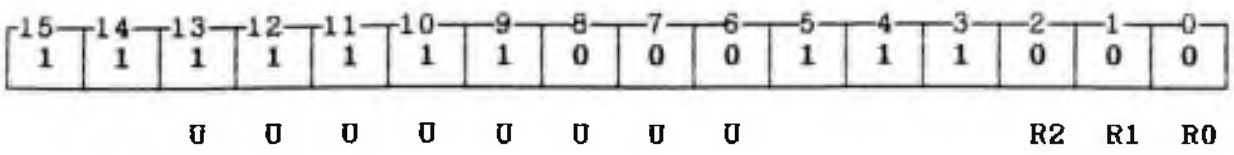
Tabla 4.1. **Mapa de memoria.**

Dirección	Descripción	Función
00000- 3FFFF	R A M	Área para alojar los vectores de interrupción. Puede ser utilizada para contener programas/datos.
04000- FDFFF	No usada	
FE000- FFFFF	R O M	Utilizada para almacenar rutinas de de inicialización.

El procesador i80188 posee la capacidad de seleccionar bloques de memoria, con ayuda de terminales dedicadas a estos fines. El tamaño y la posición de estos bloques de memoria es definido por software, mediante la debida programación de un conjunto de registros. El i80188 provee 6 terminales de selección y para tres distintas áreas: Memoria superior, Memoria inferior y memoria intermedia.

4.1.1.1. El bloque de memoria superior (ROM).

El i80188 provee una terminal de selección llamada $\overline{\text{UCS}}$, para el tope de la memoria. El tope de la memoria es usualmente empleado como sistema de memoria después del reset, ya que el i80188 comienza ejecutando en memoria en la localización FFFF0h. El límite superior de este bloque está siempre en la dirección FFFFFh, mientras que el límite inferior se obtiene indirectamente al programar el tamaño del bloque. El tamaño del bloque que nosotros seleccionamos dependió de la capacidad de la memoria ROM que nosotros utilizamos. Por medio de la palabra de control, además de definir el tamaño del bloque, nosotros podemos definir el número de estados de espera que debe de realizar el i80188 sobre la memoria en la cual se encuentre trabajando(Tabla 4.2). Por lo tanto, la palabra de control queda como se muestra en la figura 4.1:



offset:A0h contenido:FE38h

R0,R1 y R2 : Por medio de estos bits se indica que no se realice inserción de estados de espera y que se atienda la señal READY externa (Ver tabla 4.2).

U : Definen el tamaño del bloque de memoria, por medio de la relación:

$$2^{(8 - \#\text{de bits U})} \text{ Kbytes.}$$

La ROM que utilizamos tiene una capacidad de 8Kbytes, por lo tanto ponemos 5 bits U comenzando por la izquierda, lo cual nos da:

$$2^{(8 - 5)} \text{ Kbytes} = 8\text{Kbytes.}$$

Lo cual da como resultado un límite inferior en la dirección FE000H.

Otros : Los demás bits se encuentran alambrados a 1.

figura 4.1. Registro UMCS.

El offset del registro **UMCS** es con respecto a la posición del inicio del bloque de registros de los periféricos internos, el cual se encuentra localizado en la dirección FF00h del espacio E/S.

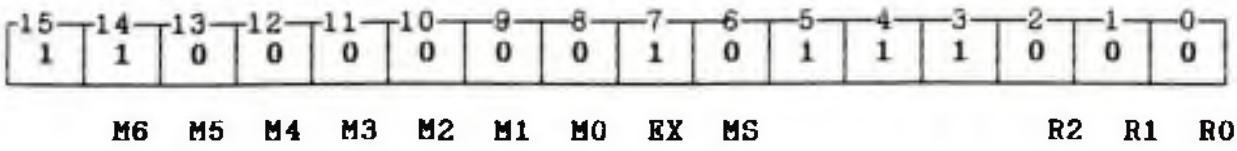
Tabla 4.2. **Programación de los bits READY.**

R2	R1	R0	Numero de estados de espera generados	RDY externo
0	0	0	0 estados de espera.	Atendido
0	0	1	1 estado de espera insertado.	
0	1	0	2 estados de espera insertados.	
0	1	1	3 estados de espera insertados.	
1	0	0	0 estados de espera.	Ignorado
1	0	1	1 estado de espera insertado.	
1	1	0	2 estados de espera insertados.	
1	1	1	3 estados de espera insertados.	

4.1.1.2. El bloque de memoria intermedia (RAM).

Nosotros deseamos que este bloque tenga un ancho de 256 Kbytes, que es la capacidad de RAM que se adicionó al sistema y que éste se encuentre localizado a partir de la dirección física 00000h. Para lo cual decidimos utilizar el bloque de memoria intermedio.

Este bloque de memoria se define programando dos registros, el que define el tamaño del bloque (**MPCS** Memory Peripherals Chips Select) como se ve en la figura 4.3 y el que define la posición de este bloque (**MMCS** Mid-Range Memory Chip Select) como podemos apreciar en la figura 4.4.



offset:A8h contenido:C0B8h

R0,R1 y R2 : Por medio de estos bits especificamos que las terminales **$\overline{PCS4}$ – $\overline{PCS6}$** funcionen sin inserción de estados de espera y atendiendo a la señal externa READY. (Ver tabla 4.2).

EX : Determinamos 7 líneas de selección **$\overline{PCS}$** (ver tabla 4.3).

MS : Periféricos mapeados en espacio E/S.(ver tabla 4.3).

Mi : Define el tamaño del bloque de selección de memoria individual por medio de las siguientes relaciones. Sólo un bit puede ser seleccionado a la vez:

Selección individual = $2^{(i+1)}$K bytes.

Selección Total = $2^{(i+3)}$K bytes.

La RAM que utilizamos tiene una capacidad de 256K bytes, por lo tanto seleccionamos el bit **M6**, lo cual nos da:

Selección individual = $2^{(6+1)}$ Kbytes = 128K bytes.

Selección total = $2^{(6+3)}$ Kbytes = 512K bytes.

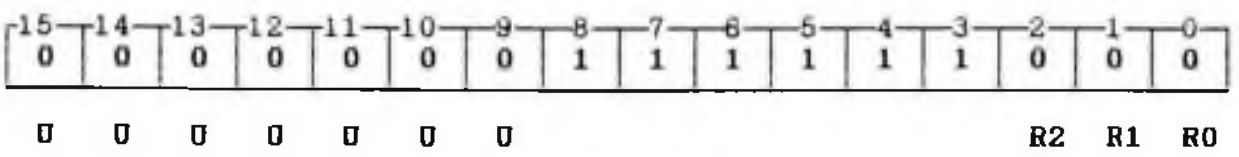
Otros : Los demás bits se encuentran alambrados a 1.

figura 4.3. Registro MPCS

Tabla 4.3. MS, EX valores de programación.

bit	Descripción
MS	1 = Periférico mapeado en espacio memoria. 0 = Periférico mapeado en espacio E/S.
EX	0 = 5 líneas $\overline{PCS}$. A1, A2 provistos. 1 = 7 líneas $\overline{PCS}$. A1, A2 no provistos.

En la programación de la figura anterior nos dimos cuenta que los bits 0 a 2 programan la inserción de estados de espera de los periféricos no de la memoria, a continuación podemos observar como se programó la dirección base del bloque en curso.



offset:A6h contenido:01F8h

R0,R1 y R2 : Por medio de estos bits especificamos que las terminales $\overline{\text{MCS0}}$ – $\overline{\text{MCS3}}$ funcionen sin inserción de estados de espera y atendiendo a la señal externa READY. (Ver tabla 2).

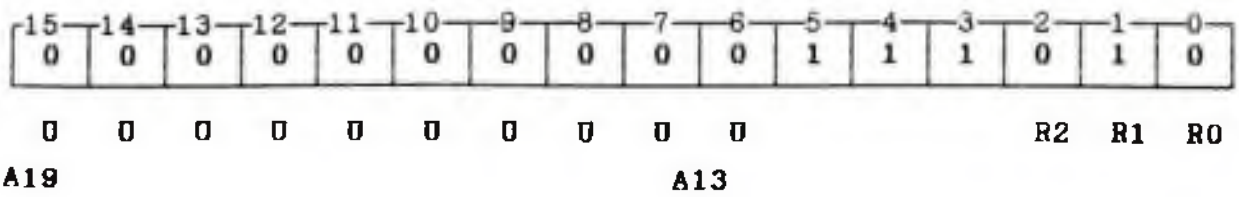
U : Estos bits determinan la dirección base del bloque de memoria intermedia y se colocan aquí sólo los 7 bits más significativos, es decir, del **A13** al **A19**.

Otros : Los demás bits se encuentran alambrados a 1.

figura 4.4. Registro MMCS

4.1.2. Definición de las posiciones de los dispositivos periféricos.

El i80188 puede generar hasta siete señales de selección de periféricos. Estas selecciones de periféricos son bloques contiguos de 128 bytes sobre una dirección base programable localizada en espacio de memoria o espacio E/S. Esto se realiza programando el registro **PACS** (PeripherAl Chip Selects), como se muestra en la figura 4.5.



offset:A4h contenido:003Ah

R0,R1 y R2 : Por medio de estos bits especificamos que las terminales **PCS0** - **PCS3** funcionen con inserción de dos estados de espera y atendiendo a la señal externa READY. (Ver tabla 2).

U : Definen la dirección base del bloque de **PCS**. Cuando éstos se encuentran mapeados en espacio memoria, los bits **A13** a **A19** de la dirección física son puestos en **U**. Cuando estos se encuentran mapeados en E/S sólo los bits **A13** a **A15** de la dirección son puestos en **U**. Debido a que nosotros seleccionamos la dirección base 0 ponemos únicamente ceros.

Otros : Los demás bits se encuentran alambrados a 1.

figura 4.5. Registro PACS.

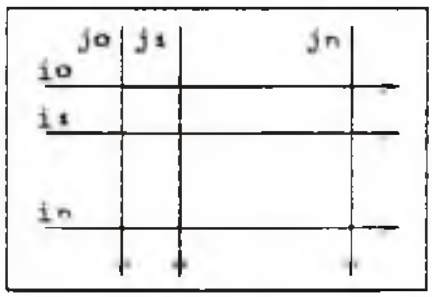
4.1.3. El refresco de memoria.

Este sistema se basa en que la memoria MT1259-12 refresca 512 celdas (un renglón de la matriz que lo forma), con el simple hecho de leer cualquier elemento del reglón n.

Entonces, para refrescar la memoria sólo basta con realizar pseudoperaciones de lectura en todos los renglones de la memoria, es decir, sólo basta con poner en la memoria la dirección del renglón y generar la señal **RAS** para que éste quede refrescado en su totalidad. Este sistema fue diseñado en dos partes, la parte realizada en hardware y la parte realizada en software.

La realizada en hardware, consistió en configurar la circuitería de la memoria RAM, de tal manera que las lecturas o escrituras se realicen columna por renglón, como se muestra en la figura 4.6.

i = 9 bits menos
    significativos
    de la dirección.
    (renglón).
j = 9 bits más
    significativos.
    de la dirección.
    (columna).



n = 512.
i = 0, 1, ..., n.
j = 0, 1, ..., n.

figura 4.6. Matriz de celdas de memoria.

La implementación de esta manera de acceso a la memoria RAM trae la ventaja de que si queremos refrescar esta memoria, sólo basta con leer 512 posiciones de memoria contiguas para dejar toda la RAM refrescada, ya que como observamos en la figura 4.6, los bits menos significativos seleccionan a los renglones de la matriz de celdas.

Cuando se efectúen lecturas sucesivas a través de todo el rango de memoria direccionable, cada celda de memoria de la RAM dinámica de la tarjeta se estará refrescando 2048 veces debido a que:

i) Son 9 bits los necesarios para direccionar los renglones de la DRAM y 2^9 son 512 renglones que tiene la DRAM.(figura 4.7).

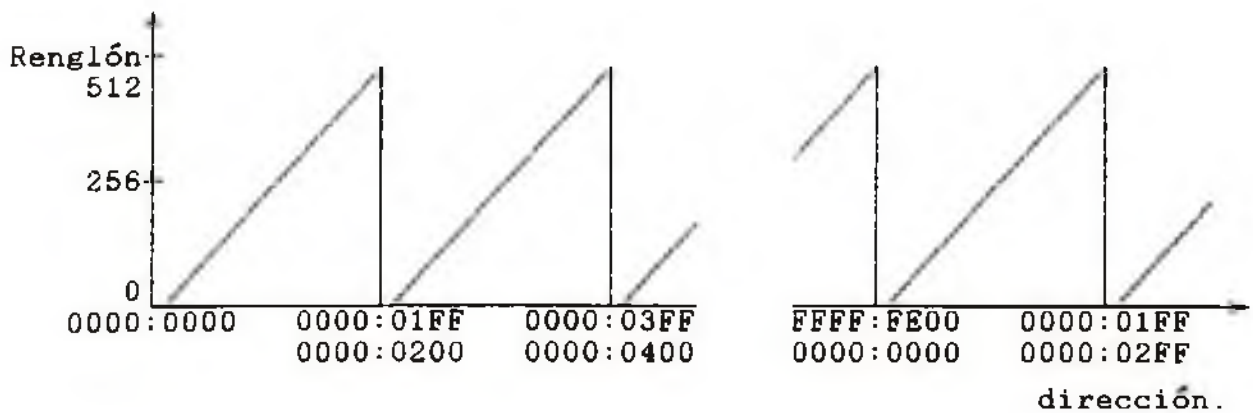


figura 4.7. Representación gráfica del refresco de memoria.

ii) El número total de posiciones de memoria que puede direccionar el i80188 es de 1048576 bytes.

iii) Entonces en número de veces que la dirección espejo hará presencia en el barrido de memoria para cada celda es 1048576/512 = 2048 veces.

La otra parte, la realizada por software, toma como base las características del hardware, el canal de DMA cero (para una mayor velocidad de lectura) y las siguientes consideraciones:

i) Son necesarios leer 512 datos contiguos sin importar sus 11 bits más significativos de dirección, para que el refresco de toda la RAM dinámica se realice:

$$X_{10}X_{18}X_{17}X_{16}X_{15}X_{14}X_{13}X_{12}X_{11}X_{10}X_{9}\ A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$$

ii) Según el manual de estos dispositivos, los renglones de memoria deben de refrescarse en intervalos de tiempo no mayores a 5 ms. Sí tenemos que refrescar 512 renglones (que es toda la DRAM) en este intervalo de tiempo, entonces el tiempo que debe transcurrir entre el refresco de un renglón y otro contiguo de memoria es:

5 ms / 512 renglones = 9.765625 μs/renglón.

Así a cada renglón le toca su refresco cada 5 ms.

Si se implementa una rutina que refresque sólo un renglón de memoria cada 10 μs, esto permitirá alternar de una manera más equilibrada la ejecución de programa y el refresco de memoria. Gráficamente, el refresco y la ejecución de un programa se puede apreciar en la figura 4.8.

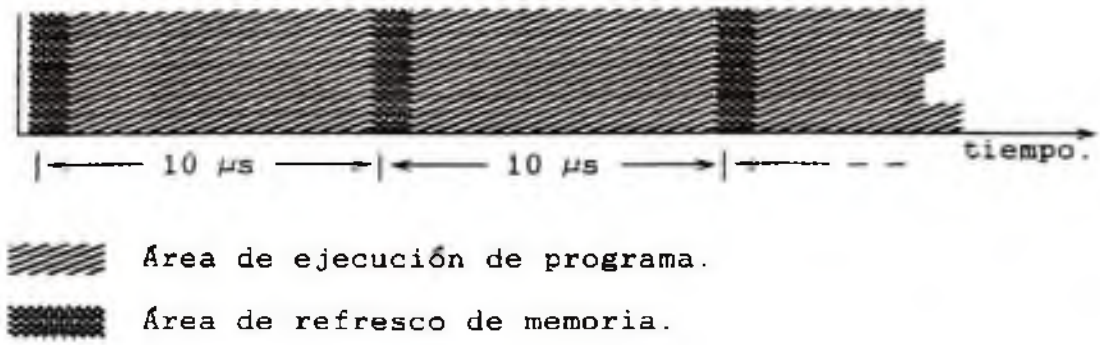


figura 4.8. Reprentación gráfica de la ejecución alternada de un programa y el refresco de memoria.

Implementación en software. Tenemos ya varios elementos para realizar el refresco de memoria, que son, el hardware, el canal cero de DMA, y algunos conceptos claros, sólo falta definir con que elemento se realizarán las mediciones de tiempo para realizar las lecturas en el momento preciso. Un periférico incluído en el procesador puede resolver nuestro problema, este es el TMR2 (TiMeR 2).

Ya definidos todos los elementos que intervendrán en el refresco de memoria, falta sólo plantear cual será la secuencia de eventos a realizar. Entonces, lo que se desea hacer es lo descrito en la figura 2, producir una rutina que cada 10 μs realice una lectura a memoria y este dato leído lo mande a un puerto inexistente. Aquí intervienen dos factores, la transferencia de datos y la temporización.

4.1.3.1. La temporización (Programación del timer).

La temporización se llevará a cabo por medio del timer 2, porque éste se encuentra internamente conectado con los canales de DMA, cosa que no sucede con los timer 0 y 1. Para que el timer responda como nosotros queremos, éste debe ser programado para cumplir con el siguiente requisito:

i) Debe producir a su salida una forma de onda periódica con un período de 10 μs.

Solución: La programación del timer se logra mandando datos a

tres registros, que son:

a) El registro de cuenta (60h).

b) El registro de máxima cuenta (62h).

c) El registro de la palabra de control (66h).

Programación del registro de cuenta. Este registro contiene la cuenta con la cual comenzará a contar el timer. Tiene 16 bits de longitud y se incrementa cada cuatro ciclos del reloj interno del procesador. El offset de este registro es de 60h. Para nuestro fin es conveniente que este registro comience en 0.

Programación del registro de máxima cuenta. Este registro debe de contener la máxima cuenta que deseamos que el registro de cuenta alcance. Cuando el registro de cuenta alcanza la máxima cuenta, este se pone a cero y produce una salida baja durante un ciclo de reloj. Entonces si nosotros deseamos que este nivel bajo aparezca cada 10 μs, tenemos que determinar cual es la cuenta máxima. Esta la determinamos de la siguiente manera:

Tenemos que la cuenta del registro de cuenta varía cada cuatro pulsos de reloj, entonces esto nos dice que si la cuenta máxima es de 1, la frecuencia que alcanzará este timer es de 2MHz, ya que nuestro reloj interno es de 8MHz. En base a esto podemos deducir la siguiente expresión:

$$\text{máxima\_cuenta} = \frac{\text{periodo\_deseadoxCLKOUT}}{4}$$

Si período_deseado = 10 μs y CLKOUT = 8MHz entonces:

$$\text{máxima\_cuenta} = \frac{(10\ \mu s)(8MHz)}{4} = 20 = 14h.$$

Por lo tanto el offset 62h que es el registro de máxima cuenta debe de ser enviado el valor 14h.

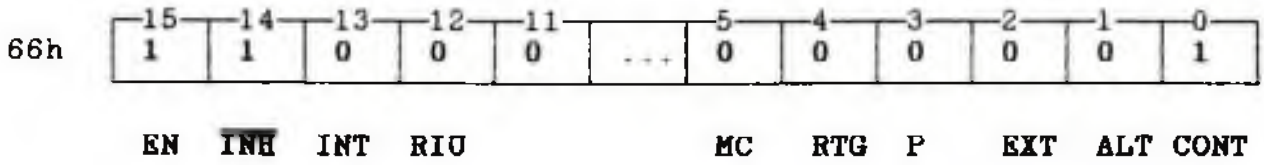
Programación del registro de la palabra de control. En este registro se define el modo de operación del timer. El modo de operación que nosotros queremos que tenga el timer debe tener las siguientes características:

i) Que el timer al llegar a su máxima cuenta no se detenga (bit

0)

ii) Cuando alcance la máxima cuenta no produzca interrupción (bit 13)

Entonces la palabra de control queda de la siguiente forma mostrada en la figura 4.9.



EN : Puesto a 1 indica que se habilite el timer 2.

$\overline{\text{INH}}$: Es un bit de enmascaramiento y puesto a 1 podremos modificar el bit EN.

INT : Puesto a cero indicamos que no se active interrupción al alcanzar el timer su cuenta máxima.

MC : Indica cuando se alcanza la máxima cuenta y debe ser puesto a cero.

CONT : Después de alcanzar la máxima cuenta continuar.

OTROS : Los demás bits están alambrados a cero.

figura 4.9. Palabra de control para el timer 2.

Al offset 66h que es la posición del registro de la palabra de control del timer 2 se manda el dato C001h . De esta manera queda programado el timer 2.

4.1.3.2. La transferencia de datos (programación del controlador de DMA).

La transferencia de datos. Esta se pretende hacer con la ayuda de uno de los dos canales de DMA que posee el i80188. Los objetivos son:

i) Leer los datos desde la dirección 00000h hasta la dirección FFFFFh.

ii) Los datos leídos mandarlos a un puerto inexistente.

iii) Que cada ciclo de DMA mueva sólo un dato.

iv) Cada ciclo de DMA será activado por una señal desde el timer 2, para así obtener la periodicidad deseada.

v) Que estas acciones se repitan indefinidamente.

Solución: Los objetivos planteados pueden llegar a ser alcanzados sí mandamos los datos correctos a seis registros que posee el canal de DMA cero y que son:

a) El registro del apuntador fuente (C0h).
b) El registro del apuntador fuente (4 bits superiores) (C2h).
c) El registro del apuntador destino (C4h).
d) El registro del apuntador destino (4 bits superiores) (C6h).
e) El registro de cuenta de transferencia (C8h).
f) El registro de la palabra de control (CAh).

Programación de los registros del apuntador fuente. En estos registros debe estar contenida la dirección física de la cual se va a comenzar a mover datos. La dirección inicial que a nosotros nos interesa es la 00000h. Esta dirección debe ser mandada a los registros de la siguiente manera. Los 16 bits menos significativos deben ir en el registro con posición C0h.y los 4 bits más significativos deben ser enviados al registro con posición C2h.

Programación del registro del apuntador destino. En estos registros debe estar contenida la dirección a la cual van a llegar los datos sacados de la posición de memoria apuntada por los registros explicados anteriormente. La dirección que nos interesa es la 0400h que es la dirección de un puerto inexistente. Entonces, esta dirección tiene que ser enviada a los registros de igual manera que en los registros anteriormente explicados. Los 16 bits menos significativos deben ir en el registro con posición C4h. Y los 4 bits más significativos deben ser enviados al registro con posición C6.

Programación del registro de cuenta de transferencia. Este registro determina el número de transferencias que se deben realizar en cada ciclo de DMA. En él se almacena el número de transferencias deseadas y durante el ciclo de DMA este número se decrementa hasta llegar a cero, para marcar así el fin del ciclo.

A nosotros nos interesa efectuar sólo una transferencia cada ciclo de DMA y lo que determinará esto es el timer 2 en combinación con la palabra de control del canal cero de DMA. Por lo tanto a nosotros nos da lo mismo poner un número que otro y lo que decidimos fue dejar el número cero, ya que este timer contará, llegará a su máxima cuenta, se pondrá a cero y continuará. Este es enviado a la posición C8h.

Programación de la palabra de control. Por medio de esta palabra se determina el funcionamiento del canal cero de DMA y los registros antes programados son sólo datos que el canal utilizará. De este canal nos interesa:

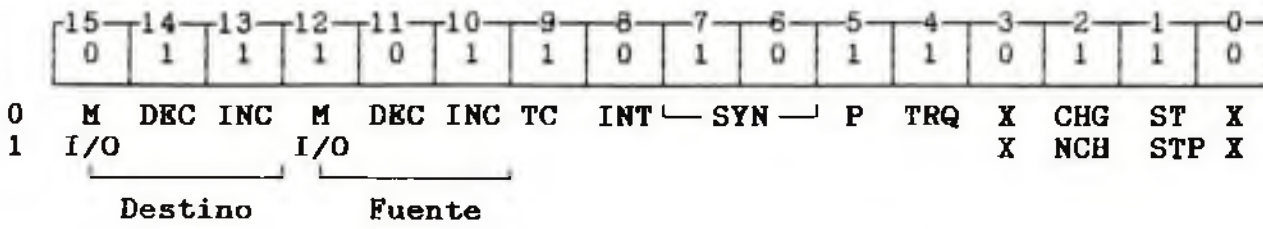
i) Que el apuntador fuente recorra todas las posiciones de la memoria direccionable (bit 12) en forma ascendente (bit 11 y 10).

ii) Que el apuntador destino permanezca estático (bit 14 y 13) apuntando hacia un puerto (bit 15).

iii) Que para cada ciclo de DMA la cuenta de transferencia se decremente, sin embargo que no termine el ciclo (bit 9).

iv) Que la habilitación de cada ciclo esté determinada por el Timer 2 (bit 4), pero sólo un ciclo de DMA (bits 6 y 7) por cada habilitación.

En base a esto la palabra de control del DMA queda de la manera mostrada en la figura 4.10.



Destino : El apuntador destino debe de apuntar hacia un espacio de E/S y que permanezca sin cambio después de cada transferencia.

Fuente : El apuntador fuente debe apuntar hacia un espacio de memoria y que se incremente a cada transferencia.

TC : Con este bit puesto a 1 indicamos que cuando el registro de transferencia de cuenta llegue a cero, el funcionamiento del canal de DMA no se detenga.

INT : Este bit lo ponemos a 1 para indicar que no se realice ninguna interrupción al fin del ciclo de DMA.

SYN : Con estos bits estamos indicando que los datos son enviados al destino de una manera sincronizada. Con esto logramos que el timer 2 sincronice el funcionamiento del canal.

P : Este bit es puesto a 1 para indicar que este canal tiene la máxima prioridad con respecto al otro canal.

TDRQ : Puesto a 1 este bit indica que el canal de DMA será habilitado por medio del timer 2.

CHG/$\overline{\text{NOCHG}}$: Este bit es un bit de enmascaramiento y es puesto a 1 para de esta manera poder modificar el bit **ST/$\overline{\text{STOP}}$**.

ST/$\overline{\text{STOP}}$: Este bit es el encargado de determinar si el ciclo de DMA continúa o se detiene.

OTROS : Los bits 0 y 3 son bits que no importan.

figura 4.10. Palabra de control del controlador de DMA.

Por lo tanto la palabra que debe mandarse a la posición CAh, es la 74B6h. Y de esta manera queda programado el refresco de memoria.

4.1.4. Inicialización de los segmentos.

La colocación de los segmentos se realiza dentro de la RAM De los cuatro resgistros de segmento que posee el i80188, el que más nos interesa es el segmento de pila, ya que este registro, en conjunto con el registro apuntador de pila, son necesarios para realizar llamadas a subrutinas.

De los demás registros de segmento, no nos preocupamos, ya que todos son inicializados debidamente, durante la ejecución de un programa y estos son restablecidos al final de la ejecución.

El registro de segmento de pila es cargado con el valor 3000h y el registro apuntador de pila es cargado con el valor 0800h. Con esta inicialización, podemos construir rutinas y hacer llamados a estas, logrando así, en ensamblador, una programación estructurada.

4.1.5. Definición de las rutinas primitivas de monitoreo.

Estas rutinas fueron realizadas en ensamblador, residen en la memoria ROM y están pensadas para efectuar una fácil comunicación entre el COPROCESADOR y la IBM PC/XT/AT, a través de la circuitería DIALOGO. Estas rutinas son tres y son:

4.1.5.1. Rutina LOAD.

Sirve para cargar un bloque de datos de longitud n a partir de la posición segmento:osffset, en la memoria RAM del coprocesador. La secuencia de operación de esta rutina se inicia recibiendo la longitud del boque a cargar, la longitud máxima que puede cargar esta rutina es de 64 Kbytes, la cual se pone en el registro contador CX. Las siguientes instrucciones a recibir son la posición del bloque, que consiste en la recepción del segmento, el cual es puesto en el registro ES y la recepción del offset, que es puesto en el registro SI. Teniendo los datos necesarios para la carga del bloque, se procede a recibir datos en el registro AX, provenientes de la circuitería "diálogo" y mandados a la posición de memoria deseada por medio de la instrucción:

```
MOV ES:[SI],AL
```

Esta instrucción se ejecuta el número de veces marcado por el

contenido del registro CX, ya que este registro controla a un ciclo LOOP. Para cada iteración el registro SI se incrementa en uno.

4.1.5.2. Rutina READ.

Es la rutina encargada de entregarnos un bloque de datos de longitud n residentes en la posición segmento:offset. La operación de esta rutina comienza con la recepción de la longitud del bloque a entregar, esta longitud tiene un máximo de 64 Kbytes y el número que representa a ésta se almacena en el registro CX. La siguiente acción es recibir la posición del bloque, la cual se recibe en dos partes, primero el offset y después el segmento. El offset es cargado en el registro SI y el segmento en el registro ES. Y los datos direccionados por estos registos son almacenados en el registro acumulador AX y mandados a la IBM PC/XT/AT a través de la circuitería "diálogo", por medio de la siguiente instrucción:

```
MOV AL,ES:[SI]
```

El número de transferencias está limitado por un ciclo LOOP en conjunto con el registro CX. Para cada iteración el registro SI se incrementa en uno.

4.1.5.3. Rutina RUN.

Esta rutina se encarga de ejecutar un programa residente en memoria. Su operación necesita sólo de la posición del programa a correr, la cual se recibe en dos partes, primero el offset y luego el segmento. El offset lo almacena en el registro AX y de ahí es introducido a la pila, lo mismo se hace con el segmento. Ya residente en la pila la posición de la rutina a correr, se hace un llamado largo por medio de la instrucción:

```
CALL DWORD PTR [BP]
```

Logrando así la ejecución del programa. Al fin de éste, se restablece la pila.

4.1.5.4. El menú de comandos.

Ahora, la manera por la cual se tiene acceso a estas rutinas, es por medio de un menú, el cual está formado por un ciclo infinito que escudriña el registro de estado de la circuitería

"diálogo" en espera de un comando. Los comandos son números clave, en donde cada rutina tiene asignado un número. Así tenemos que la rutina LOAD tiene asignado el número 1, la rutina RUN tiene asignado el número 2 y la rutina READ tiene asignado el número 3. Entonces, la secuencia de instrucciones que debemos de enviar al coprocesador para que nos envíe un bloque de 00FFh datos que inicia en la dirección FECC:1A33h es, primero mandar el comando, es decir, un 3, que significa "leer" y el procesador salta entonces a la rutina READ para esperar por las demás instrucciones, las cuales consisten en, primero mandar la longitud del bloque. Como podemos observar, ésta es una palabra y a través de la circuitería "diálogo" sólo pueden pasar bytes, entonces es obvio que tenemos que partir éstos en dos partes, en menos significativa y en parte más significativa. Por lo tanto para mandar el tamaño del bloque tenemos que mandar primero un FFh y luego un 00h. Para el caso de la dirección sería los mismo, es decir, para el offset se envía primero 33h y luego 1Ah y para el segmento primero se envía CCh y luego FEh. De esta manera sólo resta leer los 00FFh datos.

4.2. Las herramientas de trabajo.

En la sección anterior vimos como se implementaron algunas rutinas primitivas de comunicación y cual es su funcionamiento. En esta sección aprovecharemos éstas, para crear un programa que resida en la memoria de la IBM PC/XT/AT y que sea amigable. Para esto desarrollamos un **cargador-monitor** del tipo absoluto. Las funciones que tiene éste son enumeradas a continuación:

a) Es capaz de relocalizar en la memoria del coprocesador, programas ejecutables residentes en disco.
b) Puede cargar bloques de datos residentes en disco.
c) Es capaz de ejecutar programas residentes en la memoria del coprocesador
d) Puede mostrarnos el contenido de la memoria del coprocesador.
e) Salta a mostrarnos cualquier área de memoria indicándole la posición que queremos observar

Estas funciones fueron implementadas en lenguaje C. Y su funcionamiento detallado se describe enseguida.

a) El relocalizador es una rutina que sólo abre archivos .EXE residentes en disco, los carga en la memoria de la IBM PC/XT/AT, efectúa las modificaciones necesarias para que corran en el segmento indicado por el usuario y de aquí son cargados en la memoria del coprocesador. Solamente puede relocalizar en direcciones múltiplos de 10h. En la figura 4.11 podemos ver como se ve el cargador en la pantalla de la IBM PC/XT/AT, en el momento en el que podemos decir si un archivo es relocalizado o cargado.

b) El cargador es una rutina que abre cualquier tipo de archivo residente en disco, los carga en la memoria de la IBM PC/XT/AT y de aquí son cargados en la posición de memoria del coprocesador indicada por el usuario. En la figura 4.11 podemos ver como se ve el cargador en la pantalla de la IBM PC/XT/AT, en el momento en el que podemos decir si un archivo es cargado o relocalizado.

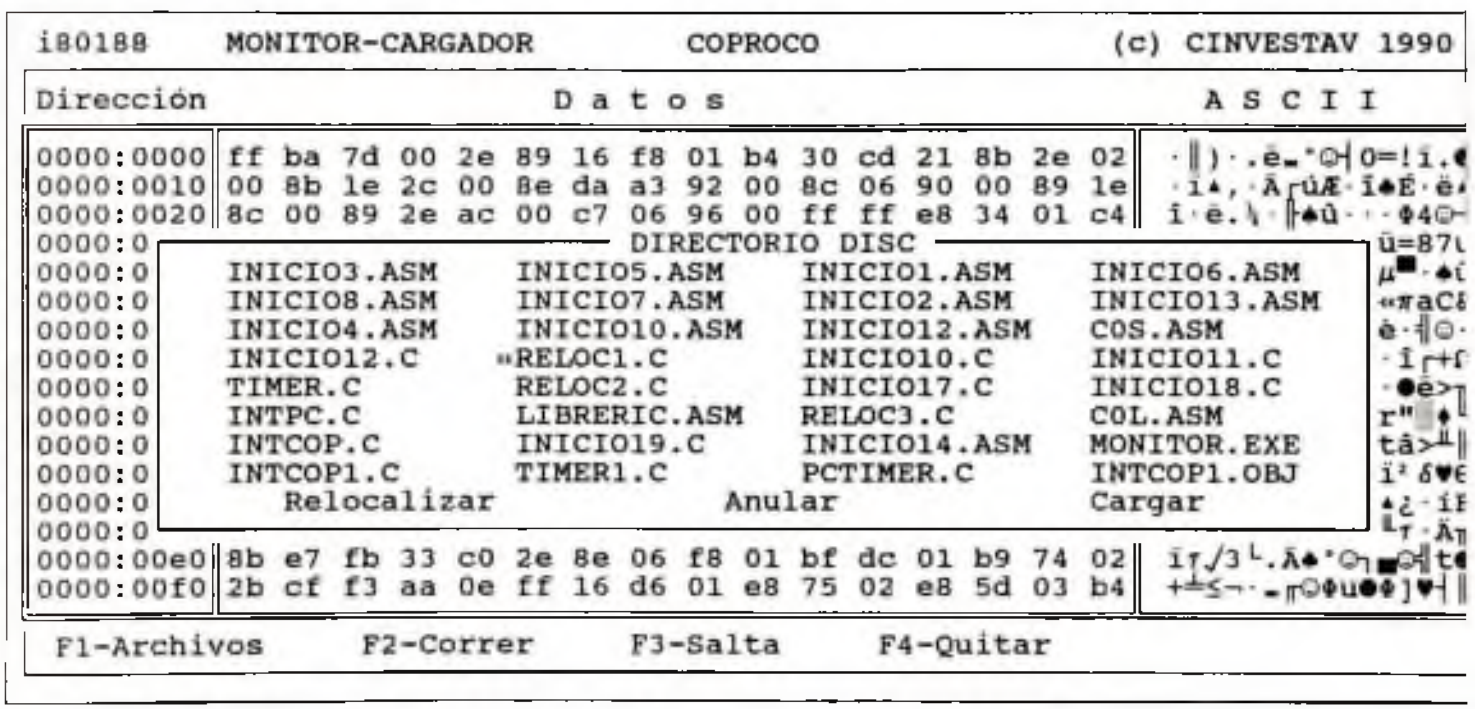


figura 4.11. El cargador-Relocalizador de archivos.

c) La función **corre** hace uso de la rutina primitiva RUN residente en la ROM del coprocesador y se ejecuta con sólo indicar la dirección de inicio del programa a correr.

d) Esta capacidad del monitor-cargador nos permite observar el contenido de cualquier área de memoria del coprocesador. Tiene la ventaja de mostrarnos en hexadecimal y en ASCII. Además de que podemos "navegar" por la memoria del coprocesador por medio de las flechas de control del cursor y por las teclas de salto de página como si se tratara de un editor de texto. En la figura 4.12 podemos observar como se ve el monitor en la pantalla de la IBM PC/XT/AT.

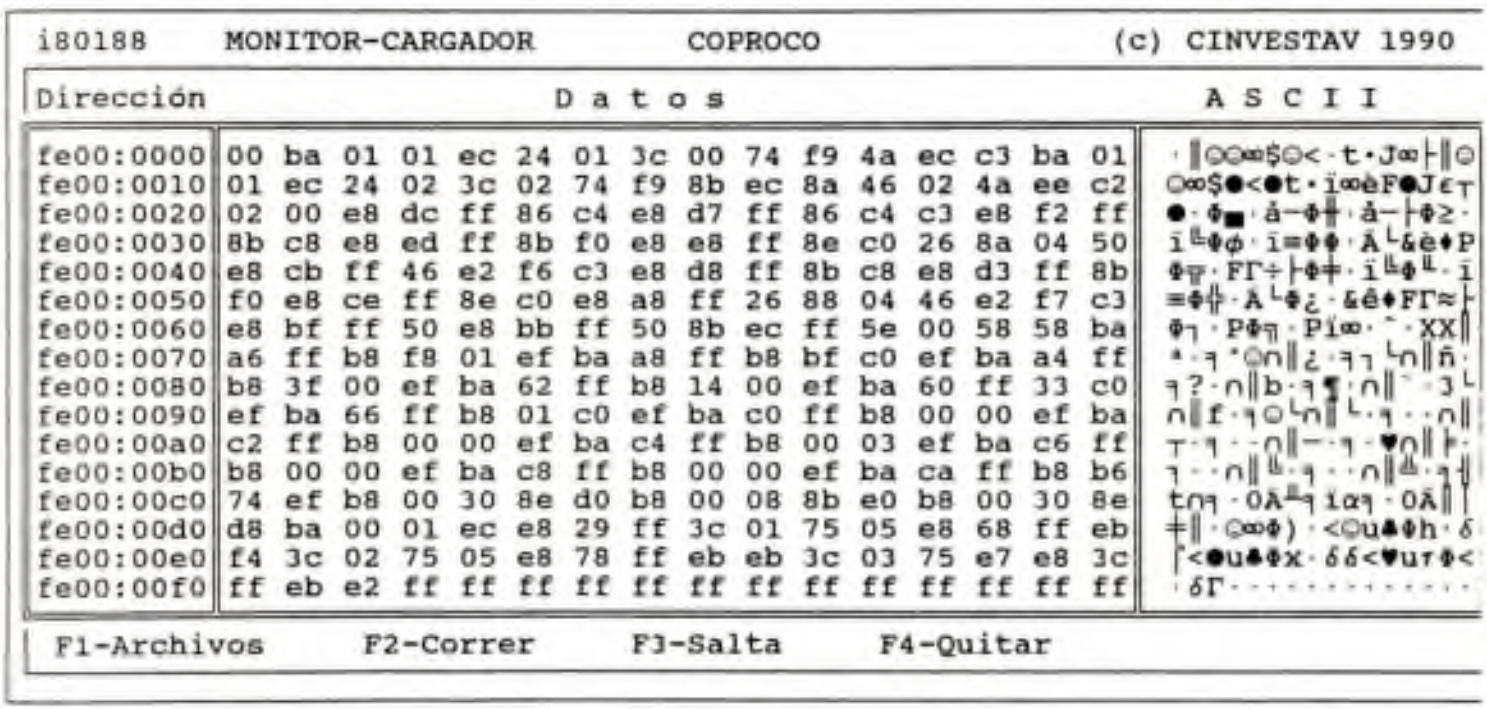


figura 4.12. El monitor del i80188.

e) Para observar el contenido de direcciones alejadas, este monitor tiene un comando de salto, el cual sólo requiere la nueva posición de memoria a observar.

4.2.1. Sencilla modificación a los archivo C0S.OBJ y C0L.OBJ de turbo C.

Esta modificación se hizo con el objeto de que los programas hechos en el compilador de turbo C puedan correr en la tarjeta coprocesadora.

Cuando un programa es compilado en turbo C, se produce un programa .OBJ el cual es enlazado con las librerías de turbo C y con un archivo C0*.OBJ, dependiendo del modelo en el que se esté trabajando. Este archivo C0*.OBJ contiene rutinas que son útiles en un ambiente de MS-DOS. Por lo tanto tenemos que efectuar algunas modificaciones para eliminar estas rutinas que hacen llamadas al sistema. La modificación consistió en quitar todas las rutinas que hacen llamadas al sistema y dejar sólo la inicialización de los segmentos, la llamada a _MAIN y el restablecimiento de los segmentos a su estado anterior. Esta modificación sólo se hizo para los modelos SMALL y LARGE.

CONCLUSIONES.

Los objetivos planteados para la realización de la tarjeta fueron cubiertos en su totalidad y de una manera satisfactoria.

Durante el diseño/construcción de la tarjeta nos topamos con múltiples problemas (es obvio), pero de los cuales vale la pena mencionar dos. El problema de la temporización del periférico Z8530, el cual se solucionó rápidamente gracias a la experiencia de mi asesor y el otro, el del acceso al bus de la IBM PC/XT/AT, el cual fue insalvable, ya que definitivamente esta máquina no lo permite. Una alternativa para tener el mismo efecto (transferencia de datos por DMA), fue la de aumentar el software y reduciendo el hardware, cosa que es más complicada que la primera, pero se pudo lograr el objetivo.

Por otra parte, un diseño de este proyecto es un proyecto parcial, ya que este servirá como base para la realización de algo mucho mayor, que requiere la participación de varios tesistas y la cooperación incondicional de todas las partes involucradas en este proyecto. Esto último es un factor que se necesita con urgencia en la sección.

Como sugerencias y aclaraciones sólo puedo mencionar que la parte del software de inicialización, y el monitor del coprocesador se pueden mejorar mucho. Ya que, varias de las rutinas desarrolladas no se optimizaron exhaustivamente. El objetivo principal era que funcionarán y no importaba más. Por parte del hardware, tal vez se pueda reducir la circuitería, con el uso de PALs y con memorias de alta densidad, además de que se ocultarían muchas de las partes funcionales del sistema.

Para concluir sólo puedo mencionar que la realización de esta tesis me sirvió para reafirmar los conocimentos del diseño de sistemas digitales y los temas relacionados con programación de bajo nivel, como es, la programación de sistemas.

BIBLIOGRAFÍA.

[1] Advanced Micro Device. **Advanced micro device ISDN**, Advanced micro device, U.S.A., 1989.

[2] Borland. **Turbo C User´s Guide**. Borland, U.S.A., 1988.

[3] Choisser, John, **The XT-AT handbook for Ingeniers, Programmers and other serious PC/XT and PC/AT users**, U.S.A. 1988.

[4] Davies,Barber, Price, **Computer networks and their protocols**, Wiley & Sons, Londres, 1979.

[5] Donovan, John, **System programming**, McGraw-Hill International, Tokio, 1972.

[6] Duncan, Ray, **Advanced MS-DOS programming**, Microsoft Press, U.S.A., 1988.

[7] Hall,Douglas, **Microprocessors and Interfacing Programming and Hardware**, McGraw-Hill, Singapore, 1988.

[8] IBM, **Technical Reference IBM PC**, International Business Machines Corporation, 1983.

[9] Intel, **Family user´s manual**, Intel corporation, U.S.A., 1980.

[10] Intel, **Microprocesors and Peripherals Volume I**, intel corporation, U.S.A, 1985.

[11] Intel, **Microsystem Componenetes handbook**, intel corporation, U.S.A., 1985.

[12] Intel, **Microcomunication handbook**, intel corporation, U.S.A.,1989.

[13] Mitel, **Miroelectronic data book**, Mitel Corp., Canada, 1989.

[14] Taub & Schilling, **Principles of communication Systems**, McGraw-Hill, U.S.A., 1989.

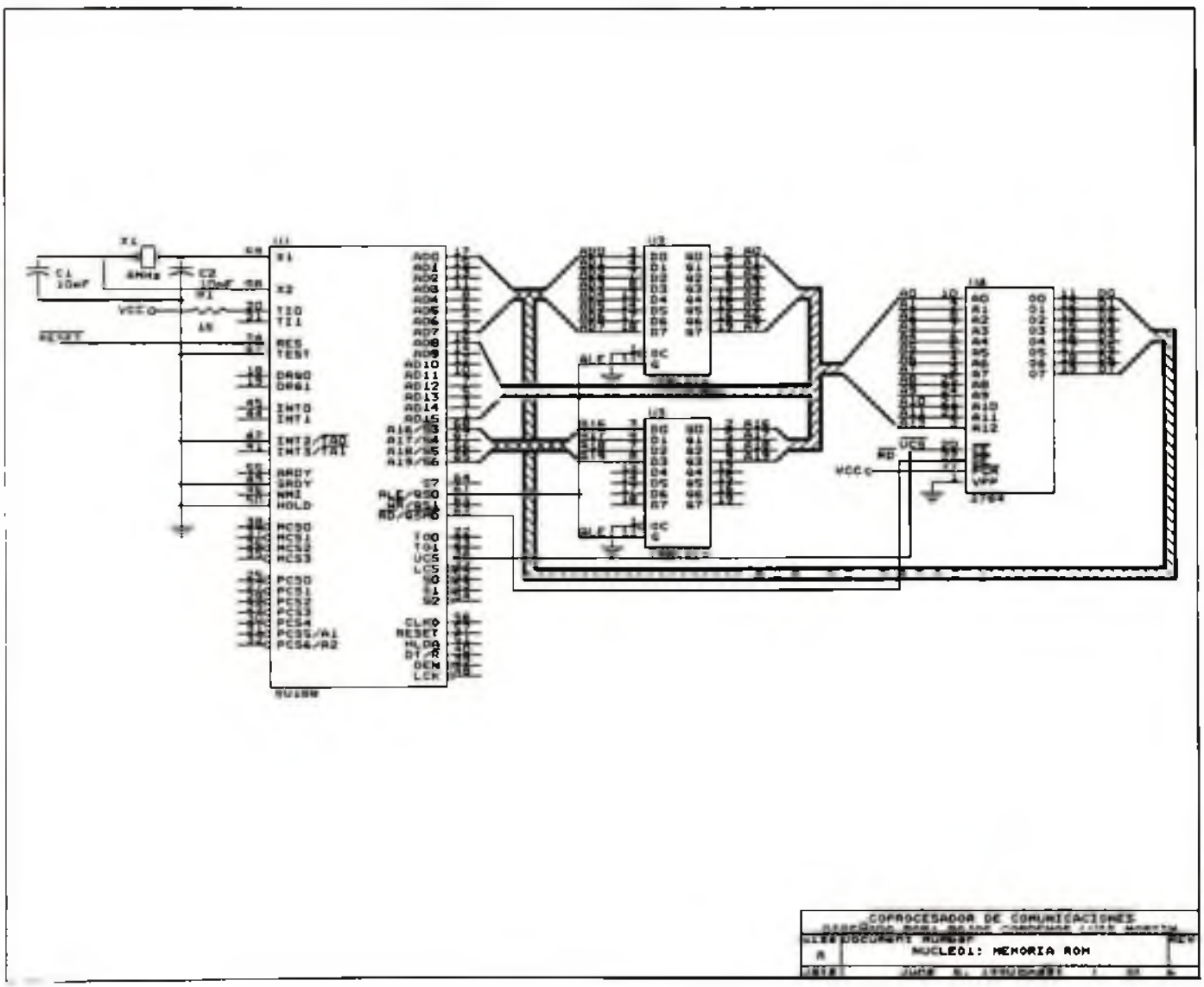
[15] Seyer, Martin, **RS-232 made easy**, Prentice Hall inc., U.S.A.,1984.

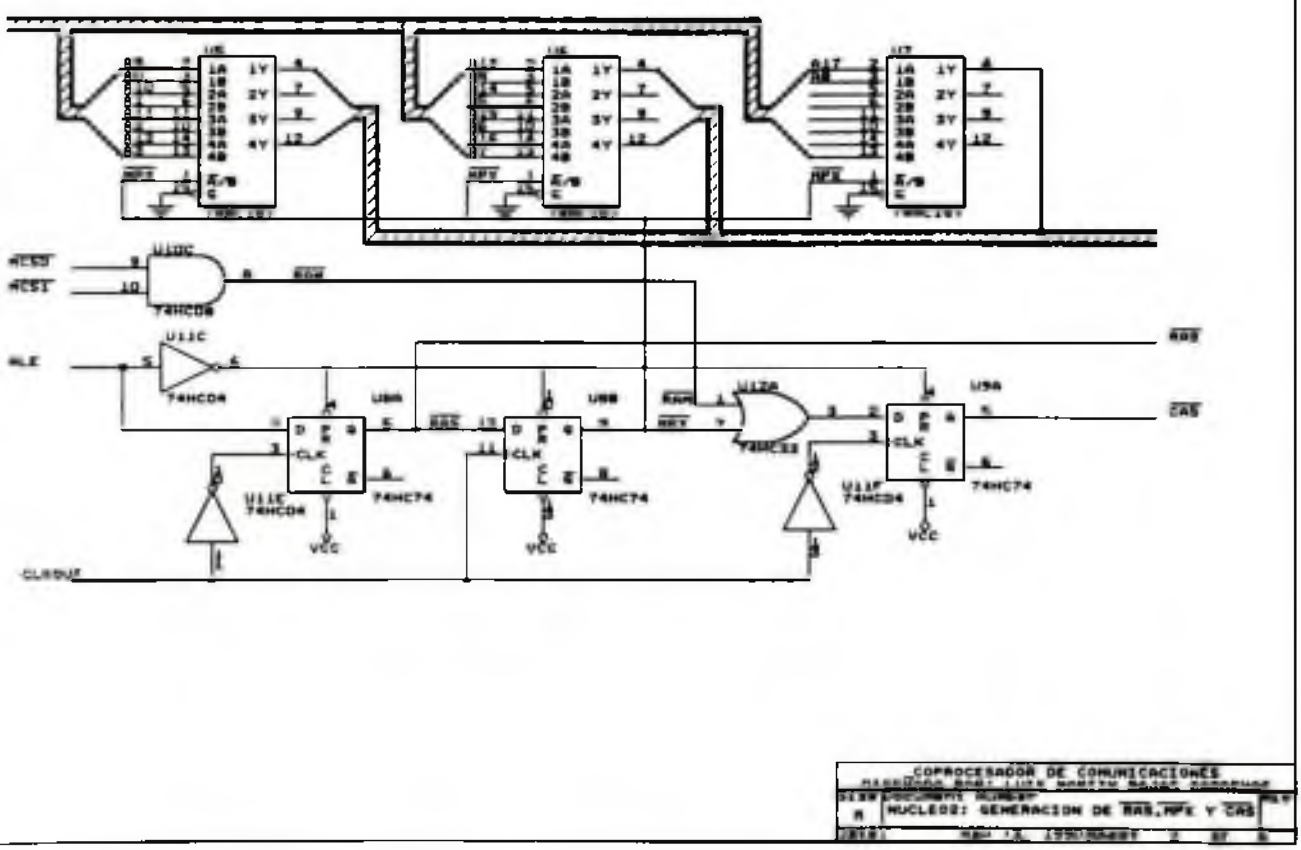
[16] Stalings, William, **Data and computer comunications**, McMillan, U.S.A, 1989.

[17] Tanenbaum, Andrew, **Computer networks**, Prentice Hall, U.S.A., 1981.

[18] Zilog, **Components data book**, Zilog Inc., U.S.A., 1984.

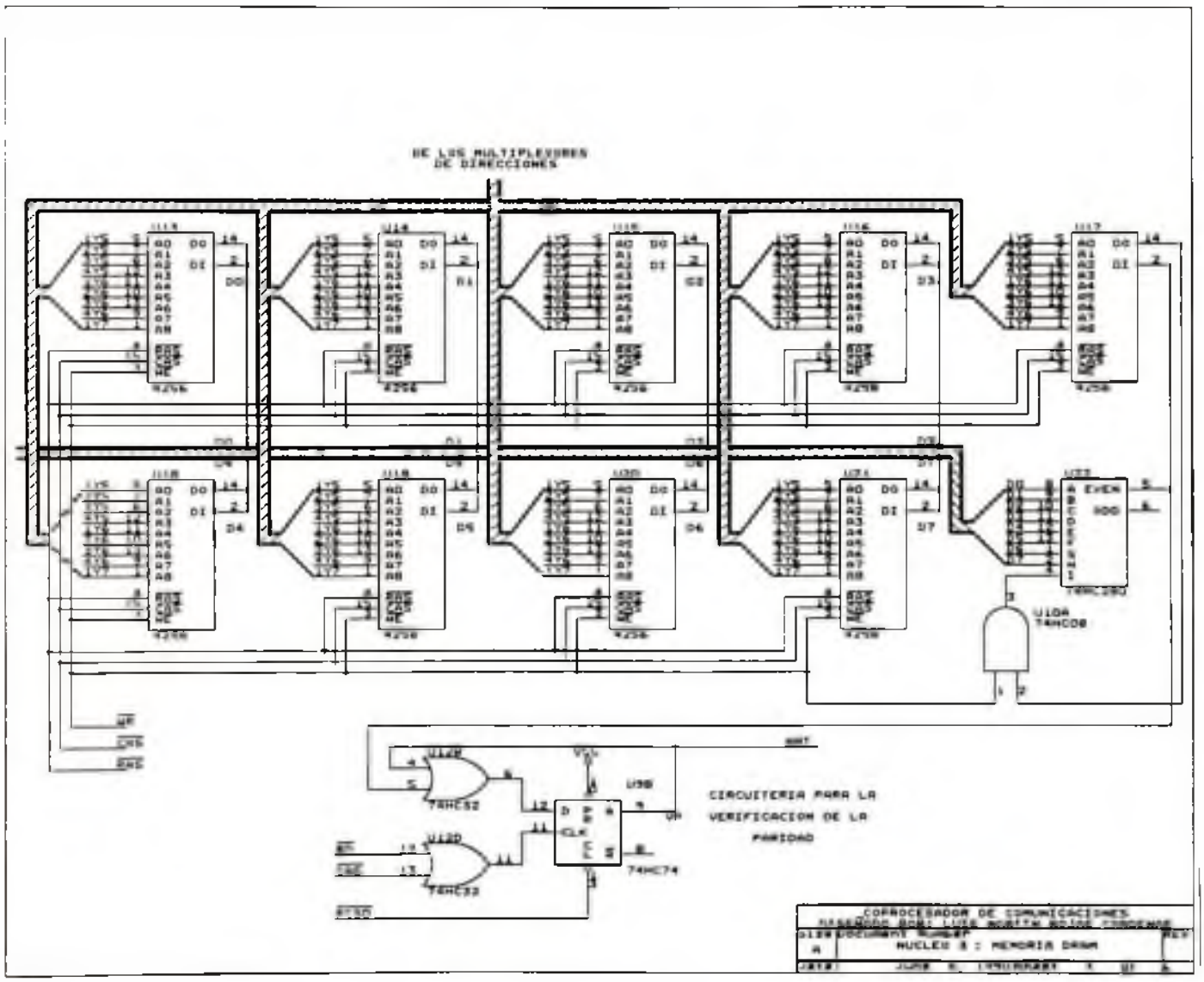
APENDICE I

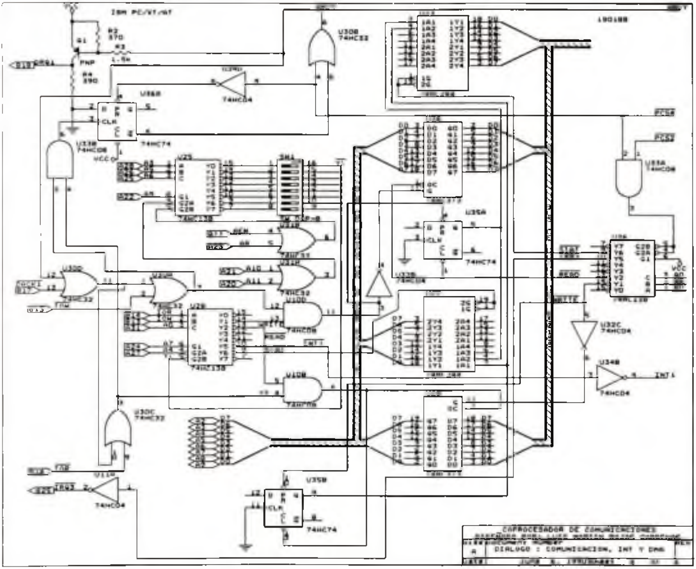


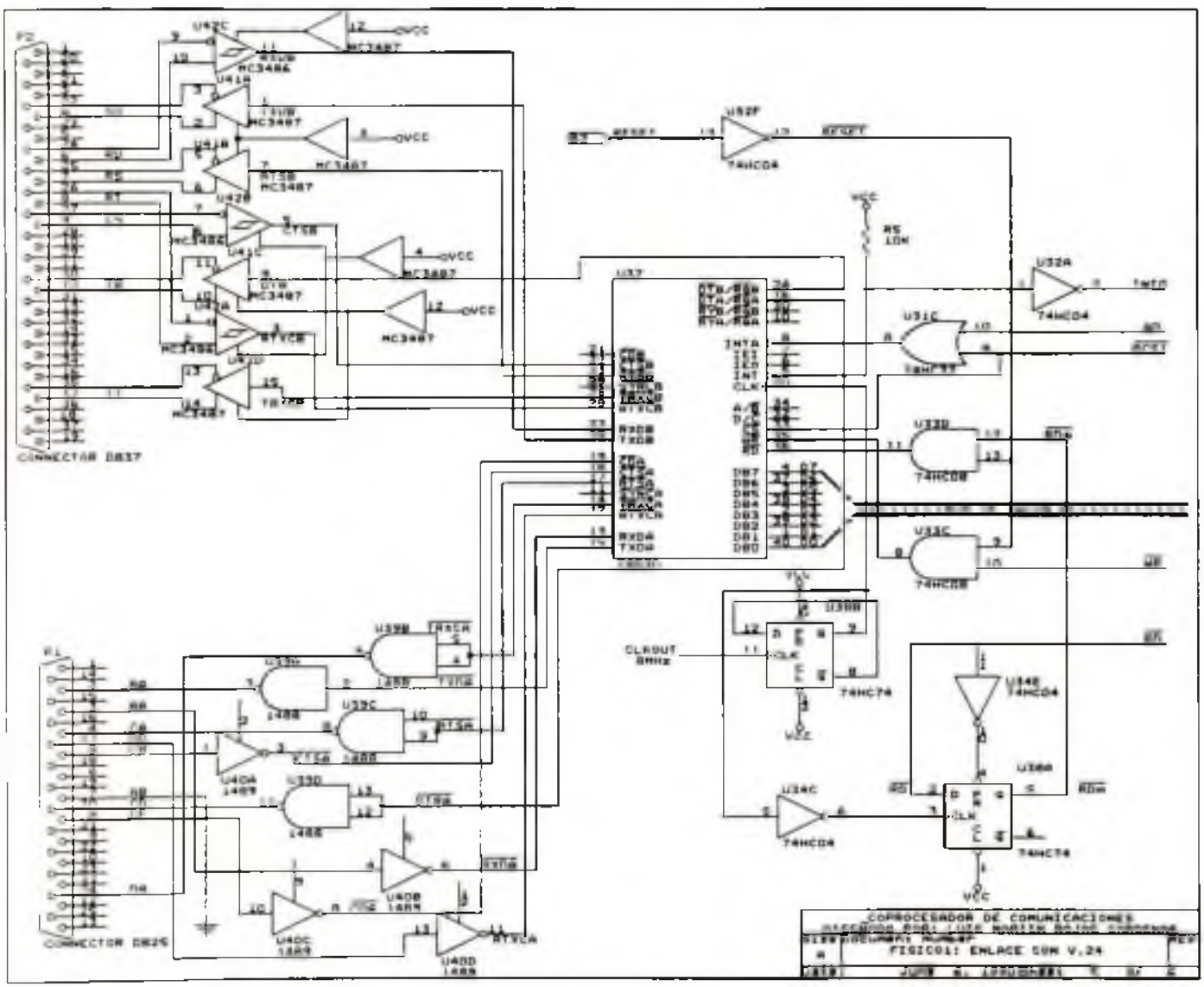


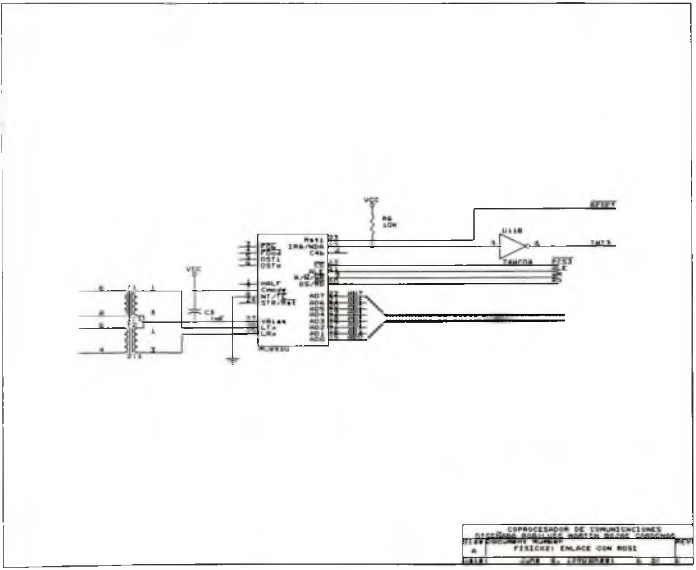
COPROCESADOR DE COMUNICACIONES

NUCLEO2: GENERACION DE RAS, MPX Y CAS

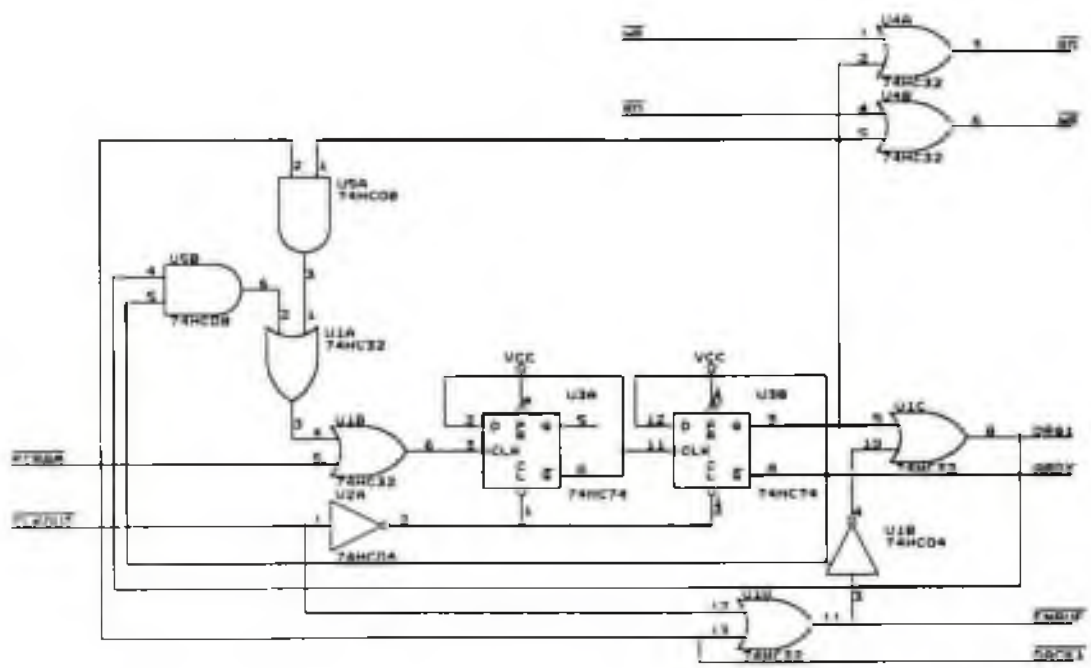

DE LOS MULTIPLEXORES
DE DIRECCIONES
U10A
74HC08
CIRCUITERIA PARA LA
VERIFICACION DE LA
PARIDAD
74HC74
NUCLEO 8 : MEMORIA DRAM


IBM PC/XT/AT
U30B
74HC32
74HC04
74HC74
74HC08
74HC138
74HC32
COPROCESADOR DE COMUNICACIONES
DIALOGO : COMUNICACION, INT Y DMA


CONECTOR DB37
CONECTOR DB25
74HC04
74HC08
74HC74
MC3487
MC3486
1488
1489
VCC
RS
LDN
CLKOUT
8MHz
COPROCESADOR DE COMUNICACIONES
FISICO1: ENLACE CON V.24


VCC
R6
10K
U11B
74HC04
C3
COPROCESADOR DE COMUNICACIONES
A

A P E N D I C E I I


U4A
74HC32
U4B
74HC32
U5A
74HC08
U5B
74HC08
U1A
74HC32
U1B
74HC32
U2A
74HC04
U3A
74HC74
U3B
74HC74
U1C
74HC32
U1B
74HC04
U1D
74HC32
VCC
VCC

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL I.P.N.
APARTADO POSTAL 14-740
MEXICO 07000, D.F.

EL JURADO DESIGNADO POR LA SECCION DE COMPUTACION DEL DEPARTAMENTO DE INGENIERIA ELECTRICA, ACEPTO EL DIA 19 DEL MES DE JUNIO DEL AÑO 1990 EL TRABAJO DE TESIS " DISEÑO Y CONSTRUCCION DE UN COPROCESADOR PARA COMUNICACIONES "

DESARROLLADO POR EL ALUMNO: LUIS MARTIN ROJAS CARDENAS ; PARA SU IMPRESION Y TRAMITES CORRESPONDIENTES DEL EXAMEN DE GRADO.

DR. JAN JANECEK HYAN

M. EN C. CARLOS E. HIRSCH GANIEVICH

M. EN C. ANDRES VEGA GARCIA

'lda.