12
24

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y CONSTRUCCION DE UN MULTIPLEXOR DE IMPRESION PARA COMPUTADORAS PERSONALES

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA
AREA: SISTEMAS DIGITALES

P R E S E N T A N

GABRIEL ARELLANO GUTIERREZ
GUILLERMO GARDUÑO AMBRIS
JORGE MORALES CHONG

Y QUE PARA OBTENER EL TITULO DE

DIRECTOR: ING. EDUARDO RAMIREZ

MEXICO, D.F. 1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

En la década de los años 70's, el lanzamiento al mercado de los equipos personales de cómputo, abrió un nuevo y amplio campo a lo que hasta entonces había sido el diseño, construcción y operación de computadoras.

En aquella época los equipos de cómputo representaban grandes máquinas que solían requerir de cuidados especiales tales como un clima artificial que les conservara a la temperatura adecuada, grandes salones en donde se instalaran, etcétera.

Las microcomputadoras, como se les llamó a los equipos recién lanzados, no necesitaron de cuidados tan delicados, además se convirtieron en máquinas portátiles que podían ser desplazadas de un lugar a otro con la mayor facilidad. Podían ser instaladas en cualquier oficina y ¡ hasta en casas particulares !

Dado que los nuevos productos resultaron un éxito, las microcomputadoras personales o mejor conocidas como PC's (personal computers), comenzaron a introducirse rápidamente en casi todas las actividades. Así las oficinas vieron llegar los nuevos equipos, en los hospitales la gente comenzó a trabajar con PC's, las fábricas e industrias también participaron en la utilización de PC's para el monitoreo y control de sus procesos, las escuelas y universidades compraron equipo PC para introducir a los alumnos al nuevo mundo de cómputo que se estaba generando.

A medida que las PC's abarcaron más centros de trabajo, también comenzaron a surgir nuevas aplicaciones para ellas. El nuevo mercado solicitó de diseñadores y fabricantes que le dieran un valor agregado con nuevos productos. Fue entonces que una gran parte de los diseñadores tanto de hardware como los de software hemos enfocado nuestro trabajo a generar productos que hagan más y más poderosas a las

microcomputadoras.

La presente tesis está orientada a solucionar un cuello de botella en centros de cómputo donde se utilizan PC's. Tales centros pueden estar localizados en lugares como oficinas, escuelas y universidades, centros de desarrollo y otros lugares en donde las PC's envíen información a ser impresa en cualesquiera dispositivos impresores (impresoras y graficadores).

En dichos centros, es frecuente que la mayoría del equipo lo constituyan, lógicamente, las PC's y la minoría los dispositivos impresores. Algunas de las razones por las cuales no existe un número igual de PC's que de dispositivos impresores son:

1 Que los envíos a imprimir no son tan constantes como el trabajo que se hace en las PC's.

2 Por lo anterior resulta que el costo de una impresora o graficador es muy grande, si se le compara con el tiempo en que lo ocupa una PC.

Dicho lo anterior, los usuarios de PC's tienen varios caminos para solucionar el problema de impresión :

La primera opción es esclavizar una computadora para ser "atada" al impresor, así cada vez que alguien desee enviar un trabajo de impresión, tendrá que trasladar su archivo hacia aquella computadora y entonces enviar su documento. Lo anterior se complica adicionalmente si no es uno el que desea imprimir en un momento dado, sino varios. Entonces se tendrá que hacer una fila para poder obtener un turno y así enviar el documento a imprimir, esto con las complicaciones intrínsecas de las filas.

Una segunda forma es no trasladar los archivos, sino la impresora. Lo anterior implica llevar la impresora de un lugar a otro, así, conectar y desconectar cables. Esto lleva consigo una tarea más dilatada y con el lógico deterioro de cables, impresora y eficiencia.

La tercera opción es enlazar a todas las máquinas PC's dentro de una "red". Esta ya permite a los usuarios enviar sus impresiones sin levantarse de sus asientos y sin mover máquinas de un lugar a otro. Sin embargo, debido al alto costo que representa

la adquisición de una red, sólo puede ser aplicada en lugares en donde, además del trabajo de impresión se requieran compartir muchos otros recursos entre los usuarios de las PC's. Además, la red también puede esclavizar a una computadora como el "servidor" de la red.
Otro camino que se puede escoger es el enlazar a las PC's con dispositivos especiales para compartir la impresora. Ya existen algunos de éstos en el mercado.

Extendiéndonos en esta última opción mencionaremos que los hay de varios tipos. Los primeros a la venta fueron mecánicos. Para enlazar a una determinada computadora con el disositivo impresor es necesario oprimir un botón o bien girar una perilla selectora. Estos aparatos por lo regular se presentan en configuraciones de enlace de 4, 6 u 8 PC's, lo que implica que se requiere más de un aparato para el enlace de más computadoras. Además no siempre es posible colocar la perilla o el botón cerca de todos los usuarios, así que es necesario trasladarse hacia el selector. Posteriormente han aparecido a la venta dispositivos no mecánicos que reconocen automáticamente cual de las PC's conectadas desea enviar la impresión. Estos últimos equipos resuelven el problema de accionar una perilla selectora, pero aún así presentan como los anteriores, la dificultad de que se enlazan totalmente a la computadora que primero envió la señal para imprimir, y no atienden a nadie más sino hasta terminar con ese trabajo. Esta característica vuelve al problema de la fila que, aunque ya no es de gente, sí es de computadoras detenidas.

Nuestra tesis presenta el diseño de un equipo que permite compartir los dispositivos impresores con las PC's conectadas, resolviendo las deficiencias que presentan aparatos similares. En páginas subsecuentes se mostrarán las características y ventajas que tiene este diseño sobre sus análogos, tanto en operación como en costo.

COMUNICACION ENTRE EQUIPOS DIGITALES

Introducción

Este capítulo debe constituir la sólida base para entender la parte de comunicaciones del 3GJ. Hablamos en el inciso 2.1 acerca del modelo OSI-ISO el cual nos permite conocer los diversos niveles de comunicación que pueden existir y de esta manera situarnos en el contexto de las comunicaciones y con propiedad poder discernir cuando se requiera un circuito o un procedimiento de comunicación.*

En el inciso 2.2 de este capítulo encontramos las diferencias entre la transmisión serie y paralelo, mostramos ventajas y desventajas y averiguamos cuándo podemos utilizar una o la otra. Hablamos también acerca de los niveles que se utilizan en una comunicación serie y de conceptos tales como el bit de start, bit de stop, nivel de idle, marca y espacio.

Adentrándonos un poco más en la transmisión serie encontramos las diferencias entre comunicación HDX, FDX y SPX.

Asimismo analizamos circuitos básicos que pueden recibir y transmitir, y vemos cómo se producen errores llamados de sobreflujo, marco o estructura y paridad.

En el inciso 2.3 nos adentramos a profundidad en el estándar RS-232-C por considerarlo de vital interés para el buen desarrollo del 3GJ. Así pues, comenzamos con la historia del estándar, su versión en el CCITT (Comité Consultatif International Telephonique et Telegraphique), qué es lo que abarca esta norma, para después hablar de características eléctricas y más tarde funcionales.

* Con 3gj nos referimos al controlador de impresión.

A continuación seguimos profundizando en el estudio del RS-232-C en cuanto a la xparte funcional, determinando cómo nos podríamos enlazar a la red conmutada, a líneas privadas, en modo HDX o FDX, cuándo y con qué **pines** *lograríamos una comunicación síncrona, cuándo se utilizan las señales secundarias de la norma, qué hacer en el caso de que los equipos no requieran de modems (dispositivos moduladores/demoduladores) para enlazarse y haya necesidad de hacer conexiones "cruzadas", para finalmente llegar a determinar qué señales de todas las que ofrece la norma, requerimos para el 3GJ.*

En el inciso 2.4 analizamos protocolos y determinamos si realmente se necesitan para el 3GJ o no.

En el inciso 2.5 estudiamos el estándar CENTRONICS para comunicación en paralelo, el cual por su sencillez es fácil de explicar en unos cuantos párrafos en los que se describe qué es, sus características eléctricas y funcionales.

2.1 El Modelo OSI-ISO

El proyecto que hemos desarrollado permite que exista una comunicación de varias computadoras personales hacia una o varias impresoras. Aún cuando no se constituye una red de computadoras en forma, sí se hace necesario que nos ajustemos a un cierto modelo, lo más universal posible para que encontremos compatibilidad con el resto de los equipos a los cuales vamos a conectar el 3GJ.

Con el objeto de tener cierta normatividad, la Organización Internacional de Estándares (ISO), determinó el modelo OSI: Open System Interconnection, en donde se esquematizan los diferentes niveles de comunicación en los que opera una red de computadoras.

En este modelo se definen siete estratos, cada uno de los cuales tiene funciones bien definidas, así, un elemento para la comunicación entre equipos digitales puede colocarse perfectamente en alguna de las 7 capas de que consta el modelo. También podemos decir que si tenemos una clara definición de este modelo, podremos distinguir la relación que exista entre diferentes elementos de la red o cómo uno afecta al otro. Procedamos ahora a una breve descripción de estos niveles.

Nivel 1. Nivel Físico

Este nivel abarca la conexión real física entre los elementos individuales de la red y el cable o medio con que se conectan unos con otros. En este nivel se tiene que ver con las características mecánicas del conector, del cable o medio, señales eléctricas como pueden ser voltajes, corrientes, potencia, etcétera. En este nivel cae el estándar CENTRONICS, normas de la EIA (Electrical Industry Association) como la RS-232-C, RS-422, RS-423, RS-449 y algunas recomendaciones del CCITT.

Nivel 2. Nivel de Liga de Datos (Data Link)

En este nivel se manejan los formatos que pueden tener los mensajes, velocidades de transmisión, señales de control y otras funciones semejantes. Aquí es donde se clasifican los diversos protocolos de comunicación, los cuales se explicarán de una forma breve más adelante. Sin embargo, adelantándonos un poco, diremos para ejemplificar este nivel, que en un protocolo cualquiera el mensaje se "envuelve" en una serie de bits que van a servir de clave al receptor para saber si realmente se trata de un mensaje válido o no. Ejemplos de protocolos son el HDLC, BISINC y otros muchos.
El aparato por nosotros desarrollado, no maneja ningún protocolo conocido, el protocolo que maneja fue diseñado por nosotros , decisión que fue tomada en aras de obtener un dispositivo más sencillo tanto en hardware como en software.

Nivel 3 y 4. De Red y de Transporte

Estos niveles se relacionan con los circuitos de comunicaciones. De hecho, no basta un cable para comunicarse de un lugar a otro, así pues, en una red telefónica se requiere de circuitos que permitan el intercambio automático de llamadas telefónicas, circuitos que asignen un número telefónico. En una red de computadoras se requiere de circuitos que direccionen los mensajes y controlen el circuito, previniendo errores y colisiones, y protegiendo así el manejo de las señales de un punto a otro.

No existe una clara distinción entre los niveles 3 y 4, pero sí se hace una división lógica entre los niveles altos y bajos. Dicha división es entre los niveles 4 y 5. Los primeros 4 niveles se relacionan con la transmisión de datos, los siguientes 3 con el uso de éstos una vez que se han comunicado o transportado.

También hay ocasiones en que se tienen que realizar traducciones entre redes de diferentes tipos. Es en el nivel 4 donde se realizan estas traducciones.

Nivel 5. De Sesión

Este nivel se relaciona con un "diálogo" entre las redes o elementos pertenecientes a una red. Como ejemplos representativo podemos mencionar cuando se envía a imprimir un archivo desde una terminal hacia una impresora o un mensaje a otra parte de la red. En ambos casos se esta conduciendo una "sesión" en una forma que tiene significado para los elementos de la red.

Nivel 6. De Presentación

El objetivo de este nivel es presentar ante cada dispositivo de la red los datos en un formato apto para su reconocimiento; efectuando, si es necesario, la conversión de un formato a otro, tal como puede ser la conversión de un formato EBCDIC a ASCII.

Nivel 7. De Aplicación

Este nivel tiene que ver con las aplicaciones. Es el uso particular que se le da a un dispositivo de la red y su relación con el operador humano. En este nivel se realizan las tareas útiles, razón que justifica o forma el objetivo central de tener una computadora en un primer plano de utilización.

A su vez, un código o elemento de la red puede estar situado en varios de los niveles del modelo OSI-ISO, como es el caso del código ASCII, el cual se sabe que puede estar catalogado en el nivel 6, de PRESENTACION, asimismo este código posee caracteres de control los cuales se utilizan en muchos protocolos por lo que también puede catalogarse en el segundo nivel, de DATA LINK.

El equipo por nosotros desarrollado, se relaciona muy cercanamente con el modelo aquí planteado. El conocimiento de estos niveles nos ayudó al diseño tanto del circuito como de los programas necesarios para hacerlo funcionar.

2.2 Principios Básicos de Comunicación

Transmisión Serie y Paralelo

En este inciso se dará una breve explicación de cómo los datos pueden fluir de un dispositivo a otro. Se sabe que los datos pueden transmitirse en paralelo o en serie. El transmitir datos en paralelo significa que se cuenta con un alambre para transmitir cada uno de los bits que forman una palabra o un caracter. Además se cuenta con unas cuantas líneas de control (strobe, acknowledge, busy, etcétera), las cuales se encargan de dictaminar cuándo los bits que forman el mensaje están listos para transmitirse, cuándo se han recibido o bien si tienen que esperar a enviarse debido a que el dispositivo que los recibirá no puede tomarlos en ese momento (Véase el inciso 'Estándar CENTRONICS' en este mismo capítulo).

En contraposición, en la transmisión serie se cuenta con un solo alambre o hilo por el cual fluyen los bits, uno por uno. Los bits enviados por esta única línea no son solamente el mensaje a transmitir, sino que también se envían algunos caracteres de control. Asimismo, existen dos maneras de manejar la transmisión serie, a saber: la síncrona, en la cual se conoce de manera sincronizada el instante en el cual se enviarán los datos; y la asíncrona en la cual no se conoce el momento en el cual se enviará el mensaje.

A estas alturas del inciso cabe preguntarse cuándo utilizar la transmisión paralela y cuándo la serie. Ambos tipos de transmisión tiene ventajas uno sobre el otro. Así pues, la ventaja de la transmisión en paralelo es que necesariamente debe ser más rápida que la serie puesto que en un mismo instante pueden viajar todos los bits constitutivos de un caracter, procedimiento físicamente imposible en una transmisión serie. Por otro lado, transmitir en serie llega a ser más costeable, sobre todo cuando la distancia a cubrir es muy grande, así, es lógico pensar que para cubrir una distancia de 5 km entre una terminal y una computadora es más económico tender un hilo que tender 8 ó 10. También se ha observado que la transmisión en paralelo deja de ser eficiente a distancias, a veces, arriba de los 2.5 metros puesto que por efecto de inductancias y/o capacitancias entre los mismos hilos no todas las señales alcanzan el nivel que deben de tener en el extremo del receptor.

La transmisión en paralelo se utiliza dentro de una computadora en donde la distancia

entre un circuito y otro es muy peqeueña y en donde la velocidad de procesamiento es tan alta que lo ideal es que los bits constitutivos de una palabra fluyan en el mismo instante. Es también muy común la transmisión en paralelo cuando los equipos están a distancias cortas como puede ser el caso de una microcomputadora y su impresora. Cuando las distancias aumentan es mejor utilizar la transmisión serie.

El equipo por nosotros desarrollado utiliza ambos tipos de transmisiones. Está pensado para tener comunicación con varias microcomputadoras, las cuales pueden estar muy lejos del multiplexor debiendo por tanto existir comunicación en serie entre la computadora personal y el 3GJ. Por otra parte, la comunicación con la impresora será preferentemente en paralelo por cuestiones de velocidad, y el número de impresoras a las que podrá direccionar el 3GJ en esta forma será de dos, las cuales, está pensado, se localicen cerca del multiplexor. Además tendrá la opción de comunicarse con una impresora en forma serie, ya que existen dispositivos impresores como los graficadores los cuales sólo reciben en forma serie o bien, para el remoto caso en que la impresora se encuentre muy lejos del multiplexor.

Niveles o Estados Lógicos en la Transmisión

Pasemos ahora a hablar acerca de los bits utilizados en el flujo de datos. La historia se remonta a los primeros teleimpresores, en los cuales por construcción y para evitar fallas, se consideró que una línea en estado desocupado (idle) era aquélla por la cual fluía una corriente (Recuérdese que los teleimpresores actuaban a base de accionar solenoides). Así pues, la transmisión de datos empezaba al interrumpirse el flujo de corriente en la línea. Por convención, al estado de idle se le llamó MARCA o "1" en tanto que al de interrupción de corriente se le llamo ESPACIO o "0".

También se determinó que una transmisión asíncrona debería de empezar pasando del estado de idle o marca a un estado de espacio conocido como "START BIT". A continuación debería fluir el número de bits correspondiente al código y que puede ir desde 5 hasta el más común que es el ASCII con 8 bits. Terminando la transmisión con uno, uno y medio o hasta dos "BITS DE STOP" los cuales son niveles "1". Después de esto, la línea debe de regresar a un estado de idle hasta que se requiera transmitir otro caracter, momento en el cual la línea cambiará de nivel a causa del START BIT (véase la figura 2.1).

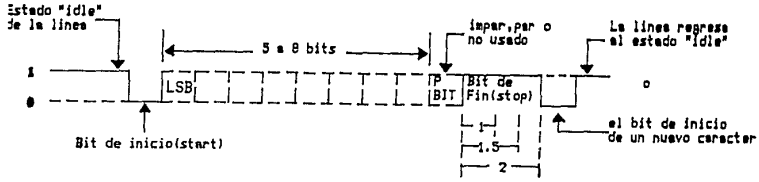


Figura 2.1 Formato de un Caracter de Datos Asíncronos

El primer código empleado fue uno de 5 bits ideado por Emil Baudot en 1874, y utilizado en el manejo de teleimpresores. Este es un ingenioso código con el cual se podían manejar hasta 63 caracteres diferentes, los cuales eran primordialmente dígitos, caracteres en minúsculas, mayúsculas y un caracter especial para el cambio a mayúsculas. Fue de 5 bits y no más debido a que no existía necesidad de transmitir más que letras y números, también porque existían problemas de sincronía entre los equipos transmisores y los receptores, estos últimos generados por deficiencias en la regulación del voltaje y la frecuencia en el suministro eléctrico. Así pues, tecnológicamente era la mayor cantidad de bits que podía manejarse.

Sin embargo, años después, en las imprentas existió la necesidad de poder manejar mayor cantidad de caracteres por lo que surgieron códigos de 6 bits y, por necesidades de crecimiento actualmente el más popular de los códigos es el código ASCII el cual con 8 bits puede manejar hasta 256 símbolos diferentes.

Transmisión Serie Asíncrona y Síncrona

En la modalidad asíncrona, el envío de datos puede efectuarse cuando se desee, en una forma digamos "aleatoria", en tanto que en la modalidad síncrona el momento en que los datos se enviarán será perfectamente conocido tanto por el lado transmisor como por el receptor, y además, la información se enviará por paquetes. En esta

última modalidad, el intercambio de información es más eficiente, puesto que sólo se envía un start y un stop bit a lo mucho al principio y fin de bloque respectivamente, en tanto que en la transmisión asíncrona por cada caracter enviado se tiene que enviar un Start bit al principio y un stop bit al final como lo muestra la figura 2.1.

De la modalidad síncrona hablaremos más adelante.

Transmisión SPX,HDX,FDX

Existen varias maneras de utilizar una línea o medio de enlace en donde sabemos existen un transmisor y un receptor.

En la primera únicamente se permite que la información fluya hacia el receptor pero no se autoriza que el receptor se convierta en transmisor. Esta modalidad se conoce como Transmisión Simplex o SPX.

En la segunda, denominada Half Duplex o HDX, se permite que la información fluya en ambos sentidos pero no simultáneamente. Si existen 2 puntos (A y B) que requieren comunicarse. Primeramente A transmite hacia B y una vez que A finalizó de transmitir, B puede hacerlo. Pero sólo hasta entonces.

La comunicación Full Duplex, FDX, es la más versátil de todas, ya que permite que ambos puntos se comuniquen al mismo tiempo: en forma simultánea.

Velocidades de Transmisión

En una transmisión serie se manejan velocidades cuyas unidades son los bits por segundo (bps) y el baud. La velocidad de transmisión entre dos equipos digitales se mide en bits por segundo, en tanto que la velocidad de información o señalización se mide en bauds. Es frecuente que estos dos términos coincidan sobre todo cuando una señal esta compuesta por un bit, pero cabe aclarar que son dos términos diferentes. En esta tesis, trataremos de hacer un uso correcto de ambos conceptos.

Existen varios circuitos para detectar el arribo de información asíncrona como el mostrado en la figura 2.2 .

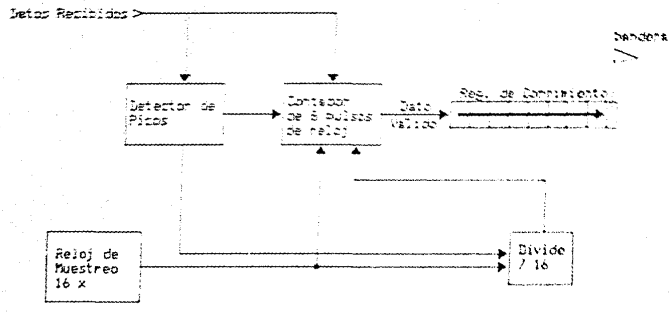


Figura 2.2 Receptor Asíncrono de Datos Serie

Este circuito es capaz de detectar la transición entre un uno y un cero, es decir, la llegada de un bit de start. Basa su funcionamiento en un reloj que muestrea a una velocidad (frecuencia) que es igual a 16 veces la frecuencia del reloj de recepción (16x el reloj de recepción). Así, cuando se detecta un cambio de nivel se dispara un circuito detector de ruido, el cual toma una muestra (muestrea) la señal 8 ciclos del reloj de muestreo después. Si en ese momento el nivel sigue siendo cero, entonces se asume que ha sido un verdadero bit de start y que habrá que recibir el caracter.

Dieciséis ciclos de reloj después, se vuelve a muestrear la línea leyéndose en esta manera el siguiente bit. A veces también en aras de mejorar el circuito disminuyendo la posibilidad de leer ruido, se muestrea muchas veces más, dígase 32x o 64x el reloj de recepción. Sólo que entonces debe de tenerse en cuenta que a mayor frecuencia de muestreo, mayor capacidad deben de tener los contadores de muestreo y

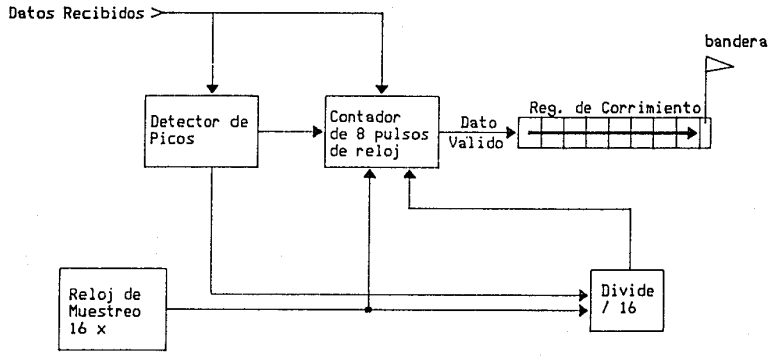


Figura 2.2 Receptor Asíncrono de Datos Serie

Este circuito es capaz de detectar la transición entre un uno y un cero, es decir, la llegada de un bit de start. Basa su funcionamiento en un reloj que muestrea a una velocidad (frecuencia) que es igual a 16 veces la frecuencia del reloj de recepción (16x el reloj de recepción). Así, cuando se detecta un cambio de nivel se dispara un circuito detector de ruido, el cual toma una muestra (muestrea) la señal 8 ciclos del reloj de muestreo después. Si en ese momento el nivel sigue siendo cero, entonces se asume que ha sido un verdadero bit de start y que habrá que recibir el caracter.

Dieciséis ciclos de reloj después, se vuelve a muestrear la línea leyéndose en esta manera el siguiente bit. A veces también en aras de mejorar el circuito disminuyendo la posibilidad de leer ruido, se muestrea muchas veces más, dígase 32x o 64x el reloj de recepción. Sólo que entonces debe de tenerse en cuenta que a mayor frecuencia de muestreo, mayor capacidad deben de tener los contadores de muestreo y

además,debe de considerarse que a velocidades altas, como 9600 bps, la frecuencia de muestreo excede las capacidades de muchos circuitos construidos con tecnología CMOS tal como el 8251 que es utilizado en nuestro circuito multiplexor. Una vez recibido el caracter en un registro de corrimiento, se prende una bandera que es ***poleada*** *(verificada a intervalos) por la computadora o bien interrumpe a ésta. La computadora puede entonces leer el caracter en paralelo y procesarlo.*

Errores de Sobreflujo

Sin embargo, puede advertirse que la computadora sólo tiene el tiempo que dura el bit de stop para tomar el caracter, de lo contrario, empezará a llegar el siguiente y se encimará con el primer produciéndose un error de encimamiento u "OVERRUN" y leyéndose todo de manera errónea. Algunos circuitos para transmitir son capaces de detectar esta condición de error prendiendo una bandera que puede leerse en el registro de status. El tiempo en el que el microprocesador tome el dato puede aumentarse si se obtiene una copia del registro de corrimiento en donde ha llegado el caracter, así, se dispondrá del tiempo en que tarda en llegar el siguiente caracter para leer el que llegó primero.

Errores de Marco o Estructura

Estructurar es el proceso de decidir qué grupos de 8 bits constituyen un caracter. En ocasiones, porque la línea se abre, porque el reloj de muestreo se pierde o porque el transmisor envía otra clase de señal, el bit de stop no se detecta. Puede ser que inclusive se llegue a detectar una marca perteneciente a información de otro caracter o bien un "0" produciéndose una condición de error llamada "ERROR DE ESTRUCTURA", "ERROR DE MARCO" o "FRAMING ERROR".

Es fácil corregir este error. Una manera de hacerlo, puede ser haciendo que la línea permanezca en estado de idle durante un largo periodo de tiempo. Varios circuitos de comunicaciones son capaces de tomar de nuevo la estructura correcta. Se puede ejemplificar cómo hacerlo: Escriba varios caracteres y anteponga los bits de start y al final coloque los bits de stop correspondientes. Suponga que se produce un error de estructura y se toma el siguiente "1" que aparezca en la lista como bit de stop. A continuación después de 8 bits debe seguir un bit de stop, si no es así, se buscará un siguiente 1 para tomarlo como bit de stop. Repitiéndose este proceso finalmente

se retomará la estructura adecuada.

Paridad

Otra manera de evitar errores es por medio del bit de paridad. Después de enviar el caracter de información y antes de enviar el bit de stop, se envía un bit (bit de paridad) que indica si en el caracter existe un numero par de "1"s o no. Si existen, entonces el bit de paridad valdrá "0" para evitar alterar ese número par de "1"s. Si se envían un numero impar entonces el bit de paridad es "1" para hacer que el número de "1"s sea par.

Esta paridad en la que se hace que el número de "1"s sea par se conoce como paridad par. También puede hacerse lo contrario, es decir, hacer que el número de bits enviados sea impar, y esta paridad se conoce como impar. Así en el lado receptor, podrán contarse el número de "1"s y compararse con el bit de paridad para detectarse errores. De encontrarse errores puede prenderse una bandera que avisará al microprocesador que un error de paridad existe para que se tomen las medidas adecuadas y corregirlo.

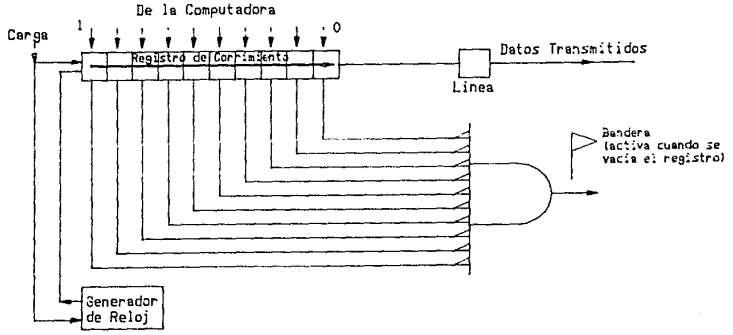


Figura 2.3 Transmisor Asíncrono de Datos Serie

Existen otras muchas maneras de revisar errores las cuales involucran algoritmos más complicados, y también existen circuitos capaces de manejar estos algoritmos. Dadas las condiciones que tenemos en las líneas y equipos que utilizamos en nuestro proyecto, nosotros no manejamos esos algoritmos.

Circuitos para Transmitir

Un circuito de transmisión puede ser aún más sencillo. Puede consistir básicamente en un registro de corrimiento el cual es cargado en paralelo por la computadora y que puede programarse o alambrarse de tal manera que al principio del caracter se ponga un bit de start y después del bit de paridad (si se usa) se ponga un "1" como un bit de stop.

Por medio de un reloj de transmisión puede lograrse que se mueva el registro de corrimiento sacando su información hacia un flip flop el cual es conectado a la línea. Conforme se vayan sacando los bits, en el registro de corrimiento se irán colocando ceros de forma que cuando todo el caracter haya sido transmitido y existan sólo ceros en el registro de corrimiento, se prenderá una bandera que interrumpa al procesador o sea poleada por él avisándole que el **buffer** *(registro intermedio) está vacío y que puede cargar otro caracter.(Véase la figura 2.3.)*

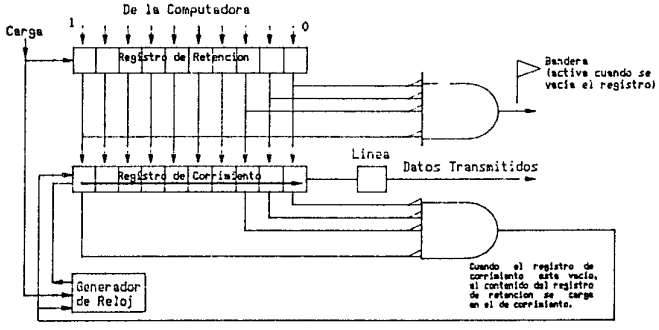


Fig 2.4 Transmisor Asíncrono con Doble Buffer

La eficiencia puede mejorarse si se pone otro buffer de tal forma que la computadora pueda dejarlo cargado previamente y así cuando se vacíe el registro de corrimiento,se cargue automáticamente con el dato depositado anteriormente y se empiece a transmitir inmediatamente. De esta forma la computadora tiene el tiempo que dura la transmisión de este caracter para volver a cargar el buffer. (Véase la figura 2.4.)

2.3 Estándar RS-232C

En los primeros años de las comunicaciones de datos, los asociados a la American Telephone and Telegraph Company (AT&T) fueron los que marcaron la pauta en cuanto a estándares. Así, fabricantes de modems como los "Laboratorios Bell" y la "Western Electric" marcaron el estándar en esta industria. Sin embargo, conforme fue avanzando el tiempo, nuevos fabricantes de equipos fueron apareciendo en el mercado de forma que llegó el momento en que existió la necesidad de crear un verdadero estándar.

Es así como los ingenieros de la "Electrical Industry Asociation" en colaboración con los "Laboratorios Bell" y fabricantes independientes de modems y computadoras desarrollaron un estándar para la interface entre Equipo Terminal de Datos (Data Terminal Equipment, DTE) y Equipo De Comunicación de datos (Data Communication Equipment, DCE) empleando un intercambio de datos en forma serie. Este estándar es conocido como RS-232-C. La letra C hace referencia a la tercera revisión a este estándar.

Se han desarrollado nuevos estándares como el RS-422 y el RS-423 los cuales se tiene pensado que suplanten al RS-232, sin embargo, hoy por hoy, el RS-232 es el más utilizado en la actualidad por fabricantes de modems y computadoras.
Es bien sabido que en cada país, cada gobierno es el encargado de proporcionar o regular el servicio telefónico, Asimismo, los fabricantes de equipos de comunicaciones vendían éstos en diferentes países por lo que era muy recomendable ponerse de acuerdo en cuanto a qué normas seguir.

Fue entonces cuando el "Comité Consultatif International Telephonique et Telegraphique" (CCITT) en cooperación con los países de las Naciones Unidas promulgaron una serie de recomendaciones que cubren todas las fases de las telecomunicaciones, desde procesos para operar hasta procesos para controlar la

interferencia eléctrica, provocada por los cables de transporte eléctrico en cables para comunicaciones. Estas recomendaciones son revisadas en asambleas plenarias aproximadamente cada 4 años. Resultado de estas asambleas es un libro en varios volumenes con la actualización a dichas recomendaciones conocido por el color de su pasta. (Para esta tesis nos estamos basando en el "libro naranja" emitido por la asamblea en Génova en 1976.) El volumen que se refiere a transmisión de datos es el VIII y las recomendaciones tienen los prefijos V y X.

Retornando al RS-232-C es importante notar que hace referencia a:

1. Características eléctricas de la señal.
2. Características mecánicas de la interfase.
3. Descripción funcional de las señales en el intercambio de información.
4. Especificaciones para sistemas en casos especiales.

En esta tesis profundizaremos sólo un poco en los puntos 1 y 3 anteriores.

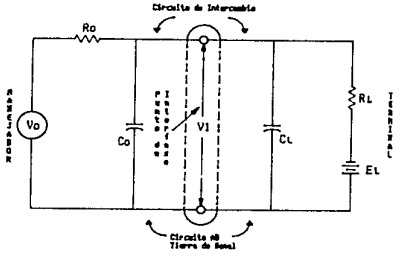


Figura 2.5 Circuito de Intercambio Equivalente

El decir que un equipo es compatible con el RS-232 generalmente se refiere a que no viola las disposiciones mecánicas y eléctricas de la norma, sin embargo en la mayoría de los casos no basta con cumplir con los niveles de voltaje de la señal sino que ésta debe de estar presente en el momento adecuado. Ahora bien, aunque es sabido que un RS-232 puede manejar hasta 25 señales, no siempre todas se utilizan. El número de señales que se utilicen en mucho dependerá del equipo con el que se trabaje, como se verá más adelante.

Características Eléctricas

Resumiremos las características eléctricas de la señal para lo cual nos será útil el

circuito mostrado en la figura 2.5.

1. La señal se considerará Marca "1" si su voltaje V1 es menor a -3V con respecto a la tierra de señal. Se considerará un Espacio "0" si es mayor a +3V con respecto a la tierra de señal. La región entre los -3V y +3v seconsidera una región de transición cuyo estado no está definido.

2. El circuito manejador deberá dar un voltaje entre -5 y -15V para ser considerado como una Marca, "1" o control apagada. Deberá estar entre +5 y +15V para considerarse como un espacio, "0" o control encendido. Esta regla en conjunto con la 1 deja un margen de ruido de 2V como máximo.

3. Cuando se cumplan los puntos expresados en la regla 8 y además el voltaje EL sea 0, el voltaje V1 deberá estar entre los 5 y 15 V.

4. El voltaje de circuito abierto no debe exceder los 25 V.

5. La impedancia del circuito manejador deberá ser mayor a 300 ohms con el circuito apagado.

6. Un pin de salida debe ser capaz de mantenerse en circuito corto con otro sin dañar o dañarse a sí mismo manteniendo una corriente máxima de 0.5 A.

7. El circuito manejador deberá cambiar a no menos de 30 V/μs y la zona de transición deberá pasarla en 1 ms o 4% del tiempo que dura 1 bit, la que resulte más pequeña.

8. La impedancia de carga RL y CL deberá ser menor a 7000 ohms cuando se mida con un voltaje aplicado entre los 3 y 25V y mayor a 3000 ohms cuando el voltaje aplicado sea mayor a 25V.

9. La capacitancia en el circuito terminal CL no deberá exceder los 2500 pF, incluyendo la capacitancia del cable.

Las reglas antes enunciadas se resumen en la tabla de la figura 2.6.

Características Funcionales

Es común en esta clase de trabajos profesionales remitir al lector a un apéndice o a un manual en donde se enuncia la parte funcional del estándar. Nosotros además explicaremos, en propias palabras, cómo funciona.

Nivel de Salida Lógica con una carga entre 3 y 7 Kohms	5V < Voh <15V -15V < Vol <-5V
Voltaje de Salida en Circuito Abierto	/Vo/ < 25V
Impedancia de Salida con el Circuito Apagado	Ro > 300 ohms
Corriente de Circuito Corto	/Io/ < 0.5 A
Slew Rate del Circuito de Salida	dv/dt < 30 V us
Impedancia de entrada del Receptor	3 Kohms < Rin < 7 Kohm
Voltaje de Entrada del Receptor	+/- 15 V compatible con la salida
Salida del Circuito Receptor con entrada desconectada	MARCA
Salida del Circuito Receptor con entrada en + 3 V	ESPACIO
Salida del Circuito Receptor con entrada en - 3 V	MARCA
desde + 15 hasta + 5	"0" Lógico = ESPACIO = CONTROL ON
desde + 5 hasta + 3	Margen de Ruido
desde + 3 hasta - 3	Región de Transición
desde - 3 hasta - 5	Margen de Ruido
desde - 5 hasta - 15	"1" Lógico = MARCA = CONTROL OFF

Figura 2.6 Especificaciones Eléctricas Condensadas del EIA RS-232-C

Es posible clasificar las 25 señales en cuatro grupos fundamentales

1. *Señales de tierra:*
   - *Tierra Física*
   - *Tierra de Señal*
2. *Señales de Temporización:*
   - *Reloj de Transmisión*
   - *Reloj de Recepción*
3. *Señales de control:*
   - *Data Terminal Ready (DTR)*
   - *Data Set Ready (DSR)*
   - *Request to Send (RTS)*
   - *Clear to Send (CTS)*
   - *Data Carrier Detect (DCD)*
   - *Las 3 arriba mencionadas, pero secundarias*
   - *Ring Indicator (RI)*
   - *Signal Quality (SQ)*
4. *Señales de Datos:*
   - *Transmisión (TX)*
   - *Recepción (RX)*
   - *Señales Secundarias de Transmisión y Recepción*

Si se posee un equipo con puerto serie, existen innumerables aplicaciones en donde nos podemos enlazar. No en todos los casos es necesario utilizar la totalidad de los pines con que cuenta un RS-232. En los siguientes párrafos diremos cómo puede utilizarse cada línea, y luego particularizaremos para el caso del 3GJ, en donde diremos qué pines usamos y por qué.

Es bien sabido que si queremos enlazar nuestra computadora por medio de un modem y utilizando la red conmutada de servicio público telefónico (RCSPT) podemos hacerlo con:

- *-Bases de Datos Públicas*
- *-Oficinas de Servicios*
- *-Servicios de Mensajes*
- *-Intercambio de información entre computadoras*

Al marcar un número telefónico donde existe un modem, la llamada se anuncia en

al pin 22, Ring Indicator, el cual se encarga de avisar que se recibe una llamada. El modem en el lado remoto detectará el DTR (pin 20 de la computadora y la terminal). Si este aparato así lo desea, el usuario o mediante una computadora o indicará para establecer el enlace por lo que el modem local recibe la llamada. Si está encendido el DTR, entonces el modem comienza la secuencia encendiendo el DSR (pin 6) por otra parte en el lado local también se encenderá el DSR siempre y cuando este encendido el DTR del equipo DTE local. De no ser así quedará establecido el enlace. Si el DTR del lado local se encuentra apagado muchos modems modernos incluyen un circuito de Time Out que hace que el modem remoto cuelgue después de cierto tiempo en que no recibe señal del lado local.

Lo anterior se puede resumir de la siguiente manera:

1. El número es marcado.
2. El Ring Indicator se enciende en el lado remoto.
3. Si el DTR del DTE remoto está encendido, entonces el modem remoto encenderá el DSR.
4. En el modem local se encenderá el DSR.
5. Se presionará un botón el DTE local para que la línea en el lado local se conmute del aparato telefónico al modem y el DTR de ambos lados mantendrá la continuidad del enlace.
6. El enlace está establecido.
7. El intercambio de datos puede llevarse al cabo.

En la actualidad existen modems "inteligentes" capaces de marcar diferentes números telefónicos almacenados en su memoria con simplemente presionar botones al frente del modem o requerírselo desde la terminal o computadora. Pero por mucha eficiencia e inteligencia que posean, la misma debe de ser transparente en el momento del enlace y manejar el estándar de la manera convencional.

La comunicación como se enunció anteriormente, se mantendrá mientras exista DTR en ambos lados. El operador de la terminal podrá cortar el enlace de una de las siguientes maneras:

1. Manualmente cortando el enlace (por medio de botones en el modem).

el pin 22, Ring Indicator, la cual se enciende cada vez que se detecta una corriente de llamada. El modem en el lado remoto detectará el DTR (pin 20) de la computadora y la terminal. Si está apagada, será señal de que el equipo está apagado o indispuesto para establecer el enlace por lo que el modem nunca contestará al llamado. Si está encendido el DTR, entonces el modem contestará la llamada, encendiendo su DSR (pin 6) y por otra parte en el lado local también se encenderá el DSR siempre y cuando esté encendido el DTR del equipo DTE local. De ser así quedará establecido el enlace. Si el DTR del lado local se encuentra apagado, muchos modems modernos incluyen un circuito de **Time Out** *que hace que el modem remoto cuelgue después de cierto tiempo en que no recibe señal del lado local.*

Lo anterior se puede resumir de la siguiente manera:

1. *El número es marcado*
2. *El Ring Indicator se anuncia en el lado remoto.*
3. *Si el DTR del DTE remoto está encendido, entonces el modem remoto encenderá el DSR.*
4. *En el modem local se encenderá el DSR.*
5. *Se presionará un botón el DTE local para que la línea en el lado local se conmute del aparato telefónico al modem y el DTR de ambos lados mantendrá la continuidad del enlace.*
6. *El enlace está establecido.*
7. *El intercambio de datos puede llevarse al cabo.*

En la actualidad existen modems "inteligentes" capaces de marcar diferentes números telefónicos almacenados en su memoria con simplemente presionar botones al frente del modem o requerírselo desde la terminal o computadora. Pero por mucha circuitería e inteligencia que posean, la misma debe de ser transparente en el momento del enlace y manejar el estándar de la manera convencional.

La comunicación como se enunció anteriormente, se mantendrá mientras exista DTR en ambos lados. El operador de la terminal podrá cortar el enlace de una de las siguientes maneras:

1. *Manualmente cortando el enlace (por medio de botones en el modem).*

2. *Poniendo la terminal en modo local (se apaga automáticamente el DTR).*
3. *Desconectando o apagando la terminal.*

En el lado donde se encuentre el equipo central de cómputo el enlace se puede cortar de una de las siguientes maneras:

1. *Al apagar o desconectar la computadora.*
2. *Cuando el programa en ejecución controla el DTR a voluntad.*
3. *Frecuentemente si el enlace se hace con un equipo muy grande, existirá un Front End Processor (Procesador de Comunicaciones) el cual esperará una secuencia Ctrl-D la cual será interpretada para inactivar el DTR cortando el enlace.*

Ahora que ya explicamos la manera en la que se logra, se mantiene y se corta un enlace, hablaremos acerca del intercambio de datos. Primeramente trataremos la comunicación HDX, la cual requiere un control más estricto de la línea.

HDX y FDX

Cuando la terminal local requiere transmitir algo, levantará su RTS, el modem al detectarlo, monitoreará al modem remoto para saber si aquél en ese mismo momento no está ya enviando datos. Si es así, el DCD del modem local estará levantando indicando que la terminal local no puede enviar datos. Si no lo está, entonces el modem local elevará su pin de CTS indicándole a la terminal local que puede enviar datos en ese momento.

En ese mismo instante, en el modem remoto deberá elevarse el DCD evitando que la terminal remota envíe información. A continuación, la terminal podrá enviar su información por el pin 2 Transmisión, datos que llegarán finalmente a la terminal o computadora remota por el pin 3 Recepción. La terminal local podrá enviar información mientras mantenga el control de la línea. Esto lo va a conseguir mientras mantenga en alto el RTS, ya que al apagarlo, el lado remoto si así lo necesita podrá enviar información tomando el control de la línea en la secuencia antes mencionada.

Podemos resumir lo anterior de la siguiente forma:

1. *El RTS se activa. Se monitorea que el DCD no esté encendido.*
2. *Se enciende el CTS indicándole a la terminal que puede empezar a transmitir.*
3. *En ese momento en el pin 2 del DTE que tiene el control de la línea empieza a salir la información.*
4. *En el lado remoto esa información se presentará en el pin 3 (Recepción).*
5. *El lado que transmite lo podrá seguir haciendo mientras mantenga encendido su RTS.*
6. *En el momento de inactivar el RTS, cualquiera de los dos lados, local o remoto, podrá tomar el control de la línea.*

Una comunicación full duplex es más flexible ya que se puede transmitir y recibir simultáneamente. La secuencia de pasos a seguir es la misma, salvo que se modifica el paso 2 y 6 ya que independientemente de que exista el DCD o no, el lado que requiera transmitir lo podrá hacer, y el DCD fungirá solamente como una señal que avisa que en ese momento se está recibiendo información, pero en ese mismo momento podrán transmitirse datos. Por lo que respecta al punto 6, ninguno de los dos lados tendrá el control exclusivo de la línea.

Líneas Privadas

Ya hemos hablado del comportamiento de las señales en una línea conmutada, pero qué sucede en una línea privada. Una línea privada se utiliza:

- *Cuando el flujo de información es demasiado grande como para estar realizando enlaces por línea conmutada en cada ocasión.*
- *Cuando se requiere establecer un puente de comunicación las 24 horas del día.*
- *Cuando uno se quiere evitar la tediosa tarea de marcar un número telefónico (sobre todo aquí en México) cada vez que se quiere enlazar con un mismo lugar.*

Las razones antes enunciadas son lo suficientemente poderosas para que una compañía desee contratar una línea privada para transmisión de datos.

En una línea privada la función de algunos pines del RS-232 cambia un poco y llega a facilitarse el enlace hasta en un 50% como a continuación vamos a ver.

Para empezar, el Ring Indicator no se requiere puesto que jamás se va a marcar un número telefónico. Por otro lado, las señales DTR y DSR en ambos lados se requieren con permanencia continua y su función es indicar que tanto el DTE como el DCE están encendidos. El proceso de intercambio de datos tiene el mismo comportamiento que en las líneas conmutadas.

Como puede apreciarse se simplifica mucho el funcionamiento de las señales, por lo menos de aquellas que lograban, mantenían y cortaban el enlace al grado de que ahora sólo deben estar presentes. Algunas desaparecen como el Ring Indicator y uno ya no es necesario marcar un número telefónico.

Comunicación Síncrona

Ya en este capítulo hemos mencionado algunas de las características de la comunicación síncrona. Recordemos que dijimos que este tipo de comunicación está orientada a enviar bloques de caracteres en vez de enviar de caracter en caracter, ya que en este caso por cada caracter enviado se requiere enviar un start y un stop bit lo cual vuelve deficiente el uso de la línea. En la comunicación síncrona, sólo es necesario enviar un start y un stop bit al principio y fin de bloque. Así, todos los caracteres se juntan en un gran buffer y cuando ya suman un número considerable, entonces se envían. Con este método se logran velocidades altas siempre y cuando se tenga una buena línea y una temporización adecuada. Así, en una línea conmutada es común observar velocidades de 4800 bps en tanto que en una línea privada se pueden lograr velocidades más altas.

La temporización se logra por medio de dos relojes, uno de transmisión y otro de recepción. El reloj de transmisión puede proporcionarlo tanto el DTE como el DCE. Si lo proporciona el DTE se hará en el pin 24 en tanto que si lo proporciona el DCE se hará en el pin 15. El reloj de recepción sólo puede proporcionarlo el DCE en el pin 17. Como puede apreciarse, existen varias fuentes desde donde se puede temporizar,

sólo que mientras más fuentes existan, más difícil es mantener la sincronía. Es por esta última razón por lo que es recomendable utilizar una sola fuente de forma tal que esa fuente alimente tanto al reloj de transmisión como al de recepción.

Aunque esta última forma sea más eficiente, no es utilizada por el circuito 3GJ debido a que las computadoras personales no cuentan con circuitos de comunicación síncrona, además de que la mayoría de los paquetes utilizan la comunicación asíncrona.

Señales Secundarias

Estas señales son utilizadas para diversos propósitos. Generalmente cuando se manejan, es porque existe un canal secundario utilizado generalmente para control. Dicho canal puede ser HDX o FDX independientemente del canal principal. Una aplicación de dicho canal puede ser cuando se tiene una impresora conectada en el canal principal. Este canal bien puede ser SPX ya que la impresora no requiere más que recibir, salvo el caso en que se presentan condiciones de error como puede ser el caso en el que se llene su buffer o bien que se termine el papel. Si en esta línea existe un canal secundario, bien puede utilizarse para que mediante un protocolo XON/XOFF la impresora pueda notificar a la computadora de esta condición de error.

Conexiones Cruzadas

Mejor conocidas como CROSSOVERs, se utilizan en el caso de que se requiera conectar dos equipos DTE's entre sí o DCE's entre sí. Hasta ahora hemos estudiado el caso en el que el pin 2 se conecta con el 2, el 6 con el 6 y así sucesivamente. Pero, si requerimos conectar 2 terminales o una computadora a una terminal, estamos conectando 2 DTE's.

Las tierras nunca tendrán problema y no se requiere hacer crossover alguno. En cuanto a las señales de datos, podemos decir que lo que transmite uno, el otro lo recibe, así, podemos conectar el pin 2 de una con el 3 de la otra y viceversa.

En las señales de control también deben hacerse crossovers, así cuando un DTE quiera transmitir, encenderá su RTS, debido a que no existe DCE que le conteste con un CTS y un DCD, entonces se deberá puentear el RTS al CTS y al DCD para que el

DCD pueda transmitir.

Las señales de temporización tienen el siguiente arreglo. Bien uno de los DTE's puede generar el reloj de transmisión (pin 15), mismo que se puenteará al reloj de recepción del mismo (pin 17) y al reloj de transmisión del otro DTE (pin 24). En la figura 2.7 se muestra el diagrama de un crossover.

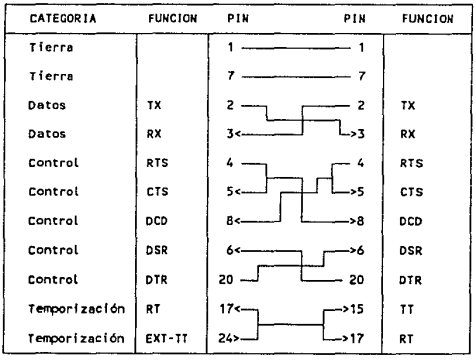


Figura 2.7 Diagrama Crossover

Pueden hacerse crossovers con DCE's, aunque son poco comunes. También existen equipos que pueden programarse para que se comporten como DCE o DTE por medio de puentes que se hagan en la tarjeta de su circuito, botones en el mismo, o programación de sus circuitos.

Conexiones en el 3GJ

Después de analizar a bastante detalle las funciones de las señales del RS-232 podemos decidir qué clase de señales necesitamos.

Así vemos que mientras no lo enlacemos a la red conmutada, no requerimos de Ring Indicator. El 3GJ no proporciona DTR por lo que en la computadora que le envía al

3GJ se puentea su DTR con el DSR y DCD.

El 3GJ controla al CTS de la microcomputadora para detener el envío de información.

Las patas de tierra no requieren crossover y de hecho la única que se utiliza en nuestro circuito es la tierra de señal (pin 7). En cuanto a las señales de datos, se efectúa un crossover entre las patas 2 y 3.

Es de notar que debido a que no utilizamos todas las señales hemos empleado un conector de 9 pines en vez de 25 en el lado del 3GJ llamado DB9.

En la figura 2.8 mostramos el esquema de conexiones del 3GJ con las microcomputadoras, que de hecho es el mismo que cuando se conecta el 3GJ a una impresora serie.

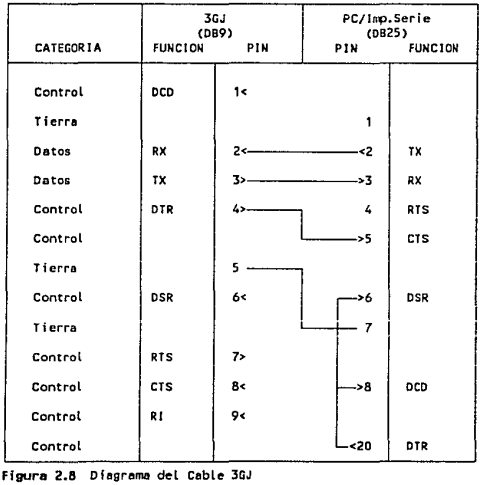


Figura 2.8 Diagrama del Cable 3GJ

2.4 Protocolos, breve descripción

Un protocolo es básicamente un conjunto de reglas para operar un sistema de comunicaciones. Este conjunto de reglas están pensadas para resolver problemas de operación en las siguientes áreas:

1. Estructura, encuadramiento o "Framing"- Determinando cuáles son los bits que constituyen caracteres y lo más importante de todo, qué bloques constituyen los mensajes que se están manejando.

2. Control de Errores- Cuando se transmite un mensaje es importante saber si los bits que fluyeron en la línea son correctos, para lo cual se utilizan diversos métodos de corrección de errores como pueden ser el vertical, longitudinal o CRC. También se involucra con la aceptación de un mensaje dado, o bien con la petición para que se retransmita un mensaje.

3. Control de Secuencia- Por medio de esta técnica se puede evitar que se dupliquen mensajes, que se pierdan o se puede tomar conocimiento de mensajes que son retransmitidos al ser detectados por el control de errores.

4. Transparencia- En muchas ocasiones se requiere transmitir los resultados de mediciones. Es necesario que estos bits en dado caso no vayan a confundir al circuito receptor con caracteres de control de los puntos 1,2 ó 3. Por tanto, en estos casos el protocolo debe ser transparente.

5. Control de línea- En el caso de tener una línea Half Duplex o multipunto, el protocolo puede determinar qué elemento va a transmitir y cuál va a recibir.

6. Casos Especiales- Debe así solucionarse el problema de qué debe transmitir un elemento en caso de que no tenga mensajes que transmitir.

7. Control de "Timeout"- Debe solucionar el problema de qué hacer en caso de que un mensaje deje de recibirse de improviso.

8. Control de Inicio- Deberá encargarse de llevar al cabo el proceso de inicio de transmisión en caso de así necesitarse.

Además, en base a la forma en que se estructura el mensaje, los protocolos pueden clasificarse en:

1. *Protocolos orientados a caracter.*
2. *Protocolos orientados a la contabilización de bits.*
3. *Protocolos orientados a bit*

Los protocolos orientados a caracter se caracterizan por enviar un caracter (por ejemplo un STX) al principio del bloque de caracteres a enviar y otro al terminar éste (por ejemplo un ETX). Un ejemplo clásico de este tipo de protocolos es el es el Binary Synchronous Protocol (BYSINC) de IBM.

Los protocolos orientados a la contabilización de bits envían además de su caracter de inicio de bloque, una cuenta de cuántos caracteres se enviarán y otra información de control que indica cuántos mensajes se recibieron correctamente. Un ejemplo de estos protocolos es el "Digital Data Communication Message Protocol" (DDCMP) de la Digital Equipment Corporation.

Los protocolos orientados a bit, envían casi todo un caracter como bandera antes de otro caracter de mensaje. Por ejemplo puede utilizarse el 01111110 como bandera que señaliza que a continuación se envía un mensaje y otra bandera igual cuando se ha transmitido el caracter de mensaje. Un ejemplo de este tipo de protocolo es el "Sinchronous Data Link Control" (SDLC) de IBM.

Los protocolos son muy utilizados cuando se maneja una comunicación síncrona, porque es en ésta cuando se envían bloques grandes de información que hay que cuidar que no se pierdan. En el caso del 3GJ la comunicación que se maneja es asíncrona, por lo que se envía de caracter en caracter. Sin embargo fue necesario crear un pequeño protocolo orientado a que realice algunas de las funciones "extras" como puede ser la indicación por parte de la PC de qué impresora se quiere utilizar, o de que el mensaje ha terminado.

No utilizamos un protocolo en forma porque no lo requerimos. Como recordamos, un protocolo se encuentra ya en el segundo nivel OSI-ISO. Podemos decir que muchos

de los problemas de control de la línea por ejemplo, los resolvemos en el primer nivel OSI-ISO manejando líneas como el CTS de la PC.

No requerimos encuadre porque se sabe perfectamente cuándo termina un caracter ya que se recibe un stop bit. Tampoco utilizamos control de errores, (ni siquiera la paridad,) porque en pruebas realizadas con el 3GJ nunca surgió la necesidad de hacerlo. De lo anterior se desprende que el mensaje nunca se repite, nunca se envía de más y se conoce el estatus de las impresoras y PC's para saber si están prendidas o no. No se requiere transparencia, pues los caracteres que se envían en su gran mayoría corresponden a letras y números en la tabla ASCII. El control de línea como ya se mencionó se realiza en el nivel inferior OSI-ISO. La línea puede permanecer sin enviar información, y cuando se quiera iniciar este proceso basta con enviar un bit de start. Por último, si súbitamente se deja de recibir información, puede tomarse conocimiento de esto analizando las banderas de overrun y framing del puerto serie involucrado en el 3GJ.

2.5 Estándar CENTRONICS

De la misma manera que existe un estándar para la comunicación serie, existe otro para la comunicación paralelo. El más popular es el CENTRONICS, diseñado por la fábrica de impresoras del mismo nombre.

Este estándar es muchísimo más sencillo de comprender que el RS-232. Por principio de cuentas, utiliza niveles TTL por lo que no hay necesidad de hacer conversiones de voltaje para adecuarse a la línea. En su parte eléctrica, este estándar define 36 señales que se muestran en la tabla de la figura 2.9.

PIN	RET	SEÑAL	DIR	DESCRIPCION
1	19	$\overline{\text{STROBE}}$	ent	Pulso para permitir la entrada de datos. Su anchura debe ser mayor a 0.5 us en el lado receptor. Es una señal activa baja.
2	20	DATO 1	ent	
3	21	DATO 2	ent	

Continuación

PIN	RET	SEÑAL	DIR	DESCRIPCION
4	22	DATO 3	ent	
5	23	DATO 4	ent	Estas señales representan información del primero al octavo bit de datos en paralelo respectivamente. Un nivel alto representa un "1" lógico.
6	24	DATO 5	ent	
7	25	DATO 6	ent	
8	26	DATO 7	ent	
9	27	DATO 8	ent	
10	28	$\overline{\text{ACKNLG}}$	sal	De 5 us. de duración; un "bajo" indica que los datos han sido recibidos por la impresora y está lista para recibir más.
11	29	BUSY	sal	Un "alto" indica que la impresora no puede recibir datos. Adquiere este nivel en los siguientes casos: 1. Durante la entrada de datos 2. Durante la operación de impresión 3. En estado OFFLINE 4. Durante estado de error
12	30	PE	sal	Un "alto" indica que la impresora no tiene papel
13	-	SLCT	sal	La señal indica que la impresora está en modo seleccionado
14	-	$\overline{\text{AUTO}}$ FEED XT	ent	"bajo" indica que el papel avanza una línea después de imprimir.
15	-	NC	-	No se usa
16	-	OV	-	Nivel lógico de tierra
17	-	PG	-	Tierra de Chasis
18	-	NC	-	No se usa
19-30	-	GND	-	Señales de Tierras de Retorno
31	-	$\overline{\text{INIT}}$	-	Nivel "bajo" indica controlador de impresora en estado inicial y buffer vacío. Normalmente en alto. La anchura del pulso debe ser mayor a 50 us en el receptor.
32	-	$\overline{\text{ERROR}}$	sal	El nivel es "bajo" cuando la impresora está en estado PE, OFFLINE y ERROR.
33	-	GND	-	Misma función que pines 19 a 30.
34	-	NC	-	No se usa.
35	-	-	-	Conectada a 5 Volts por medio de una resistencia de 4.7 Kohms

Continuación

PIN	RET	SEÑAL	DIR	DESCRIPCION
36	-	$\overline{\text{SLCT IN}}$	ent	La entrada de datos solo es posible cuando esta señal está en "bajo". Normalmente en la fábrica se conecta a tierra con un "DIP SW".

NOTAS: 1. "DIR" se refiere a la dirección del flujo de la señal vista desde la impresora.
2. "RET" se refiere a retorno en "Twisted Pair" y se conecta a tierra.
Cuando se alambra la interfase asegúrese de utilizar un "Twisted Pair" para cada señal, sin dejar de conectar a tierra en el lado de retorno.Para prevenir ruido efectivamente estos cables deben ser aterrizados conectándose al chasis del sistema.
3. Todas las condiciones de la interfase son basadas en niveles TTL. Ambos, los tiempos de subida y de bajada deben ser menores a 0.2 us.
4. La transferencia de datos no debe ser llevada a cabo ignorando las señales $\overline{\text{ACKNLG}}$ o BUSY (Dicha transferencia debe realizarse únicamente después de confirmar cualesquiera de los dos niveles anteriores.

Figura 2.9 Conexiones de Pines y Descripciones para una interfase paralelo tipo CENTRONICS entre la IBM PC y la EPSON FX-100 (IBM Corporation).

El hecho de que existan tantas señales se debe en mucho a que por cada línea de información, existe una línea de tierra para evitar o reducir la inducción de ruido en las líneas. Estas líneas de retorno de tierra deben de unirse en común en el extremo donde se encuentra la computadora. Adicionalmente a estas líneas de tierra de señal, existen una tierra lógica y otra de chasis que deben de unirse con las anteriores en el mismo punto.

El resto de los pines cae en dos categorías:

- *Señales enviadas a la impresora para indicarle qué hacer.*
- *Señales de estatus provenientes de la impresora.*

Las principales señales de control enviadas a la impresora son:

1. *INIT , la cual le ordena a la impresora que comience su rutina de inicialización.*

2. *STROBE , la cual le indica a la impresora que puede tomar los datos que se encuentran en sus líneas respectivas para imprimirlos.*

Las principales señales provenientes de la impresora son:

1. *ACKNLG , en donde la impresora indica que ha recibido el caracter de datos y que está lista para que le envien otro.*

2. *BUSY , la cual indica que la impresora se encuentra todavía procesando la información y que es conveniente que no le envíen más.*

3. *PE , la cual indica que la impresora se ha quedado sin papel.*

4. *SLCT , que indica que la impresora se ha dado cuenta de que ha sido seleccionada.*

5. *ERROR , la cual indica que una condición de error cualquiera se ha producido.*

En la figura 2.10 se muestra un esquema de cómo se comportan estas señales.

Por principio de cuentas, la computadora revisa el estado de BUSY, cuando es bajo, indica que la impresora está lista para recibir un caracter de datos el cual ya se encuentra en las líneas respectivas. Por lo menos 0.5 us después de que los datos se encuentran en la línea respectiva, se baja el STROBE indicándole a la impresora que puede tomar su caracter.

Lo anterior origina que la señal BUSY se vuelva a elevar. Después de que la señal de STROBE se ha habilitado unos 0.5 us, se puede desactivar. Cuando la impresora ha recibido el caracter, ésta baja su ACKNLG por lo menos 5 us reseteando asimismo la señal de BUSY , indicándole a la computadora que puede enviar otro caracter.

Se puede utilizar para trabajar la señal de BUSY o la de ACKNLG indistintamente. El 3GJ utiliza además de las líneas de datos,la de ACKNLG y la de STROBE como control únicamente.

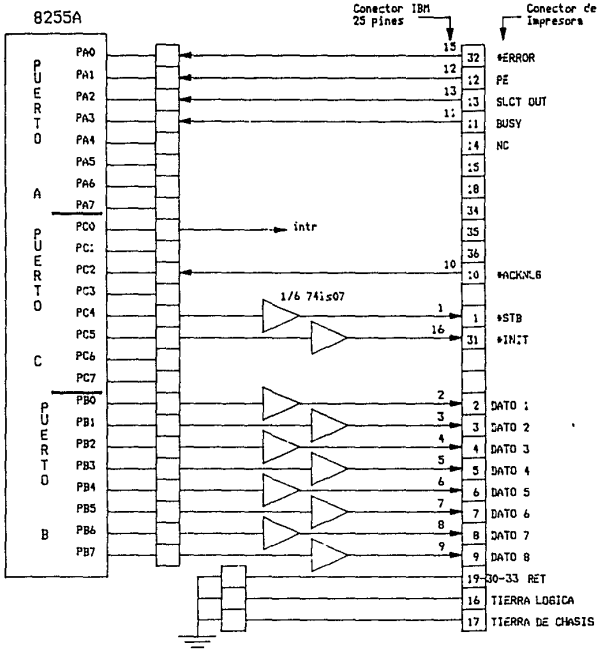


Figura 2.10 Circuito para acoplar la entrada paralelo tipo CENTRONICS de una impresora a un 8255 (Específicamente utilizado en un SDK-86).

MICROPROCESADORES Y MICROPROCESADOR 8088

Introducción

Vivimos en una sociedad con orientación computacional y constantemente somos asediados con múltiples términos relacionados con las computadoras. En este capítulo veremos algunos de estos términos y hablaremos sobre tipos de computadoras en forma general.

3.1 Computadoras

¿ Qué es una computadora ?

En la figura 3.1 vemos un diagrama de bloque de una computadora sencilla. Los dispositivos más importantes son la unidad central de procesamientos de datos o CPU, la memoria, y los dispositivos de entrada y salida o I/O. Conectando estos dispositivos se crean tres conjuntos de líneas llamadas "canales". Los tres canales son: el canal de direcciones, el canal de datos, y el canal de control.

Memoria

La sección de memoria generalmente consta de una combinación de RAM y de ROM. También puede tener discos flexibles, discos duros, o discos ópticos por laser. La memoria tiene dos objetivos. El primero es almacenar los códigos binarios que representan las instrucciones que el usuario desea ejecutar. El segundo objetivo es almacenar aquellos códigos binarios con los que la computadora va estar trabajando.

Entrada/Salida

Esta sección permite traer información del mundo exterior a la computadora y viceversa. Periféricos como impresoras, teclados, monitores, modems, etcétera son dispositivos que pertenecen a esta sección. Esto permite a la computadora y al usuario comunicarse entre si.Los actuales dispositivos físicos usados para conectar los canales de la computadora son generalmente llamados puertos. Un puerto de entrada provee información desde un teclado, un convertidor analógico/digital, ú otra fuente para ser leída dentro de la computadora bajo el control del CPU. Un puerto de salida es usado para enviar la información desde la computadora a otro periférico, tal como el monitor, la impresora, o un convertidor digital analógico. Físicamente un puerto de entrada o de salida son un conjunto de circuitos que permiten el paso de la información cuando son activados por una señal de control del CPU.

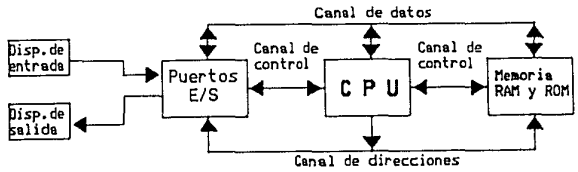


Figura 3.1 Diagrama de Bloques de una Computadora Sencilla

Unidad Central de Procesamiento

El CPU controla la operación de la computadora. El toma instrucciones en código binario de la memoria, decodifica las instrucciones en una serie de acciones simples y las lleva al cabo. El CPU contiene una unidad lógica aritmética o ALU, la cual puede ejecutar operaciones de adición, de substracción, de OR, de AND, de inversión, o de OR exclusiva en palabras binarias cuando se le ordena que lo haga. El CPU también contiene: Un contador de direcciones, el cual es usado para retener la dirección de la siguiente instrucción o dato que debe ser tomado de la memoria, un registro de

propósito general, que es usado para un almacenamiento temporal de información binaria y una que genera las señales de control de los canales.

Canales de direcciones

El canal de direcciones consiste de 16, 20, 24, o más líneas paralelas de señal. En estas líneas el CPU manda la dirección de la localidad de la memoria en la que va a escribir o de la que va a leer. El número de localidades que el CPU puede direccionar esta determinado por el número de líneas de direcciones. Si el CPU tiene N líneas de direcciones entonces puede direccionar 2 a la N potencia localidades. Cuando el CPU lee de un puerto o escribe información a un puerto, la dirección del puerto también es enviada por el canal de direcciones.

Canal de datos

El canal de datos consiste de 8, 16, 32 o mas líneas paralelas de señal. Las líneas del canal de datos son bidireccionales. Esto quiere decir que el CPU puede leer de o escribir datos a estas líneas ya sea que provengan de memoria o de un puerto. Muchos dispositivos en un sistema tendrán conectadas sus salidas al canal de datos, pero sólo una salida de un sólo dispositivo será activada a un tiempo. Cualquier salida de cualquier dispositivo conectado al canal de datos debe ser de tres estados para que se mantegan los datos cuando el dispositivo no esta en uso.

Canal de control

El CPU manda señales en el canal de control para activar las salidas de dispositivos de, memoria direccionados o puertos. Señales típicas del canal de control son las que habilitan la lectura o escritura de memoria, o de dispositivos de entrada/salida. Por ejemplo, para leer un byte de una localidad de memoria, el CPU manda la dirección del byte deseado en el canal de direcciones y manda una señal de lectura de memoria en el canal de control. La señal de ésta última activa el dispositivo direccionado, para que saque el dato al canal de datos, de donde es leído por el CPU.

propósito general, que es usado para un almacenamiento temporal de información binaria y una que genera las señales de control de los canales.

Canales de direcciones

El canal de direcciones consiste de 16, 20, 24, o más líneas paralelas de señal. En estas líneas el CPU manda la dirección de la localidad de la memoria en la que va a escribir o de la que va a leer. El número de localidades que el CPU puede direccionar esta determinado por el número de líneas de direcciones. Si el CPU tiene N líneas de direcciones entonces puede direccionar 2 a la N potencia localidades. Cuando el CPU lee de un puerto o escribe información a un puerto, la dirección del puerto también es enviada por el canal de direcciones.

Canal de datos

El canal de datos consiste de 8, 16, 32 o mas líneas paralelas de señal. Las líneas del canal de datos son bidireccionales. Esto quiere decir que el CPU puede leer de o escribir datos a estas líneas ya sea que provengan de memoria o de un puerto. Muchos dispositivos en un sistema tendrán conectadas sus salidas al canal de datos, pero sólo una salida de un sólo dispositivo será activada a un tiempo. Cualquier salida de cualquier dispositivo conectado al canal de datos debe ser de tres estados para que se mantegan los datos cuando el dispositivo no esta en uso.

Canal de control

El CPU manda señales en el canal de control para activar las salidas de dispositivos de, memoria direccionados o puertos. Señales típicas del canal de control son las que habilitan la lectura o escritura de memoria, o de dispositivos de entrada/salida. Por ejemplo, para leer un byte de una localidad de memoria, el CPU manda la dirección del byte deseado en el canal de direcciones y manda una señal de lectura de memoria en el canal de control. La señal de ésta última activa el dispositivo direccionado, para que saque el dato al canal de datos, de donde es leído por el CPU.

Hardware, Software y Firmware

Cuando se está trabajando con computadoras solemos escuchar constantemente términos como hardware, software, y firmware. Hardware es el nombre que se le da los dispositivos físicos y a la circuiteria dentro de la computadora. Software se refiere a los programas escritos para la computadora. Firmware es el término usado para los programas almacenados en ROMs o en otros dispositivos que mantienen su información aún cuando se corte la energía.

3.2 Tipos de Computadoras

Mainframes

Las computadoras vienen en una gran variedad de tamaños y capacidades. Las más grandes y poderosas son generalmente llamadas mainframes. Los Mainframes pueden llegar a ocupar una habitación entera. Son diseñados para trabajar a muy altas velocidades, con palabras de datos generalmente de 64 bits o mayores y tienen una cantidad masiva de memoria. Computadoras de este tipo tienen aplicaciones militares, empresariales (como en aseguradoras o bancos), etcétera Algunos ejemplos de estas computadoras son la IBM 4381, la Honeywell DPS8, y la Cray- X-MP/48.

Minicomputadoras

Las versiones de menor escala a la de los mainframes son comúnmente llamadas minicomputadoras. La unidad principal de un minicomputador generalmente cabe en un rack o en una caja. La minicomputadora es mucho mas lenta, trabaja con palabras mas pequeñas (generalmente de 32 bits), y no contienen tanta memoria como la de un mainframe. Computadoras de este tipo son usadas para el procesamiento de información de una empresa, control industrial, etcétera Algunos ejemplos de estas máquinas son la Digital Equipment Corp. VAX 11/780 y la Data General MV/8000II.

Microcomputadoras

Como su nombre lo implica, son computadoras de tamaño pequeño. Su rango va de pequeños controladores que trabajan directamente con palabras de 4 bits y que

pueden direccionar unos pocos miles de bytes de memoria hasta unidades más grandes que pueden trabajar con palabras de 32 bits y que pueden direccionar millones o miles de millones de bytes de memoria. Algunas de las más recientes y más poderosas microcomputadoras tiene todas o casi todas las características de un minicomputador actual. Por ello es muy difícil dibujar una línea divisoria entre estos dos tipos de computadores. Una característica distintiva de los microcomputadores es que el CPU es usualmente un solo integrado llamado microprocesador. Los microcomputadores se usan para todo, desde máquinas inteligentes de coser hasta sistemas de diseño. Algunos ejemplos de microcomputadoras son el chip controlador 8051 de intel; el SDK86 que es un kit de diseño de una sola tarjeta; la Computadora personal IBM (PC), y la computadora Apple Macintosh.

3.3 Tipos Comunes de Microprocesadores

Evolución de los microprocesadores

Una forma común de clasificar a los microprocesadores es por el número de bits que su ALU pueda manejar a un mismo tiempo. En otras palabras, un microprocesador con un ALU de 4 bits será referido como un microprocesador de 4 bits, sin importar el número de líneas de direcciones o el número de líneas de datos que tenga. El primer microprocesador que se produjo fue el 4004 de Intel en 1971. Contenía 2300 transistores PMOS. El microprocesador 4004 éra un dispositivo de 4 bits y fue creado con la intención de usarse en calculadoras. Sin embargo, algunos diseñadores se dieron cuenta que este microprocesador podía substituir tarjetas de equipos digitales llenas de circuitos lógicos. Además, este micro daba la facilidad de cambiar la funcionalidad del sistema con sólo cambiar el programa y no tener que rediseñar el hardware. Este es uno de los factores que impulsaron la evolución de los microprocesadores.

En 1972 Intel lanza al mercado un nuevo microprocesador llamado el 8008, el cual era capaz de trabajar con palabras de 8 bits. Sin embargo el 8008 requería de 20 o mas dispositivos adicionales para formar un CPU funcional. En 1974 Intel produjo el 8080, el cual tenía un conjunto de instrucciones más amplio que el del 8008 y sólo necesitaba dos dispositivos adicionales para trabajar como un CPU funcional. Además,

el 8080 usaba transistores NMOS, por lo cual trabajaba mucho más rápido que el 8008. El 8080 es conocido como un microprocesador de segunda generación.

Poco después del 8080, Motorola sacó a la venta al MC6800, otro CPU de propósito general de 8 bits. El 6800 tenía la ventaja de que únicamente utiliza una fuente de +5V, en cambio el 8080 funciona con una de +5V, -5V y 12 Volts. Por un buen tiempo estos microprocesadores (de 8 bits) encabezaron las listas de ventas. Algunos de sus competidores fueron el 6502, usados como CPU en la computadora Apple II y el Zilog Z80, usado como CPU en las computadoras TRS-80 de Radio Shack.

Los diseñadores encontraban más y más aplicaciones para los microprocesadores, y al mismo tiempo presionaban a los fabricantes a diseñar dispositivos con arquitecturas óptimas, para realizar cierto tipo de tareas. En respuesta a las necesidades expresadas, los microprocesadores han evolucionado en tres grandes rubros durante los últimos 10 años.

Controladores Dedicados

Estos dispositivos han sido usados para controlar máquinas "inteligentes", como por ejemplo hornos de micro-ondas, lavadoras de ropa, máquinas de tejido, etcétera. Texas Instruments produce millones de microprocesadores de 4 bits de la familia TMS1000 para este tipo de aplicaciones. En 1976 Intel introduce el 8048, el cual contiene un CPU de 8 bits, RAM, ROM, y algunos puertos de entrada/salida, todo en un mismo chip de 40 pines. Otros fabricantes continuaron con dispositivos semejantes. Estos son continuamente llamados microcontroladores. Algunos de ellos son, el 8051 de intel, y el MC6801 que por ejemplo, contiene contadores programables, un puerto serial (UART), así como CPU, ROM, RAM, y puertos paralelos. Un chip de estos más reciente es el Intel 8096, que contiene un CPU de 16 bits, ROM, RAM, UART, puertos, timers, y un convertidor analógico digital de 10 bits.

Procesadores de Bit-Slice

Otra de las direcciones de la evolución de los microprocesadores es la de procesadores de bit-slice. Para ciertas aplicaciones los CPUs de propósito general tal como el 8080 y el 6800, no son lo suficientemente rápidos o su conjunto de instrucciones son insuficientes. Para estas aplicaciones algunos fabricantes producen

dispositivos que pueden ser usados para un CPU (conocido como custom CPU). Un ejemplo de éstos, es la familia de dispositivos Advanced Micro Devices 2900. Esta familia incluye ALUs de 4 bits, multiplexores, secuenciadores, y otras partes que se necesitan para construir un CPU a la medida. El término de slice (rebanada) viene del hecho de que estas partes se pueden conectar en paralelo para trabajar con palabras de 8 bits, 16 bits, o de 32 bits. En otras palabras, un diseñador puede agregar tantas rebanadas como se necesiten para una aplicación en especial. El diseñador no sólo diseña el hardware, si no que también crea su conjunto de instrucciones por medio de microcódigos.

CPUs de propósito general

La tercera y más grande parte en la evolución de los microprocesadores es la de los CPUs de propósito general, que le han dado a las microcomputadoras casi todo el poder de las minicomputadoras anteriores. Después de que Motorola introduce el MC6800, Intel produce el 8085, una actualización del 8080, el cual sólo requiere +5 volts de alimentación. Después Motorola produce el MC6809 que tiene algunas instrucciones de 16 bits, pero es todavía básicamente un microprocesador de 8 bits. En 1978 Intel fabrica el 8086 que ya es un procesador de 16 bits. Algunos procesadores de 16 bits ya existían a la venta como el National Pace y el 9900 de Texas Instruments, pero aparentemente el mercado todavía no estaba listo. Después de que Intel produjo el 8086, Motorola lanza el MC68000 de 16 bits. El 8086 y el 68000 trabajan directamente con palabras de 16 bits en vez de 8, pueden direccionar un millón o más bytes de memoria, y pueden ejecutar instrucciones mucho más rápido que procesadores de 8 bits. También estos procesadores tienen instrucciones únicas para funciones que, en un procesador de 8 bits implicarían varias instrucciones

La evolución en este camino continuó hasta procesadores de 32 bits, que pueden trabajar con giga o tera bytes de memoria. Algunos ejemplos de estos son el 80386 de Intel, el 68020, de Motorola, y el 32032 de National.

En el apéndice se muestran las características de varios microprocesadores. En base a nuestras necesidades, fue seleccionado el microprocesador de Intel IAPX-8088 debido a las siguientes características que para nosotros fueron de interés:

+ ***Es capaz de direccionar hasta 1 Mbyte de memoria.*** *Esta característica permite*

que el circuito 3GJ posea un espacio suficientemente grande para poder almacenar una gran cantidad de información proveniente de las PC's y así liberar cuanto antes a las microcomputadoras.

+ ***Puede trabajar hasta 8 MHz.*** *Lo que resulta extremadamente ventajoso. La velocidad de proceso de los datos recibidos redunda en una mayor transparencia para los usuarios de la red 3GJ.*

+ ***Existe una gran cantidad de software de apoyo, tanto para la programación como para monitoreo (debugs).*** *Actualmente se han desarrollado diversos paquetes de apoyo a la programación del microprocesador 8088 tales como ensambladores y debugs principalmente. Teniendo en cuenta que estos últimos son la herramienta principal en la detección de errores de programación, resulta bastante útil que en el mercado se pueda hallar este tipo de ayuda.*

+ ***Es el microprocesador en el que están basadas casi todas las microcomputadoras personales (PC) IBM y compatibles,*** *por lo que el probar programas se convierte en una tarea más amigable. Los programas de prueba pueden realizarse en una PC que se encuentra en casi todas partes.*

+ ***Sus periféricos se consiguen con facilidad dentro del mercado nacional.*** *Esto constituye una gran ventaja debido a que los costos de mantenimiento y reparación del aparato disminuyen considerablemente. Además los tiempos de adquisición de las partes resultan menores.*

Debido a que nuestra tesis se basa en el microprocesador 8088 de Intel, en el siguiente capítulo vamos a describir la operación y programación de la familia 8086, 8088, 80186, 80188, 80286.

3.4 Los Microprocesadores 8086, 8088, 80186, 80188, y el 80286

El 8086 de Intel es un microprocesador de 16 bits diseñado para ser usado como CPU de una microcomputadora. El término "16 bits" quiere decir que ALU, registros internos, y la mayoría de sus instrucciones son diseñadas para trabajar con palabras

(de 16 bits). El 8086 tiene un canal de datos de 16 bits, así que puede leer o escribir a memoria y a puertos 16 u 8 bits. También posee un canal de direcciones de 20 bits, así que puede direccionar 1,048,576 localidades de memoria. Cada una de las direcciones de memoria representa una localidad de uno o dos bytes. Las palabras serán guardadas en dos localidades consecutivas. Si el primer byte de una palabra está en una dirección par, el 8086 puede leer la palabra completa en una sola operación. Si el primer byte de la palabra está en una dirección impar, el 8086 leerá el primer byte en una operación y el segundo byte en otra operación.

El 8088 de Intel tiene el mismo ALU, los mismos registros, y el mismo conjunto de instrucciones del 8086. El 8088 tiene un canal de direcciones de 20 bits, así que puede direccionar hasta 1,048,576 localidades de memoria. Sin embargo el 8088, tiene un canal de datos de 8 bits, así que sólo puede leer o escribir de memoria y de puertos 8 bits a la vez. Para leer una palabra de 16 bits de dos localidades sucesivas de memoria el 8088 siempre tiene realizar dos operaciones de lectura. El 8088 es en paréntesis, el CPU de la computadora personal IBM XT y de otras computadoras personales.

El 80186 es una versión mejorada del 8086, y el 80188 del 8088. El 80188 y el 80186 además de tener un CPU de 16 bits, tienen dispositivos periféricos integrados. El conjunto de instrucciones del 80188 y el 80186 es más amplio que el del 8088 y 8086. Adicionalmente cabe mencionar que programas hechos para el 8086 y el 8088 son 100% compatibles para un 80188 y un 80186, aunque por tener instrucciones adicionales los programas hechos para 80188 o 80186 no necesariamente corren en 8086 o en 8088.

El 80286 de Intel es una versión avanzada del 8086, especificamente diseñado para ser usado como CPU de una sistema multiusuario o multitarea. Programas escritos para 8086, pueden correr en 80286 en su modo real de direccionamiento. El 80286 es el CPU de las computadoras AT.

Arquitectura Interna del 8088

Como se puede observar en la figura 3.2, el CPU del 8088 está separado en dos partes funcionales, la unidad del canal de interface o BIU, y la unidad de ejecución(EU). Dividiendo el trabajo entre estas dos unidades se acelera el

procesamiento.

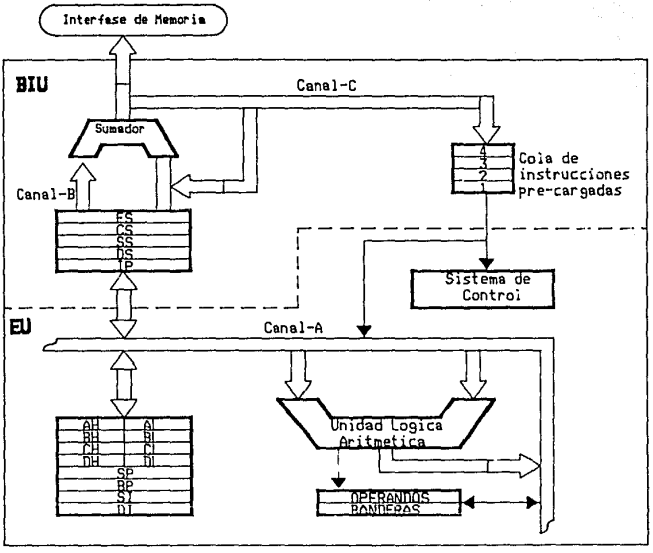


Figura 3.2 Arquitectura Interna del 8088

La Unidad del Canal de Interface

El BIU manda hacia el exterior las direcciones, captura las instrucciones de memoria, lee información de los puertos y de la memoria. En otras palabras, el BIU maneja todas las transferencias de datos y direcciones de los canales, para la unidad de ejecución. A continuación describiremos las diferentes partes del BIU.

La Cola (Queue)

Para acelerar la ejecución del programa, el BIU captura por adelantado hasta 6 bytes de instrucción de memoria. Los bytes de instrucción capturados por adelantado, son guardados para el EU en un primer grupo de registros de entrada-salida, llamados Queue. El BIU puede estar capturando bytes de instrucción, mientras que el EU está decodificando o ejecutando una instrucción, lo cual no requiere el uso de los canales. Cuando el EU está listo para la siguiente instrucción, simplememte la lee del queue en el BIU. Esto es mucho más rápido que enviar la dirección al sistema de memoria y esperar a que la memoria nos envíe el byte o bytes de regreso. Exceptuando los casos de las instrucciones de JUMP y CALL, en donde el queue debe ser vaciado y después recargado empezando en una nueva dirección, este plan de precaptura y de cola acelera considerablemente el proceso. Capturar la siguiente instrucción, mientras la presente se está ejecutando se llama entubar (pipelining).

Registros de Segmento

El BIU contiene cuatro registros de segmento de 16 bits. Estos son: el registro de segmento de código (CS), el registro de segmento de pila (SS), el registro de segmento extra (ES), y el registro de segmento de datos (DS)). Estos registros de segmento, son usados para guardar los 16 bits superiores, de las direcciones de inicio de 4 segmentos de memoria, con las que el 8088 va a trabajar en un tiempo determinado. EL BIU del 8088 envía 20 bits de direcciones, o sea que puede direccionar 1,048,576 bytes de memoria. Sin embargo, en cualquier momento el 8088 sólo trabaja con 4 segmentos de 64 Kbytes dentro del rango de 1Mbyte. En la figura

3.3, se muestra cómo estos cuatro segmentos podrían estar posicionados en memoria en un tiempo determinado. Los cuatro segmentos pueden estar separados como se indica, o para programas muy pequeños que no requieren los 64 Kbytes en cada segmento, éstos pueden traslaparse. Un sistema mínimo, por ejemplo, podría empezar con sus 4 segmentos en la misma dirección 00000H.

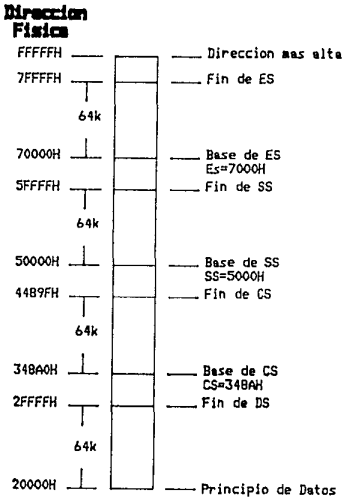


Figura 3.3 Una forma de distribuir cuatro segmentos de 64KBytes en el espacio de direccionamiento del 8088, 1MByte.

Repitiendo de nuevo, un registro de segmento, es usado para contener los 16 bits superiores de las direcciones de inicio para cada uno de los registros. Por ejemplo, el registro de segmento de código, guarda los 16 bits superiores de la dirección de inicio del segmento de donde el BIU está capturando los bytes de instrucciones de código. El BIU siempre inserta ceros para los cuatro bits menos significativos (nibble) de los 20 bits de direcciones de un segmento. Por ejemplo, si el registro de segmento de código contiene un 384AH, entonces el segmento de código empezaría en la dirección 384A0H. En otras palabras, un segmento de 64Kbytes puede ser posicionado en cualquier parte dentro del espacio de direccionamiento de 1 Mbyte, pero el segmento siempre empezará en una dirección con ceros en sus 4 bits menos significativos. La parte de un segmento de inicio de dirección almacenada en un registro de segmento es comúnmente llamada "base segment", (segmento base).

Una STACK (pila) es una sección de memoria puesta para almacenar direcciones y datos mientras que un subprograma se ejecuta.

El segmento extra y el segmento de datos son usados para guardar los 16 bits superiores de la direcciones de inicio de dos segmentos de memoria que son usados para datos.

Apuntador de Instrucciones

La siguiente parte que veremos del BIU es el registro apuntador de instrucciones (IP). Como ya habíamos mencionado el registro de segmento de código guarda los 16 bits superiores de la dirección de inicio del segmento, de donde el BIU está capturando los bytes de código de las instrucciones. El registro IP contiene los 16 bits de direcciones del siguiente byte de código dentro de este segmento de código. El valor contenido en el IP es referido como el "offset", ya que este valor tiene que ser sumado al valor del CS para crear una dirección física de 20 bits. En la figura 3.4 podemos observar su funcionamiento. El registro CS apunta a la base o principio del segmento de código presente. El IP contiene la distancia u offset de está dirección base del siguiente byte de instrucción que va a ser capturado. La figura 3.4 nos muestra cómo el offset de 16 bits es adicionado al CS para darnos como resultado una dirección física de 20 bits. Cabe hacer incapié en que los dos números de 16 bits no son sumados directamente. Un modo de explicar esto, es decir que el contenido del registro CS sufre un corrimiento hacia la izquierda de 4 bits, antes de ser sumados al IP.

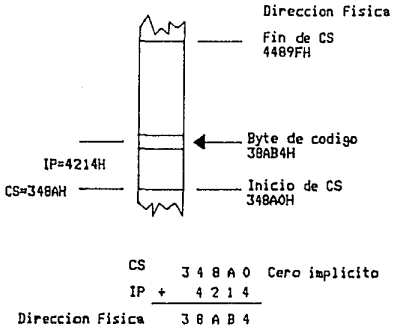


Figura 3.4 Obtención de la dirección física

La dirección física del 8088 es representada muchas veces en la forma" base:offset" en vez de un sólo número.Por ejemplo, la dirección física 38AB4H se representaría

como 348A:4214.

La Unidad de Ejecución (EU)

La unidad de ejecución del 8088 le indica al BIU dónde debe capturar las instrucciones o datos, decodifica las instrucciones, y ejecuta las instrucciones. A continuación veremos las partes funcionales de la unidad de ejecución.

El EU contiene circuitería de control, que es la que dirige las operaciones internas. Un decodificador en el EU, traduce las instrucciones capturadas de memoria, en una serie de acciones que lleva al cabo. El EU tiene una unidad aritmética lógica de 16 bits que puede ejecutar operaciones de suma, resta, incremento, decremento, complemento, cambio, y lógicas como AND, OR, o XOR,

Registro de banderas

Una bandera es un flip flop que indica ciertas condiciones producidas por la ejecucuión de una instrucción, o controla operaciones del EU. Un registro de banderas de 16 bits en el EU, contiene 9 banderas activas. La figura 3.5 nos muestra la

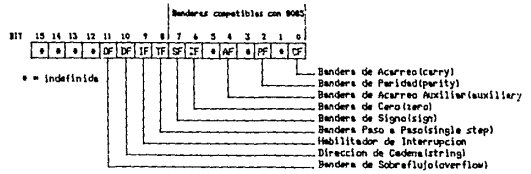


Figura 3.5 Registro de Banderas del 8088

localización de las nueve banderas en el registro . Seis de las nueve banderas son utilizadas para indicar los resultados producidos al ejecutarse la instrucción. Por ejemplo, un flip-flop llamado bandera de acarreo será puesto en uno, si la suma de

dos números binarios de 16 bits produce un acarreo del bit más significativo. Si hay acarreo del bit más significativo en la suma, entonces la bandera estará en cero.

Las seis banderas condicionales de este grupo son: la bandera de acarreo (CF), la bandera de paridad (PF), la bandera auxiliar de acarreo (AF), la bandera de cero (ZF), la bandera de signo (SF), y la bandera de sobre-flujo (OF). Algunas instrucciones del 8088 checan estas banderas para ver cual de las dos alternativas de acción se debe hacer en la ejecución de una instrucción.

Las tres banderas restantes del registro de banderas son usadas para controlar ciertas operaciones del procesador. Estas banderas son diferentes de las seis banderas condicionales en la forma en que son activadas y reseteadas. Las seis banderas condicionales son encendidas o reseteadas por EU en las bases de un resultado de una operación aritmética o lógica. Las banderas de control son deliberadamente encendidas o reseteadas con instrucciones específicas puestas en el programa. Las tres banderas de control son la bandera de "Trap" (TF), que se usa para correr un programa paso a paso; la bandera de interrupción (IF), que se usa para permitir o no permitir la interrupción de un programa, y la bandera de dirección (DF), que es usada con instrucciones de string.

Registros de Propósito General

El EU tiene 8 registros de propósito general etiquetados como: AH, AL, BH, BL, CH, CL, DH, Y DL. Estos registros se pueden usar individualmente para un almacenamiento temporal de 8 bits. El registro AL es también llamado acumulador. Este registro tiene ciertas características que los otros registros no tienen.

Ciertos pares de estos registros pueden usarse para almacenar palabras de 16 bits. Los registros pares aceptados son AH y AL, BH y BL, CL y CH, y DL y DH. El par AH-AL es referido como el registro AX, el par BH-BL como BX, CH-CL como CX y el par DH-DL como el registro DX. Para operaciones de 16 bits el registro AX es llamado acumulador.

El juego de registros del 8088 es muy parecido al de los microprocesadores 8080 y 8085. Fue diseñado así para que los programas hechos para 8080 y 8085 fueran fácilmente traducidos, para correr en 8086 ú 8088. La ventaja de usar registros internos

para el almacenamiento temporal de información, es que como la información ya esta dentro del EU, puede ser mucho más rápidamente accesada que si se tuviera que accesar desde memoria.

Registro Apuntador de Pila (SP)

Un Stack (pila) es una sección de memoria puesta para almacenar direcciones y datos mientras que una subrutina se está ejecutando. El 8088 permite poner un segmento independiente de stack de 64 Kbytes. Los primeros 16 bits de las direcciones de inicio son guardadas por el registro de segmento de pila. El registro de SP contiene el offset de 16 bits, que va desde el principio del segmento a la localidad de memoria donde la última palabra fue almacenada recientemente en el Stack. La localidad de memoria donde fue almacenada la última palabra recibe el nombre de tope del stack. Esto lo podemos observar en la siguiente figura 3.6.

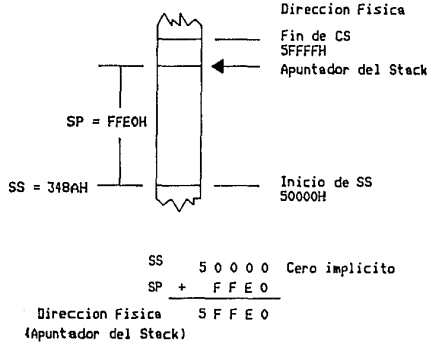


Figura 3.6 Obtención de una dirección física en el Stack

La dirección física, para una lectura o escritura del stack, se produce de la suma de los contenidos del SP y el registro de segmento de stack(SS). Para hacer esto los contenidos del SS sufren un corrimiento de 4bits hacia la izquierda y el contenido del SP se suma en este momento. En la figura 3.6 podemos observar ésta operación.

Otros Apuntadores y Registros de Indices

Además del registro apuntador del stack, SP, el EU contiene un registro apuntador de base de 16 bits, BP. También contiene un registro de origen de índice de 16 bits (SI), y un registro de destino de índice de 16 bits (DI). Estos tres registros pueden ser usados para almacenamiento temporal de datos,igual que los registros de propósito general

descritos anteriormente. Sin embargo, su uso principal es contener el offset de 16 bits de una palabra de datos en uno de los segmentos. El SI, por ejemplo, puede ser usado para guardar el offset de una palabra de datos en el segmento de datos. La dirección física del dato en memoria sería generada en este caso haciendo un corrimiento de 4 bits del registro de segmento de datos y sumando el resultado al SI.

3.5 Programación del 8088

Lenguajes de programación

Para correr un programa, una microcomputadora debe tener el programa almacenado en forma binaria en localidades sucesivas de memoria. Hay tres niveles de lenguaje que se pueden usar para escribir un programa en una microcomputadora.

Lenguaje de Máquina

Uno puede escribir programas tan simples como una sequencia de códigos binarios en vez de las instrucciones, para que la microcomputadora los ejecute. Esta forma binaria del programa es referida como lenguaje máquina porque es la forma requerida por la máquina. Sin embargo, es muy difícil, si no es que imposible, para un programador memorizar los miles de códigos en binario de las instrucciones para CPU como el 8088. Además es muy fácil cometer un error si se está trabajando con series muy grandes de 1's y de 0's. Usar una representación hexadecimal para estos códigos binarios puede ayudar, pero aún así, son miles de códigos de instrucciones como para hacerles frente.

Lenguaje Ensamblador

Para hacer la programación más sencilla muchos programadores usan un programa ensamblador. Lo que éste hace es traducir el programa que está en ensamblador a lenguaje máquina para que pueda ser cargado en memoria y después ejecutado. El lenguaje en ensamblador usa dos, tres, o cuatro mnemónicos para representar cada tipo de instrucción. El mnemónico, es sólo un dispositivo para ayudarnos a recordar algo. Las letras en lenguaje ensamblador, mnemónicos, son básicamente iniciales o

formas abreviadas de las palabras en inglés para la operación ejecutada por el microprocesador. Por ejemplo, el mnemónico para restar (SUBSTRACT) es SUB, el mnemónico para "or exclusiva" es XOR, y el mnemónico para la instrucción copiar un dato de una loacalidad a otra es MOV (MOVE).

El formato de presentación del programa del lenguaje ensamblador es escrito usualmente en una forma estándar, teniendo 4 campos. La figura 3.7 muestra el formato de presentación del lenguaje ensamblador con sus cuatro campos.

Campo de Etiqueta	Campo de Código	Campo de Operandos	Campo de Comentarios
OTRO:	ADD	AL,07H	; Factor de corrección

Figura 3.7 Formato de un enunciado en lenguaje ensamblador

El primer campo es la etiqueta, que es un símbolo o un grupo de símbolos usados para representar una dirección que no es específicamente conocida en el tiempo que el enunciado es escrito. Las etiquetas son generalmente seguidas de dos puntos. Las etiquetas no tienen que estar presentes en el enunciado siempre, sólo son insertadas cuando son necesarias.

El segundo campo es el código de operación el cual contiene el mnemónico de la instrucción que va a ser ejecutada. Los mnemónicos de las instrucciones son a veces llamados códigos de operación (operation codes ú op. code). En la figura 3.7 vemos que el mnemónico es ADD, el cual significa que queremos hacer una suma.

El tercer campo es el campo de operandos, y este contiene datos, direcciones de memoria, direcciones de puertos, o el nombre del registro en donde la instrucción se vaya a ejecutar. Operando es sólo otro nombre del ente de datos, afectado por la instrucción. En el ejemplo de la figura 3.7 sólo hay dos operandos, AL y 07H. AL representa al registro acumulador y el 07H representa el número 07 en hexadecimal. Este enunciado en lenguaje ensamblador dice, suma el número 07H al contenido del

registro del acumulador. Por convención de Intel el resultado será puesto en el registro o en la localidad de memoria especificada antes de la coma en el campo de operando. Para el caso de nuestro ejemplo el resultado será puesto en el registro AL. Otro ejemplo podría ser el enunciado ADD BH,AL, que convirtiendolo a lenguaje máquina y ejecutandolo, sumaría el contenido del registro AL al contenido del registro BH. El resultado se quedaría en el registro BH.

El cuarto campo es el campo del comentario, que empieza con un punto y coma. Este campo es muy importante. El comentario no forma parte del programa de la máquina. Se escriben comentarios en un programa para recordar lo que la instrucción o grupo de instrucciones realizan en el programa.

Para resumir del porqué usamos lenguaje ensamblador, veamos más de cerca la instrucción ADD. El formato general de la instrucción ADD del 8088 es:

ADD destino,origen

El origen o fuente puede ser un número escrito en la instrucción, el contenido de un registro en específico, o el contenido de una localidad de memoria. El destino puede ser un registro en específico o una cierta localidad de memoria. Sin embargo, el origen y el destino no pueden ser ambos la localidad de memoria en una instrucción. Más adelante en el tema de modos de direccionamiento veremos todos los caminos en los que el origen de un operando y el destino de un resultado pueden ser especificados.El punto aquí es que sólo el mnemónico, ADD, en conjunto con un origen específico y un destino determinado pueden representar una gran cantidad de instrucciones del 8088 en una forma muy sencilla.

Puede surgir la siguiente pregunta: ¿Cómo puedo traducir mi programa en ensamblador a lenguaje máquina? Existen dos métodos para esto. El primer método de hacer la traducción es encontrando el código binario para cada instrucción en las tablas del libro del fabricante. El segundo método es usar un programa ensamblador. Un programa ensamblador es aquél que puede ser corrido en una computadora. Lee las instrucciones del lenguaje ensamblador y genera su código binario para cada una. Para elaborar programas en lenguaje ensamblador de una forma muy sencilla, el ensamblador y otros programas son necesarios (debug, link, code-view).

Lenguajes de Alto Nivel

Otra forma de escribir un programa para una microcomputadora es en un lenguaje de alto nivel como son el Basic, Fortran, o el Pascal. Estos lenguajes usan enunciados que son aún más parecidos al idioma inglés que los del lenguaje ensamblador. Cada enunciado de alto nivel puede representar muchas instrucciones de código de máquina. Un programa interprete o compilador es usado para hacer la traducción de lenguajes de alto nivel a códigos de máquina. Un programa se puede escribir más rápidamente en un lenguaje de alto nivel que en lenguaje ensamblador porque el lenguaje de alto nivel trabaja con bloques de construcción más grandes. Sin embargo programas escritos en lenguajes de alto nivel compilados corren mas lentamente que programas hechos en lenguaje ensamblador. Programas que envuelven hardware de control, tales como robots y sistemas de control en fábricas o programas que tienen que correr tan rápido como sea posible generalmente es preferible que sean escritos en lenguaje ensamblador. Lo mismo sucede con programas que manejan cantidades masivas de información es mejor que sean escritos en lenguajes de alto nivel. La decisión de que tipo de lenguaje usar es cada vez más difícil, ya que ensambladores recientes permiten usar muchas de las características de un lenguaje de alto nivel, y además algunos lenguajes de alto nivel usan características de los lenguajes ensamblador.

Modos de Direccionamiento Inmediato y por Registro

Antes de poder ver diferentes técnicas de programación en lenguaje ensamblador del 8088, es necesario ver los diferentes modos en el que el 8088 puede accesar datos con los que va a trabajar. A las diferentes formas en que el 8088 puede accesar información recibe el nombre de modos de direccionamiento. En los enunciados del lenguaje ensamblador el modo de direccionamiento es indicado en la instrucción. Usaremos la instrucción MOV para ilustrar los diferentes modos de direccionamiento.

La instrucción MOV tiene el siguiente formato:

MOV destino,origen

Cuando se ejecuta, esta instrucción copia una palabra o un byte desde la localidad

de origen a la localidad destino. El origen puede ser un número escrito directamente en la instrucción, un registro específico, o una localidad de memoria especificada de una o 24 diferentes maneras. El destino puede ser un registro o una localidad de memoria determinada de una o 24 maneras. El origen y el destino no pueden ser ambos localidades de memoria en una instrucción.

Modo Inmediato de Direccionamiento

Supongamos que en un programa necesitamos el número 437BH en el registro CX. La instrucción MOV CX,437BH se puede usar para lograr esto. Cuando se ejecuta, esta instrucción pondrá el valor numérico inmediato 437BH en el registro CX de 16 bits. Esto es conocido como modo de direccionamiento inmediato porque el número que va a ser puesto en el registro CX será puesto en dos localidades de memoria inmediatamente después de el código de la instrucción MOV.

Una instrucción similar, MOV CL,48H puede ser usada para cargar el número de 8 bits 48H en el registro de 8 bits CL. Uno también puede dar instrucciones para cargar este número de 8 bits en una localidad de memoria o un número de 16 bits en dos localidades consecutivas.

Modo de Direccionamiento por Registro

Modo de direccionamiento por registro significa que un registro es el fuente de un operando para una instrucción. La instrucción MOV CX,AX, por ejemplo, copia el contenido del registro de 16 bits al registro de 16 bits CX. Hay que recordar que la localización del destino es especificada en la instrucción antes del origen. También hay que hacer notar que los contenidos del registro AX son copiados al registro CX, no son realmente movidos. En otras palabras, el contenido del registro CX es sobre escrito, y el contenido de AX se mantiene intacto. Se puede copiar el contenido de cualquier registro de 16 bits a otro registro de 16 bits, o el contenido de un registro de 8 bits a otro de 8bits. Sin embargo, no se puede usar una instrucción como MOV CX,AL porque esto es un intento de copiar un operando de un byte (AL) a un operando destino de una palabra (CX). El byte en AL cabe en CX, pero el 8088 no sabría en que mitad de CX poner ese byte. Si tratamos de usar esta instrucción y estamos usando un buen ensamblador, el ensamblador nos dirá que tenemos un error. Para copiar un byte de AL a CX se puede usar la siguiente instrucción MOV CL,AL.

Esta instrucción copiaría el byte de AL al byte menos significativo de CX.

Acceso de Datos en Memoria

El modo de direccionamiento que a continuación se describe son usados para especificar la localidad de un operando en memoria. Ya hablamos comentado como el 8088 creaba la dirección física para los códigos de instrucciones sumando el offset en el registro apuntador de instrucciones (IP) a la base del segmento de código en el registro CS. Hay que recordar que el contenido de CS sufría un corrimiento de 4 bits hacia la izquierda antes de ser sumado al IP. También mencionamos como el 8088 accesaba localidades de la pila (STACK) sumando el offset en el registro apuntador de pila a la base del segmento de pila en el registro SS. Aquí de nuevo el contenido del registro SS sufre un corrimiento hacia la izquierda de 4 bits antes de ser sumado al SP.

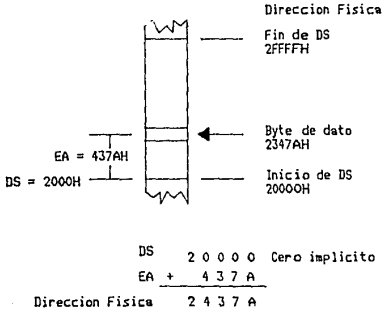


Figura 3.8 Obtención de la dirección física de un dato

Para accesar datos de memoria, el 8088 tiene que crear una dirección física de 20 bits. Logra esto sumando un valor de 16 bits llamado dirección efectiva a una de las bases de los cuatro segmentos. La dirección efectiva (EA) representa un desplazamiento ú offset de el operando deseado de la base del segmento. En la mayoría de los casos, cualquiera de las bases de segmento puede ser especificada, pero el segmento de datos es el más usado. En la figura 3.8 podemos observar cómo la dirección física de 20 bits es generada por el BIU. La dirección de inicio del segmento de datos en la figura es 2000H, así que el registro de segmento de datos contendrá un 2000H. El BIU hace el corrimiento de los 4 bits hacia la izquierda y suma la dirección efectiva, 437AH, al resultado. La dirección física enviada a la memoria por el BIU sería 2437AH.

La unidad de ejecución calcúla la dirección efectiva para un operando usando información que uno especifica en la instrucción. Uno puede decirle al EU que use un número en la instrucción como la dirección efectiva, que use los contenidos de un registro en específico como la dirección efectiva, o que calcule la dirección efectiva agregando un número o dos al contenido de registros específicos.

Modo Directo de Direccionamiento

El modo mas sencillo de direccionamiento de memoria es aquél en el que la dirección efectiva es un número de 8 bits o de 16 bits, escrito en la instrucción. La instrucción MOV CL,[437AH] es un ejemplo. Los corchetes cuadrados que tiene el número son para indicar que el número que se va a cargar en CL es el contenido del desplazamiento 437AH a partir de la base del segmento de datos. La dirección física actual de 20 bits, se producirá del corrimiento de 4 bits hacia la izquierda de la base del segmento de datos DS y sumándole a éste la dirección efectiva 437AH. En la figura 3.8 nos muestra cómo se hace esta operación. Este modo de direccionamiento se le llama directo porque el desplazamiento del operando con respecto a la base del segmento, es especificado directamente en la instrucción. El desplazamiento en la instrucción sería agregado a la base del segmento de datos a menos que se use el prefijo "de hacer a un lado" (override) para decirle al BIU que agregue el desplazamiento a otra base de segmento.

Otro ejemplo del modo de direccionamiento directo es la instrucción MOV BX,[437AH]. Cuando se ejecuta, esta instrucción copia una palabra desde memoria al registro BX. Como cada dirección de memoria del 8088 representa un byte de almacenamiento, la palabra debe ser de dos localidades de memoria. El byte contenido en el desplazamiento de 437AH con respecto a la base del segmento de datos, será copiado en el registro BL. El contenido la dirección más alta, desplazamiento 437BH, será copiado en el registro BH. El 8088 accesará en forma automática el número requerido de bytes de memoria para una instrucción dada.

Los dos ejemplos anteriores nos muestran como el modo directo de direccionamiento puede ser usado para especificar el origen de un operando. La instrucción MOV [437AH],BX por ejemplo, va a copiar el contenido del registro BX en las dos localidades de memoria del segmento de datos. El contenido de BL será copiado en

la localidad de memoria con desplazamiento 437AH. El contenido de BH será copiado a la localidad de memoria con desplazamiento de 437BH.

Segmentación

Uno podría preguntarse porqué Intel diseñó la familia 8088 para que use memoria segmentada. Una primera razón es que trabajando con segmentos de memoria de sólo 64 Kbytes a un tiempo, el 8088 únicamente tiene que operar con una dirección efectiva de 16 bits para accesar cualquier localidad del segmento. En otras palabras, debido a la segmentación el 8088 sólo tiene que manipular y almacenar componentes de 16 bits de dirección. Una segunda razón tiene que ver con el tipo de microcomputadora en la que la familia 8088 (CPU) se va a usar. En un sistema de tiempo compartido, muchos usuarios comparten el CPU. El CPU trabaja en el programa de un usuario aproximadamente por 20 milisegundos. Después de haber trabajado por 20 milisegundos por cada uno de los usuarios, el CPU regresa a trabajar con el programa del primer usuario. Cada vez que el CPU cambia de un programa de un usuario a otro, tiene que accesar una nueva sección de código y una nueva sección de datos. La segmentación hace el cambio mucho mas fácil. A cada programa de usuario se le puede asignar un conjunto lógico de segmentos separados para sus códigos y sus datos. El programa de cada usuario va contener desplazamientos u offsets a partir de estos segmentos base. Para cambiar del programa de un usuario al un segundo usuario todo lo que se tiene que hacer es recargar los cuatro registros de segmento con las direcciones de los segmentos base asignados al programa del segundo usuario. En otras palabras, la segmentación convierte en algo fácil el mantener separados los programas y datos de cada usuario. La segmentación también facilita el cambio de un programa de un usuario a otro.

3.6 Escribiendo Programas en Ensamblador

Instrucciones de Inicialización

Antes de escribir un programa es necesario determinar una serie de instrucciones adicionales. El objetivo de estas instrucciones adicionales es el inicializar varias partes del sistema tales como los registros de segmento, banderas, y dispositivos de puertos programables.

Por ejemplo, si se está usando el stack en un programa, entonces se debe de incluir una instrucción que cargue el registro del apuntador del stack con el offset del tope del stack. La mayoría de los sistemas de microcomputadoras contienen una serie de periféricos programables tales como los puertos, temporizadores, y controladores. Entonces se deben incluir instrucciones que manden palabras de control a estos dispositivos para decirles la función que uno desea que ejerzan.

El mejor modo de hacer la inicialización es por medio de una lista que contenga todos los registros, dispositivos programables, y banderas de un sistema en el que se esté trabajando. Una lista de inicialización de un sistema basado en el 8088 sería de la siguiente forma :

Lista de Inicialización

Registro de segmento de datos
Registro de segmento de pila
Registro de segmento extra
Registro de apuntador de pila
Registro de apuntador de base
Registro de índice de origen
Registro de índice de destino
Puerto programable 8255
Controlador de prioridades de interrupciones 8259A
Contador programable 8254
Puerto serial programable 8251A
Inicializar variables de datos
Bandera de dirección Reset/Clear y bandera de habilitación de interrupciones.

Cabe hacer notar que el registro de segmentos de código no está presente en la lista. El registro de segmento de código es cargado con el valor de inicio correcto por el comando del sistema que uno usa para correr el programa.

Para ampliar la información de algunos microprocesadores, en la sección de apéndices se encuentran las características de ellos.

3.7 Periféricos

El USART 8251

El puerto serie que utiliza el 3GJ es el Chip "Universal Synchronous Asynchronous Receiver Transmiter" USART 8251 de Intel. Este circuito como lo anuncia el fabricante puede manejar velocidades de transmisión hasta de 56 kbps en modo síncrono y 9.6 kbps en modo síncrono. Puede detectar errores de paridad, overrun y encuadramiento, es completamente compatible con el 8088, viene en un empaque de 28 pines, es completamente compatible con TTL y tiene una alimentación de 5V.

Este circuito posee un pin CS para poder seleccionarlo, otra de RESET que se conecta al 8284 para resetearlo al mismo tiempo que el microprocesador, una pata de CLOCK la cual en el 3GJ se conecta al PCLOCK del 8284. Posee también otras patas de control que le permiten saber al circuito si los datos en paralelo que entran o salen de su buffer son propiamente datos o caracteres de estatus o de control. Este circuito posee unas patas cuya aplicación bien puede ser de conexión hacia un modem . Dichas patas son el DSR, DTR, CTS y RTS . El 3GJ no utiliza estas patas para controlar la comunicación ya que lo hace de otra manera.Obviamente posee un par de pines por los que puede transmitir y recibir y unos pines que permiten conocer el estatus de la transmisión y recepción. Este estatus se puede conocer también leyendo el registro apropiado en el chip.

Si se quiere transmitir, primero se debe de preguntar por la pata o el bit TXE (a veces es posible omitir este paso), a continuación,un carácter llegará en forma paralela, el carácter se convertirá de forma paralela a serie y podrá transmitirse cuando lo indique el pin o bit TXRDY. Se puede saber cuando el carácter ha sido transmitido preguntando por el pin o bit TXE.

Al recibirse por completo un carácter, se podrá saber si se pregunta por la señal o bit RXRDY. En ese momento el microprocesador podrá tomar su carácter. Si no lo hace así y llega otro carácter, se encimará con el primero y se producirá un error de overrun. Es importante mencionar que la señal o bit RXRDY no se desactiva sino hasta que el microprocesador no haga una lectura de ese puerto y tome el carácter.

Para programarse, requiere dos palabras : una de modo y otra de comando. Pueden programarse varias opciones como es el factor por el que debe dividir el chip el reloj de transmisión y recepción. Es importante notar que para velocidades menores a 9600 es conveniente utilizar el factor 64X en tanto que a 9600 lo ideal es usar el factor 16X ya que si se maneja el factor 64X, la frecuencia del reloj de transmisión/recepción es tan alta que sobrepasa las capacidades del circuito MOS que es el 8251 y entonces funciona mal el circuito. También pueden programarse longitudes de carácter de 5, 6, 7 y 8 bits, habilitar o no la paridad que se desee,el número de stop bits, habilitar o no la transmisión (para lo cual también es muy importante que el CTS del chip se habilite en bajo), habilitar o no la recepción, manejar el nivel de algunas de las patas de propósito general para manejo de modems del chip como el DTR y RTS, inclusive, también es posible resetear al circuito enviando un bit en la palabra de comando.

En la palabra de estatus leida por software puede saberse el estado de la pata DSR, TXRDY, TXE y RXRDY, el estado de las banderas de overrun, paridad y framing.

Es importante hacer notar que la mayoría de las microcomputadoras sólo manejan un puerto asíncrono. El 3GJ utiliza un puerto asíncrono/síncrono. De tal forma que si las microcomputadoras llegasen a utilizar comunicación síncrona, para lograr transnmisiones más rápidas, el 3GJ está capacitado para soportarlo. Y sólo resta hacer modificaciones en software y un mínimo en hardware.

El PPI 8255

Como mencionamos al principio de este capítulo, el 3GJ tiene capacidad para mandar a impresión hacia dos impresoras de puerto paralelo. Para conseguir esto, hemos utilizado al Programmable Peripheral Interface (PPI) 8255 de Intel, el cual es compatible con TTL, con el microprocesador 8088 y encapsulado de 40 pines. Con este circuito pueden seleccionarse 24 salidas que pueden constituirse en hasta 3 puertos de 8 bits ya sean de entrada o salida, o bien 2 puertos de 8 bits y dos de 4 bits, configurar puertos de 8 bits con pines de control manejados directamente desde el microprocesador o por el circuito en sí.

Tiene un pin que se conecta a la salida de reset del 8284, pines para escribir tanto caracteres de control,o datos. Definitivamente que la operación de este circuito es más sencilla que la del USART 8251 y por ende tiene menos pines de control. De

hecho, carece de palabras de estatus para leer.

Puede programarse en tres modos:

1.Modo 0. Esto es cuando se desea utilizar los puertos como entrada o salida sin handshake. Pueden entonces programarse los puertos A y B como entradas o salidas indistintamente en tanto que el puerto C puede partirse en 2 nibbles (el más y el menos significativo), los cuales pueden ser indistintamente entradas o salidas.

2.Modo 1. En esta opción pueden programarse los puertos A y B como entradas y salidas indistintamente y el puerto C va a proveer señales de Handshake para estos puertos automáticamente.

3.Modo 2. Sólo puede programarse el puerto A en esta opción. Así, el puerto A se convierte en Bidireccional y 5 de los bits del puerto C proporcionan las señales de handshake. El puerto B puede utilizarse en modo 0 o bien en modo 1 con las señales de handshake proporcionadas por los 3 bits restantes del puerto C.

Estos modos se determinan por medio de una de las dos palabras de programación con que cuenta el circuito. La otra palabra, sirve básicamente para indicar qué bit debe prenderse o apagarse en el puerto C en caso de que se programe alguna parte como salida en modo 0.El 3GJ utiliza el modo 0 ya que ninguna de los otros dos modos fue compatible con el handshake de la impresora que manejamos. De esta forma, usando el puerto A y B en modo 0 podemos de alguna manera controlar el tiempo que permanece una señal de handshake como es el STROBE activa.

HARDWARE, SELECCION Y DISEÑO

Introducción

En esta sección se hará la descripción de los métodos que utilizamos para el diseño del hardware. También se mencionarán las razones por las cuales llegamos a determinada configuración de circuitos. Se expondrá un análisis de tiempos, los cuales nos llevaron a realizar la selección de los circuitos integrados definitivos.

4.1 El Microprocesador

En la figura 4.1 se muestra la configuración básica del microprocesador 8088 que utilizamos en la construcción del circuito 3GJ.

Como se observa no todas las líneas de direcciones pasan a través de latches de demultiplexión y retén. Esto es debido a que no todas las líneas de dirección se requieren para realizar la decodificación y el microprocesador da la potencia necesaria para que aquellas que no sufren el proceso antes mencionado lleguen sin pérdida. El resto de la configuración es más estándar.

4.2 Decodificación

En lo referente a circuitos de decodificación existen las opciones de utilizar varias tecnologías como puede ser ECL, MOS, I2L, TTL, etcétera Nos dimos cuenta de que no tenemos problemas en cuanto al consumo de potencia, como para utilizar circuitos MOS o bien problemas de velocidades como para utilizar circuitos ECL. En cambio, sí resultaría contrapruducente utilizar estos circuitos. Tendríamos por ejemplo que

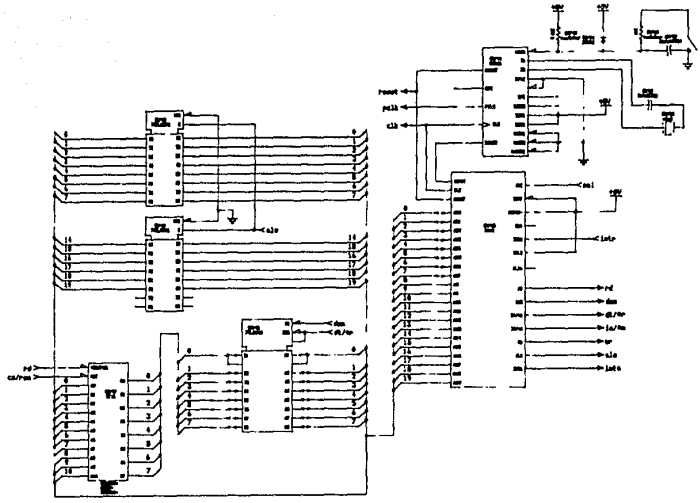


Figura 4.1 Configuración Básica del Microprocesador 8088, para el circuito 3gj

los circuitos con los que más experiencia tenemos y en México existen en casi todas sus modalidades, decidimos utilizar a esta familia en la construcción de nuestro Hardware. Los fabricantes de estos circuitos son Texas Instruments, Signetics e Intel. La mayoría de los circuitos fueron manufacturados por los dos primeros fabricantes mencionados y a Intel se recurrió cuando el circuito no era ofrecido por los fabricantes primeramente mencionados o cuando era demasiado sofisticado, como es el caso del circuito de reloj 8284.

En la figura 4.2 se presenta un diagrama del mapa de memoria y puertos que utiliza

el circuito 3GJ.

Dirección	Memoria
00000	ZONA DE MEMORIA RAM
F7FFF	
F8000	ZONA DE MEMORIA ROM
FFFFF	
Dirección	**Puertos**
0000	ZONA DE PUERTOS SERIE PARA DATOS
000F	
0020	ZONA DE PUERTOS SERIE DE CONTROL
002F	
0010	ZONA DE PUERTOS ADICIONALES
001F	

Figura 4.2 Mapa de Memoria y Puertos

La decodificación se implementó con los circuitos que aparecen en la figura 4.3.

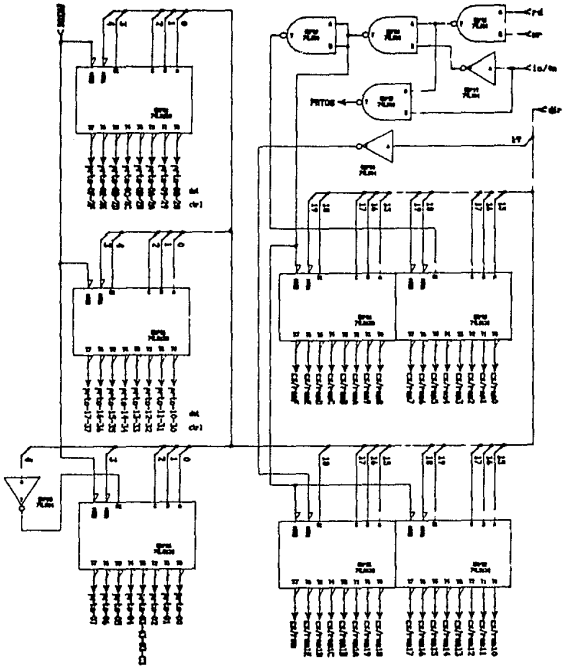


Figura 4.3 Decodificación de Memoria y Puertos

4.3 Memoria

Un elemento de gran importancia en el 3GJ es la memoria que va a tener. Recordemos que el objetivo del aparato es poder recibir archivos de 16 microcomputadoras simultáneamente permitiendo a estas desentenderse del archivo que enviaron a impresión, siendo entonces el aparato el encargado de averiguar en que momento puede imprimir este archivo. Para lograr esto, es de entenderse que se requiere una gran cantidad de memoria. Para escoger la memoria partimos de la siguiente información:

Debe ser una memoria de lectura y escritura ya que deben de poder grabarse los archivos que vienen de las PC's y leerse de ellas en el momento en que se envían a impresión.

La memoria podía ser permanente o volátil. El 3GJ no requiere tener la información de manera permanente ya que en la mayoría de los casos la información permanece permanentemente en el disco de la PC que lo envía. Por esto y porque los dispositivos de alamacenamiento permanente, como pueden ser las unidades de disco tienen un costo elevado, se optó por usar memorias volátiles. Así mismo se pensó en utilizar solamente memoria primaria pues no se tiene planeado que los archivos permanezcan en el 3GJ durante mucho tiempo, la idea es que sean impresos en un breve lapso.

En el mercado de circuitos integrados día con día se establecen nuevas marcas, diseños y modelos de circuitos. En períodos cortos de tiempo los circuitos sufren cambios que mejoran las características que sus antecesores. Los circuitos que participan con mayor entusiasmo en esta carrera de mejoramiento son las memorias. Tanto las memorias estáticas como las dinámicas son reemplazadas por nuevos productos especificamente más densos y más veloces.

Se requiere en un momento dado toda una discusión para saber qué tipo de RAM utilizar, si estática o dinámica. Debido a la gran cantidad de memoria a utilizar, se empezaron a buscar opciones en cuanto a memoria dinámica y se vió la necesidad de tener un controlador de memoria dinámica capaz de manejar 1 MB de memoria. Después de mucho buscar se encontró uno el cual sólo se podía conseguir

pidiéndolo al fabricante en los Estados Unidos. Dicho circuito a su vez requería de un circuito anexo para poder funcionar apropiadamente con el microprocesador 8088. Por otro lado se investigó qué memorias existían y se encontraron memorias que almacenaban 256Kb x 1 y unas memorias de 1 Mb x 1. Con los chips de 256 kb x 1 se requieren 16 chips y con el de 1Mb x 1 se necesitan 8 más los circuitos controladores.

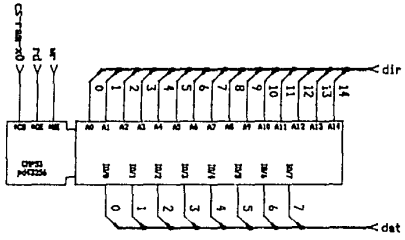


Figura 4.4 Configuración de un Bloque RAM

Se encontraron también memorias estáticas que manejan como máximo 32kb x 8 y que obviamente no requieren circuitos controladores. Su tiempo de acceso es aceptable para la velocidad a la que se pretende trabajar. Con esta opción se requieren 32 chips.
Para decidir por una u otra opción se tuvo en cuenta que deseamos que el 3GJ pueda crecer en forma modular, y en lo referente a la memoria se planeó que creciera en bancos de 256 KB de forma que el dueño del aparato puede en un momento dado en base al tipo de archivos que se mandan decidir si solo requiere 256 KB de memoria y ademas puede crecer en expansiones de 256 KB ya que comprar chips de 1Mb x 1 puede resultar en un momento dado caro. Sin embargo, todavía quedó la posibilidad de poner bancos de 256 KB con memorias dinámicas. No se aceptó esta opción debido a que como de todas formas se iba a requerir el controlador de 1 Mega el cual es caro, difícil de conseguir y potencialmente podía retrasar el desarrollo del proyecto. Así pues, se decidió contruir los bancos con memorias estáticas de 32 KB.

En la figura 4.4 se detalla la conexión de uno de los bancos de memoria RAM.

Parámetros del Tiempo

Al construir una determinada arquitectura basada en un microprocesador se debe de tener en cuenta que el microprocesador es capaz de trabajar a una determinada velocidad, (por lo regular los microprocesadores de propósito general trabajan en el rango de unidades de megahertz). Este parámetro es de gran importancia sobretodo si se decide que el diseño interactúe con otros dispositivos. De lo anterior se desprende que otros dispositivos puedan ser más rápidos o más lentos que el microprocesador. En un sistema microprocesado, es necesario que todos los circuitos actúen en forma armónica con el microprocesador. Para ello es necesario uniformizar 'os rangos de velocidad en los cuales trabajan los circuitos. Se pueden presentar problemas de tiempos cuando los circuitos alrededor del microprocesador, consumen más tiempo que éste al leer o escribir en el canal común. Estos problemas se presentan con regularidad en diversas zonas de una arquitectura, aunque las más comunes resultan ser bloques como el de decodificación tanto de puertos como de memoria. También se llegan a tener problemas con el tiempo de respuesta de las memorias que se utilizan ya sean RAM o ROM. El problema que se tiene en las etapas de decodificación es causado por dos razones :

1.- *El tiempo de respuesta de los circuitos*
2.- *Demasiadas etapas en la decodificación*

En algunas ocasiones llegan a presentarse ambos problemas. Cualquiera que sea el caso se deben tomar las precauciones necesarias para salvar tales líos.
En cuanto a las memorias el caso en específico es que se puede haber seleccionado para el sistema, una memoria con un tiempo de respuesta muy grande para trabajar a la frecuencia del microprocesador. En tal caso habrá que averiguar si lo anterior se cumple y en caso afirmativo, dar la solución al problema.

Para el diseño del circuito 3GJ hicimos el siguiente análisis de tiempos :

En primer lugar ¿cuánto tiempo nos da el micro ? En la figura 4.5 aparecen los diagramas de tiempo de los ciclos de escritura y lectura del microprocesador 8088/8086. De acuerdo con los diagramas de tiempo :

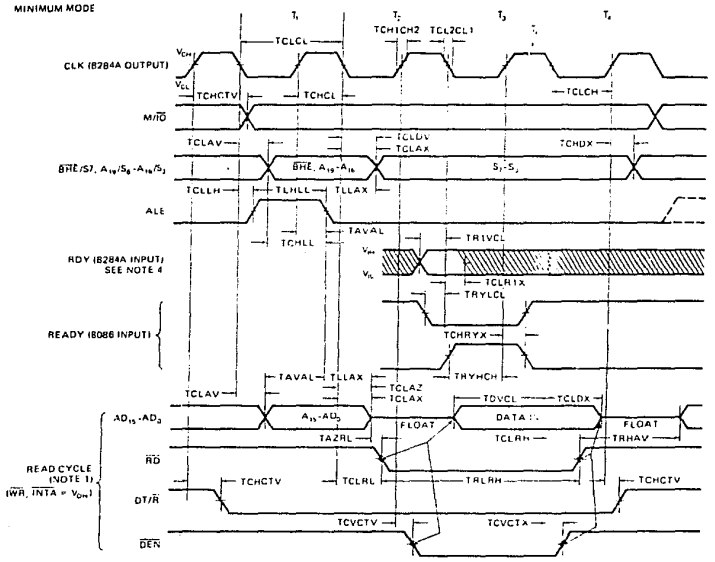
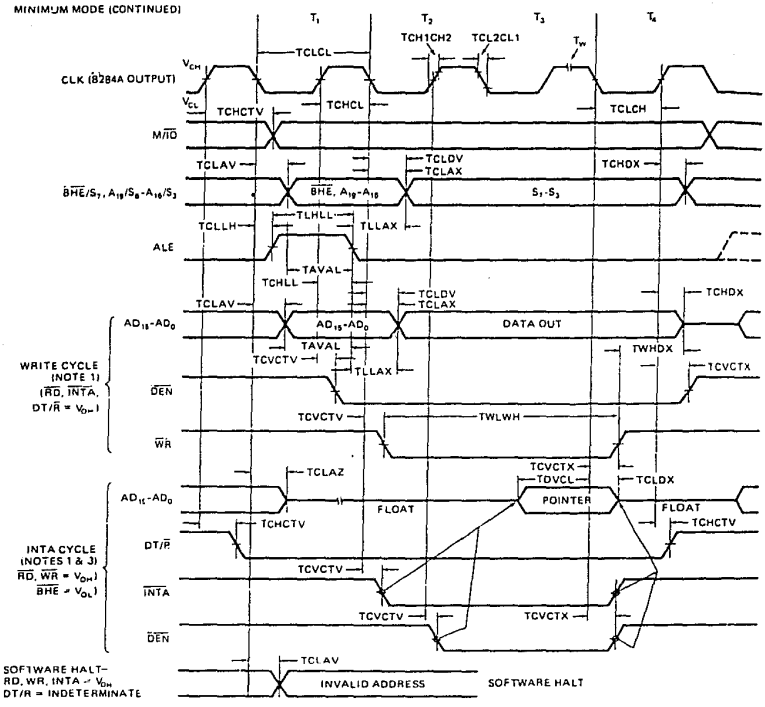


Figura 4.5a Diagrama de tiempos del Ciclo de Lectura



NOTES:
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Two INTA cycles run back-to-back. The 8086 LOCAL ADDR/DATA BUS is floating during both INTA cycles. Control signals shown for second INTA cycle.
4. Signals at 8284A are shown for reference only.
5. All timing measurements are made at 1.5 V unless otherwise noted.

Figura 4.5b Diagrama de tiempos del ciclo de Escritura

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

SYMBOL	PARAMETER	8086		8086-(Preliminary)		8086-2		UNITS	TEST CONDITIONS
		MIN.	MAX	MIN.	MAX.	MIN.	MAX.		
TCLCL	CLK Cycle Period	200	500	100	500	125	500	ns	
TCLCH	CLK Low Time	118		53		68		ns	
TCHCL	CLK High Time	69		39		44		ns	
TCH1CH2	CLK Rise Time		10		10		10	ns	From 1.0 V to 3.5 V
TCL2CL1	CLK Fall Time		10		10		10	ns	From 3.5 V to 1.0 V
TDVCL	Data in Setup Time	30		5		20		ns	
TCLDX	Data in Hold Time	10		10		10		ns	
TR1VCL	RDY Setup Time into 8284A (See Notes 1, 2)	35		35		35		ns	
TCLR1X	RDY Hold Time into 8284A (See Notes 1, 2)	0		0		0		ns	
TRYHCH	READY Setup Time into 8086	118		53		68		ns	
TCHRYX	READY Hold Time into 8086	30		20		20		ns	
TRYLCL	READY Inactive to CLK (See Note 3)	−8		−10		−8		ns	
THVCH	HOLD Setup Time	35		20		20		ns	
TINVCH	INTR, NMI, $\overline{TEST}$ Setup Time (See Note 2)	30		15		15		ns	
TILIH	Input Rise Time (Except CLK)		20		20		20	ns	From 0.8 V to 2.0 V
TIHIL	Input Fall Time (Except CLK)		12		12		12	ns	From 2.0 V to 0.8 V
TCLAV	Address Valid Delay	10	110	10	50	10	60	ns	
TCLAX	Address Hold Time	10		10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	10	40	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH-20		TCLCH-10		TCLCH-10		ns	
TCLLH	ALE Active Delay		80		40		50	ns	
TCHLL	ALE Inactive Delay		85		45		55	ns	
TLLAX	Address Hold Time to ALE Inactive	TCHCL-10		TCHCL-10		TCHCL-10		ns	
TCLDV	Data Valid Delay	10	110	10	50	10	60	ns	
TCHDX	Data Hold Time	10		10		10		ns	
TWHDX	Data Hold Time After WR	TCLCH-30		TCLCH-25		TCLCH-30		ns	
TCVCTV	Control Active Delay 1	10	110	10	50	10	70	ns	
TCHCTV	Control Active Delay 2	10	110	10	45	10	60	ns	
TCVCTX	Control Inactive Delay	10	110	10	50	10	70	ns	
TAZRL	Address Float to READ Active	0		0		0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	10	70	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	10	60	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL-45		TCLCL-35		TCLCL-40		ns	
TCLHAV	HLDA Valid Delay	10	160	10	60	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL-75		2TCLCL-40		2TCLCL-50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL-60		2TCLCL-35		2TCLCL-40		ns	
TAVAL	Address Valid to ALE Low	TCLCH-60		TCLCH-35		TCLCH-40		ns	
TOLOH	Output Rise Time		20		20		20	ns	From 0.8 V to 2.0 V
TOHOL	Output Fall Time		12		12		12	ns	From 2.0 V to 0.8 V

*C_L = 20 − 100 pF for all 8086 Outputs (In addition to 8088 selfload)

NOTES:
1. Signal at 8284A shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state. (8 ns into T3).

Figura 4.5c Tabla de Parámetros de los Ciclos de Lectura y Escritura

En la línea de direcciones AD15-AD0 está marcado un período de tiempo como TCLAV después del borde de bajada del reloj en el ciclo T1. TCLAV significa " time from clock low to address valid ". TCLAV es entonces el período desde que el reloj permanece bajo y existen direcciones válidas. De acuerdo a la tabla de tiempos el período TCLAV es de a lo más 110 ns. Siguiendo la línea AD15-AD0, los datos permanecerán como datos válidos durante un tiempo TDVCL antes del borde de bajada del reloj en el ciclo T3. TDVCL significa " time data valid before clock goes low ". La hoja de datos da un tiempo de 30 ns para este parámetro.
El período de tiempo entre el final de TCLAV y el inicio de TDVCL es el tiempo permitido para obtener una dirección correcta o bien enviar una dirección y que el dispositivo la tome como correcta. Este tiempo se determina Entonces como :

TPA = 3 ciclos de reloj - TCLAV - TDVCL
nota: TPA (tiempo permitido en direcciones)

Realizamos estos cálculos para dos frecuencias a las que puede trabajar el circuito 3GJ :

$TPA_{4.9\ MHz} = 3 * (204\ ns) - 110\ ns - 30\ ns$

$\mathbf{TPA_{5\ MHz} = 472\ ns}$

$TPA_{6.6\ MHz} = 3 * (150\ ns) - 110\ ns - 30\ ns$

$\mathbf{TPA_{6.6\ MHz} = 310\ ns}$

En segundo lugar, ¿el tiempo de reacción de la memoria en sus direcciones es suficiente?

En los diagramas del circuito 3GJ observamos que las direcciones llegan a la memoria a través de latches 74ls373. Hay que tomar en cuenta el tiempo de retraso de este circuito para saber con que retraso llegan las señales. De las hojas de especificación

del circuito tenemos que el tiempo de propagación máximo es de 18 ns.

Si restamos este retraso al tiempo que nos da el microprocesador obtenemos el tiempo que le damos a la memoria para reaccionar a las señales de direcciones :

$$TPA_{4.9\ MHz} = TPA_{4.9\ MHz} - 18\ ns$$

$$\mathbf{TPA_{4.9\ MHz} = 454\ ns}$$

$$TPA_{6.6\ MHz} = TPA_{6.6\ MHz} - 18\ ns$$

$$\mathbf{TPA_{6.6\ MHz} = 292\ ns}$$

En tercer lugar hay que considerar el tiempo de reacción de las memorias, este intervalo de tiempo es referido como t acc "output access time". Consultando en un manual de Intel para memorias hallamos el tiempo de acceso (tacc) de diferentes memorias :

Memoria	tacc (ns)
2716	450
2716-1	350
2732	450
2732-A	250
2732-A-2	200
27128	200

De acuerdo a lo anterior podemos obtener varias combinaciones que dependen de la frecuencia del sistema y el tiempo de reacción de las diferentes memorias :

4.9 MHz $TPA = 454ns$ *2716 2716-1 2732 2732-A 2732-A-2*

6.6 MHz TPA = 292ns *2732-A 2732-A-2 27128*

La tabla anterior ilustra el tipo de memoria que se puede utilizar dependiendo de la frecuencia con a la que opere el sistema.

En cuarto lugar ¿cuánto consume el circuito decodificador ?
*De los diagramas del circuito 3GJ vemos que las señales que intervienen en las etapas de decodificación son las direcciones, y la señal IO/*M.*

*Si observamos los diagramas de tiempo del micro en el ciclo de lectura nos damos cuenta de que la señal IO/*M cambia casi justo después de iniciado el ciclo T1. Las direcciones se retrasan un tiempo TCLAV que como ya se mencionó es a lo más de 110 ns. El pulso selección (*CE) de la memoria debe estar presente en el mismo intervalo de tiempo que las direcciones, es decir, entre el fin de TCLAV y el inicio de TDVCL. Tal período de tiempo ya fue calculado para el análisis de direcciones y por lo tanto tomaremos el dato de aquella tabla*

El circuito que interviene en la decodificación de la memoria es el 74ls138. Buscando en las hojas de datos de éste circuito hallamos que su tiempo de propagación máximo es de 20 ns. Restando éste tiempo al de la tabla mencionada tenemos que:

$$TPA_{4.9\ MHz} = TPA_{4.9\ MHz} - 18\ ns - 20\ ns$$

$$TPA_{4.9\ MHz} = 434\ ns$$

$$TPA_{6.6\ MHz} = TPA_{6.6\ MHz} - 18\ ns - 20\ ns$$

$$TPA_{6.6\ MHz} = 272\ ns$$

*De las hojas de datos de las memorias obtuvimos que el tiempo de reacción de en la entrada *CE (chip enable) es igual al tiempo de reacción de las direcciones con*

lo que elaboramos la siguiente tabla en la que se muestran las memorias que se adecúan a trabajar con cada una de la frecuencias del sistema.

4.9 MHz TPA = 434ns *2716-1 2732-A 2732-A-2 27128*

6.6 MHz TPA = 272ns *2732-A 2732-A-2 27128*

El último parámetro que hubo que considerar para acoplar la memoria, fue el tiempo de reacción de las salidas de la memoria; t oe (output enable time).
*El microprocesador toma como datos válidos en su canal a las señales que aparecen en el intervalo TDVCL. Si observamos en los diagramas del circuito 3GJ la señal conectada a la entrada *OE (output enable) de la memoria es la señal *RD. Esta señal es generada un tiempo TCLRL después del borde de bajada del reloj del ciclo T1. Entonces el tiempo permitido para colocar datos en el canal es :*

TPD = 2 Ciclos de reloj - TCLRL

nota: TPD tiempo permitido para datos

Realizamos estos cálculos para dos frecuencias a las que puede trabajar el circuito 3GJ :

TPD 4.9 MHz *= 2 * (204 ns) - 100 ns*

TPD* 4.9 MHz *= 308 ns

TPD 6.6 MHz *= 2 * (150 ns) - 100 ns*

TPD* 6.6 MHz *= 200 ns

Ahora hay que considerar el tiempo de reacción de las memorias, este intervalo de tiempo es referido como t oe "output enable time". Consultando el manual de Intel

para memorias hallamos toe de diferentes memorias :

Memoria	toe (ns)
2716	150
2716-1	100
2732	150
2732-A	100
2732-A-2	70
27128	80

De acuerdo a lo anterior podemos obtener varias combinaciones que dependen de la frecuencia del sistema y el tiempo de reacción de las diferentes memorias :

4.9 MHz TPD = 308ns *2716 2716-1 2732 2732-A 2732-A-2 27128*

6.6 MHz TPD = 200ns *2716 2716-1 2732 2732-A 2732-A-2 27128*

Particionamiento de Memoria

Ya que se está hablando de memorias y aún cuando el alambrado definitivo no quedó así hablaremos acerca de un truco que ayudó mucho a la hora de estar diseñado el circuito, el software y que merece mención especial.

Al principio, la totalidad de las pruebas hechas al circuito se realizaron con programas que corrían en una ROM 2716 de 2KB. Como no contábamos con borradores de EPROM's y los programas de prueba eran muy pequeños (los más grandes eran de 128 localidades, se optó por particionar la ROM en bloques de 128 localidades. Esto era necesario pues siempre que arrancaba el micro comenzaba a correr el programa, en las ultimas 16 localidades de memoria en donde se le enviaba a otra dirección donde ya empezaba a correr el programa por lo que era necesario de alguna manera

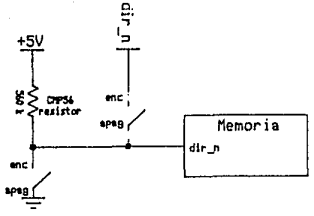


Figura 4.6 Circuito para el Particionamiento de memoria ROM

engañar al microprocesador. Esto se puede lograr permitiendo que el microprocesador maneje 7 líneas de direcciones de la memoria y las 4 restantes se envían a 5V o a tierra generando así los bloques. Para que el programa funcionara correctamente, en las ultimas 16 localidades del bloque se guardaba el salto el cual para el microprocesador estaría hasta el final del mapa de memoria, en este salto se enviaba al microprocesador realmente al principio del bloque pero para el sería una dirección que podría estar en cualquier parte del mapa.

Posteriormente se empezaron a crear programas más grandes que requerían bloques de hasta 512 KB por lo que en ocasiones ciertas líneas de direcciones altas (A7, A8, A9, A10) requerían ser manejadas por el microprocesador y en otras ocasiones se enviaban a 5V o a tierra. Para lograr esta función se alambró como se muestra en la figura 4.6.

4.4 Periféricos

El Puerto Paralelo

Como se mencionó en páginas anteriores el circuito 3GJ está capacitado para el manejo de tres dispositivos impresores que trabajen en forma simultánea, es decir, que estén imprimiendo las tres al mismo tiempo archivos diferentes y de manera independiente una de las otras. Abundando en lo ya mencionado, también se describió que dos de los dispositivos impresores se manejan a través de información enviada en forma paralelo y el tercer dispositivo a través de información transmitida en formato serie.

En la implementación que se diseñó para el circuito 3GJ seleccionamos, para el manejo de estos dispositivos, un producto de la familia Intel del microprocesador utilizado (8088). Es un puerto paralelo de tres canales, el 8255.
Este puerto lo configuramos de forma tal que fuera capaz de manejar dos canales de 8 datos más dos señales de control para el comando de los impresores. En la figura 4.5 se muestra la disposición de los tres canales del 8255 para nuestro objetivo.

La decodificación de puertos habilita al 8255 justo cuando se requiere su intervención dentro de la rutina de operación del circuito, el circuito de reinicialización(reset) está directamente conectado al puerto de forma que al ejecutar un reset en el aparato, este puerto no quede programado y opere de forma incorrecta.

En la figura 4.7a y 4.7b se puede apreciar el alambrado del puerto paralelo.

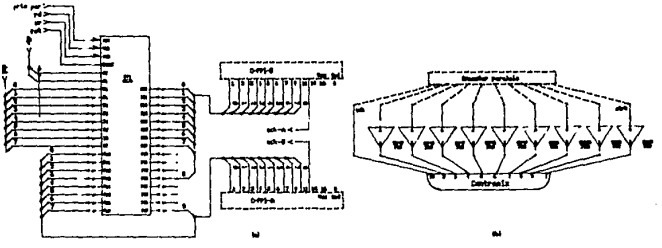


Figura 4.7a Configuración del Puerto Paralelo
Figura 4.7b Interfase de Salida Paralelo

El Puerto Serie

El enlace que establece el circuito 3GJ con las microcomputadoras personales (PC's), se lleva a cabo a través de una comunicación serie asíncrona. Seleccionamos este

tipo de comunicación debido a que presenta las siguientes cualidades :

+ Es un enlace soportado por el sistema operativo de cualquier microcomputadora. Este tipo de comunicación es establecido también en la mayoría de los equipos de cómputo.

+ Es un tipo de enlace económico. Dado que la información viaja en serie a través de un sólo alambre, se utilizan menos alambres por metro cubierto, que por ejemplo en una comunicación paralelo.

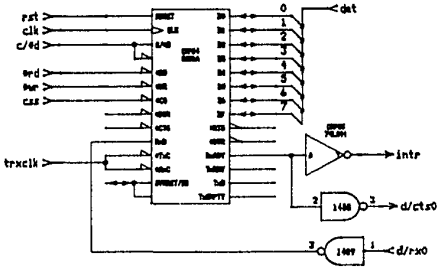


Figura 4.8 Circuito de Salida Serie

+ Con la comunicación serie se alcanzan distancias de enlace bastante grandes. La transmisión en forma serial es menos susceptible a la contaminación por ruido, que como lo es la comunicación en forma paralela.

Entonces las PCs envían su información hacia el circuito 3GJ a través de su puerto serie. En la figura 4.8 se muestra la forma de conexión que se utilizó para el manejo de los 16 puertos serie con los que opera el equipo 3GJ.

4.5 Atención a Periféricos

Existen dos maneras en que se puede atender a los usuarios del 3GJ que quieren enviar un archivo de impresión. Dichos métodos a saber son el poleo y las interrupciones.

Para saber si una computadora desea enviar un archivo, se puede utilizar un método de poleo, el cual mantiene al microprocesador bastante ocupado preguntando qué

computadora requiere enviar información. Para esta opción, por algún puerto en especial se estaría preguntando por el RTS de la PC (Véase capitulo 2), y cuando estuviera presente se entendería que esa computadora quiere enviar un archivo. En este proceso el microprocesador pierde mucho tiempo preguntando a los usuarios si van a enviar algo, en vez de realizar tareas más importantes.

Por el método de interrupciones se libera al microprocesador de esta tarea, permitiéndole además, realizar otras de mayor prioridad como puede ser el administrar los bloques de memoria, entre otras cosas. En el 3GJ, se puede saber que una computadora quiere enviar información por medio de interrupciones. Así, la computadora que quiera enviar información, interrumpirá al microprocesador, éste, si se encuentra desocupado la atenderá y, en caso de estar ocupado, la línea de interrupción permanecerá activada hasta que el microprocesador pueda atenderla. Este tiempo de espera será pequeño.
De lo expuesto puede verse que es mejor utilizar el método de interrupciones que el de poleo, por lo que es el que se implementó en el 3GJ.

Protocolo XON/XOFF o DTR para PC's e impresora Serie

Aunque en el estricto significado de los términos no constituyen un protocolo (Ver capitulo 2), se les conoce popularmente así. La utilidad de ambos métodos es controlar el flujo de información. XON/XOFF se basa en que el receptor del mensaje envíe un comando XOFF (^Q, DC1 u 11H), con lo cual el transmisor deja de enviar el mensaje y una instrucción XON (^S,DC3 o 13H) con la que el receptor indica que el mensaje puede continuar enviándose. El protocolo DTR implica que el flujo de información se maneje a través de las líneas de control del RS-232-C (Ver capitulo 2).

Así pues, ¿cuál utilizar? Es fácil porque partimos del hecho de que las PC's no emplean el protocolo XON/XOFF, por lo que se eligió el DTR en lo que respecta de las PC's al 3GJ.

Sin embargo, es tan común el otro protocolo que la impresora serie si lo maneja y es por ello que se eligió para la comunicación 3GJ-impresora serie.

Manejo de Interrupciones

Se idearon varios circuitos con los que se pretendía conectar las PC's al 3GJ. Ellos representan una de las partes más importantes del proyecto por lo que vale la pena comentarlos a continuación.

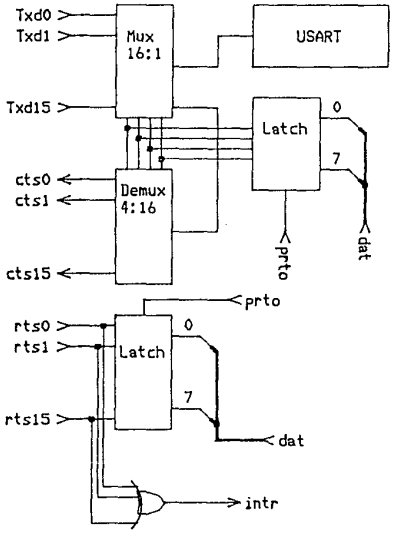


Figura 4.9 Circuito de Interrupciónes, con señal RTS

Inicialmente se ideó el circuito de la figura 4.9 el cual pretendía que fuera el RTS de las PC's el que interrumpiera. El microprocesador al atender la interrupción leería de un latch o buffer que PC había interrumpido y al detectarla habilitaría su CTS e inhabilitaría los de las demás PC's permitiendo que la que quería transmitir empezara a enviar el mensaje. El mensaje llegaría a la línea de recepción del 8251 por medio de un multiplexor que el microprocesador manejaría directamente. Con este circuito tuvimos muchos problemas pues observamos que la señal RTS no se comportaba como debiera de ser (Ver capitulo 2) por lo que tuvo que desecharse esta opción.

Se diseñó entonces el circuito presentado en la figura 4.10. El funcionamiento es parecido al anterior solo que ya no es la señal RTS la que interrumpe sino que es la misma línea de datos la que lo hace. Se pensaba que iba a ser el start bit el que al llegar al flip flop, permanecería allí interrumpiendo al microprocesador hasta que él mismo lo borrara. El microprocesador al atender inhabilitaría

interrupciones para evitar que el cambio de nivel de datos volviera a interrumpir, vería quien interrumpió, habilitaría el CTS de la PC que hubiera interrumpido y con el multiplexor recibiría los datos en el 8251.

Pensamos que este circuito podía mejorarse quitando los flip flops ya que lo más probable es que tan pronto llegará el Start bit de una PC, ésta interrumpiría al microprocesador y el rápidamente la atendiera inhabilitando a las demás.

Ya se había alambrado y probado este circuito cuando nos dimos cuenta de que era un error muy grave el multiplexar el 8251 y encadenarlo a una PC, de la cual recibe información. Con esto lo que sucedería es que la recepción de datos de las PC's no se haría de forma simultánea sino que primero se recibe de una, y otra no puede enviar hasta que la primera haya terminado.

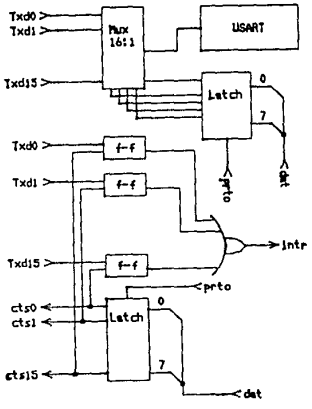


Figura 4,10 Circuito de Interrupciones, señal de datos

Nuestro circuito 3GJ debe entonces contar con 16 USART 8251!. Definitivamente que se eleva el costo del circuito y se complica bastante el alambrado, pero la eficiencia del equipo se incrementa notablemente. Esto podemos verlo desde varios puntos de vista. Así pues, es evidente que con 16 usart's, atendemos simultáneamente a 16 computadoras y que una vez que el caracter que llega se convierta a paralelo, el microprocesador puede ir a leerlo rápidamente liberando al usart en cuestión, y permitiéndole así recibir otro caracter. Además, como el proceso se ejecuta por interrupciones el microprocesador es avisado de cuando ir a recoger ese caracter.

Ahora supongamos un caso critico que bien puede darse en un circuito que anuncia

tener gran capacidad de memoria. Es el caso de cuando las 16 computadoras intentan enviar cada una un archivo de 36 KB. Ocupémonos de la recepción. Así pues, a 9600 bps, un bit se recibe en 104 us. Un caracter que consta de 8 bits, un bit de start y uno de stop tarda en recibirse 1.04 ms y en consecuencia 36 KB tardan en recibirse 38.28 s. Como son 16 usuarios, el último será atendido 38.28 x 16 = 10'13.3" por lo que ya llega a ser ineficiente su funcionamiento y la atención a usuarios deja de ser simultánea. El tiempo de atención puede ser mayor pues no se ha tomado en cuenta el tiempo en el que el microprocesador manda a imprimir o realiza otro tipo de manejos internos.

El 3GJ puede ser muchísimo más eficiente en este sentido si tiene 16 USART's para comunicarse con las PC's pues así es capaz de recibir verdaderamente en forma simultánea de las 16 computadoras por lo que un archivo de 36 KB de cada PC a 9600 puede enviarse en tan sólo 38.28 s. tiempo que sólo depende de la velocidad de transmisión y no de quien desocupe un circuito dado. Así a todos los usuarios se les atenderá inmediatamente y no mas rápido al primero y muy lento al usuario 16.
Todo lo expuesto es cómo el 3GJ utiliza 16 puertos serie. La señal que utilizamos entonces para interrumpir es la llamada RXRDY, la cual permanece activa mientras que no se lea el caracter del USART.

El Circuito de Interrupciones

El circuito 3GJ en su parte de atención a dispositivos, como ya se ha mencionado, trabaja a base de interrupciones. Tales interrupciones provienen, por un lado, de las microcomputadoras personales (PC) conectadas en la red 3GJ. Las interrupciones hechas por las PC's son interpretadas como una solicitud de atención para poder enviar información. La otra parte de las interrupciones lo forman las hechas por el (los) dispositivo(s) impresor (es). Estas últimas interrupciones también son interpretadas como solicitud de atención, pero para avisar que han recibido un caracter. Las interrupciones hechas por las computadoras son mascarables mientras que las hechas por las impresoras, no lo son.
La filosofía del funcionamiento del circuito 3GJ se basa en que el dispositivo impresor, dado que trabaja a menor velocidad que las computadoras, requiere de una rutina de trabajo en la cual no se le haga perder tiempo. Esta es la razón por la cual ellos utilizan interrupciones del tipo no mascarable.

La implementación del circuito de interrupciones es sencilla y puede describirse como una compuerta "OR" , que avisa de la ocurrencia de una interrupción, y un canal de 16 bits que es utilizado como estafeta para localizar el dispositivo interruptor.

En la figura 4.11 se muestra el circuito de interrupciones utilizado en el circuito 3GJ.

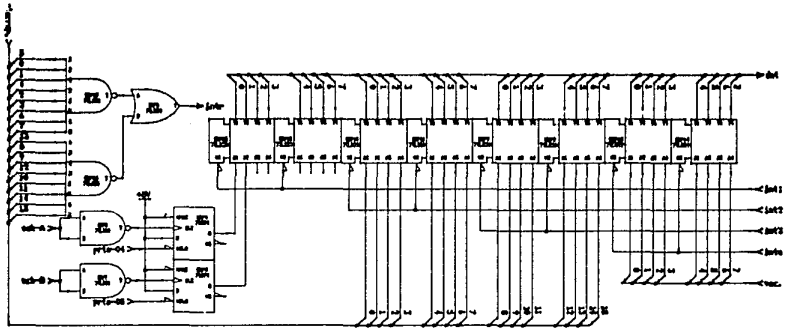


Figura 4.11 Circuito de Interrupciones Mascarables

En la construcción del circuito 3GJ se utilizaron dos circuitos de interrupciones. Se explicó anteriormente el utilizado por las interrupciones mascarables, ahora en la figura 4.12 se presenta el circuito utilizado por las interrupciones no mascarables.

Este circuito está muy apegado al funcionamiento de los dispositivos impresores paralelo. Este tipo de dispositivos envían una señal al momento de recibir un dato que les fue enviado, pero la duración de esta señal de un tiempo determinado por lo que el circuito de interrupciones tuvo que adaptarse a estos requerimientos.

La forma de acoplar los circuitos de los

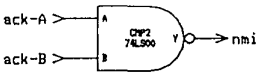


Figura 4.12 Circuito de Interrupciones No Mascarables

dispositivos impresores con nuestra lógica de interrupciones, fue implementada a través de circuitos Flip-Flop's, cuya función está en capturar la señal enviada por el impresor en el momento de su ocurrencia, y de ahí no perderla sino hasta que el microprocesador sea capaz de reconocerla.

4.6 Fuente

Existen dos alternativas para la fuente de energía del sistema, y hemos visto que se acostumbran a tomar en este tipo de proyectos. Puede construirse la fuente, pero también es común comprarla e instalarla. Para el 3GJ se estudiaron ambas posibilidades y el circuito que se muestra a continuación resultó ser tan caro como comprar una fuente para computadora en Estados Unidos (50 USDLS) por lo que podemos tomar ambas opciones, sobre todo la se pueda obtener más rápidamente. En la figura 4.13 se muestra el diagrama de la fuente diseñada.

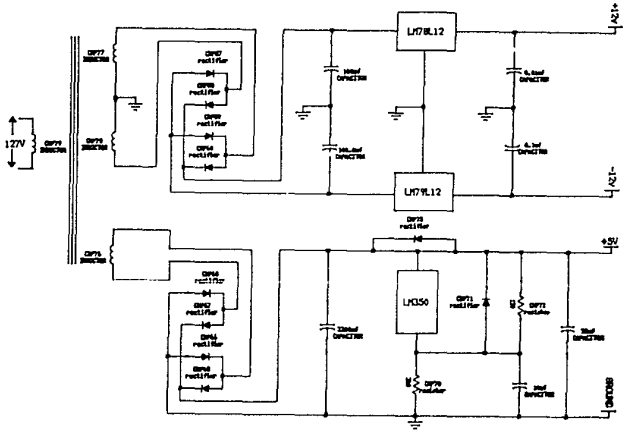


Figura 4.13 Circuito de la Fuente

SOFTWARE

Introducción

En este capítulo explicaremos toda la programación relacionada con el proyecto. Tuvimos que desarrollar diferentes tipos de programas: programas de prueba para el circuito, programas de apoyo para desarrollo de otros programas, programas definitivos para el circuito y el programa para las computadoras de la red.

Los programas de prueba del circuito se explicarán en otra sección llamada construcción.

Para el desarrollo de los programas definitivos tuvimos que seleccionar entre diferentes posibilidades e ideas con el fin de obtener las mejores características para el equipo 3GJ.

5.1 Opciones de Diseño

Forzosamente las diferentes formas de hacer el programa definitivo dependían a su vez del Hardware. A continuación describimos, muchas de las ideas que se nos presentaron en alguna ocasión, pero que desechamos por las que finalmente aplicamos.

1) Atender a una sola computadora a la vez. Encadenarnos con una sola computadora, desde que nos mandara el primer caracter hasta el último.

Inconvenientes:

- El tiempo de espera para las PC's, resultaba demasiado grande. Si vamos a conectar hasta 16 computadoras en la red y casi todos quieren imprimir más o menos al mismo

tiempo (como suele ocurrir en los laboratorios de cómputo de la Facultad), el tiempo que esperaría un usuario sería casi el mismo que el que espera cuando está en la fila con su disco flexible frente a la computadora que tiene conectada la impresora.

- Desperdicio del microprocesador. Para hacer así las cosas basta con un procesador más pequeño. Si la máxima velocidad de transmisión es de 9600 bauds y el 8088 trabaja a 5 MHz la mayor parte del tiempo el micro estaría esperando al puerto.

Conclusión:

El dispositivo deberá atender a todos a la vez, de esta forma el micro aprovechará el tiempo durante el cual un puerto serie está recibiendo cada uno de los bits que conforman el byte, para atender a otros usuarios o bien realizar otras tareas. De esta forma el tiempo que un usuario de la red tiene que esperar para liberar a su PC se minimiza de manera considerable.

2) Monitoreo de las computadoras para ver si se desea transmitir un archivo. En vez de que las computadoras interrumpan, el procesador debe revisar constantemente si alguien quiere transmitir.

Inconvenientes:

- Pérdida de tiempo en la atención a puertos que probablemente no estén transmitiendo. Si el micro se dedica a revisar si alguna de las PC's conectadas desea enviar información, aún cuando nadie desee transmitir nada, estará desperdiciando tiempo que puede ocupar en tareas importantes.

- Mayor complicación del programa de recepción. Dentro de las rutinas de monitoreo se necesitarían incluír normas de jerarquía o rutinas para la atención aleatoria o bien cualesquiera formas de organización para atender a los usuarios de la red.

Conclusión:

Para aumentar la eficiencia del programa esperamos ser interrumpidos por el usuario que necesite imprimir.

3) Asignar tamaños fijos de la memoria a cada computadora. Dividir la memoria disponible en partes iguales entre las computadoras conectadas.

Inconveniente:

- Esta técnica es una ineficiente forma de administrar la memoria disponible en el aparáto. Es fácil que ocurra que mientras en la zona asignada a una computadora sobra lugar, en la de otra el espacio ya se terminó. Además si se dañara una parte de los bancos de memoria una o varias PC's habrían pérdido su zona para almacenamiento y con ello quedrían fuera de la red. Si se solucionara el problema anterior por medio de software, éste incrementaría su complejidad notablemente.

Conclusión:

Si dividimos la memoria en pequeños bloques y se los vamos asignando a los archivos que lo requieran el desperdicio se minimiza, ya que sólo en el último bloque va a haber bytes desocupados. Si ocurriese el caso del daño en los bloques de memoria, al no existir una partición fija asignada a cada usuario, entonces el resto de la memoria en buen estado puede ser utilizado por cualquera de las PC.

4) Manejar un archivo para cada computadora. Usar sólo una tabla de recepción con 16 renglones, uno para cada computadora.

Inconveniente:

Si la impresora está ocupada imprimiendo un archivo, el usuario debe esperar a que su primera impresión termine de salir para poder enviar la segunda.

Conclusión:

Se necesita que los archivos una vez recibidos sean puestos en una tabla de impresión permitiendo de ésta forma que la tabla de recepción de esa computadora quede disponible para otro archivo.

5) Esperar al fin de archivo para empezar a imprimir. Para poder imprimir varias copias del mismo archivo recibir primero todo el archivo y después empezar a imprimirlo.

Inconveniente:

- Si tenemos una configuración con poca memoria en el equipo y alguien envía un archivo bastante grande, por ejemplo una tesis, puede llenarse la memoria sin que se haya recibido el final del archivo y sin que se esté imprimiendo algo, por lo tanto, sin que se esté vaciando la memoria. Entonces el aparato quedaría detenido sin poder recibir más, ni comenzar a imprimir.

Conclusión:

Es mejor ir imprimiendo los bloques una vez que van quedando listos cada uno. De este modo es imposible que se llene la memoria, no importa la configuración, a menos que la impresora esté detenida. Pero eso está fuera de nuestro control.

5.2 Programas de Soporte

Para establecer un programa de pruebas más eficiente, requerimos de lograr comunicación entre nuestros circuitos y nosotros. Tal objetivo sólo sería posible si lográmos incluir un sistema monitor con comunicación en nuestro diseño. A fin de lograr tal objetivo fijamos una serie de pasos intermedios que nos acercarían a nuestra meta. Para ello nos dimos a la tarea de hacer un programa que estableciera esa comunicación a través de una computadora personal (PC) y el System Design Kit 86 (SDK 86). Este último cuenta con un programa monitor que proporciona el fabricante, a fin de comandar la tarjeta SDK-86 a través de una terminal. Aprovechando esa cualidad del SDK-86 e investigando la forma en la que logra la comunicación con las terminales, se hicieron una serie de programas orientados a establecer comunicación entre una PC y el SDK-86 a través de su puerto serie. Posteriormente estos programas fueron trasladados y adaptados a nuestro circuito para lograr el objetivo antes mencionado. A continuación se hará una descripción de los programas elaborados.

La PC como Terminal

El primer paso para lograr la comunicación entre la PC y los circuitos consistía en hacer un programa que convirtiera a ésta en una terminal tonta. Es decir, la computadora se dedicaría sólo a recibir información y desplegarla en la pantalla, y los caracteres escritos en su teclado se transmitirían por su puerto serie. Este y los demás programas que se escribieron se basan en las llamadas a rutinas del BIOS. Estos programas están diseñados en base a estas llamadas porque así no tenemos que estar profundamente involucrados con el hardware de la PC, además los programas escritos haciendo llamadas a rutinas de servicio del BIOS resultan ser mayormente compatibles con diferentes marcas de computadores personales. Esto último resulta ser debido a que cada fabricante especifica diferentes localidades y direcciones para su arquitectura.

En la figura 5.1 se muestra el algoritmo que sigue el programa llamado TERMINAL, que fue el primer programa escrito para el fin que perseguimos.

Las partes del algoritmo que pueden ser implementadas con llamadas a las rutinas de servicio del BIOS son las siguientes:

1-Enviar un caracter a la pantalla y avanzar el cursor una posición.

2-Leer el estado que tiene el puerto serie a fin de detectar la presencia de un caracter listo y leerlo.

3-Enviar un caracter a través del puerto serie.

4-Detectar cuando se ha presionado una tecla y lectura del código de la tecla oprimida.

Para el manejo del puerto serie de la PC se utilizaron las funciones de la rutina de interrupción 14H del BIOS. Esta rutina ejecuta diversas funciones que dependen del valor que contenga el registro AH al momento de hacer la llamada.

Si AH = 0 el valor en el registro AL es utilizado para programar el puerto serie según la tabla de la figura 5.2

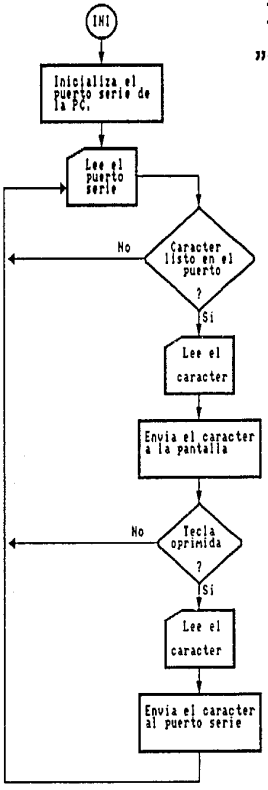


Figura 5.1 Diagrama de Flujo del Programa "TERMINAL".

Si AH = 1 el caracter en el registro AL es enviado por el puerto serie.

De la misma forma si AH = 2 se lee el puerto serie y el caracter leído se almacena en el registro AL.

Finalmente si AH = 3 entonces lo que se almacena en el registro AH es el status del puerto, según la figura 5.2
De la misma forma en que se echó mano de las rutinas de la interrupción 14H del BIOS, así también se hizo con las rutinas de la interrupción 16H. Esta rutina de interrupción contiene funciones para el manejo del teclado de la PC.

Aquí si el registro AH = 0 entonces se hace una lectura del teclado, y regresa el código ASCCI del caracter en el registro AL y el código de la tecla en el registro AH.

Si AH = 1 el acceso al puerto es para preguntar si se ha oprimido alguna tecla. En este caso :

ZF = 1 no hay caracter.
ZF = 0 existe un caracter disponible.

En la figura 5.2 se explica cómo funciona la interrupción 14h:

```
AH = 0                                  BIT DE   LONGITUD
          BPS              PARIDAD      PARADA   DE  DATO
      7    6    5         4     3        2      1     0
      -------------       -------       ---     --------
      0 0 0   110         X 0 Sin      0-1     1 0    7 bits
      0 0 1   150         0 0 Non      1-2     1 1    8 bits
      0 1 0   300         1 1 Par
      0 1 1   600
      1 0 0  1200
      1 0 1  2400
      1 1 0  4800
      1 1 1  9600

AH = 1
     AL = valor del caracter a enviar
     DX = número del puerto serie (0 a 3)
    Salida :
     AH=2 estado de operación
     bit7 incapaz de transmitir
```

Continuación

```
      bit6,0 igual que en la salida del
                estado del puerto

AH = 2
     DX = número del puerto serie (0 a 3)
     Salida :
               AL = valor del caracter recibido
               AH = estado de operación
               bit7 datos no recibidos
               bits4,1 igual que en la salida del
               estado del puerto
AH = 3
     DX = número del puerto serie (0 a 3)
     Salida :
            AH = estado de control de línea
               bit7 tiempo fuera
               bit6 reg. de transmisión vacío
               bit5 reg. espera de transmisión vacío
               bit4 detección de suspensión
               bit3 error de marco
               bit2 error de paridad
               bit1 error de desbordamiento
               bit0 datos del receptor listos
            AL = estado del modem
               bit7 detección de línea de recepción
               bit6 indicador de anillo
               bit5 DSR
               bit4 CTS
               bit3 detector cambio señal recepción
               bit2 borde de salida del anillo
               bit1 DSR cambio
               bit0 CTS cambio
```

Figura 5.2 Encabezado de la Interrupción 14h.

El manejo de la información en la pantalla se llevó al cabo a través de las rutinas de interrupción del DOS. Estas rutinas así como las del BIOS realizan numerosas funciones que dependen de los parámetros que se le den. A la interrupción 21H se le ha asignado la mayoría de las funciones de DOS.

En la sección de apéndices se muestra una tabla completa de las tareas asignadas a cada una de las funciones de la interrupción 21H.

Los resultados que obtuvimos con la creación de este programa fueron que la PC se comportara tal cual lo habíamos planeado, como una terminal del SDK-86. Así que nos era posible comandar el SDK-86 a través de la PC, es decir, desplegar el

contenido de la memoria, sustituir los valores almacenados en determinada localidad, observar los registros, las banderas, etcétera.

Del Disco al Kit

El segundo paso en el intento de comunicar la PC con nuestro circuito fue la creación de un programa que fuera capaz de leer un archivo grabado en un disco de la PC, enviarlo hacia el SDK-86 y finalmente correrlo.

Tal programa llevó el nombre de CARGA y está basado en el programa anterior TERMINAL. Este programa contiene una rutina que le permite leer un archivo, almacenarlo en un buffer y finalmente transmitirlo.

Para lograr lo anterior se volvió a echar mano de las funciones de la rutina de interrupción 21H del DOS. En la figura 5.3 se presenta el algoritmo del programa CARGA el cual ya es capaz de leer un programa desde disco y enviárselo al SDK-86.

Queremos resaltar la importancia de esto último debido a que permite cargar y ejecutar programas en un tiempo considerablemente menor. Durante la operación normal de un kit de diseño como lo es el SDK-86, los programas son introducidos directamente a través del teclado numérico con el que cuentan. Esta operación consume una gran cantidad de tiempo, sobretodo cuando se trata de programas grandes (como era nuestro caso), además existe una probabilidad importante de que se cometa un error al oprimir una tecla equivocada y con esto escribir una instrucción por otra.

Lo anterior lo solucionamos utilizando el programa monitor serie con el que cuenta el kit SDK-86 y escribiéndole al programa CARGA la rutina por medio de la cual lee un archivo de disco y lo transmite por el puerto serie de la PC.

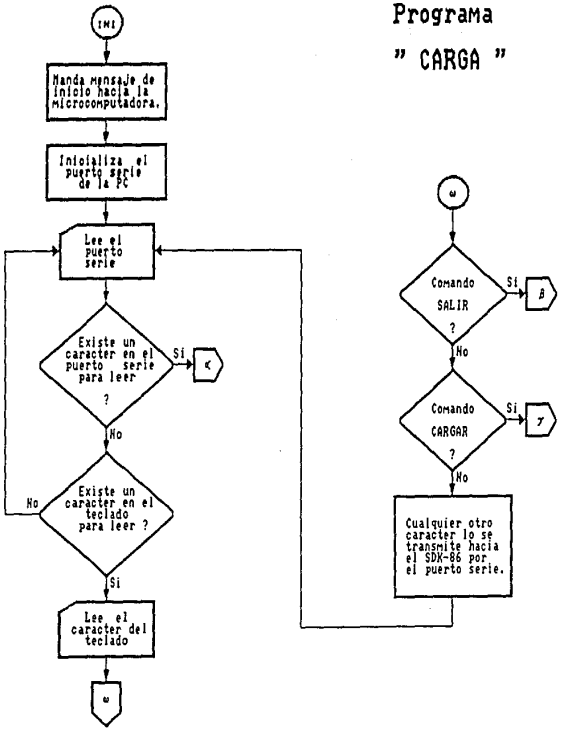


Figura 5.3a Diagrama de Flujo del Programa "CARGA".

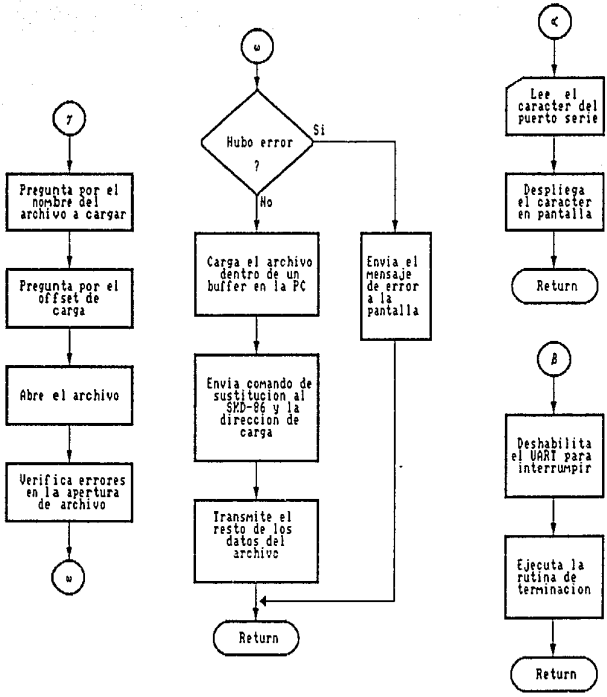


Figura 5.3b Diagrama de Flujo del Programa "CARGA".

Comunicándose con el 3GJ

Una vez que los programas probados en el kit SDK-86 funcionaron correctamente requerimos probarlos en el circuito 3GJ. Para tal propósito elaboramos un programa monitor que corriera en el circuito 3GJ. La descripción del programa monitor 3GJ se detallará a continuación. Es importante mencionar que el programa CARGA funciona de la misma forma como si lo hiciera con el monitor del circuito SDK-86.

Como anticipadamente puede suponerse, haber logrado lo anterior nos abrió un campo de pruebas extraordinariamente amplio sobre nuestro circuito.

Para probar el 3GJ y cargar su software fue necesario crear un programa MONITOR. El programa por el alcance que se deseaba que tuviera, no sólo era cuestión de ensamblarlo en la PC y cargarlo en la ROM del 3GJ, sino que era necesario después de ensamblarlo, hacerlo correr con el debugger de la PC, un debugger llamado CODE VIEW,el cual por su versatilidad hacía que las pruebas fueran hasta cierto punto sencillas y muy completas. Después de que se logró hacerlo correr en CODE VIEW, dicho programa fue instalado en el SDK 86 con ayuda de un monitor también desarrollado por nosotros mismos y una vez que en este circuito funcionó satisfactoriamente, se instaló en el 3GJ y se hizo funcionar completamente. Poco tiempo después se le fueron haciendo modificaciones al programa y se obtuvieron versiones hasta la 1.3 que es la que finalmente presentamos.

Por tanto, el monitor 3GJ es el resultado de una serie de esfuerzos en programación que comenzaron desde saber utilizar el puerto de la PC, pasando por el estudio del circuito SDK 86 y elaboración de un monitor exclusivo para éste.

El programa fue realizado en su totalidad en lenguaje ensamblador y como casi el resto de los programas se ensambló con el MASM 4.0, ligado con el LINK.EXE y transformado con el EXE2BIN.EXE.

A continuación se muestra el diagrama de flujo del MONITOR 3GJ (figura 5.4):

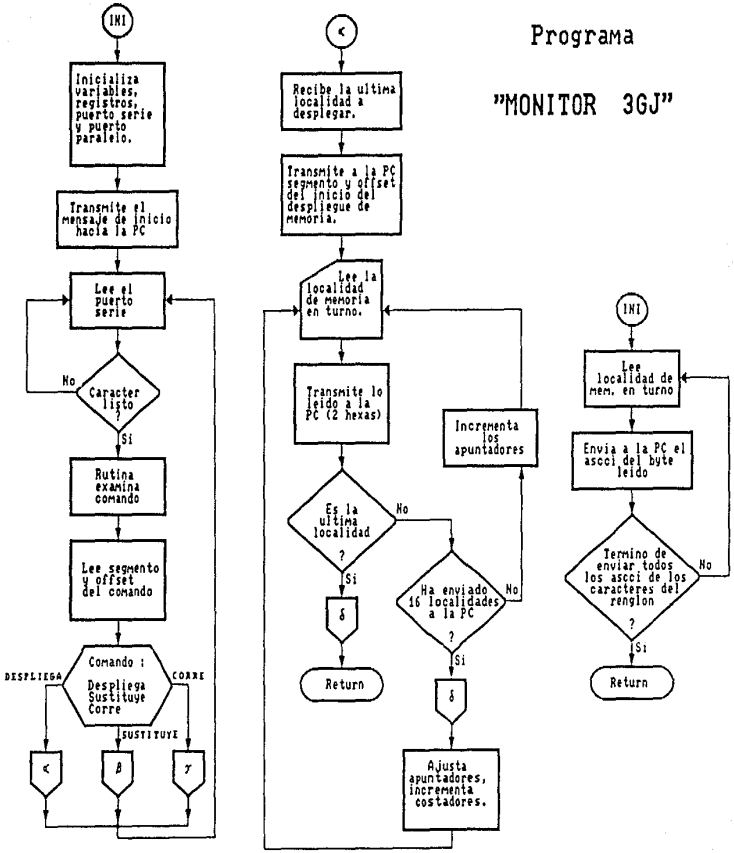


Figura 5.4a Diagrama de Flujo del Programa "MONITOR 3GJ".

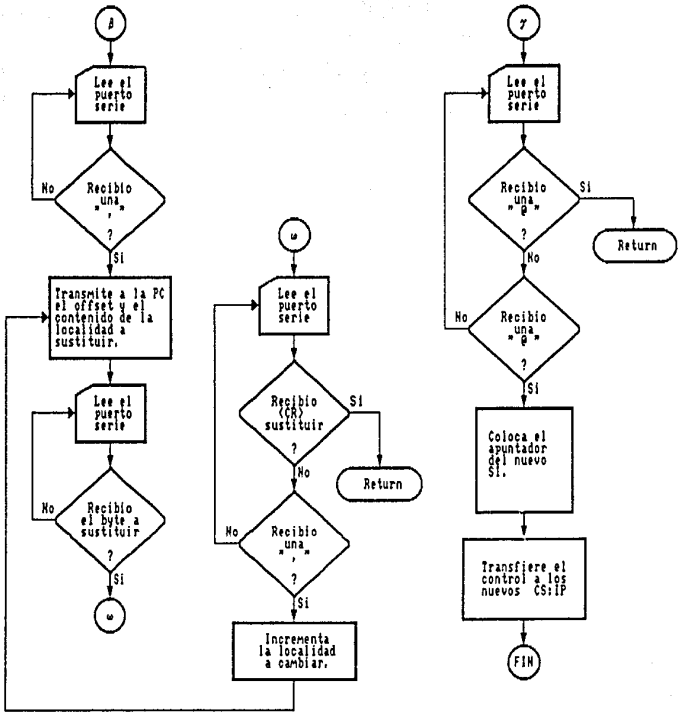


Figura 5.4b Diagrama de Flujo del Programa "MONITOR 3GJ".

El objetivo del monitor 3GJ es desplegar una o varias localidades de memoria (según se le indique), sustituir una o varias localidades de RAM y poder transferir el programa a un nuevo CS e IP.

El programa consiste en una serie de anidamientos, tiene desde luego un procedimiento principal, tres procedimientos secundarios y otro conjunto de rutinas muy elementales las cuales son llamadas repetitivamente desde las principales e inclusive desde ellas mismas. Podrá observarse en el algoritmo que el procedimiento principal está al principio, después continúan los secundarios y por último aquellas rutinas que llamamos muy elementales.

El procedimiento principal comienza enviando el mensaje de inicio a la PC y además inicializa todas las variables y constantes (que son las direcciones de los puertos) que se utilizarán en el programa.

Como es de entenderse, el programa también debe inicializar los registros del microprocesador haciendo inclusive un llamado a la subrutina INICIALIZA, en la que se inicializa el valor de varios de los registros del microprocesador.

Asimismo, es necesario inicializar el puerto serie del 3GJ, para lo cual debemos garantizar que dicho puerto se encuentra listo para recibir la palabra de comando en donde se envía el bit de RESET. Puede ser que de entrada el circuito se haya programado previamente y se encuentre listo para recibir esta palabra, es decir, que ya se le haya enviado la palabra de modo (entonces a continuación se le debe de enviar la palabra de RESET), o bien que el circuito se acabe de energizar y entonces estará listo para recibir la primera palabra, o sea, la de modo. Como estas dos situaciones pueden suceder indistintamente, las prevemos enviando tres palabras (00) al puerto serie sucesivamente, las cuales en sí no significan nada, pero nos aseguran que la cuarta palabra a enviar sea una de comando en la que se le de el RESET al circuito. A continuación ya podemos enviar una nueva palabra de modo al 8251 y otra nueva de comando donde le indiquemos que va a trabajar con palabras de 8 bits, sin paridad, con 1 STOP BIT y en modo FULL DUPLEX. Adicionalmente notemos que cada vez que se envía una palabra al puerto serie se corre un loop de tiempo para permitir que el circuito asimile la orden enviada.

A continuación, también se inicializa el PPI 8255 en modo 0 haciendo que el puerto A y C sean salidas. Esto no requiere de mayor explicación.

Seguidamente, el 3GJ envía el siguiente mensaje a la PC indicando que se encuentra listo para trabajar:

```
> MONITOR 3GJ VERSION 1.3
        Marzo 1989
>_
```

Para esto hace uso de unas cuantas subrutinas como la llamada TXSTRING, la cual requiere que se coloque el SI como apuntador en donde empieza el mensaje y en CX la longitud que va a tener éste. Esta rutina se encargará entonces de enviar todo el mensaje.

Es útil mencionar que una de las rutinas más empleadas tanto en procedimientos principales como secundarios es la TXCAR, la cual se encarga de enviar hacia la PC caracter por caracter. Esta rutina es la que hace que aparezca cada caracter y en especial podemos ver que se usa para enviar el último '>' del mensaje.

A continuación puede observarse un par de rutinas las cuales están en espera de los tres comandos esenciales : Despliega, Sustituye y Corre, en donde si no se reciben no puede avanzar el programa, y una vez recibidos los comandos es necesario que la PC envie un espacio en blanco. De esto se encarga la subrutina E_BLANCO.

Luego, es necesario guardar en las variables destinadas para ello, el segmento y offset inicial para lo cual se hace uso de subrutinas tales como E_2HEXA la cual estará en espera de que desde la PC se le envíen dos dígitos hexas, para guardarlos en una localidad de memoria. Como el CS e IP constan de 16 bits cada uno, para almacenarlos en memoria es necesario utilizar 4 localidades. por lo que el proceso anterior se repite 3 veces más. Inclusive, en el listado se menciona el número de localidad en RAM segmento 0 donde se localizan las variables. Es en este proceso en donde la PC envía el segmento y offset, entre estos dos la PC debe enviar un separador (:) el cual es esperado por la rutina E_SEPARADOR.

Las siguientes instrucciones del programa, compararán el comando recibido por la rutina E_COMANDO para averiguar a cuál de las 3 subrutinas secundarias se transferirá el programa:

Despliega
Sustituye
Corre

A continuación se explicarán estas tres subrutinas.

Procedimiento Despliega

Si el comando digitado al principio de la rutina principal fue D, entonces al iniciar la rutina despliega, por medio de ésta podrá desplegarse el contenido de una o varias localidades. Si después de digitar el comando D, el segmento y offset respectivo, se envía desde la PC un <CR>, entonces sólo se mostrará esa localidad de memoria. Si se envía un separador (:) y a continuación otro offset, entonces se desplegará un rango de localidades que abarcará los dos offsets.

Así pues, una vez que se ha indicado desde la PC la(s) localidad(es) a mostrar, con la rutina LEE_DIR, se transmite hacia la PC el offset desde donde se empezará a transmitir. En ese mismo renglón podrán desplegarse el contenido en hexadecimal y ASCII de hasta 16 localidades de memoria. Para conseguir esto, debe inicializarse un contador de 16 (10H) y un apuntador de localidades de memoria (SI), se lee el dato de memoria por medio de la rutina LEE_DATO y se transmite a la PC por medio de la rutina TX2HEXA y a continuación un espacio en blanco. En este momento es necesario saber si ésta es la última localidad a transmitir comparando apuntadores con datos previamente guardados en variables de RAM, de ser así, se efectúa un brinco a la parte de la rutina que despliega los ASCII's del contenido de las localidades de memoria, (rutina que se explicará más adelante), de lo contrario decrementará el contador de 16. Mientras no llegue a cero, incrementará el apuntador de localidades (SI) y se regresará a leer otro dato. Si el contador ha llegado a cero, entonces llamará a la rutina para enviar el contenido de localidades en ASCII, y se comenzará el siguiente renglón enviando a la PC el offset de la primera dirección de ese renglón, se inicializa el contador de 16 localidades de nuevo y se repite el proceso de enviar

contenidos de memoria.

La rutina que envía el contenido de la memoria en ASCII comienza guardando adecuadamente el apuntador de la última localidad mostrada en hexadecimal, apunta ahora a la primera localidad a mostrar en ASCII, si dicho contenido tiene un valor entre 32D y 126D enviará el ASCII correspondiente, si sale de este rango, entonces sólo transmitirá un '.' hacia la PC. Mientras no sea el último contenido a transmitir, se repetirá el proceso, cuando se haya transmitido el último contenido, entonces el apuntador para mostrar localidades en hexadecimal recuperará su valor y se terminará esta rutina.

El despliegue de un renglón debe aparecer más o menos de la siguiente manera:

234B 21 34 67.....33 47 4A I4E.....3GJ

Procedimiento Sustituye

Este procedimiento sustituye una localidad o un conjunto localidades de memoria. Esta es la rutina base para el acoplamiento del monitor con el programa carga que corre en la PC para cargar archivos desde el disco.

El procedimiento comienza esperando una coma que debe ser enviada desde la PC. En el momento de recibirla, el 3GJ transmite el offset de la localidad que va a sustituir, utilizando para esto de nuevo la rutina LEE_DIR, además transmite un espacio en blanco y a continuación con ayuda de la rutina LEE_DATO y TX2HEXA transmite a la PC el contenido de la localidad a sustituir y luego un guión.

A continuación queda en espera de un par de dígitos hexadecimales, los cuales van a quedar sustituidos en la localidad de memoria que se apunta. Para finiquitar el cambio, es necesario enviar desde la PC una ',' o un <CR>.

Si se envía una ',' incrementará el apuntador de la localidad a sustituir en uno y reiniciará el proceso. Si recibe un <CR>, entonces dará por terminada esta rutina. El programa Carga que corre en la PC se acopla perfectamente a este procedimiento y se encarga de enviar el contenido de localidades y ',' forma en que se pueden cargar programas completos, terminando dicha carga con un <CR> enviado desde

la PC.

Procedimiento Corre

Este procedimiento es el más sencillo de todos pues sólo carga un nuevo Segmento de Código y Apuntador de Programa de tal manera que pueda empezarse a correr un nuevo programa.

El procedimiento transfiere el CS e IP a otra zona de memoria con lo que se puede hacer correr un programa localizado en otra zona. Esto se hace de dos formas:

1.Se graba un salto EA en la localidad 0464H y a continuación se salta allí para que de ahí se salte a en forma absoluta a la zona donde esta el nuevo programa.

2.Se hace un salto intersegmento indirecto a través de SI el cual apunta a la loc 0465H que es donde esta el nuevo CS e IP.

Se dejó esta última forma por considerarse más versátil. Además si uno se retracta de hacer correr el programa, puede evitarlo tecleando una @ después de escribir la instrucción. Por ejemplo:

```
>C 0045:2000@
```

Si se envía un <CR> se transferirá hacia el otro programa. Es importante saber que el nuevo programa empezará con los registros AX,DS,BP,DI,SS blanqueados, SP=0500H , SI=0465H.

Rutinas Auxiliares

A continuación se enumerararán y se explicarán brevemente las rutinas anexas a los procedimientos anteriores las cuales permiten que en un momento dado se pueda hacer más corto el programa.

INICIALIZA: Es llamada desde el programa principal y desde la rutina de corre, se utiliza básicamente para blanquear los principales registros del microprocesador.

E_COMANDO: Sirve para detectar que llegue un comando correcto ya sea una D,S, o C. Es llamada desde el programa principal.
E_2HEXA: Es llamada desde el programa principal y sirve para detectar dos números hexadecimales que son tecleados en la PC.

E_SEPARADOR: Esta rutina tiene como finalidad esperar un separador que en este caso son (:)

E_BLANCO: Es llamada desde el programa principal y su finalidad es esperar un espacio en blanco y transmitirlo a la PC.

E_CR_ARR: Esta rutina detecta un <CR> o una @.

E_CR: Esta rutina sirve sólo para detectar un <CR>.

E_COMA: Esta rutina sirve sólo para detectar una (,).

E_CRP: Es llamada desde la rutina de Despliega y se usa para saber si sólo se desplegará un dato o un bloque de datos.

E_COMACR: Sólo espera de la PC un <CR> o una coma.

TX2HEXA: Transmite dos caracteres hexadecimales a la PC, pero antes los convierte a ASCII.

MANDA_DIR: Es llamada desde la rutina Lee_dir y lo que hace es transmitir a la PC el offset de inicio de la rutina despliega y un espacio en blanco de separación.

LEE_DIR: Sirve para cargar el segmento y offset en DX y SI además de llamar a una rutina para enviar a la PC la primera dirección (offset) con el que se trabajará.

LEE_DATO: Coloca en AL un dato de memoria apuntado por SI.

HEX_ASC: Convierte un dato numérico o letra mayúscula o minúscula de forma hexadecimal a ASCII.

PUNTO: Es llamada desde el final de la rutina Despliega y se utiliza para enviar el caracter ASCII a la PC siempre y cuando esté entre 32 y 126 Decimal.
ES_HEXA: Detecta si el dato recibido desde la PC es un número entre 0 y F o en su defecto 0 y f. Es llamada desde la rutina E_2HEXA.

ASC_HEXA: Convierte un dígito o caracter en mayúsculas o minúsculas recibido de la PC de ASCII a hexadecimal. Es llamada desde E_2HEXA.

DATOLISTO: Pregunta al puerto serie si llegó un caracter de la PC y lo toma.

EXCOM: es llamada desde E_COMANDO y es utilizada para validar un caracter como comando.

TXCRLF: Esta rutina es complementada por TX_STRING,su finalidad es transmitir un <CR LF> el cual se encuentra grabado en ROM, por eso antes de llamar a TXSTRING se pone un contador de 2 en CX y SI se pone donde empieza el mensaje.

TX_STRING: Es llamada desde el programa principal dándole como datos una dirección en SI donde empieza el mensaje y la longitud del mismo dada por CX para que pueda transmitir el mensaje.

TXCAR: Es utilizada siempre que se quiere enviar algo por el puerto serie.

Versatilidad para Monitorear

Cuando logramos comunicarnos perfectamente bien con el circuito 3GJ a través del programa Carga, decidimos darle versatilidad a la comunicación. Elaboramos un programa similar al programa Carga pero que fuese más versatil. El nuevo programa tendría que ser capaz de manejar la pantalla, de establecer colores, ventanas, sonidos, sobretodo que trabajase a mayor velocidad de transmisión y recepción. Tales características harían de este programa una herramienta más amigable para el manejo del programa monitor del circuito 3GJ. Este programa llamado LAB está codificado en el lenguaje de programación Pascal, compilador de la versión 5.0.

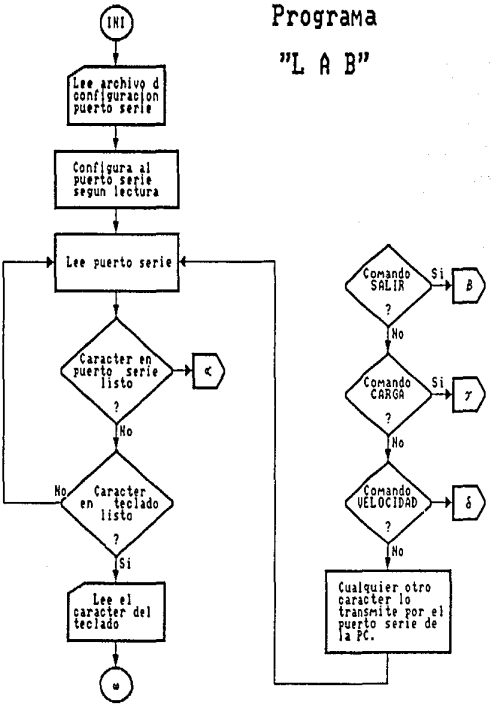


Figura 5.5a Diagram de Flujo del Programa "LAB".

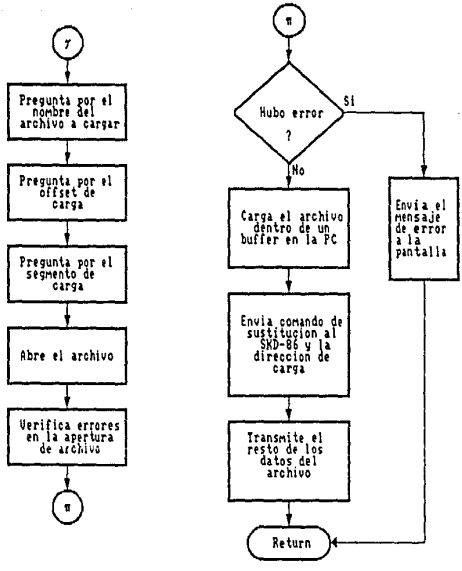


Figura 5.5b Diagrama de Flujo del Programa "LAB".

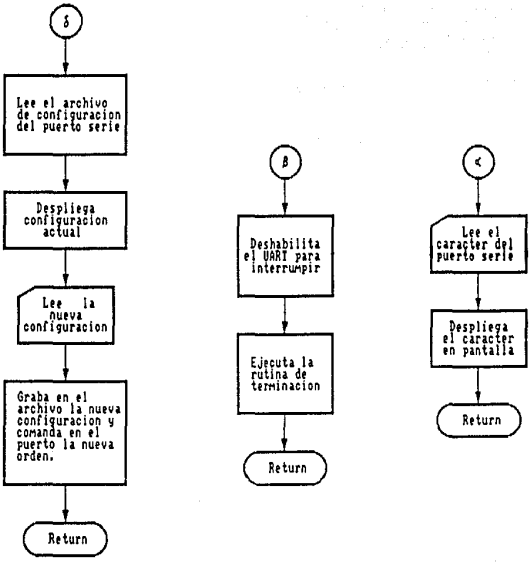


Figura 5.5c Diagrama de Flujo del Programa "LAB".

Las rutinas básicas son las mismas en las que se basan los programas "terminal" y "carga" sólo que al escribirlas en un lenguaje de alto nivel su manejo resulta mucho más sencillo.

El diagrama de flujo del programa LAB es muy similar a los de los programas anteriores y se presenta en la figura 5.5.

5.3 Programas Definitivos

En el diseño final se tienen tres programas en el circuito y uno en las PC's de la red. Los del circuito: uno se está ejecutando todo el tiempo y dos se ejecutan sólo cuando es necesario y son activados por medio de interrupciones de Hardware.

El primero de estos programas es el PRINC.ASM y se encarga del arranque de todo el sistema y del control de la impresora serie. En la sección correspondiente a este programa explicamos la mayor parte de las variables utilizadas.

El segundo programa es el de atención a las computadoras conectadas en la red, RECIBE.ASM. Es activado por medio de una interrupción normal (INTR) la cual indica que se ha recibido un caracter. La rutina toma este caracter y hace lo que sea pertinente según el caso: Si es el primer caracter de un archivo, si es un caracter de control, si con este caracter se inicia un nuevo bloque, etcétera.

El tercer programa se encarga de atender a las impresoras paralelo, se llama IMPRIME.ASM. Lo activa una interrupción del tipo NO MASCARABLE (NMI), lo que quiere decir que tiene prioridad sobre las otras interrupciones. Es activado cada vez que la impresora avisa que ya recibió un caracter y está lista para recibir el siguiente. En este programa se busca y se transmite el siguiente caracter, o bien, si el archivo ya se terminó, se busca el siguiente archivo de la lista que debe ser impreso en la impresora que interrumpió.

NOTA: Como la computadora no nos envía una marca de fin de archivo [EOF] nosotros definimos una palabra de comando con la cual nos puede indicar una de dos cosas:

- Fin de Archivo.

- Cuál es la impresora que quiere usar.

Esta palabra de comando es enviada por medio del programa de apoyo que corre residente en la memoria de la computadora. Su estructura es la siguiente:

- 3 (tres) bytes clave, escogidos entre los caracteres no desplegables de la tabla ASCII.

- 1 (un) byte cuyos últimos dos bits indican:

00 = Impresora serie
01 = Impresora paralelo 1
10 = Impresora paralelo 2
11 = Fin de Archivo

El Programa Principal. PRINC.ASM.

Este programa realiza varias funciones:

1) Memoria:

Revisa toda la memoria que es posible direccionar (1 MB) y elabora un mapa de memoria que nos permita en lo sucesivo saber de qué disponemos y de qué no. Cabe hacer notar que gracias a la forma en que se hace este manejo de memoria no es necesario conectar bloque de circuitos RAM de un tamaño específico, ni necesariamente todos en direcciones contiguas, de este modo podemos poner cualquier número de circuitos y en cualquier posición. El programa los localiza y marca como disponibles.

La revisión la hacemos escribiendo en la memoria y después leyendo lo que acabamos de escribir. Nuestra unidad básica para el uso de la memoria es el bloque de 512 bytes, de modo que si nos encontramos un error en un bloque todo el bloque queda deshabilitado. Es por esto que podemos poner circuitos RAM en cualquier posición, simplemente los huecos serán marcados como deshabilitados.

La Tabla de Memoria tiene la siguiente estructura:

a) Tiene 2048 elementos, uno por cada bloque de 512 Bytes,
*(2048 * 512 = 1 MB).*

b) El bit 0 nos indica si el bloque está disponible o no, es decir, si al momento de hacer la revisión inicial este bloque funcionó adecuadamente o no. Un valor de 0 (cero) indica que NO se dispone de este bloque. En cambio un valor de 1 (uno) indica que SI lo podemos usar.

c)El bit 1 nos indica si el bloque está ocupado o no. Si durante la operación un bloque ha sido llenado con los datos de un archivo decimos que está ocupado. Si un bloque no ha sido utilizado o bien ya se utilizó pero su contenido ya fue impreso, el bloque NO está ocupado. El valor 0 (cero) indica que NO podemos usar el bloque porque está ocupado. El 1 (uno) por su parte indica que SI podemos usar el bloque, o sea, que no está ocupado.

d)El bit 2 sirve para señalar si un bloque que ya ocupado está o no, listo para ser impreso. Si tiene una valor de 0 (cero), el bloque no está listo para ser impreso. Si tiene un valor de 1 (uno) entonces el bloque ya está listo para que lo tome la rutina de impresión.

Durante el lapso de tiempo en que el bloque está siendo recibido este bit permanece en 0. Una vez que se ha recibido completo o que se recibió la marca de fin de archivo el programa de recepción lo pone en 1.

De este modo nunca podrá estar encendido el bit1 si no está encendido el bit0. Y a su vez el bit2 nunca estará encendido si el bit1 no lo está.

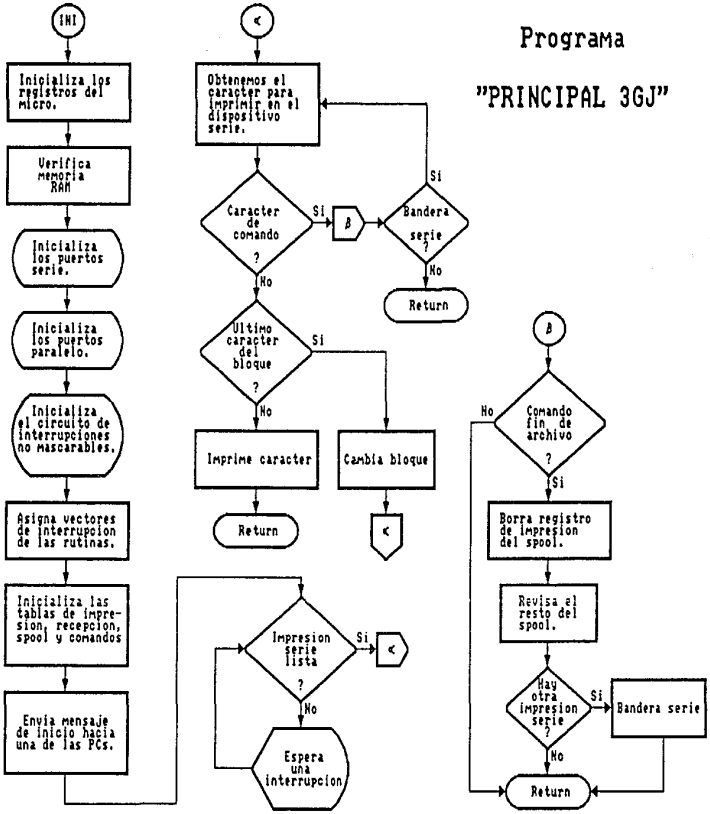


Figura 5.6 Diagrama de Flujo del Programa "PRINCIPAL 3GJ".

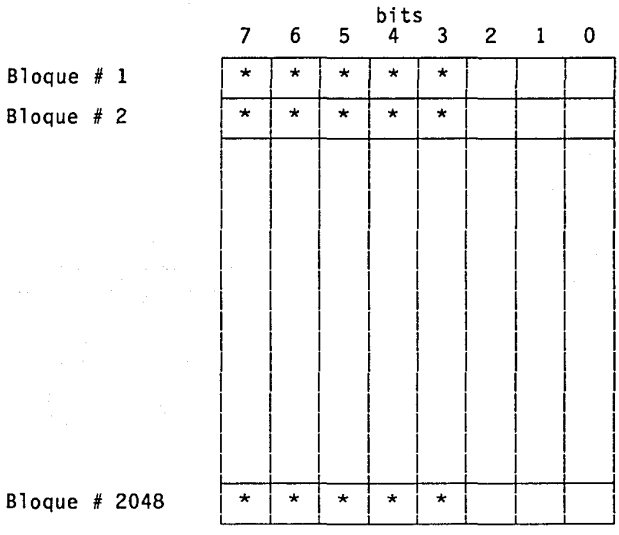


Figura 5.7 La Tabla de Memoria.

La estructura de un bloque de memoria es como sigue: Los primeros 508 bytes se utilizan para guardar información, los últimos contienen, en la mayoría de los casos, cuál es el siguiente segmento del archivo.Los otros casos son:

-Cuando el bloque en cuestión es el último del archivo los cuatro bytes están en blanco.
-Cuando coincide que la marca de fin de archivo (4 bytes) ocupa este espacio porque

la información ocupa el espacio disponible hasta el byte 507.

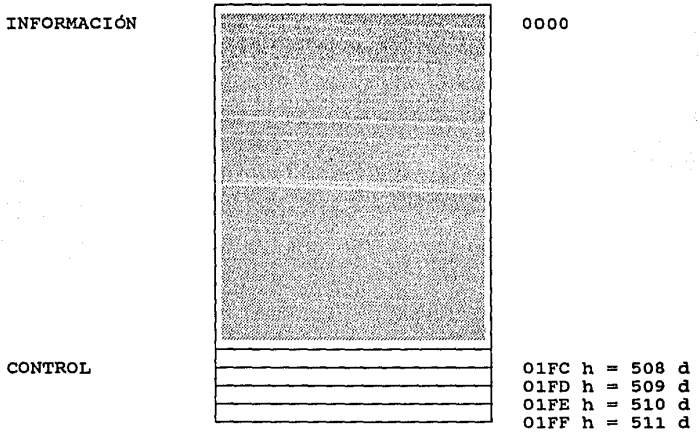


Figura 5.8 Diagrama de un bloque de Memoria.

Para simplificar el manejo de la información hacemos que cada bloque corresponda a un segmento. De este modo en vez de manejar 16 segmentos de 64K (máximo tamaño posible de un segmento), manejamos 2048 segmentos de 512 bytes cada uno.

Es muy sencillo obtener, a partir del número de bloque (11 bits), el segmento correspondiente (16 bits). Simplemente hacemos un corrimiento hacia la izquierda de 5 bits insertando ceros del lado derecho. De este modo al hacer la suma del segmento con el offset se utilizan los 20 bits que tenemos en total.

Ejemplo:

```
      # bloque:    935 d =   3A7h  =          011 1010 0111 b
   Corrimiento:           7500 h  =    0111 0100 1110 0000 b
Máximo offset :    511 d =  1FF h  =         1 1111 1111 b
       Sumando:
                                          0111 0100 1110 0000 b
                                    +        1 1111 1111 b
                                    ---------------------------
  Dir. efectiva:                    0111 0100 1111 1111 1111 b
```

Casos posibles :

```
0000 0000 b = 0 d => No disponible.
0000 0001 b = 1 d => Disponible, desocupado, no listo
                     para ser impreso.
0000 0011 b = 3 d => Disponible, ocupado, no listo para
                     ser impreso.
0000 0111 b = 7 d => Disponible, ocupado, listo para
                     ser impreso.
```

Nótese que los tres programas necesitan un área de datos y STACK en RAM, y la estructura que acabamos de describir reserva el total de la memoria a los archivos de impresión que vamos a manejar.

Lo que hacemos para obtener espacio para la misma tabla de memoria y el resto de las variables utilizadas es marcar los bloques de la parte baja de la memoria como NO DISPONIBLES y es en ellos donde trabajamos.

Existe otra excepción en el mapa de memoria. En la parte mas alta de la memoria se localiza la memoria ROM, donde están los programas. Los últimos 32K (64 Bloques) de la memoria tampoco están disponibles para archivos, cuando el programa prueba esta zona de la memoria los marca como si fueran circuitos dañados porque cuando escribe sobre ellos no lee lo que escribió.

En la siguiente figura vemos el mapa de memoria.

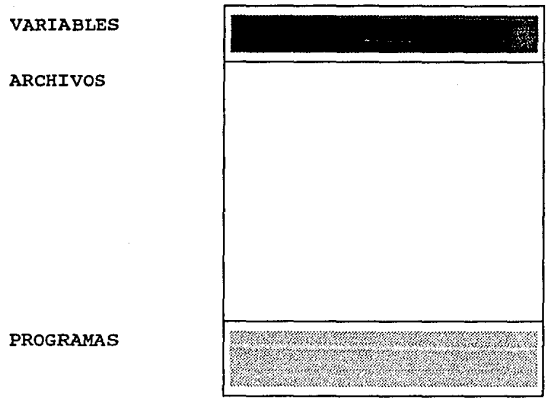


Figura 5.9 Mapa de Memoria.

2) Puertos.

Inicializa los puertos, tanto los serie como los paralelo. Los puertos serie son inicializados a 9600 bauds que es nuestra velocidad estándar. Los puertos paralelos se inicializan sólo para salida.

3) Interrupciones.

Escribe en las posiciones adecuadas de la parte baja de la memoria las direcciones a las que debe ir el procesador para atender a ambos tipos de interrupciones.

Para la interrupción de las impresoras forzosamente se usa el vector 2 (dos) por ser

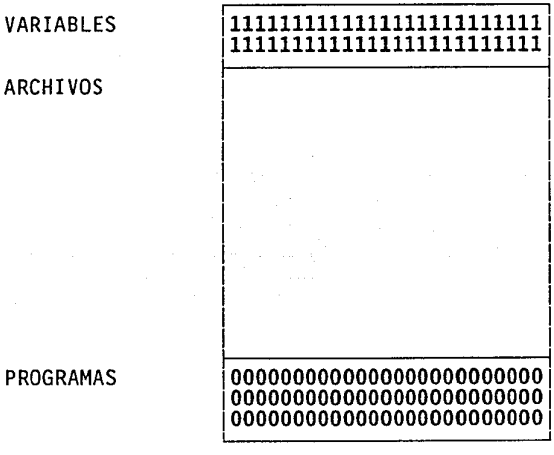


Figura 5.9 Mapa de Memoria.

2) Puertos.

Inicializa los puertos, tanto los serie como los paralelo. Los puertos serie son inicializados a 9600 bauds que es nuestra velocidad estándar. Los puertos paralelos se inicializan sólo para salida.

3) Interrupciones.

Escribe en las posiciones adecuadas de la parte baja de la memoria las direcciones a las que debe ir el procesador para atender a ambos tipos de interrupciones.

Para la interrupción de las impresoras forzosamente se usa el vector 2 (dos) por ser

éste el exclusivo para una NMI.
Para las interrupciones mascarables usamos cualquier vector de los disponibles.

4) Tablas.

En el sistema se usan varias tablas que son inicializadas con 0 (cero). Las tablas son:

a) Tabla de Recepción:

Tiene 16 renglones (0 -> 15), uno por cada computadora conectada:

REC_SEG.
Es el segmento en el que estamos recibiendo, corresponde al número de bloque pero con un corrimiento de cinco bits a la izquierda. En 0 (cero) indica que no estamos recibiendo de esta computadora.

REC_OFF.
Es el offset, dentro del segmento, donde se va a guardar el siguiente caracter que llegue.

REC_SEGINI.
Es el segmento en donde empieza el archivo que se está recibiendo en este momento.

REC_COM.
Es el número de caracteres de comando que hemos recibido. Varía de 0 a 3 porque la palabra de comando es de 4 bytes.

REC_IMPR.
Es el número de la impresora por la que se van a imprimir los archivos de esta computadora:

0 = Impresora serie.
1 = Impresora paralelo 1.
2 = Impresora paralelo 2.

REC_SP.
Es el renglón, dentro de la tabla de impresión (SPOOL), en donde se encuentra el archivo que estamos recibiendo.

b) Tabla de Impresoras:

Maneja 3 renglones (0 ->2), uno por cada impresora conectada. Una serie y dos paralelo:

0 = Impresora serie.
1 = Impresora paralelo 1.
2 = Impresora paralelo 2.

TAB_IMPRSEG.
Es el segmento en donde estamos imprimiendo, si tiene 0 (cero) es porque esta impresora está detenida.
TAB_IMPROFF.
Es el offset, dentro del segmento, en donde se encuentra el siguiente caracter a imprimir.

c) Tabla de Impresión (SPOOL):

Tiene 50 renglones, es el número máximo de archivos que podemos manejar simultáneamente. Para esta tabla manejamos apuntadores:
Un apuntador de recepción que nos indica cual es el siguiente lugar disponible de la tabla. Tres apuntadores de impresión, nos indican en dónde va cada una de las impresoras dentro de la cola

SP_SEGINI.
Es el segmento en donde empieza el archivo.

SP_IMPR.
Es la impresora por la que se debe imprimir el archivo.

Nunca los apuntadores de impresión sobrepasan al de recepción. Cuando el apuntador de recepción llega al final de la tabla vuelve a empezar al principio de modo que se tiene una lista circular.

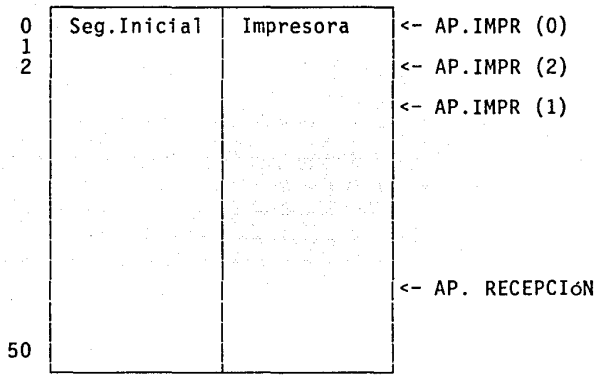


Figura 5.10 Tabla de Impresión (SPOOL).

d) Tabla de Caracteres de Comando.

Para facilitarnos las comparaciones cuando queremos saber si la computadora nos esta mandando una palabra de comando o si es algo del archivo (Pudiera parecerse a una serie de caracteres de control de un graficador).

Guardamos aquí la serie de 4 bytes con el indicador de fin de archivo al final (los bits 0 y 1 del último byte son "1").

5) Habilita las interrupciones.

6) Permanece en un ciclo esperando a que la impresora serie requiera atención. La forma en que lo sabemos es porque la rutina de recepción pone un valor diferente de 0 (cero) en la variable LISTO_SERIE.

7) Atención a la impresora serie.

Para comunicarnos con la impresora serie utilizamos el protocolo XON-XOFF. Como en este programa no estamos recibiendo, lo que hacemos, es verificar constantemente la variable LISTO_SERIE, mientras tenga el valor 3 es que podemos seguir enviando caracteres, cuando tome un valor distinto de tres es porque la rutina de recepción detectó que la impresora tiene el buffer lleno.

La rutina se queda esperando a que nuevamente se pueda transmitir y continúa hasta que se acaba el bloque o el archivo:

Si se acaba el bloque, continúa en el que sigue, tomando la dirección de los últimos bytes del bloque terminado. Antes de empezar a imprimirlo verifica que el bloque en cuestión esté listo para ser impreso.

Si se acaba el archivo busca en la tabla de impresión (SPOOL) por si hay más impresiones para esta impresora. Cuando busca otros archivos en la tabla de impresión se detiene al momento de alcanzar al apuntador de recepción debido a que se tiene la seguridad de que no hay más archivos esperando ser impresos en la impresora serie.

Tanto cuando se acaba el bloque, como cuando se acaba el archivo se quita la marca de ocupado, y de listo para imprimir en el renglón correspondiente al bloque en el mapa de memoria.

El Programa de Recepción. RECIBE.ASM.

Este programa es activado por interrupción. Al momento de empezar lo único que sabemos es que acaba de llegar un caracter desde una de las computadoras de la red. Lo que debe hacer este programa es decidir qué hacer con él, y hacerlo.

Este programa es el más complicado de los del circuito, intentaremos explicar la

forma en que opera ayudándonos de textos escritos en negrilla, que servirán como si fueran etiquetas o nombres de párrafo en un programa.

Recepción Adecuada. *Leemos la palabra de status del puerto serie para estar seguros de que no hubo error en la transmisión y que el dato ya puede ser leido.*

Una vez que se ha leído el dato del puerto serie lo primero que hay que hacer es verificar cuál de las computadoras interrumpió o si fue la impresora serie.

Impresora Serie. *Si fue la impresora saltamos a una rutina al final del programa en donde verificamos lo que se recibió y actualizamos el valor de una variable. Cuando es un comando de detener es que la impresora no puede recibir, modificamos el valor de una variable y finaliza. Si se recibió un comando sigue se vuelve a modificar la bandera y finaliza.*

Computadora de la Red. *Si fue una de las computadoras checamos si en los caracteres anteriores ya estábamos recibiendo parte de la palabra de comando.*

Si ya estábamos recibiendo parte de la palabra de comando puede ser que el dato que acaba de llegar sea el que sigue de los cuatro o que se trataba solamente de una cadena parecida.

Si el dato que acaba de llegar no coincide con el que se espera entonces guardamos en el archivo la cadena que se recibió.Por otro lado si el dato coincide puede ser que sea el último de la cadena o que solo sea parte.

Ultimo Comando. *Si no es el último simplemente incrementamos el contador de caracteres de comando y saltamos al final del programa. Si efectivamente es el último caracter tenemos la opción de que la cadena sea el fin de archivo o solamente sea una selección de impresora. Si es esto último simplemente actualizamos el dato en la tabla de recepción.*

Fin de Archivo. *Si es un fin de archivo insertamos un salto de hoja y un retorno de carro y escribimos la cadena de comando en su lugar. Además marcamos el bloque como listo para imprimir y liberamos la tabla de recepción para cuando la computadora con la que estamos trabajando mande su siguiente archivo. De aquí*

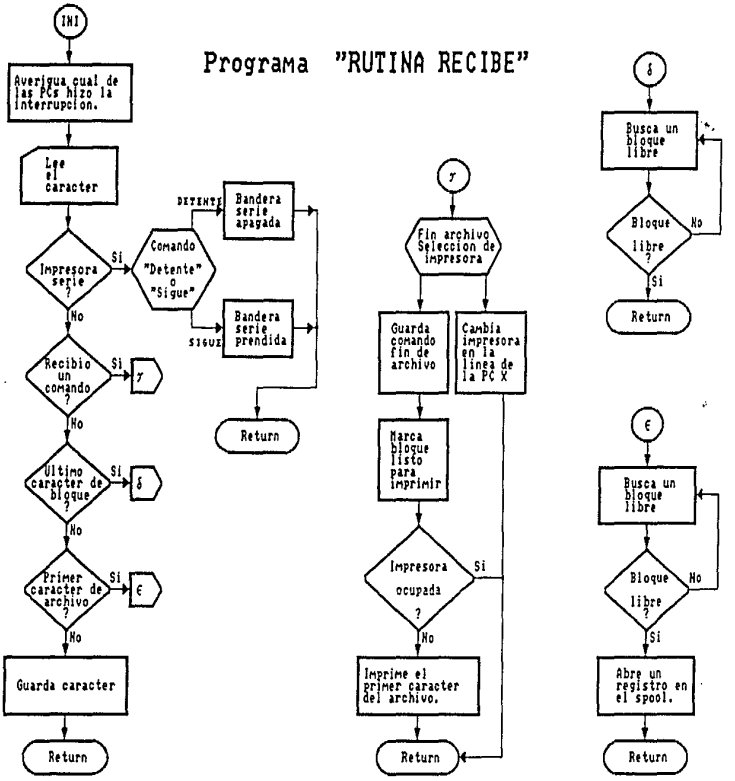


Figura 5.11 Diagrama de Flujo del Programa "RECIBE".

decidimos si comienza la impresión

Primer Caracter. *Cuando no hemos recibido ya caracteres de comando puede ser que el que acaba de llegar sea el primero. Si lo es incrementamos el contador de caracteres de comando y finaliza. Si no es el primero de la palabra de comando consideramos que se trata de un caracter normal, sólo guardamos el caracter.*

Guardando un Caracter. *Si se trata del primero hay que localizar un bloque desocupado en la memoria. Debemos también actualizar los datos en la tabla de impresión (SPOOL) y de recepción correspondientes al archivo que ahora empieza. Adelantamos el apuntador de recepción. Finalmente, guardamos el dato en su lugar y saltamos al fin del programa.*

Si ya está asignado un bloque a esta recepción. Lo que debemos verificar antes de guardar el caracter es si no se ha llenado el bloque. Si aún no es el final del bloque simplemente guardamos el caracter, actualizamos la tabla de recepción y saltamos al final del programa.

Si se llenó el bloque, marcamos el bloque anterior como listo para ser impreso. Actualizamos en la tabla de recepción y guardamos el caracter.

Decide Impresión. *Primero que nada vemos si la impresora está ocupada o no. Si no está haciendo nada mandamos el primer caracter del bloque que acabamos de terminar de recibir, actualizamos la tabla de impresoras y saltamos al final de la rutina.*

Si la impresora está ocupada puede ser que está funcionando o que imprime un archivo pero el bloque que sigue para imprimir aún no ha sido recibido completo. Si no está esperando nada saltamos al final de la rutina.

NOTA:

Cuando mandamos a imprimir tenemos que ver de cuál de las impresoras se trata. Si es la impresora serie aquí no mandamos nada a imprimir, solamente modificamos la bandera serie. Si es alguna de las paralelo hay que escoger entre los dos puertos y fijarse al momento de mandar la señal de control adecuada.

El Programa de Impresión. IMPRIME.ASM.

El programa es activado como atención a una interrupción de tipo NMI, es decir, la de más alta prioridad. Lo que hace es atender a las impresoras paralelo. La NMI es activada por la impresora cuando ya está lista para recibir un nuevo caracter.

Primeramente identificamos cual de las dos impresoras nos interrumpió. Puede darse el caso de que la señal que recibamos se deba a que encendieron la impresora y no realmente que estemos imprimiendo un archivo. Filtramos aquí esa eventualidad.

Una vez que sabemos quien nos interrumpió traemos de memoria el caracter a imprimir. Este caracter que traemos puede ser el primero de la cadena de fin de archivo, si es así verificamos si los tres que siguen lo son. Si estámos a medio bloque simplemente hay que mandar a imprimir el caracter y saltar al final del programa. Si llegamos al final del bloque liberamos el bloque en cuestión para que pueda ser reutilizado. Si el siguiente bloque está listo para ser impreso, comenzamos su impresión y finaliza. Si no está, simplemente ejecutamos lo anterior.

Al encontrar el fin de archivo, se tiene que averiguar si existe otra impresión en espera. Si al ir avanzando llegamos a otro archivo para esta impresora nos detenemos ahí, actualizamos apuntador de impresion y tabla de impresoras y mandamos el primer caracter. Si no hallamos nada, regresamos.

El Programa Residente para las Computadoras de la Red

El circuito multiplexor 3GJ es un equipo que establece comunicación con microcomputadoras personales (PC). Esta comunicación se lleva a cabo entre los equipos gracias a dos programas. Uno de ellos es el programa que corre en el multiplexor 3GJ y que es además el encargado de administrarlo. En el otro lado, las PC establecen comunicación con el 3GJ gracias a un programa que corre en cada una de estas computadoras.

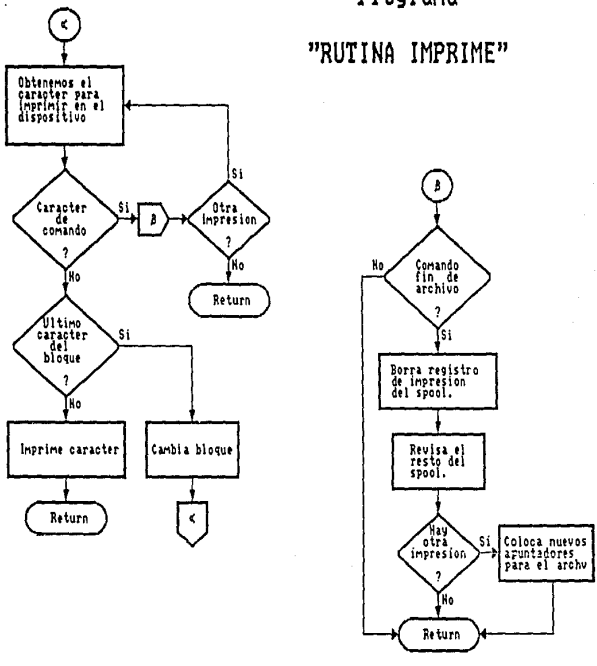


Figura 5.12 Diagrama de Flujo del Programa "IMPRIME".

Para dar transparencia al usuario de la red 3GJ, el programa que corre en los equipos PC lo hace en forma residente. El programa residente detecta cuando se hace uso del puerto de impresión y procesa la información que se desea enviar a través de ese puerto para enviarla bajo las reglas de la red 3GJ. Una vez que se ha detectado la utilización del puerto de impresión se despliega en pantalla un mensaje al usuario, que le permite recordar :
Que está trabajando en la red de impresión 3GJ, que puede llamar al menú de opciones de la red 3GJ y que necesita enviar el fin de archivo de su documento.

El programa residente de la red de impresión 3GJ cuenta con un menú a través del cual, permite al usuario de la red seleccionar el dispositivo impresor en el desea su documento (la red 3GJ es capaz de soportar hasta 3 dispositivos impresores). Este menú de opciones también es utilizado por el usuario para el envío de su fin de archivo. Y finalmente como opción adicional, el menú permite al usuario de la red utilizar un dispositivo impresor no compartido por los demás usuarios y conectado directamente a esa computadora en específico. Este último comando permite imprimir un documento que por ejemplo, requiera de un determinado tipo de papel, diferente al papel que se utiliza en los impresores que comparten la red. O bien, que sea impreso en otra calidad diferente a la de los impresores compartidos.

En la figura 5.9 se hace la descripción del algoritmo del programa residente que opera en la red 3GJ.

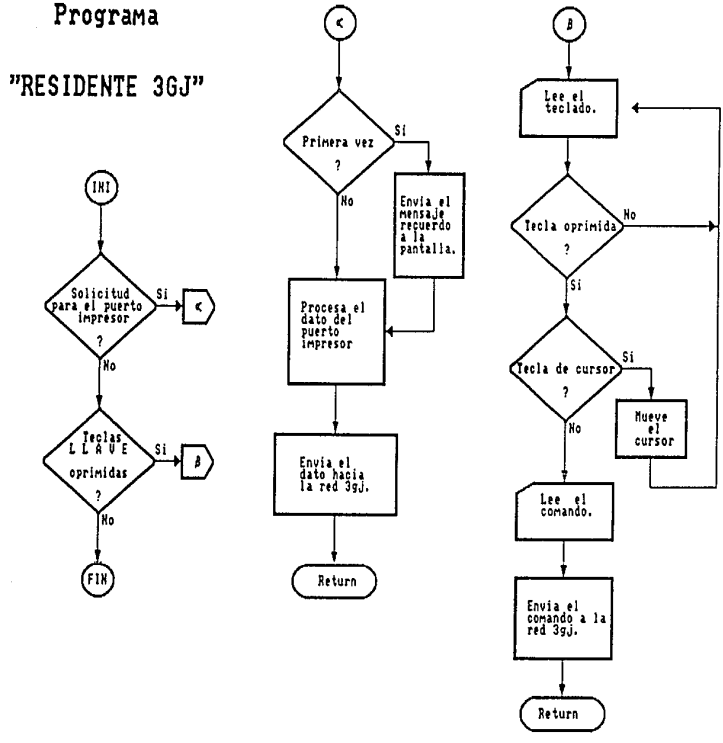


Figura 5.13 Diagrama de Flujo del Programa "RESIDENTE 3GJ".

CONSTRUCCION

Introducción

A continuación describimos brevemente el proceso de diseño de tarjetas de circuito impreso y contrucción de nuestro circuito por medio de Wire Wrap (alambre enrollado).

6.1 Diseño de Tarjetas

Los diagramas lógicos de las tarjetas fueron dibujados en el paquete Capture Design System de la minicomputadora HP9000 serie 300 e impresos en un graficador Hp7585.

Ya teniendo los planos se revisaron y modificaron las configuraciones en cada tarjeta de una forma más ágil y segura. Sin embargo, no nos fue posible utilizar todos los recursos que existen para la HP9000 (es capaz de simular los sistemas dibujados en ella), debido a las siguientes razones:

+ No había suficiente espacio en su disco duro para el paquete simulador (Hilo).

+ En la biblioteca de la HP9000 no se encontraban todos los circuitos que utilizamos.

También existe un paquete para la HP9000, que sirve para diseñar las tarjetas de circuito impreso. Desgraciadamente no fue donado a la UNAM por Hewlett Packard, y la facultad no cuenta con los recursos económicos suficientes para adquirirlo. Esto implicó el diseñar las tarjetas en otra máquina y con otro paquete.

Se pensó hacerlo con el programa Smartwork en una computadora personal, pero se eliminó esta idea dada la complejidad de nuestros circuitos y los pocos recursos con los que cuenta el paquete. Otra opción fue hacerlo en un mainframe en el Dirección General de Servicios de Cómputo Académico. Aquí el diseño de los circuitos impresos se hace en forma automática, por lo que se decidió tomar esta opción.

El mainframe que usamos es la computadora IBM 4381, en combinación con las terminales 5291 y la 8050 (gráfica). El paquete que utiliza el mainframe es el CBDS (Circuit Board Design System) está dividido a su vez en dos subpaquetes: LOKI y SPRIG. En LOKI se dibuja el circuito lógico de las tarjetas y SPRIG realiza el trazado automático de las pistas.

Desgraciadamente no pudimos concluir el diseño del 3GJ en ese sistema debido a los problemas que a continuación enlistamos:

+ La base de datos del CBDS está dañada, lo que ocasiona que sea imposible alterar las características de default (tamaño, espaciamiento entre líneas, número de capas, etc.); lo anterior provoca el no poder seleccionar ciertos componentes que necesitamos.

+ El programa que controla al graficador, no está instalado apropiadamente. Así que no pudimos obtener los dibujos de las tarjetas.

6.2 Alambrado en Wire Wrap (Alambre enrollado) y Programas de Prueba

Se requiere tener un orden al alambrar las tarjetas de wire wrap. A continuación describimos la metodología que seguimos para alambrar el 3GJ. El código de colores que establecimos en el alambrado del 3GJ fue el siguiente:

En las tarjetas madre (principal), de memoria y de puerto paralelo:

Amarillo:	*Datos*
Negro:	*Direcciones*
Azul:	*Datos_Direcciones (a la salida del microprocesador).*
Rojo:	*Señales de Control*
Blanco:	*+5V*
Morado:	*Tierra:*
Verde:	*Interrupciones*

En las tarjetas de puertos serie los colores son:

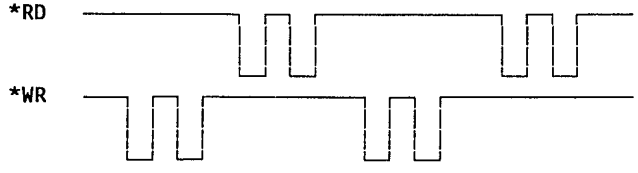
Rojo: *Control o +12 V*
Negro: *Direcciones o -12V*
Amarillo: *Datos o Transmisión de datos del 8251.*

Pasos para Alambrar las Tarjetas de Wire Wrap

1. Alambrar en la tarjeta madre una configuración básica:

8088 (Microprocesador)
8284 (Generador de reloj)
74245 (Latches para el canal de datos)
74373 (Latches para el canal de direcciones)
Switches para particionar la ROM
2716 (ROM)
74138's (decodificadores de memoria y puertos)

Se probó esta configuración por medio de unos programas llamados WWBASICO y WWBAS. Ambos programas prueban el alambrado básico. El primero lee una palabra de ROM y el segundo almacena un caracter ASCII en el acumulador. Las señales de RD y WR se deben de ver funcionando de manera diferente a:



ya que ésta es la forma que tienen cuando no existe ROM, no está conectada, o bien no se direcciona adecuadamente.

Se sabrá que los programas anteriores están funcionando correctamente, cuando se puedan ver los niveles de datos y direcciones en el osciloscopio en forma

sincronizada. Si se conoce el comportamiento correcto de las señales del procesador es posible leer en el osciloscopio las direcciones y/o los códigos de las instrucciones que se están ejecutando. Como estos programas son muy pequeños, es posible observar todo el cambio de niveles que corresponde a la ejecución completa del programa en un solo barrido.

2. Se prueba el correcto funcionamiento de decodificadores de memoria y puertos con el programa WWDECTST el cual activa una línea de cada decodificador durante un corto periódo de tiempo.

3. Se alambra una tarjeta serie con cuatro 8251's y se prueba con los programas:

WWABSE:
Envía abecedarios a la pantalla infinitamente.

WWECHO:
Es un eco del teclado de la PC y corre en conjunción con el programa CARGA de la PC.

4. Se arma una tarjeta de memoria RAM y se prueba con:

WWECHO2:
Es un eco del teclado, se recibe un caracter de la PC, lo escribe en RAM, lo lee de ésta y lo regresa a la PC. (Usa la misma localidad de RAM siempre).

WWRMTST1:
Prueba 1er chip de 32 KB de RAM

WWRMTST2:
Prueba 2do chip de 32 KB de RAM

5. Alambrar el puerto paralelo y probarlo con los programas:

WWPAR1:
Envía un caracter diferente a cada puerto del 8255.

IMPR2:
Envía abecedarios en mayúsculas por el puerto A y minúsculas por el B sin esperar el ACKNLG de la impresora. En la impresión

se

notará que faltan letras y eso se debe a que el 3GJ no espera ACKNLG de la impresora para enviar el siguiente caracter.

6. Alambrar los circuitos para las interrupciones NMI, es decir, el Flip Flop correspondiente, el 74244 y el circuito de NMI. Esta parte se prueba con los siguientes programas:

INTEC1:

(Rutina principal)

INTESUB1:

(Rutina de Interrupción). Envían abecedarios en mayúsculas por el puerto A y minúsculas por el puerto B. En esta ocasión no deben omitirse caracteres.

INTEC2:

(Rutina Principal)

INTESUB2:

(Rutina de Interrupción). Realizan las mismas funciones que los anteriores, pero además al irse a la rutina de interrupción se lee la palabra por el puerto correspondiente para saber quien interrumpió.

7. Terminar de alambrar los chips de interrupciones mascarables (Recepción de PC's) y probar con los siguientes programas:

RXRDY0A:

(Rutina Principal)

RXRDYSUA:

(Rutina de Interrupción) Son ecos de teclado. Al presionar una tecla en la PC se envía un caracter por el puerto serie, es tomado por el microprocesador del 3GJ y enviado de regreso a la pantalla de la PC (eco), adicionalmente se manda un nibble cuyo contenido refleja el estado de los 4 RXRDY que se recibieron.

6.3 Programas de Prueba

A continuación se listan los programas utilizados para probar las etapas descritas.

```
; --------------- WW-BASICO
; 17.01.89
; Este programa prueba el alambrado básico del wire-wrap,
; leyendo una palabra de ROM.
; ----------------------------------------------------

CSEG  SEGMENT 'CODIGO'
      ASSUME CS:CSEG

;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -
      LOCROM     EQU  0
;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -

PRINC PROC

      MOV AX,OFFFFH
      MOV DS,AX
      MOV SI,0
OTRAVEZ:
      MOV AL,[SI+0]
      NOP
      JMP OTRAVEZ
PRINC ENDP                  ; Fin

CSEG  ENDS
      END


; --------------- WWBAS
; 17.01.89
; Este programa prueba el alambrado básico del wire-wrap,
; moviendo una 'J' al acumulador.
; ----------------------------------------------------

CSEG  SEGMENT 'CODIGO'
      ASSUME CS:CSEG

PRINC PROC

MUEVE:
      MOV  AL,'J'
      NOP
      NOP
      JMP  MUEVE

PRINC ENDP                  ; Fin
CSEG  ENDS
      END
```

```
------------------- WADECTST
; 25.01.89
; Este programa prueba los decodificadores de RAM y de puertos.
; Del decodificador RAM parte más baja prueba la primera salida, ; del siguiente, la segunda y así sucesivamente
hasta probar la
; cuarta del cuarto decodificador. A continuación prueba la
; primera salida del primer decodificador, la segunda del segundo
; y la tercera del tercero. Después vuelve a empezar a probar la
; decodicicación de RAM.

GRUPO GROUP CSEG, DSEG
      ASSUME CS:GRUPO, DS:GRUPO

DSEG  SEGMENT PUBLIC

      RAM       DB   0

DSEG  ENDS

CSEG  SEGMENT PUBLIC

PRINC PROC  NEAR
INIC:
      MOV   AX,0          ; Inicializa DS

OTRA:
      MOV   DS,AX
      MOV   CX,7FFFH

REPITE:
      MOV   AL,RAM
      LOOP  REPITE

      MOV   AX,DS
      ADD   AX,4800H
      JNC   OTRA
      MOV   AX,0              ; Inicia la prueba de prtos
      MOV   DS,AX
      MOV   DX,4H
OTRA1:
      MOV   CX,7FFFH
REP1:
      IN    AL,DX
      LOOP  REP1
      ADD   DX,9H
      CMP   DX,16H
      JBE   OTRA1
      JMP   INIC

PRINC ENDP

CSEG  ENDS

      END   PRINC
```

```
; ---------- WAABSE
; 10.10.88
; Este programa envía abecedarios a la pantalla infinitamente
; ------------------------------------------------------------
CSEG  SEGMENT 'CODIGO'
      ASSUME CS:CSEG

;- - - - - - - - - D A T O S - - - - - - - - - - - - - - -
      PRTOSE      EQU  14H
      CTRL        EQU  34H
;- - - - - - - - - D A T O S - - - - - - - - - - - - - - -

PRINC PROC
; ----Inicializa el prto serie
      MOV   AL,4FH
      OUT   CTRL,AL
      MOV   AL,25H
      OUT   CTRL,AL

; --- Las letras
ABC:  MOV   BL,'A'
TX?:  IN    AL,CTRL
      TEST  AL,01
      JZ    TX?
      MOV   AL,BL
      OUT   PRTOSE,AL
      CMP   BL,'Z'
      JE    ABC
      INC   BL
      JMP   TX?

PRINC ENDP                  ; Fin

CSEG  ENDS
      END


; ---------- WAECHO
; 02/11/1989
; Este programa es un eco del teclado a través de nuestro puerto
; serie
; ------------------------------------------------------

CSEG  SEGMENT 'CODIGO'
      ASSUME CS:CSEG

;- - - - - - - - - D A T O S - - - - - - - - - - - - - - -
      PRTOSE      EQU  08H
      CTRL        EQU  28H
;- - - - - - - - - D A T O S - - - - - - - - - - - - - - -

PRINC   PROC

;----- ESPERA DEL CARACTER Y TRANSMISION DEL MISMO
```

```
OTRO:
RX?:
      IN      AL,CTRL
      TEST    AL,00000010B
      JZ      RX?
      IN      AL,PRTOSE
      MOV     BL,AL
TX?:
      IN      AL,CTRL
      TEST    AL,00000001B
      JZ      TX?
      MOV     AL,BL
      OUT     PRTOSE,AL
      JMP     OTRO

PRINC ENDP                    ; Fin

CSEG  ENDS
      END


; ---------------- WWECHO2
; 02/11/1989
; Este programa es un eco del teclado a través de nuestro puerto
; serie, la diferencia con el anterior es que el caracter
; recibido de la PC se guarda en memoria antes de ser enviado de
; regreso a la pantalla.
; ------------------------------------------------------

GRUPO GROUP   CODIGO_SEG, DATOS_SEG
      ASSUME CS:GRUPO, DS:GRUPO

CODIGO_SEG      SEGMENT PUBLIC

;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -
      PRTOSE      EQU  0AH
      CTRL        EQU  2AH
;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -

PRINC PROC
      XOR   AX,AX
      MOV   DS,AX
; --- Inicializa el prto serie
      MOV   AL,4FH
      OUT   CTRL,AL
      MOV   AL,25H
      OUT   CTRL,AL

;----- ESPERA DEL CARACTER Y TRANSMISION DEL MISMO
OTRO:
RX?:
      IN      AL,CTRL
      TEST    AL,00000010B
```

```
      JZ      RX?
      IN      AL,PRTOSE
      MOV     ALMACENA,AL
TX?:
      IN      AL,CTRL
      TEST    AL,00000001B
      JZ      TX?
      MOV     AL,ALMACENA
      OUT     PRTOSE,AL
      JMP     OTRO

PRINC ENDP                    ; Fin

CODIGO_SEG      ENDS

DATOS_SEG       SEGMENT PUBLIC
ORG             1A5H
ALMACENA        DB 0

DATOS_SEG       ENDS
      END       PRINC

; ------------------ WWRAMTST1
; 27.10.88. Versión modificada el 26 Feb 89
; Este programa prueba RAM y después manda a paralelo
; y a serie.
; Es necesario guardar el programa en una zona que no
; se vaya a probar asimismo como el stack.(Guardar el pro-
; grama en la dirección 0800:0600).
; Para variar el bloque a probar basta modificar el va-
; lor de DS y SS.
; Si detecta un error en RAM saca QUINCE M's
; Sino detecta error manda 32 A'es y después DIEZ B's
; Este programa prueba la 1a. RAM (32K)

CSEG  SEGMENT
      ASSUME CS:CSEG, DS:CSEG, SS:CSEG

;- - - - - - - - - D A T O S  - - - - - - - - - - - - - - - - - -
      PRTOA       EQU  003H
      PRTOC       EQU  083H
      PRTOCTRL    EQU  0C3H
      PRTOSE      EQU   08H
      PRTOCSE     EQU   28H
;- - - - - - - - - D A T O S  - - - - - - - - - - - - - - - - - -
PRINC PROC  NEAR
      MOV   AX,0000H        ; Inicializa DS
      MOV   DS,AX
      MOV   AX,0800H        ; Colcca SS en los sigs 32k de
      MOV   SS,AX           ; RAM
      MOV   SP,500H

; --- Inicializa el 8255
      MOV   AL,80H
```

```
      OUT   PRTOCTRL,AL
      MOV   AL,' '
      OUT   PRTOA,AL
      MOV   AL,80H
      OUT   PRTOC,AL

; --- Abre contadores
      MOV   CX,7FFFH        ; Contador de 32K
      MOV   DX,0            ; Contador de 1K
      MOV   SI,0000H        ; Apuntador a RAM
      MOV   AX,'AA'         ; Dato modelo

REPITE:
      MOV   [SI],AL         ; Guarda en RAM
      XOR   AL,AL           ; Limpia
      MOV   AL,[SI]         ; Saca de RAM
      CMP   DX,03FFH        ; Compara si ya tiene 1K
      JNE   NOSALE
      CALL  TXCAR           ; Saca una 'A' por el serie
      OUT   PRTOA,AL        ; y el paralelo
      PUSH  AX
      MOV   AL,0            ; strobe (C7)   en bajo
      OUT   PRTOC,AL
      MOV   AL,80H          ; strobe (C7)   en alto
      OUT   PRTOC,AL
; --- Espera a al impresora
      MOV   BX,0C00H
ESP_A:
      DEC   BX
      JNZ   ESP_A
      MOV   DX,0            ; Reinicia contador de 1K

      POP   AX
NOSALE:
      CMP   AH,AL           ; Compara
      JNE   YATXM
      INC   SI              ; Incrementa apuntador
      INC   DX              ; Incrementa contador de 1K
      LOOP  REPITE

YATXB:
      MOV   CX,10
MAS_B:
      MOV   AL,'B'
      CALL  TXCAR           ; Saca una 'B' por el serie y
      OUT   PRTOA,AL        ; el paralelo
      MOV   AL,0            ; strobe (C7)   en bajo
      OUT   PRTOC,AL
      MOV   AL,80H          ; strobe (C7)   en alto
      OUT   PRTOC,AL
; --- Espera a al impresora
      MOV   BX,0C00H
ESP_B:
      DEC   BX
```

```
      JNZ   ESP_B
      LOOP  MAS_B
      JMP   FIN

YATXM:
      MOV   CX,15
MAS_M:
      MOV   AL,'M'
      CALL  TXCAR           ; Saca una 'M' por el serie y
      OUT   PRTOA,AL        ; el paralelo
      MOV   AL,0            ; strobe (C7)   en bajo
      OUT   PRTOC,AL
      MOV   AL,80H          ; strobe (C7)   en alto
      OUT   PRTOC,AL
; --- Espera a al impresora
      MOV   BX,0C00H
ESP_M:
      DEC   BX
      JNZ   ESP_M
      LOOP  MAS_M

FIN:  JMP   FIN

TXCAR:
      PUSH   DX           ; debido a que DX y AX
      PUSH   AX           ; tienen valores importan-
      MOV    DX,PRTOCSE   ; tes se guardan en STACK
C_TXCAR:
      IN     AL,DX        ; Se pregunta si está
      TEST   AL,00000001b; listo para transmitir
      JZ     C_TXCAR
      MOV    DX,PRTOSE    ; A continuación se saca
      POP    AX           ; el dato por el prto
      OUT    DX,AL        ; adecuado y se recupera
      POP    DX           ; antiguo valor de DX
      RET

PRINC ENDP

CSEG  ENDS
      END   PRINC


; ------------------  WWRAMTST2
; 27.10.88. Versión modificada el 26 Feb 89
; Este programa prueba RAM y después manda a paralelo
; y a serie.
; Es necesario guardar el programa en una zona que no
; se vaya a probar asimismo como el stack. (Guardar el pro-
; grama en la zona más baja de RAM  0000:0600).
; Para variar el bloque a probar basta modificar el va-
; lor de DS y SS.
; Si detecta un error en RAM saca QUINCE M's
; Si no detacta error manda 32 A'es y después DIEZ B's
```

```
; Este programa prueba la 2a. RAM (32K)

CSEG  SEGMENT
      ASSUME CS:CSEG, DS:CSEG, SS:CSEG

;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -
      PRTOA       EQU  003H
      PRTOC       EQU  083H
      PRTOCTRL    EQU  0C3H
      PRTOSE      EQU   08H
      PRTOCSE     EQU   28H
;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -

PRINC PROC  NEAR
      MOV   AX,0800H        ; Inicializa DS
      MOV   DS,AX
      MOV   AX,0000H        ; Coloca SS en los sigs 32k de
      MOV   SS,AX           ; RAM
      MOV   SP,500H

; --- Inicializa el 8255
      MOV   AL,80H
      OUT   PRTOCTRL,AL
      MOV   AL,' '
      OUT   PRTOA,AL
      MOV   AL,80H
      OUT   PRTOC,AL

; --- Abre contadores
      MOV   CX,7FFFH        ; Contador de 32K
      MOV   DX,0            ; Contador de 1K
      MOV   SI,0000H        ; Apuntador a RAM
      MOV   AX,'AA'         ; Dato modelo

REPITE:
      MOV   [SI],AL         ; Guarda en RAM
      XOR   AL,AL           ; Limpia
      MOV   AL,[SI]         ; Saca de RAM
      CMP   DX,03FFH        ; Compara si ya tiene 1K
      JNE   NOSALE
      CALL  TXCAR           ; Saca una 'A' por el serie
      OUT   PRTOA,AL        ; y el paralelo
      PUSH  AX
      MOV   AL,0            ; strobe (C7)   en bajo
      OUT   PRTOC,AL
      MOV   AL,80H          ; strobe (C7)   en alto
      OUT   PRTOC,AL
; --- Espera a al impresora
      MOV   BX,0C00H
ESP_A:
      DEC   BX
      JNZ   ESP_A
      MOV   DX,0            ; Reinicia contador de 1K
```

```
      POP   AX
NOSALE:
      CMP   AH,AL           ; Compara
      JNE   YATXM
      INC   SI              ; Incrementa apuntador
      INC   DX              ; Incrementa contador de 1K
      LOOP  REPITE

YATXB:
      MOV   CX,10
MAS_B:
      MOV   AL,'B'
      CALL  TXCAR           ; Saca una 'B' por el serie y
      OUT   PRTOA,AL        ; el paralelo
      MOV   AL,0            ; strobe (C7)   en bajo
      OUT   PRTOC,AL
      MOV   AL,80H          ; strobe (C7)   en alto
      OUT   PRTOC,AL
; --- Espera a al impresora
      MOV   BX,0C00H
ESP_B:
      DEC   BX
      JNZ   ESP_B
      LOOP  MAS_B
      JMP   FIN

YATXM:
      MOV   CX,15
MAS_M:
      MOV   AL,'M'
      CALL  TXCAR           ; Saca una 'M' por el serie y
      OUT   PRTOA,AL        ; el paralelo
      MOV   AL,0            ; strobe (C7)   en bajo
      OUT   PRTOC,AL
      MOV   AL,80H          ; strobe (C7)   en alto
      OUT   PRTOC,AL
; --- Espera a al impresora
      MOV   BX,0C00H
ESP_M:
      DEC   BX
      JNZ   ESP_M
      LOOP  MAS_M

FIN:  JMP   FIN

TXCAR:
      PUSH    DX            ; debido a que DX y AX
      PUSH    AX            ; tienen valores importan-
      MOV     DX,PRTOCSE    ; tes se guardan en STACK
C_TXCAR:
      IN      AL,DX         ; Se pregunta si está
      TEST    AL,00000001b; listo para transmitir
      JZ      C_TXCAR
      MOV     DX,PRTOSE     ; A continuación se saca
```

```
      POP    AX          ; el dato por el prto
      OUT    DX,AL       ; adecuado y se recupera
      POP    DX          ; antiguo valor de DX
      RET

PRINC ENDP

CSEG  ENDS
      END    PRINC


; --------------  WWPAR1
; 10.10.88
; Este programa prueba el alambrado, en wire-wrap,
; del puerto paralelo.
; Saca tres datos diferentes por cada uno de los puertos.
; Los datos sólo se pueden ver en el osciloscopio.
; ------------------------------------------------------

CSEG  SEGMENT
      ASSUME CS:CSEG, DS:CSEG

;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -
      PRTOA      EQU  03H
      PRTOB      EQU  43H
      PRTOC      EQU  83H
      CTRL       EQU  0C3H
;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -

PRINC PROC  NEAR
      MOV    AX,CSEG        ; Inicializa DS
      MOV    DS,AX

; --- Programamos el 8255
      MOV    AL,80H
      OUT    CTRL,AL

; --- Salen los datos
      MOV    AL,'C'
      OUT    PRTOA,AL
      MOV    AL,'J'
      OUT    PRTOB,AL
      MOV    AL,'G'
      OUT    PRTOC,AL

FIN:  JMP    FIN

PRINC ENDP

CSEG  ENDS
      END


; ----------------- IMPR 2
```

```
; 27 JULIO 88, versión modificada el 20 Febrero 89
; Este programa debe hacer lo mismo que IMPR1 salvo
; que lo hace
; sin interrupciones, comienza a imprimir abecedarios
; sin esperar
; ACK de la impresora, en mayúsculas por el puerto A
; y en minúsculas por el puerto B


CSEG  SEGMENT
      ASSUME CS:CSEG, DS:CSEG

;- - - - - - - - - D A T O S - - - - - - - - - - - - - - - - -
      PRTOA       EQU  00003H
      PRTOB       EQU  00043H
      PRTOC       EQU  00083H
      PRTOCTRL    EQU  000C3H

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

PRINC PROC  NEAR
      MOV   AX,CSEG         ; Inicializa DS
      MOV   DS,AX
PROCESO:
      MOV   DX,PRTOCTRL
      MOV   AL,080H         ; Pal. control Modo 0, sal.A,
                            ; Cl=sal , Cu=sal
      OUT   DX,AL           ; Programamos el 8255

ABC:  MOV   BL,'A'          ; Inicia el abcdario
XYZ:  MOV   AL,BL
      MOV   DX,PRTOA        ; Va por el puerto 1-A del
                            ; circuito
      OUT   DX,AL
      OR    AL,00100000B    ; Va a obtener el caracter pero
                            ; en minúscula para sacarlo
                            ; por el puerto B
      MOV   DX,PRTOB
      OUT   DX,AL
      MOV   DX,PRTOC        ; El strobe va por el puerto 1-C
      MOV   AL,01000111B    ; strobe (C7,prto A; C3,prto B)
                            ; en bajo
      OUT   DX,AL
      MOV   AL,11001111B    ; strobe (C7,prto A; C3,prto B)
                            ; en alto
      OUT   DX,AL
      INC   BL              ; Incrementa la letra
      CMP   BL,'Z'
      JG    ABC             ; Si es mayor que 'Z' comienza                ; de nuevo,
      JMP   XYZ             ; si no, la imprime.

PRINC ENDP

CSEG  ENDS
```

```
      END   PRINC

;  INTEC1
;  18 de AGOSTO 1988 Versión modificada el 21-Febrero-1989
;  Este programa manda a imprimir una 'A' y se asocia
;  a la rutina de interrupción intesub1.asm  en donde se
;  se manda a imprimir el resto de las interrupciones.
;
CSEG  SEGMENT
      ASSUME CS:CSEG, DS:CSEG, SS:CSEG, ES:CSEG

;- - - - - - - - - D A T O S  - - - - - - - - - - - -
      PRTOA       EQU  003H
      PRTOB       EQU  043H
      PRTOC       EQU  083H
      PRTOCTRL    EQU  0C3H
      IP2L        EQU  00H
      IP2H        EQU  07H
      CS2L        EQU  00H
      CS2H        EQU  00H
;- - - - - - - - - D A T O S  - - - - - - - - - - - - - - -
PRINC PROC  NEAR
      MOV   AX,0            ; Inicializa DS
      MOV   DS,AX
      MOV   SS,AX
      MOV   ES,AX
      MOV   SP,500H
      MOV   BP,AX
      MOV   SI,AX
      MOV   DI,AX

;----------Inicialización puerto paralelo para salida
      MOV   DX,PRTOCTRL     ;
      MOV   AL,10000000B    ; pal control modo 0. sal A, Cl sal                    ; Cu sal
      OUT   DX,AL

UNO:
;-----------------ASIGNA EN RAM EL IP Y CS DE LA RUTINA ;INTERRUPCION
      MOV   SI,08H
      MOV   AL,IP2L
      MOV   [SI+0],AL
      MOV   AL,IP2H
      MOV   [SI+1],AL
      MOV   AL,CS2L
      MOV   [SI+2],AL
      MOV   AL,CS2H
      MOV   [SI+3],AL

;---------------PROCEDIMIENTO PRINÇIPAL
      MOV   AL,'A'
      MOV   BL,AL
      MOV   DX,PRTOA        ; POR EL PUERTO 'P2A' DEL KIT
      OUT   DX,AL           ; 'Vámonos....!'
```

```
      OR    AL,00100000B    ; Convierte el caracter en
                            ; minúscula
      MOV   DX,PRTOB
      OUT   DX,AL
      MOV   DX,PRTOC
      MOV   AL,01000111B    ; strobe (C7ptoA,C3ptoB)en bajo
      OUT   DX,AL
      MOV   AL,11001111B    ; strobe (C7ptoA,C3ptoB)en alto
      OUT   DX,AL
      STI
FIN:  JMP   FIN

PRINC ENDP

CSEG  ENDS
      END   PRINC


;  INTESUB1
;  18 de Agosto de 1988 .Versión modificada el 21 Febrero 89
;  Este programa manda a imprimir desde la letra 'B' en
;  adelante (según la secuencia ASCII), regresando por
;  cada letra impresa al
;  programa principal llamado INTEC1.ASM
;
CSEG  SEGMENT
      ASSUME CS:CSEG, DS:CSEG

;- - - - - - - - - D A T O S  - - - - - - -
      PRTOA       EQU  003H
      PRTOB       EQU  043H
      PRTOC       EQU  083H
      PRTOCTRL    EQU  0C3H
 ;- - - - - - - - - - D A T O S  - - - - - - - - - - - - - - -
PRINC PROC  NEAR
      MOV   AX,0            ; Inicializa DS
      MOV   DS,AX
      INC   BL
      CMP   BL,'Z'
      JBE   CONT
      MOV   BL,'A'
CONT: MOV   AL,BL           ; Carga siguiente letra
      MOV   DX,PRTOA
      OUT   DX,AL           ; 'Vámonos....!'
      OR    AL,00100000B    ; Convierte el caracter en
      MOV   DX,PRTOB
      OUT   DX,AL           ; minúsculas y lo saca por B
      MOV   AL,01000111B    ; strobe (C7ptoA,C3ptoB)en bajo
      MOV   DX,PRTOC
      OUT   DX,AL
      MOV   AL,11001111B    ; strobe (C7ptoA,C3ptoB)en alto
      OUT   DX,AL
      STI
      IRET
```

```
PRINC ENDP

CSEG  ENDS
      END   PRINC


;  INTEC2
;  18 de AGOSTO 1988 Versión modificada el 26-Febrero-1989
;  Este programa manda a imprimir una 'A' y se asocia
;  a la rutina de interrupción intesub2.asm  en donde se
;  se manda a imprimir el resto de las interrupciones.
;  Además, en la rutina de interrupción se pregunta por
;  qué ACK interrumpió y se despliega en pantalla, además
;  del abecedario. Primero se despliega letra y después
;  quién interrumpe.

CSEG  SEGMENT
      ASSUME CS:CSEG, DS:CSEG, SS:CSEG, ES:CSEG

;- - - - - - - - - D A T O S - - - - - - - - - - - - -
      PRTOA       EQU  003H
      PRTOB       EQU  043H
      PRTOC       EQU  083H
      PRTOCTRL    EQU  0C3H
      IP2L        EQU  00H
      IP2H        EQU  07H
      CS2L        EQU  00H
      CS2H        EQU  00H
      FLIPYA      EQU  04H
      FLIPYB      EQU  05H
;- - - - - - - - - D A T O S - - - - - - - - - - - - - - - -
PRINC PROC  NEAR
      MOV   AX,0            ; Inicializa DS
      MOV   DS,AX
      MOV   SS,AX
      MOV   ES,AX
      MOV   SP,500H
      MOV   BP,AX
      MOV   SI,AX
      MOV   DI,AX

;----------Inicialización puerto paralelo para salida
      MOV   DX,PRTOCTRL     ;
      MOV   AL,10000000B    ; pal control modo 0. sal A, CL sal                    ; Cu sal
      OUT   DX,AL

UNO:
;-----------------ASIGNA EN RAM EL IP Y CS DE LA RUTINA ;INTERRUPCION
        MOV   SI,08H
        MOV   AL,IP2L
        MOV   [SI+0],AL
        MOV   AL,IP2H
        MOV   [SI+1],AL
```

```
        MOV   AL,CS2L
        MOV   [SI+2],AL
        MOV   AL,CS2H
        MOV   [SI+3],AL

;---------------PROCEDIMIENTO PRINCIPAL
       MOV   DX,FLIPYA       ; Manda clear
       OUT   DX,AL           ; a los
       MOV   DX,FLIPYB       ; Flip Flops
       OUT   DX,AL           ; de impresoras
       MOV   AL,'A'
       MOV   BL,AL
       MOV   DX,PRTOA        ; POR EL PUERTO 'P2A' DEL KIT
       OUT   DX,AL           ; 'Vámonos....!'
       OR    AL,00100000B    ; Convierte el caracter en
                             ; minúscula
       MOV   DX,PRTOB
       OUT   DX,AL
       MOV   DX,PRTOC
       MOV   AL,01000111B    ; strobe (C7ptoA,C3ptoB)en bajo
       OUT   DX,AL
       MOV   AL,11001111B    ; strobe (C7ptoA,C3ptoB)en alto
       OUT   DX,AL
FIN:   JMP   FIN

PRINC  ENDP

CSEG   ENDS
       END   PRINC

;  INTESUB2
;  18 de Agosto de 1988 .Versión modificada el 26 Febrero 89
;  Este programa manda a imprimir desde la letra 'B' en
;  adelante (según la secuencia ASCII), regresando por
;  cada letra impresa al
;  programa principal llamado INTEC2.ASM
;  Además se manda a la terminal la letra que se imprime y
;  la palabra que se lee al detectar que se interrumpe.
CSEG   SEGMENT
       ASSUME CS:CSEG, DS:CSEG

;- - - - - - - - - D A T O S  - - - - - - -
       PRTOA      EQU  003H
       PRTOB      EQU  043H
       PRTOC      EQU  083H
       PRTOCTRL   EQU  0C3H
       PRTOSE     EQU   08H
       PRTOCSE    EQU   28H
       QUIEN      EQU   00H
       FLIPYA     EQU   04H
       FLIPYB     EQU   05H
 ;- - - - - - - - - D A T O S  - - - - - - - - - - - - - - -

PRINC  PROC  NEAR
```

```
        MOV   AX,0              ; Inicializa DS
        MOV   DS,AX
        MOV   AL,BL             ; Manda a la PC la letra que
        CALL  TXCAR             ; imprimió
        MOV   DX,QUIEN          ; Pregunta quién interrumpe
        IN    AL,DX
        MOV   DX,FLIPYA         ; Da
        OUT   DX,AL             ; Clear
        MOV   DX,FLIPYB         ; a los
        OUT   DX,AL             ; Flip Flops de impresoras.
        CALL  HEX_ASC           ; Convierte lo que leyo a
        CALL  TXCAR             ; ASCII y lo manda a la PC.
        INC   BL
        CMP   BL,'Z'
        JBE   CONT
        MOV   BL,'A'
CONT:   MOV   AL,BL             ; Carga siguiente letra
        MOV   DX,PRTOA
        OUT   DX,AL             ; 'Vámonos....!'
        OR    AL,00100000B      ; Convierte el caracter en
        MOV   DX,PRTOB
        OUT   DX,AL             ; minúsculas y lo saca por B
        MOV   AL,01000111B      ; strobe (C7ptoA,C3ptoB)en bajo
        MOV   DX,PRTOC
        OUT   DX,AL
        MOV   AL,11001111B      ; strobe (C7ptoA,C3ptoB)en alto
        OUT   DX,AL
        IRET

; Esta rutina convierte un dato numérico o letra mayúscula o
; minúscula de forma hexadecimal a ASCII.
HEX_ASC:
        AND     AL,00001111B
        CMP     AL,09H
        JA      SU_37
        OR      AL,30H
        RET
SU_37:
        ADD     AL,37H
        RET

; La siguiente rutina es utilizada siempre que se quiere en-
; viar algo por el puerto serie.
TXCAR:
        PUSH    DX              ; debido a que DX y AX
        PUSH    AX              ; tienen valores importan-
        MOV     DX,PRTOCSE      ; tes se guardan en STACK
C_TXCAR:
        IN      AL,DX           ; Se pregunta si está
        TEST    AL,00000001b; listo para transmitir
        JZ      C_TXCAR
        MOV     DX,PRTOSE       ; A continuación se saca
        POP     AX              ; el dato por el prto
        OUT     DX,AL           ; adecuado y se recupera
```

```
        POP     DX              ; antiguo valor de DX
        RET

PRINC   ENDP

CSEG    ENDS
        END     PRINC


;RXRDY0A
;Este programa es la rutina principal de RXRDYSUB.ASM y solo
;está en espera de una interrupción de un RXRDY de un puerto
;serial 21 feb 1989
;

CSEG    SEGMENT
        ASSUME CS:CSEG, DS:CSEG, SS:CSEG, ES:CSEG

;- - - - - - - - - D A T O S  - - - - - - - - - - - - -
        IP2L        EQU  00H
        IP2H        EQU  07H
        CS2L        EQU  00H
        CS2H        EQU  00H
        CTRL        EQU  2AH  ; Aquí se va a poner la dir. del
        PRTOSE      EQU  0AH  ; prto. serie a probar.
;- - - - - - - - - D A T O S  - - - - - - - - - - - - - -
PRINC   PROC  NEAR
        MOV   AX,0            ; Inicializa DS
        MOV   DS,AX
          MOV   SS,AX
          MOV   ES,AX
          MOV   SP,500H
        MOV   BP,AX
          MOV   SI,AX
          MOV   DI,AX

UNO:
;-----------------ASIGNA EN RAM EL IP Y CS DE LA RUTINA ;INTERRUPCION
          MOV   SI,80H
          MOV   AL,IP2L
          MOV   [SI+0],AL
          MOV   AL,IP2H
          MOV   [SI+1],AL
          MOV   AL,CS2L
          MOV   [SI+2],AL
          MOV   AL,CS2H
          MOV   [SI+3],AL


; ----- Inicializa el prto serie
        MOV   AL,4FH
        OUT   CTRL,AL
        MOV   AL,25H
```

```
      OUT   CTRL,AL

;----------------PROCEDIMIENTO PRINCIPAL

      STI
ESPERA: JMP ESPERA

PRINC   ENDP

CSEG    ENDS
      END   PRINC


;RXRDYSUA
;Este programa es la rutina de RXRDY y aquí lo que
;hacemos es ver quién nos interrumpió y mandar el caracter a
;la PC . 21 feb 1989

CSEG    SEGMENT
      ASSUME CS:CSEG, DS:CSEG

;---------------D A T O S ----------------
      QUIEN     EQU  001H
      CTRL      EQU  02AH ; Aquí ponemos la dir. del prto.
      PRTOSE    EQU  00AH ; serie a probar.

;---------------D A T O S ----------------

PRINC   PROC  NEAR

      MOV   AX,0          ; Inicializa DS
      MOV   DS,AX

;VER QUIEN ME INTERRUMPIO

      IN    AL,QUIEN

;CONVERTIR A ASCII

      AND   AL,00001111B
      CMP   AL,09H
      JA    SU_37
      OR    AL,30H
      JMP   TRANS
SU_37:
      ADD   AL,37H

;TRANSMITIR A LA PC QUIEN ME INTERRUMPIO

TRANS:
      MOV   BL,AL
TX?:  IN    AL,CTRL
      TEST  AL,00000001B
      JZ    TX?
```

```
        MOV   AL,BL
        OUT   PRTOSE,AL

;RECIBIR CARACTER Y TRANSMITIRLO A LA PC

RX?:    IN    AL,CTRL
        TEST  AL,00000010B
        JZ    RX?
        IN    AL,PRTOSE
        MOV   BL,AL
TX1?:   IN    AL,CTRL
        TEST  AL,00000001B
        JZ    TX1?
        MOV   AL,BL
        OUT   PRTOSE,AL
        STI
        IRET

PRINC   ENDP

CSEG    ENDS
        END   PRINC
```

CONCLUSIONES

Podemos obtener varias conclusiones interesantes de este proyecto.

Pocos son los proyectos de tesis en los que se llega a construir un modelo que funcione completamente. La mayor parte de las veces los diseños se quedan en los escritorios o en las bibliotecas de la Universidad. El Multiplexor 3GJ ha roto con este esquema, existe un modelo funcionando.

En el mercado de productos para microcomputadoras no existe actualmente ningún controlador como el ahora presentado. Son varias las ventajas sobre ellos:

- *El dispositivo puede manejar desde 4 hasta 16 PC's, por medio de tarjetas expandibles.*
- *Memoria expandible chip por chip (de 32 KB) hasta 1 MB.*
- *El controlador no se "esclaviza" con una PC a la vez:* ***TODOS*** *pueden mandar a imprimir simultáneamente.*
- *El proceso es "transparente" para las computadoras personales, no hay que mover perillas, ni presionar botones, ni esperar a que alguien más termine su impresión.*
- *Posibilidad de conectar mayores distancias ya que se usa el puerto serie de las PC's.*
- *El cable de conexión es de solo 4 hilos por lo que el costo es mucho menor que con el típico cable paralelo para impresoras.*
- *Todos los componentes utilizados son fáciles de conseguir y de bajo costo por lo que el precio final es sensiblemente menor.*

En resumen, no sólo se tuvo la visión acerca del tipo de aparato que hacía falta, sino que también fue posible el diseño del mismo y lo que es más, su construcción final.

Hay que hacer notar lo importante que es el tener una buena organización cuando

se llevan a cabo trabajos interdiciplinarios como este. Cuando no se tiene una adecuada coordinación hay mucha pérdida de tiempo y de recursos.

También es importante mencionar que gracias al enorme apoyo recibido del Centro de Diseño Electrónico, en lo que a instalaciones y equipo se refiere, fue posible el llevar a buen término la labor que nos impusimos. A diferencia de otros muchos lugares sostenidos por el presupuesto oficial, en este caso los recursos han sido aprovechados.

En México podemos hacer proyectos, en materia tecnológica y científica, del más avanzado nivel. Tenemos los medios y los conocimientos. En este proyecto, como en otros trabajos similares, se ha demostrado que sí podemos.

"POR MI RAZA HABLARA EL ESPITITU"

Gabriel Arellano Gutiérrez
Guillermo Garduño Ambrís
Jorge Morales Chong
Guillermo Vignau Esteva

Junio 1989

BIBLIOGRAFIA

Hall, Douglas V.
Microprocessors and Interfacing
Programming and Hardware
Primera Edición
Mc Graw Hill
1986

Seyer, Martin D.
RS-232 Made Easy
Connecting Computers, Printers, Terminals and Modems.
Primera Edición
Prentice Hall, Inc.
1984

John E. McNamara
Technical Aspects of data Communication
Segunda Edición
Digital Equipment Corporation
1982

Lewis, James D.
8086 & 8088
Microprocessor Instant Reference Card
Micro Logic Corp.
1984

Cortés Barrios, Ramón

Programación de Microcomputadoras
Basadas en el Microprocesador 8088
Primera Edición
Editorial Limusa
1988

Intel Corporation
SDK - 86 User's Manual
Primera Edición
1988

Microsoft Corporation
Microsoft Codeview
Window Oriented Debugger for the MS-DOS Operating System
Primera Edición
1986

Microsoft Corporation
MS-DOS 3.2 Reference Manual
Primera Edición
1986

Titus, Jon
EDN's 15th Annual uP/uC Chip Directory
EDN Magazine Edition
Volumen 33, Número 22
pp 164 - 270
Octubre 27, 1988

Hewlett-Packard Company
LaserJet Series II Printer
User's Manual
Manual Part No. 33440-90901
Primera Edición
1987

Norton, Peter & Socha, Jhon
Peter Norton's Assembly Language Book for the IBM PC
Primera Edición
Prentice Hall Press
1986

Apéndice 

Tabla ASCII

IBM-US (PC-8) Symbol Set

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 0	▶ 16	SP 32	0 48	@ 64	P 80	` 96	p 112	Ç 128	É 144	á 160	▒ 176	└ 192	╨ 208	α 224	≡ 240
1	☺ 1	◀ 17	! 33	1 49	A 65	Q 81	a 97	q 113	ü 129	æ 145	í 161	▓ 177	┴ 193	╤ 209	β 225	± 241
2	● 2	↕ 18	" 34	2 50	B 66	R 82	b 98	r 114	é 130	Æ 146	ó 162	│ 178	┬ 194	╥ 210	Γ 226	≥ 242
3	♥ 3	‼ 19	# 35	3 51	C 67	S 83	c 99	s 115	â 131	ô 147	ú 163	┤ 179	├ 195	╙ 211	π 227	≤ 243
4	♦ 4	¶ 20	$ 36	4 52	D 68	T 84	d 100	t 116	ä 132	ö 148	ñ 164	╡ 180	─ 196	╘ 212	Σ 228	⌠ 244
5	♣ 5	§ 21	% 37	5 53	E 69	U 85	e 101	u 117	à 133	ò 149	Ñ 165	╢ 181	┼ 197	╒ 213	σ 229	⌡ 245
6	♠ 6	▬ 22	& 38	6 54	F 70	V 86	f 102	v 118	å 134	û 150	ª 166	╖ 182	╞ 198	╓ 214	μ 230	÷ 246
7	• 7	↨ 23	' 39	7 55	G 71	W 87	g 103	w 119	ç 135	ù 151	º 167	╕ 183	╟ 199	╫ 215	τ 231	≈ 247
8	◘ 8	↑ 24	(40	8 56	H 72	X 88	h 104	x 120	ê 136	ÿ 152	¿ 168	╣ 184	╚ 200	╪ 216	Φ 232	° 248
9	○ 9	↓ 25	) 41	9 57	I 73	Y 89	i 105	y 121	ë 137	Ö 153	⌐ 169	║ 185	╔ 201	┘ 217	Θ 233	∙ 249
A	◙ 10	→ 26	* 42	: 58	J 74	Z 90	j 106	z 122	è 138	Ü 154	¬ 170	╗ 186	╩ 202	┌ 218	Ω 234	· 250
B	♂ 11	← 27	+ 43	; 59	K 75	[91	k 107	{ 123	ï 139	¢ 155	½ 171	╝ 187	╦ 203	█ 219	δ 235	√ 251
C	♀ 12	∟ 28	, 44	< 60	L 76	\ 92	l 108	| 124	î 140	£ 156	¼ 172	╜ 188	╠ 204	▄ 220	∞ 236	n 252
D	♪ 13	↔ 29	- 45	= 61	M 77	] 93	m 109	} 125	ì 141	¥ 157	¡ 173	╛ 189	═ 205	▌ 221	φ 237	² 253
E	♫ 14	▲ 30	. 46	> 62	N 78	^ 94	n 110	~ 126	Ä 142	₧ 158	« 174	┐ 190	╬ 206	▐ 222	ε 238	■ 254
F	☼ 15	▼ 31	/ 47	? 63	O 79	_ 95	o 111	⌂ 127	Å 143	ƒ 159	» 175	└ 191	╧ 207	▀ 223	∩ 239	

Apéndice B

Norma RS-232-C

This appendix contains extracts from the Electronic Industries Association RS-232-C Standard, which outlines the interface between data terminal equipment and data communication equipment employing serial binary data interchange. The standard, in its entirety, is available for order from the following address:

EIA Engineering Department
Standards Sales
2001 Eye Street, N.W.
Washington, D.C. 20006
(202) 457-4966

Notice

EIA engineering standards are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such standards shall not in any respect preclude any member or nonmember of EIA from manufacturing or selling products not conforming to such standards, nor shall the existence of such standards preclude their voluntary use by those other than EIA members whether the standard is to be used either domestically or internationally.

Recommended standards are adopted by EIA without regard to whether or not their adoption may involve patents on articles, materials, or processes. By such action, EIA does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting recommended standards.

1.1 This standard is applicable to the interconnection of data terminal equipment (DTE) and data communication equipment (DCE) employing serial binary data interchange. It defines:

Section 2. Electrical Signal Characteristics
Electrical characteristics of the interchange signals and associated circuitry.

Section 3. Interface Mechanical Characteristics
Definition of the mechanical characteristics of the interface between the data terminal equipment and the data communication equipment.

Section 4. Functional Description of Interchange Circuits
Functional description of a set of data, timing, and control interchange circuits for use at a digital interface between data terminal equipment and data communication equipment.

Section 5. Standard Interfaces for Selected Communication System Configurations
Standard subsets of specific interchange circuits defined for a specific group of data communication system applications.

In addition, the standard includes:
Section 6. Recommendations and Explanatory Notes
Section 7. Glossary of New Terms

1.2 This standard includes thirteen specific interface configurations intended to meet the needs of fifteen defined system applications. These configurations are identified by type, using alphabetic characters A through M. In addition, type Z has been reserved for applications not covered by types A through M and where the configuration of interchange circuits is to be specified, in each case, by the supplier.

1.3 This standard is applicable for use at data signaling rates in the range from zero to a nominal upper limit of 20,000 bits per second.

1.4 This standard is applicable for the interchange of data, timing, and control signals when used in conjunction with electronic equipment, each of which has a single common return (signal ground), that can be interconnected at the interface point. It does not apply where electrical isolation between equipment on opposite sides of the interface point is required.

1.5 This standard applies to both synchronous and nonsynchronous serial binary data communication systems.

1.6 This standard applies to all classes of data communication service, including:

1.6.1 Dedicated leased or private-line service, either two-wire or four-wire. Consideration is given to both point-to-point and multipoint operation.

1.6.2 Switched-network service, either two-wire or four-wire. Consideration is given to automatic answering of calls; however, this standard does not include all of the interchange circuits required for automatically originating a connection. (See EIA Standard RS-366, "Interface Between Data Terminal Equipment and Automatic Calling Equipment for Data Communication.")

1.7 The data set may include transmitting and receiving signal converters as well as control functions. Other functions, such as pulse regeneration, error control, etc., may or may not be provided. Equipment to provide these additional functions may be included in the data terminal equipment or in the data communication equipment, or it may be implemented as a separate unit interposed between the two.

1.7.1 When such additional functions are provided within the data terminal equipment or the data communication equipment, this interface standard shall apply only to the interchange circuits between the two classes of equipment.

1.7.2 When additional functions are provided in a separate unit inserted between the data terminal equipment and the data communication equipment, this standard shall apply to both sides (the interface with the data terminal equipment and the interface with the data communication equipment. . .) of such separate unit.

1.8 This standard applies to all of the modes of operation afforded under the system configurations indicated in Section 5, Standard Interfaces for Selected Communication System Configurations.

Pin number	Circuit	Description
1	AA	Protective ground
2	BA	Transmitted data
3	BB	Received data
4	CA	Request to send
5	CB	Clear to send
6	CC	Data set ready
7	AB	Signal ground (common return)
8	CF	Received line signal detector
9	—	(Reserved for data set testing)
10	—	(Reserved for data set testing)
11		Unassigned
12	SCF	Secondary received line signal detector
13	SCB	Secondary clear to send
14	SBA	Secondary transmitted data
15	DB	Transmission signal element timing (DCE source)
16	SBB	Secondary received data
17	DD	Receiver signal element timing (DCE source)
18		Unassigned
19	SCA	Secondary request to send
20	CD	Data terminal ready
21	CG	Signal quality detector
22	CE	Ring indicator
23	CH/CI	Data signal rate selector (DTE/DCE source)
24	DA	Transmit signal element timing (DTE source)
25		Unassigned

Figure A-1 Interface Connector Pin Assignments

Interchange Circuits

Circuit AA: Protective Ground (CCITT 101)
Direction: Not applicable
This conductor shall be electrically bonded to the machine or equipment frame. It may be further connected to external grounds as required by applicable regulations.

Circuit AB: Signal Ground or Common Return (CCITT 102)
Direction: Not applicable
This conductor establishes the common ground reference potential for all interchange circuits except Circuit AA (Protective Ground). Within the data communication equipment, this circuit shall be brought to one point, and it shall be possible to connect this point to Circuit AA by means of a wire strap inside the equipment. This wire strap can be connected or removed at installation, as may be required to meet applicable regulations or to minimize the introduction of noise into electronic circuitry.

Circuit BA: Transmitted Data (CCITT 103)
Direction: TO data communication equipment
Signals on this circuit are generated by the data terminal equipment and are transferred to the local transmitting signal converter for transmission of data to remote data terminal equipment.
The data terminal equipment shall hold Circuit BA (Transmitted Data) in marking condition during intervals between characters or words, and at all times when no data are being transmitted.
In all systems, the data terminal equipment shall not transmit data unless an ON condition is present on all of the following four circuits, where implemented.

1. Circuit CA (Request to Send)
2. Circuit CB (Clear to Send)
3. Circuit CC (Data Set Ready)
4. Circuit CD (Data Terminal Ready)

All data signals that are transmitted across the interface on interchange circuit BA (Transmitted Data) during the time an ON condition is maintained on all of the above four circuits, where implemented, shall be transmitted to the communication channel. . . .

Circuit BB: Received Data (CCITT 104)
Direction: FROM data communication equipment
Signals on this circuit are generated by the receiving signal converter in response to data signals received from remote data terminal equipment via the remote transmitting signal converter. Circuit BB (Received Data) shall be held in the Binary One (Marking) condition at all times when Circuit CF (Received Line Signal Detector) is in the OFF condition.
On a half-duplex channel, Circuit BB shall be held in the Binary One (Marking) condition when Circuit CA (Request to Send) is in the ON condition and for a brief interval following the ON to OFF transition of Circuit CA to allow for the completion of transmission (see Circuit BA, Transmitted Data) and the decay of line reflections. . . .

Circuit CA: Request to Send (CCITT 105)
Direction: TO data communication equipment
This circuit is used to condition the local data communication equipment for

data transmission and, on a half-duplex channel, to control the direction of data transmission of the local data communication equipment.

On one-way-only channels or duplex channels, the ON condition maintains the data communication equipment in the transmit mode. The OFF condition maintains the data communication equipment in a nontransmit mode.

On a half-duplex channel, the ON condition maintains the data communication equipment in the transmit mode and inhibits the receive mode. The OFF condition maintains the data communication equipment in the receive mode.

A transition from OFF to ON instructs the data communication equipment to enter the transmit code. . . . The data communication equipment responds by taking such action as may be necessary and indicates completion of such actions by turning ON Circuit CB (Clear to Send), thereby indicating to the data terminal equipment that data may be transferred across the interface point on interchange Circuit BA (Transmitted Data).

A transition from ON to OFF instructs the data communication equipment to complete the transmission of all data which was previously transferred across the interface point on interchange Circuit BA and then assume a nontransmit mode or a receive mode, as appropriate. The data communication equipment responds to this instruction by turning OFF Circuit CB (Clear to Send) when it is prepared to respond again to a subsequent ON condition of Circuit CA.

Note: A nontransmit mode does not imply that all line signals have been removed from the communication channel. . . .

When Circuit CA is turned OFF, it shall not be turned ON again until Circuit CB has been turned OFF by the data communication equipment.

An ON condition is required on Circuit CA as well as on Circuit CB, Circuit CC (Data Set Ready), and, where implemented, Circuit CD (Data Terminal Ready) whenever the data terminal equipment transfers data across the interface on interchange Circuit BA.

It is permissible to turn Circuit CA ON at any time when Circuit CB is OFF regardless of the condition of any other interchange circuit.

Circuit CB: Clear to Send (CCITT 106)
Direction: FROM data communication equipment

Signals on this circuit are generated by the data communication equipment to indicate whether or not the data set is ready to transmit data.

The ON condition, together with the ON condition on interchange circuits CA, CC, and, where implemented, CD, is an indication to the data terminal equipment that signals presented on Circuit BA (Transmitted Data) will be transmitted to the communication channel.

The OFF condition is an indication to the data terminal equipment that it should not transfer data across the interface on interchange Circuit BA.

The ON condition of Circuit CB is a response to the occurrence of a simultaneous ON condition on Circuit CC (Data Set Ready) and Circuit CA (Request to Send), delayed as may be appropriate to the data communication equipment for establishing a data communication channel (including the removal of the Mark Hold clamp from the Received Data interchange circuit of the remote data set) to remote data terminal equipment.

Where Circuit CA (Request to Send) is not implemented in the data communication equipment with transmitting capability, Circuit CA shall be assumed to be in the ON condition at all times, and Circuit CB shall respond accordingly.

Circuit CC: Data Set Ready (CCITT 107)
Direction: FROM data communication equipment

Signals on this circuit are used to indicate the status of the local data set. The ON condition on this circuit is presented to indicate [all of the following]:

1. The local data communication equipment is connected to a communication channel ("Off Hook" in switched service).
2. The local data communication equipment is not in test (local or remote), talk (alternate voice), or dial* mode. . . .
3. The local data communication equipment has completed, where applicable:
   a. any timing functions required by the switching system to complete call establishment, and;
   b. the transmission of any discreet answer tone, the duration of which is controlled solely by the local data set.

Where the local data communication equipment does not transmit an answer tone, or where the duration of the answer tone is controlled by some action of the remote data set, the ON condition is presented as soon as all the other listed conditions (1, 2, and 3a) are satisfied.

This circuit shall be used only to indicate the status of the local data set. The ON condition shall not be interpreted as either an indication that a communication channel has been established to a remote data station or the status of any remote station equipment.

The OFF condition shall appear at all other times and shall be an indication that the data terminal equipment is to disregard signals appearing on any other interchange circuit with the exception of Circuit CE (Ring Indicator). The OFF condition shall not impair the operation of Circuit CE or Circuit CD (Data Terminal Ready).

When the OFF condition occurs during the progress of a call before Circuit CD is turned OFF, the data terminal equipment shall interpret this as a lost or aborted connection and take action to terminate the call. Any subsequent ON condition on Circuit CC is to be considered a new call.

When the data set is used in conjunction with automatic calling equipment (ACE), the OFF to ON transition of Circuit CC shall not be interpreted as an indica-

*The data communication equipment is considered to be in the dial mode when circuitry directly associated with the call-origination function is connected to the communication channel. These functions include signaling to the central office (dialing) and monitoring the communication channel for call progress or answer-back signals.

tion that the ACE has relinquished control of the communication channel to the data set. Indication of this is given on the appropriate lead in the ACE interface (see EIA Standard RS-366).

Note: Attention is called to the fact that if a data call is interrupted by alternate voice communication, Circuit CC will be in the OFF condition during the time that voice communication is in progress. The transmission or reception of the signals required to condition the communication channel or data communication equipment in response to the ON condition of interchange Circuit CA (Request to Send) of the transmitting data terminal equipment will take place after Circuit CC comes ON, but prior to the ON condition on Circuit CB (Clear to Send) or Circuit CF (Received Line Signal Detector).

Circuit CD: Data Terminal Ready (CCITT 108.2)
Direction: TO data communication equipment

Signals on this circuit are used to control switching of the data communication equipment to the communication channel. The ON condition prepares the data communication equipment to be connected to the communication channel and maintains the connection established by external means (e.g., manual call origination, manual answering, or automatic call origination).

When the station is equipped for automatic answering of received calls and is in the automatic answering mode, connection to the line occurs only in response to a combination of a ringing signal and the ON condition of circuit CD (Data Terminal Ready); however, the data terminal equipment is normally permitted to present the ON condition on Circuit CD whenever it is ready to transmit or receive data, except as indicated below.

The OFF condition causes the data communication equipment to be removed from the communication channel following the completion of any "in proces" transmission. See Circuit BA (Transmitted Data). The OFF condition shall not disable the operation of Circuit CE (Ring Indicator).

In switched-network applications, when Circuit CD is turned OFF, it shall not be turned ON again until Circuit CC (Data Set Ready) is turned OFF by the communication equipment.

Circuit CE: Ring Indicator (CCITT 125)
Direction: FROM data communication equipment

The ON condition of this circuit indicates that a ringing signal is being received on the communication channel.

The ON condition shall appear approximately coincident with the ON segment of the ringing cycle (during rings) on the communication channel.

The OFF condition shall be maintained during the OFF segment of the ringing cycle (between "rings") and at all other times when ringing is not being received. The operation of this circuit shall not be disabled by the OFF condition on Circuit CD (Data Terminal Ready).

Circuit CF: Received Line Signal Detector (CCITT 109)
Direction: FROM data communication equipment

The ON condition on this circuit is presented when the data communication equipment is receiving a signal which meets its suitability criteria. These criteria are established by the data communication equipment manufacturer.

The OFF condition indicates that no signal is being received or that the received signal is unsuitable for demodulation.

The OFF condition of Circuit CF (Received Line Signal Detector) shall cause Circuit BB (Received Data) to be clamped to the Binary One (Marking) condition.

The indications on this circuit shall follow the actual onset or loss of signal by appropriate guard delays.

On half-duplex channels, Circuit CF is held in the OFF condition whenever Circuit CA (Request to Send) is in the ON condition and for a brief interval of time following the ON to OFF transition of Circuit CA. (See Circuit BB.)

Circuit CG: Signal Quality Detector (CCITT 110)
Direction: FROM data communication equipment

Signals on this circuit are used to indicate whether or not there is a high probability of an error in the received data.

An ON condition is maintained whenever there is no reason to believe that an error has occurred.

An OFF condition indicates that there is a high probability of an error. It may, in some instances, be used to call automatically for the retransmission of the previously transmitted data signal. Preferably the response of this circuit shall be such as to permit identification of individual questionable signal elements on Circuit BB (Received Data).

Circuit CH: Data Signal Rate Selector (DTE Source) (CCITT 111)
Direction: TO data communication equipment

Signals on this circuit are used to select between the two data signaling rates in the case of dual rate synchronous data sets or the two ranges of data signaling rates in the case of dual range nonsynchronous data sets.

An ON condition shall select the higher data signaling rate or range of rates.

The rate of timing signals, if included in the interface, shall be controlled by this circuit as may be appropriate.

Circuit CI: Data Signal Rate Selector (DCE Source) (CCITT 112)
Direction: FROM data communication equipment

Signals on this circuit are used to select between the two data signaling rates in the case of dual rate synchronous data sets or the two ranges of data signaling rates in the case of dual range nonsynchronous data sets.

An ON condition shall select the higher data signaling rate or range of rates.

The rate of timing signals, if included in the interface, shall be controlled by this circuit as may be appropriate.

Circuit DA: Transmitter Signal Element Timing (DTE Source) (CCITT 113)
Direction: TO data communication equipment

Signals on this circuit are used to provide the transmitting signal converter with signal element timing information.

The ON to OFF transition shall nominally indicate the center of each signal element on Circuit BA (Transmitted Data). When Circuit DA is implemented in the DTE, the DTE shall normally provide timing information on this circuit whenever the DTE is in a power ON condition. It is permissible for the DTE to withhold timing information on this circuit for short periods provided Circuit CA (Request to Send) is in the OFF condition. (For example, the temporary withholding of timing information may be necessary in performing maintenance tests within the DTE.)

Circuit DB: Transmitter Signal Element Timing (DCE Source) (CCITT 114)
Direction: FROM data communication equipment

Signals on this circuit are used to provide the data terminal equipment with signal element timing information. The data terminal equipment shall provide a data signal on Circuit BA (Transmitted Data) in which the transitions between signal elements nominally occur at the time of the transitions from OFF to ON condition of the signal on Circuit DB. When Circuit DB is implemented in the DCE, the DCE shall normally provide timing information on this circuit whenever the DCE is in a power ON condition. It is permissible for the DCE to withhold timing information on this circuit for short periods provided Circuit CC (Data Set Ready) is in the OFF condition. (For example, the withholding of timing information may be necessary in performing maintenance tests within the DCE.)

Circuit DD: Receiver Signal Element Timing (DCE Source) (CCITT 115)
Direction: FROM data communication equipment

Signals on this circuit are used to provide the data terminal equipment with received signal element timing information. The transition from ON to OFF condition shall nominally indicate the center of each signal element on Circuit BB (Received Data). Timing information on Circuit DD shall be provided at all times when Circuit CF (Received Line Signal Detector) is in the ON condition. It may, but need not, be present following the ON to OFF transition of Circuit CF...

Circuit SBA: Secondary Transmitted Data (CCITT 118)
Direction: TO data communication equipment

This circuit is equivalent to Circuit BA (Transmitted Data) except that it is used to transmit data via the secondary channel.

Signals on this circuit are generated by the data terminal equipment and are connected to the local secondary channel transmitting signal converter for transmission of data to remote data terminal equipment.

The data terminal equipment shall hold Circuit SBA (Secondary Transmitted Data) in marking condition during intervals between characters or words and at all times when no data are being transmitted.

In all systems, the data terminal equipment shall not transmit data on the secondary channel unless an ON condition is present on all of the following four circuits, where implemented:

1. Circuit SCA (Secondary Request to Send)
2. Circuit SCB (Secondary Clear To Send)
3. Circuit CC (Data Set Ready)
4. Circuit CD (Data Terminal Ready)

All data signals that are transmitted across the interface on interchange Circuit SBA during the time when the above conditions are satisfied shall be transmitted to the communication channel...

When the secondary channel is usable only for circuit assurance or to interrupt the flow of data in the primary channel (less than 10 baud capability), Circuit SBA (Secondary Transmitted Data) is normally not provided, and the channel carrier is turned ON or OFF by means of Circuit SCA (Secondary Request to Send). Carrier OFF is interpreted as an interrupt condition.

Circuit SBB: Secondary Received Data (CCITT 119)
Direction: FROM data communication equipment

This circuit is equivalent to Circuit BB (Received Data) except that it is used to receive data on the secondary channel.

When the secondary channel is usable only for circuit assurance or to interrupt the flow of data in the primary channel, Circuit SBB is normally not provided. See interchange Circuit SCF (Secondary Received Line Signal Detector).

Circuit SCA: Secondary Request to Send (CCITT 120)
Direction: TO data communication equipment

This circuit is equivalent to Circuit CA (Request to Send) except that it requests the establishment of the secondary channel instead of requesting the establishment of the primary data channel.

Where the secondary channel is used as a backward channel, the ON condition of Circuit CA (Request to Send) shall disable Circuit SCA and it shall not be possible to condition the secondary channel transmitting signal converter to transmit during any time interval when the primary channel transmitting signal converter is so conditioned. Where system considerations dictate that one or the other of the two channels be in transmit mode at all times but never both simultaneously, this can be accomplished by permanently applying an ON condition to Circuit SCA (Secondary Request to Send) and controlling both the primary and secondary channels, in complementary fashion, by means of Circuit CA (Request to Send). Alternatively, in this case, Circuit SCB need not be implemented in the interface.

When the secondary channel is usable only for circuit assurance or to interrupt the flow of data in the primary data channel, Circuit SCA shall serve to turn ON the secondary channel unmodulated carrier. The OFF condition of Circuit SCA shall turn OFF the secondary channel carrier and thereby signal an interrupt condition at the remote end of the communication channel.

Circuit SCB: Secondary Clear to Send (CCITT 121)
Direction: FROM data communication equipment

This circuit is equivalent to Circuit CB (Clear to Send) except that it indicates the availability of the secondary channel instead of indicating the availability of the primary channel. This circuit is not provided where the secondary channel is usable only as a circuit assurance or an interrupt channel.

Circuit SCF: Secondary Received Line Signal Detector (CCITT 122)
Direction: FROM data communication equipment

This circuit is equivalent to Circuit CF (Received Line Signal Detector) except that it indicates the proper reception of the secondary channel line signal instead of indicating the proper reception of a primary channel received-line signal.

Where the secondary channel is usable only as a circuit assurance or an interrupt channel (see Circuit SCA, Secondary Request to Send), Circuit SCF shall be used to indicate the circuit assurance status or to signal the interrupt. The ON condition shall indicate circuit assurance or a noninterrupt condition. The OFF condition shall indicate circuit failure (no assurance) or the interrupt condition.

Apéndice C

Microprocesadores Diversos

650X, 65C0X

AVAILABILITY: Now
COST: The prices for both NMOS and CMOS were said to have dropped to less than $1. However, "legitimate" US price said to be $2 to $3 for NMOS and twice that for CMOS.
SECOND SOURCE: Rockwell, California Micro Devices, NCR, and WDC (Western Design Center). WDC created some of the CMOS designs, which it has licensed (UMC in Taiwan, ITT-Intermetall in West Germany, etc), which is one explanation why second sources have proliferated.
CORE: WDC claims to have developed the semicustom 6502 core as NCR and others now use it. Many suppliers now specify it as part of their cell libraries.

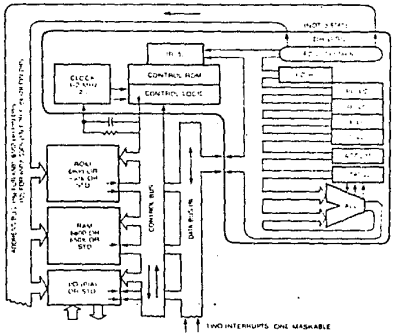
Description: Original design team's goal was to achieve as much PDP-11-style addressing capability as would fit in an economical chip. Because of the µP's short 8-bit index registers, it is optimally suited only to applications requiring access of smaller blocks of memory (although it benchmarks ahead of most other 8-bit µPs with respect to its speed of execution of high-level languages such as Basic and Pascal). New CMOS parts also have small economical die that gets still smaller with today's finer geometries. See 6500/1 for 1-chip versions and 65SC816/802 for 16-bit-internal version.

8-BIT NMOS AND CMOS

Originator Commodore (Westchester, PA) no longer sells chips to the merchant market. Contact second sources.

Status: The falling share of market for this µP appears to indicate that it has reached the end of its lifecycle. However, the architecture lives on in the form of 1-chip versions (see 6500/1 and especially the 50740) and ASIC versions. Some of these have very large unit volumes, so the 6502 architecture may remain, by volume, the leading 8-bit architecture in the world. WDC is shipping 8-MHz parts and claims to be developing even faster chips.

HARDWARE — CHARACTERISTICS — SOFTWARE



Notes on CMOS versions:
1. CMOS 65CXX family members are slight enhancements of NMOS counterparts and can serve as plug-in replacements.
2. Among hardware enhancements are new 4-phase clock that gives decreased memory access time and a memory-lock (ML) output and bus-enable (BE) input that simplify multiprocessor designs.
3. Among the software enhancements are the treating of all unused op codes as NOPs and removing the page-boundary restrictions on JMP indirect.
4. Decimal mode is automatically set Off upon reset or interrupt, and the N, V, and Z flags are made active during decimal mode.
5. A BRK followed by interrupt is executed.
6. See instruction set for comments on new instructions.

I—DATA-MANIPULATION INSTRUCTIONS
Arithmetic and logical. Decimal mode via control bit in status register. Can operate on locations in memory space (which can be either RAM or I/O ports). CMOS parts have bit manipulation.
II—DATA-MOVEMENT INSTRUCTIONS
True indexed addressing, though index offset limited to 8 bits in two CPU registers—X and Y. Short-form addressing to zero page. Has two sophisticated indirect-indexed and indexed-indirect instructions for handling tables. CMOS parts have indexed absolute indirect and zero-page indirect.
III—PROGRAM-MANIPULATION INSTR
Conditional branches with signed relative addresses. Nonmaskable and/or maskable interrupt, depending on model. CMOS parts have branches on bit test.
Stack pointer for implementing 256-byte LIFO in external RAM.
IV—PROGRAM-STATUS-MANIP INSTR
Push and pull status register from memory stack. Set and clear carry, decimal mode and interrupt bits (6502 and 6512 have external input to one status bit, useful for handshaking with peripherals).
V—POWER-SAVING INSTRUCTIONS
WAIT and STOP on 65C02 respectively stop processor and disconnect clock to lower power consumption.

Specification summary: Common-memory architecture with instructions, data, and I/O in same 64k-byte space; 57 instructions (68 for CMOS). Many instructions provide choice of 13 PDP-11-type addressing modes (15 for CMOS). Advanced indexed-indirect addressing mode. NMOS and CMOS silicon-gate, depletion-mode circuitry requires one 5V, 250-mV supply. Some CMOS parts can run at 8-MHz clock (125 nsec/cycle). CMOS parts require 4 mA/MHz for operation and 10 µW for standby.

HARDWARE — SUPPORT — SOFTWARE

From Rockwell: LCE low-cost emulator ($1250) that will optionally interface to IBM PC host.
From Western Design Center: Toolbox Design System in-circuit emulator (ICE) runs with an Apple IIGS host and can communicate with an IBM PC via a serial link ($4995).
From California Micro Devices: GEM-I in-circuit emulator package ($3750) capable of interfacing with a variety of host computers including ISIS development system and Apple. Functions as a stand alone assembler and disassembler using a nonintelligent terminal. Evaluation board for 65SC150 ($499) that functions as in-circuit system when coupled with GEM-I.
From NCR: Hardware emulator interfaces to Apple IIe through RS-232C. Allows complete in-circuit software debug.
From Dynatem (Irvine, CA): AIM-65 single-board computer and RIM industrial modules.

From Rockwell: Cross software for Intel ISIS-II and personal development system ($250). Support (in firmware) for assembly ($35), monitor ($65), Basic ($65), PL/65 ($85), Forth ($65), Pascal-"instant" ($100), math package ($30), and disk operating system ($50).
From California Micro Devices: 65SC00 macroassembler for Apple Computer ($100), assembler for Intel ISIS ($1800), and Fortran assembler ($1800).
From NCR: Monitor for use in conjunction with emulator. Supports breakpoint, change memory and registers, software trace and real-time execution, etc.
From others: Because the 6500 has been so widely used, there are innumerable sources of software at different language levels; for example, Byte Works (Albuquerque, NM); S-C Software (Dallas, TX); Roger-Wagner Publishing (El Cajon, CA), and 2500 AD (Aurora, CO).

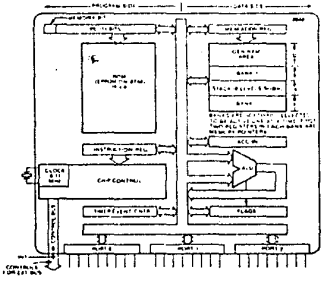
AVAILABILITY: Now.
COST: Masked-ROM parts less than $2 in high volume (100k qty). EPROM parts cost $18 in 100 qty. CMOS parts cost as low as $3 in 100k qty. Windowless-PROM parts cost $8 in 5k qty.
SECOND SOURCE: Toshiba, NEC, Signetics/Philips, National, Oki, Siemens, Fujitsu, GE-Intersil, UMC (Taiwan), with volume being spread out among suppliers.
CORE: Zymos has been using 80C49 as core for semicustom for a number of years. Others are following as 8048/49 combines widespread popularity with reasonably small core size.

Intel Corp
Embedded Controller Operation
5000 W Chandler Blvd
Chandler, AZ 85226
Phone (602) 961-8051

Description: Broad family of 1-chip controller-type µCs, including version that can function as slave (8041). Basic models don't have serial communication ports (some versions from Philips do), but they can use 8080/85 peripherals for I/O expansion. See 8051 listing for enhanced version.

Status: Intel is still bullish about its 8048, saying total family shipments were reported to be 70 million in '87. However, we note that Intel chose the 8051 over the 8048 as the kick-off core for ASIC, and Intel says it has no definite plans to ever use the 8048 as an ASIC core.

HARDWARE CHARACTERISTICS SOFTWARE



Notes:
1. Diagram is for basic 8048. Table indicates some of other basic parts, most of which exist in both NMOS and CMOS.
2. CMOS parts are designated 80C48, 80C49, 80C50, etc.
3. There are many other variations on basic 8048 among the many suppliers. For example, Intel's 8041/42 chips are software compatible but can be configured as slaves to host µPs for interface applications. The National NS 405/455 uses the 8048 core as basis of a terminal controller. Siemens has telecomm-oriented 80C382/482. A number of semicustom houses use the 8048 as a core processor in their libraries.

PART NO	MEMORY (BYTES) ROM	EPROM	RAM	PACKAGE PINS PARALLEL I/O	TOTAL
8035	0	0	64	3x8	40
8048	1k	0	64	3x8	40
8748	0	1k	64	3x8	40
8039	0	0	128	3x8	40*
8049	2k	0	128	3x8	40*
8749	0	2k	128	3x8	40*
8040	0	0	128	3x8	40
8050	4k	0	256	3x8	40

*ALSO AVAILABLE IN 44-LEAD PLCC PACKAGE.

I—DATA-MANIPULATION INSTRUCTIONS
Arithmetic and logic
Bit set and reset
Two working banks of 8-bit registers
II—DATA-MOVEMENT INSTRUCTIONS
Both internal and external RAM are fully accessible by instruction set.
Indirect and direct data fetches
III—PROGRAM-MANIPULATION INSTR
Decrement and skip if zero
Over 20 conditional branches
8-level stack with expansion capability
Two vectored interrupts
Two programmable flag bits under software control
IV—PROGRAM-STATUS-MANIP INSTR
Status word is fully accessible and is stored in the stack

Note: Described are the 90 basic instructions for the 8048/8748.

Specification summary: Split-memory architecture with 1k to 4k bytes of program ROM (or EPROM) on chip and 64 to 256 bytes in separate space, also on chip. I/O has its own space and instructions to operate directly on I/O ports. All spaces are expandable: program memory to 4k bytes, data memory to 256 bytes, I/O to unlimited amounts. I/O can use 8080/85 peripherals. Devices have 8-level stack for subroutine nesting and interrupt response. Dual banks of working registers allow rapid context switching. Family members execute their 1- and 2-cycle instructions at 1-cycle times ranging from 1.36 to 15 µsec. NMOS 5V technology in 40-pin DIP and 44-pad chip carriers; UV-erasable ROMs (EPROMs) and windowless PROM parts are available. CMOS versions available with idle and power-down features and optional flatpack packages.

HARDWARE SUPPORT SOFTWARE

From Intel: Intel now plays down 8048 support, saying that there are now numerous third-party OEM suppliers of PC-hosted emulators for the 8048 family.
From NEC: Ekakit 84C-1 stand-alone emulator (less than $2000).

From Intel: ASM-48 package with linker to run on Intel microcomputer development systems running ISIS operating system ($1500 for 8-copy license)
From others: Because of the broad-based popularity of this family, dozens of independent sources of development and application software exist, including support on universal development systems from Tektronix, Applied Microsystems, etc.
Program library: Insite Library contains variety of application programs.

8051/8052 FAMILY

8-BIT NMOS AND CMOS

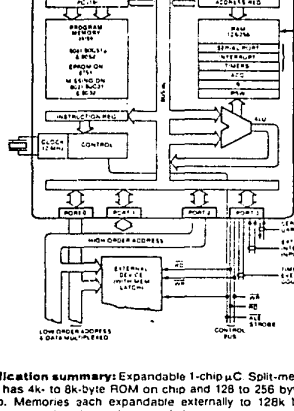
AVAILABILITY: Now for 8051, 80C51, 8031, 80C31, 8751, 87C51, 8032, and 8052, as well as special versions from second sources (see notes). Many other versions such as 80C451 and 83C451 available from Intel.
COST: $4.50 in 100k qty for 8051; $32 in 1k qty for 8751; $6.50 in 100 qty for 80C51; $5.35 in 100k qty for 8052; $44 for 87C51; $70 for EEPROM UPI-452 slave version. (1k qty.)
SECOND SOURCE: Siemens, Signetics/Philips, AMD, Fujitsu, Oki, and Harris-Matra (France) licensed.
CORE: Intel's ASIC Components Group is using the 8051 as its starting µP core. RCA and Fujitsu also using it as an ASIC core.

Intel Corp
Embedded Controller Operation
5000 W Chandler Blvd
Chandler, AZ 85226
Phone (602) 961-8051

Description: Expandable single-chip "controller," an enhanced version of the same supplier's widely used 8048 family. Architecturally, it features the more "regular" nonpaged form of addressing for easier programming, more interrupts with extra RAM register banks to service them, increased stack depth, and new instructions such as multiply, divide, and compare. In peripheral support, it adds a full-duplex hardware UART and enlarged timer/counter capability.

Status: Generally thought of as the leader among the newer, more powerful 8-bit 1-chip µCs. It faces stiff competition from both high-end 8-bit µCs, such as Mitsubishi's 50740 version of the 8[illegible] Motorola's 68HC11, NEC's 7811, Hitachi's 647180, and National's COP800, as well as from the new 16-bit µCs, such as Intel's own 8096 and National's 16040.

HARDWARE — CHARACTERISTICS — SOFTWARE



Specification summary: Expandable 1-chip µC. Split-memory architecture has 4k- to 8k-byte ROM on chip and 128 to 256 bytes of RAM on chip. Memories each expandable externally to 128k bytes. Four 8-bit ports on chip, but only one of these remains a port when all off-chip expansions and on-chip special functions are used. Special functions included on chip are full-duplex hardware UART (to 500k baud), two or three 16-bit timer/counters, and interrupt system to service these internal functions along with two external interrupts with 3-to 7-µsec latency. Instructions are a superset of the 8048s, with paged addressing eliminated. At 12-MHz clock, most instructions take 1 µsec; multiply or divide requires 4 µsec. Supplier's high-density HMOS silicon-gate n-channel technology used to achieve small die size and good speed. Packaged in 40-pin DIP and 44-pad chip carriers. 8051 is also available in CMOS (80C51) with 12- or 16-MHz performance and idle/power-down modes.

I—DATA-MANIPULATION INSTRUCTIONS
Arithmetic, including add, subtract, multiply, and divide
Bit manipulation, including complex tests on bits (and branching on results)
II—DATA-MOVEMENT INSTRUCTIONS
Register addressing for the eight working registers in the four register banks
Direct, immediate, and indirect data addressing for more general data accessing
Table look-up in ROM via data pointer
III—PROGRAM-MANIPULATION INSTR
Depth of subroutining limited only by available space in 128-or 256-byte on-chip RAM
Conditional jumps on status-register flags
Conditional jumps on comparisons
Vectored interrupts to service two external interrupts, timers, and UART
IV—PROGRAM-STATUS-MANIP INSTR
CPU's program-status word fully accessible via software. Status bits in timer and UART also software accessible

Notes:
1. The 14 members of the 8051 family have between 128 and 256 bytes of RAM and differ mainly in their amount and form of on-chip ROM. The 8051 and 80C51 incorporate 4k bytes of masked ROM. The 8751 and 87C51 have 4k bytes of EPROM. The 83C51FA has 8k bytes of masked ROM and a programmable counter array (PCA). The 87C51FA has 8k bytes of EPROM and a PCA. The 8031, 80C31, and 80C51FA have no on-chip ROM. (Hence, because it must use ports to access external memory, only port 1 is available for I/O.) The 8052 has 8k bytes of masked ROM. The 8032 has no on-chip ROM. The 8052 and 8032 have 256 bytes of on-chip RAM.
2. The 8051's so-called Boolean-processor capabilities refer to the way instructions can single out bits in RAM, accumulators, I/O registers, etc, and perform complex bit tests and comparisons, then execute relative jumps based on results.
3. The slave version of the 80C51, the UPI-452, is counterpart of UPI-42 (8041/42) for 8048 family. It is intended for software-customizable interfaces.
4. Intel has one model of 8052 preprogrammed with a full Basic interpreter.
5. Siemens has developed proprietary enhancements called 80515/535. They feature 16K ROM, with additional I/O ports, 15-µsec 8-bit A/D with eight input channels, 12 interrupts with four programmable priority levels. They are 12-MHz (1-µsec cycle) NMOS, packaged in TAB (Micropack).

HARDWARE — SUPPORT — SOFTWARE

From Intel: ICE-5100/252 in-circuit emulator ($5495) supports the entire MCS-51 family including 8051, 8052, and 80C52. Comes with macroassembler and editor. The emulator is hosted on an IBM PC AT/XT running DOS 3.1 or later, as well as Intellec Series III/IV development systems. ICE-51 in-circuit emulator ($4995) hosted on Series III/IV Intellec supports 8051 at 12 MHz.
SDK-51 System Design Kit ($950) is a single-board computer for low-cost development of 8051 applications.
From Siemens: Meta-ICE-80515 in-circuit emulator for 80515, hosted on IBM PC.

From Intel: ASM-51 and PL/M-51, both containing a relocation and linkage utility, are available for the IBM PC and Intel microcomputer development systems.
From others: A number of third-party software suppliers have developed C compilers for 8051 that have special features suited to microcontroller applications. Among these are Micro Computer Control (Hopewell, NJ) for $1495 and Archemides Software (San Francisco, CA) for $851. Both are hosted on IBM PC.

8085AH/80C85

8-BIT NMOS AND CMOS

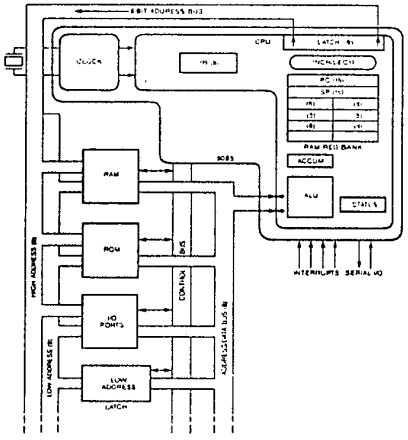
AVAILABILITY: Now for both in NMOS 8085 and for CMOS 80C85 versions.
COST: Prices for these older multisourced parts have dropped to $1 and below, with prices as low as $0.65 for volume purchases. CMOS parts, especially faster ones, are more expensive. Radiation-hardened CMOS parts are very expensive ($300 to $800).
SECOND SOURCE: 8085: NEC and Toshiba (and Intel) had most of market between them in '86, but Mitsubishi, Siemens, and AMD also shipped parts. 80C85: Oki active with Harris and Calmos supplying nuclear-radiation-hardened CMOS to military and aerospace customers.

Description: Based on the older 8080 μP, this family has proven a good general-purpose, midrange μP, though not the most efficient one for small programs. 8085 executes 8080 instructions, but with simpler hardware. Z80 (see elsewhere in this directory) is an enhanced 8080 but has different package pinouts and bus operation. New 8088 (see elsewhere in this directory) is vaguely software compatible, but 8-bit-bus 8088 version of 8086 can interface to 8080 and 8085 peripherals.

Intel Corp
3065 Bowers Ave
Santa Clara, CA 95051
Phone (408) 987-8080

Status: The venerable 8080—the μP that gave legitimacy to the μP revolution—is obsolete. It had less than 0.28% of the 8-bit-μP market in '86, according to Dataquest. The 8085 is also starting to fall off, according to Dataquest figures. Still, it was in second place behind the Z80.

HARDWARE — CHARACTERISTICS — SOFTWARE



I—DATA-MANIPULATION INSTRUCTIONS
Arithmetic and logic
BCD arithmetic
Double-precision operations (instructions string two data bytes together as 16-bit word)
II—DATA-MOVEMENT INSTRUCTIONS
Uses three pairs of so-called GP registers as pointers in CPU RAM bank to address low- and high-order bits of 16-bit memory address. Can perform multiple indexing with these, but takes additional steps compared with classical index-register concept. 8085 has two additional instructions—RIM and SIM—that interface with new serial-I/O pins (as well as interrupt system)
III—PROGRAM-MANIPULATION INSTR
Uses stack pointer (SP) to create LIFO stacks in external RAM for unlimited subroutine nesting
All GP registers can be incremented and decremented
Multiple-interrupt capability
Bus controls allow addition of DMA
IV—PROGRAM-STATUS-MANIP INSTR
Software access to status register

How 8085 differs from 8080:
8085 has on-chip clock, needs only a 5V supply, and has relaxed memory-access time. But because it multiplexes lower eight bits of address on data bus, it's not pin compatible with 8080. New pins gained by multiplexing implement address-latch strobe, four additional interrupts, and two serial-I/O lines. For small "few-chip" μP systems, a designer can use 8155/56 and 8355/8755 combo chips with built-in address latches.

Specification summary: Common instruction and data architecture (64k bytes) with optionally separate I/O space (256 addresses). Three 16-bit pointer registers allow efficient addressing of 64k-byte main-memory space. 78 basic instructions with 2-μsec typ register-to-accumulator addition execute time. 8085A has on-chip clock and needs only 5V. High-speed (5 MHz) and CMOS versions of the 8085A are available.

HARDWARE — SUPPORT — SOFTWARE

Most of the vendors of third-party μP development systems have included 8080/8085 development components as a routine part of their catalog. Typically, they use IBM PCs as host.

Most of the many companies that supply 8080/8085 development systems also supply the software. Also, many software houses have 8080/8085 software in every conceivable category.

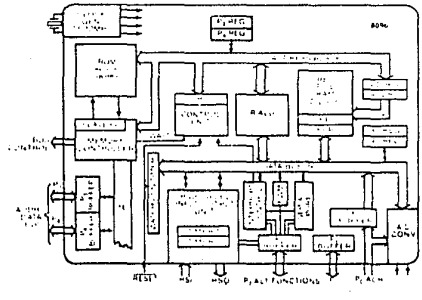
AVAILABILITY: NMOS 8096 family is in production. The 8098 chip offers an 8-bit external bus. The EPROM version is also available. The higher-performance CMOS version 80C196 is now in production.
COST: Less than $8 in 10k qty.
SECOND SOURCE: Signetics/Philips.

Description: Highly integrated 16-bit microcontroller combining 16-bit CPU with extensive I/O handling. On-chip memory includes 8k bytes of ROM and 232 bytes of register-file RAM. I/O capabilities include an 8-channel, 10-bit ADC, full-duplex UART, 8-level priority interrupt, pulse-width-modulated output, high-speed pulsed I/O, four 16-bit software timers, five 8-bit I/O ports, and a watchdog timer.

Intel Corp
Embedded Controller Operation
5000 W Chandler Blvd
Chandler, AZ 85226
Phone (602) 961-8051

Status: The earliest of the 16-bit microcontrollers continues to have a large share of 16-bit market. However, because that market is still young, it's too early to tell whether this will be another case of Intel dominance. The only other 16-bit µC shown with any volume was the Thomson-Mostek 68200, but following the Thomson-SGS merger it has dropped out of the picture. Meanwhile, the National 16040 and NEC 783XX (78312), which are newer designs, are being aggressively marketed and could pose threats. Actually, with the advent of the ASIC approach, the definition of this market is blurring; for now, any µP that is in core form in an ASIC library could have memory added and become a "µC," even Intel's own 80188/80186s.

HARDWARE CHARACTERISTICS SOFTWARE



Hardware Notes:
1. The initial NMOS 8096 family consists of parts that come with or without A/D converters (and S/H circuits) and onboard ROM, and with either 48 I/O lines (68-pin package) or 32 I/O lines (48-pin package). They have option of either 8- or 16-bit system bus. The 8098 offers an external 8-bit bus. The 8k-byte EPROM version has onboard programming capability and read/write selectivity.
2. New CMOS version 80C196 has 2× NMOS performance.
3. Four high-speed trigger inputs record times at which external events occur. Storage in 8-deep FIFO.
4. Six high-speed pulse outputs can trigger external events at preset times. Commands are stored in 8-deep content-addressable memory. Output section can concurrently run as many as four software timers simultaneously.
5. 16-bit watchdog timer allows recovery from hardware or software error.

I—DATA-MANIPULATION INSTRUCTIONS
8- and 16-bit signed and unsigned arithmetic in binary, including multiply and divide
Logicals
Bit, byte, word, and double-word operations
II—DATA-MOVEMENT INSTRUCTIONS
Addressing modes include Direct, Immediate, Indirect, Indexed, and Indirect with Autoincrement
Load and store, push and pop
III—PROGRAM-MANIPULATION INSTR
Has calls, jumps, and returns
Conditional jumps upon Boolean functions of flags within ±128 bytes of instruction
Iteration control of loops
IV—PROGRAM-STATUS-MANIP INSTR
Zero, sign, overflow, carry, overflow trap, interrupt enable, sticky bit (records previous value of carry during right shifts)
Can set and clear some bits

Specification summary: 16-bit µC with split-memory architecture and 8k-byte ROM and 232 bytes of register-file RAM on chip. External memory expandable to 64k bytes, with data bus dynamically programmable as 8 or 16 bits. Register-to register architecture with ALU operating directly on register file. Has 8-channel, 10-bit A/D converter, four 16-bit software timers, PWM output, five 8-bit I/O ports, full-duplex serial port and high-speed pulse I/O ports. At 12-MHz clock, 16-bit addition takes 1 µsec, 16 × 16 multiply or 32/16 divide takes 6.5 µsec. Average instruction-execution time equals 1 to 2 µsec. New CMOS parts have 2× performance of NMOS. In 48-pin DIP, 68-pin PLCC or 68-pin pin-grid array.

HARDWARE SUPPORT SOFTWARE

From Intel: Low-cost development kit ($2695) includes iSBE-96 emulator board and ASM-96 macroassembler and runs on IBM PC host as well as Intellec Series III and IV. Real-time emulation to 12 MHz. VLSICE-96 advanced emulator provides real-time emulation up to 12 MHz and is hosted on IBM PC as well as Intellec Series III and IV. Intel also offers support hardware for the 80C196 chip.
Programming support for EPROM versions supplied through Intel's line of universal PROM programmers as well as third-party programmers from companies such as Data I/O.

From Intel: Macroassembler (ASM-96), PL/M-96 and C-96 compilers. PL/M and C supply hardware-control features such as interrupts. Each software package includes relocation/linkage utility (RL-96), library management utility (LIB-96), object-to-hex conversion utility (OH-96), and FPAL-96, a 32-bit floating-point utility. Software packages run on an IBM PC and compatible computers. $750 for a single-user license.
From Archimedes (San Francisco, CA): ANSI C-8096 compiler with additional features like control of interrupt. Hosted on IBM PC ($995), MicroVAX ($3995), and VAX ($5995).
From Cybernetic Micro Systems (San Gregorio, CA): Graphic programming and simulation aids that run on IBM PC ($295 and $995).

8086/8088, 80186/80188

8/16-BIT NMOS AND CMOS

AVAILABILITY: Now for both NMOS and CMOS 8086/88. Now for 8- and 10-MHz 80186. Now for 6-, 8-, and 10-MHz 80188. Now for 10-, 12.5-, and 16-MHz 80C186.
COST: At 100 qty, under $5 for 8086/88, under $10 for 80186/186 in PLCC. $18 for 80C186 in 1k qty.
SECOND SOURCE: For 8086/8088: AMD, Harris, Matra-Harris, Fujitsu, Siemens, Oki. For 80186/8188: AMD, Fujitsu, Siemens.
CORE: Intel's ASIC group says it will be incorporating 80C186 in its cell library.

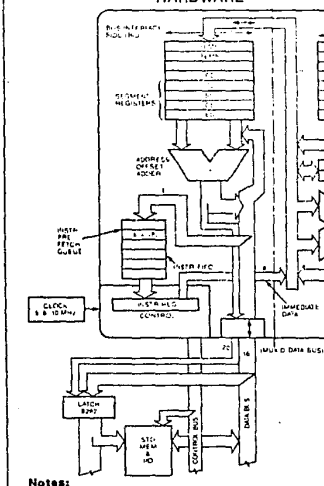
Description: Supplier's objective when 8086 was introduced back in '78 was to offer a machine that matched performance of latest mid-range minis but retained some upward compatibility with widely used 8080/85. 8088 is intended as highest performance 8-bit µP. Floating-point math coprocessor (8087) available to enhance performance. 80186 and 80188 are intended as higher-integration counterparts of 8086 and 8088. They incorporate some of the often-used support-chip functions on CPU chip, somewhat in anticipation of ASIC standard-cell trend (and in fact Intel plans to add 8086 family members to its ASIC cell library to give customers a chance to design their own higher-integration combinations).

Intel Corp
Embedded Controller Operation
5000 W Chandler Blvd
Chandler, AZ 85226
Phone (602) 961-8051

Intel Corp
3065 Bowers Ave
Santa Clara, CA 95051
Phone (408) 987-8080

Status: Next to the 8080/Z80 family group, the 8086 family has been the most successful µP family. The most visible application for the family has been in the IBM PC and its many clones. The 80186/88 high-integration versions covered here were intended for PC applications, but the 80188/86 never caught on with PC makers. Now that new designs for the PC market have switched to the 80286 and 80386, Intel is directing the 80186/88 at embedded applications, and is claiming success—2000 design wins for 80186 and 600 design wins for new 80C186.

HARDWARE CHARACTERISTICS SOFTWARE



Notes:
1. Diagram is for initial family member, 8086.
2. 8088 is downgraded version of 8086. It has only 8-bit-wide external data output bus (only 8 lower bits of address bus are multiplexed for data). Some pin functions have been changed. Prefetch queue is only 4 bytes (to prevent overuse of bus). Instruction execution is slower as all 16-bit fetches and writes take 4 extra cycles.
3. 80186/88 integrate support functions on chip to reduce system costs. Functions added are clock generation, 2-channel DMA, interrupt controller, 3 16-bit timers, memory- and peripheral-chip-select logic, and wait-state generator. 80188 is 8-bit external data-bus version of 80186; it has shortened prefetch queue, and instructions take longer.
4. Math coprocessors implementing IEEE-754 floating-point standard are part of family.

I—DATA-MANIPULATION INSTRUCTIONS
8- and 16-bit signed and unsigned arithmetic in binary or decimal, including multiply and divide
Logicals
Bit, byte, word, and block operations
II—DATA-MOVEMENT INSTRUCTIONS
Addressing modes include literal, relative (to register and to segment), register, base plus index, and base relative indexed
Use of segment registers. Programmer can, through software, set up four areas in memory with four segment registers—a program area, a stack area, and two data areas. These areas need not be full 64k, and they can overlap. Programmer can alter the four area locations by modifying the segment-register contents.
III—PROGRAM-MANIPULATION INSTR
Has call, jump, and return instructions both inside program segments and to different segments. Intrasegment call and jump use self-relative displacement for position-independent code. Conditional jump upon Boolean functions of flags within ± 128 bytes of instruction. Iteration control of loops, a repeat prefix for rapid iteration in hardware-repeated string operations.
Note: Jumps can occupy varying amounts of execution time, because with BIU's instruction prefetch, the program counter can be ahead of itself.
IV—PROGRAM-STATUS-MANIP INSTR
In addition to 8080/85 flags: overflow, interrupt enable, direction (for strings), and single-step trap flags.

Notes:
1. Enhanced CPU in 80186/188 includes new instructions: Pusha, Popa handle all registers at once. Immediate mode for Push and Imul; Ins and Outs for strings. Bound for address ranging; Enter and Leave for stack-frame saving and restoration.
2. Further enhancements in 80C186 include power saving with programmed clock division, and DRAM-control circuit.

Specification summary for 8086/88: 16-bit CPU that can reach 1M byte using "segment" address-extension registers. Register-to-register operations execute at 0.6 µsec with 5-MHz clock (0.37 µsec with 8-MHz clock). HMOS ion-implanted, depletion-load, silicon-gate circuitry; requires 5V at 340 mA (substrate bias generated on chip). In 40-pin DIP, device is pin programmed to switch 8 pins from minimum to maximum external system mode. Harris CMOS 8086 dissipates only 10 mA/MHz when running, and clock can be stopped for 500 µA standby.

Specification summary for 80186/188: Highly integrated µPs that combine functions of most common iAPX 86 system components onto one chip. Have same memory reach as 8086/88 but with improved execution times on some instructions. HMOS II ion-implanted, depletion-load, silicon-gate circuitry requires 5V at 300 mA (90 mA and less for CMOS). Housed in 68-pin JEDEC Type A ceramic leadless chip carrier and a ceramic pin-grid array. Plastic leaded chip carrier also.

HARDWARE SUPPORT SOFTWARE

From Intel: ICE in-circuit emulator ($7995) supports 8086/8088 and 80186/80188 to 10 MHz. Emulators are hosted on IBM PC and Intellec Series III/IV development systems. ICE186/188 in-circuit emulator supports 16-MHz development operation.
From others: Because of popularity, family is widely supported by third-party universal development systems.

From Intel: Macroassembler, including linker, locator, mapper, and librarian. High-level-language compilers include PL/M, C, Fortran, and Pascal. Pscop-86 provides source-level debug with full source-code display. Hosts include PC-DOS, VAX/VMS, and Intel development systems. Prices start at $750 (for DOS versions).
From others: Because of wide base of 8086/8088-based systems, and in particular the IBM PC, there exists a lot of third-party software of all sorts, enough to fill whole catalogs. Check with Intel and various trade journals.

80286

16-BIT NMOS

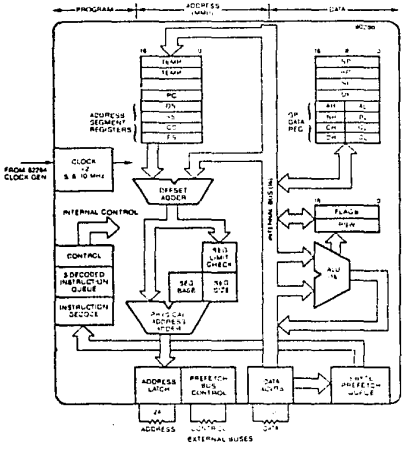
AVAILABILITY: In production with 8, 10, 12.5 and 16 MHz (AMD for 16 MHz). CMOS 80C286 12.5 MHz in production and 20 MHz sampling.
COST: In 100 qty: $37 for 8 MHz, $59 for 10 MHz, $95 for 12.5 MHz, and $150 for 16 MHz in LCCs (PGAs more). For 80C286: $125 for 10 MHz and to $170 for 16 MHz, also in 100 qty.
SECOND SOURCE: AMD, Siemens, and Fujitsu. Harris for CMOS 80C286.

Description: An evolutionary extension of the 8086 with special capabilities for multitasking systems. Has on-chip memory-management and protection functions that support intertask isolation, program and data security, and 4 levels of privilege within a task. Memory management supports as much as 1G bytes of virtual-address space per task, mapped into a 16M-byte physical memory. Device is upward compatible with 8086/88 software.

Intel Corp
3065 Bowers Ave
Santa Clara, CA 95051
Phone (408) 987-8080

Status: Currently, the 80286 has the highest volume in the 8086 family. The total volume of 80286 chips was 4 million units for 1987. The 286's popularity has been based on the PC/AT, and the 286 will also share in the expected growth of the IBM PS/2 market. Its big sister, the 80386, will take over some of the 286's applications, but that will take time. Certainly the second sources will wish to give the 286 as long and thriving a life as they can, as so far Intel has shown no inclination to ever let them second source the 80386. Therefore it's logical to expect more enhanced 286s, such as the 16-MHz version from AMD and the CMOS version from Harris.

HARDWARE CHARACTERISTICS SOFTWARE



I—DATA-MANIPULATION INSTRUCTIONS
8- and 16-bit signed and unsigned arithmetic in binary or decimal, including multiply and divide
Logical operations on bytes, words, and blocks
II—DATA-MOVEMENT INSTRUCTIONS
Addressing modes include literal, relative (to register and to segment), register, base plus index, base relative indexed, and register indirect
Programmers can manipulate 16,383 segments in memory by means of memory-base descriptor tables and 4 segment registers. These segments can be between 1k and 64k bytes in length
III—PROGRAM-MANIPULATION INSTR
Has calls, jumps, and returns within the same protection level, across protection boundaries, and between tasks
Intrasegment calls and jumps use self-relative displacement for position-independent code
Intersegment calls and jumps use the memory-based descriptor tables to provide position-independence of code
Conditional jumps upon Boolean functions of flags within ±128 bytes of instruction
Iteration control of loops
String instructions, including repeat, for rapid iteration
IV—PROGRAM-STATUS-MANIP INSTR
8085 flags (carry, auxiliary carry, parity, zero, and sign) plus overflow, interrupt enable, direction (strings), trap (single-step), I/O privilege level, and nested task. Flag register is software accessible

Notes:
1. Has high-level-language support instructions
2. Virtual-address translation, memory management, and protection performed by CPU for faster execution
3. Trusted instructions can only be executed at highest protection levels.

Notes:
1. Support chips for 8C286: 82C284 clock, 82288 bus controller, 80287 floating-point numeric processor ($290 for 10 MHz, 100 qty), and 82258 advanced DMA coprocessor.
2. A new trend is for third-party VLSI houses to do high-integration chip sets to consolidate the devices being used around popular platforms, which for the 80286 would be the IBM PC/AT. Chip sets for the PC/AT are being offered by Chips and Technologies (San Jose, CA), Zymos (Sunnyvale, CA), VLSI Technology (Phoenix, AZ), and Hudson & Supinger (Santa Clara, CA); and also by Intel.

Specification summary: 16-bit CPU with 1G-byte virtual-address space per user, mapped onto 16M byte physical-address space. Bus cycles execute in 250 nsec at 8-MHz clock (200 nsec at 10 MHz), requiring 0.25 µsec for register-to-register moves at 8-MHz clock, with 8M-byte/sec bus bandwidth. HMOS on-implanted, silicon-gate circuitry in a large chip (335 - 339 mils, approximately 134,000 transistors). Requires 5V at 600 mA. Has two operating modes: Real-address mode emulates 8086, protected virtual-address mode native to 286. Housed in a 68-pin JEDEC Type A leadless chip carrier, PLCC, and PGA.

HARDWARE SUPPORT SOFTWARE

From Intel: ICE in-circuit emulator ($9995) supports 80286 at 8 and 10 MHz. It is hosted on IBM PC/AT/XT and Intellec Series III/IV development systems. ICE286 ($12,495) supports 80286 at 12.5 MHz. iPAT Performance Analysis Tool, consisting of a hardware base unit, an interface to ICE, and host software for the PC/AT/XT as well as Intellec Series III/IV, iPAT provides high-level access to target-system performance analysis and test-case code-coverage analysis for the 80286.
From others: Number of third parties support 286 on their universal development systems, for example, American Microsystems Corp (Beaverton, OR)

From Intel: Macroassembler (ASM 286) that includes systems builder, binder, mapper, and librarian. Compilers for C, Pascal, PL/M, Fortran, and Ada. For applications running in virtual 8086 mode, any of Intel's 8086 software tools can be used. Hosts include PC-DOS, VAX/VMS, and Intel development systems. Prices are $750 for DOS. Real-time operating systems Intel's iRMX 286 available.
From others: Other operating systems and compilers being developed by third-party software houses include MP/M-286 (Digital Research), Xenix-286 (Microsoft), Concurrent 286 (Mark Williams), Concurrent DOS (Digital Research), Unix System V (Digital Research), and of course OS/2 by Microsoft (Redmond, WA)

AVAILABILITY: 16, 20, and 25 MHz in production (at 4 locations). 80386SX in production, 80376 samples available now.

COST: In 100 qty, $299 for 16-MHz 80386, $484 for 20-MHz 80386. 80386SX in production, $219 (100). New 80376, $72 (1000).

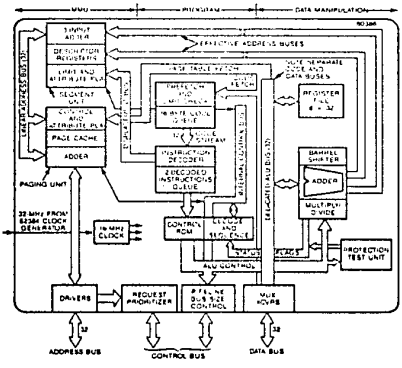
SECOND SOURCE: None announced or planned in immediate future.

Description: The 32-bit member of the 8086 family, suitable for both multiprocessing and multitasking. Contains a full 32-bit, largely uncharacterized register set (some competitors debate this) and an on-chip MMU containing selectable segmentation and paging support with a 32-entry TLB. Has slower emulation mode in which it is 100% binary compatible with the 8086 and 80286, allowing 8086 and 80286 and 80386 applications to run concurrently.

Intel Corp
3065 Bowers Ave
Santa Clara, CA 95051
Phone (408) 987-8080

Status: All things point to the 80386 remaining the dominant 32-bit µP, certainly for the next 5 years and probably until the year 2000. The 386 is not necessarily the best µP, but it's the sole µP carrying the IBM PC momentum into the 32-bit world. Not satisfied that it "owns" the MS-DOS and OS/2 world, Intel is now aggressively after the Unix world and now has samples of the 80376—a version of the 386 aimed at the embedded-controller world. Intel recently introduced the 80386SX, another version of the 386 that supplies a 16-bit data bus and a 24-bit address bus. The new chip is aimed the installed base of 8086 and 80286 chips.

HARDWARE CHARACTERISTICS SOFTWARE



Notes:
1. No on-chip cache, but 20-MHz 82385 cache controller ($125 for 20 MHz at 10k qty) for implementing 32k-byte external cache. Has posted-write and bus-watch features.
2. MMU on chip said to allow for memory management with no penalty in bus bandwidth (if off chip, supplier says, an extra cycle would be needed). Allows choices of segmentation or paging singly or in combination for multiuser protection and for virtual memory.
3. The 80386 has its own math coprocessor, the 80387 ($441 for 16 MHz, $583 for 20 MHz, 100 qty).
4. Along with the 80387 and 82385, the 80386 can use the 82380 32-bit peripheral combination chip that incorporates DMA and interrupt support and interval timers, etc.
5. The 80376 is compatible with the 386 programming model, but cannot run 8086 or real-mode programs. The chip has a 16-bit external bus.

I—DATA-MANIPULATION INSTRUCTIONS
Bit manipulation and bit-string manipulation (aided by 64-bit barrel shifter)
Conversion between bytes, words, and double words
Arithmetic, including 16-bit and 32-bit operands and 32-bit signed and unsigned multiply and divide
(80387 math coprocessor has full IEEE-754 instructions, including all transcendentals)

II—DATA-MOVEMENT INSTRUCTIONS
String moves and gang push and gang pop of all registers
Instructions to insert and extract bit strings (additional addressing modes for existing instructions allow more flexibility in assignment of registers)

III—PROGRAM-MANIPULATION INSTR
Repeat instructions based on flags
Enter and leave procedure instructions, conditional or unconditional branch to anywhere in 4G-byte memory space

IV—PROGRAM-STATUS-MANIP INSTR
Flag instructions mostly same as on 8088 (contains 4 debug registers, allowing breakpoints on data or code accesses, even when in ROM)

V—HLL AND OS INSTRUCTIONS
Instructions for checking array bounds
Segment assignment instructions
Load and store descriptor tables for protection (processor context switch via 1 instruction)

Notes:
1. Only those instructions beyond basic 8086 instructions described.
2. 80386 said to be object-code compatible with previous members of 8086 family and can run their operating systems. There is a "virtual 8086" mode in which 8086 (and 8088) code can be run within the protected 386 environment.

Specification summary: A more or less standard, "classical" 32-bit minicomputer architecture that has a basic register set similar to the previous 16-bit members of 8086 family so that it can directly run their machine code. It has added features that make it more general and suited to larger 32-bit environments: data-manipulation instructions that can be applied to almost any register, high-level-language-oriented instructions, operating-system-oriented instructions, and on-chip MMU. Performance can be 9k Dhrystones when operating at 16 MHz and with sufficiently fast (45-nsec) memory. Fabricated in 1.5-µm CMOS (supplier calls it CHMOS-III), it's expected to consume no more than 400 mA at 32-MHz external clock (16 MHz internal). Packaged in 132-lead ceramic PGA.

HARDWARE SUPPORT SOFTWARE

ICE-386 in-circuit emulator for 80386 hosted on Intel 286/310 running Xenix 286, allowing full 16-MHz operation in continuous or single-step mode. Can store more than 2000 frames of program-execution history. Has high-level-language symbolics. Can analyze time taken by code. Supports 80287 and 80387 coprocessors.
iSBC 386/20 single-board computer for Multibus I and iSBC 386/100 single-board computer for Multibus II. Besides the usual features expected of supplier's single-board computers, these incorporate 64k-byte caches to permit 16-MHz execution of 386. Starter kits are $9490, $7995, and $3860 in 100 qty.

From Intel: ASM-386 macroassembler ($600) and PMON-386 ($3500), DOS-hosted software debugger (DMON-386 ($2500) is unhosted version). Also iC-386 and PL/M-386 high-level languages, RLL-386 set of relocation linkage and library utilities ($600).
From others: Rapidly growing third-party support, of which most important are MS-DOS and OS/2 from Microsoft (Bellevue, WA). (There are variations in DOS such as Concurrent DOS by Digital Research (Monterey, CA).) Next is Unix V from AT&T (Morristown, NJ) and Zenix from Microsoft. Also real-time executives from Heady Systems (Palo Alto, CA), JMI Software (Spring House, PA), and others. In addition there are dual combinations of operating systems such as Unix-DOS from Phoenix (Norwood, MA), Locus (Santa Monica, CA), and Interactive Systems (Santa Monica, CA); CTOS-DOS from Convergent Technologies (San Jose, CA); and DOS-DOS from Intelligent Graphics (Santa Clara, CA).
Note: Some software depends on 386 mode.

6804/6805

8-BIT NMOS AND CMOS

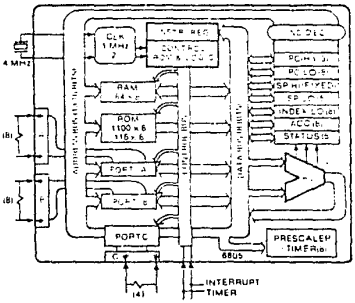
AVAILABILITY: Now for most models.
COST: $0.49 to $40. The $0.49 is 1M qty of 6804J1 (500k minimum order). CMOS parts remain more expensive than NMOS.
SECOND SOURCE: Hitachi, RCA, and Thomson; RCA for CMOS parts only.
CORE: Motorola and NCR have joint ASIC pact that will use CMOS 6805 as core along with NCR's similar 6502 μP core (SGS has S8 core, which has somewhat similar architecture to 6804.)

Motorola Microprocessor Products Group
6501 William Cannon Dr W
Austin, TX 78735
Phone (512) 440-2000

Description: Family of 1-chip μCs based loosely on 6800 architecture, but in some ways more like 6502 (especially 6805). Family offers various amounts of I/O, RAM, and ROM. Internal bus frequencies span dc to 2 MHz. Some parts contain on chip an A/D converter, EEPROM, serial I/O, and software security. The 6804s are meant for lowest-end applications. They use some serial data paths internally to reduce chip sizes.

Status: Supplier's steady commitment to this family over past 10 years has apparently paid off. It trails only the 8048/49 family (which has 20% of 8-bit-μC market) and the 50740 (which has 15%+). RCA is concentrating its efforts on the CMOS side of family and is bringing out its own enhancements. Motorola continues to expand the 6805 family; it plans to add at least five 6805-derivative parts in 1988.

HARDWARE CHARACTERISTICS SOFTWARE



Hardware Notes:
1. Diagram is for nonexpandable Model P2 in 28-pin package.
2. Comparison of 6805 with 6800: Stack pointer has only five working bits, so stack is only 32 bytes deep. Only one accumulator. Index register only 8 bits wide, so it can only span 256 memory locations. Program counter only 11 bits (adequate for P2's 2k-byte RAM + ROM memory space). Only one external interrupt.
3. Note additional 116 bytes in ROM for built-in self-check program that tests I/O, ROM pattern, RAM, and interrupts. Program is initiated by special pin.
4. RCA has emulator versions (68EM05/C4,D2) for prototyping and low-volume production. These are ROMless devices with all ROM access buses brought out for direct interfacing to industry-standard EPROMs. Come in 40-pin piggyback (for 2764).

I—DATA-MANIPULATION INSTRUCTIONS
All 6800 arithmetic, logic, and shift instructions. Bit set, clear, and branch on bit test (on tests can be made quite generally on all I/O and memory bits). 68HC05 has 8 × 8 multiply
II—DATA-MOVEMENT INSTRUCTIONS
Relative addressing allows data relocation
True indexing within the 256-location limits of 8-bit index
III—PROGRAM-MANIPULATION INSTR
16 conditional branches, including branch of interrupt line test
Mostly the same conditional branches of the 6800, but with more emphasis on branch upon bit and interrupt tests
Only 15 levels of subroutine nesting, including interrupt returns; 31 levels on certain new parts
Four sources of interrupts: external, timer, software, and reset. 68HC05 has vectored interrupts to service its serial communication and peripheral interfaces
IV—PROGRAM-STATUS-MANIP INSTR
Instructions for manipulating bits in status register (and in timer)
V—POWER-SAVING INSTRUCTIONS
CMOS 6804s and 6805s have Stop and Wait instructions and will safely reset themselves when the clock is applied again

Specification summary: Common-memory architecture, in which instructions, data, I/O, and timers all share the same memory space. This allows I/O to be bit rotated, bit manipulated, etc. Dedicated bit manipulation includes bit set/clear and branch on bit set/clear. A 4-MHz oscillator provides a 1-MHz internal cycle on most -05 versions. New 68HC05s have a 2.1-MHz internal bus speed. Included are parts with program security, on-chip EEROM, A/D converter, serial peripheral interface (SPI), and PLL frequency synthesizer. Family consists of NMOS and CMOS parts in 20-, 28-, and 40-pin DIPs (also chip carriers, etc). NMOS requires 5V supply, while CMOS will operate over 3 to 6V.

FAMILY		SPEED BUS (MHz)	INSTR	ON-CHIP ROM	MEM RAM	I/O PINS	TIMER	INTERRUPTS	POWER CONSUMPTION (mW)	PINS
6804	MIN	0	47	05k	32	16	—	3	001	20
	MAX	2	42	2k	128	20	YES	4	~400	28
68HC04 P4			42	3744	176	20 +2	YES	1	NA	28
6805	MIN	0	51	1k	64	16	—	3	001	28
	MAX	2	59	4k	176	32	YES	5	~700	40
68HC05	MIN	0	62	2k	96	32	YES	2	025	40
	MAX	2 1	62	77k	176	32	YES	2	025	40

NOTES:
1 CMOS VERSIONS CAN BE STOPPED (CLOCK = DC) IN THIS CONDITION POWER DROPS TO 10 μW
2 SOME 6805 DEVICES CAN BE EXPANDED EXTERNALLY TO 8k MEMORY; RCA 68HC63 BRINGS OUT 16 LINES FOR 64k ADDRESS SPACE
3 SPECIAL FUNCTIONS SUCH AS SERIAL COMMUNICATION PORTS & A/D CONVERTERS ARE AVAILABLE AMONG FAMILY MEMBERS

HARDWARE SUPPORT SOFTWARE

From Motorola: HDS-200 hardware/software development station; operates stand-alone or interfaced to virtually any host with an RS-232C line (including Motorola's Exor-trademarked stations). The less-costly 68705EVM (HMOS) or 1468705EVM (CMOS) boards, which have ports to a terminal and host computer, provide target-system emulation.
From RCA: Single-board evaluation kit that will interface to IBM PC via RS-232C.
From others: A number of third-party companies provide hardware emulators for the 6805 family: Sophia Systems (Santa Clara, CA), American Automation (Tustin, CA), etc. Most of these interface to IBM PCs.

From Motorola: Software can be obtained free for downloading over phone lines by calling (512) 440-3733.
From others: Many cross macroassemblers and linking loaders, some relocatable. RELMS (San Jose, CA) has cross support for Intel development systems. Avocet Systems Inc (Rockport, ME) has cross-assemblers for 6805 and 6804 that run on IBM PC, etc.

6800/6802 AND 6809/6309

8-BIT NMOS AND CMOS

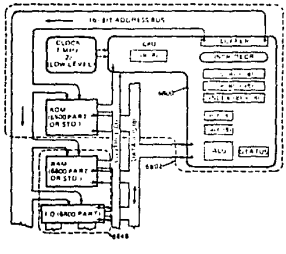
AVAILABILITY: Now.
COST: As with other mature µPs, costs have dropped (to a few dollars per µP), except when a part is at end of its life, in which case prices might rise again.
SECOND SOURCE: Hitachi, Fujitsu, and Thomson Semiconductors.

Motorola Microprocessor Products Group
6501 William Cannon Dr W
Austin, TX 78735
Phone (512) 440-2000

Description: The 8-bit 6800 CPU was the original part in the family named after it. That family has been broadened to include not only the 2-chip 6802/6846 and 6809 covered here but also the 1-chip 6801, the low-end 1-chip devices, and the 6804 and the 6805. Note, though, that new CPU members of family aren't precisely compatible with the original 6800, especially at the low and high ends. Even the 6809 here is only software compatible with the original 6800 at source-code level.

Status: Introduced in 1974, the 6800 has been the foundation of one of the longest lived and broadest µP families of all. Among its progeny must be included the 6809 covered here and the following Motorola µPs and µCs, which are described elsewhere in this directory: the 6804, 6805, 6801, and 68HC11. The 6800 itself is now past its prime and is not recommended for new designs; we retain it in the directory for reference. But the newer 6802 and 6809 continue to be shipped in volume. For new designs, Motorola steers designers either upwards to 16- and 32-bit 68000 family (68008 has 8-bit bus) or downwards to the 68HC11.

HARDWARE — CHARACTERISTICS — SOFTWARE



Notes:
1. Diagram shows 6800 and 6802. The 6809 has another 16-bit index and a second "user" stack pointer, which make the 6809 more powerful than the 6800; these additional resources give the 6809 many more instructions. On simple benchmarks, the 6809 is 270% faster than the equivalent-speed 6800, programs in 42% fewer instructions, and uses 33% less code.
2. Basic 6809 version has on-chip clock. A minimum system results with the following parts: 6809, 6810, and 6846. 6809E version has off-chip clock. An early valid-memory-address (VMA) signal on 6809E allows 3-MHz bus operation with a 2-MHz memory. External clock permits multiprocessing.
3. The memory-management unit (6829) allows the 6809 to run 32 concurrent protected tasks (per management unit) in 2M-byte address space.
4. Hitachi CMOS version (6309) has 2-, 2.5-, and 3-MHz bus timing, and the Sync and CWAI instructions allow a low-power sleep mode.

PART	DESCRIPTION	CLOCK SPEED (MHz)	ROM (KB)	RAM (KB)	AVAIL	COST (10K QTY)
6800	CPU; NEEDS 2-Ø CLOCK	1-2	—	—	NOW	$4.95
6802	CPU, CLOCK & RAM	1-2 (4 MHz EST)	—	128	NOW	$4.95
6809	CPU	2	—	—	NOW	$9.95
6309	CPU CMOS	3	—	—	NOW	$9.90

I—DATA-MANIPULATION INSTRUCTIONS
Arithmetic and logic
Instructions to take advantage of two accumulators
6809 has unsigned 8 × 8 multiply with 16-bit product
II—DATA-MOVEMENT INSTRUCTIONS
Can reach the first 256 locations of memory with short instructions
6809 can use four index registers for merging three source blocks into one destination block
Can autoincrement and autodecrement by one or two directly and indirectly. (Page zero can be software relocated during program execution, effectively increasing its size)
Indexing uses the "true indexing" relationship between base and offset (0, 5, 8, 16 bits) rather than the 6800 relationship
Can utilize the user stack for Polish-notation operations or interpretive languages
III—PROGRAM-MANIPULATION INSTR
Has PDP-11-type branches and conditional branches. Unlimited subroutine nesting via stack pointer addressing LIFO stacks in RAM
Does not have vectored interrupt, but can achieve function with software (or with 6828 priority interrupt controller)
6809 has extensive relative addressing with wide reach, which allows creation of position-independent code and opens door to use of off-the-shelf, mass-produced standard firmware in ROMs
IV—PROGRAM-STATUS-MANIP INSTR
6809 has instructions for manipulating the status register (condition-code register). It may be transferred or exchanged with any 8-bit register, or pushed or pulled on either stack; any number of flag bits may be set or cleared in one instruction
V—POWER-SAVING INSTRUCTIONS
6309 has SYNC and CWAI to put CMOS CPU in sleep mode. Sync instruction stops µP until it gets go-ahead signal from interrupt line

Specification summary for 6800: Common-memory architecture with 16-bit (64k-byte) memory space for instructions, data, and I/O; all data 8 bits wide. Instruction set patterned after the PDP-11 mini as closely as possible in shorter word machine with limited CPU registers. Execution times from 2 to 5 µsec. NMOS circuitry requires one 5V supply, 500 mW; housed in 40-pin DIP. Versions with −55 to +125°C range also available.
Specification summary for 6809: An 8-bit machine with extensive 16-bit addressing capability. Has two 16-bit index registers and a 16-bit user stack pointer that can also be software-specified as a third index register. Upwardly compatible with 6800, but only at source-code level. Bus operates at 2 MHz, so basic speed is similar to that of 6500, but greater efficiency of 16-bit addressing increases throughput. Instruction set has 59 mnemonics and seven addressing selections for a total of 1464 instruction-addressing options. Instructions vary in length from 1 to 5 bytes, with register-inherent operations executing in 1 µsec at 2-MHz bus speed (320-nsec memory access). Longest instruction takes 20 cycles. The 6800 direct or page-zero register is retained but can be software relocated anywhere in memory via programmable register. The chip requires one 5V supply. Two versions, each in 40-pin DIP.

HARDWARE — SUPPORT — SOFTWARE

From Motorola: Emulators range from low-cost (hundreds of dollars) boards to HDS-300 system (about $5000) plus personality modules ($500).
Support systems and OEM boards available from Motorola Semiconductor Div, 5005 E McDowell Rd, Phoenix, AZ 85008. Phone (602) 244-6900 or (602) 438-3500.
From others: Tektronix and Hewlett-Packard development systems support the 6800. Micro Industries (Westerville, OH) says it has acquired an exclusive license to Motorola "Micromodule" 8-bit boards.

From Motorola: Software can be obtained free for downloading over phone lines by calling (512) 440-3733. The basic assemblers and other tools are for IBM PC.
Two versions of Basic are available for the 6809: Basic-M and Basic09. The latter is designed to be fast and to permit structured programming. A Pascal compiler diskette is available.

68000 FAMILY

8/32-BIT, 16/32-BIT, 32/32-BIT NMOS AND CMOS

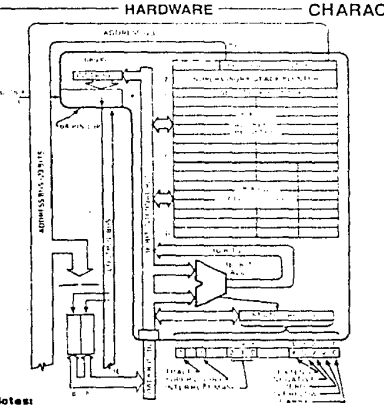
AVAILABILITY: Now for production quantities of all models to 25-MHz 68020. Samples of 68030.
COST: In 100 qty, from $10 for low-end 68008 and 68000 to $135 for 12.5-MHz 68020 and $530 for 25-MHz 68020. 68881 math coprocessor is $107 for 12.5 MHz and $347 for 25 MHz. Production pricing for 68030 not available.
SECOND SOURCE: Rockwell, Hitachi, Mostek, Signetics/Philips, and Thompson SGS, all licensed with mask interchange for 16-bit parts. Thompson was to be second source for 32-bit 68020, but Motorola says it plans to keep 68020 and 68030 to itself for time being.

Motorola Integrated Circuits Div
3501 Ed Bluestein Blvd
Austin, TX 78721
Phone (512) 928-6000

Description: Family based on a modern minicomputer architecture using a basic group of 16 fairly general, 32-bit registers. Family members have various addresses and data-bus widths and different ALU widths. The bottom of the line, the 68008, has a narrow 8-bit data bus. The middle member, the 68000, has a mid-sized 16-bit data bus and ALU and 24-bit addressing. Current top of the line, the 68030, is full 32 bits throughout with instruction and data caches and MMU on board.

Status: Part of the success of the 68000 family is due to the success of the Apple Macintosh II, and much of the rest is due to 68020's popularity among Unix-based workstations. The new 68030, which is now available, is similar to the 68020 but with an extra cache for data in addition to the 68020 instruction cache and with the 68851 MMU on board. There is considerable speculation about the competition the 68030 will receive from the new RISC µPs, including Motorola's own 88000 RISC chips.

HARDWARE CHARACTERISTICS



Notes:
1. Diagram favors the basic 68000, which although it has 32-bit-wide registers, has 16-bit-wide ALU and data buses and only 23-bit-wide address bus. It comes in 64-pin DIP and 68-pin grid array.
2. Bottom-of-the-line 68008 has only 8-bit data bus and 20- or 22-bit address bus. It comes in 48-pin DIP.
3. Upper-range 68010 and 68012 are similar to 68000 but support virtual memory. 68010 has 24-bit address bus and comes in a 64-pin DIP or 68-pin grid array. 68012 has full 32 bits of address and comes in 84-pin grid array.
4. Top-of-the-line 68020 and 68030 are full 32 bits throughout, including ALU and address and data paths. Both have instruction caches and the 68030 also has a data cache and an MMU.
5. Two important support chips, not shown, are the 68881 floating-point coprocessor and the 68851 MMU. Both are in CMOS.
6. The 68070 by Philips/Signetics includes various support functions on chip.

SOFTWARE

I—DATA-MANIPULATION INSTRUCTIONS
Arithmetic, including multiply and divide (signed and unsigned)
Logicals and rotates and shifts
Can handle bits, BCD nibbles, bytes, short (16 bits), and long (32 bits) words
(Floating-point coprocessors 68881/2 available)
II—DATA-MOVEMENT INSTRUCTIONS
Five basic address modes are register direct, register indirect, immediate, absolute, and program-counter relative. To these modes can be added postincrementing, predecrementing, offsetting, and indexing.
Can use eight 32-bit address registers as indexers or stack pointers. The eight 32-bit data registers can also serve as indexers.
III—PROGRAM-MANIPULATION INSTR
Branch and jump to subroutine. Branch conditionally.
Link and unlink instructions, invoking one address register as frame pointer (used to establish temporary local environments in structured programming).
Seven levels of priority interrupts, including nonmaskable, with 256 possible interrupt vectors.
IV—PROGRAM-STATUS-MANIP INSTR
16-bit status register is software accessible.
Sophisticated trap operations help user debug programs.
Trace mode.
V—SYSTEM-CONTROL INSTR
Privileged instructions for operating systems and multiprocessor communication.

Specification summary: 68020: full 32-bit CPU version of the 68000 family that's object-code compatible with all members. Has 16 32-bit general-purpose data and address registers, 32-bit ALU with barrel shifter, and 32-bit data bus. Also has full 32-bit address bus that can reach 4G bytes of direct linear external memory. Supports instruction-continuation-type virtual memory. Has 256-byte instruction cache on chip and 3-stage pipelining. At 25-MHz maximum clock, executes 5 MIPS. For tight inner loops with so few instructions that they can be contained in cache, and when data can be contained in registers, will operate at burst modes to 12 MIPS. With 68881, it can run at 1.25M Whetstones. Has 18 addressing modes and instructions to support structured high-level languages and sophisticated operating systems. Fabricated in 1.5-µm CMOS with 1.5W power dissipation and packaged in 114-pin grid array. 68030 is similar to 68020 but also has data cache and incorporates 68851 MMU. It will run at 20 to 30 MHz and have 2× 68020 performance at systems level. It is fabricated in 1.2-µm CMOS (with planned shrinkage to 1.1 µm). Packaged in 128-pin grid array.

HARDWARE SUPPORT

HDS-300 hardware/software development station ($15k to $20k) provides real-time emulation of 68000 family µPs with bus-state-analyzer support and source-level debugging. MEX68KECB educational computer board is based on 68000. VM04 is a 68020-based 32-bit VERSAmodule interconnected within a target system using the 32-bit, asynchronous, Versabus interconnect standard. VME130 is a 68020-based, 32-bit VME bus module using Eurocard mechanical format.
From third parties: Family is widely supported by makers of universal µP development systems such as Applied Microsystems (Redmond, WA). Also, the VME Bus system architecture is used in a broad range of applications with more than 150 independent suppliers of compatible products.

SOFTWARE

VersaDOS real-time operating system, system V/68 operating system, CP/M-68K operating system, concurrent DOS-68K operating system, and VRTX real-time operating system ($6775 from Hunter Systems). Unix support from Motorola includes direct ports of Unix V, AT&T, X assembler for Exormacs and VME/10, X-C compiler VME/10 and Exormacs for VAX/780 are available.
From third parties: Supplier has catalog listing the considerable outside support for family. New type of support is software to allow 68000 to run MS-DOS (8086) programs: by emulation from Phoenix (Norwood, MA) and by Insignia (London, UK, but with offices in San Francisco); and by binary translation from Hunter Systems (Palo Alto, CA). The latter's package, XDOS ($600), uses a binary compilation approach to generate disks that will run at competitive speeds on 68000 µPs.

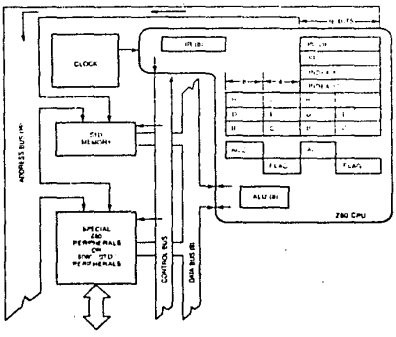
AVAILABILITY: Now for 6- and 8-MHz NMOS and CMOS versions.
COST: Because of the many aggressive second sources for this most-widely-used part, NMOS prices have dropped to $1 (6 MHz) and less (4 MHz). CMOS volume prices have dropped to $1.70 (6 MHz) and $1.30 (4 MHz), both in high volume.
SECOND SOURCE: Sharp, SGS, NEC, Toshiba, and Thomson-Mostek. Toshiba, Sharp, and SGS as well as Zilog have CMOS versions. Additional sources mentioned by Zilog are Gold Star, VLSI Technology, and Rohm.
CORE: Both Zilog and Hitachi are considering the Z80 μP as an ASIC core in their enhanced versions, the 64180 and the Z280.

Description: Superset of widely used 8080/85; adds hardware and software features. Not pin-for-pin compatible with 8080 or 8085, but can use 8080 software and peripherals—though to do so would not take full advantage of Z80, and its peripherals and might require additional logic for interfacing.

Zilog Inc
210 Hacienda Ave
Campbell, CA 95008
Phone (408) 370-8000

Status: By far the most successful 8-bit μP. The Z80 is still being used in new designs but may be superseded by the new enhanced versions. Of these, the Hitachi 64180 seems to be the most popular, but the Zilog Z280 represents the greatest Z80 enhancement. Whatever happens, one thing is certain: The Z80's momentum will probably last for the rest of this century, especially in ASIC core form.

HARDWARE CHARACTERISTICS SOFTWARE



I—DATA-MANIPULATION INSTRUCTIONS
8-bit arithmetic and logicals
16-bit arithmetic BCD add and subtract
Nine types of rotate and shift directly on any register or memory location
Can set, reset, or test bit in any register or memory location
II—DATA-MOVEMENT INSTRUCTIONS
8- or 16-bit register or memory loads
Two index registers allow indexed addressing
Extensive memory-block move/search commands
III—PROGRAM-MANIPULATION INSTR
Uses 16-bit stack pointer with LIFO stack with RAM
Relative-jump capability. Interrupt capability with three types of selectable response
IV—PROGRAM-STATUS-MANIP INSTR
Seven flag bits, including arithmetic and overflow, can be stored and tested

Notes:
1. Support chips include peripheral interface (PIO), timer (CTC), serial communications (SIO), and DMA. All provide daisy-chained vectored interrupt for CPU and are being converted to CMOS.
2. Several enhancements of Z80 exist or are imminent. All are in CMOS. The first is the Hitachi 64180, to which many Z80 designers are converting. The second is the supplier's Z280, which boosts the Z80 into minicomputer performance. In addition, the NEC 78XX single-chip device is similar. Most are covered elsewhere in this directory.

Specification summary: Upwardly compatible with 8080A software, but adds 50 instructions, some of which are advance block-move and block-search macros. Instructions executed in 0.5 to 1.8 μsec (1.5 μsec avg) for 8-MHz Z80 and 1.0 to 5.5 μsec (2 μsec avg) for 4-MHz Z80A. 6- and 8-MHz versions are also available. User can switch between two identical banks of CPU registers for fast response to interrupts. NMOS circuitry requires single-phase clock and one 5V supply at 60 mA for Z80; 90 mA for Z80A. TTL-compatible I/O and built-in automatic-refresh signals for dynamic RAMs. MIL-temperature parts available. CMOS version consumes only 15 mA at 4 MHz and less than 10 μA when in power-down (clock-stopped) mode. Housed in 40-pin DIP. CMOS versions available also in flat pack and PLCC.

HARDWARE SUPPORT SOFTWARE

From Zilog: "Z-Scan" emulator boxes that can be used alone or with host computers. Z-Scan-80 that will provide emulation for the Z80H ($6695).
From SGS: UX-8/22 development system based on CP/M and two 8-in. floppy disks. Package for full-speed in-circuit emulation.
From others: Some of the many third parties that supply Z80 hardware support are Applied Micro, Boston Systems, Emulogic, Hewlett-Packard, Huntsville Microsystems, Nicolet, Orion, Sophia Systems, Tektronix, and Zax. Contact Zilog for addresses.

From Zilog: Software for the various development systems. Macro-assembler with relocatable assembler, linking loader, file-maintenance programs, and resident Basic, Cobol, C, Fortran, and PLZ (Zilog-created language that comes in "lower" level that mixes assembly-and system-language statements with a "higher" C language). Z80 has cross-software package (assembler, etc) that runs on DEC VAX or Zilog S8000 under Unix.
From SGS: Software package for UX-8/22, including debugger, disassembler, and tracer.
From others: A lot of software of all sorts, including the popular CP/M operating system (Digital Research) and the MS/X operating system (from Microsoft), which is popular in Japan. Contact Zilog for names and addresses of several dozen others.

Z8000/Z80000

16/32-BIT NMOS AND CMOS

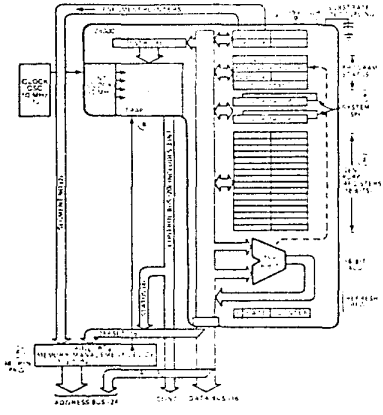
AVAILABILITY: Now for NMOS Z8000 at 4, 6, 10, and 12 MHz. Now for NMOS Z80000 at 8 and 10 MHz. CMOS versions for Z8000 in '89 and for Z80000 in '90.
COST: $13.80 for Z8000 in 10k qty
SECOND SOURCE: AMD (licensed), SGS (Italy and Anzona), and Sharp for Z8000. NEC for Z80000, by mask exchange
CORE: Zilog is incorporating both Z8000 and Z80000 as cores in its "in-house" ASIC library, planning to use Zbus for their systems on silicon. Says that 160 × 160 mil Z8000 core is small enough to leave room for other functions on practical 400 × 400-mil ASIC chip.

Zilog Inc
210 Hacienda Ave
Campbell, CA 95008
Phone (408) 370-8000

Description: One of first μPs to have architectural features of a modern minicomputer. Original 16-bit Z8000 comes in 40-pin package for addressing 64k-byte memory or in 48-pin package for addressing 8M-byte memory. Said by many industry observers to be architecturally more powerful than 8086 but less powerful than 68000. Supplier says military has found it to be highest performance 16-bit μP, offering best CPU speed, interrupt handling, and character-string search. New 32-bit version, Z80000, is superminicomputer-like enhancement that remains object-code compatible with the Z8000. Has cache for data and instructions and an MMU.

Status: The Z8000 has, according to Zilog, found most acceptance in real-time control applications, particularly military. Supplier says it has been shipping samples of the much-delayed Z80000 for 6 months and some customers have found it will run at over 16 MHz in their systems. Zilog will be pushing the Z80320 "32-bits-for-32-bucks" derivative of the Z80000. Supplier has again slipped on its schedule for CMOS versions, and now says it will be '89 for Z8000 and '90 for Z80000.

HARDWARE — CHARACTERISTICS — SOFTWARE



Notes:
Supplier has companion peripherals suitable for both processors:
For Z8000, a range of DMA, FIFO, data ciphering (NBS), communications and counter/timer parts.
For Z80000, two 32-bit parts: a Z32104 CMOS DMA controller, 32-bit address and data buses with 8-bit peripheral bus; and a Z32106 CMOS floating-point coprocessor that implements IEEE P754 format.

I—DATA-MANIPULATION INSTRUCTIONS
Arithmetic, including add, subtract, decimal adjust, increment, decrement, multiply (signed), divide (signed)
Logicals, including AND, OR, exclusive OR, compare, test, complement, rotate, and shift (by n)
Operations can be on bit, BCD nibble, byte, 16-bit word, or 32-bit double word, and can use any of the 16 general-purpose registers as accumulator
The Z32106 floating-point processor will do IEEE-754 operations
II—DATA-MOVEMENT INSTRUCTIONS
Eight addressing modes using general-purpose registers as indexers and stack pointers
Comprehensive set of block-transfer and string-manipulation macroequivalents, including many dedicated to I/O space
III—PROGRAM-MANIPULATION INSTR
Call and call relative (± 4096 bytes)
System call using special system stack pointer
Jump conditionals
IV—PROGRAM-STATUS-MANIP INSTR
Set and reset flags, complement flags. Set-multiple-interrupt modes
Tests for the micro in and micro out lines for multiple-microprocessor configurations
V—SYSTEM-CONTROL INSTRUCTIONS
The 80000 has privileged instruction for exclusive use by an operating system

Specification summary: Common-memory architecture with optional separate I/O space and separate "systems" stack. Z8000 is 16-bit μP that has directly addressable memory space of 8M bytes (8001, 8003) using segment pointers, expandable to 48M bytes using the six available memory spaces and an MMU. Executes 110 basic instructions with 410 combinations at speeds ranging from 0.30 μsec through 1 or 2 μsec to 7 μsec for 16-bit multiply, all at 10-MHz system clock (4 and 6 MHz also available). Eight large-computer-style addressing modes. NMOS, requiring one 5V supply (plus substrate-decoupling capacitor) in either 40-or 48-pin package. Z80000 is a 32-bit upward-compatible version of Z8000 and can run same software. 6-stage pipelining of instruction fetch/execute cycle and 256-byte on-chip associative cache for instructions and data for improved performance (and use of 100- to 120-nsec memories). Also on-chip MMU for virtual memory with address bus a full 32 bits for 4G-byte memory space. At 25-MHz clock has 12.5-MHz (80-nsec) instruction cycles that give 12.5-MIPS burst rate (when doing loops out of cache) and 5 MIPS continuously (4 MIPS with MMU virtual-memory translation). 16 × 16 multiply in 1.2 μsec and 32 × 32 in 1.9 μsec. 2-μm NMOS dissipating 3 to 4W with 1½-μm CMOS promised for '88 (Z8000) and '89 (Z80000). Initial samples have been packaged in ceramic PGAs but lower-cost Z80320 will have muxed address and data buses and be in 68-pin PLCC.

HARDWARE — SUPPORT — SOFTWARE

From Zilog: Z-Scan 8000 in-circuit emulator ($5500). 500-pg Z8000 technical manual
From others: Applied Micro, Boston Systems, Hewlett-Packard, Kontron, Orion, Single Board Sol, Sweet Micro System, and Tektronix. Contact supplier for addresses.

From Zilog: Real-time application software (PC-based). C and PLZ/SYS compilers.
From others: Real-time executive from Ready Systems (Palo Alto, CA), VRTX-8002 ($5775), which is suited to embedded applications, and an Ada compiler ($795) from Meridian Software Systems (Laguna Hills, CA). Contact supplier for names and addresses of others.

Apéndice D

Hojas Técnicas

PRELIMINARY

iAPX 88/10
8-BIT HMOS MICROPROCESSOR
8088/8088-2

- 8-Bit Data Bus Interface
- 16-Bit Internal Architecture
- Direct Addressing Capability to 1 Mbyte of Memory
- Direct Software Compatibility with iAPX 86/10 (8086 CPU)
- 14-Word by 16-Bit Register Set with Symmetrical Operations
- 24 Operand Addressing Modes
- Byte, Word, and Block Operations
- 8-Bit and 16-Bit Signed and Unsigned Arithmetic in Binary or Decimal, Including Multiply and Divide
- Compatible with 8155-2, 8755A-2 and 8185-2 Multiplexed Peripherals
- Two Clock Rates:
  5 MHz for 8088
  8 MHz for 8088-2
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel® iAPX 88/10 is a new generation, high performance microprocessor implemented in N-channel, depletion load, silicon gate technology (HMOS), and packaged in a 40-pin CerDIP package. The processor has attributes of both 8- and 16-bit microprocessors. It is directly compatible with iAPX 86/10 software and 8080/8085 hardware and peripherals.

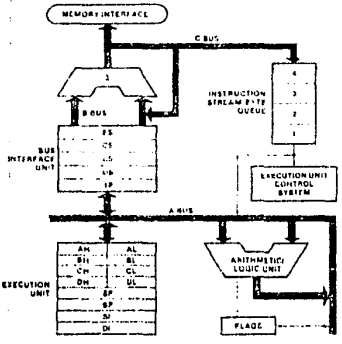


Figure 1. iAPX 88/10 CPU Functional Block Diagram

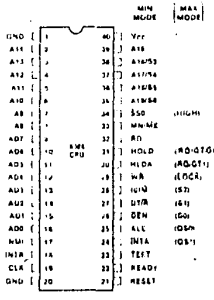


Figure 2. iAPX 88/10 Pin Configuration

FUNCTIONAL DESCRIPTION

Memory Organization

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries. (See Figure 3.)

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU will automatically execute two fetch or write cycles for 16-bit operands.

Certain locations in memory are reserved for specific CPU operations. (See Figure 4.) Locations from addresses FFFF0H through FFFFFH are reserved for operations including a jump to the initial system initialization routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be located. Locations 00000H through 003FFH are reserved for interrupt operations. Four-byte pointers consisting of a 16 bit segment address and a 16-bit offset address direct program flow to one of the 256 possible interrupt service routines. The pointer elements are assumed to have been stored at their respective places in reserved memory prior to the occurrence of interrupts.

Minimum and Maximum Modes

The requirements for supporting minimum and maximum 8088 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8088 is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes, dependent on the condition of the strap pin. When the MN/$\overline{MX}$ pin is strapped to GND, the 8088 defines pins 24 through 31 and 34 in maximum mode. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 8088 generates bus control signals itself on pins 24 through 31 and 34.

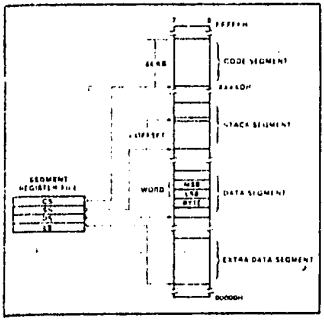


Figure 3. Memory Organization

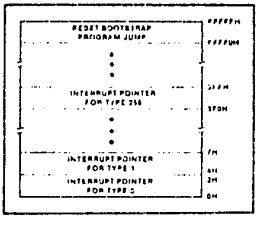


Figure 4. Reserved Memory Locations

Memory Reference Need	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic with all instruction prefetch.
Stack	STACK (SS)	All stack pushes and pops. Memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references when: relative to stack, destination of string operation, or explicitly overridden
External (Global) Data	EXTRA (ES)	Destination of string operations. Explicitly selected using a segment override.

The minimum mode 8088 can be used with either a multiplexed or demultiplexed bus. The multiplexed bus configuration is compatible with the MCS 85™ multiplexed bus peripherals (8155, 8156, 8355, 8755A, and 8185). This configuration (See Figure 5) provides the user with a minimum chip count system. This architecture provides true 8088 processing power in a highly integrated form.

The demultiplexed mode requires one latch (for 64K addressability) or two latches (for a full megabyte of addressing). A third latch can be used for buffering if the address bus loading requires it. An 8286 or 8287 transceiver can also be used if data bus buffering is required. (See Figure 6.) The 8088 provides $\overline{DEN}$ and DT/$\overline{R}$ to control the transceiver, and ALE to latch the addresses. This configuration of the minimum mode provides the standard demultiplexed bus structure with heavy bus buffering and relaxed bus timing requirements.

The maximum mode employs the 8288 bus controller. (See Figure 7.) The 8288 decodes status lines $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$, and provides the system with all bus control signals. Moving the bus control to the 8288 provides better source and sink current capability to the control lines, and frees the 8088 pins for extended large system features. Hardware lock, queue status, and two request/grant interfaces are provided by the 8088 in maximum mode. These features allow co-processors in local bus and remote bus configurations.

Bus Operation

The 8088 address/data bus is broken into three parts — the lower eight address/data bits (AD0-AD7), the middle eight address bits (A8-A15), and the upper four address bits (A16-A19). The address/data bits and the highest four address bits are time multiplexed. This technique provides the most efficient use of pins on the processor, permitting the use of a standard 40 lead package. The middle eight address bits are not multiplexed, i.e. they remain valid throughout each bus cycle. In addition, the bus can be demultiplexed at the processor with a single address latch if a standard, non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T1, T2, T3, and T4. (See Figure 8). The address is emitted from the processor during T1 and data transfer occurs on the bus during T3 and T4. T2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device,

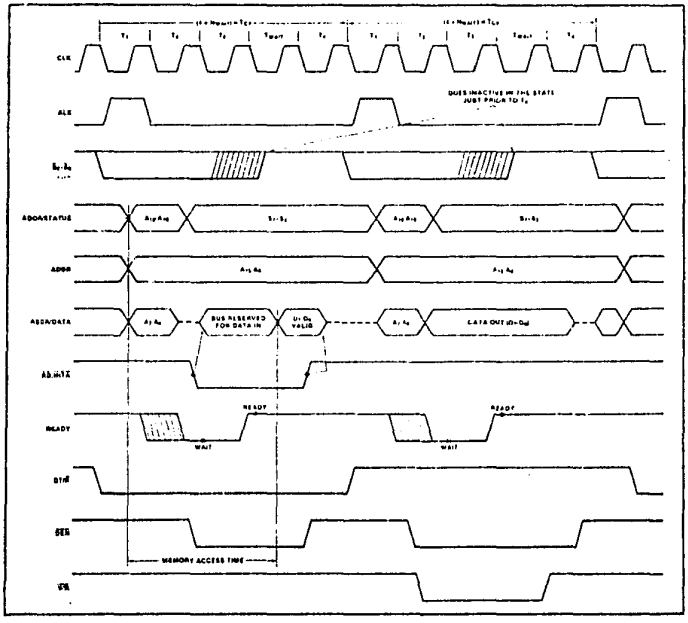


Figure 8. Basic System Timing

"wait" states (Tw) are inserted between T3 and T4. Each inserted "wait" state is of the same duration as a CLK cycle. Periods can occur between 8088 driven bus cycles. These are referred to as "idle" states (Ti), or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T1 of any bus cycle, the ALE (address latch enable) signal is emitted (by either the processor or the 8288 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ are used by the bus controller in maximum mode, to identify the type of bus transaction according to the following table:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CHARACTERISTICS
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S3 through S6 are multiplexed with high order address bits and are therefore valid during T2 through T4. S3 and S4 indicate which segment register was used for this bus cycle in forming the address according to the following table:

S_4	S_3	CHARACTERISTICS
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S5 is a reflection of the PSW interrupt enable bit. S6 is always equal to 0.

I/O Addressing

In the 8086, I/O operations can address up to a maximum of 64K I/O registers. The I/O address appears in the same format as the memory address on bus lines A15-A0. The address lines A19-A16 are zero in I/O operations. The variable I/O instructions, which use register DX as a pointer, have full address capability, while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space. I/O ports are addressed in the same manner as memory locations.

Designers familiar with the 8085 or upgrading an 8085 design should note that the 8085 addresses I/O with an 8-bit address on both halves of the 16-bit address bus. The 8088 uses a full 16-bit address on its lower 16 address lines.

EXTERNAL INTERFACE

Processor Reset and Initialization

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8088 RESET is required to be HIGH for greater than four clock cycles. The 8088 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 clock cycles. After this interval the 8088 operates normally, beginning with the instruction in absolute location FFFF0H. (See Figure 4.) The RESET input is internally synchronized to the processor clock. At initialization, the HIGH to LOW transition of RESET must occur no sooner than 50 μs after power up, to allow complete initialization of the 8088.

If INTR is asserted sooner than nine clock cycles after the end of RESET, the processor may execute one instruction before responding to the interrupt.

All 3-state outputs float to 3-state OFF during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF.

Interrupt Operations

Interrupt operations fall into two classes: software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the instruction set description in the *iAPX 88* book or the *iAPX 86,88 User's Manual*. Hardware interrupts can be classified as nonmaskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256 element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 4), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type." An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to vector through the appropriate element to the new interrupt service program location.

Non-Maskable Interrupt (NMI)

The processor provides a single non-maskable interrupt (NMI) pin which has higher priority than the maskable interrupt request (INTR) pin. A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW to HIGH transition. The activation of this pin causes a type 2 interrupt.

NMI is required to have a duration in the HIGH state of greater than two clock cycles, but is not required to be synchronized to the clock. Any higher going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves (2 bytes in the case of word moves) of a block type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur

"wait" states (Tw) are inserted between T3 and T4. Each inserted "wait" state is of the same duration as a CLK cycle. Periods can occur between 8088 driven bus cycles. These are referred to as "idle" states (Ti), or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T1 of any bus cycle, the ALE (address latch enable) signal is emitted (by either the processor or the 8288 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ are used by the bus controller in maximum mode, to identify the type of bus transaction according to the following table:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CHARACTERISTICS
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S3 through S6 are multiplexed with high order address bits and are therefore valid during T2 through T4. S3 and S4 indicate which segment register was used for this bus cycle in forming the address according to the following table:

S_4	S_3	CHARACTERISTICS
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S5 is a reflection of the PSW interrupt enable bit. S6 is always equal to 0.

I/O Addressing

In the 8088, I/O operations can address up to a maximum of 64K I/O registers. The I/O address appears in the same format as the memory address on bus lines A15-A0. The address lines A19-A16 are zero in I/O operations. The variable I/O instructions, which use register DX as a pointer, have full address capability, while the direct I/O instructions directly address one or two of the 256 I/O-byte locations in page 0 of the I/O address space. I/O ports are addressed in the same manner as memory locations.

Designers familiar with the 8085 or upgrading an 8085 design should note that the 8085 addresses I/O with an 8-bit address on both halves of the 16-bit address bus. The 8088 uses a full 16-bit address on its lower 16 address lines.

EXTERNAL INTERFACE

Processor Reset and Initialization

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8088 RESET is required to be HIGH for greater than four clock cycles. The 8088 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 clock cycles. After this interval the 8088 operates normally, beginning with the instruction in absolute location FFFF0H. (See Figure 4.) The RESET input is internally synchronized to the processor clock. At initialization, the HIGH to LOW transition of RESET must occur no sooner than 50 μs after power up, to allow complete initialization of the 8088.

If INTR is asserted sooner than nine clock cycles after the end of RESET, the processor may execute one instruction before responding to the interrupt.

All 3-state outputs float to 3-state OFF during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF.

Interrupt Operations

Interrupt operations fall into two classes: software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the instruction set description in the IAPX 88 book or the IAPX 88/88 User's Manual. Hardware interrupts can be classified as nonmaskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256 element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 4), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type." An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to vector through the appropriate element to the new interrupt service program location.

Non-Maskable Interrupt (NMI)

The processor provides a single non-maskable interrupt (NMI) pin which has higher priority than the maskable interrupt request (INTR) pin. A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW to HIGH transition. The activation of this pin causes a type 2 interrupt.

NMI is required to have a duration in the HIGH state of greater than two clock cycles, but is not required to be synchronized to the clock. Any higher going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves (2 bytes in the case of word moves) of a block type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur

before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

Maskable Interrupt (INTR)

The 8088 provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable (IF) flag bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block type instruction. During interrupt response sequence, further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt, or single step), although the FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored, the enable bit will be zero unless specifically set by an instruction.

During the response sequence (See Figure 9), the processor executes two successive (back to back) interrupt acknowledge cycles. The 8088 emits the LOCK signal (maximum mode only) from T2 of the first bus cycle until T2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle, a byte is fetched from the external interrupt system (e.g., 8259A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The interrupt return instruction includes a flags pop which returns the status of the original interrupt enable bit when it restores the flags.

HALT

When a software HALT instruction is executed, the processor indicates that it is entering the HALT state in one of two ways, depending upon which mode is strapped. In minimum mode, the processor issues ALE delayed by one clock cycle, to allow the system to latch the halt status. Halt status is available on IO/$\overline{M}$, DT/$\overline{R}$, and $\overline{SSO}$. In maximum mode, the processor issues appropriate HALT status on $\overline{S2}$, $\overline{S1}$, and $\overline{S0}$, and the 8288 bus controller issues one ALE. The 8088 will not leave the HALT state when a local bus hold is entered while in HALT. In this case, the processor reissues the HALT indicator at the end of the local bus hold. An interrupt request or RESET will force the 8088 out of the HALT state.

Read/Modify/Write (Semaphore) Operations via LOCK

The LOCK status information is provided by the processor when consecutive bus cycles are required during the execution of an instruction. This allows the processor to perform read/modify/write operations on memory (via the "exchange register with memory" instruction), without another system bus master receiving intervening memory cycles. This is useful in multiprocessor system configurations to accomplish "test and set lock" operations. The $\overline{LOCK}$ signal is activated (LOW) in the clock cycle following decoding of the LOCK prefix instruction. It is deactivated at the end of the last bus cycle of the instruction following the LOCK prefix. While $\overline{LOCK}$ is active, a request on a $\overline{RQ}$/$\overline{GT}$ pin will be recorded, and then honored at the end of the LOCK.

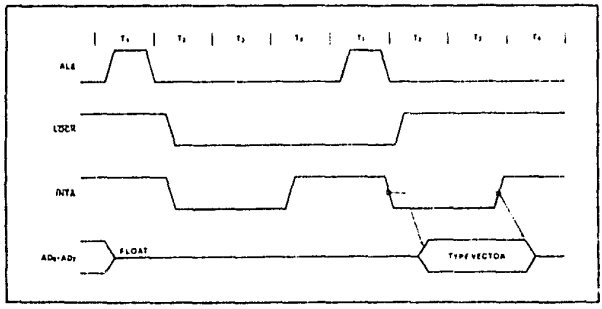


Figure 9. Interrupt Acknowledge Sequence

External Synchronization via $\overline{TEST}$

As an alternative to interrupts, the 8088 provides a single software-testable input pin ($\overline{TEST}$). This input is utilized by executing a WAIT instruction. The single WAIT instruction is repeatedly executed until the $\overline{TEST}$ input goes active (LOW). The execution of WAIT does not consume bus cycles once the queue is full.

If a local bus request occurs during WAIT execution, the 8088 3-states all output drivers. If interrupts are enabled, the 8088 will recognize interrupts and process them. The WAIT instruction is then refetched, and reexecuted.

Basic System Timing

In minimum mode the MN/$\overline{MX}$ pin is strapped to V_{CC} and the processor emits bus control signals compatible with the 8085 bus structure. In maximum mode, the MN/$\overline{MX}$ pin is strapped to GND and the processor emits coded status information which the 8288 bus controller uses to generate MULTIBUS compatible bus control signals.

System Timing — Minimum System

(See Figure 8.)

The read cycle begins in T1 with the assertion of the address latch enable (ALE) signal. The trailing (low going) edge of this signal is used to latch the address information, which is valid on the address data bus (AD0-AD7) at this time, into the 8282/8283 latch. Address lines A8 through A15 do not need to be latched because they remain valid throughout the bus cycle. From T1 to T4 the IO/$\overline{M}$ signal indicates a memory or I/O operation. At T2 the address is removed from the address/data bus and the bus goes to a high impedance state. The read control signal is also asserted at T2. The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later, valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver (8286/8287) is required to buffer the 8088 local bus, signals DT/$\overline{R}$ and $\overline{DEN}$ are provided by the 8088.

A write cycle also begins with the assertion of ALE and the emission of the address. The IO/$\overline{M}$ signal is again asserted to indicate a memory or I/O write operation. In T2, immediately following the address emission, the processor emits the data to be written into the addressed location. This data remains valid until at least the middle of T4. During T2, T3, and T_W, the processor asserts the write control signal. The write ($\overline{WR}$) signal becomes active at the beginning of T2, as opposed to the read, which is delayed somewhat into T2 to provide time for the bus to float.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge ($\overline{INTA}$) signal is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 9.) In the second of two successive $\overline{INTA}$ cycles, a byte of information is read from the data bus, as supplied by the interrupt system logic (i.e. 8259A priority interrupt controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into the interrupt vector lookup table, as described earlier.

Bus Timing — Medium Complexity Systems

(See Figure 10.)

For medium complexity systems, the MN/$\overline{MX}$ pin is connected to GND and the 8288 bus controller is added to the system, as well as an 8282/8283 latch for latching the system address, and an 8286/8287 transceiver to allow for bus loading greater than the 8088 is capable of handling. Signals ALE, $\overline{DEN}$, and DT/$\overline{R}$ are generated by the 8288 instead of the processor in this configuration, although their timing remains relatively the same. The 8088 status outputs ($\overline{S2}$, $\overline{S1}$, and $\overline{S0}$) provide type of cycle information and become 8288 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 8288 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 8288 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence, data is not valid at the leading edge of write. The 8286/8287 transceiver receives the usual T and $\overline{OE}$ inputs from the 8288's DT/$\overline{R}$ and $\overline{DEN}$ outputs.

The pointer into the interrupt vector table, which is passed during the second $\overline{INTA}$ cycle, can derive from an 8259A located on either the local bus or the system bus. If the master 8259A priority interrupt controller is positioned on the local bus, a TTL gate is required to disable the 8286/8287 transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll".

The 8088 Compared to the 8086

The 8088 CPU is an 8-bit processor designed around the 8086 internal structure. Most internal functions of the 8088 are identical to the equivalent 8086 functions. The 8088 handles the external bus the same way the 8086 does with the distinction of handling only 8 bits at a time. Sixteen-bit operands are fetched or written in two consecutive bus cycles. Both processors will appear identical to the software engineer, with the exception of execution time. The internal register structure is identical and all instructions have the same end result. The differences between the 8088 and 8086 are outlined below. The engineer who is unfamiliar with the 8086 is referred to the IAPX 86, 88 User's Manual, Chapters 2 and 4, for function description and instruction set information. Internally, there are three differences between the 8088 and the 8086. All changes are related to the 8-bit bus interface.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . − 65°C to + 150°C
Voltage on Any Pin with
Respect to Ground . . . . . − 1.0 to + 7V
Power Dissipation . . . . . 2.5 Watt

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

(8088: T_A = 0°C to 70°C, V_{CC} = 5V ±10%)*
(8088-2: T_A = 0°C to 70°C, V_{CC} = 5V ±5%)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	+0.8	V	(See note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} +0.5	V	(See note 1,2)
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 µA
I_{CC}	Power Supply Current: 8088 8088-2 P8088		340 350 250	mA	T_A = 25°C
I_{LI}	Input Leakage Current		±10	µA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±10	µA	0.45V ≤ V_{OUT} ≤ V_{CC}
V_{CL}	Clock Input Low Voltage	-0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.9	V_{CC} +1.0	V	
C_{IN}	Capacitance if Input Buffer (All input except AD_0-AD_7, RQ/GT		15	pF	fc = 1 MHz
C_{IO}	Capacitance of I/O Buffer (AD_0-AD_7, RQ/GT		15	pF	fc = 1 MHz

*Note: For Extended Temperature EXPRESS V_{CC} = 5V ±5%

Note 1: V_{IL} tested with MN/$\overline{MX}$ Pin = 0V
V_{IH} tested with MN/$\overline{MX}$ Pin = 5V
MN/$\overline{MX}$ Pin is a strap Pin.
Note 2: Not applicable to $\overline{RQ}$/GT0 and $\overline{RQ}$/GT1 Pins (Pin 30 and 31)

A.C. CHARACTERISTICS (8088: T_A = 0°C to 70°C, V_{CC} = 5V ±10%)* (8088-2: T_A = 0°C to 70°C, V_{CC} = 5V ±5%)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

		8088		8088-2			
Symbol	Parameter	Min.	Max.	Min.	Max.	Units	Test Conditions
TCLCL	CLK Cycle Period	200	500	125	500	ns	
TCLCH	CLK Low Time	118		68		ns	
TCHCL	CLK High Time	69		44		ns	
TCH1CH2	CLK Rise Time		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	30		20		ns	
TCLDX	Data In Hold Time	10		10		ns	
TR1VCL	RDY Setup Time into 8284 (See Notes 1, 2)	35		35		ns	
TCLR1X	RDY Hold Time into 8284 (See Notes 1, 2)	0		0		ns	
TRYHCH	READY Setup Time into 8088	118		68		ns	
TCHRYX	READY Hold Time into 8088	30		20		ns	
TRYLCL	READY Inactive to CLK (See Note 3)	-8		-8		ns	
THVCH	HOLD Setup Time	35		20		ns	
TINVCH	INTR, NMI, TEST Setup Time (See Note 2)	30		15		ns	
TILIH	Input Rise Time (Except CLK)		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12	ns	From 2.0V to 0.8V

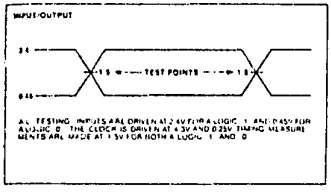
*Note: For Extended Temperature EXPRESS V_{CC} = 5V ±5%

A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

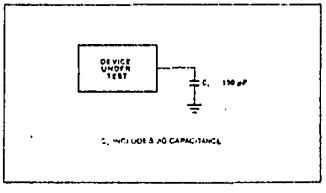
Symbol	Parameter	8088 Min.	8088 Max.	8088-2 Min.	8088-2 Max.	Units	Test Conditions
TCLAV	Address Valid Delay	10	110	10	60	ns	
TCLAX	Address Hold Time	10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH-20		TCLCH-10		ns	
TCLLH	ALE Active Delay		80		50	ns	
TCHLL	ALE Inactive Delay		85		55	ns	
TLLAX	Address Hold Time to ALE Inactive	TCHCL-10		TCHCL-10		ns	
TCLDV	Data Valid Delay	10	110	10	60	ns	C_L = 20-100 pF for all 8088 Outputs in addition to internal loads
TCHDX	Data Hold Time	10		10		ns	
TWHDX	Data Hold Time After $\overline{WR}$	TCLCH-30		TCLCH-30		ns	
TCVCTV	Control Active Delay 1	10	110	10	70	ns	
TCHCTV	Control Active Delay 2	10	110	10	60	ns	
TCVCTX	Control Inactive Delay	10	110	10	70	ns	
TAZRL	Address Float to READ Active	0		0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL-45		TCLCL-40		ns	
TCLHAV	HLDA Valid Delay	10	160	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL-75		2TCLCL-50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL-60		2TCLCL-40		ns	
TAVAL	Address Valid to ALE Low	TCLCH-60		TCLCH-40		ns	
TOLOH	Output Rise Time		20		20	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12	ns	From 2.0V to 0.8V

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING: INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". THE CLOCK IS DRIVEN AT 4.3V AND 0.25V. TIMING MEASUREMENTS ARE MADE AT 1.5V FOR BOTH A LOGIC "1" AND "0".

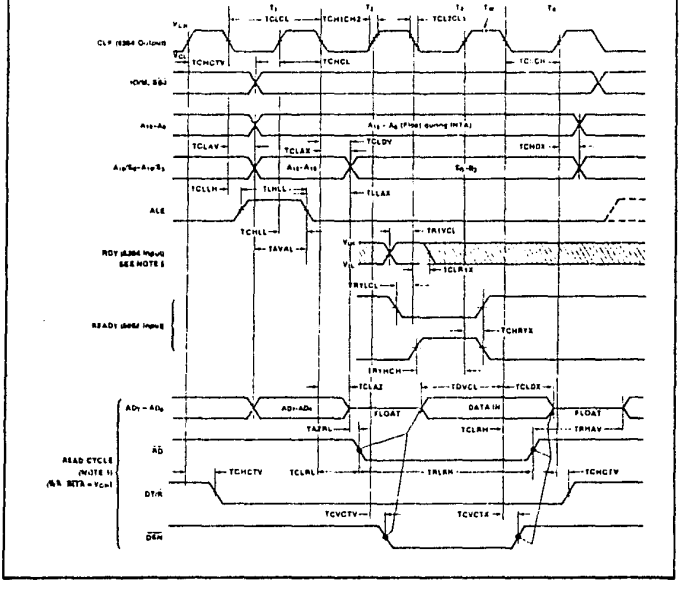
A.C. TESTING LOAD CIRCUIT



C_L INCLUDES JIG CAPACITANCE

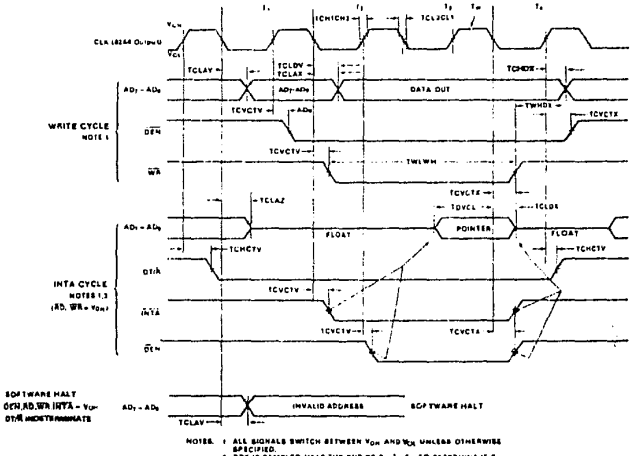
WAVEFORMS

BUS TIMING—MINIMUM MODE SYSTEM



WAVEFORMS (Continued)

BUS TIMING—MINIMUM MODE SYSTEM (Continued)



NOTES:
1. ALL SIGNALS SWITCH BETWEEN V_{OH} AND V_{OL} UNLESS OTHERWISE SPECIFIED.
2. RDY IS SAMPLED NEAR THE END OF T_2, T_3, T_W TO DETERMINE IF T_W MACHINES STATES ARE TO BE INSERTED.
3. TWO INTA CYCLES RUN BACK TO BACK. THE 8086 LOCAL ADDR/DATA BUS IS FLOATING DURING BOTH INTA CYCLES. CONTROL SIGNALS ARE SHOWN FOR THE SECOND INTA CYCLE.
4. SIGNALS AT 8284 ARE SHOWN FOR REFERENCE ONLY.
5. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED.

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 8288 BUS CONTROLLER)

TIMING REQUIREMENTS

		8088		8088-2			
Symbol	Parameter	Min.	Max.	Min.	Max.	Units	Test Conditions
TCLCL	CLK Cycle Period	200	500	125	500	ns	
TCLCH	CLK Low Time	118		68		ns	
TCHCL	CLK High Time	69		44		ns	
TCH1CH2	CLK Rise Time		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	30		20		ns	
TCLDX	Data In Hold Time	10		10		ns	
TR1VCL	RDY Setup Time into 8284 (See Notes 1, 2)	35		35		ns	
TCLR1X	RDY Hold Time into 8284 (See Notes 1, 2)	0		0		ns	
TRYHCH	READY Setup Time into 8088	118		68		ns	
TCHRYX	READY Hold Time into 8088	30		20		ns	
TRYLCL	READY Inactive to CLK (See Note 4)	-8		-8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, TEST) (See Note 2)	30		15		ns	
TGVCH	RQ/GT Setup Time	30		15		ns	
TCHGX	RQ Hold Time into 8088	40		30		ns	
TILIH	Input Rise Time (Except CLK)		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12	ns	From 2.0V to 0.8V

NOTES:
1. Signal at 8284 or 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state (8 ns into T3 state).
4. Applies only to T2 state (8 ns into T3 state)

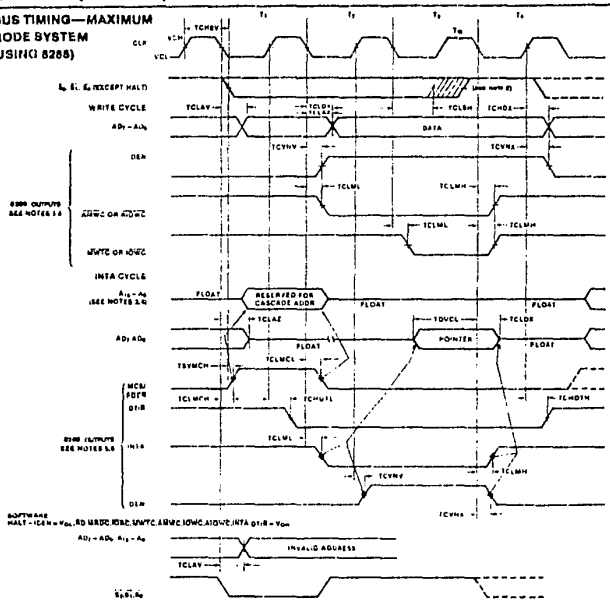
A.C. CHARACTERISTICS

TIMING RESPONSES

Symbol	Parameter	8088 Min.	8088 Max.	8088-2 Min.	8088-2 Max.	Units	Test Conditions
TCLML	Command Active Delay (See Note 1)	10	35	10	35	ns	
TCLMH	Command Inactive Delay (See Note 1)	10	35	10	35	ns	
TRYHSH	READY Active to Status Passive (See Note 3)		110		65	ns	
TCHSV	Status Active Delay	10	110	10	60	ns	
TCLSH	Status Inactive Delay	10	130	10	70	ns	
TCLAV	Address Valid Delay	10	110	10	60	ns	
TCLAX	Address Hold Time	10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	ns	
TSVLH	Status Valid to ALE High (See Note 1)		15		15	ns	
TSVMCH	Status Valid to MCE High (See Note 1)		15		15	ns	
TCLLH	CLK Low to ALE Valid (See Note 1)		15		15	ns	
TCLMCH	CLK Low to MCE High (See Note 1)		15		15	ns	
TCHLL	ALE Inactive Delay (See Note 1)		15		15	ns	
TCLMCL	MCE Inactive Delay (See Note 1)		15		15	ns	C_L = 20-100 pF for all 8088 Outputs in addition to internal loads
TCLDV	Data Valid Delay	10	110	10	60	ns	
TCHDX	Data Hold Time	10		10		ns	
TCVNV	Control Active Delay (See Note 1)	5	45	5	45	ns	
TCVNX	Control Inactive Delay (See Note 1)	10	45	10	45	ns	
TAZRL	Address Float to Read Active	0		0		ns	
TCLRL	RD Active Delay	10	165	10	100	ns	
TCLRH	RD Inactive Delay	10	150	10	80	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL-45		TCLCL-40		ns	
TCHDTL	Direction Control Active Delay (See Note 1)		50		50	ns	
TCHDTH	Direction Control Inactive Delay (See Note 1)		30		30	ns	
TCLGL	GT Active Delay		85		50	ns	
TCLGH	GT Inactive Delay		85		50	ns	
TRLRH	RD Width	2TCLCL-75		2TCLCL-50		ns	
TOLOH	Output Rise Time		20		20	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12	ns	From 2.0V to 0.8V

WAVEFORMS (Continued)

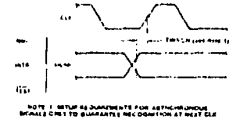
BUS TIMING—MAXIMUM MODE SYSTEM (USING 8288)



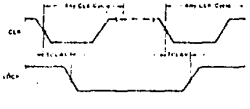
NOTES:
1. ALL SIGNALS SWITCH BETWEEN V_{OH} AND V_{OL} UNLESS OTHERWISE SPECIFIED.
2. RDY IS SAMPLED NEAR THE END OF T_2, T_3, T_W TO DETERMINE IF T_W MACHINE STATES ARE TO BE INSERTED.
3. CASCADE ADDRESS IS VALID BETWEEN FIRST AND SECOND INTA CYCLES.
4. TWO INTA CYCLES RUN BACK TO BACK. THE 8088 LOCAL ADDR/DATA BUS IS FLOATING DURING BOTH INTA CYCLES. CONTROL FOR POINTER ADDRESS IS SHOWN FOR SECOND INTA CYCLE.
5. SIGNALS AT 8284 OR 8288 ARE SHOWN FOR REFERENCE ONLY.
6. THE ISSUANCE OF THE 8288 COMMAND AND CONTROL SIGNALS (MRDC, MWTC, AMWC, IORC, IOWC, AIOWC, INTA, AND DEN) LAGS THE ACTIVE HIGH 8288 CEN.
7. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED.
8. STATUS INACTIVE IN STATE JUST PRIOR TO T_4.

WAVEFORMS (Continued)

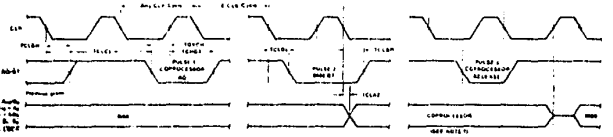
ASYNCHRONOUS SIGNAL RECOGNITION



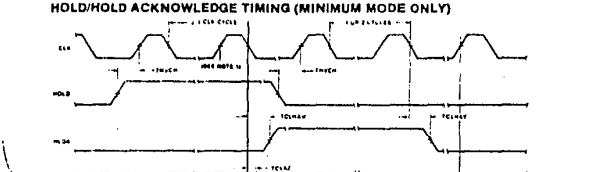
BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)

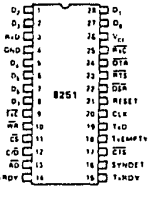


PROGRAMMABLE COMMUNICATION INTERFACE

- Synchronous and Asynchronous Operation
  - Synchronous:
    5-8 Bit Characters
    Internal or External Character Synchronization
    Automatic Sync Insertion
  - Asynchronous:
    5-8 Bit Characters
    Clock Rate — 1, 16 or 64 Times Baud Rate
    Break Character Generation
    1, 1½, or 2 Stop Bits
    False Start Bit Detection
- Baud Rate —DC to 56k Baud (Sync Mode)
  DC to 9.6k Baud (Async Mode)
- Full Duplex, Double Buffered, Transmitter and Receiver
- Error Detection — Parity, Overrun, and Framing
- Fully Compatible with 8080 CPU
- 28-Pin DIP Package
- All Inputs and Outputs Are TTL Compatible
- Single 5 Volt Supply
- Single TTL Clock

The 8251 is a Universal Synchronous/Asynchronous Receiver / Transmitter (USART) Chip designed for data communications in microcomputer systems. The USART is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM Bi-Sync). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPT. The chip is constructed using N-channel silicon gate technology.

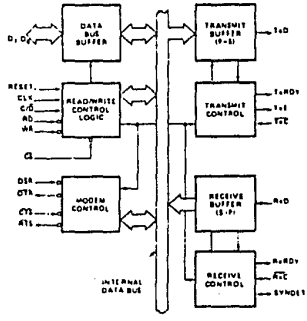
PIN CONFIGURATION



Pin Name	Pin Function
D_7-D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
RD	Read Data Command
WR	Write Data or Control Command
CS	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
TxC	Transmitter Clock
TxD	Transmitter Data
RxC	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (Not character for 8080)
TxRDY	Transmitter Ready (Ready for char. from 8080)

Pin Name	Pin Function
DSR	Data Set Ready
DTR	Data Terminal Ready
SYNDET	Sync Detect
RTS	Request to Send Data
CTS	Clear to Send Data
TxE	Transmitter Empty
Vcc	+5 Volt Supply
GND	Ground

BLOCK DIAGRAM



8251 BASIC FUNCTIONAL DESCRIPTION

General

The 8251 is a Universal Synchronous/Asynchronous Receiver/Transmitter designed specifically for the 8080 Microcomputer System. Like other I/O devices in the 8080 Microcomputer System its functional configuration is programmed by the system's software for maximum flexibility. The 8251 can support virtually any serial data technique currently in use (including IBM "bi-sync").

In a communication environment an interface device must convert parallel format system data into serial format for transmission and convert incoming serial format data into parallel system data for reception. The interface device must also delete or insert bits or characters that are functionally unique to the communication technique. In essence, the interface should appear "transparent" to the CPU, a simple input or output of byte-oriented system data.

Data Bus Buffer

This 3-state, bi-directional, 8-bit buffer is used to interface the 8251 to the 8080 system Data Bus. Data is transmitted or received by the buffer upon execution of INput or OUTput instructions of the 8080 CPU. Control words, Command words and Status information are also transferred through the Data Bus Buffer.

Read/Write Control Logic

This functional block accepts inputs from the 8080 Control bus and generates control signals for overall device operation. It contains the Control Word Register and Command Word Register that store the various control formats for device functional definition.

RESET (Reset)

A "high" on this input forces the 8251 into an "idle" mode. The device will remain at "idle" until a new set of control words is written into the 8251 to program its functional definition.

CLK (Clock)

The CLK input is used to generate internal device timing and is normally connected to the Phase 2 (TTL) output of the 8224 Clock Generator. No external inputs or outputs are referenced to CLK but the frequency of CLK must be greater than 30 times the Receiver or Transmitter clock inputs for synchronous mode (4.5 times for asynchronous mode).

$\overline{WR}$ (Write)

A "low" on this input informs the 8251 that the CPU is outputting data or control words, in essence, the CPU is writing out to the 8251.

$\overline{RD}$ (Read)

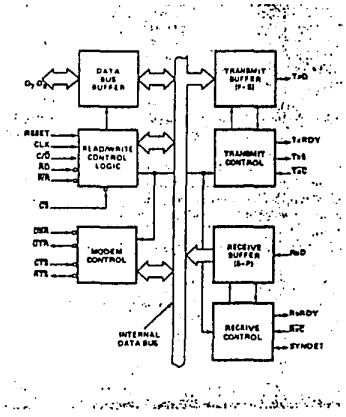
A "low" on this input informs the 8251 that the CPU is inputting data or status information, in essence, the CPU is reading from the 8251.

C/$\overline{D}$ (Control/Data)

This input, in conjunction with the $\overline{WR}$ and $\overline{RD}$ inputs informs the 8251 that the word on the Data Bus is either a data character, control word or status information.
1 = CONTROL 0 = DATA

$\overline{CS}$ (Chip Select)

A "low" on this input enables the 8251. No reading or writing will occur unless the device is selected.



C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	8251 → DATA BUS
0	1	0	0	DATA BUS → 8251
1	0	1	0	STATUS → DATA BUS
1	1	0	0	DATA BUS → CONTROL
X	X	X	1	DATA BUS → 3-STATE

Modem Control

The 8251 has a set of control inputs and outputs that can be used to simplify the interface to almost any Modem. The modem control signals are general purpose in nature and can be used for functions other than Modem control, if necessary.

$\overline{DSR}$ (Data Set Ready)

The $\overline{DSR}$ input signal is general purpose in nature. Its condition can be tested by the CPU using a Status Read operation. The $\overline{DSR}$ input is normally used to test Modem conditions such as Data Set Ready.

$\overline{DTR}$ (Data Terminal Ready)

The $\overline{DTR}$ output signal is general purpose in nature. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{DTR}$ output signal is normally used for Modem control such as Data Terminal Ready or Rate Select.

$\overline{RTS}$ (Request to Send)

The $\overline{RTS}$ output signal is general purpose in nature. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{RTS}$ output signal is normally used for Modem control such as Request to Send.

$\overline{CTS}$ (Clear to Send)

A "low" on this input enables the 8251 to transmit data (serial) if the Tx EN bit in the Command byte is set to a "one."

Transmitter Buffer

The Transmitter Buffer accepts parallel data from the Data Bus Buffer, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin.

Transmitter Control

The Transmitter Control manages all activities associated with the transmission of serial data. It accepts and issues signals both externally and internally to accomplish this function.

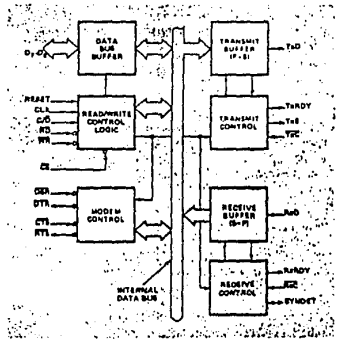
TxRDY (Transmitter Ready)

This output signals the CPU that the transmitter is ready to accept a data character. It can be used as an interrupt to the system or for the Polled operation the CPU can check TxRDY using a status read operation. TxRDY is automatically reset when a character is loaded from the CPU.

TxE (Transmitter Empty)

When the 8251 has no characters to transmit, the TxE output will go "high". It resets automatically upon receiving a character from the CPU. TxE can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplexed operational mode.

In SYNChronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be transmitted automatically as "fillers".



$\overline{TxC}$ (Transmitter Clock)

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the frequency of $\overline{TxC}$ is equal to the actual Baud Rate (1X). In Asynchronous transmission mode, the frequency of $\overline{TxC}$ is a multiple of the actual Baud Rate. A portion of the mode instruction selects the value of the multiplier; it can be 1x, 16x or 64x the Baud Rate.

For Example:

If Baud Rate equals 110 Baud,
$\overline{TxC}$ equals 110 Hz (1x)
$\overline{TxC}$ equals 1.76 kHz (16x)
$\overline{TxC}$ equals 7.04 kHz (64x).
If Baud Rate equals 9600 Baud,
$\overline{TxC}$ equals 614.4 kHz (64x).

The falling edge of $\overline{TxC}$ shifts the serial data out of the 8251.

Receiver Buffer

The Receiver accepts serial data, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU. Serial data is input to the RxD pin.

Receiver Control

This functional block manages all receiver-related activities.

RxRDY (Receiver Ready)

This output indicates that the 8251 contains a character that is ready to be input to the CPU. RxRDY can be connected to the interrupt structure of the CPU or for Polled operation the CPU can check the condition of RxRDY using a status read operation. RxRDY is automatically reset when the character is read by the CPU.

$\overline{RxC}$ (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the frequency of $\overline{RxC}$ is equal to the actual Baud Rate (1x). In Asynchronous Mode, the frequency of $\overline{RxC}$ is a multiple of the actual Baud Rate. A portion of the mode instruction selects the value of the multiplier; it can be 1x, 16x or 64x the Baud Rate.

For Example: If Baud Rate equals 300 Baud,
$\overline{RxC}$ equals 300 Hz (1x)
$\overline{RxC}$ equals 4800 Hz (16x)
$\overline{RxC}$ equals 19.2 kHz (64x).
If Baud Rate equals 2400 Baud,
$\overline{RxC}$ equals 2400 Hz (1x)
$\overline{RxC}$ equals 38.4 kHz (16x)
$\overline{RxC}$ equals 153.6 kHz (64x).

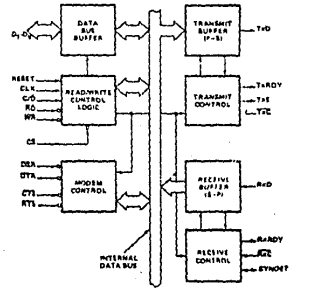
Data is sampled into the 8251 on the rising edge of $\overline{RxC}$.

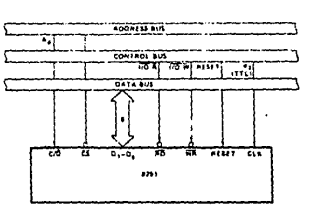
NOTE: In most communications systems, the 8251 will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the same. Both $\overline{TxC}$ and $\overline{RxC}$ will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

SYNDET (SYNC Detect)

This pin is used in SYNChronous Mode only. It is used as either input or output, programmable through the Control Word. It is reset to "low" upon RESET. When used as an output (Internal Sync mode), the SYNDET pin will go "high" to indicate that the 8251 has located the SYNC character in the Receive mode. If the 8251 is programmed to use double Sync characters (bi-sync), then SYNDET will go "high" in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

When used as an input, (external SYNC detect mode), a positive going signal will cause the 8251 to start assembling data characters on the falling edge of the next $\overline{RxC}$. Once in SYNC, the "high" input signal can be removed. The duration of the high signal should be at least equal to the period of $\overline{RxC}$.





8251 Interface to 8080 Standard System Bus

D.C. Characteristics:

T_A = 0°C to 70°C; V_{CC} = 5.0V ± 5%; V_{SS} = 0V

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	V_{SS}-.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.6mA
V_{OH}	Output High Voltage	2.2			V	I_{OH} = -100μA (DB_{0-7}) I_{OH} = -100μA (Others)
I_{DL}	Data Bus Leakage			50	μA	V_{OUT} = 4.5V
I_{LI}	Input Load Current			10	μA	@ 5.5V
I_{CC}	Power Supply Current		45	80		

Capacitance

T_A = 25°C; V_{CC} = V_{SS} = 0V

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}.

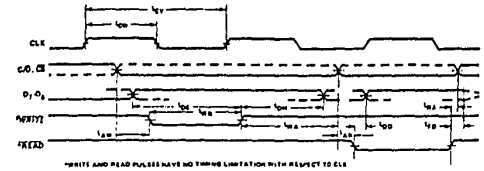
A.C. Characteristics:

$T_A = 0°C$ to $70°C$; $V_{CC} = 5.0V \pm 5\%$; $V_{SS} = 0V$

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t_{CY}	Clock Period	.420		1.35	μs	
$t_{\phi W}$	Clock Pulse Width	220		300	ns	
t_{R}, t_{F}	Clock Rise and Fall Time	0		50	ns	
t_{WR}	$\overline{WRITE}$ Pulse Width	430			ns	
t_{DS}	Data Set-Up Time for $\overline{WRITE}$	0			ns	
t_{DH}	Data Hold Time for $\overline{WRITE}$	65			ns	
t_{AW}	Address Stable before $\overline{WRITE}$	20			ns	
t_{WA}	Address Hold Time for $\overline{WRITE}$	35			ns	
t_{RD}	READ Pulse Width	430			ns	
t_{DD}	Data Delay from $\overline{READ}$			350	ns	$C_L = 100pF$
t_{DF}	$\overline{READ}$ to Data Floating	150			ns	$C_L = 100pF$
t_{AR1}	Address Stable before $\overline{READ}$, CE (C/D)	50			ns	
t_{RA1}	Address Hold Time for $\overline{READ}$, CE	5			ns	
t_{RA2}	Address Hold Time for $\overline{READ}$, C/D	370			ns	
t_{DT_X}	TxD Delay from Falling Edge of TxC			1	μs	$C_L = 100pF$
t_{SR_X}	Rx Data Set-Up Time to Sampling Pulse	2			μs	$C_L = 100pF$
t_{HR_X}	Rx Data Hold Time to Sampling Pulse	2			μs	$C_L = 100pF$
f_{T_X}	Transmitter Input Clock Frequency 1X Baud Rate 16X and 64X Baud Rate	 DC DC		 56 615	 KHz KHz	
f_{R_X}	Receiver Input Clock Frequency 1X Baud Rate 16X and 64X Baud Rate	 DC DC		 56 615	 KHz KHz	
t_{TX}	TxRDY Delay from Center of Data Bit			16	CLK Period	$C_L = 50pF$
t_{RX}	RxRDY Delay from Center of Data Bit	15		20	CLK Period	
t_{IS}	Internal Syndet Delay from Center of Data Bit	20		25	CLK Period	
t_{ES}	External Syndet Set-Up Time before Falling Edge of RxC			15	CLK Period	

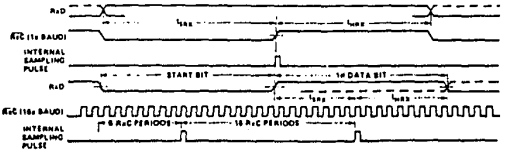
Note: The TxC and RxC frequencies have the following limitation with respect to CLK.
For ASYNC Mode, t_{T_X} or $t_{R_X} > 4.5\ t_{CY}$
For SYNC Mode, t_{T_X} or $t_{R_X} > 30\ t_{CY}$

READ AND WRITE TIMING

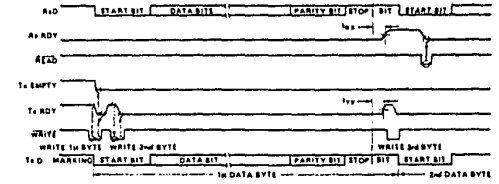


TRANSMITTER CLOCK AND DATA

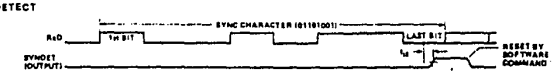
RECEIVER CLOCK AND DATA



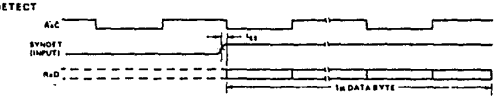
Tx RDY AND Rx RDY TIMING (ASYNC MODE)



INTERNAL SYNC DETECT



EXTERNAL SYNC DETECT



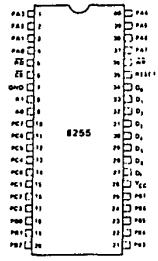
PROGRAMMABLE PERIPHERAL INTERFACE

- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with MCS™-8 and MCS™-80 Microprocessor Families
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40 Pin Dual In-Line Package
- Reduces System Package Count

The 8255 is a general purpose programmable I/O device designed for use with both the 8008 and 8080 microprocessors. It has 24 I/O pins which may be individually programmed in two groups of twelve and used in three major modes of operation. In the first mode (Mode 0), each group of twelve I/O pins may be programmed in sets of 4 to be input or output. In Mode 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining four pins three are used for handshaking and interrupt control signals. The third mode of operation (Mode 2) is a Bidirectional Bus mode which uses 8 lines for a bidirectional bus, and five lines, borrowing one from the other group, for handshaking.

Other features of the 8255 include bit set and reset capability and the ability to source 1mA of current at 1.5 volts. This allows darlington transistors to be directly driven for applications such as printers and high voltage displays.

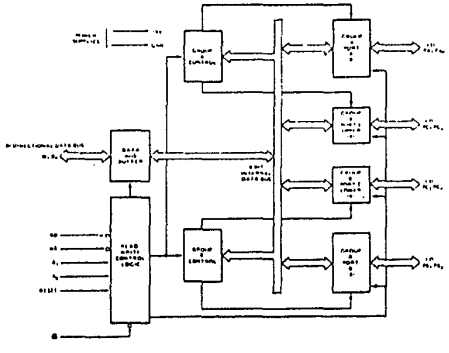
PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255 BLOCK DIAGRAM



8255 BASIC FUNCTIONAL DESCRIPTION

General

The 8255 is a Programmable Peripheral Interface (PPI) device designed for use in 8080 Microcomputer Systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the 8080 system bus. The functional configuration of the 8255 is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state, bi-directional, eight bit buffer is used to interface the 8255 to the 8080 system data bus. Data is transmitted or received by the buffer upon execution of INput or OUTput instructions by the 8080 CPU. Control Words and Status information are also transferred through the Data Bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the 8080 CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

($\overline{CS}$)

Chip Select: A "low" on this input pin enables the communication between the 8255 and the 8080 CPU.

($\overline{RD}$)

Read: A "low" on this input pin enables the 8255 to send the Data or Status information to the 8080 CPU on the Data Bus. In essence, it allows the 8080 CPU to "read from" the 8255.

($\overline{WR}$)

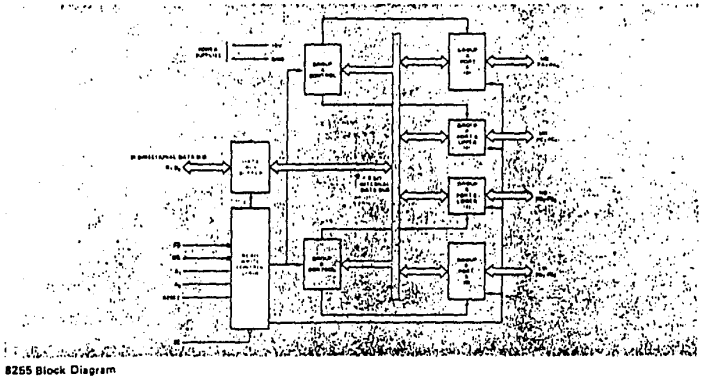
Write: A "low" on this input pin enables the 8080 CPU to write Data or Control words into the 8255.

(A_0 and A_1)

Port Select 0 and Port Select 1: These input signals, in conjunction with the $\overline{RD}$ and $\overline{WR}$ inputs, control the selection of one of the three ports or the Control Word Register. They are normally connected to the least significant bits of the Address Bus (A_0 and A_1).

8255 BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A – DATA BUS
0	1	0	1	0	PORT B – DATA BUS
1	0	0	1	0	PORT C – DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS – PORT A
0	1	1	0	0	DATA BUS – PORT B
1	0	1	0	0	DATA BUS – PORT C
1	1	1	0	0	DATA BUS – CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS – 3-STATE
1	1	0	1	0	ILLEGAL CONDITION



8255 Block Diagram

(RESET)

Reset: A "high" on this input clears all internal registers including the Control Register and all ports (A, B, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the 8080 CPU "outputs" a control word to the 8255. The control word contains information such as "mode", "bit set", "bit reset" etc. that initializes the functional configuration of the 8255.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A – Port A and Port C upper (C7-C4)
Control Group B – Port B and Port C lower (C3-C0)

The Control Word Register can Only be written into. No Read operation of the Control Word Register is allowed

Ports A, B, and C

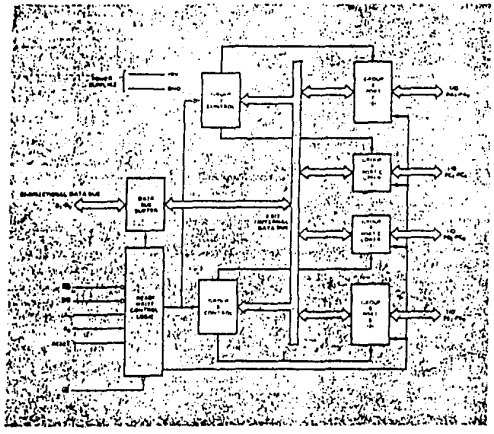
The 8255 contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255.

Port A: One 8-bit data output latch/buffer and one 8-bit data input latch.

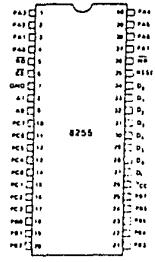
Port B: One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C: One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with Ports A and B.

8255 BLOCK DIAGRAM



PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (8-BIT)
PB7-PB0	PORT B (8-BIT)
PC7-PC0	PORT C (8-BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255 DETAILED OPERATIONAL DESCRIPTION

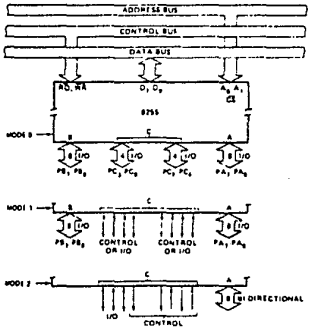
Mode Selection

There are three basic modes of operation that can be selected by the system software:

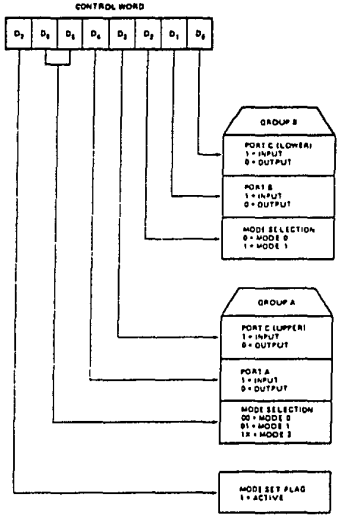
Mode 0 — Basic Input/Output
Mode 1 — Strobed Input/Output
Mode 2 — Bi-Directional Bus

When the RESET input goes "high" all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the RESET is removed the 8255 can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single OUTput instruction. This allows a single 8255 to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance: Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.



Basic Mode Definitions and Bus Interface

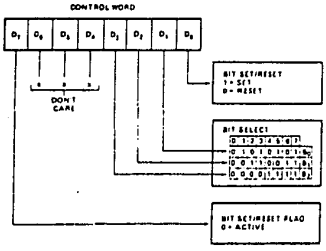


Mode Definition Format

The Mode definitions and possible Mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255 has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.



Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255 is programmed to operate in Mode 1 or Mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from Port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the Bit set/reset function of Port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without effecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET) — INTE is SET — Interrupt enable
(BIT-RESET) — INTE is RESET — Interrupt disable

Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

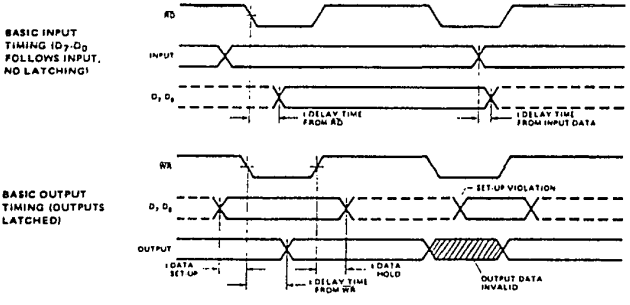
Operating Modes

Mode 0 (Basic Input/Output)

This functional configuration provides simple Input and Output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

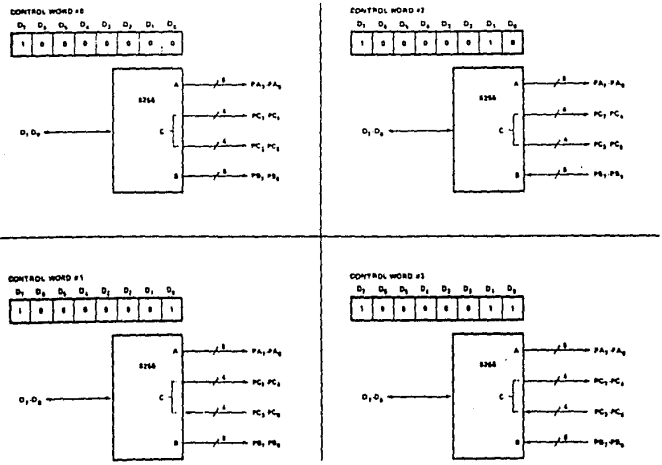


Mode 0 Timing

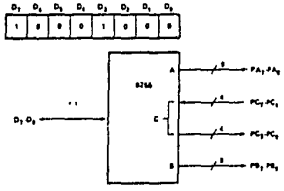
MODE 0 PORT DEFINITION CHART

A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

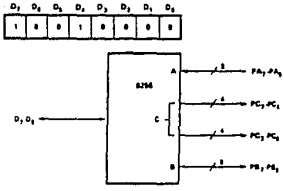
MODE 0 CONFIGURATIONS



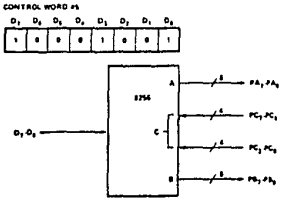
CONTROL WORD #4



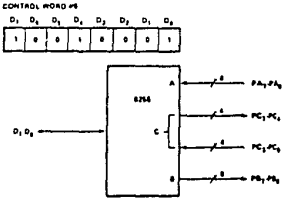
CONTROL WORD #8



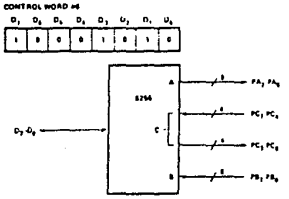
CONTROL WORD #5



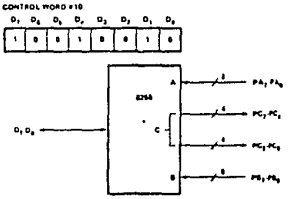
CONTROL WORD #9



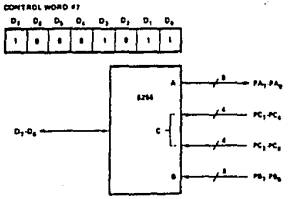
CONTROL WORD #6



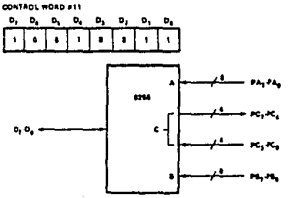
CONTROL WORD #10

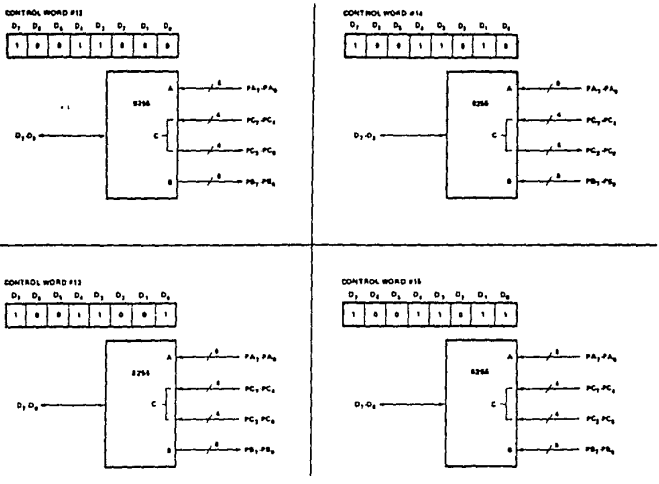


CONTROL WORD #7



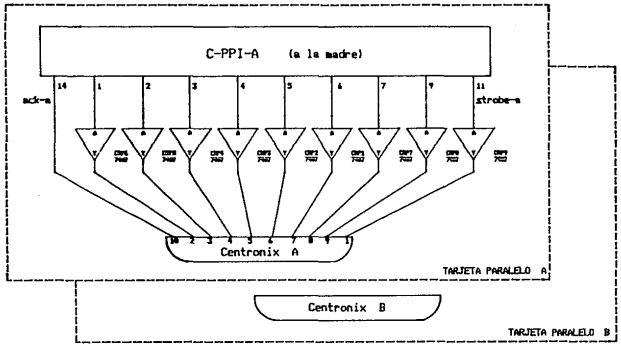
CONTROL WORD #11



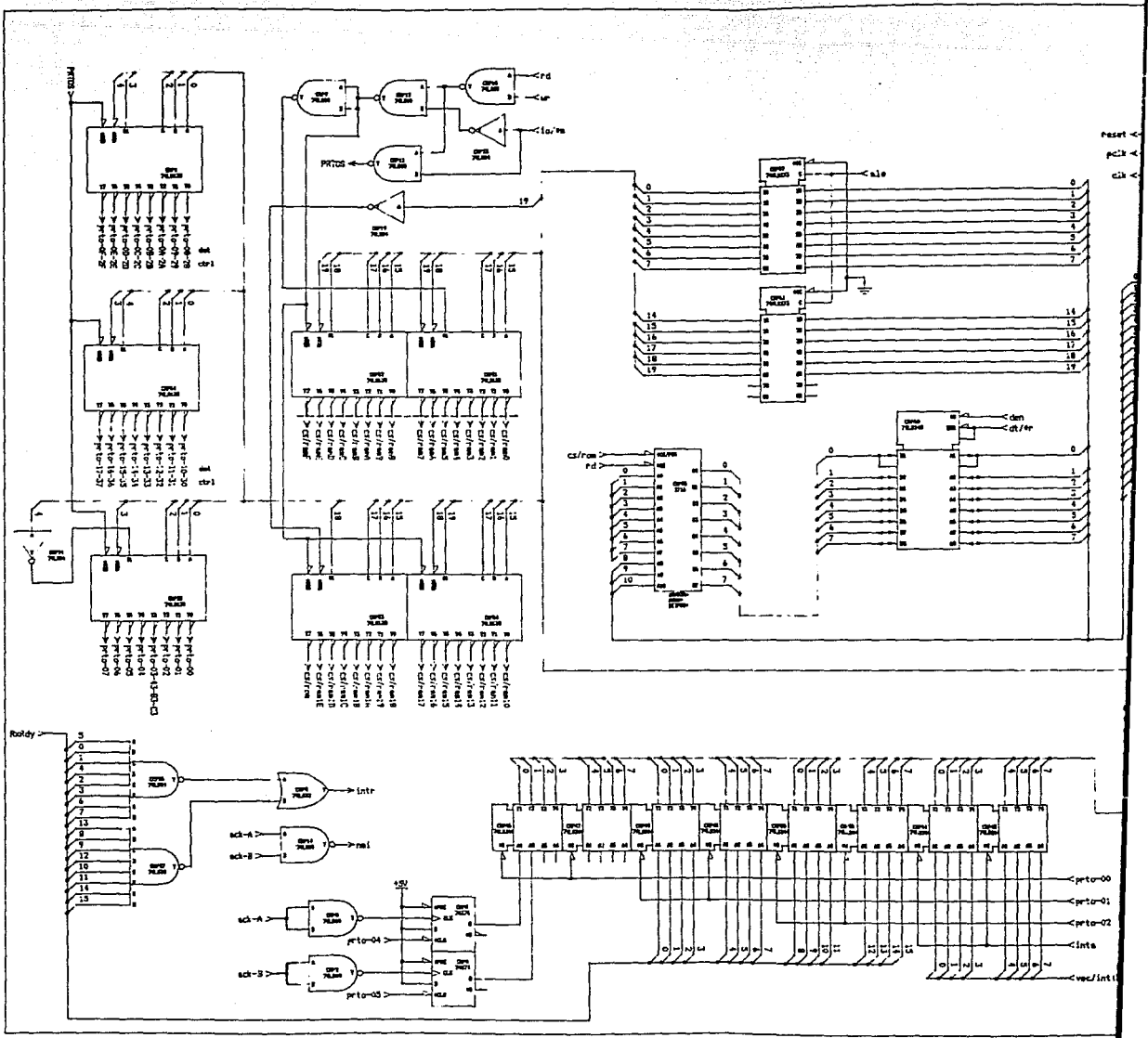

CONTROL WORD #12
CONTROL WORD #14
CONTROL WORD #13
CONTROL WORD #15
8255
A
B
C

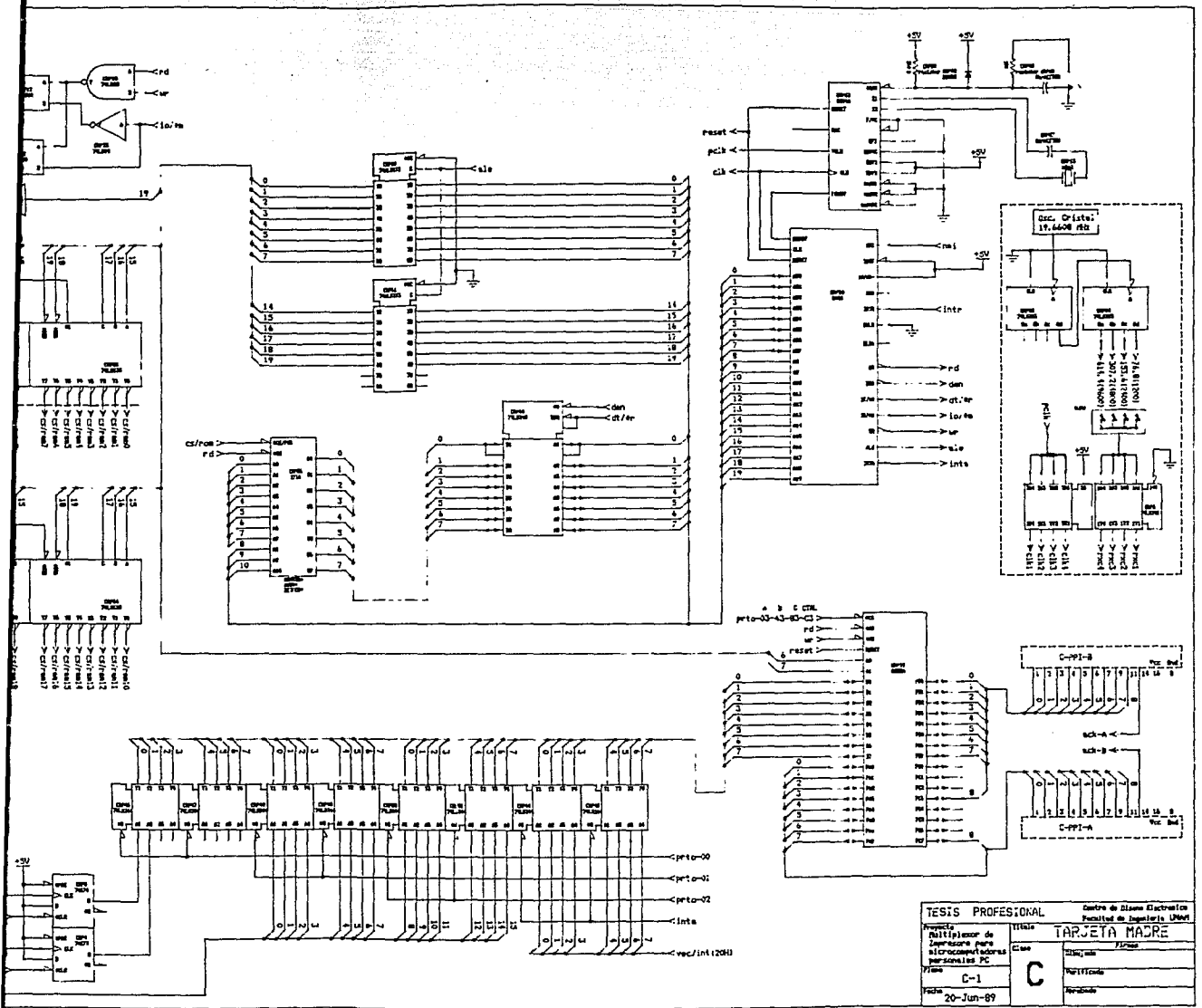
Apéndice E

Planos 3GJ

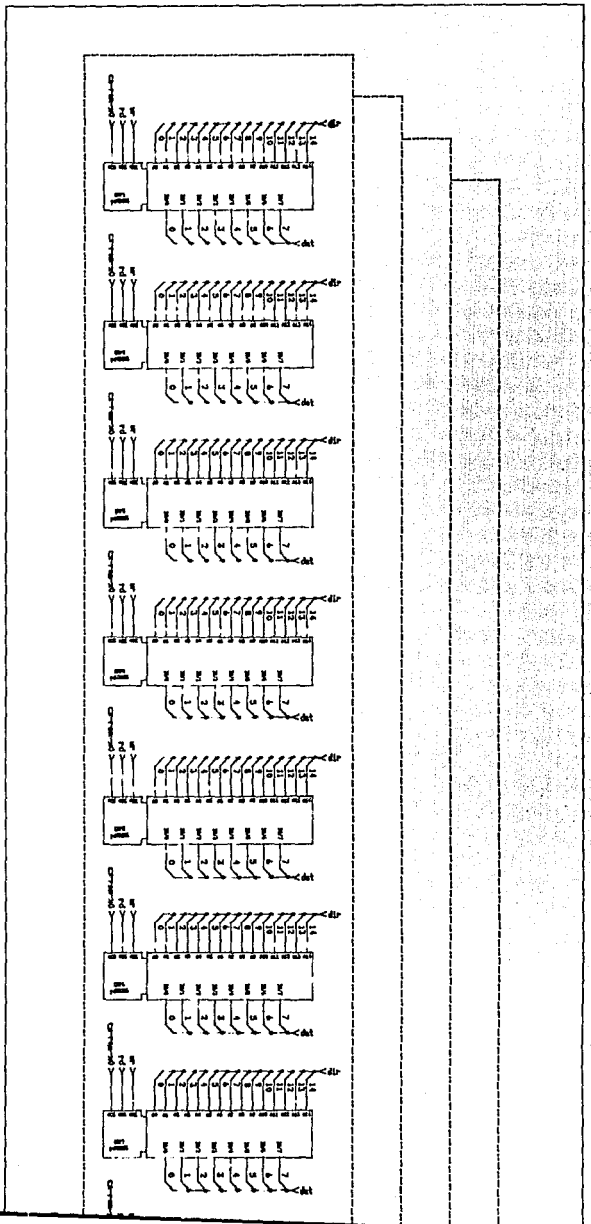


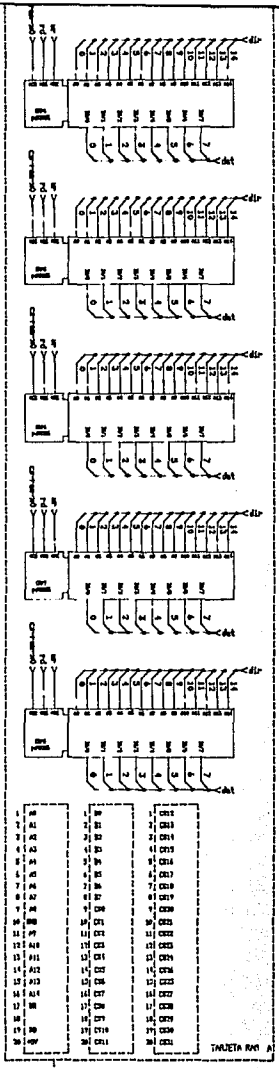
TESIS PROFESIONAL		Centro de Diseño Electrónico Facultad de Ingeniería UNAM
Proyecto: Multiplexor de Impresora para microcomputadoras personales PC	Título: TARJETAS PARALELO	Firmas
Plano: C-1	Clave: C	Dibujado
Fecha: 20-Jun-89		Verificado
		Aprobado


rd
wr
io/m
PRTOS
19
ale
reset
pclk
clk
cs/rom
rd
den
dt/r
Rxrdy
intr
ack-A
ack-B
rxrdy
nmi
+5V
prto-04
prto-03
prto-00
prto-01
prto-02
inta

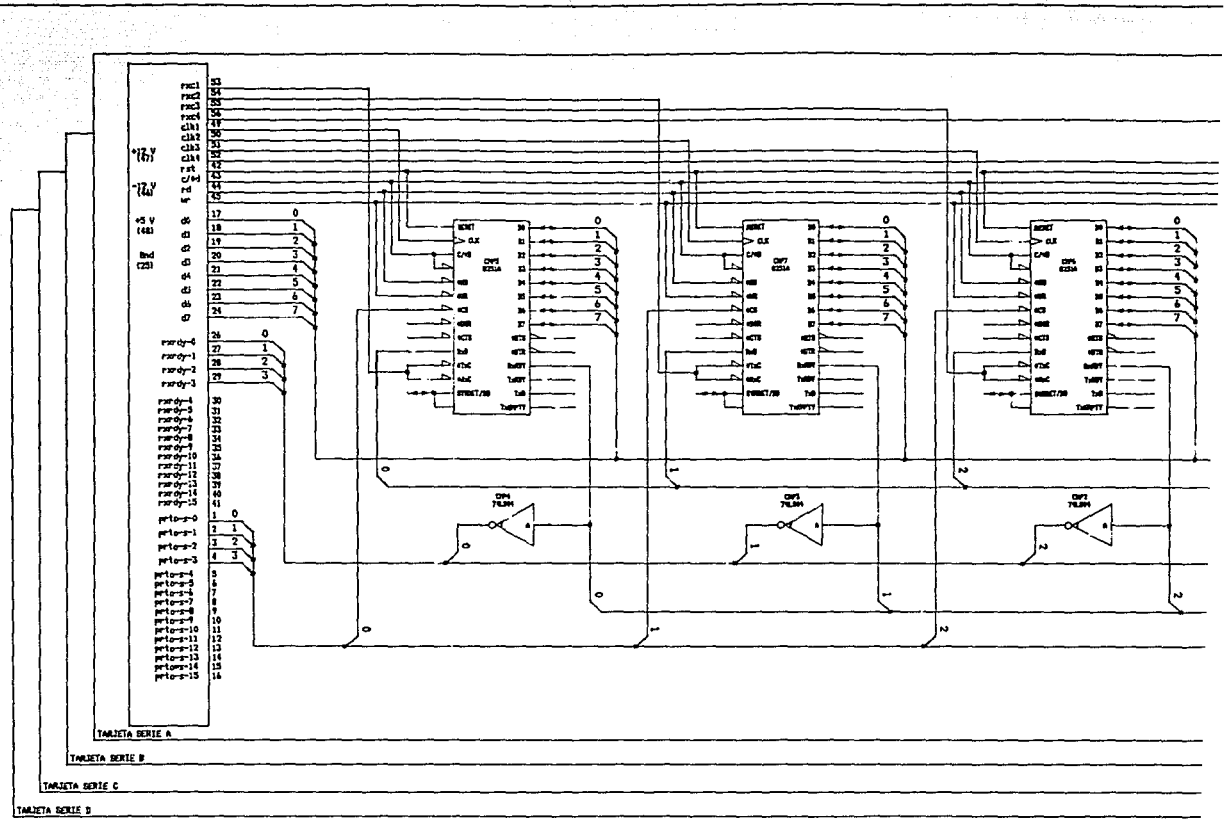


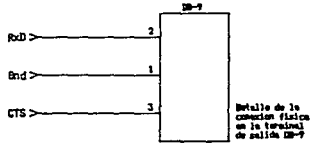
TESIS PROFESIONAL		Centro de Diseño Electrónico Facultad de Ingeniería UNAM
Proyecto Multiplexor de Impresora para microcomputadoras personales PC	Título TARJETA MADRE	
Plano C-1	Clave C	Firmas Diseñado Revisado Aprobado
Fecha 20-Jun-89		

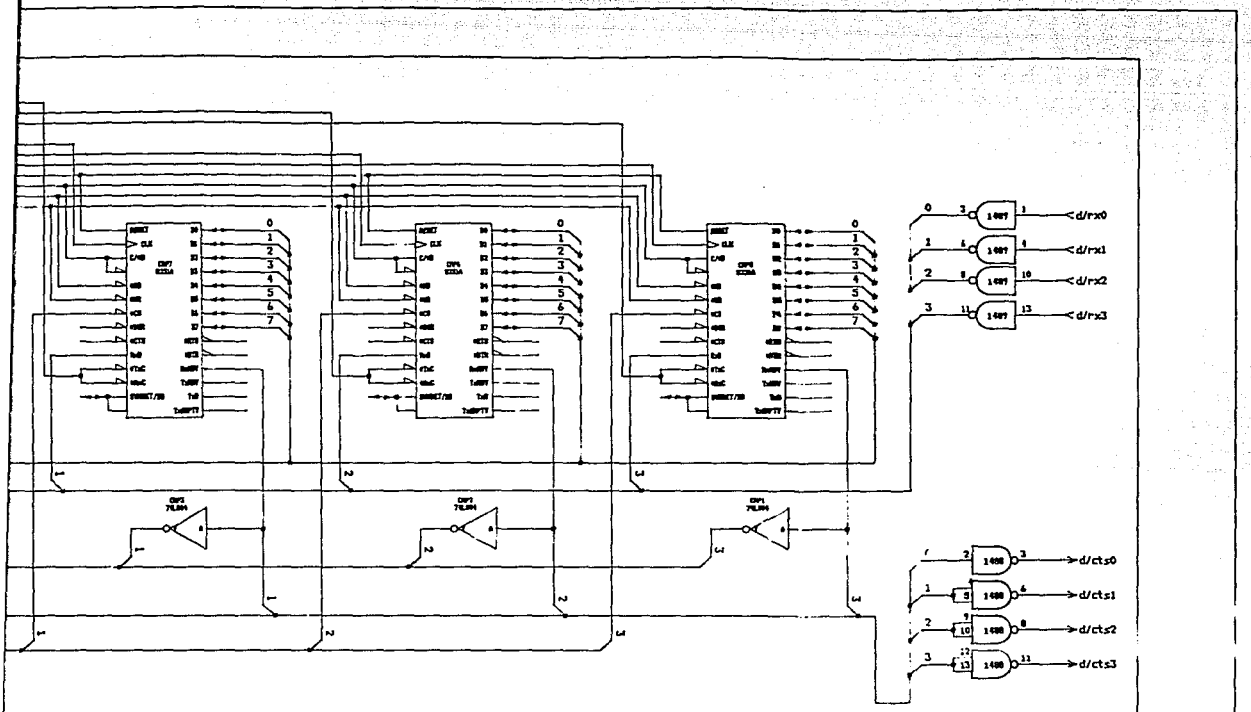




TESIS PROFESIONAL		Centro de Ciencias Aplicadas y Desarrollo Tecnológico, Facultad de Ingeniería UNAM	
Proyecto: multiplexor de impresoras para microcomputadoras personales PC	Título: TARJETAS RAM		
Plano: C-1	Clave: C	Dibujó	Firma
Fecha: 20-Jun-89		Verificado	
		Aprobado	


TARJETA SERIE A
TARJETA SERIE B
TARJETA SERIE C
TARJETA SERIE D


DB-9
RxD
Gnd
CTS
Detalle de la conexion fisica en la terminal de salida DB-9



Detalle de la conexion fisica en la terminal de salida DB-9

TESIS PROFESIONAL		Centro de Diseño Electronico Facultad de Ingenieria UNAM
Proyecto: Multiplexor de impresora para microcomputadoras personales PC	Titulo: TARJETAS SERIE	
Plano: C-1	Clase: C	Firmas: Dibujado / Verificado / Aprobado
Fecha: 20-Jun-89		

Apéndice F

Set de Instrucciones 8088/8086

MICRO LOGIC CORP.

MLC

HACKENSACK, NJ

8086 & 8088

MICROPROCESSOR INSTANT REFERENCE CARD

MICRO CHART®

Instruction Set

INST	ADDR	CODE
AAA	none	37
AAD	none	D5 [illegible]
AAM	none	D4 [illegible]
AAS	none	3F
ADC	r,r	10 [illegible]
ADC	m,r	10 [illegible]
ADC	rr,rr	11 [illegible]
ADC	mm,rr	11 [illegible]
ADC	r,m	12 [illegible]
ADC	rr,mm	13 [illegible]
ADC	r,i	80 [illegible]
ADC	m,i	80 [illegible]
ADC	rr,ii	81 [illegible]
ADC	mm,ii	81 [illegible]
ADC	rr,i	83 [illegible]
ADC	mm,i	83 [illegible]
ADC	AL,i	14 [illegible]
ADC	AX,ii	15 [illegible]
ADD	r,r	00 [illegible]
ADD	m,r	00 [illegible]
ADD	rr,rr	01 [illegible]
ADD	mm,rr	01 [illegible]
ADD	r,m	02 [illegible]
ADD	rr,mm	03 [illegible]
ADD	r,i	80 [illegible]
ADD	m,i	80 [illegible]
ADD	rr,ii	81 [illegible]
ADD	mm,ii	81 [illegible]
ADD	rr,i	83 [illegible]
ADD	mm,i	83 [illegible]
ADD	AL,i	04 [illegible]
ADD	AX,ii	05 [illegible]
AND	r,r	20 [illegible]
AND	m,r	20 [illegible]
AND	rr,rr	21 [illegible]
AND	mm,rr	21 [illegible]
AND	r,m	22 [illegible]
AND	rr,mm	23 [illegible]
AND	r,i	80 [illegible]
AND	m,i	80 [illegible]
AND	rr,ii	81 [illegible]
AND	mm,ii	81 [illegible]
AND	AL,i	24 [illegible]
AND	AX,ii	25 [illegible]
CALL	dd	E8 [illegible]
CALL	rr	FF [illegible]
CALL	mm	FF [illegible]
CALL	aaaa	9A [illegible]
CALL	dw	FF [illegible]
CBW	none	98
CLC	none	F8
CLD	none	FC
CLI	none	FA
CMC	none	F5
CMP	r,r	38 [illegible]
CMP	m,r	38 [illegible]
CMP	rr,rr	39 [illegible]
CMP	mm,rr	39 [illegible]
CMP	r,m	3A [illegible]
CMP	rr,mm	3B [illegible]
CMP	r,i	80 [illegible]
CMP	m,i	80 [illegible]
CMP	rr,ii	81 [illegible]
CMP	mm,ii	81 [illegible]
CMP	rr,i	83 [illegible]
CMP	mm,i	83 [illegible]
CMP	AL,i	3C [illegible]
CMP	AX,ii	3D [illegible]
CMPS	byte	A6
CMPS	word	A7
CWD	none	99
DAA	none	27
DAS	none	2F
DEC	m	FE [illegible]
DEC	mm	FF [illegible]
DEC	AL	FE [illegible]
DEC	CL	[illegible]
DEC	DL	[illegible]
DEC	BL	[illegible]
DEC	AH	[illegible]
DEC	CH	[illegible]
DEC	DH	[illegible]
DEC	BH	[illegible]
DEC	AX	[illegible]
DEC	CX	[illegible]
DEC	DX	[illegible]
DEC	BX	[illegible]
DEC	SP	[illegible]
DEC	BP	[illegible]
DEC	SI	[illegible]
DEC	DI	[illegible]
DIV	r	[illegible]
DIV	m	[illegible]
DIV	rr	[illegible]
DIV	mm	[illegible]
ESC	0,r	D8 [illegible]
ESC	1,r	[illegible]
ESC	2,r	[illegible]
ESC	3,r	[illegible]
ESC	4,r	[illegible]
ESC	5,r	[illegible]
ESC	6,r	[illegible]
ESC	7,r	[illegible]
ESC	0,mm	[illegible]
ESC	1,mm	[illegible]
ESC	2,mm	[illegible]
ESC	3,mm	[illegible]
ESC	4,mm	[illegible]
ESC	5,mm	DD [illegible]
ESC	6,mm	DE [illegible]
ESC	7,mm	DF [illegible]
HLT	none	F4
IDIV	r	F6 [illegible]
IDIV	m	F6 [illegible]
IDIV	rr	F7 [illegible]
IDIV	mm	F7 [illegible]
IMUL	r	F6 [illegible]
IMUL	m	F6 [illegible]
IMUL	rr	F7 [illegible]
IMUL	mm	F7 [illegible]
IN	AL,i	E4 [illegible]
IN	AX,i	E5 [illegible]
IN	AL,DX	EC
IN	AX,DX	ED
INC	m	FE [illegible]
INC	mm	FF [illegible]
INC	AL	FE [illegible]
INC	CL	FE [illegible]
INC	DL	FE [illegible]
INC	BL	FE [illegible]
INC	AH	FE [illegible]
INC	CH	FE [illegible]
INC	DH	FE [illegible]
INC	BH	FE [illegible]
INC	AX	40
INC	CX	41
INC	DX	42
INC	BX	43
INC	SP	44
INC	BP	45
INC	SI	46
INC	DI	47
INT	3	CC
INT	i	CD [illegible]
INTO	none	CE
IRET	none	CF
JA	d	77 [illegible]
JAE	d	73 [illegible]
JB	d	72 [illegible]
JBE	d	76 [illegible]
JC	d	72 [illegible]
JCXZ	d	E3 [illegible]
JE	d	74 [illegible]
JG	d	7F [illegible]
JGE	d	7D [illegible]
JL	d	7C [illegible]
JLE	d	7E [illegible]
JMP	d	EB [illegible]
JMP	dd	E9 [illegible]
JMP	rr	FF [illegible]
JMP	mm	FF [illegible]
JMP	aaaa	EA [illegible]
JMP	dw	FF [illegible]
JNA	d	76 [illegible]
JNAE	d	72 [illegible]
JNB	d	73 [illegible]
JNBE	d	77 [illegible]
JNC	d	73 [illegible]
JNE	d	75 [illegible]
JNG	d	7E [illegible]
JNGE	d	7C [illegible]
JNL	d	7D [illegible]
JNLE	d	7F [illegible]
JNO	d	71 [illegible]
JNP	d	7B [illegible]
JNS	d	79 [illegible]
JNZ	d	75 [illegible]
JO	d	70 [illegible]
JP	d	7A [illegible]
JPE	d	7A [illegible]
JPO	d	7B [illegible]
JS	d	78 [illegible]
JZ	d	74 [illegible]
LAHF	none	9F
LDS	rr,dw	C5 [illegible]
LEA	rr,m	8D [illegible]
LES	rr,dw	C4 [illegible]
LOCK	prfx	F0
LODS	byte	AC
LODS	word	AD
LOOP	d	[illegible]
LOOPE	d	[illegible]
LOOPZ	d	[illegible]
LOOPNZ	d	[illegible]
LOOPNE	d	[illegible]
MOV	r,r	[illegible]
MOV	m,r	[illegible]
MOV	rr,rr	[illegible]
MOV	mm,rr	[illegible]
MOV	r,m	[illegible]
MOV	rr,mm	[illegible]
MOV	m,i	[illegible]
MOV	mm,ii	[illegible]
MOV	AL,i	[illegible]
MOV	CL,i	[illegible]
MOV	DL,i	[illegible]
MOV	BL,i	[illegible]
MOV	AH,i	[illegible]
MOV	CH,i	[illegible]
MOV	DH,i	[illegible]
MOV	BH,i	[illegible]
MOV	AX,ii	[illegible]
MOV	CX,ii	[illegible]
MOV	DX,ii	[illegible]
MOV	BX,ii	[illegible]
MOV	SP,ii	[illegible]
MOV	BP,ii	[illegible]
MOV	SI,ii	[illegible]
MOV	DI,ii	[illegible]
MOV	AL,aa	[illegible]
MOV	AX,aa	[illegible]
MOV	aa,AL	[illegible]
MOV	aa,AX	[illegible]
MOV	sr,rr	8E [illegible]
MOV	sr,mm	8E [illegible]
MOV	rr,sr	8C [illegible]
MOV	mm,sr	8C [illegible]
MOVS	byte	A4
MOVS	word	A5
MOVSB	none	A4
MOVSW	none	A5
MUL	r	F6 [illegible]
MUL	m	F6 [illegible]
MUL	rr	F7 [illegible]
MUL	mm	F7 [illegible]
NEG	r	F6 [illegible]
NEG	m	F6 [illegible]
NEG	rr	F7 [illegible]
NEG	mm	F7 [illegible]
NOP	none	90
NOT	r	F6 [illegible]
NOT	m	F6 [illegible]
NOT	rr	F7 [illegible]
NOT	mm	F7 [illegible]
OR	r,r	08 [illegible]
OR	m,r	08 [illegible]
OR	rr,rr	09 [illegible]
OR	mm,rr	09 [illegible]
OR	r,m	0A [illegible]
OR	rr,mm	0B [illegible]
OR	r,i	80 [illegible]
OR	m,i	80 [illegible]
OR	rr,ii	81 [illegible]
OR	mm,ii	81 [illegible]
OR	rr,i	83 [illegible]
OR	mm,i	83 [illegible]
OR	AL,i	0C [illegible]
OR	AX,ii	0D [illegible]
OUT	i,AL	E6 [illegible]
OUT	i,AX	E7 [illegible]
OUT	DX,AL	EE
OUT	DX,AX	EF
POP	mm	8F [illegible]
POP	AX	58
POP	CX	59
POP	DX	5A
POP	BX	5B
POP	SP	5C
POP	BP	5D
POP	SI	5E
POP	DI	5F
POP	ES	07
POP	CS	0F
POP	SS	17
POP	DS	1F
POPF	none	9D
PUSH	mm	FF [illegible]
PUSH	AX	50
PUSH	CX	51
PUSH	DX	52
PUSH	BX	53
PUSH	SP	54
PUSH	BP	55
PUSH	SI	56
PUSH	DI	57
PUSH	ES	06
PUSH	CS	0E
PUSH	SS	16
PUSH	DS	1E
PUSHF	none	9C
RCL	r,1	D0 [illegible]
RCL	m,1	D0 [illegible]
RCL	rr,1	D1 [illegible]
RCL	mm,1	D1 [illegible]
RCL	r,CL	D2 [illegible]
RCL	m,CL	D2 [illegible]
RCL	rr,CL	D3 [illegible]
RCL	mm,CL	D3 [illegible]
RCR	r,1	D0 [illegible]
RCR	m,1	D0 [illegible]
RCR	rr,1	D1 [illegible]
RCR	mm,1	D1 [illegible]
RCR	r,CL	D2 [illegible]
RCR	m,CL	D2 [illegible]
RCR	rr,CL	D3 [illegible]
RCR	mm,CL	D3 [illegible]
REP	prfx	[illegible]
REPE	prfx	[illegible]
REPZ	prfx	[illegible]
REPNE	prfx	[illegible]
REPNZ	prfx	[illegible]
RET	none	[illegible]
RET	ii	[illegible]
RET	aa	[illegible]
RET	ii,aa	[illegible]
ROL	r,1	D0 [illegible]
ROL	m,1	D0 [illegible]
ROL	rr,1	D1 [illegible]
ROL	mm,1	D1 [illegible]
ROL	r,CL	D2 [illegible]
ROL	m,CL	D2 [illegible]
ROL	rr,CL	D3 [illegible]
ROL	mm,CL	D3 [illegible]
ROR	r,1	D0 [illegible]
ROR	m,1	D0 [illegible]
ROR	rr,1	D1 [illegible]
ROR	mm,1	D1 [illegible]
ROR	r,CL	D2 [illegible]
ROR	m,CL	D2 [illegible]
ROR	rr,CL	D3 [illegible]
ROR	mm,CL	D3 [illegible]
SAHF	none	[illegible]
SAL	r,1	D0 [illegible]
SAL	m,1	D0 [illegible]
SAL	rr,1	D1 [illegible]
SAL	mm,1	D1 [illegible]
SAL	r,CL	D2 [illegible]
SAL	m,CL	D2 [illegible]
SAL	rr,CL	D3 [illegible]
SAL	mm,CL	D3 [illegible]
SAR	r,1	D0 [illegible]
SAR	m,1	D0 [illegible]
SAR	rr,1	D1 [illegible]
SAR	mm,1	D1 [illegible]
SAR	r,CL	D2 [illegible]
SAR	m,CL	D2 [illegible]
SAR	rr,CL	D3 [illegible]
SAR	mm,CL	D3 [illegible]
SBB	r,r	18 [illegible]
SBB	m,r	18 [illegible]
SBB	rr,rr	19 [illegible]
SBB	mm,rr	19 [illegible]
SBB	r,m	1A [illegible]
SBB	rr,mm	1B [illegible]
SBB	r,i	80 [illegible]
SBB	m,i	80 [illegible]
SBB	rr,ii	81 [illegible]
SBB	mm,ii	81 [illegible]
SBB	rr,i	83 [illegible]
SBB	mm,i	83 [illegible]
SBB	AL,i	1C [illegible]
SBB	AX,ii	1D [illegible]
SCAS	byte	AE
SCAS	word	AF
SEG		
CS	prfx	2E
DS	prfx	3E
ES	prfx	26
SS	prfx	36
SHL	r,1	D0 [illegible]
SHL	m,1	D0 [illegible]
SHL	rr,1	D1 [illegible]
SHL	mm,1	D1 [illegible]
SHL	r,CL	D2 [illegible]
SHL	m,CL	D2 [illegible]
SHL	rr,CL	D3 [illegible]
SHL	mm,CL	D3 [illegible]
SHR	r,1	D0 [illegible]
SHR	m,1	D0 [illegible]
SHR	rr,1	D1 [illegible]
SHR	mm,1	D1 [illegible]
SHR	r,CL	D2 [illegible]
SHR	m,CL	D2 [illegible]
SHR	rr,CL	D3 [illegible]
SHR	mm,CL	D3 [illegible]
STC	none	F9
STD	none	FD
STI	none	FB
STOS	byte	AA
STOS	word	AB
SUB	r,r	28 [illegible]
SUB	m,r	28 [illegible]
SUB	rr,rr	29 [illegible]
SUB	mm,rr	29 [illegible]
SUB	r,m	2A [illegible]
SUB	rr,mm	2B [illegible]
SUB	r,i	80 [illegible]
SUB	m,i	80 [illegible]
SUB	rr,ii	81 [illegible]
SUB	mm,ii	81 [illegible]
SUB	rr,i	83 [illegible]
SUB	mm,i	83 [illegible]
SUB	AL,i	2C [illegible]
SUB	AX,ii	2D [illegible]
TEST	r,r	84 [illegible]
TEST	r,m	84 [illegible]
TEST	rr,rr	85 [illegible]
TEST	rr,mm	85 [illegible]
TEST	r,i	F6 [illegible]
TEST	m,i	F6 [illegible]
TEST	rr,ii	F7 [illegible]
TEST	mm,ii	F7 [illegible]
TEST	AL,i	A8 [illegible]
TEST	AX,ii	A9 [illegible]
WAIT	none	9B
XCHG	r,r	[illegible]
XCHG	r,m	[illegible]
XCHG	rr,rr	[illegible]
XCHG	rr,mm	[illegible]
XCHG	AX,CX	[illegible]
XCHG	AX,DX	[illegible]
XCHG	AX,BX	[illegible]
XCHG	AX,SP	[illegible]
XCHG	AX,BP	[illegible]
XCHG	AX,SI	[illegible]
XCHG	AX,DI	[illegible]
XLAT	byte	[illegible]
XOR	r,r	30 [illegible]
XOR	m,r	30 [illegible]
XOR	rr,rr	31 [illegible]
XOR	mm,rr	31 [illegible]
XOR	r,m	32 [illegible]
XOR	rr,mm	33 [illegible]
XOR	r,i	80 [illegible]
XOR	m,i	80 [illegible]
XOR	rr,ii	81 [illegible]
XOR	mm,ii	81 [illegible]
XOR	rr,i	83 [illegible]
XOR	mm,i	83 [illegible]
XOR	AL,i	34 [illegible]
XOR	AX,ii	35 [illegible]

Cycle Codes

A1	4	A2	11(15)	A3	unused
B1	80		15	B3	12
C1	83	B2	8(12)+	C3	12(16)
D1	3		12+		16
E1	16+	C2	101-112	D3	5 17
F1	16(24)+	D2	107-118+	E3	5 19
	24+	E2	165-184	F3	8+
G1	9+	F2	11+	G3	8(12)+
H1	9(13)+	G2	80-90		12+
	13+	H2	86-104+	H3	18
I1	17+	I2	128-154	I3	18(26)
J1	17(25)+	J2	11(15)+		26
	25+		15+	J3	70-77
K1	18	K2	10	K3	76-83+
	23	L2	10(14)	L3	118-133
L1	16		14	M3	4+5N
	20	M2	8	N3	17(21)+
M1	21(25)+	N2	8(12)		25+
	29+		12	O3	8
N1	28	O2	52		12
	36		72	P3	16(20)+
O1	37(45)+	P2	51		24+
	53+		71	Q3	11
P1	2	Q2	4 53		15
Q1	10+		4 73	R3	10
R1	10(14)+	R2	32		14
	14+		44	S3	8+4N
S1	22	S2	4 16	T3	20+4N+
T1	22(30)	T2	8 18	U3	N/A
	30	U2	15	V3	16
U1	5	V2	11		20
V1	15+	W2	18(22)+	W3	20
W1	15(23)+		22+		24
	23+	X2	24(32)+	X3	26
X1	80-90		32+		34
Y1	86-96+	Y2	16(24)+	Y3	25
Z1	144-162		24+		33
		Z2	2+	Z3	15(19)
					19

A4	171-190(175-194)+
	175-194+
B4	134-160(136-164)+
	136-164+
C4	124-139(128-143)+
	128-143+
D4	154-172(158-176)+
	158-176+
E4	20(28)+4N+
	28+4N+
F4	9+17N
G4	9+17(25)N
	9+25N
H4	9+10N
I4	9+10(14)N
	9+14N
J4	9+22N
K4	9+22(30)N
	9+30N
L4	9+15N
M4	9+15(19)N
	9+19N
N4	9+13N
P4	9+13(17)N
	9+17N

Second Byte Table

T		X	0	1	2	3	4	5	6	7
7	S	(BX+SI)	00	08	10	18	20	28	30	38
8	U	(BX+DI)	01	09	11	19	21	29	31	39
8	M	(BP+SI)	02	0A	12	1A	22	2A	32	3A
7		(BP+DI)	03	0B	13	1B	23	2B	33	3B
5	P	(SI)	04	0C	14	1C	24	2C	34	3C
5	N	(DI)	05	0D	15	1D	25	2D	35	3D
6	T	(dd)	06	0E	16	1E	26	2E	36	3E
5	S	(BX)	07	0F	17	1F	27	2F	37	3F
11		(BX+SI+d)	40	48	50	58	60	68	70	78
12	T	(BX+DI+d)	41	49	51	59	61	69	71	79
12	O	(BP+SI+d)	42	4A	52	5A	62	6A	72	7A
11		(BP+DI+d)	43	4B	53	5B	63	6B	73	7B
9	D	(SI+d)	44	4C	54	5C	64	6C	74	7C
9	A	(DI+d)	45	4D	55	5D	65	6D	75	7D
9	T	(BP+d)	46	4E	56	5E	66	6E	76	7E
9	A	(BX+d)	47	4F	57	5F	67	6F	77	7F
11		(BX+SI+dd)	80	88	90	98	A0	A8	B0	B8
12	I	(BX+DI+dd)	81	89	91	99	A1	A9	B1	B9
12	N	(BP+SI+dd)	82	8A	92	9A	A2	AA	B2	BA
11		(BP+DI+dd)	83	8B	93	9B	A3	AB	B3	BB
9	M	(SI+dd)	84	8C	94	9C	A4	AC	B4	BC
9	E	(DI+dd)	85	8D	95	9D	A5	AD	B5	BD
9	M	(BP+dd)	86	8E	96	9E	A6	AE	B6	BE
9		(BX+dd)	87	8F	97	9F	A7	AF	B7	BF
0	R	AX or AL	C0	C8	D0	D8	E0	E8	F0	F8
0	E	CX or CL	C1	C9	D1	D9	E1	E9	F1	F9
0	G	DX or DL	C2	CA	D2	DA	E2	EA	F2	FA
0		BX or BL	C3	CB	D3	DB	E3	EB	F3	FB
0	D	SP or AH	C4	CC	D4	DC	E4	EC	F4	FC
0	A	BP or CH	C5	CD	D5	DD	E5	ED	F5	FD
0	T	SI or DH	C6	CE	D6	DE	E6	EE	F6	FE
0	A	DI or BH	C7	CF	D7	DF	E7	EF	F7	FF

N	0	1	2	3	4	5	6	7
8	AL	CL	DL	BL	AH	CH	DH	BH
9	AX	CX	DX	BX	SP	BP	SI	DI
L	ES	CS	SS	DS	—	—	—	—
M	ES	—	SS	DS	—	—	—	—
K	INC	DEC	CALL (2)	CALL (3)	JMP (2)	JMP (3)	PUSH	
J	INC	DEC						
I	TEST		NOT	NEG	MUL	IMUL	DIV	IDIV
H	TEST		NOT	NEG	MUL	IMUL	DIV	IDIV
G	ROL CL	ROR CL	RCL CL	RCR CL	SHL/SAL CL	SHR CL		SAR CL
F	ROL CL	ROR CL	RCL CL	RCR CL	SHL/SAL CL	SHR CL		SAR CL
E	ROL	ROR	RCL	RCR	SHL/SAL	SHR		SAR
D	ROL	ROR	RCL	RCR	SHL/SAL	SHR		SAR
C	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
B	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
A	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP

(2) = 'near' transfer indirect via word in reg or mem.

(3) = 'far' transfer indirect via double word in mem.

Example

After reading "About the Tables", usage of the tables can be verified by assembly and disassembly of

1) 1408 ADC AL,8
2) 0BD1 OR CX,DX
3) C6440708 MOV (SI+7),8
4) D1C6 ROL SI
5) D104 LP ROL (SI)
6) 72FC JC LP
7) D50A AAD

The following notes help avoid difficulty (when converting to hex) and correspond to the lines above:

1) Use "AL,i" - not "r,i".
2) Use "rr,rr" - not "r,r". Read about Second Byte Table to convert XX.
3) Parentheses indicate mem ptr and form is "m,i". Form of first operand is "(SI+d)".
4) Use "SI" from reg part of section X.
5) Use "(SI)" from mem part of section X.
6) Read about "Relative Jumps".
7) Special case for disassembly.

Hex and Decimal Conversion

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	2
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	3
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	4
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	5
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	6
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	7
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	8
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	9
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	A
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	B
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	C
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	D
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	E
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Memory Locations

00000 - 00003	Type 0 interrupt pointer for divide-error
00004 - 00007	Type 1 interrupt pointer for single-step
00008 - 0000B	Type 2 interrupt pointer for Non-Mask-int
0000C - 0000F	Type 3 interrupt pointer for 1-byte- inst
00010 - 00013	Type 4 interrupt pointer for INTO inst
00014 - 0007F	Type 5 thru 31 interrupt pointers reserved for Intel products
00080 - 000FF	Type 32 thru 255 available interrupt pointers (or general memory use)
00400 - FFFEF	Main memory space
FFFF0 - FFFFB	CPU jumps to code here upon Reset
FFFFC - FFFFF	Reserved for Intel products

ASCII

LSD \ MSD		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	_	o	DEL

Unused

Pinouts

8086

GND	1	40	VCC
AD14	2	39	AD15
AD13	3	38	A16/S3
AD12	4	37	A17/S4
AD11	5	36	A18/S5
AD10	6	35	A19/S6
AD9	7	34	BHE*/S7
AD8	8	33	MN/MX*
AD7	9	32	RD*
AD6	10	31	HOLD (RQ*/GT0*)
AD5	11	30	HLDA (RQ*/GT1*)
AD4	12	29	WR* (LOCK*)
AD3	13	28	M/IO* (S2*)
AD2	14	27	DT/R* (S1*)
AD1	15	26	DEN* (S0*)
AD0	16	25	ALE (QS0)
NMI	17	24	INTA* (QS1)
INTR	18	23	TEST*
CLK	19	22	READY
GND	20	21	RESET

On 8088 AD8 to AD15 are A8 to A15, pin 28 is IO/M* (S2*), pin 34 is SS0 (HIGH). Max mode is in parentheses. * means active low.

MICRO LOGIC CORP

HACKENSACK, NJ

Hex to Instruction Conversion

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	ADD XB byte	ADD XB word	ADD BX byte	ADD BX word	ADD AL,i	ADD AX,i	PUSH ES	POP ES	OR XB byte	OR BX word	OR BX byte	OR BX word	OR AL,i	OR AX,i	PUSH CS	POP CS	0
1	ADC XB byte	ADC XB word	ADC BX byte	ADC BX word	ADC AL,i	ADC AX,i	PUSH SS	POP SS	SBB XB byte	SBB XB word	SBB BX byte	SBB BX word	SBB AL,i	SBB AX,i	PUSH DS	POP DS	1
2	AND XB byte	AND XB word	AND BX byte	AND BX word	AND AL,i	AND AX,i	SEG =ES	DAA	SUB XB byte	SUB BX word	SUB BX byte	SUB BX word	SUB AL,i	SUB AX,i	SEG =CS	DAS	2
3	XOR XB byte	XOR XB word	XOR BX byte	XOR BX word	XOR AL,i	XOR AX,i	SEG =SS	AAA	CMP XB byte	CMP BX word	CMP BX byte	CMP BX word	CMP AL,i	CMP AX,i	SEG =DS	AAS	3
4	INC AX	INC CX	INC DX	INC BX	INC BP	INC SI	INC DI	INC AX	DEC CX	DEC DX	DEC BX	DEC BP	DEC SI	DEC DI	DEC DI	DEC DI	4
5	PUSH AX	PUSH CX	PUSH DX	PUSH BX	PUSH BP	PUSH SI	PUSH DI	POP AX	POP CX	POP DX	POP BX	POP BP	POP SI	POP DI	POP DI	POP DI	5
6	JO	JNO	JC/JB JNAE	JNC/JNB JAE	JE JZ	JNE JNZ	JBE JNA	JNBE JA	JS	JNS	JP JPE	JNP JPO	JL JNGE	JNL JGE	JLE JNG	JNLE JG	6
7	LEA	N	OP														7
8	AX	BX	AX	CX	TEST BX byte	TEST BX word	XCHG BX byte	XCHG BX word	MOV XB byte	MOV XB word	MOV BX byte	MOV BX word	MOV BX w rm	MOV XL	LEA BX rm	MOV MX	POP DX	8
9	NOP	XCHG AX,CX	XCHG AX,DX	XCHG AX,BX	XCHG AX,SP	XCHG AX,BP	XCHG AX,SI	XCHG AX,DI	CBW	CWD	CALL asss	WAIT	PUSHF	POPF	SAHF	LAHF	9
A	MOV AL,mm	MOV AX,mm	MOV mm,AL	MOV mm,AX	MOVS byte	MOVS word	CMPS byte	CMPS word	TEST AL,i	TEST AX,i	STOS byte	STOS word	LODS byte	LODS word	SCAS byte	SCAS word	A
B	MOV AL,i	MOV CL,i	MOV DL,i	MOV AH,i	MOV CH,i	MOV DH,i	MOV BH,i	MOV AX,i	MOV CX,i	MOV DX,i	MOV BX,i	MOV SP,i	MOV BP,i	MOV SI,i	MOV DI,i	MOV DI,i	B
C			RET n near	RET	LES BX rm,dw	LDS BX rm,dw	MOV XB rm,i	MOV BX rm,i			RET n far	RET far	INT 3	INT n	INTO	IRET	C
D	DX	EA	FF	GX	AAM (0A 0A)	AAD (0S 0A)		XLAT	ESC 0	ESC 1	ESC 2	ESC 3	ESC 4	ESC 5	ESC 6	ESC 7	D
E	LOOPNZ LOOPNE	LOOPZ LOOPE	LOOP	JCXZ	IN AL,i	IN AX,i	OUT i,AL	OUT i,AX	CALL dd	JMP dd	JMP asss	IN AL,DX	IN AX,DX	OUT DX,AL	OUT DX,AX		E
F	LOCK		REPNE REPNZ	REP REPE REPZ	HLT	CMC			CLC	STC	CLI	STI	CLD	STD	JX	KX	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Miscellaneous Notes

COMPATIBILITY: The 8088 and 8086 are 100% compatible on machine and assembly languages.

SEGMENTS: Memory segments are 64K byte addressed at the 16 megabyte space. The segments can be designated to code, stack, data, and extra seed. These physical locations are given by three respective segment (CS, SS, DS, ES) bases 16, Locations within a segment are specified by a 16-bit offset (or segment address) relative to the beginning of the segment.

ALIGNMENT: On both processors, words can start at even or odd addresses. However, on the 8086, each seed as store of an odd segment word adds 4 cycles to execution time. 8088 programmers should align word on the stack.

DESTINATION (L) SOURCE: Instructions that take data from some "source" and put a result at some "destination" and written in the form: MNEMONIC DESTINATION SOURCE

BYTE ORDER: Two byte and two word values (displacements and addresses) are stored in jump-type, and data areas are stored with Least significant half at Lower address

RELATIVE JUMPS: The destination address of a relative jump is the sum of the signed displacement and the address of the first byte of the next instruction.

STRING POINTERS: For string operations, while SI points into the DATA segment, note that DI points into the ES EXTRA segment.

BP FOR STACK: When register BP is selected in an instruction, the variable is assumed to reside in the STACK segment.

RESERVED PORTS: Ports OOFBH thru OOFFH of the 8A/A I/O locations are reserved for later production.

INTERRUPT NOTES: When a segment register and another value must be updated together (without the possibility of an intervening interrupt) (8 2 SS and SP), the segment register should be changed first and followed immediately by the other value. Interrupts and the other value (interrupts are not recognized immediately after a move to segment register. POP segment register or parts unaffected by NMI, software, stopped and hardware) property during repeated string operations and protected additional prefixes are the only seed assuming there are no programmer conflicts with string data. The NMI and INTR interrupt lines are multiplexed by edge and level triggered.

RESET: A hardware Reset sets CS=FFFF, DS=SS=ES=0000, FLAGS=0, and starts executing code at location FFFF0.

ARITHMETIC SHIFTS: All single-bit rotates and shifts set OF=1 if the MSB (sign bit) is changed by the operation, or the sign bit receives an original value. OF is cleared. OF is undefined after multi-bit operations.

PARITY FLAG: The parity flag affects the parity of only the low order 8 bits of operand. (Flag is set if even number of one-bits, cleared if odd.)

STRING TERMS: Pre-fed 8D and Unpacked BCD have respectively two and one binary coded decimal digits per byte. Unpacked ASCII = 30H plus ASCII.

LOGICAL INSTRUCTIONS: AND, OR, TEST, and XOR instructions clear the OF and CF flags.

SEGMENT OVERRIDE EXCEPTIONS: A segment override prefix can be attached to instructions (placed just before the opcode byte) to cause data to be addressed as any of the three alternatives to the default segment assignments. stack, doesn't store, string manipulations, and instruction fetches.

DERIVATIONS: This card is based on Intel publications.

About the Tables

FLAG CODES TABLE: In the FLAG CODES table, 'U' indicates that the flag becomes undefined. Otherwise the listed flag is affected according to the operation.

INSTRUCTION DESCRIPTION TABLE: The single letter column corresponds to the lettered column at the FLAG CODES table

HEX COLUMN OF INSTRUCTION SET: Non-HEX values for the second byte refer to sections of the SECOND BYTE TABLE (see below). Following the listed opcode (bytes) go an immediate displacement or address if appropriate and finally immediate data if specified.

'C' COLUMN OF INSTRUCTION SET: These codes refer to the CYCLE CODES TABLE.

CYCLE CODES TABLE: Listed numbers are execution times in clock periods (cp) for 8088 and 8086 timed either. the 8086 time is given first and the 8088 is given on the next line. A '+' terminates processes to add additional time for each EA clock periods per section 'T' of the SECOND BYTE TABLE. For the 8088, instruction fetch time is included. Word seed data is at an odd address. 8088 times assume the queue is at even address. Conditional jumps have a range JUMP times for FAILURE.SUCCESS. Note that sequence fetch and overlapping execution limit the figures shown. A series of fast executing instructions can drain the instruction queue and increase execution time, and instruction prefetch can sometimes overlap execution and increase execution time. The actual time for a code sequence is governed by typically between a 10% of the theoretical time although in special cases it can be much more. For the 8088, instruction segment size effect speed in some cases but usually not substantially.

SECOND BYTE TABLE: This table shows combination bit and three two of the second byte (excluding prefixes) of an instruction. The table is referred to by other parts of your and in such forms as XB, BX, XL, XX, etc. for examples, directs you to find the first segment of the 2-byte instruction somewhere between X and the second centered in section 2. Section 3 is located between X and segment 1. The second byte of the rest of the instruction is stands you to find the DISP, operand and scales to section 0 for the machine code. For disassembly, first find the second byte in the number mostly and determine the instruction from the assembler mnemonic in the indicated section. When assembling, these sum register values are taken from the bottom part of section 0 while register pointers are taken from the upper parts.

HEX TO INSTRUCTION TABLE: To convert from hex to an instruction, start with the first digit (MSD) and across for the second. Two-character codes under bases 0 and 1 refer to sections of the SECOND BYTE TABLE but only when they appear on the first of the two hex digits. On the second line, two-character codes refer to registers.

ADDRESSING COLUMN OF INSTRUCTION SET:

r	= byte register
w	= word register
i	= immediate byte value
I	= immediate word value
d	= immediate signed byte displacement
dd	= immediate signed word displacement
dw	= immediate two byte address offset from segment start (address bus of byte or word)
asss	= displacement byte byte address (3 byte offset followed by 3 byte segment address=16)
m	= memory field as specified by mm combinations of section X of SECOND BYTE TABLE
mm	= memory field specified by memory operators of section X of SECOND BYTE TABLE. (With CALL or JMP instructions, memory has 2 byte offset from segment start of point to get to.)
r	= reg or mem byte
w	= reg or mem word
sr	= segment register
mm	= memory double-word specified by memory operator of section 3 of SECOND BYTE TABLE. (With CALL or JMP instructions, memory has 2 byte offset from segment start followed by 2 byte segment address=16 of point to go to.)
xs	= within segment
as	= another segment
Q	= mm or r

Where 'byte' or 'word' is listed, the assembler may require a qualifying reference to locate

Instruction Description

Flag Codes

A	= A C OU PU SU ZU
B	= AU CU OU P S 2
C	= AC O S P Z
D	= AU C O P S Z
E	= EVERY FLAG
F	= NO OTHERS
G	= A D P S 2
H	= C D
I	= I T
J	= A C P S Z
K	= AU CU OU PU SU ZU
L	= AU C O PU SU ZU
M	= A C OU P S Z
N	= NONE

Flags

A	= Aux carry flag
C	= Carry flag
D	= Direction flag
I	= Interrupt enable
O	= Overflow flag
P	= Parity flag
S	= Sign flag
T	= Trap flag
Z	= Zero flag

Registers

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL
SP	STACK POINTER	
BP	BASE POINTER	
SI	SOURCE INDEX	
DI	DESTINATION INDEX	
IP	INSTRUCTION PNTR	
F	[- - - O D I T S Z - A - P - C]	
CS	CODE SEGMENT	
DS	DATA SEGMENT	
SS	STACK SEGMENT	
ES	EXTRA SEGMENT	

Intentionally Blank

AAA	A ASCII adjust for add — Restores AL to unpacked BCD after addition of unpacked BCD number (byte add only)
AAD	B ASCII adjust for division — Modifies AL before dividing unpacked BCD numbers so that result will be unpacked BCD (also by byte only) (AL=AH*0AH+AL&H=0), AND followed by DIV only produces unpacked BCD quotients between 0-9 Quotients above 9 will be in normal binary. For unpacked BCD quotients between 00-99 use AAM. ONLY MOV REMAIN AAM
AAM	B ASCII adjust for multiplication — Restores AH and AL to unpacked BCD after multiplication of unpacked BCD numbers (byte mul only)
AAS	A ASCII adjust for subtraction — Restores AL to unpacked BCD after subtraction of unpacked BCD number (byte sub only)
ADC	C Add with carry
ADD	C Add
AND	D Logical AND (clears CF, OF)
CALL	N Call procedure (pushes return addr)
CBW	N Convert byte to word — Extends sign bit of AL throughout AH
CLC	F Clear carry flag
CLD	F Clear direction flag — Pointers for auto increment of SI and DI during string-op
CLI	F Clear interrupt-enable flag — disables interrupts (except INTR and software interrupts)
CMC	F Complement carry flag
CMP	C Compare destination with source — JG, for example, and jump if destination is greater than source
CMPS	C Compare string — Compares bytes or words pointed to by DS and SI and updates DI and SI by 1 or 2 accordingly - JG, for example, will jump if destination is greater than source
CWD	N Convert word to double-word — Extends sign bit of AX throughout DX
DAA	H Decimal adjust for addition — Restores AL to packed BCD after addition of packed BCD number (byte add only)
DAS	H Decimal adjust for subtraction — Restores AL to packed BCD after subtraction of packed BCD number (byte sub only)
DEC	G Decrement by one
DIV	K Divide unsigned — (AL = AX / source AH = rest) or (AX = DX:AX / source, DX = rest) Type: if interrupt 0 div by 0 or quotient too large
ESC	N External — for instructions to coprocessor
HALT	N Halt system
IDIV	K Divide signed — see DIV
IMUL	L Multiply signed — (AX = AL * source) or (DX:AX = AX * source) CF and OF are set when upper half of result has significant bits
IN	N Input from port
INT	I Interrupt — Activates 1 of 256 interrupt procedures by software (clears IF, TF) (Jacks with hardware Int, flags are saved on stack)
INTO	N Interrupt on overflow — If OF=1 then INT 4 (clears IF, TF if successful)
IRET	I Interrupt return — Returns from interrupt procedures whether activated by hardware or software (flags are restored from stack)
JA	N Jump if above - unsigned
JAE	N Jump if above or equal - unsigned
JB	N Jump if below - unsigned
JBE	N Jump if below or equal - unsigned
JC	N Jump if carry - if CF=1
JCXZ	N Jump if CX register zero
JE	N Jump if equal - if ZF=1
JG	N Jump if greater - signed
JGE	N Jump if greater or equal - signed
JL	N Jump if less - signed
JLE	N Jump if less or equal - signed
JMP	N Jump unconditionally
JNA	N Jump if not above - unsigned
JNAE	N Jump if not above nor equal - unsigned
JNB	N Jump if not below - unsigned
JNBE	N Jump if not below nor equal - unsigned
JNC	N Jump if not carry - if CF=0
JNE	N Jump if not equal - if ZF=0
JNG	N Jump if not greater - signed
JNGE	N Jump if not greater nor equal - signed
JNL	N Jump if not less - signed
JNLE	N Jump if not less nor equal - signed
JNO	N Jump if not overflow - if OF=0
JNP	N Jump if not parity - if PF=0
JNS	N Jump if not sign - if SF=0
JNZ	N Jump if not zero - if ZF=0
JO	N Jump if overflow - if OF=1
JP	N Jump if parity - if PF=1
JPE	N Jump if parity even - if PF=1
JPO	N Jump if parity odd - if PF=0
JS	N Jump if sign - if SF=1
JX	N Jump if zero - if ZF=1
LAHF	N Load AH from low byte of flags
LDS	N Load DS and reg (DE + 2) pointer addressed location in memory is stored into a register (first word) and register DS (second word)
LEA	N Load effective address - The address offset from beginning of segment) of the source operand (as opposed to its value) is stored into a register
LES	N Load pointer using ES - Similar to LDS
LOCK	N Lock bus - A prefix causing CPU to assert LOCK signal during execution of next-set instruction
LODS	N Load string - Loads byte or word pointed to by SI into AL or AX and updates SI by 1 or 2 accordingly
LOOP	N Loop - Decrement CX by one and jump if CX not zero
LOOPE	N Loop while equal - Decrements CX and jumps if CX not zero and ZF=1
LOOPNE	N Loop while not equal - Same as LOOPNZ
LOOPNZ	N Loop while not zero - Decrements CX and jumps if CX not zero and ZF=0
LOOPZ	N Loop while zero - Same as LOOPE
MOV	N Move - Moves to destination from source
MOVS	N Move string - Moves byte or word pointed to by SI to location pointed to by DI and updates pointers by 1 or 2 accordingly.
MOVSB	N Move string byte - See MOVS
MOVSW	N Move string word - See MOVS
MUL	L Multiply unsigned - See IMUL
NEG	C Negate - two's complement (multiply by -1)
NOP	N No operation
NOT	N Logical NOT - one's complement
OR	D Logical OR (clears CF, OF)
OUT	N Output to port
POP	N Pop word from stack - SP increases after pop
POPF	E Pop flags from stack
PUSH	N Push word onto stack - SP decreases first
PUSHF	N Push flags onto stack
RCL	K Rotate thru carry left - by 1 or by CL
RCR	K Rotate thru carry right - by 1 or by CL
REP	N Repeat string operation
RET	N Return from procedure - Not too use with near-type procedures. Optional operand (usually even #) is added to SP to dump passed parameters
ROL	H Rotate left - by 1 or by CL - CF = MSB of result
ROR	H Rotate right - by 1 or by CL - CF = MSB of result
SAHF	J Short AH into low byte of flags
SAL	D Short arithmetic left - zero fill - by 1 or by CL
SAR	D Short arithmetic right - sign expansion - by 1 or by CL. CF = last bit shifted out - note that negative numbers are rounded INCORRECTLY by SAR
SBB	C Subtract with borrow - destination minus source
SCAS	C Scan string - Compares AL or AX with byte or word pointed to by DI and updates DI by 1 or 2 accordingly. AF is undefined until jump if AL or AX is greater than string operand
SHL	D Short logical left - Same as SAL
SHR	D Short logical right - zero fill - by 1 or by CL - CF = last bit shifted out
STC	F Set carry flag
STD	F Set direction flag - Prepares for auto decrement of SI and DI during string-op
STI	F Set interrupt-enable flag - enables interrupts (see CLI)
STOS	N Store string - Stores AL or AX into location pointed to by DI and updates DI by 1 or 2 accordingly
SUB	C Subtract - destination minus source
TEST	D Test by logical AND - affects only flags (and clears CF, OF)
WAIT	N Wait for TEST pin activity
XCHG	N Exchange - switches contents of destination and source
XLAT	N Translate - Replaces AL with a byte from a table pointed to by BX. AL contains the position in the table. The first element is at position 0.
XOR	D Logical Exclusive-OR - Differing bits yield bit = 1. Less for immediate operand (CF, OF)

REPEAT INSTRUCTIONS

REP LODS	Repeats LODS CX times (1) - 9+6 cycles for bytes, 9+6 for words
REP MOVS	Repeats MOVS CX times (1) - 9+4 cycles for bytes, 9+4 for words
REP STOS	Repeats STOS CX times (1) - 9+4 cycles for bytes, 10 for words
REPE CMPS	Repeats CMPS until strings mismatch but not more than CX times (1) - 9+6 cycles for bytes, 9+6 for words
REPE SCAS	Repeats SCAS until AL or AX mismatches string but not more than CX times (1) - 9+5 cycles for bytes, 9+5 for words
REPNE CMPS	Repeats CMPS until strings match but not more than CX times (1) - 9+6 cycles for bytes, 9+6 for words
REPNE SCAS	Repeats SCAS until AL or AX matches string but not more than CX times (1) - 9+5 cycles for bytes, 9+5 for words
REPNZ CMPS	Same as REPNE CMPS
REPNZ SCAS	Same as REPNE SCAS
REPZ CMPS	Same as REPE CMPS
REPZ SCAS	Same as REPE SCAS

(1) CX is decremented each time. DI and SI) and up pointing one position (add or) and string is) of past first point of match/mismatch. ZF does not have to be zeroed before using these instructions.

Apéndice G

Hoja de Costos

COMPONENTE	CANTIDAD	PRECIO U.	PRECIO T.
Tarjeta Madre			
8088-2	1	7.45	7.45
8284	1	2.25	1.49
8255	1	1.49	1.49
74ls245	1	0.79	0.79
74ls373	2	0.79	1.58
74ls138	7	0.39	2.73
74ls00	2	0.16	0.32
74ls04	1	0.16	0.16
74ls30	2	0.17	0.34
74ls32	1	0.18	0.18
74ls74	1	0.24	0.24
74ls244	5	0.69	3.45
74ls393	1	0.79	0.79
74ls241	1	0.69	0.69
2716	1	3.95	3.95
27128A-2	1	5.95	5.95
Cristal 14,5674MHz	1	1.95	1.95
Cristal 1,8432MHz	1	2.95	2.95
Diodo	1	0.10	0.10
Cap.Electrolítico 1 uF	1	0.14	0.14
Cap. Tantalio	1	0.21	0.21
Cap. Monolítico	50	0.18	9.00
Resistencias	10	0.05	0.05
			$ 55.18 usd
Tarjeta Serie			
8251	4	1.29	5.16
1488	1	1.05	1.05
1489	1	0.98	0.98
74ls04	1	0.16	0.16
Conector DB9 hembra	4	0.55	2.20
Conector DB25 macho	4	0.45	1.80

Conector			
DB25 hembra	*4*	*0.75*	*3.00*
Cable plano			
20 hilos(pie)	*3*	*0.36*	*1.44*
			$ 19.60 usd

Tarjeta de Memoria

hm256-lp-15	*8*	*12.95*	*103.60*
Conector IDC			
20 posiciones	*3*	*1.29*	*3.87*
Cable plano			
20 hilos(pie)	*1*	*1.05*	*1.05*
			$ 112.39 usd

Tarjeta Paralelo

7407	*2*	*0.29*	*0.58*
Conector			
DB25 hembra	*1*	*0.85*	*0.85*
Conector ICC			
16 posiciones	*2*	*0.69*	*1.38*
			$ 2.59 usd

Totales

Tarjeta Madre	*$ 55.18*
Tarjeta Serie	*$ 19.60*
Tarjeta Memoria	*$ 112.39*
Tarjeta Paralelo	*$ 2.59*
	$ 189.76 usd

Nota: tipo de cambio 19.junio.1989 1 usd = 2485 pesos mexicanos