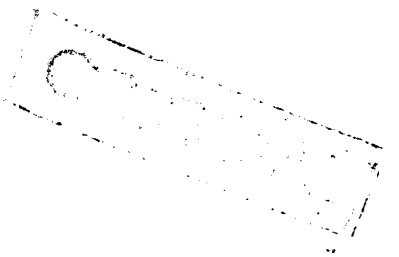
ESCUELA UNIVERSITARIA POLITECNICA DE LAS PALMAS DE GRAN CANARIA



TITULO: DISEÑO DE UN TERMINAL INTERACTIVO INTELIGENTE: EL TERMINAL MACBA-0.

Autor:

Manuel del Castillo Barrios

Tutor:

Sebastian Suarez Gil

PREFACIO

La gran proliferación de computadoras de media y gran capacidad en la sociedad actual ha puesto de manifiesto la necesidad de una comunicación rápida, fiable y cómoda entre el usuario de un sistema y el sistema mismo; no es suficiente la sola existencia de comunicación, sino que se tiende a procesos interactivos cuyo control depende cada vez menos del computador principal.

Es aquí donde el concepto de terminal cobra un significado diferente al clásico, convirtiéndose no sólo en transceptor y controlador de la comunicación, sino también en procesador local de la información comunicada, realizando esta labor independientemente del proceso ejecutado en el computador principal.

En la primera parte de este trabajo veremos algunos conceptos generales relacionados con un terminal, estableceremos las características de un terminal típico, los diferentes tipos de periféricos y de comunicación con el ordenador principal; diferenciaremos los terminales inteligentes de sus homólogos pasivos, haciendo referencia a algunas de las múltiples funciones implementables en aquellos, presentando a continuación el diseño de un terminal de este tipo en su configuración básica, llamando especialmente la atención sobre su bajo costo, flexibilidad y relativamente sencillo desarrollo.

También se desarrollará la estructura del terminal tanto a nivel hardware, analizando cada módulo por separado, como a nivel software, analizando el programa de soporte del conjunto, así como las posibles ampliaciones del sistema.

Agradecimientos:

al departamento de ordenadores,

al laboratorio de ordenadores,

al laboratorio de electronica,

al laboratorio de electronica industrial,

al laboratorio de circuitos impresos,

al sistema de desarrollo INTEL MDS-221,

todos ellos pertenecientes a la E.U.P.,

por la enorme colaboración prestada, sin la cual hubiera sido imposible la total implementación de este proyecto de fin de carrera.

Las Palmas de Gran Canaria, 15 de julio de 1986

INDICE:

TERCERA PARTE: APENDICES.

Apendice I: MANUAL DE USO DEL TERMINAL MACBA-0.

Formato de la pantalla - formato de caracter - caracteres especiales reconocidos - otros caracteres especiales - secuencias de ESCAPE reconocidas - memoria de programa - memoria de pantalla, variables, buffers y stack - monitor CRT - velocidad de transmision - fuente de alimentacion - teclado.

Apendice II: NORMAS DE COMUNICACION.

Norma de interconexion serie RS-232C - norma de interconexion paralelo CENTRONICS - norma para la interconexion digital de instrumentos programables: IEEE-488 - equipos terminales del circuito de datos.

Apendice III: LISTADOS DE SOFTWARE.

Apendice IV: ESQUEMAS GLOBALES.

Apendice V: LISTA DE COMPONENTES.

Apendice VI: BIBLIOGRAFIA.

PRIMERA PARTE

CAPITULO 1

Capitulo 1: GENERALIDADES

En este capitulo trataremos conceptos generales sobre equipos terminales asi como sobre periféricos de entrada y salida de información.

1.1 Composición de un terminal tipicos.

Existen numerosos tipos de terminales, según la aplicación para la que han sido disenados, como:

- cajas registradoras
- cajeros automáticos
- teleimpresoras
- terminales CRT
- ...

Pero todos ellos realizan la misma función básica, que es el servir de enlace entre el ordenador (o computador) principal y el usuario (u operador).

Para realizar esta función, un terminal tipico consta de tres bloques fundamentales, tal como se observa en la figura 1.1:

a) controlador de la comunicación.

b) controlador de dispositivo(s) de entrada.

c) controlador de dispositivo(s) de salida.

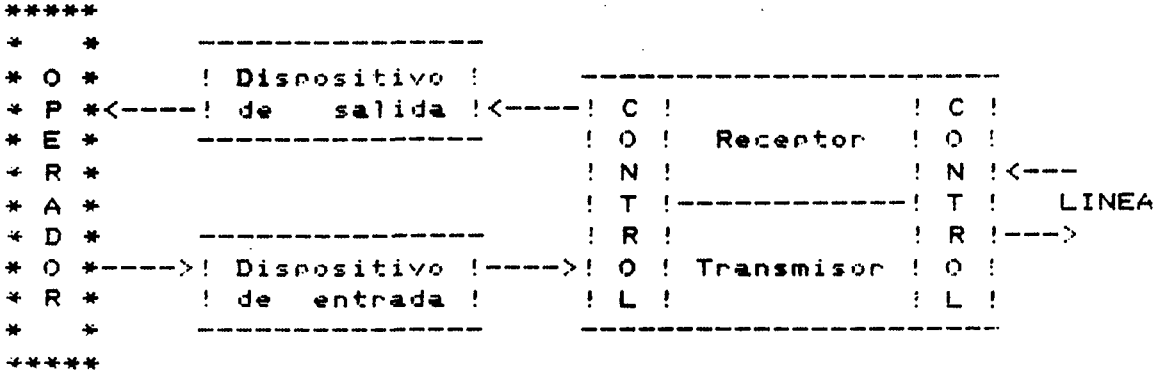


fig. 1.1
- bloques de un terminal tipico -

Así expuesto, el concepto de terminal lleva implícito el de interactividad, aunque el hecho de no ser interactivo no modifica la terminología, ya que existen equipos que funcionan como terminales de sólo entrada (en modo bloque) o de sólo salida (impresoras remotas, etc.). De todas formas, de ahora en adelante consideraremos el concepto de terminal como asociado al terminal interactivo.

La relación existente entre el usuario y el computador depende en gran medida del tipo de periférico de entrada/salida que se utilice, por lo que analizaremos algunos de estos periféricos.

1.2 Periféricos de entrada de información.

Aunque existe un gran número de periféricos de entrada, podemos llegar a establecer una relación de los más usuales, teniendo en cuenta que algunos, si no todos, pueden complementarse entre sí. Así pues, distinguiremos:

a) Teclado.

El teclado representa el método clásico de entrada de datos hacia el computador y, atendiendo a su función, podemos establecer la siguiente clasificación:

a.1 - teclados alfanuméricos, tipo máquina de escribir, que permiten introducir todo tipo de caracteres (letras, números, símbolos, etc.).

a.2 - teclados numéricos, de uso común en pequeños sistemas computadores de aprendizaje y en algunos sistemas de control.

a.3 - teclados de propósito especial, que se diseñan especialmente, de acuerdo con la aplicación a que se destinen, como el teclado de un cajero automático por ejemplo.

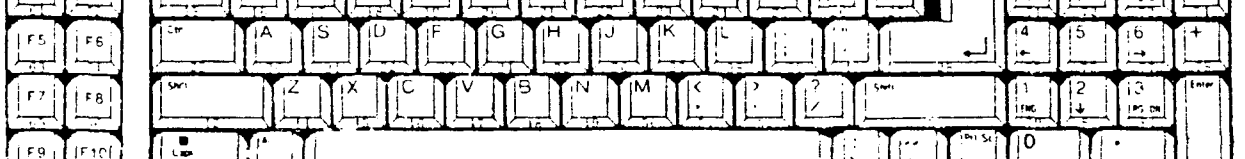
De todas formas, un teclado específico puede incluir elementos de estas tres categorías funcionales conjuntamente, como en el caso del teclado que se observa en la figura 1.2.

b) Digitalizadores de posición.

Como se ha visto en el apartado anterior, el usuario introduce la información directamente en el ordenador pulsando una serie de teclas, pero, como en el caso de un digitalizador, se puede introducir cierto tipo de datos traduciendo la posición de un dispositivo (operado por el usuario) en información digital. Así como el teclado se usa generalmente para la introducción de texto, números, etc., estos digitalizadores se suelen utilizar para enviar al computador datos de tipo gráfico, destacando, por la forma de adquisición de estos datos, los siguientes dispositivos:

b.1 - Tableta digitalizadora.- Su función es permitir la identificación de coordenadas de puntos y líneas existentes en una imagen dada, sea dibujo, fotografía, u otro tipo de gráfico.

b.2 - Lápiz óptico.- Su función es similar a la realizada por la tableta digitalizadora, con la excepción de que la imagen debe estar situada en un monitor (CRT). Una aplicación especial de este dispositivo es la de servir como lector de códigos de barras, sistema este muy generalizado que permite una transferencia directa de los datos codificados sin tener que ser "tecleados" por el operador.



- fig. 1.2 -

b.3 - Joystick.-

Este dispositivo permite identificar una dirección en un plano bidimensional, empleandose normalmente para controlar las coordenadas de objetos en una pantalla de video, para control manual de ciertos mecanismos y, sobre todo, en juegos de computador.

Existen algunos tipos de joystick que permiten la manipulación de coordenadas tridimensionales, pero no resultan tan evidentes como los ya descritos.

c) Micrófono.

Otro de los métodos de introducción de información en el computador es el llamado ASR (Automatic Speech Recognition o Reconocimiento Automático de Voz), cuyo principio de funcionamiento se basa en la comparación de los datos obtenidos en el micrófono y procesados a continuación, con un conjunto de patterns o modelos contenidos en la memoria del sistema. Si ambos coinciden, con un cierto margen de tolerancia, la palabra (o el fonema, dependiendo del tipo de codificación en memoria) es reconocida; si no ocurre esto, es rechazada.

d) Otros.

Existen otros dispositivos de entrada cuyo funcionamiento no implica necesariamente una acción directa y continuada por parte del operador, como:

- transductores
- lectora de tarjetas perforadas
- ...

1.3 Periféricos de salida de información.

Como en el apartado anterior, existe un buen número de periféricos de salida, pudiéndose considerar como más importantes los siguientes:

a) Impresora.

Su función es imprimir letras, números, etc., en papel, para tener una copia permanente de la salida ofrecida por el computador.

Dependiendo de la tecnología y del tipo de impresión podemos establecer la siguiente clasificación:

a.1 - impresoras de caracteres completos.

a.1.1 - impresoras de impacto:

- de barra
- de cilindro
- de bola
- de margarita

(un caracter cada vez)

a.1.2 - impresoras sin impacto:

- por laser
- xerográfica
- fotóptica

(una linea o página a la vez)

a.2 - impresoras de matriz de puntos.

a.2.1 - impresoras de impacto:

- de agujas - un caracter a la vez
- de peine - una linea a la vez

a.2.2 - impresoras sin impacto:

- térmica
- de inyección de tinta

(un caracter c/v)

- electrostática
- electrosensibles

(una linea c/v)

b) Tubo de rayos catódicos (CRT).

Este dispositivo, junto con la lógica asociada, forma el conocido monitor de video, cuyo uso en terminales está ampliamente extendido, proporcionándoles una absoluta capacidad de interacción.

Alrededor suyo se pueden establecer sistemas de identificación de coordenadas por lápiz óptico y por pantallas sensibles al tacto.

Existen varios tipos de CRTs, pero se pueden reducir a dos grupos fundamentales:

b.1 - Monitores para texto y gráficos de baja resolución, cuyo ancho de banda es menor de 10 MHz.

b.2 - Monitores de alta resolución con un ancho de banda mayor que 10 MHz.

c) Plotters gráficos.

Su función es producir una copia permanente de la salida ofrecida por el computador a base de dibujar lineas y curvas, combinándose éstas para formar gráficos, texto, etc.

Se diferencian de las impresoras, además de la tecnología de trazado, en el aspecto de que la copia se produce dibujando, más que imprimiendo las lineas.

Según su configuración se pueden distinguir:

- de una sola plumilla
- de múltiples plumillas
- plotters XY
- de tambor (papel móvil)
- vectoriales
- de rastreo

d) Altavoz.

Aunque la mayoria de los periféricos de salida presenta la información en forma de texto y/o gráficos, también existen equipos capaces de generar una salida audible, cuyo uso se va extendiendo lentamente.

Esta respuesta sonora del computador se puede dar en uno de estos niveles:

d.1 - sonorización de un mensaje pregrabado.

d.2 - música y sintesis de sonidos.

d.3 - sintesis de voz.

Quizas el más interesante de estos niveles es el de sintesis de voz, existiendo abundante literatura al respecto, con una mayor información sobre los equipos y el hardware necesarios para configurar este tipo de respuesta.

e) Disco magnético.

Aunque no se considere estrictamente como un periférico de salida, sino como un dispositivo de memoria auxiliar, lo incluiremos en este apartado debido a que es capaz de realizar una copia de la salida ofrecida por el computador principal independientemente de éste; es decir, en modo local (back-up, protección de ficheros personales, etc.).

En este dispositivo la información se almacena (copia) en la superficie de un disco magnético. Los datos así almacenados se localizan en pistas concéntricas, estando la superficie del disco dividida (por software o por hardware) en sectores, como si se tratase de las porciones de una tarta.

Así, para localizar un bloque cualquiera de datos, sólo es necesario combinar un número de pistas y de sectores, pudiéndose acceder a esos datos de forma pseudo-directa, y no sólo secuencial como en el caso de las cintas magnéticas.

Los discos se clasifican normalmente como duros o flexibles:

e.1 - Disco duro.-

Generalmente están hechos de metal (aluminio) y permiten una relativamente elevada capacidad de almacenamiento, de hasta 300 Mbytes en los más perfeccionados.

Pueden ser sustituibles o no sustituibles.

e.2 - Disco flexible.-

También llamados floppy disks, están construidos con un substrato plástico recubierto de una capa de oxido de hierro, que es la que permite el almacenamiento. Su configuración de cara al usuario es una envoltura en forma de tarjeta cuadrada, existiendo dos formatos standar: el disco de 8 pulgadas y el de 5 1/4 pulgadas, aunque recientemente se ha ido imponiendo, sobre todo en ordenadores personales, el disco de 3 1/2 pulgadas.

Su capacidad de almacenamiento varia entre 72 kbytes y 360 kbytes dependiendo del método de grabación y de la densidad de almacenamiento (en bits/sector y sectores/pista).

1.4 La comunicación con el computador.

La transferencia de información entre un equipo terminal de datos (ETD) y el ordenador (computador) principal se realiza generalmente caracter a caracter utilizando códigos binarios (ASCII p.e.); esta transmisión suele ser secuencial (en serie) debido al coste superior que representaría la misma transmisión en paralelo cuando se supera una cierta distancia.

Estos sistemas de comunicación serie han alcanzado una standarización tal que resulta muy económico utilizarlos incluso en distancias cortas, donde se podría realizar la comunicación en paralelo. Es por ello por lo que hay que prestar una especial atención a este tipo de sistema de transmisión, especialmente al problema de la sincronización del mensaje enviado.

Atendiendo al nivel de sincronismo, distinguiremos:

- sincronismo de bit
- sincronismo de caracter
- sincronismo de mensaje

1.4.1 sincronismo de bit.

El receptor de la información necesita saber exactamente donde empieza y donde termina cada bit en la senal recibida, para efectuar asi el muestreo de la misma en el centro de cada uno de ellos.

Para resolver este problema pueden usarse varios mètodos:

a) enviar por una linea independiente de la de datos una senal de reloj que indique el centro de cada bit de la linea de datos.

b) enviar con cada bit transmitido una información adicional que permita al receptor extraer la señal de reloj.

c) lograr que los relojes de transmisión y recepción se mantengan en fase continuamente.

En el sistema - b - el envio de la información adicional para la determinación del reloj se hace a costa de la disminución de la cantidad de datos transmitidos por unidad de tiempo.

1.4.2 sincronismo de caracter.

Para obtener este sincronismo pueden utilizarse diversos sistemas, unos basados en la utilización de líneas adicionales a las de datos, para enviar impulsos que indiquen el inicio de un bloque de caracteres, y otros basados en la inclusión en el mensaje de algún tipo de sincronismo. Con respecto a este último sistema, podemos distinguir dos métodos muy comunes: el método asíncrono y el síncrono.

a) método asíncrono.-

Con este método, la transmisión se controla por bits de inicio y de final que enmarcan cada caracter transmitido; son los denominados bits de start y de stop, utilizados por el receptor para sincronizar su reloj en cada caracter.

La especificación RS-404 de la E.I.A. (Electronic Industries Asociation) define las características del método asíncrono de transmisión serie:

- cuando no se envian datos por la línea, ésta se mantiene en estado lógico "1".

- cuando se desea transmitir un caracter se envia primero un bit de start, que pone la línea a cero lógico durante el tiempo de un bit.

- a continuación se envian todos los bits del caracter a transmitir con los intervalos que marca el reloj de transmisión.

- después del último bit del caracter se envia (opcionalmente) el bit de paridad, siguiéndole el bit de stop, que hace que la linea pase a uno lógico durante el tiempo de 1, 1.5 ó 2 bits.

Los datos codificados según estas reglas pueden ser detectados fácimente por el receptor.

Como ventaja tiene el hecho de que permite enviar caracteres a ritmos variables, y como desventaja el que sólo un 80% de la información es válida como datos.

Una referencia obligada al utilizar este método es la recomendación RS-232C de la E.I.A., que se puede encontrar en el apéndice II.

b) método sincrono.

En este método en vez de anadir bits de sincronismo a cada caracter, lo que se hace es anadir caracteres de sincronismo a cada bloque de datos.

El sincronismo de bit se consigue normalmente utilizando una linea independiente de reloj, quedando así resuelto.

Con el método asincrono, para enviar un caracter se necesitan 10 bits (sin paridad), mientras que con el método sincrono se envian dos caracteres de más por mensaje.

Una relación simple entre ambos métodos, para un mensaje de "n" caracteres seria:

- en transmisión asincrona.- 10 * n bits

- en transmisión sincrona.- (n + 2) * 8 bits

Para mensajes de menos de 8 caracteres, el método asincrono es más versatil, mientras que para un mensaje de, por ejemplo, 0.5 Kbytes, el rendimiento del método sincrono es un 25% superior al asincrono.

1.4.3 transmisión directa y transmisión via modem.

El medio de comunicación con el computador depende básicamente de la distancia entre éste y el equipo terminal, de forma que

- si la distancia es corta (tipicamente menor que 20-30 metros), la conexión puede ser directa (transmisión serie).

- si la distancia es mayor, es necesario el uso de elementos adaptadores, como los MODEMs, para una comunicación eficiente.

En el caso de utilización de MODEM, conviene conocer las diferentes alternativas existentes, para conseguir asi el máximo aprovechamiento de la linea.

Una referencia a estos dispositivos puede encontrarse en el apèndice II.

1.5 Hacia un mayor proceso local de la información

La evolución de los sistemas computadores exige de éstos una mayor funcionalidad, por lo que se hace evidente que la función básica del equipo terminal (servir de enlace) ha evolucionado hacia una nueva concepción, el terminal "inteligente", capaz de proporcionar un cierto número de funciones especiales independientes de la labor del computador principal.

Sobre esta nueva concepción versará el siguiente capitulo.

CAPITULO 2

Capitulo 2: TERMINAL INTELIGENTE VERSUS TERMINAL PASIVO

En este capitulo realizaremos una diferenciación funcional entre el terminal inteligente y su homólogo pasivo.

2.1 Caracteristicas diferenciadoras.

Atendiendo a la forma de operación de un equipo terminal, podemos distinguir dos tipos perfectamente diferenciados:

a) terminal pasivo, que pudiendo o no estar controlado por microprocesador, sólo se encarga de controlar el intercambio de información entre usuario y computador.

b) terminal activo o inteligente, que es capaz de realizar,además, funciones como:

b.1 - tratamiento de texto local.- Esta función permite gestionar directamente cierto tipo de información, como material editorial (en prensa), facilitando sencillos cambios en los textos, formatos, etc.

b.2 - tratamiento de gráficos.- Esta función permite el procesamiento y visualización de formas y sombras, cambios de escala, rotación de imagen y otras manipulaciones especiales de gráficos.

b.3 - almacenamiento en unidad de discos propia.- Ya que los discos flexibles constituyen uno de los periféricos más interesantes para almacenamiento de información en sistemas microcomputadores, su implementación en un terminal supone la posibilidad de anadir un nivel más de protección a los ficheros del usuario, suprimiéndolos de la memoria central, así como realizar copias de seguridad (back-up) personales, garantizando una mayor independencia del sistema central.

b.4 - control de impresora propia.- Con esta función se consigue una fácil disponibilidad de salida impresa de información, ya que no es necesario "hacer cola" ni tener un horario limitado para acceder a la impresora asociada al computador principal.

b.5 - tratamiento de senal.- Esta función permite configurar un terminal como dispositivo de adquisición, procesamiento y generación de senal analógica, bien directamente, mediante conversores A/D y D/A, o bien mediante instrumentos, usando para ello un GPIB (General Purpose Interface Bus - Bus de interface de propósito general) para interconectar aparatos de medida electrónicos con dicho terminal. Una referencia a uno de estos GPIB, así como a un protocolo de interconexión (el IEEE-488) puede encontrarse en el apéndice II.

2.2 La unidad de control:

- del microprocesador al microordenador.

A la hora de establecer la configuración de un sistema computador como terminal se puede partir de dos ideas diferentes:

a) realizar el diseno desde el nivel físico inferior hacia el nivel funcional; es decir, implementar esta configuración con un sistema específicamente diseñado para ello, eligiendo cada uno de sus elementos constituyentes (microprocesador, memoria, puertos de IO, etc.) y contando con un software también específicamente destinado a esta función.

Esta posibilidad implica la necesidad de unos conocimientos técnicos especializados, tanto en hardware como en software, para conseguir la máxima efectividad del conjunto, debiéndose tener en cuenta también el relativamente alto tiempo de desarrollo del sistema, dependiente en gran medida de los componentes elegidos y de las herramientas de soporte y depurado disponibles.

b) realizar el diseno desde un nivel lógico hacia el nivel funcional; es decir, aprovechar las características que presentan los microordenadores u ordenadores personales para adaptarlos como terminales.

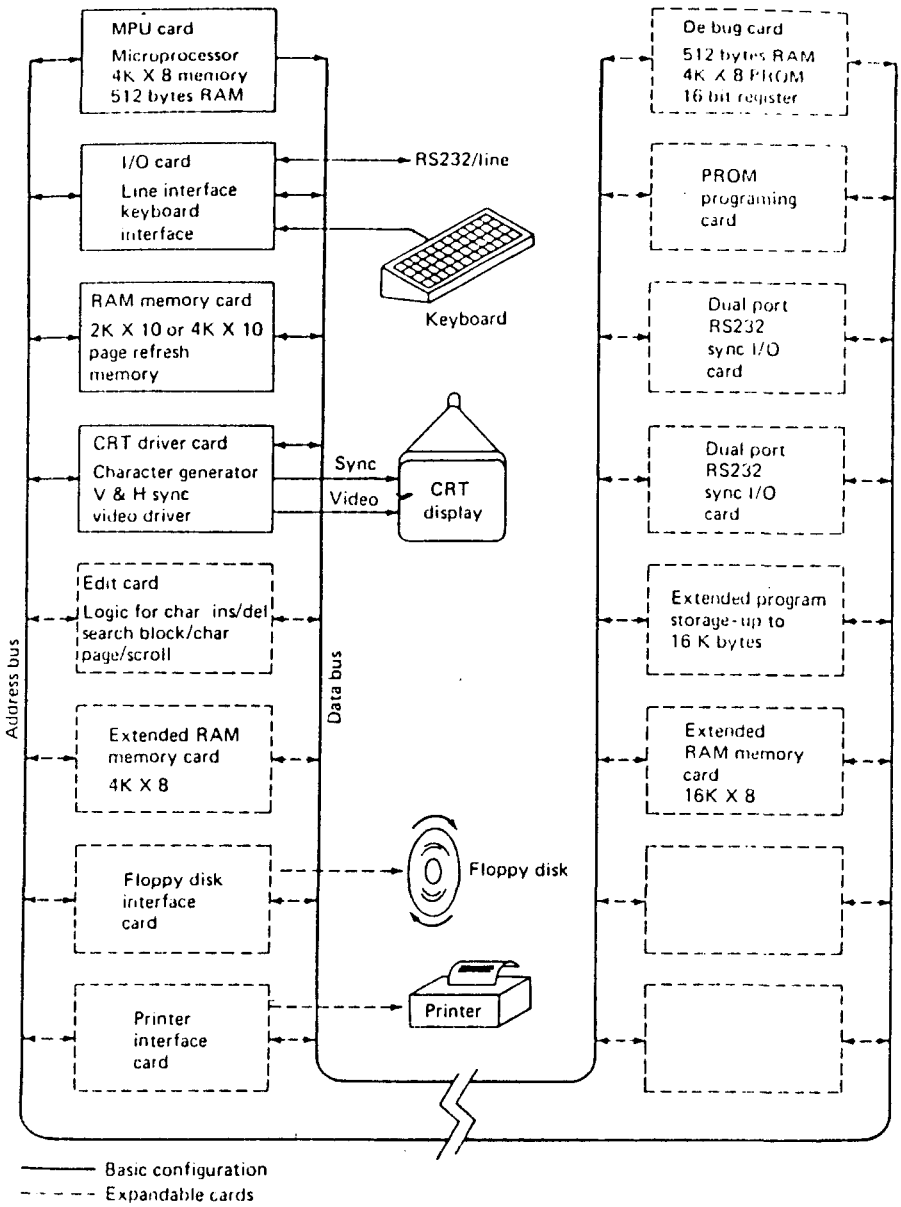
Esta solución ahorra tiempo de desarrollo del hardware y software básicos, y reduce el nivel de dificultad de diseno, pudiéndose realizar la configuración directamente por el usuario (sin grandes conocimientos especializados) por software y en alto nivel, resultando fácilmente modificable la estructura funcional en caso necesario.

Por otro lado, su estructura física es, a priori, más rígida, y normalmente resulta la solución más cara, dependiendo su rentabilidad del propósito final del terminal en cuestión.

2.3 Estructura global de un terminal inteligente.

A modo de resúmen de este capítulo y como introducción a la segunda parte de este trabajo, se puede establecer la estructura global de un terminal de tipo inteligente y, partiendo de la idea de implementación a partir del nivel físico inferior hacia el nivel funcional (apartado 2.2.a), observar la posible evolución de un sencillo terminal hacia un auténtico microordenador (fig. 2.1).

Todo lo expuesto hasta ahora nos permite, pues, acceder a un nivel más técnico del diseno de un terminal, tomando como referencia el desarrollo del terminal MACBA-0.



- fig. 2.1 -

SEGUNDA PARTE

CAPITULO 3

Capitulo 3: CONFIGURACION

Este terminal ha sido pensado para establecer una comunicación asincrona serie con el ordenador principal, bien directamente o bien via MODEM, utilizando el protocolo standar RS-232C, compatible con la norma V.24 del CCITT. La velocidad de transmisión será seleccionable desde 50 bit/s hasta 19200 bit/s, pudiendo ser programable el número de bits de stop, el número de bits/caracter y la inclusión o no de paridad (par o impar).

Como dispositivo de entrada se utilizará un teclado multicompatible, cuyas caracteristicas se pueden observar en el apéndice I, y como dispositivo de salida, un visualizador CRT (monitor) y/o una impresora (conexión centronics), aunque se puede ampliar convenientemente la gama de periféricos con sólo disenar nuevas tarjetas.

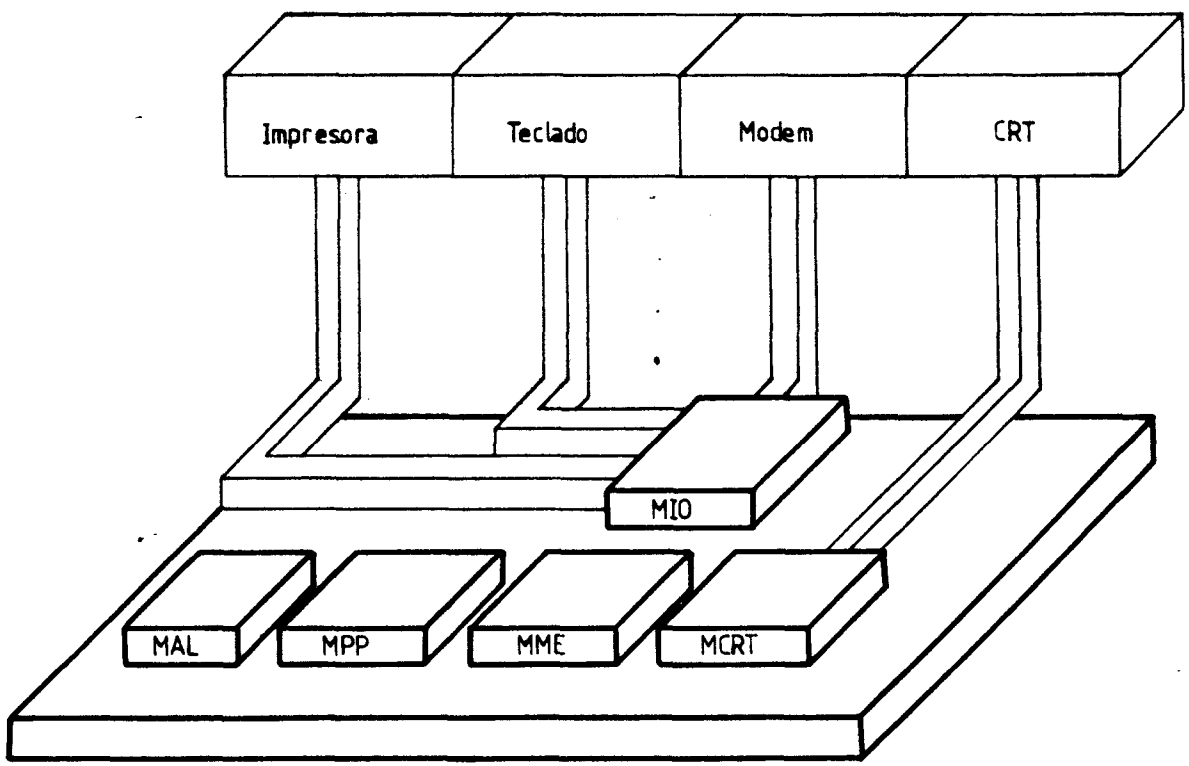
El formato de pantalla es programable (por software, aunque habria que cambiar el cristal del oscilador de video), pero nos ceniremos a uno de 80 columnas por 25 filas, teniendo como formato de caracter (también programable, con la misma salvedad que en el caso anterior) una matriz de 5x7 pixels sobre un campo de 7x10 pixels.

3.1 Configuración básica realizada.

La configuración básica de este proyecto se aprecia en la figura 3.1, en la que se distinguen varios módulo funcionales que analizaremos por separado.

MAL: módulo de alimentación.

Suministra las tensiones y potencia necesarias para el normal funcionamiento del conjunto. Sus caracteristicas se pueden encontrar en el apéndice I.



- fig. 3.1 -

MPP: módulo procesador principal.

Es el encargado del control programático del sistema y está compuesto por dos bloques:

- bloque de CPU
- bloque de transferencias (DMA)

MCRT: módulo de control de CRT.

Es el encargado, bajo el mando del MPP, del control del visualizador CRT, generando adecuadamente las senales de sincronismo, video, acceso a memoria de pantalla, etc.

MIO: módulo de entradas/salidas.

Se encarga de gestionar las entradas/salidas al/del sistema, tanto si se realizan en serie como si se hace en paralelo. Para ello, está formado por dos bloques:

- bloque de comunicación serie
- bloque de comunicación paralelo

MME: módulo de memoria.

Es el encargado del almacenamiento de programas y datos, y está formado por un banco de memoria EPROM y otro de RAM, como se verá posteriormente.

3.2 Configuración de ampliaciones posibles.

A la hora de pensar en ampliar este sistema, las limitaciones vienen dadas más que nada por la imaginación del disenador, ya que con la estructura realizada es fácil adaptar casi cualquier idea. De esas posibilidades se pueden extraer las siguientes (fig. 3.2) a título meramente informativo:

MGR: módulo de gráficos.

Su función consiste en el control de una pantalla gráfica, donde cada pixel se corresponde con uno o más bits de la memoria; este módulo puede ser capaz de generar rectas, arcos, etc., con sólo enviarle la orden correspondiente.

La base de este módulo puede ser (p.e.) el circuito integrado (CI) 82720 de INTEL.

MDIB: módulo de control de unidad de disco.

Equipado con un sistema operativo adecuado se puede conseguir que este módulo gestione una o más unidades de almacenamiento en diskette independiente del ordenador central.

La base de este módulo puede ser alguno de los siguientes CIs:

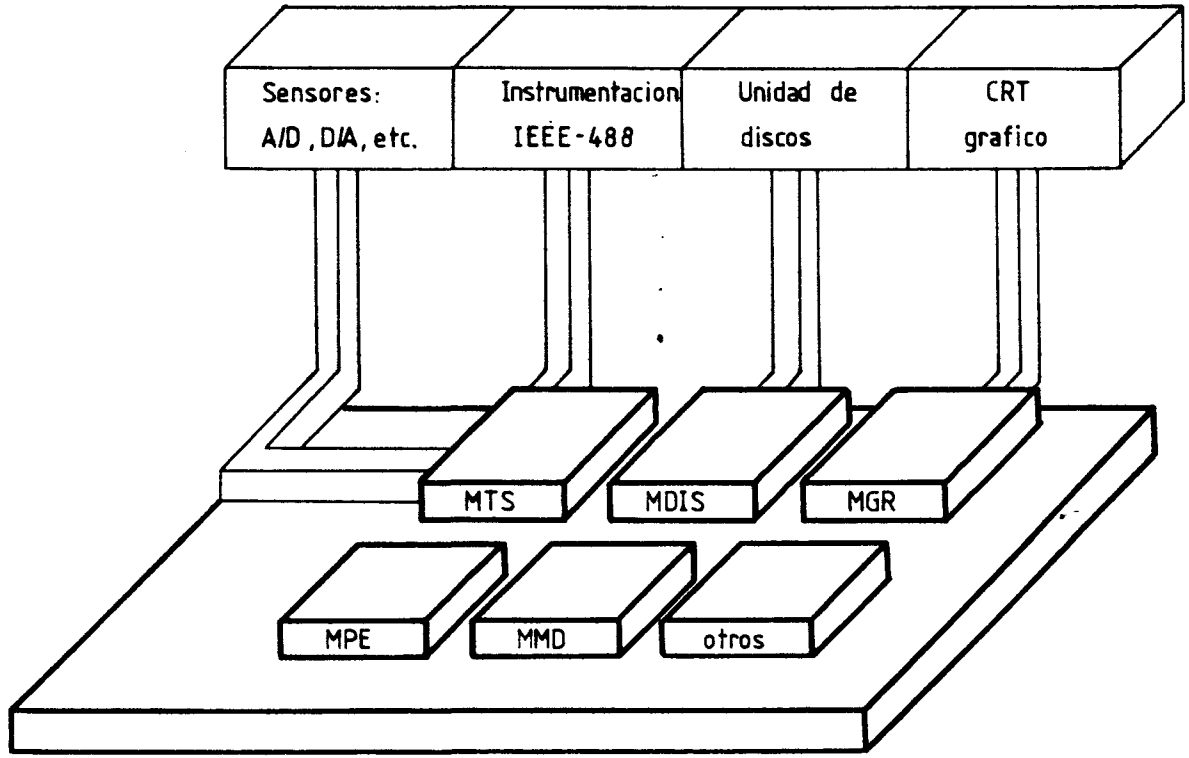
- 8272 de INTEL
- 2797, 1791 de WESTERN DIGITAL

MMD: módulo de memoria dinámica.

Esta módulo se encargaría del refresco de un banco de memoria RAM dinámica, especialmente indicado en ciertas aplicaciones, como puede ser el de memoria de pantalla para el MGR, por ejemplo.

MPE: módulo de programación de EPROM.

Este módulo sería especialmente útil, ya que proporcionaría una herramienta de desarrollo importante y permitiría la programacion no sólo de los modelos típicos de EPROM, sino también PALs, por ejemplo.



- fig. 3.2 -

MTS: módulo de tratamiento de senal.

Este módulo es particularmente interesante debido a la posibilidad de realizar un tratamiento de senal analógica (conversión A/D y D/A) y un interface compatible con la norma IEEE-488, que permite la conexión de instrumentos, como multimetros, osciloscopios, etc., a un controlador.

OTROS:

- unidades RS-232C
- ports de adquisición de datos
- implementación de algún lenguaje en EPROM
- reconocimiento de comandos (órdenes) por voz
- ...

CAPITULO 4

Capitulo 4: ESTRUCTURA A NIVEL HARDWARE

En este capitulo analizaremos de forma más detallada cada uno de los bloques que componen la configuración básica del terminal MACBA-0, así como algunos problemas que se plantean a la hora del diseno.

4.1 Diferentes soluciones a un mismo problema.

Antes de comenzar la descripción de la arquitectura del sistema, conviene destacar tres problemas fundamentales relacionados con el controlador del visualizador CRT y su interface con el bus de dicho sistema:

- el refresco de la pantalla
- la gestión del refresco de la pantalla
- la utilización de la memoria de pantalla

4.1.1 El problema del refresco de la pantalla.

Para que la imagen visualizada sea estable ante los ojos del operador, es necesario refrescar la pantalla del CRT periodicamente, al menos cada 20 ms. (50 Hz.). En este [illegible] debemos transferir todos los códigos de los caract[illegible] visualizables desde la memoria de texto hasta el módulo de control.

Se puede pensar, en principio, en separar completamente la memoria de texto de la memoria accesible por el microprocesador, pero entonces éste no podria operar sobre la misma directamente. Si como memoria de texto se utiliza parte de la memoria direccionable por el microprocesador, son posibles cuatro formas de intercambio de datos entre dicha memoria y el módulo de control de CRT:

a) mediante muestreo.- El microprocesador debe estar testeando continuamente una senal proporcionada por el sistema de control del CRT y, cuando èste se lo indique, extrae el còdigo de un caracter de la memoria de texto y se lo envia a traves de un puerto.

Este sistema es el màs simple, pero el menos adecuado, ya que sòlo se consigue una velocidad de transferencia de unos 1000 caracteres por segundo y, si tenemos en cuenta que el refresco de la pantalla debe realizarse cada 20 ms. al menos, el número màximo de caracteres visualizables serà de 20, cantidad esta muy pequena para nuestro propòsito.

Otro inconveniente es que el microprocesador precisa de todo su tiempo para llevar a cabo el control de la pantalla, no pudiendo asi realizar otras tareas.

b) mediante interrupciones.- Este sistema es anàlogo al anterior, con la diferencia de que los caracteres son pedidos al microprocesador mediante una senal de interrupciòn, con lo que se aumenta la velocidad de transferencia, pero la cantidad de caracteres visualizables sigue siendo pobre, al igual que el tiempo que tiene el microprocesador para otras tareas.

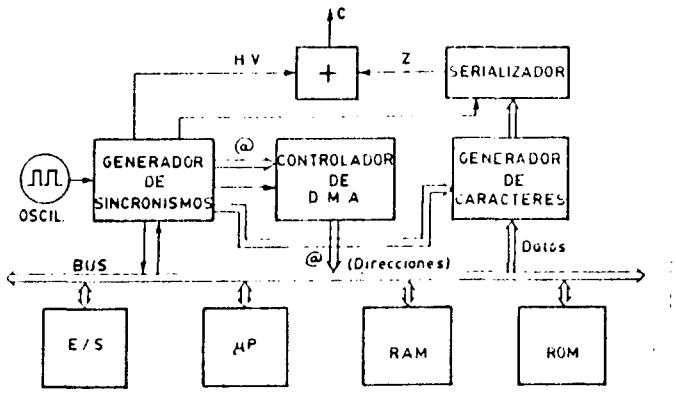
Existen modelos que utilizan un mètodo similar a èste (con variaciones) y consiguen visualizar hasta 80 x 25 caracteres, pero actuan bajo condiciones muy criticas y en un estado de gran rigidez programàtica.

c) mediante acceso directo a memoria.- Utilizando un controlador de DMA en el sistema (fig. 4.1) se puede conseguir una velocidad de transferencia del orden de 100 kbits/s a 1 Mbit/s, con lo que se pueden visualizar más de 2000 caracteres en la pantalla.

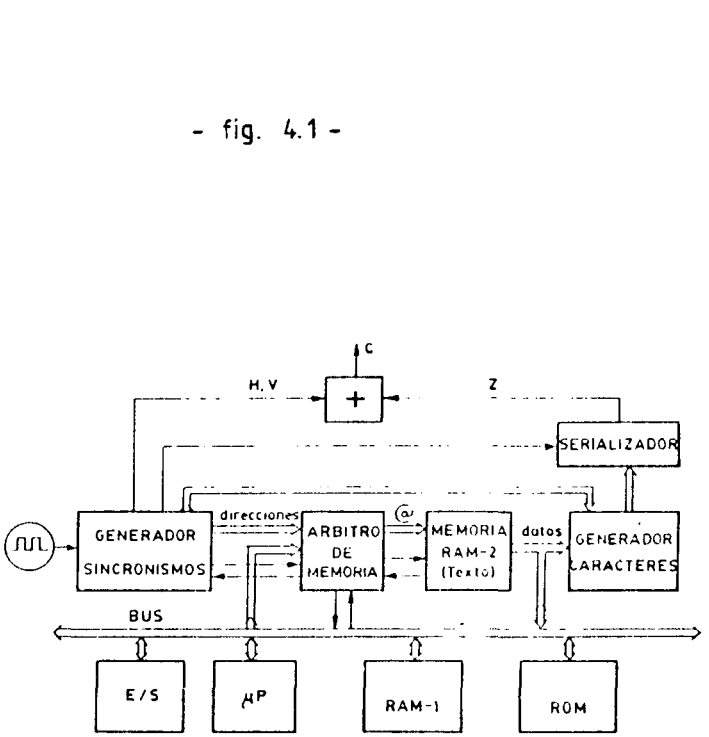
Este sistema sólo exige, para la transferencia de datos al módulo de control, un tiempo del orden del 5% al 33% del tiempo total de CPU (dependiendo del número de caracteres a visualizar), con lo que el microprocesador puede realizar otras tareas, facilitando la flexibilidad del sistema.

d) mediante acceso transparente a la memoria de texto.- En este sistema se multiplexan las direcciones de memoria proporcionadas por el controlador de CRT con las direcciones que proporciona la CPU, de forma que ambos pueden acceder a la memoria de texto de forma síncrona (fig. 4.2).

Así, el acceso a la memoria por parte de uno de los dos sistemas resulta transparente para el otro, no existiendo pérdidas de tiempo por parte de la CPU durante la transferencia. El único inconveniente estriba en la realización del árbitro para el control del acceso a la memoria de texto, ya que la solución no es única. Este sistema recibe también el nombre de acceso a memoria de doble puerto.



- fig. 4.1 -



- fig. 4.2 -

4.1.2 El problema de la gestión del refresco de la pantalla.

En el sistema de control debe existir algún elemento encargado de dar las órdenes oportunas para realizar la tarea de transferencia de información desde la memoria de texto hasta el controlador de CRT en el momento en que se deba refrescar la pantalla.

Esta tarea la puede realizar la CPU o el circuito de control, pero la efectividad del sistema sería diferente en cada caso:

a) gestión mediante la CPU.- Este método resulta imposible de aplicar en el caso de un formato de pantalla superior a 16 x 8 caracteres, debido a que el tiempo que se tarda en ejecutar el programa de control es elevado en relación a los tiempos de refresco de la pantalla.

La única forma de llevar a cabo este método sería utilizando dos líneas de interrupción que indicasen cuando comienza un cuadro y cuando una línea, y aun así, la CPU no tendría suficiente tiempo para realizar las tareas asociadas a los caracteres de control ni las relativas a pretratamiento de caracteres.

b) gestión mediante circuitos especializados.- Este es el único método práctico, y consiste en incluir la gestión del refresco de pantalla como una tarea más del circuito de control de CRT.

Si se utiliza DMA para la transdferencia de información, debe diseñarse un autómata capaz de pedir el bus a la CPU cada vez que deba refrescarse la pantalla, generar las direcciones de memoria donde se encuentran los caracteres a visualizar y extraer los códigos de dichos caracteres.

Si, por el contrario, se utiliza la forma de acceso transparente, se debe diseñar primero el árbitro de memoria y después el circuito capaz de extraer los códigos de los caracteres.

Al utilizar estos sistemas, el tiempo de ciclo del microprocesador ya no influye de forma categórica en el número de caracteres visualizables, y sólo se pierde un pequeño porcentaje del tiempo total.

La utilización de circuitos integrados especializados en el control de visualizadores CRT simplifica notablemente el problema, ya que la gestión del refresco es una función que todos ellos llevan incorporada.

En el caso que nos ocupa, se ha utilizado, como se explicará más adelante, un C.I. especializado y una transferencia de información por DMA (con otro C.I. especializado) para conseguir un equilibrio entre eficacia del sistema y complejidad de realización.

4.1.3 El problema de la utilización de la memoria de pantalla.

Para optimizar la solución al problema de los conflictos entre los diferentes procesos que requieren acceso al bus del sistema se emplean dos técnicas bastante comunes:

- bufferizado de página
- bufferizado de linea

a) bufferizado de página.- Con este método, la totalidad de la memoria de pantalla se aisla del resto del sistema. Este aislamiento viene acompanado del uso de buffers tri-estado o multiplexores de dos a uno, y siempre que un caracter tenga que ser manipulado, la CPU debe ganar el acceso al buffer de memoria y, de nuevo, puede existir conflicto entre la CPU y el controlador de CRT. Este conflicto se resuelve generalmente con el uso de un árbitro de memoria.

Asimismo, esta aproximación requiere una cierta cantidad de hardware adicional, lo que nos lleva a pensar en la segunda solución.

b) bufferizado de linea de caracteres.- Esta aproximación elimina la página de memoria bufferizada pero plantea nuevos problemas que deben ser resueltos.

En este caso, tanto la CPU como el controlador de CRT, comparten la misma memoria. Cada vez que el controlador de CRT necesita un nuevo caracter o conjunto (linea) de ellos, la actividad normal de la CPU es detenida y el controlador de CRT accede a la memoria y visualiza el dato. La forma en que el controlador de CRT adquiere los datos afectará diferentemente a la "performance" del sistema.

La necesidad de que el controlador de CRT tenga acceso a la memoria para adquirir un sólo caracter o una fila completa de ellos depende de la ausencia o presencia, respectivamente, de un buffer de linea (o fila) de caracteres separado.

Si no existiese un buffer de linea, el controlador de CRT deberia acceder a la memoria para extraer cada caracter en cada linea de pantalla (1 linea de caracteres = 7 lineas de pantalla), proceso este no muy eficiente, ya que la CPU se ve forzada a ceder el bus entre un 70% y un 80% del tiempo. Esta inactividad del procesador afecta grandemente a las prestaciones del sistema. Los terminales que usan esta aproximación se ven limitados a una velocidad de comunicación de 1200 bits/seg.

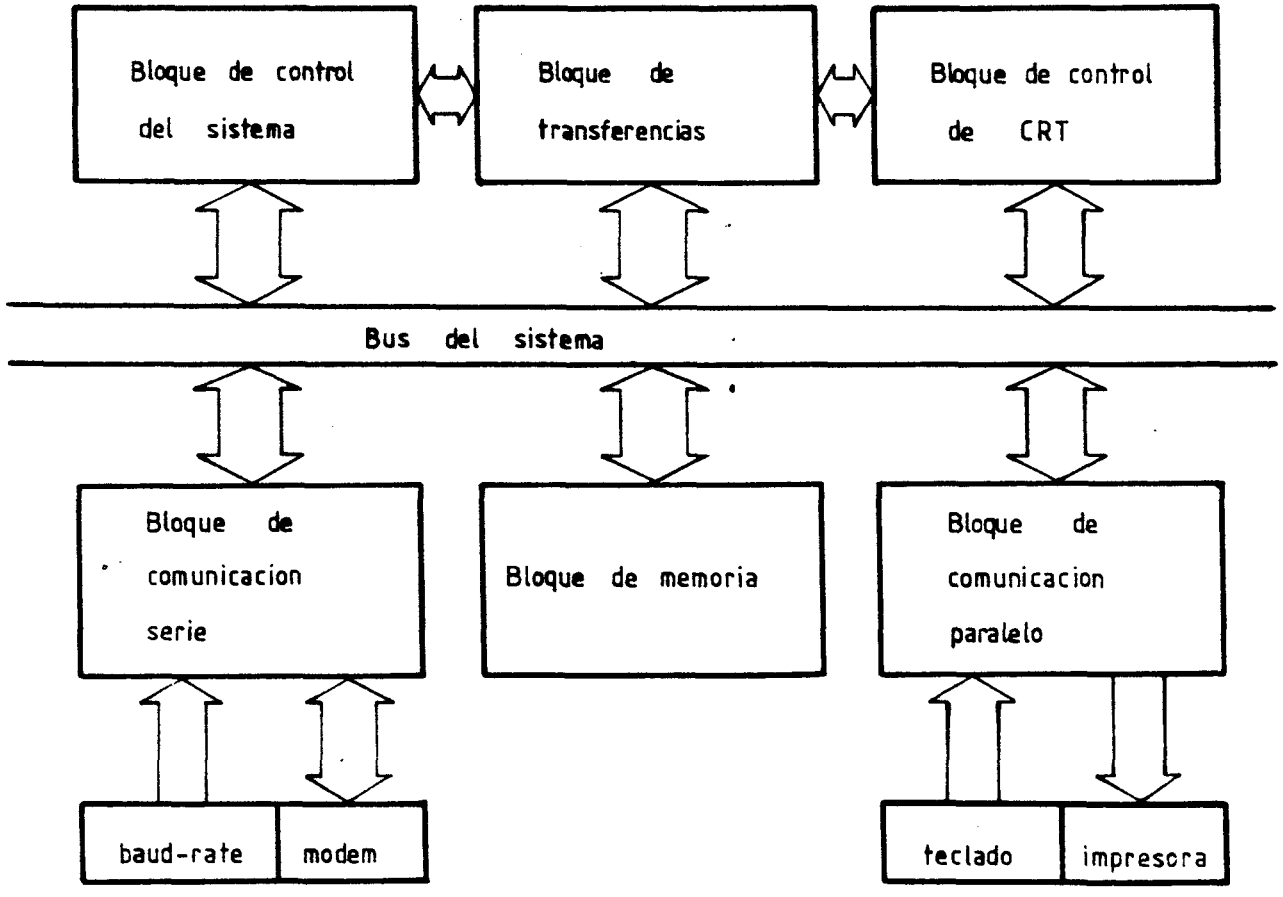
Si se usa un buffer de linea separado, el controlador de CRT sólo tiene que acceder a la memoria una vez por cada caracter y por cada fila de caracteres. Esto hace que la CPU ceda el bus sólo durante un 20% o un 30% del tiempo, pudiendose asi conseguir velocidadeds de hasta 9600 bits/seg.

El C.I. especializado que se ha utilizado es ideal para implementar esta aproximación, ya que el buffer de fila se incluye en dicho C.I.

4.2 Descripción de la arquitectura elegida.

En una primera aproximación, la arquitectura de este proyecto se observa en la figura 4.3, en la que se encuentran los diferentes bloques que componen el sistema.

A continuación analizaremos cada uno de estos bloques por separado.



- fig. 4.3 -

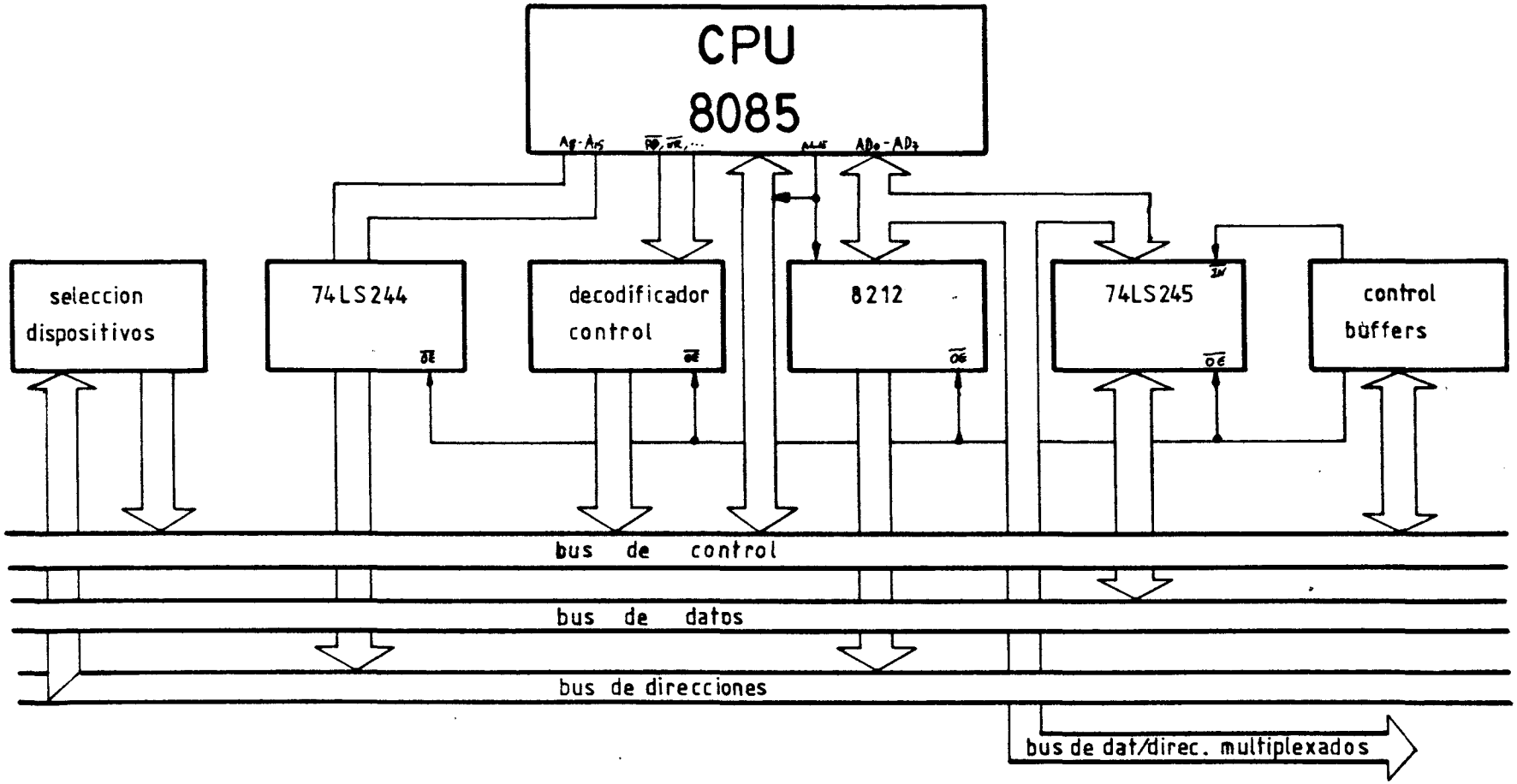
4.2.1 Bloque de control del sistema.

El nucleo de este bloque está formado por el C.I. 8085A de INTEL, que es una CPU completa de 8 bits con capacidad máxima de direccionamiento de 64 Kbytes de forma directa. Usa un bus de datos/direcciones multiplexado, formándose la dirección efectiva con los ocho bits del bus de direcciones y los ocho del bus multiplexado durante el primer estado de un ciclo de máquina. Su estructura interna es apreciable en la figura 4.4.

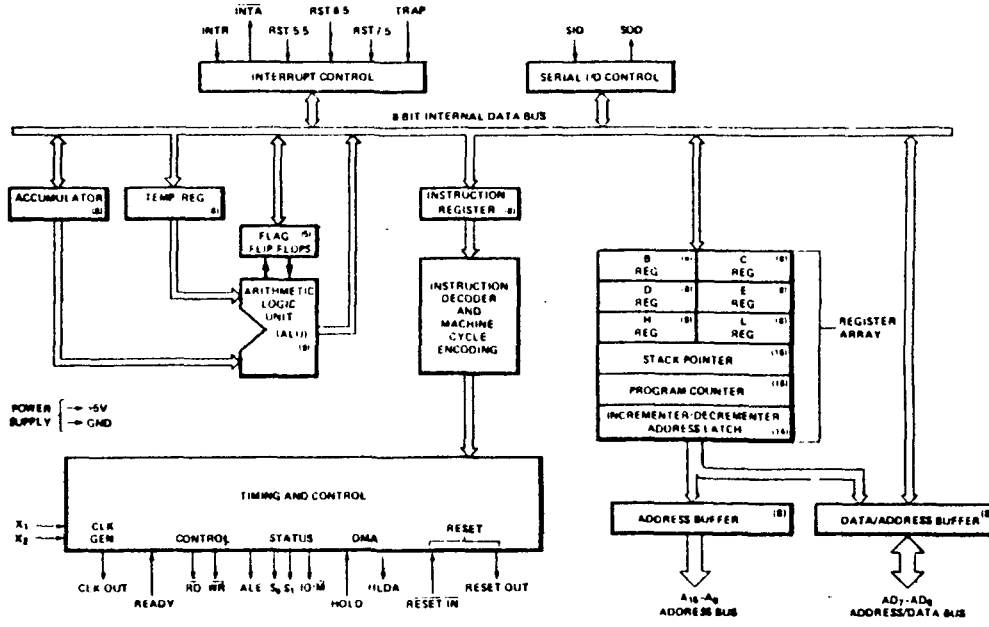
Este microprocesador tiene un ciclo de instrucción tipico de 1.3 microsegundos,lleva incluido el generador de reloj, el controlador del sistema, cuatro interrupciones vectorizadas y una direccionable por software, y además admite aritmètica decimal, binaria y de doble precisión.

La necesidad del multiplexado de direcciones/datos viene dada por la imposibilidad de usar un grupo separado de pins para cada campo de información. Considerando un microprocesador tipico de 8 bits con 40 pins, si de èstos descontamos ocho, que se usan para la alimentación, reset, clock y otras que reflejen condiciones iniciales, el chip se queda con sólo 32 pins para implementar todas las lineas de transferencia de información que se observan en la figura 4.3.

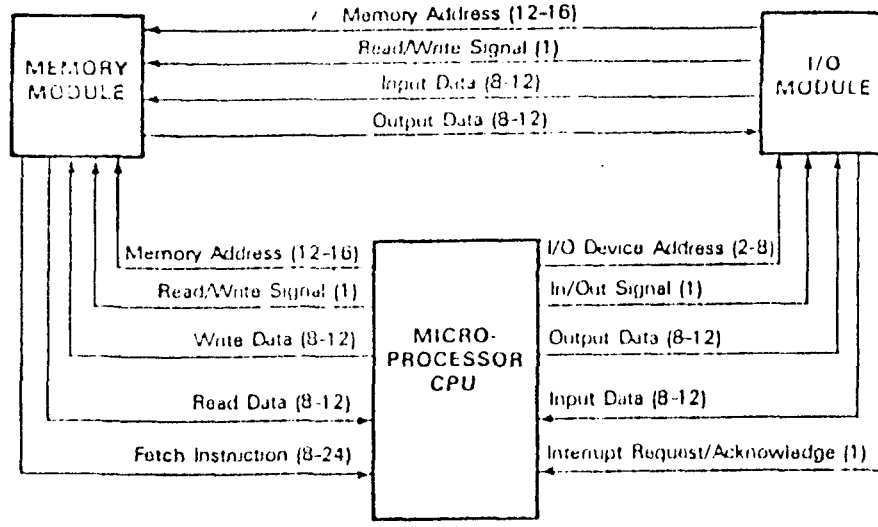
A la hora de interconectar los diferentes elementos de un sistema de control basado en microprocesador, hay que tener especial cuidado al disenar la estructura de buses, por lo que antes de exponer la estructura final conviene aclarar algunas ideas.



- fig. 4.6 -



- fig. 4.4 -



- fig. 4.5 -

a) bus de la CPU.- Este bus está formado por todas las lineas directamente conectadas a las patillas (pins) de la CPU, y, como hemos visto, la CPU puede multiplexar datos/direcciones sobre las mismas lineas.

Debido a que la dirección efectiva en un bus multiplexado no es estable durante todo el ciclo del bus, sólo algunos tipos limitados de dispositivos pueden ser conectados directamente al bus de la CPU. En esas condiciones, el disenador debe garantizar que durante un ciclo de escritura ningún dispositivo pueda corromper la información presente en el bus de datos/direcciones durante el primer estado (T1). Existen varias formas de realizar esta protección, pero no entraremos en su discusión, por no venir al caso.

b) bus del sistema.- Aunque cierto tipo de dispositivos pueden conectarse al bus de la CPU, la mayoria de las implementaciones requieren alguna circuiteria de interface para demultiplexar, bufferizar y decodificar la información existente en dicho bus. El nuevo bus así formado consiste ahora en tres buses no multiplexados (direcciones, datos y control) y se denomina bus del sistema. Conviene tener en cuenta que la colocación de buffers soluciona cualquier problema de carga de los buses.

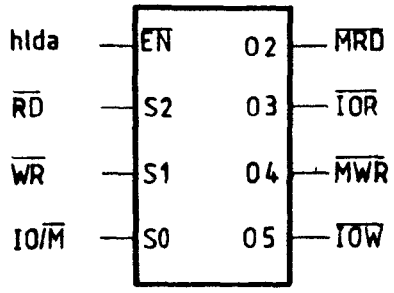
Esta será la estructura que se implemente en este bloque, utilizando un latch para demultiplexar el byte menos significativo del bus de direcciones, un buffer bidireccional para el bus de datos, un decodificador del bus de control, un decodificador para establecer la dirección del buffer de datos y un decodificador de selección de dispositivos periféricos.

La interconexión de todos estos elementos puede observarse en la figura 4.6, así como el desarrollo de los decodificadores y su correspondiente tabla de verdad.

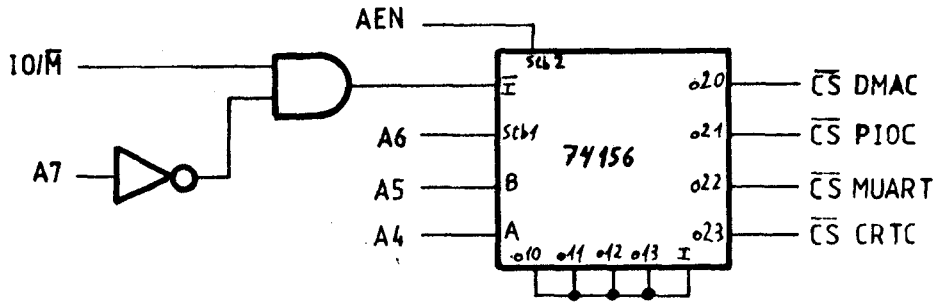
4.2.2 Bloque de transferencias: DMA.

El nucleo de este bloque está formado por el controlador de DMA (INTEL 8257) y un latch de direcciones asociado, tal como se observa en la figura 4.10, pero antes de analizar las características de este controlador conviene aclarar algunas ideas sobre las transferencias por DMA, ya que este método resulta el más eficiente (teniendo en cuenta la relación prestaciones/complejidad) de cara al controlador de CRT, como ya se ha visto.

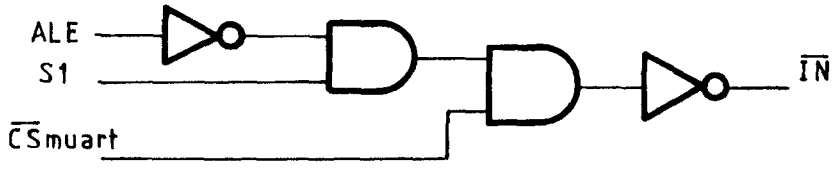
Existen dos métodos básicos de transferencia por DMA, o al menos estos dos son los más comunmente usados en sistemas microprocesadores: el método de parada del microprocesador y el método de robo de ciclo.



RD	WR	IO/M	OUT	-
0	0	0	0	-
0	0	1	1	-
0	1	0	2	MRD
0	1	1	3	IOR
1	0	0	4	MWR
1	0	1	5	IOW
1	1	0	6	-
1	1	1	7	-



A7	A6	A5	A4	IO	AEN	I	OUT
X	X	X	X	X	1	X	-
X	X	X	X	0	0	1	-
1	X	X	X	1	0	1	-
X	1	X	X	1	0	1	-
0	0	0	0	1	0	0	0
0	0	0	1	1	0	0	1
0	0	1	0	1	0	0	2
0	0	1	1	1	0	0	3



ALE	S1	CS	IN
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

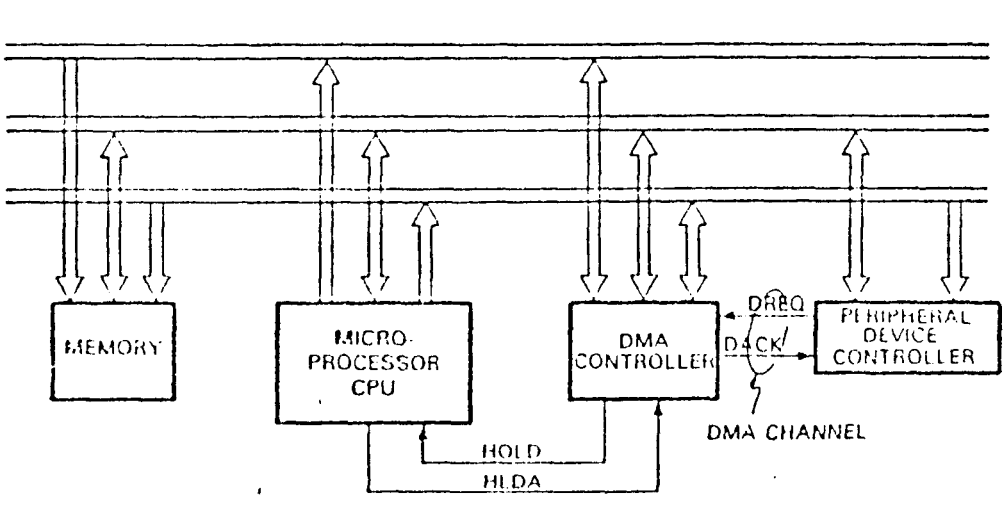
- fig. 4.6 (cont.) -

a) método de parada de la CPU.- Este método constituye la aproximación más simple.

La operación del microprocesador se suspende mientras tiene lugar la transferencia por DMA. El controlador de DMA adquiere entonces el control del bus del sistema para gestionar la transferencia, mientras que la CPU esta en tri-estado y no tiene ningun control sobre su bus. Ambos controladores comparten el tiempo de utilización del bus en unidades de ciclo de instrucción o de ciclo de transferencia.

Consideremos la configuración simplificada de la figura 4.7, en la que un dispositivo periférico está conectado a un canal del controlador de DMA. Cuando el dispositivo requiere el servicio, manda una senal de petición del canal de DMA (DREQ) al controlador de DMA. A continuación éste activa la senal HOLD de la CPU. Esta completa el ciclo en curso, desconectando sus buses e indica esta desconexión activando la senal de reconocimiento HLDA. Con esta senal, el bus es entregado al canal con la máxima prioridad de entre los solicitantes activando la senal de reconocimiento DACK, permitiendo así al solicitante el acceso a memoria.

A partir de aquí el direccionamiento de la memoria es gestionado por el controlador de DMA hasta que se llegue al final del bloque de memoria a transferir. La temporización explicada se puede observar en la figura 4.9.



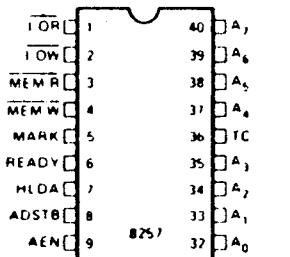
- fig. 4.7 -

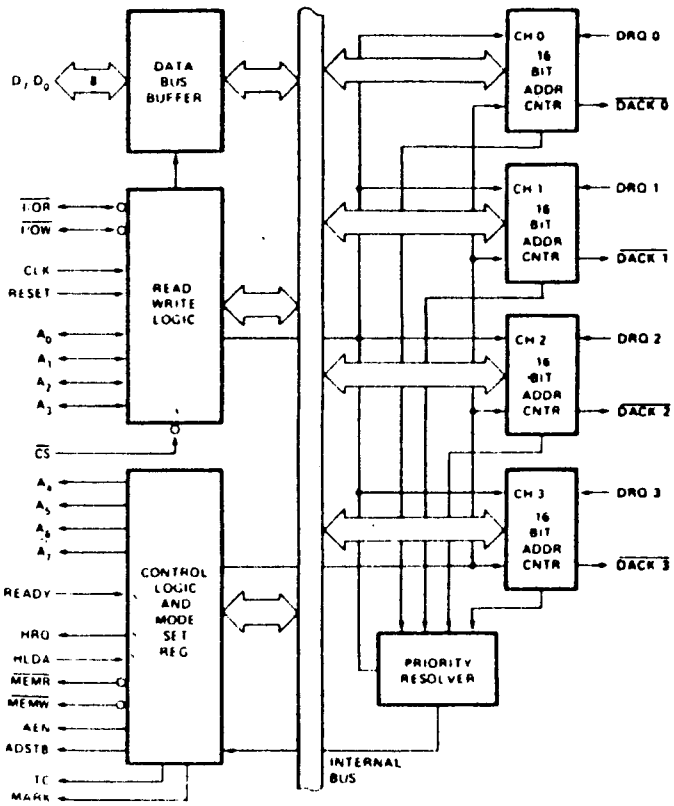
b) método de robo de ciclo.- Mientras que el método anterior se usa normalmente para transferencias de tipo "burst", en las que se transfiere un pequeno bloque de palabras mientras la CPU está desconectada, el método de robo de ciclo sólo transfiere una o dos palabras a la vez, entre ciclos de CPU.

Aunque este método normalmente implica un coste adicional en hardware, se considera el de mejor relación coste/prestaciones, y es el que presenta el menor retraso por transferencia de palabra.

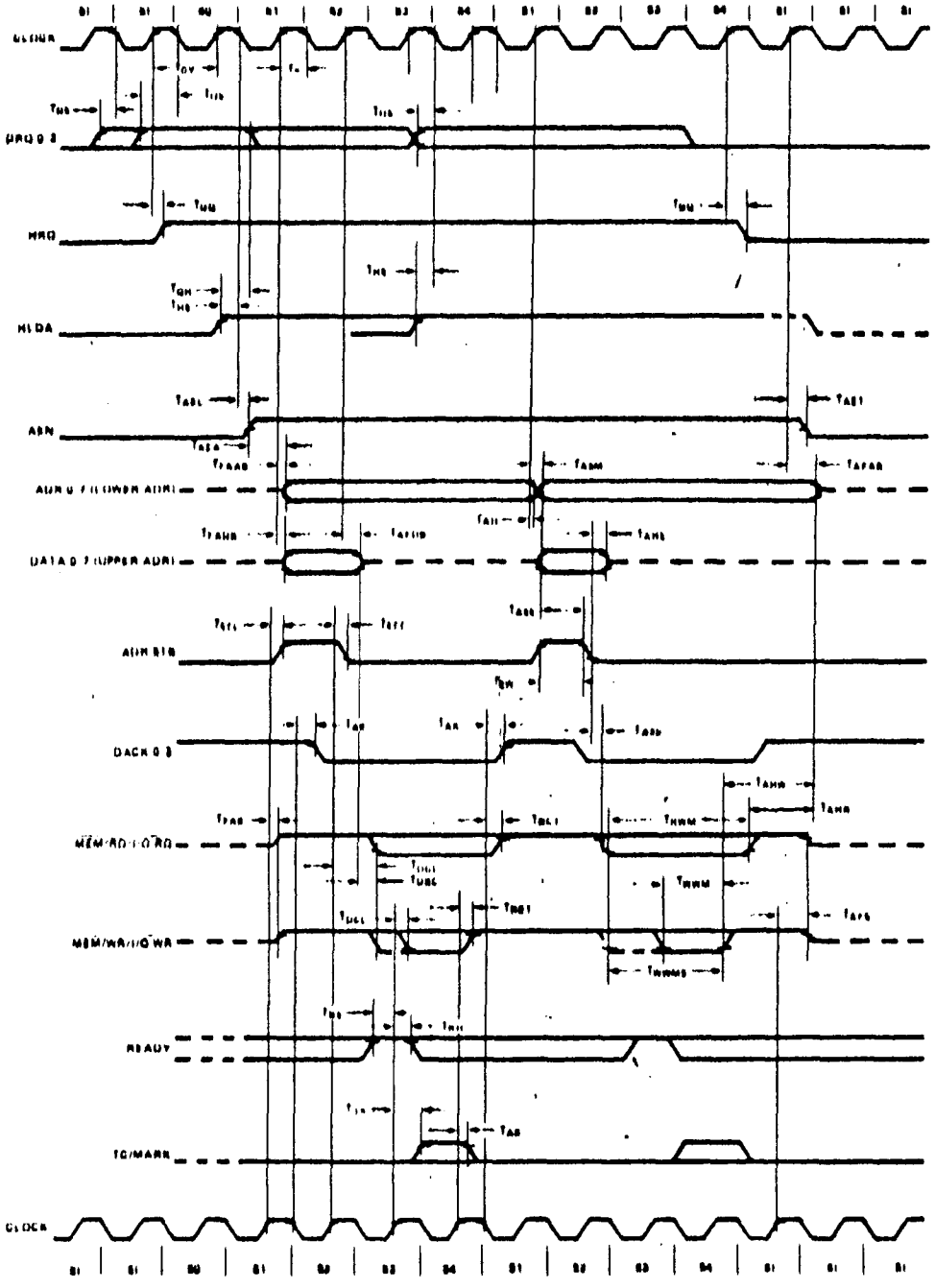
Con este método se usa el bus cuando no lo hace la CPU, transfiriendo los datos con la mínima interferencia en la ejecucion del programa de la CPU. El controlador de DMA gana inmediatamente al acceso a memoria, pero este acceso tiene una duración limitada (de pocos microsegundos), suficiente para la transferencia de una o dos palabras cada vez.

El C.I. integrado utilizado, el 8257, es un controlador de DMA programable que, acoplado a un latch 8212 (p.e.), forma un completo controlador de cuatro canales para ser usado en sistemas microprocesadores INTEL (fig.4.8). Después de haber sido inicializado por programa, este circuito puede transferir bloques de datos de hasta 16 Kbytes de forma directa entre la memoria y un periférico, sin otra intervención de la CPU.





- fig. 4.8 -



- fig. 4.9 -

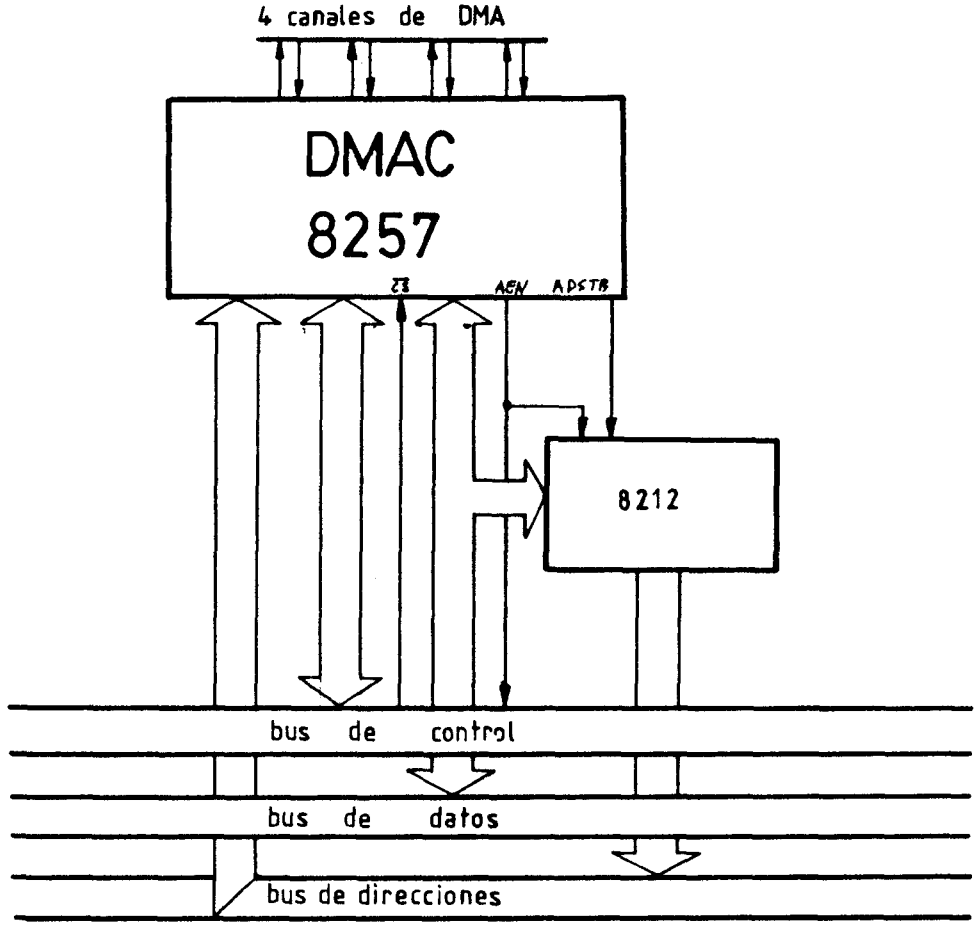
Tras recibir una solicitud de transferencia, el 8257 adquiere el control del bus, reconoce y selecciona el periférico que está conectado al canal de mayor prioridad activado, emite los ocho bits menos significativos del bus de direcciones hacia las lineas A0-A7, los ocho bits más significativos hacia el latch 8212 via bus de datos (que los mantendrá en las lineas A8-A15) y genera las senales de control apropiadas que permitan las funciones tanto de lectura, como de escritura, de forma que el periférico pueda recibir o depositar un octeto directamente desde o hacia la posición determinada de memoria.

Este circuito puede mantener el control del bus del sistema y repetir la secuencia de transferencia mientras el periférico mantenga la solicitud de DMA. De este modo, el 8257 puede transferir un bloque de datos hacia o desde un periférico de alta velocidad en un único tren de impulsos. Cuando se han transferido el número apropiado de octetos, el 8257 activa la senal TC informando de que la operación ha sido completada.

Asimismo, este circito permite tres modos distintos de operación:

- lectura de memoria, que hace que los datos sean transferidos desde la memoria hasta el periférico.

- escritura en memoria, que realiza la operación contraria.



- fig. 4.10 -

- verificación de memoria, que no provoca por si misma una transferencia de datos. Cuando el 8257 genera un ciclo de verificación actua de modo similar al descrito para las operaciones de transferencia, excepto que no se genera ninguna senal de control de memoria ni de entrada/salida. El periférico que provoca esta operación puede así ejecutar procesos de verificación, como puede ser la acumulación del CRC, etc.

Los diferentes bloques en que se divide el circuito son:

- canales de acceso directo a memoria.- El circuito dispone de cuatro canales de DMA, incluyendo cada uno de ellos dos registros de 16 bits; uno de ellos es un registro de dirección de memoria y el otro, un registro contador. En el primer registro se carga la dirección de memoria donde comienza el bloque a transferir, y en el segundo, los 14 bits menos significativos especifican el número de bytes a transferir menos uno; los dos bits restantes especifican el tipo de acceso a memoria.

- registro del bus de datos.- Es de ocho bits, bidireccional y tri-estado.

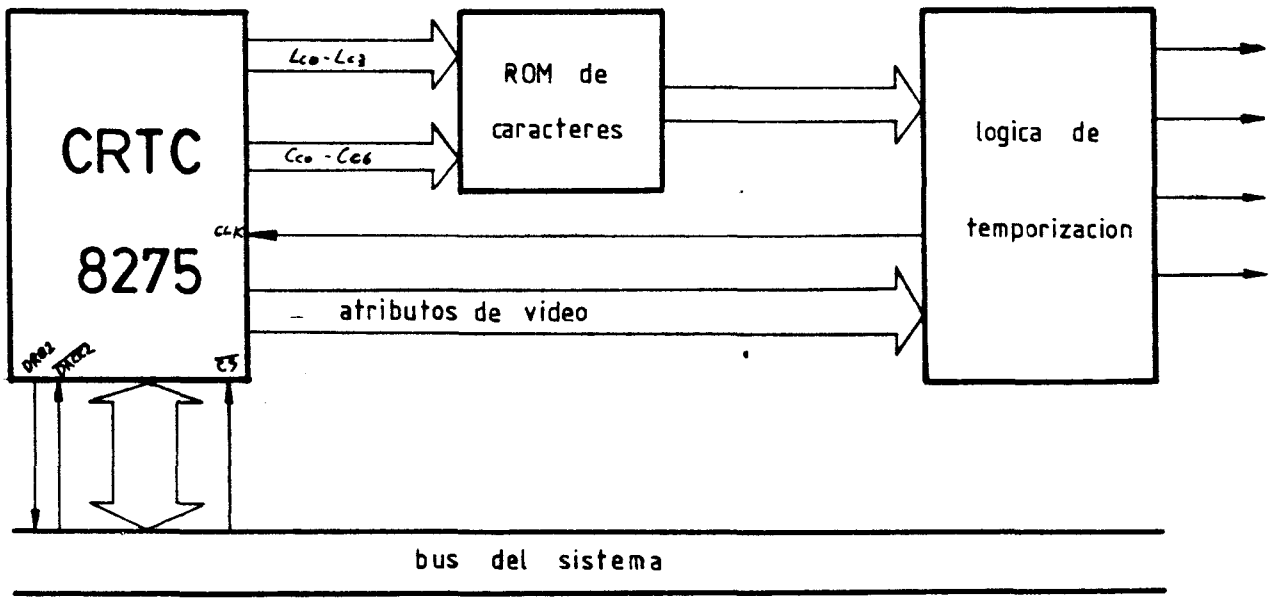
- lógica de lectura/escritura.- Cuando la unidad central está programando o leyendo uno de los registros internos, esta lógica acepta las senales enviadas por la CPU, comportándose como un dispositivo más de entrada/salida. Cuando el 8257 actua como master del bus, esta lógica genera las senales de lectura/escritura necesarias para controlar el enlace entre el periférico que ha solicitado el control y la memoria. Nótese que durante las transferencias DMA los circuitos que no participen de ella deben ser desconectados utilizando la senal AEN, inhibiendo la selección de esos periféricos.

- lógica de control.- Este bloque contiene la secuencia de operaciones a realizar durante los ciclos de acceso a memoria, generando las apropiadas senales de control y los 16 bits de dirección que especifican la posición de memoria que debe ser accedida.

4.2.3 Bloque de control de CRT.

Este bloque se constituye en torno al C.I. 8275 de INTEL, según la estructura indicada en la figura 4.11, pero antes de explicar el circuito y la lógica de interface se hace necesario aclarar algunos conceptos relativos a la generación y representación de la información de video.

La imagen visualizada en el CRT se construye por medio de una serie de lineas que cruzan su superficie. Normalmente, el barrido comienza en la esquina superior izquierda de la pantalla y se desplaza simultaneamente de izda. a dcha. y de arriba hacia abajo, para ocupar así toda la superficie.



- fig. 4.11 -

Son necesarios, pues, tres circuitos de control:

- control de movimiento horizontal
- " " " vertical
- " de intensidad del haz luminoso

Cuando el haz alcanza el final de una linea, es forzado hacia el principio de la linea siguiente a una velocidad mucho mayor que la del barrido en sí. Esta acción se conoce como retrazado. Durante el tiempo empleado para este retrazado la intensidad del haz se anula para no interferir con la imagen visualizada.

Cuando el haz alcanza el final de la pantalla entra en un proceso de retrazado vertical hacia la esquina superior izquierda. El tiempo que tarda el haz en ir desde la parte alta de la pantalla hasta la parte baja, y volver otra vez hacia la parte alta es lo que se conoce como periodo de cuadro. La frecuencia de cuadro standar europea es de 50 Hz. (20 ms./cuadro) y la de barrido horizontal es de 15625 Hz. (64 microsegundos/linea).

Aunque estas frecuencias son las standar en los sistemas comerciales, se puede trabajar con otros valores, dependiendo del ancho de banda del CRT utilizado. Así, existen monitores que trabajan con frecuencias de barrido horizontal desde 18 KHz. hasta 30KHz.

En el caso que nos ocupa utilizaremos una frecuancia de cuadro de 60 Hz. (16.67 ms./cuadro), lo que implica una frecuencia de linea de 15750 Hz. (63.5 microsegundos/linea) y una resolución vertical de 262.5 lineas; no se usará entrelazado de lineas, o barrido en dos campos.

Lo primero que debe hacer cualquier controlador de CRT es generar pulsos que definan la temporización de linea horizontal y la de cuadro vertical. Esto se hace normalmente dividiendo una frecuencia patrón. En la mayoria de los CRTs la frecuencia de barrido horizontal admite una tolerancia de +- 500 Hz., lo que significa que el margen en el que podemos establecer nuestra frecuencia de linea está entre 15250 Hz. y 16250 Hz, o, dicho de otra forma, la resolución vertical puede variar entre 256 y 270 lineas.

Los caracteres que se visualizan en la pantalla están formados por una serie de puntos o pixels que son extraidos del controlador uno a uno y sincronizados con el haz electrónico del CRT. Los circuitos que generan la temporización adecuada se conocen como generador de clock de pixel y generador de clock de caracter. La frecuencia del clock de caracter es igual a la del clock de pixel dividida entre el número de pixels utilizados para formar un caracter en el eje horizontal. El clock de pixel puede calcularse de la siguiente forma:

CLOCK de pixel (Hz.) = (N + R) * D * L * F

donde N es el número de caracteres por fila.

R es el número de caracteres de incremento debido al retrazado.

D es el número de pixels por caracter.

L es el número de lineas por cuadro.

F es la frecuencia de cuadro.

En este diseno, N=80, R=20, D=7, L=270 y F=60, con lo que se obtiene una frecuencia de pixel:

CLOCKp = 11.34 MHz.

El número R puede variar de un sistema a otro, ya que se usa para establecer los márgenes en la parte izda. y dcha. de la pantalla del CRT. En este diseno en particular se encontrò empìricamente un valor òptimo de R igual a 20.

El número de puntos por caracter puede variar dependiendo del generador de caracteres utilizado y de la separaciòn entre caracteres. En este caso, se usa una matriz de 5x7 puntos (x,y) para definir un caracter, dejando dos pixels de separaciòn entre caracteres, con lo que D=7 (fig. 4.12).

El número de lineas por cuadro puede determinarse con la siguiente fòrmula:

$$L = (H * Z) + V$$

donde H es el número de lineas por caracter.

Z es el número de caracteres por cuadro en el eje vertical.

V es el número de lineas empleadas para el retrazado vertical.

En nuestro caso, una matriz de 5x7 puntos se situarà en un campo de 7x10, con lo que H=10. Como se visualizaràn 25 filas de caracteres, Z=25. El número de lineas de retrazado vertical es V=20, lo que nos dà L=270 lineas por cuadro, que entra en el intervalo aceptable establecido anteriormente.

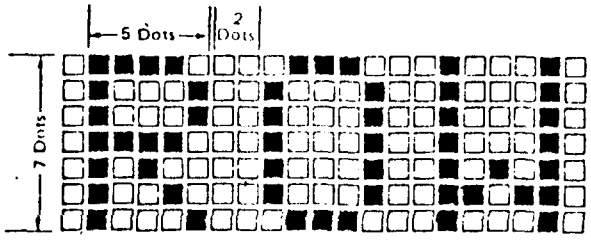
La elecciòn de L=270 implica una frecuencia de barrido horizontal de 16200 Hz., con una frecuencia de cuadro de 60Hz.

El número V se elige para establecer los márgenes en la parte alta y baja de la pantalla. El efecto de retrazado vertical requiere un cierto tiempo, típicamente entre 900-1200 microsegundos. Un número de líneas de retrazado vertical igual a 20, con un período de línea de 61.2 microsegundos hacen un total de 1234.5 microsegundos, tiempo de sobra para que el haz electrónico alcance la esquina superior izquierda de la pantalla.

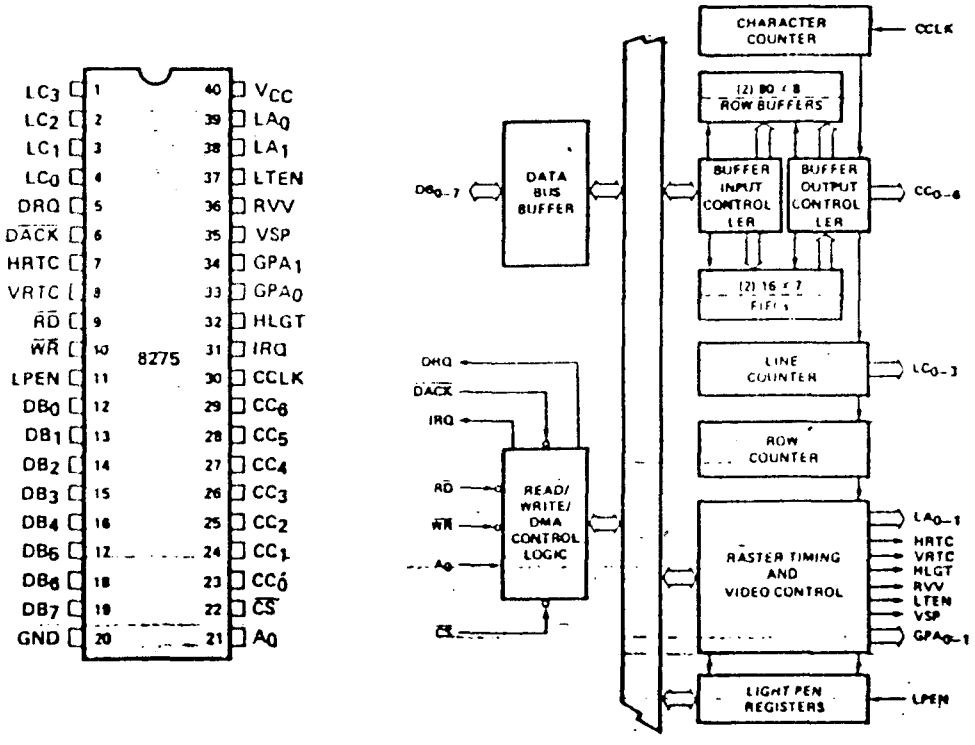
La elección de H y Z depende sólo de las preferencias de diseño. Si H aumenta, el tamaño de carácter a lo largo del eje vertical aumenta. Z es simplemente el número de filas de caracteres que se visualizan, constituyendo claramente una opción de diseño.

El diagrama de bloques y la configuración de patillas del 8275 se muestran en la figura 4.13, siendo la siguiente una descripción de su capacidad.

El 8275, habiendo sido programado previamente para un formato específico de pantalla, genera una serie de señales de petición de DMA que permiten la transferencia de una fila de caracteres desde la memoria de pantalla hasta los buffers del 8275. Este C.I. presenta los códigos de los caracteres al generador externo de caracteres por medio de las líneas de direccionamiento CC0-CC6. La lógica exterior de temporización de pixel se usa entonces para transferir los datos de salida del generador de caracteres, en serie, a la entrada de vídeo del CRT. La fila de caracteres se visualiza línea a línea, usándose las salidas LC0-LC3 para seleccionar la línea en curso. Este proceso se repite para cada fila.



- fig. 4.12 -



- fig. 4.13 -

Al principio de la última fila visualizada, el 8275 activa la línea de interrupción IRQ; esta línea se conecta normalmente a una entrada de interrupción del procesador de control del sistema.

La interrupción hace que la CPU ejecute una subrutina que reinicializa los parámetros del controlador de DMA para el siguiente ciclo de refresco de la pantalla.

Un apropiado refresco del CRT requiere que ciertos parámetros del 8275 sean programados antes del comienzo de la generación de imagen estable. El 8275 tiene dos tipos de registros de programación, los registros de comando (CREG) y los registros de parámetros (PREG); también tiene un registro de estado (SREG). Los registros de comando sólo pueden ser escritos y el de estado sólo puede ser leído.

El 8275 espera la recepción de un comando seguido de una secuencia de 0 a 4 parámetros, dependiendo del tipo de comando. El juego de instrucciones del 8275 está compuesto por ocho comandos, que son los siguientes:

COMANDO	No. de param.	NOTAS
RESET	4	Parámetros de formateado.
START DISPLAY	0	------
STOP DISPLAY	0	------
READ LIGHTPEN	2	Parámetros de lectura.
LOAD CURSOR	2	Coordenadas X e Y
ENABLE INTERRUPT	0	------
DISABLE INTERRUPT	0	------
PRESET COUNTERS	0	Borra contadores internos.

Para establecer el formato de la pantalla, el 8275 proporciona un número de formatos programables, que pueden ir de 1 a 80 caracteres por fila, de 1 a 64 filas por cuadro y de 1 a 16 lineas horizontales por fila.

Además de la transferencia de caracteres desde la memoria hasta el CRT, el 8275 proporciona la función de control del cursor, pudiendose elegir cuatro formatos de éste: linea de subrayado (parpadeante o no) y bloque inverso (parpadeante o no).

El 8275 proporciona dos salidas de temporización, HRTC y VRTC, que se utilizan en la sincronización de los osciladores vertical y horizontal del CRT con el ciclo de refresco del 8275. Además, cuando cualquioera de estas dos senales está activada, también lo está una tercera salida, la VSP (supresión de video). Con esto la lógica de temporización de pixel inhibirá la salida de video durante el tiempo que esté activada VSP. Otra salida, LTEN se usa para forzar la salida de video hacia un estado de luminosidad, independientemente del estado de VSP. Esta característica se usa para colocar el cursor en la pantalla y controlar los atributos visuales.

La salida HLGT facilita el atributo de incrementar la intensidad del haz luminoso, pero no ha sido utilizada. La última senal de temporización es RVV, que invierte, cuando está activada, la senal de video.

Los atributos visuales son códigos especiales que, leidos desde la memoria de pantalla, afectan a las características visuales de un caracter o bloque de ellos.

Los códigos de atributo de caracter pueden ser usados (no en este diseno) para generar simbolos gráficos directamente. Esto se logra activando selectivamente las lineas de atributo (LA0-LA1), la linea de supresión de video y la salida LTEN.

Los atributos de bloque son códigos de control que afectan a los caracteres visuales de un bloque de caracteres, comenzando con el caracter siguiente al código hasta e incluyendo el caracter que precede al siguiente código de control o hasta el final del cuadro o frame.

Existen seis atributos de bloque:

a) Parpadeo.- Los caracteres afectados parpadean debido a la activación periodica de la linea VSP. La frecuencia de parpadeo es igual a la de refresco de la pantalla dividida por 32 (F=1.875 Hz.).

b) Alta luminosidad.- Se activa la linea HLGT.

c) Video inverso.- Los caracteres afectados aparecen en video inverso debido a la activación de RVV.

d) Subrayado.- Los caracteres afectados son subrayados debido a la activación de LTEN.

e) De propósito general.- Hay dos salidas adicionales del 8275 que actuan como atributos de bloque de propósito general, programables independientemente. En este diseno se usan para seleccionar uno de los dos juegoa de caracteres disponibles.

El 8275 puede ser programado para proporcionar códigos de control visibles o invisibles. En el primer caso, todos los códigos de control ocupan una posición en la pantalla, apareciendo como espacios en blanco. En el segundo caso, los buffers FIFO (first in - first out) de fila del 8275 son activados y se coloca, en la posición del código de control, el siguiente caracter, actuando el atributo correspondiente.

El 8275 posee una entrada para lápiz óptico, directamente accesible en este diseno, que permite establecer las coordenadas de la pantalla en la que se ha situado este lápiz óptico. La verificación de la exactitud de las coordenadas debe hacerse por software.

Asimismo, existen cuatro códigos especiales que pueden ayudar al disenador a reducir memoria, software o exceso de transferencias inútiles por DMA. Estos códigos se colocan como si de otro caracter se tratase, y son:

a) fin de fila.- Activa la senal VSP hasta el final de la fila correspondiente.

b) fin de fila,parada de DMA.- Es similar al anterior, pero además detiene la transferencia por DMA hasta el final de la fila.

c) fin de pantalla.- Activa VSP hasta el final del cuadro.

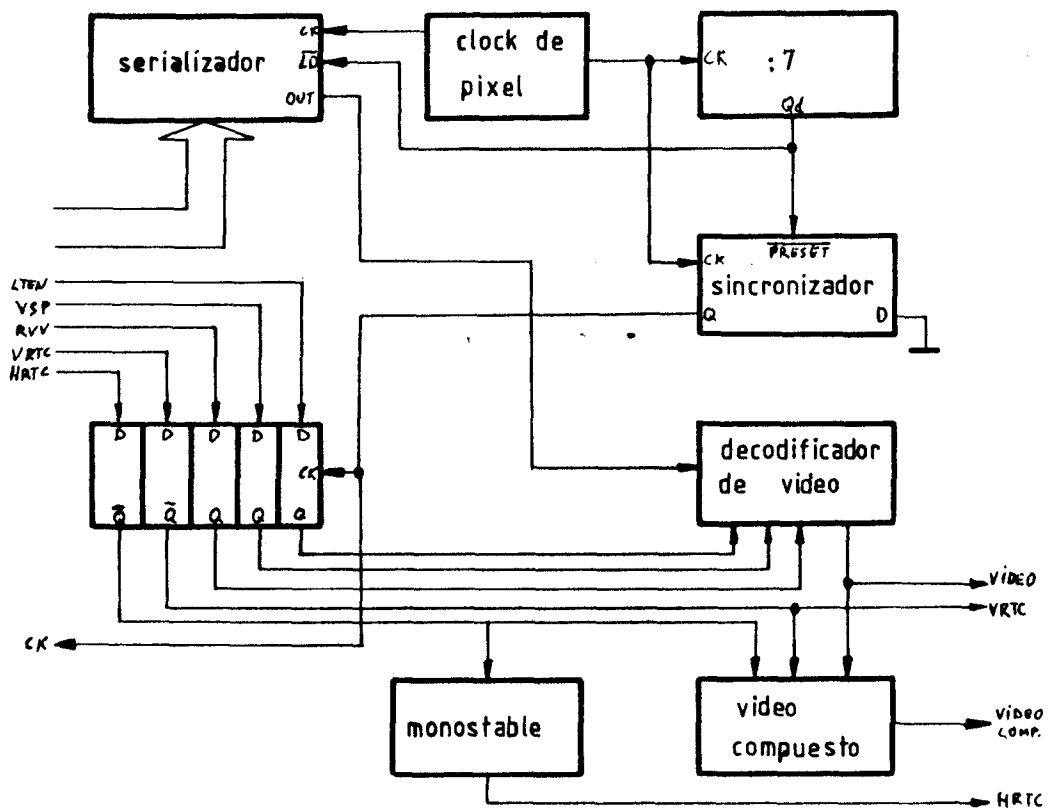
d) fin de pantalla, parada de DMA.- Similar al anterior, pero detiene la transferencia por DMA hasta el final del cuadro.

El 8275 puede ser programado también para requerir transferencias por DMA de un sólo byte o por "burst's" o pequenos bloques de 2, 4 u 8 caracteres. El intervalo entre bloques es, también, programable, lo que permite un diseno a medida.

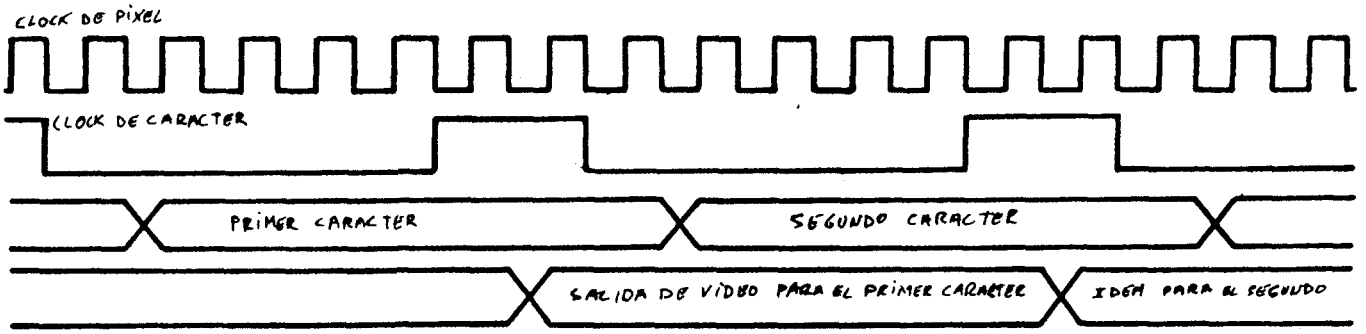
En la figura 4.11 se puede ver la estructura básica de este bloque, estructura esta que se ve ampliada, en lo que a lógica de temporización se refiere, en la figura 4.14.

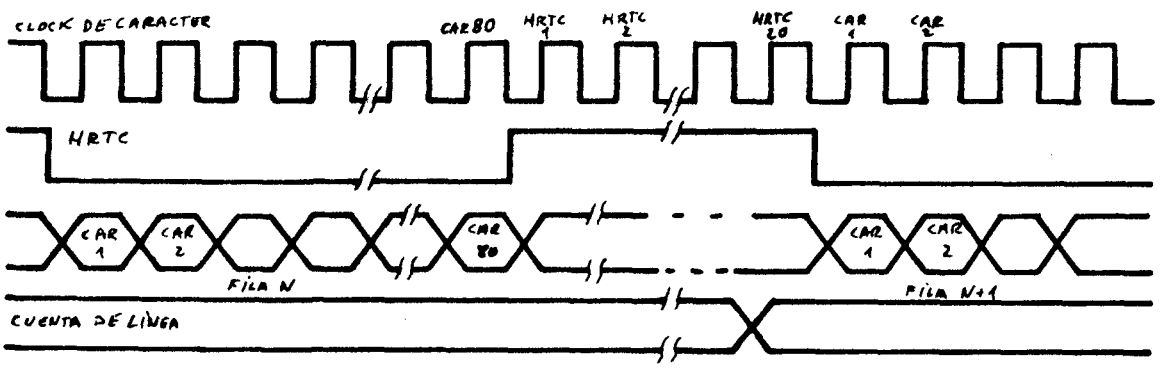
Los timings correspondientes a esta circuiteria se pueden observar en la figura 4.15 y 4.16, así como el decodificador de video y su tabla de verdad.

El circuito de generación de video compuesto es típico y no necesita mayor comentario.

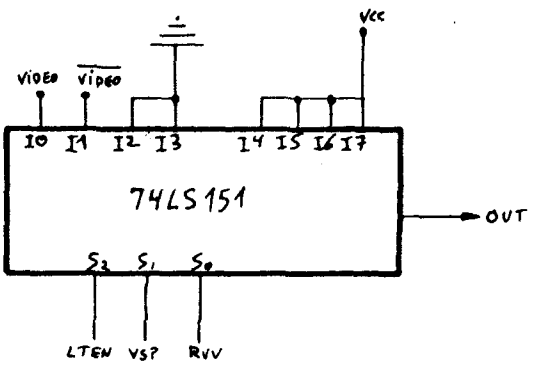


- fig. 4.14 -





- fig. 4.15 -



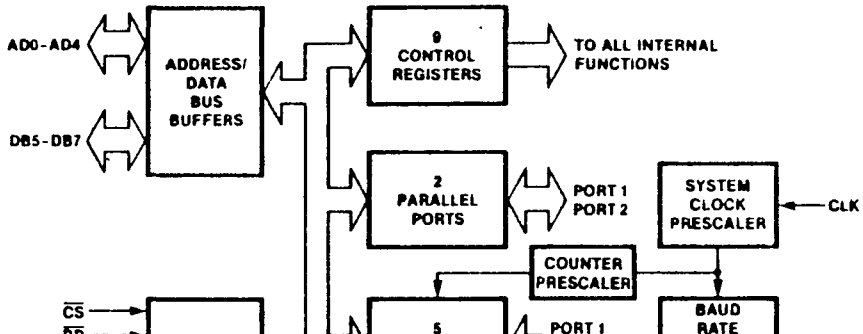
LTEN	VSP	RVV	OUT
0	0	0	video
0	0	1	video
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- fig. 4.16 -

4.2.4 Bloque de comunicación serie.

Aunque este bloque se ha denominado de forma genérica como de "comunicación serie", esta función es sólo una parte de las desarrolladas por él. Ello es debido a que el nucleo de este bloque está formado por el C.I. 8256 de INTEL, denominado MUART (Multifunction Universal Asinchronus Receiver Transmiter - Transmisor receptor asincrono universal multifunción), cuya estructura se observa en la figura 4.17, que combina en un sólo chip de 40 pins cinco funciones usadas comúnmente. Esta MUART permite comunicaciones serie asincronas, entradas/salidas en paralelo, temporizadores, contadores y control de interrupciones.

a) comunicaciones en serie.- La porción de MUART dedicada a la comunicación en serie contiene un transmisor-receptor asincrono capaz de operar en full-duplex (UART). Se incluye también un generador programable de velocidad de transmisión (baud-rate), que permite una gran variedad de valores (desde 50 bits/s. hasta 19200 bits/s. con el generador interno y hasta 1 Mbit/s. con un clock exterior) sin componentes externos. La UART puede ser programada por la CPU para una variedad de tamanos de caracter (5,6,7 u 8 bits), generación y detección de paridad, detección de error y gestión de bits de start/stop. El receptor testea cada bit en su centro, pero esto es modificable por programa. También existe la posibilidad de detección de "breaks"; el transmisor puede enviar "breaks" y ser controlado por uno de los pins exteriores.

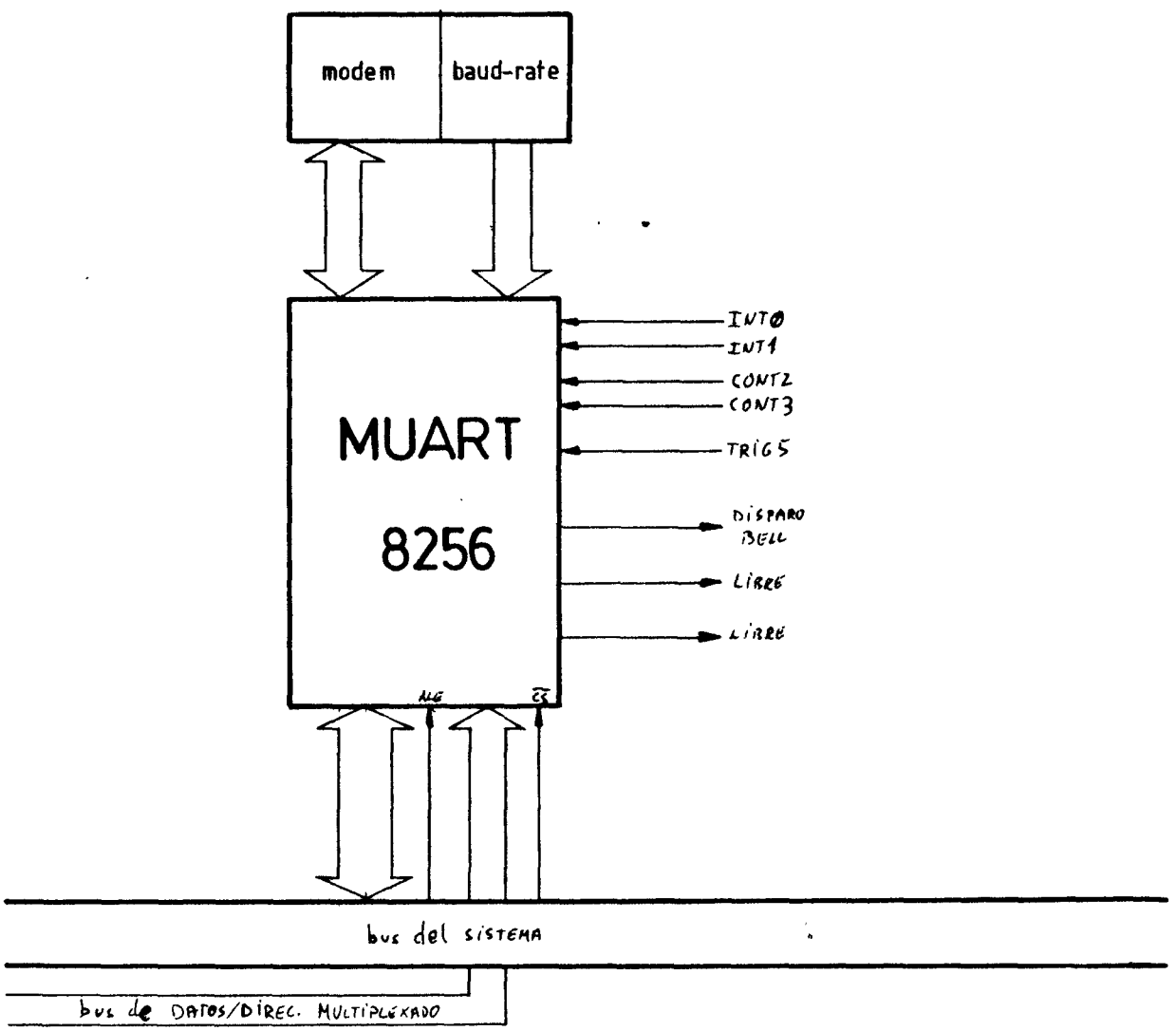


- fig. 4.17 -

b) Entradas/salidas en paralelo.- La MUART incluye 16 bits de entrada/salida en paralelo de propósito general. Ocho de ellos (port 1) pueden ser programados individualmente como entrada o salida, o ser usados para otras funciones especiales. Los otros ocho bits (port 2) pueden ser usados como nibbles (conjunto de 4 bits) o como bytes. Estos ocho bits también incluyen la capacidad de "hand-shaking" usando dos pins del port 1. En nuestro caso, se usa el port 2 como un nibble de entrada (para selección de baud-rate) y otro de salida, quedando dos de estos bits de salida disponibles en el circuito para cualquier propósito, usando para ello la variable MPORT2$BUF del programa (fig. 4.18).

c) contadores-temporizadores.- Hay cinco contadores/temporizadores de ocho bits en la MUART. Los temporizadores pueden ser programados para usar una base de tiempos de 1 KHz. o de 16 KHz., generada desde el clock del sistema (seleccionable). Cuatro de los contadores/temporizadores pueden ser conectados en cascada para formar dos de 16 bits, y el otro puede ser reinicializado por una senal exterior.

d) controlador de interrupciones.- Con esta MUART se puede configurar un controlador de interrupciones de ocho niveles, tanto si se trata de interrupciones anidadas como si son de servicio normal. Siete de estas ocho interrupciones sirven a funciones de la MUART (contadores, temporizadores, UART y/o una interrupción exterior programable) y se proporciona una interrupción exterior de propósito general.



- fig. 4.18 -

La MUART soporta sistemas 8085 y 8086/88, con interrupción vectorizada directa o mediante el método de "polling" para determinar la causa de dicha interrupción.

Las funciones programables se pueden apreciar en la figura 4.18, pero para una información más detallada se recomienda consultar la bibliografía indicada al final del trabajo.

4.2.5 Bloque de entradas/salidas en paralelo.

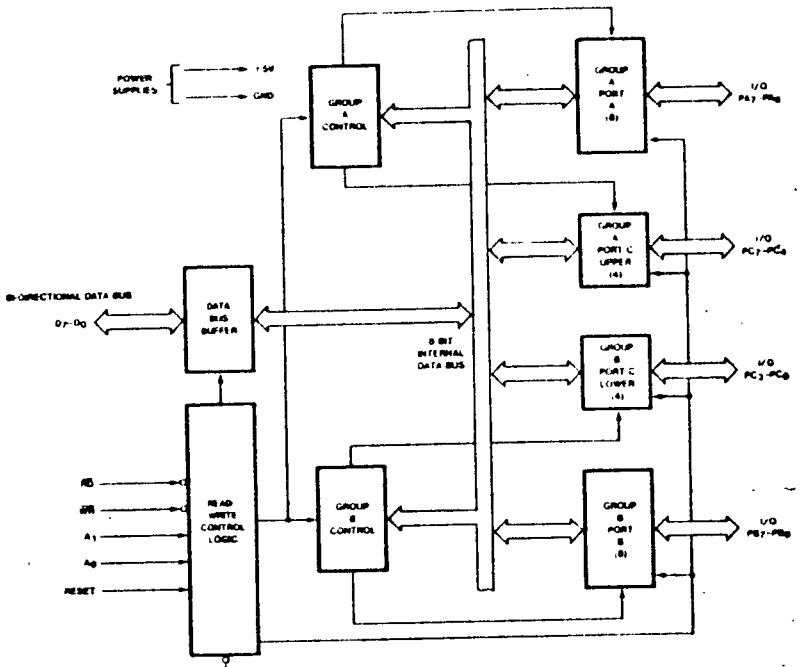
En este bloque se realiza la gestión del teclado y la impresora, empleando para ello el C.I. 8255 de INTEL, cuya denominación es PPI (Programable Peripheral Interface) y cuya estructura se observa en la figura 4.19.

El 8255 es un circuito programable de entradas/salidas de propósito general que dispone de 24 terminales de entrada/salida que pueden ser programados independientemente en dos grupos de doce y se pueden emplear en tres modos diferentes:

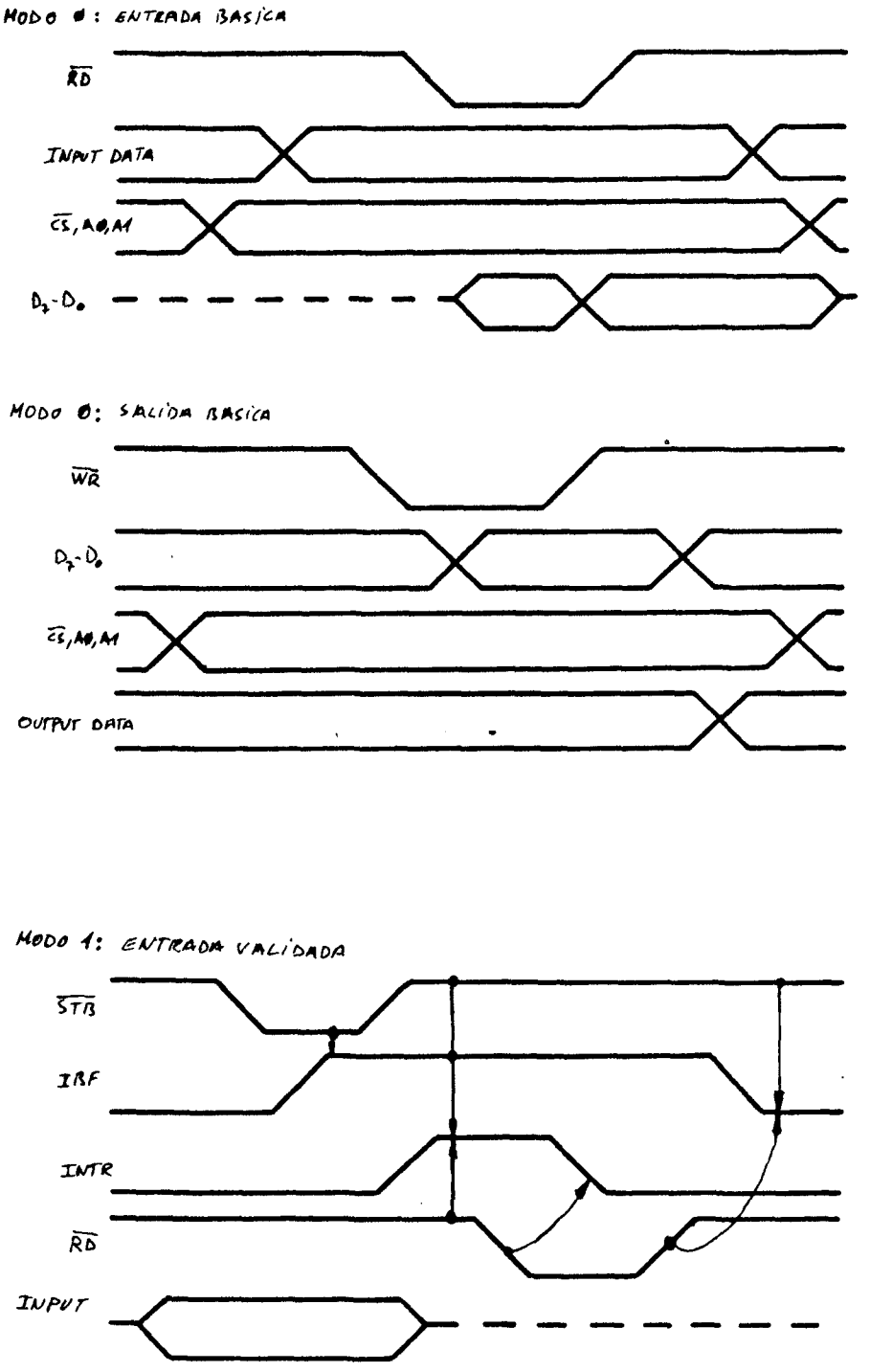
a) modo 0.- Cada grupo de doce terminales puede ser programado en grupos de cuatro para actuar como entradas o como salidas. Admite hasta 16 configuraciones diferentes.

b) modo 1.- Cada grupo puede ser programado para disponer de ocho lineas de entrada o salida, mientras que las tres restantes, por cada grupo, son utilizadas para controlar la comunicación, ya sea por espera o por interrupción.

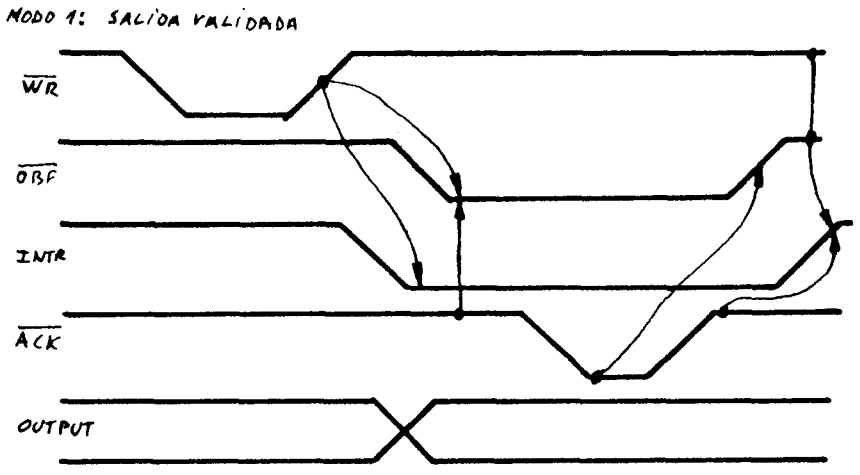
En este diseno se usará el modo 1: en modo entrada con handshake e interrupción, para el teclado, y en modo salida con handshake y espera, para la impresora.

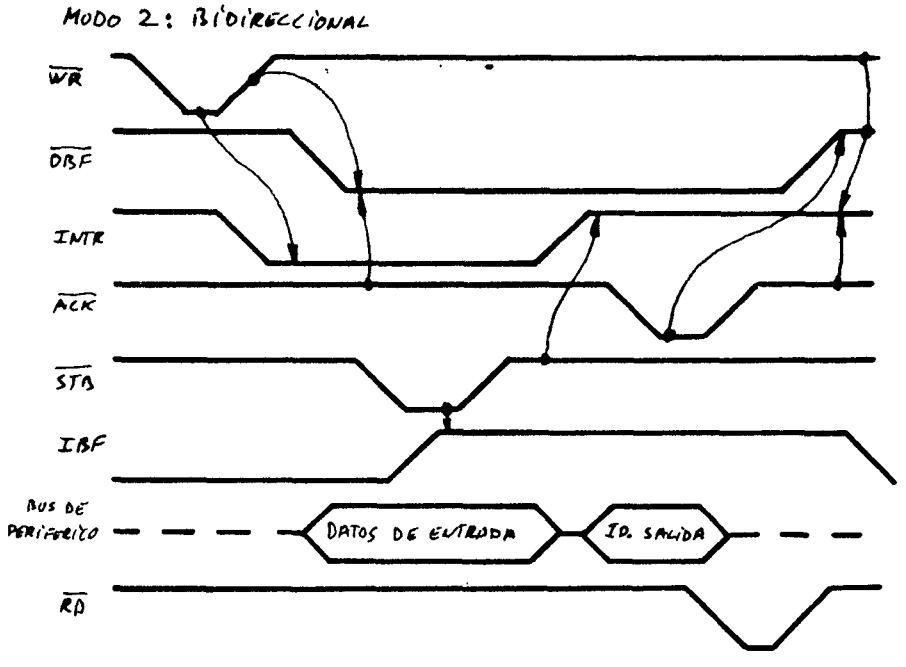


- fig. 4.19 -



- fig. 4.20 -





- fig. 4.21 -

c) modo 2.- En este modo se adopta la configuración de bus bidireccional, empleando ocho líneas para datos y cinco para control de la comunicación.

Las temporizaciones empleadas en cada caso se pueden observar en las figuras 4.20 y 4.21.

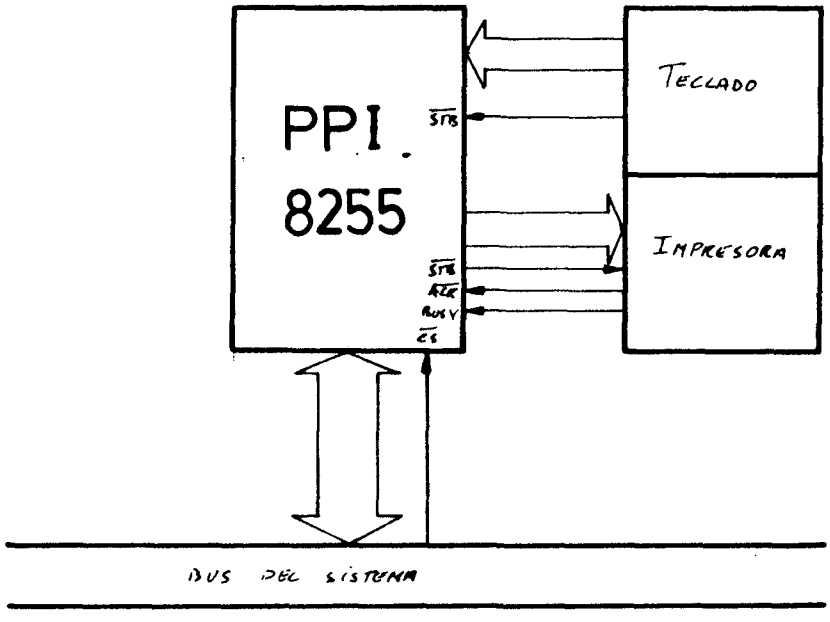
La configuración funcional de cada puerto es programada por el sistema. En esencia, la CPU emite una palabra de control al 8255 conteniendo información sobre modo de funcionamiento, activación de bits, restauración de bits, etc., con el fin de inicializar el circuito. El registro de control sólo puede ser escrito.

La estructura de este bloque se observa claramente en la figura 4.22.

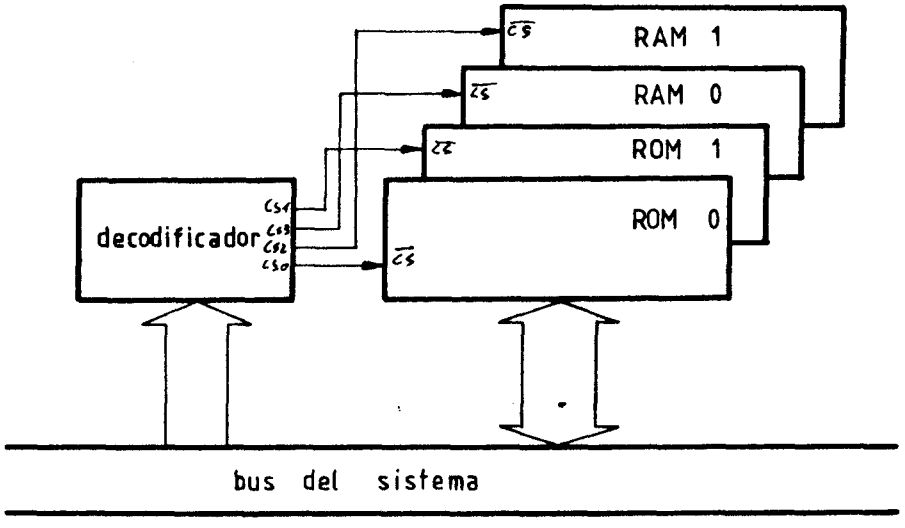
4.2.6 Bloque de memoria.

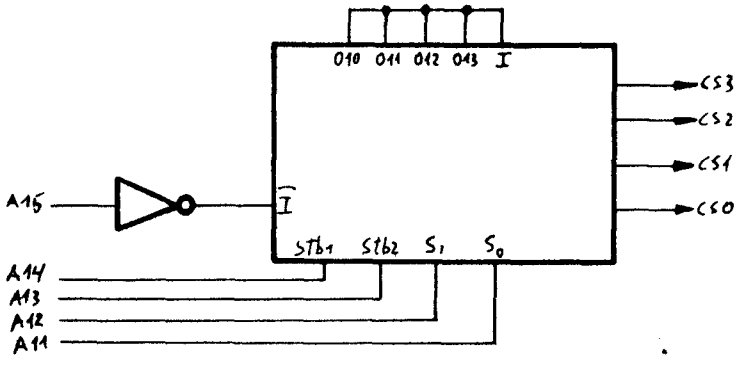
En el diseño de este bloque se ha pensado en la necesidad de un direccionamiento eficiente de la memoria utilizada, con el fin de favorecer la ampliación del sistema sin grandes complicaciones. Como se ve en la figura 4.23, el bloque de memoria está compuesto por 4 Kbytes de EPROM, en los que se incluye el programa de control del sistema, y 4 Kbytes de RAM, con capacidad para dos pantallas, buffer de impresora, variables y stack. Esta memoria está completamente decodificada; es decir, a cada dirección de memoria física le corresponde una y sólo una dirección lógica, evitando así replicas del mapeado de memoria inicial a lo largo de los 64 Kbytes directamente direccionables.

El decodificador de selección de memoria se puede observar también en la figura 4.23.



- fig. 4.22 -





- fig. 4.23 -

4.2.7 Estructura final.

La estructura final, a nivel de diagrama de bloques, se puede observar en la figura 4.24, y engloba a todos los bloques que han sido descritos. De todas formas, en el apèndice IV se expone con màs detalle esta estructura.

4.3 Divisiòn en dos tarjetas (PCBs).

Para conseguir la màxima compenetraciòn de funciones a nivel de tarjeta o PCB (Printed Circuit Board) y el mìnimo nivel de interferencia por adiciòn de nuevas tarjetas al sistema, se han concentrado los bloques analizados anteriormente en dos PCBs:

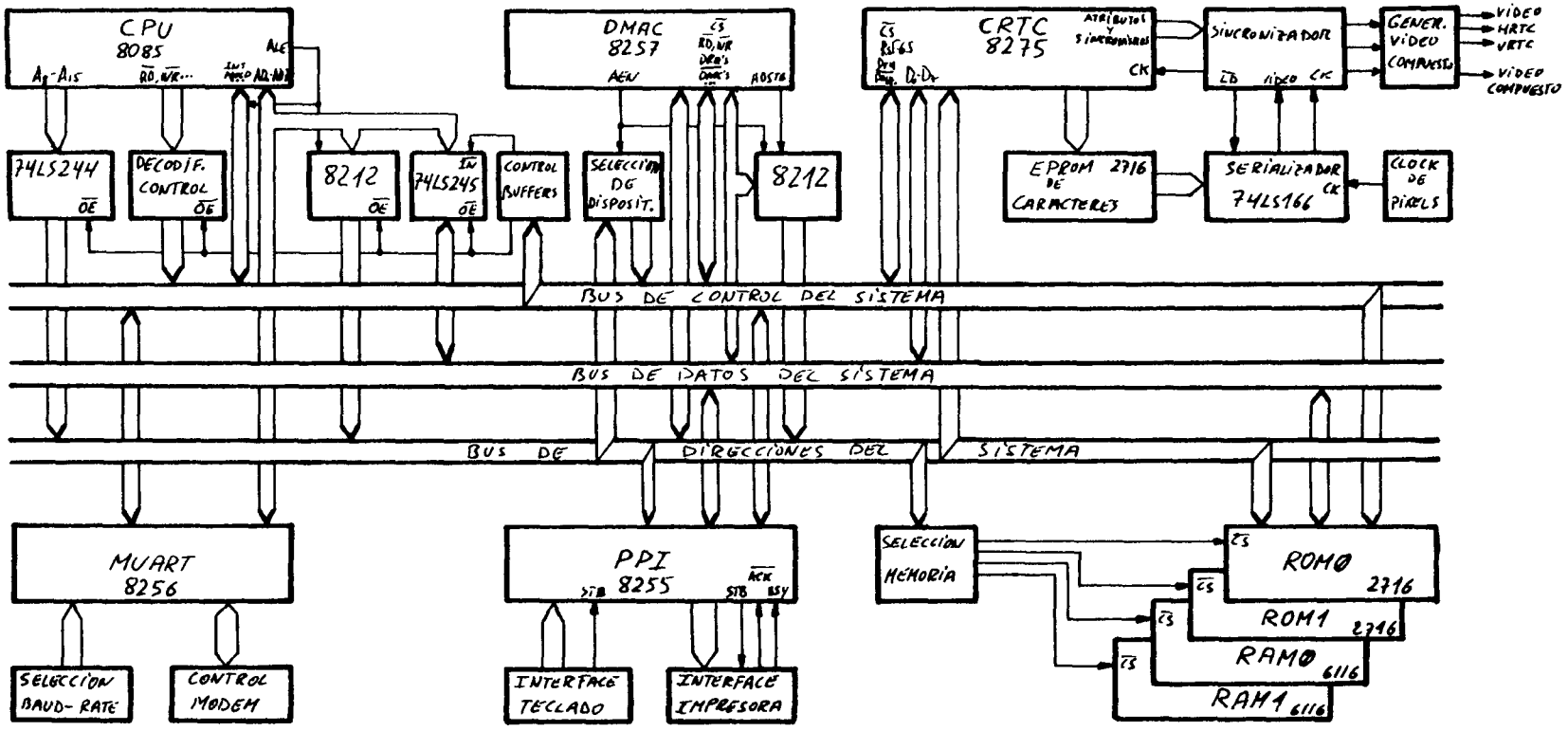
a) tarjeta de control.- que incluye los siguientes bloques:

- bloque de control del sistema
- bloque de transferencias
- bloque de comunicaciòn serie y funciones especiales.
- bloque de comunicaciòn en paralelo

b) tarjeta de CRT.- que incluye:

- bloque de control del CRT
- bloque de memoria

En el caso de que una posible ampliaciòn del sistema suponga la anulaciòn de la tarjeta "b" (p.e. un mòdulo de control de pantalla gràfica), el bloque de memoria podria implementarse en esta ampliaciòn sin afectar en gran medida a la configuraciòn del resto del sistema.



- fig. 4.24 -

Asimismo, el bloque de memoria puede quedar distribuido en varias tarjetas, dependiendo de las ampliaciones que se disenen como complemento de la configuración básica presentada.

Para garantizar la interconexión de estas tarjetas se hace necesario establecer un bus generalizado, que se tratará en el siguiente apartado.

4.4 El bus del sistema: MACBA-62.

La necesidad de establecer una conexión sencilla y fiable (no tanto a nivel físico, sino a nivel lógico) obliga al análisis de las diferentes soluciones que se pueden plantear y que, como en este caso, son de dos tipos:

- utilización de un bus normalizado
- creación de un bus propio

4.5.1 Buses normalizados.

La utilización de un bus normalizado supone la solución más sencilla para la interconexión de módulos en un sistema flexible, ya que las líneas disponibles han sido convenientemente seleccionadas para satisfacer todas las necesidades en cuanto a sistemas microcomputadores se refiere.

A modo de ejemplo, veremos las características de los tres buses que más han logrado imponerse en el mercado, teniendo cada uno de ellos sus particularidades:

- MUBUS: un intento de normalización a nivel europeo
- S100: un bus que ha pasado a ser norma internacional, propuesta por el IEEE
- MULTIBUS: bus propuesto por INTEL que se ha convertido en una norma muy extendida

Veamos las caracterlsticas de cada uno de ellos:

a) MUBUS.

Esta norma ha sido propuesta en la revista de la asociaciòn europea de usuarios de microprocesadores, EUROMICRO NEWSLETTER, y también en la editada por el laboratorio de calculadoras digitales de E.P.F. Lausanne (Suiza), MICROSCOPE y ha sido adoptada por fabricantes suizos, italianos y espanoles, así como por varias universidades.

Las lineas del bus se clasifican en cuatro grupos:

- 16 lineas de direcciòn
- 16 lineas de datos
- 20 lineas de control
- 10 lineas de alimentaciòn

Todas las senales de control son activas a nivel bajo, y se controlan mediante puertas con salida open-colector, siendo las màs destacables:

- ADMEM .- direcciòn de memoria vàlida
- ADPER .- direcciòn de periférico vàlida
- REFRESH .- direcciòn de refresco de DRAM vàlida
- NOTREADY .- peticiòn de suspensiòn del ciclo del procesador
- WRITE .- define el ciclo en curso como de escritura (WRITE=0) o de lectura (WRITE=1)
- NODA .- el dato en el bus no es vàlido
- RESET .- inicializa a todo el sistema
- INTREQ .- peticiòn de interrupciòn
- INTACK .- respuesta a INTREQ

- INTIN .- entrada de la cadena de prioridad de interrupción

- INTOUT .- salida de la cadena de prioridad de interrupción

- NMI .- petición de interrupción no enmascarable

- HOLDREQ .- solicitud del bus

- HOLDACK .- cesión del bus

- HOLDIN .- entrada de la cadena de prioridad de solicitud del bus

- HOLDOUT .- salida de la cadena de prioridad de solicitud del bus

- PROCREQ .- petición del procesador

- USERSCLOCK .- reloj de usuario

- SYSTEMCLOCK .- reloj del sistema

Las lineas de alimentación sugeridas son:

- GND (0 v.): masa lógica del MUBUS y de alimentación

- 5 v., 12 v., 15 v., -5 v., -12 v., -15 v.

- GND (A): masa separada para convertidores A/D

- GND (X): masa separada para circuitos optoaislados

- 5 v. (X): alimentación para circuitos optoaislados

b) S-100.

Se trata de una norma útil para la comunicación entre módulos de un miccrocomputador y está formado por un conjunto de 100 lineas que comprenden direcciones, datos y control.

Este bus es muy usado, empleandose por más de un centenar de fabricantes para más de 700 tipos diferentes de tarjetas y sistemas completos.

Su aplicación en sistemas microcomputadores se refiere en particular a los que constan de un número total de dispositivos interconectados que no supere la veintena, usan una vía de transmisión eléctricamente corta y con velocidades máximas de transferencia bajas, con señales en el bus de hasta 6 MHz.

Esta norma trata de definir un sistema de interface de manejo fácil y racional, que asegure la compatibilidad de los diseños actuales y futuros, con facilidad de ampliaciones modulares y de interconexión de dispositivos de fabricantes distintos para construir sistemas completos con la menor limitación posible en el rendimiento de cada una de las partes componentes.

La especificación a la que me refiero define las señales de este bus según la norma IEEE 696.1/D2.

Aunque existen 100 posibles señales, sólo se han especificado 93, quedando 7 sin asignación. Si se distribuye funcionalmente todo el conjunto de señales del S-100 obtendremos la siguiente clasificación:

- bus de datos, con 16 líneas
- bus de direcciones, con 24 líneas
- bus de control, con un total de 27 líneas:
    - status, 8 líneas
    - salidas de contol, 5 líneas
    - entradas de control, 6 líneas
    - control de DMA, con 8 líneas
- bus de interrupciones vectorizadas, con 8 líneas
- bus de servicio, con 18 líneas

c) **MULTIBUS.**

Es un bus introducido en 1977 por INTEL una amplia difusiòn y aceptaciòn. Permite la realizaciòn de sistemas multiprocesadores de forma que el arbitraje del bus se realiza mediante un mecanismo de encadenamiento (daisy-chain).

En la figura 4.25 se representan las senales de dicho bus, asi como sus mnemònicos y una breve descripciòn de su significado. Su distribuciòn por grupos es como sigue:

- alimentaciòn, senales 1 a 12 y 75 a 86
- control, senales 13 a 34
- vectores de interrupciòn, senales 35 a 42
- direcciones, senales 43 a 58
- datos, senales 59 a 74

Una barra cruzada ("/" o slash) detràs del mnemònico indica que èsta es activa a nivel bajo.

La descripciòn de las senales se realizarà por el siguiente orden:

- control del bus
- peticiòn de interrupciones
- transferencia de datos
- alimentaciòn
- senales no definidas

c.1 - senales de control del bus.

- BCLK (Bus clock, linea 13).- Es un reloj asincrono con los de las posibles CPUs enganchadas al sistema y cuya misiòn fundamental es la de sincronizar los circuitos usados para el arbitraje del bus.

Nº señal	Nemotécnico	Descripción	Nº señal	Nemotécnico	Descripción
1	GND	Masa	2	GND	Masa
3	+5	+5 VDC	4	+5	+5 VDC
5	+5	+5 VDC	6	+5	+5 VDC
7	+12	+12 VDC	8	+12	+12 VDC
9	−5	−5 VDC	10	−5	−5 VDC
11	GND	Masa	12	GND	Masa
13	BCLK/	Reloj del bus	14	INIT/	Puesta del sistema a un estado inicial
15	BPRI/	Señal de entrada de la prioridad en el bus	16	BPRO/	Señal de salida de la prioridad en el bus.
17	BUSY/	Bus ocupado	18	BREQ/	Petición del bus
19	MRDC/	Orden de lectura en memoria	20	MWTC/	Orden de escritura en memoria.
21	IORC/	Orden de lectura en periférico de Entrada-Salida	22	IOWC/	Orden de escritura en periférico de Entrada-Salida.
23	XACK/	Aceptación de transferencia	24	INH1/	Inhibición de la RAM.
25	AACK/	Aceptación anticipada de transferencia	26	INH2/	Inhibir la ROM o PROM
27		No usadas	28		No usada
29		No usada	30		No usada
31	CCLK/	Reloj constante	32		No usada
33		No usada	34		No usada
35	INT6/	Petición de Interrupción	36	INT7/	Petición de interrupción
37	INT4/	» »	38	INT5/	» »
39	INT2/	» »	40	INT3/	» »
41	INT0/	» »	42	INT1/	» »
43	ADRE/	Señal de direcciones	44	ADRF/	Señal de direcciones
45	ADRC/	» »	46	ADRD/	» »
47	ADRA/	» »	48	ADRB/	» »
49	ADR8/	» »	50	ADR9/	» »
51	ADR6/	» »	52	ADR7/	» »
53	ADR4/	» »	54	ADR5/	» »
55	ADR2/	» »	56	ADR3/	» »
57	ADR0/	» »	58	ADR1/	» »
59	DATE/	Señal de datos	60	DATF/	Señal de datos
61	DATC/	» »	62	DATD/	» »
63	DATA/	» »	64	DATB/	» »
65	DAT8/	» »	66	DAT9/	» »
67	DAT6/	» »	68	DAT7/	» »
69	DAT4/	» »	70	DAT5/	» »
71	DAT2/	» »	72	DAT3/	» »
73	DAT0/	» »	74	DAT1/	» »
75	GND	Masa	76	GND	Masa
77	−10	−10 VDC	78	−10	−10 VDC
79	−12	−12 VDC	80	−12	−12 VDC
81	+5	+5 VDC	82	+5	+5 VDC
83	+5	+5 VDC	84	+5	+5 VDC
85	GND	Masa	86	GND	Masa

- fig. 4.25 -

- INIT (Initialization signal line, linea 14).- Senal usadas para inicializar todo el sistema. Será enviada al bus por un procesador master o por el operador.

- BPRI (Bus priorty in signal, linea 15).- Esta senal es de entrada en cada uno de los mòdulos enganchados al bus que pueden ser amos. El significado para cada uno de los mòdulos es si existe o no algùn otro mòdulo de mayor prioridad pidiendo el bus.

- BPRO (Bus priority out signal, linea 16).- Se usa, junto con BPRI para el arbitraje del bus.

- BUSY (Bus busy signal, linea 17).- Esta senal està gobernada por el mòdulo que posee el bus en un momento dado y previene a los demàs de no acceder al bus.

- BREQ (Bus request signal, linea 18).- Es usada para indicar que un mòdulo master requiere el uso del bus.

- CCLK (Constant clock, linea 31).- Es un reloj de frecuencia constante para uso general de cualquier mòdulo. El periodo mìnimo de esta senal es de 100 ns y su ciclo de trabajo del 35% al 65%.

c.2 - petición de interrupciones.

- INT0-INT7 (Interrupt request signals, lineas 35-42).- Son usadas para peticiòn de interrupciones. La senal INT0 tiene la prioridad màs alta e INT7 la màs baja.

c.3 - transferencia de datos.

- ADR0-ADRF (Address lines, lineas 43-58).- Son las 16 lineas de direcciones, donde ADR0 contiene el bit de menor peso.

- DAT0-DATF (Data lines, lineas 59-74).- Son 16 lineas bidireccionales usadas para las transferencias de datos. En los sistemas de ocho bits sólo se usan DAT0-DAT7.

- MRDC (Memory read command, linea 19).- Indica que el bsde irecciones contiene una dirección de memoria y que se ha de realizar una operación de lectura.

- MWTC (Memory write command, linea 20).- Similar a la anterior con la salvedad de que la operación es de escritura.

- IORC (I/O read command, linea 21).- Indica que ha de leerse el port de entrada cuya dirección está en el bus de direcciones.

- IOWC (I/O write command, linea 22).- Idem que la anterior, sólo que la operación es de escritura.

- XACK (Transfer acknowledge signal, linea 23).- Indica que se ha terminado una operación de lectura o escritura.

- AACK (Advanced acknowledge signal, linea 25).- Es una senal que ha sido pensada para trabajar con microprocesadores i8080, constituyendo una respuesta rápida a las operaciones de lectura/escritura de forma que la CPU pueda realizarlas sin esperar ningún ciclo adicional.

- INH1 (Inhibit RAM signal, linea 24).- Esta senal permite inhibir a la memoria RAM en los sistemas que usen memoria local en las mismas direcciones.

- INH2 (Inhibit ROM signal, linea 26).- Idem para el caso de la ROM.

c.4 - alimentaciones.

Constituyen las lineas más altas y más bajas del bus, pudiendose ver su significado en la figura 4.25.

c.5 - senales no definidas.

Para posterior definición y uso se han dejado las lineas 27, 28, 29, 30, 32, 33 y 34 sin definir.

4.5.2 Razones de creación del MACBA-62.

A pesar de las enormes facilidades que ofrece cualquiera de los buses anteriormente analizados, existen varias razones que, bajo mi punto de vista, hacen aconsejable el uso de un bus propio, denominado MACBA-62. Estas razones son:

- para conseguir un máximo aprovechamiento de las caracteristicas especiales de algunos C.I. utilizados (MUART, DMAC).

- por un exceso de lineas en los buses normalizados analizados.

- para favorecer el desarrollo de tarjetas propias a nivel universitario, sin caer en la tentación (buena o mala, según se mire) de adquirirlas en el mercado. Se trata de potenciar el aprendizaje y la investigación.

De todas formas, es posible disenar placas de adaptación a otros buses y aprovechar las caracteristicas que presentan.

4.5.3 Descripción y especificaciones del bus MACBA-62.

En la figura 4.26 se pueden observar las 62 lineas que forman este bus. Estas lineas forman seis grupos:

- alimentación (5) y no definidas (4)
- control (24)
- vectores de interrupción (5)
- direcciones (16)
- datos (8)

número de linea	Mnemohico	Función
1	Rst 7.5	Interrupción directa de la CPU.
2	+ 12 v.	Aliment. 12 v.
3	Rst 6.5	Interrupción directa. (para el MCRT)
4	Trap	Interrupción directa. (máxima prioridad)
5	Int1	Interrupción controlada por la MUART.
6	Int0	Interrupción controlada por la MUART.
7	(Bell)*	Disparo para generación de sonido de campanilla.
8	- 5 v.	Aliment. -5 v.
9	Clk.	System clock.
10	Drq1	DMA request-c. 1
11	(Dack1)*	DMA ack. -c. 1
12	Drq2	id. -c. 2 (para CRTC).
13	(Dack2)*	id. -c. 2
14 - 21	D7 - D0	Bus de datos.
22	(cs CRTC)*	Seleccion CRTC.
23 - 30	A7 - A0	Bus direc.-LSB.
31	GND	Aliment. 0 v.
32	(AEN)*	Habilitación de direcciones. (la desactiva el DMAC)
33 - 40	A8 - A15	Bus direc.-MSB.
41	TRIG5	Disparo contador 5 de la MUART.
42	FREE	Sin asignación.
43	CONT2	Entrada contador 2 de la MUART.
44	CONT3	Entrada contador 3 de la MUART.
45 - 47	FREE	Sin asignación.
48	TC	Fin de cuenta del DMAC.
49	MARK	Senalización módulo 256 del DMAC.

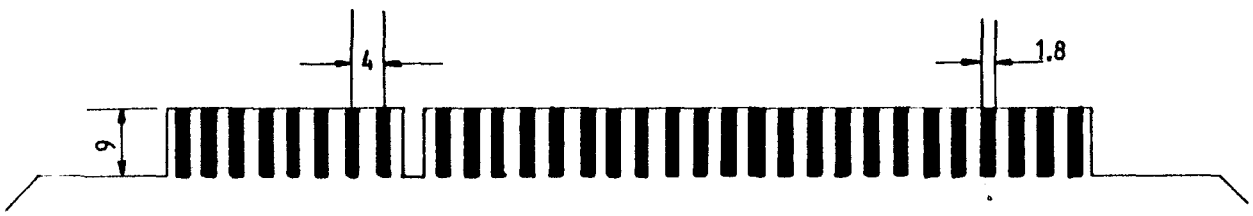
fig 4.26
- descripción de las lineas del bus MACBA-62 -

número de linea	Mnemohico	Función
50	(Dack3)*	DMA ack. - c. 3
51	Drq3	DMA request-c. 3
52	(Dack0)*	id. -c. 0
53	Drq0	id. -c. 0
54	+ 5 v.	Aliment. 5 v.
55	RESOUT	Reset out
56	(RESIN)*	Reset in
57	(IORD)*	Lectura en port.
58	(IOWR)*	Escritura "
59	(MEMRD)*	Lectura memoria.
60	(MEMWR)*	Escritura "
61	READY	Preparado.
62	- 12 v.	Aliment. -12 v.

Nota: Las lineas marcadas con ()* son activas a nivel bajo.

fig. 4.26
- continuación -

Las especificaciones de este bus se pueden ver en la figura 4.27.



- fig. 4.27 -

CAPITULO 5

Capitulo 5: ESTRUCTURA A NIVEL SOFTWARE

En este capitulo se realizará una descripción completa del software de soporte del sistema, teniendo en cuenta que ha sido desarrollado en un lenguaje de alto nivel (compilado) aplicado a microprocesadores, el PL/M 80.

El listado completo del programa puede encontrarse en el apéndice III, asi como el proceso de linkaje y localización del código objeto resultante.

5.1 Descripción del programa.

El programa consta esencialmente de dos módulos:

a) El primero realizado en ensamblador, en el que se indican los saltos a las subrutinas de servicio de las diferentes interrupciones soportables por el microprocesador INTEL 8085.

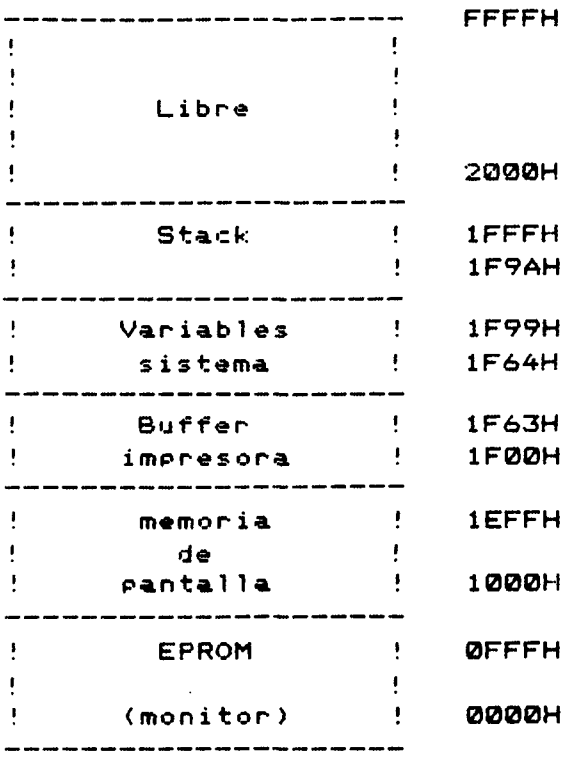
b) El segundo realizado integramente en PL/M 80, en el que pueden destacarse cuatro partes:

- inicialización
- bucle principal
- subrutinas de servicio normal
- subrutinas de servicio de interrupción

b.1 - Inicialización.

En este apartado se realiza la inicialización de los dispositivos periféricos, esto es, el paso de los parámetros necesarios a estos dispositivos para completar la configuración del sistema. También se realiza la declaración e inicialización de las variables del sistema.

El mapa de memoria después de la inicialización queda así:



Las variables del sistema indican el status del programa en cada instante, y son las siguientes:

- CURSX.- coordenada X actual del cursor.
- CURSY.- coordenada Y actual del cursor.
- ESC.- indicador de función especial.
- TEMP2.- variable temporal.
- KBCHAR.- código del último caracter leido del teclado.
- USCHAR.- código del último caracter leido de la UART.
- PROTEK.- indicador de habilitación del teclado.
- MSTATUS.- indicador de status de la MUART.
- BAUD.- código de velocidad de transmisión.
- TEMP.- variable temporal.
- MPORT2BUF.- buffer del puerto 2 de la MUART.

- PRT$FLAG.- indicador de habilitación del canal de impresora.

- EFECTOS$ESPECIALES.- buffer del registro de efectos especiales del CRT.

- LOCLIN.- indicador de modo de funcionamiento (local o remoto).

- CURAD.- direccion absoluta actual del cursor.

- LINEAD.- dirección de comienzo de la linea en curso.

- TOPDIS.- dirección de comienzo de la ventana de texto visualizada.

- PRT$POINTER.- puntero interior del buffer de impresora.

- TEMPAD.- variable temporal.

- BOTDIS.- dirección de comienzo de la última linea de la ventana de texto visualizada.

- BEGPRTBUFFER.- dirección de comienzo del buffer de impresora.

- ENDPRTBUFFER.- dirección final del buffer de impresora.

- BLOCK1$AD, BLOCK1$TC, BLOCK2$AD, BLOCK2$TC.- parámetros de reinicialización de DMA.

- SCFLAG.- indicador de envio directo desde la pantalla al ordenador.

- UFLAG.- indicador de caracter en la UART.

- KBFLAG.- indicador de caracter en el teclado.

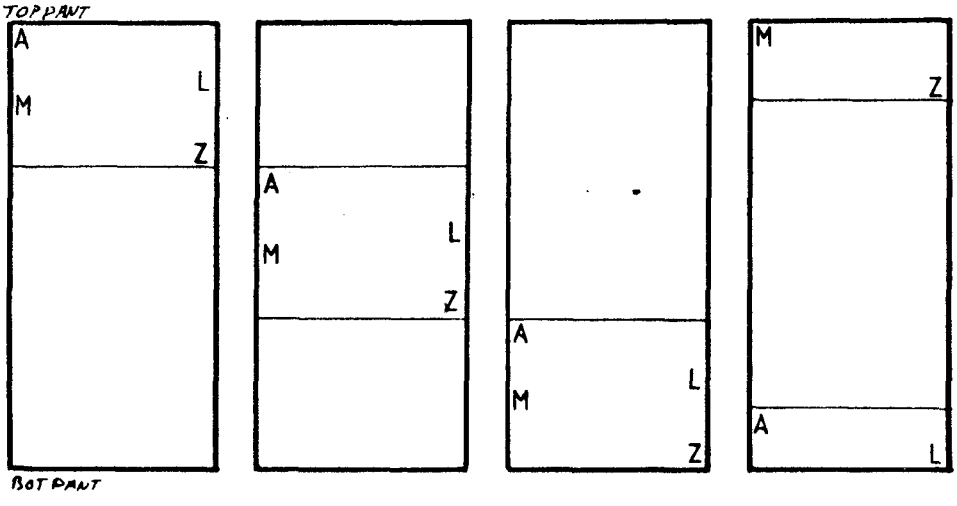
- TOPDOS, BOTDOS.- variables temporales complementarias de TOPDIS Y BOTDIS.

b.1.1 - parámetros de reinicialización de DMA.

Las variables BLOCK1$AD, BLOCK1$TC, BLOCK2$AD,BLOCK2$TC contienen los parámetros de reinicialización del controlador de DMA después de cada barrido de pantalla. La necesidad de estas variables se justifica con el método empleado para realizar el refresco de la pantalla por el controlador de DMA, que explicaré a continuación, y con el que se concibe la memoria de pantalla como una lista circular, pudiéndosele aplicar todas las propiedades de las listas circulares.

El controlador de DMA (8257) posee una función especial denominada auto-load (auto-carga) con la que se permite la programación de diferentes bloques de transferencia para que sean accedidos secuencialmente. En esencia, esta función consiste en programar los registros del canal 2 de DMA con los parámetros del primer bloque a transferir, y los registros del canal 3 de DMA con los parámetros del segundo bloque; cuando se termine la transferencia del primer bloque, y estando habilitada la función auto-load, los parámetros del canal 3 se copian en el canal 2 y se comienza así la transferencia del segundo bloque. Si ahora se introducen en el canal 3 los parámetros de otro bloque (igual o diferente), al terminarse la transferencia del segundo bloque se copian los parámetros del nuevo bloque en el canal 2 y se repite el proceso.

Considerando la memoria de pantalla como una lista circular, podemos conseguir que su representación en el visualizador CRT sea una pantalla "sin fin", resultando así muy sencillo realizar scrolls o movimientos verticales de la misma sin preocuparnos de las limitaciones físicas.



- fig. 5.1 -

Este fenòmeno es fàcilmente implementable si tenemos en cuenta las dos representaciones de la pantalla:

- por un lado la representaciòn en memoria
- por otro, la representaciòn en el visualizador

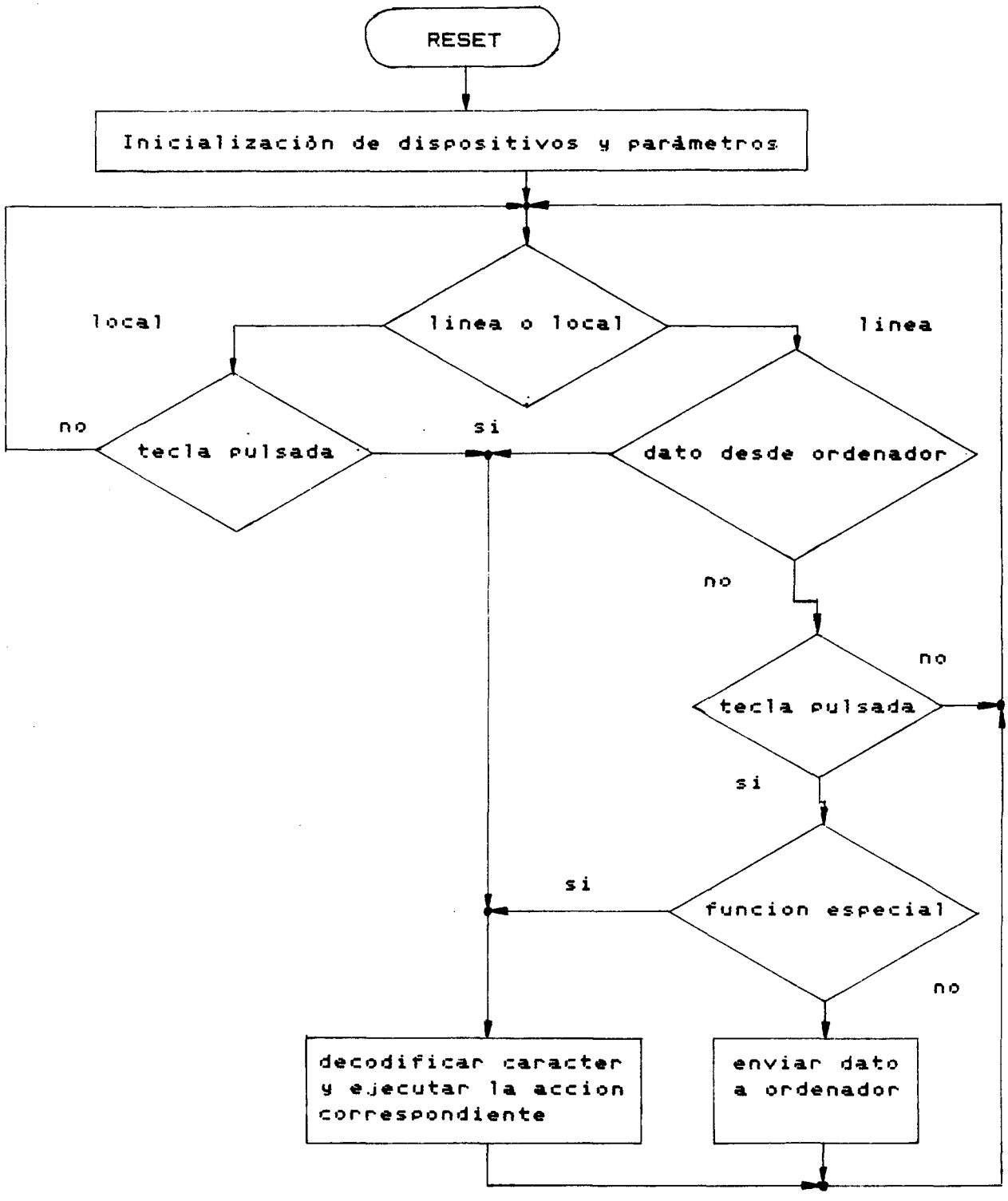
Si dividimos la pantalla en dos bloques y programamos los canales 2 y 3 de DMA con éstos, el efecto final es el de transferir de una sola vez toda la pantalla hacia el controlador de CRT, sin ser crìtico el punto de su representaciòn en memoria en el que empieza la pantalla visualizada.

El primer bloque tendrá como direcciòn de comienzo aquella que corresponda al primer caracter visualizado, y como direcciòn final, el final de la reprtesentaciòn en memoria de una pantalla. El segundo bloque tendrà como direcciòn de comienzo aquella que corresponda al principio de la representaciòn en memoria y como direcciòn final la del último caracter visualizado. Esta distribuciòn se observa mejor en la figura 5.1.

b.2 - Bucle principal.

Esta es la parte del programa que se ejecuta continuamente, y se encarga de testear el modo de funcionamiento del terminal (local/remoto), leer el teclado y la UART y llamar a las subrutinas que correspondan a la funciòn que se desarrolle en un momento determinado.

El funcionamiento del bucle principal se observa mejor en el diagrama de flujo situado en la figura 5.2.



b.3 - Subrutinas de servicio normal.

Este apartado contiene todas las subrutinas que pueden ser utilizadas por el bucle principal, e incluye varios tipos, como son:

- subrutinas de gestión de teclado e impresora.
- subrutinas de gestión de la memoria de pantalla.
- subrutinas de tratamiento de códigos de visualización.
- subrutinas de comunicación con el ordenador central.
- subrutinas de decodificación de caracteres.

Todas ellas pueden observarse más detalladamente en el apèndice III.

5.2 Iniciación a la ampliacion de funciones.

Si se observa el listado del programa (apèndice III), se comprende fàcilmente la forma en que ha sido implementada la indexación de cada una de las funciones especiales, según el grupo al que pertenezca.

Existen tres grupos de funciones:

- funciones de primer nivel (linefeed, tab, etc.).
- funciones de primer nivel transmitibles.
- funciones de segundo nivel o funciones "escape".

Cada uno de estos grupos està representado por una tabla de còdigos en la que cada còdigo representa una función determinada. Asimismo, existen tres constantes, NUM$FUNC, NUM$ESP y NUM$ESK, que definen el tamano de cada tabla, y, por último, exite una subrutina de decodificación de caracteres que llama a la subrutina de función especial según la posición del còdigo de control en su tabla correspondiente.

Con esto, si se quiere anadir una función a alguno de los tres grupos, sólo tenemos que incrementar la constante correspondiente al grupo en cuestión, insertar el código de control de la nueva función al final de esa tabla y situar una llamada, en la subrutina de decodificación y al final del grupo indicado, a la subrutina que ejecuta dicha función.

Se trata, pues, de un programa bastante completo en el que las funciones propias del terminal pueden ser incrementadas de forma muy sencilla y que puede soportar la inclusión de nuevas subrutinas que mejoren la "performance" del sistema, con un tiempo de desarrollo extremadamente corto frente a otros tipos de estructuración.

TERCERA PARTE

APENDICE I

Apendice I: MANUAL DE USO DEL TERMINAL MACBA-0

Para entrar en la explicación de las múltiples funciones implementadas en este terminal es preciso destacar sus características básicas, que son:

a) Formato de pantalla.

- 80 caracteres por fila
- 25 filas de caracteres

b) Formato de caracter.

- matriz de 5x7 pixels sobre un campo de 7x10 p.
- columnas 1 y 7 en blanco
- posición del cursor en la novena linea
- linea parpadeante como cursor

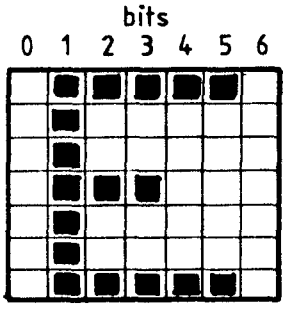
Como ya se ha mencionado, el generador de caracteres usado en este terminal es una EPROM 2716, que con sus 2Kbytes de memoria permite almacenar dos juegos de caracteres.

Las tres salidas de cuenta de linea del 8275 (LC0-LC2) se conectan a las tres lineas de direccionamiento menos significativas de la 2716, y las siete salidas con el código del caracter en curso (CC0-CC79) se conectan a las lineas de dirección A3-A9 de la misma 2716. Por otro lado, la salida de propósito general GPA0 se conecta a la linea A10 de la 2716, para conmutar los dos juegos de caracteresi.

Supongamos que se quiere visualizar la letra "E"; su código ASCII es 45H, codigo que se presenta en A3-A9. Las salidas de cuenta de linea realizarán una cuenta desde 0 hasta 7 para formar el caracter, como se observa en la figura A.1.

Codigo ASCII : 45H → direccion 01000101XXX → 228H-22FH

direccion	dato
228H	3E
229	02
22A	02
22B	0E
22C	02
22D	02
22E	3E
22F	00



- fig. A.1 -

```
:100000000100000000000000000000000000000000EF
:10001000000000000000000000000000000000000E0
:10002000000000000000000000000000000000000D0
:10003000000000000000000000000000000000000C0
:10004000000000000000000000000000000000000B0
:10005000000000000000000000000000000000000A0
:10006000000000000000000000000000000000000090
:0900700000000000000000087
:10007900000000000000000000000000000000000077
:10008900000000000000000000000000000000082A35
:10009900100C031C2A03000000000000000000000EF
:1000A900000000000000000000000000000000000047
:1000B900000000000000000000000000000000000037
:1000C900000000000000000000000000000000000027
:1000D900000000000000000000000000000000000017
:0900E900000000000000000000E
:1000F200000000000000000000000000000000000FE
:10010200000000000000008080808080000800141495
:10011200140000000000141143C143C1414000083CA9
:10012200A1C281E08000626100804323000040AA1
:10013200A042A122C000808080000000000080423
:10014200020202040800081020202010080008 2AD9
:10015201C081C2A08000008083E0808000000CD
:09016200000000080804000000 80
:10016B003C000000000000000000018000020 1000
:10017B00080402000 1C22322A26221C00080C084C
:10018B000808081C001C22201C02023E003E201006
:10019B00182022 1C00101814123E1010003E021ED4
:1001AB00202022 1C00380402 1E22221C003E20109C
:1001BB00080404040 1C22221C22221C001C2222E4
:1001CB003C20100E000000080000080000000000892
:0901DB000000080804100804 02E9
:1001E400040810000000 3E003E000000040810 2037
:1001F40010080400 1C22201008080008 1C222A3AB7
:10020400 1A023C000814222223C2222001E24241C30
:10021400 24241E001C22020202221C001E24242468
:10022400 24241E003E02020E02023E003E02020E82
:10023400020202003C02023A22223C002222223E16
:10024400222222001C0808080808 1C00702020 2014
:09025400202 1C0022120A060AF5
:10025D00122200020202020 23E0022362A2A2245
:10026D00222200 22262A3222222001C222222228F
:10027D00221C001E22221E02020200 1C2222222A01
:10028D00122C001E22221E0A1222003C02021C20E9
:10029D00201E003E0808080808080002222222222FB
:1002AD00221C00222222222214080022222222A8B
:1002BD00362200222214081422220022222214089F
:0902CD00080800 3E20100804029C
:1002D6003E001C0404040404 1C000002040810 2050
:1002E60000003820202020203800081C2A08080892
:1002F60008000000000000007E08901 1000000C9
:10030600000000003C203C223C0002021A26222269
:10031600 1E000000380484043800202C322222DB
:10032600 3C00000038248C848800382404 0E0404C1
:10033600040000008C22223C203C02021A26222293
:09034600220008000808080890D4
:10034F00002000202020A4241802022212 0A1622C4
:10035F00008808080808900000003 62A2A222280
:10036F00000000 1A26222222000000 18242424183C
:10037F00000000 1E22221E02020001C22223C202E
:10038F00200000 1A26020202000000380418201C68
:10039F00000808 1C0808089000000022222232 4C96
:1003AF00000000222222140800000022222A3E14FC
:09038F00000000221408142200C1
:1003C8000000022222223C203800003E1008043E0093
:1003D8001888890388891900080808080808087F
:1003E8000C909121909 10D0000008C2BB001000021
:0803F8000000000000000FD
:00000001FF
```

- fig. A.2 -

En la figura A.2 se puede ver el listado, en formato hexadecimal de INTEL, del primer juego de caracteres, quedando el segundo abierto para cualquier ampliación.

c) Caracteres especiales reconocidos.

- linefeed
- formfeed
- carriage return
- backspace y rubout
- break
- tab

d) Otros caracteres especiales.

- scroll (ver apartado de teclado)
- apertura y cierre del canal de impresora

e) Secuencias de Escape reconocidas.

- Esc c: Inhibir teclado
- " b: habilitar teclado
- " B: bajar cursor
- " J: limpiar la pantalla
- " K: borrar linea en curso
- " A: subir cursor
- " C: avanzar cursor
- " D: retroceder cursor
- " H: equivalente a HOME
- " Z: subir pantalla
- " Y: bajar pantalla
- " X: segundo juego de caracteres
- " W: primer juego de caracteres
- " V: subrayado ON
- " U: subrayado OFF

- " T: parpadeo ON

- " S: parpadeo OFF

- " R: inversión ON

- " Q: inversión OFF

f) Memoria de programa.

- 4 Kbytes de EPROM

g) Memoria de pantalla, variables, buffers y stack.

- 4 Kbytes de RAM

h) Monitor CRT.

- monitor YANJEN ELECTRONIC, modelo GM-1211 de 12 pulgadas.

- ancho de banda 18 MHz

- entrada de video compuesto

i) Velocidad de comunicación con el ordenador.

Esta velocidad es seleccionable, mediante cuatro microinterruptores, entre 50 bits/seg. y 19200 bits/seg., según se indica a continuación:

indicador de los microint.:		1	2	3	4
velocidad (bits/seg.):	50	1	1	1	1
	75	0	1	1	1
	100	1	0	1	1
	110	0	0	1	1
	150	1	1	0	1
	200	0	1	0	1
	300	1	0	0	1
	600	0	0	0	1
	1200	1	1	1	0
	2400	0	1	1	0
	4800	1	0	1	0
	9600	0	0	1	0
	19200	1	1	0	0
	"	0	1	0	0
	"	1	0	0	0
	"	0	0	0	0

j) Fuente de alimentación.

La fuente de alimentación utilizada es de la marca SEASONIC y actua según el principio de las fuentes conmutadas, dando los siguientes niveles de tensión e intensidad:

Tension	Intensidad	Color del cable
5 v.	5 A.	azul
- 12 v.	0.5 A.	amarillo
- 5 v.	0.5 A.	gris
12 v.	2.5 A.	rojo
0 v.	------	negro

k) Teclado.

El teclado utilizado es de tipo "inteligente" y forma parte de una serie de teclados multi-compatibles, siendo el modelo K-3.

Junto al teclado se suministra una placa de adaptación para dispositivos de entrada en paralelo validadas, como es el caso. El esquema de este teclado se puede observar en el apèndice IV.

Sus especificaciones físicas son:

- requerimientos de alimentacion: 5 v. (cc.), 280 mA.
- inclinación ajustable entre seis y doce grados.
- recorrido total de tecla: 4.3 mm. +- 0.5 mm.
- fuerza de activación: 60 gramos +- 25 grs.
- dimensiones: 440(ancho) x 185 (largo) x 37 (alto) mm.
- peso: 1.2 Kg.

Las especificaciones lógicas son:

- teclas con/sin sonido (beeper).

- funciones programables por el usuario.

- memoria salvaguardada por una bateria interna.

- códigos de control programables.

- habilitación/deshabilitación del teclado.

Si analizamos un poco más estas funciones, podremos observar que este teclado garantiza su compatibilidad con gran número de sistemas.

k.1 - Reseteado manual del sistema.

Para ello hay que pulsar F1 y F2 simultaneamente.

k.2 - Teclas de función F1 - F10.

F1 : Reset 1	F2 : Reset 2
F3 : LIST	F4 : RUN (cr)
F5 : CATALOG (cr)	F6 : CALL - 151

F7 : habilitación/deshabilitación del teclado

F8 : idem para el sonido de las teclas

F9 : sin asignar

F10: conmutación de modo local/remoto

k.3 - Teclas especiales.

- SCROLL LOCK: similar a CTRL-U.

Esta tecla se usa para mover el cursor a la derecha, y cada caracter de la pantalla por el que pase el cursor es enviado al ordenador como si se hubiese tecleado.

- CTRL-SROLL LOCK: similar a CTRL-C.

Se usa como código de "break".

k.4 - Tecla ALT para comandos de BASIC en una sola tecla.

- ALT-G = GOTO - ALT-N = NEXT

Los demàs comandos se aprecian en la figura A.3.

k.5 - Auto-repeticiòn.

Manteniendo pulsada una tecla durante màs de un segundo, el teclado enviarà el còdigo de dicha tecla repetidamente a un ritmo de 11 carecteres/seg.

k.6 - Teclado numèrico.

Pulsando "Num Lock", el LED correspondiente se enciende, lo que indica que cada tecla representa al nùmero correspondiente.

Pulsàndola otra vez, el LED se apaga y este teclado pasa al modo de funciones programables o teclas programables.

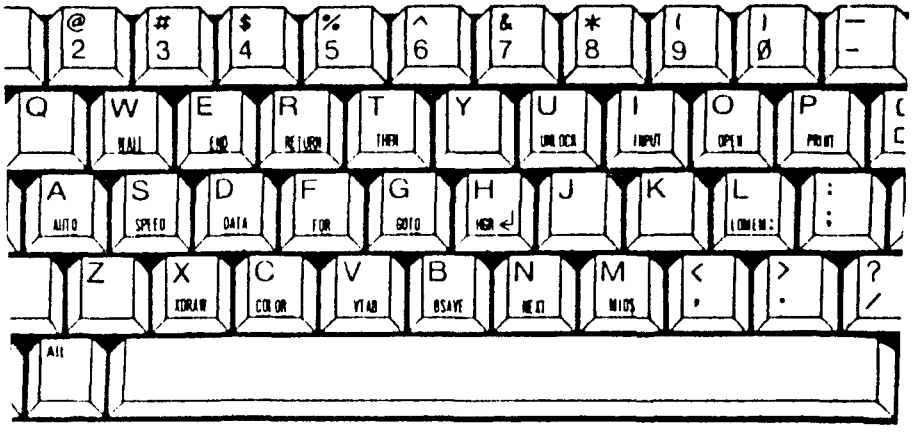
k.7 - Funciones programables.

Hay once teclas en el teclado numèrico que pueden almacenar un total de 60 bytes entre còdigos, comandos, frases y pequenos programas; estas teclas son programables por el usuario.

Estos 60 bytes son compartidos por todas las teclas de forma dinàmica, sin responder a una reparticiòn fija. Algunas de las teclas llevan impresos comandos o simbolos, aunque esto no significa nada, a menos que se programen en ellas los còdigos correspondientes.

El teclado posee una bateria (pila) interna que mantiene las funciones en memoria durante un tiempo aproximado de 5 anos.

Press [Alt]-[G] = GOTO Press [Alt]-[N] = NEXT



- fig. A.3 -

Para ver lo que hay almacenado en cada tecla sólo hay que pulsar simultaneamente ENTER (esq. inferior dcha.) y C.

Para borrar los datos de la memoria interna (funciones programables) hay que pulsar ENTER-R (el guión se utiliza como separador; no es necesario pulsarlo).

Para abrir el modo de programación, pulsar ENTER-O, y para cerrarlo, ENTER-C.

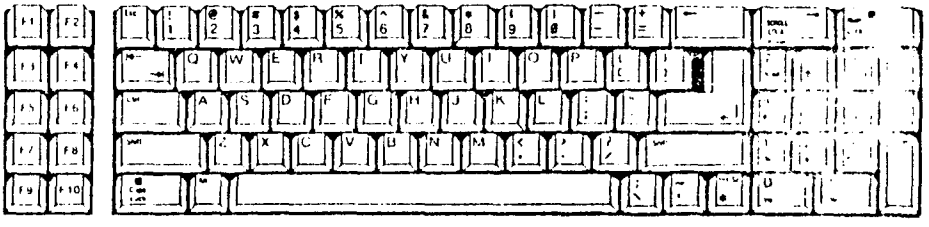
Una vez en el modo de programación, para programar una de las once teclas, sólo hay que pulsarla e introducir a continuación lo que se quiera programar, ya sea un código aislado, un comando, una frase, etc.

En caso de error al programar una tecla, sólo hay que pulsarla nuevamente y reprogramarla.

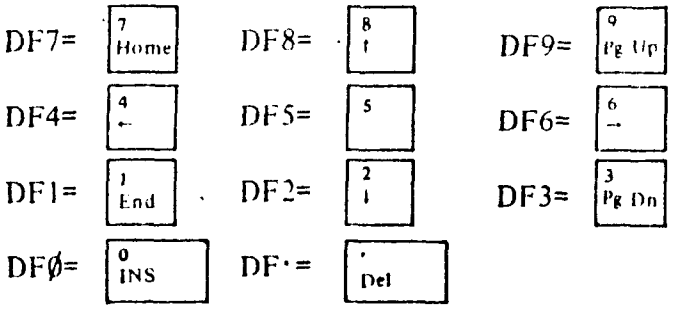
Cuando se hayan programado todas las teclas necesarias, hay que cerrar el modo de programación y ya estarán disponibles las funciones programadas.

Estas funciones de programación del teclado numérico se ven más claramente con un ejemplo:

- para programar GOTO en la tecla 7:
  - pulsar ENTER-O (abrir modo de programación)
  - pulsar la tecla 7
  - pulsar G,O,T,O o ALT-G
  - en caso de error pulsar otra vez la tecla 7 y repetir el paso anterior.
  - pulsar ENTER-C (cerrar modo de programación)



- fig. A.4 -



- fig. A.5 -

- para programar ESC-A (subir cursor) en la tecla 5 y ESC-B (bajar cursor) en la tecla 1:

- pulsar ENTER-0
- pulsar la tecla 5
- pulsar la tecla ESC
- pulsar la tecla A
- pulsar la tecla 1
- pulsar la tecla ESC
- pulsar la tecla B
- pulsar ENTER-C

La programación de mensajes y otros se realiza de forma similar.

Para terminar este apartado, conviene dar a conocer los códigos que puede generar el teclado, de acuerdo con el formato ASCII. Estos códigos se aprecian en la figura A.6.

KEY LEGEN	KEY NUMBER	ALONE	CAPLOCK	SHIFT	CONTROL	SHIFT CONTROL
A	30	61H	Same as SHIFT has been pressed	41H	01H	01H
B	48	62H	"	42H	02H	02H
C	46	63H	"	43H	03H	03H
D	32	64H	"	44H	04H	04H
E	18	65H	"	45H	05H	05H
F	33	66H	"	46H	06H	06H
G	34	67H	"	47H	07H	07H
H	35	68H	"	48H	08H	08H
I	23	69H	"	49H	09H	09H
J	36	6AH	"	4AH	0AH	0AH
K	37	6BH	"	4BH	0BH	0BH
L	38	6CH	"	4CH	0CH	0CH
M	50	6DH	"	4DH	0DH	0DH
N	49	6EH	"	4EH	0EH	0EH
O	24	6FH	"	4FH	0FH	0FH
P	25	70H	"	50H	10H	10H
Q	16	71H	"	51H	11H	11H
R	19	72H	"	52H	12H	12H
S	31	73H	"	53H	13H	13H
T	20	74H	"	54H	14H	14H
U	22	75H	"	55H	15H	15H
V	47	76H	"	56H	16H	16H
W	17	77H	"	57H	17H	17H
X	45	78H	"	58H	18H	18H
Y	21	79H	"	59H	19H	19H
Z	44	7AH	"	5AH	1AH	1AH
!1	2	31H	Not affected	21H	31H	21H
@2	3	32H	"	40H	32H	00H
#3	4	33H	"	23H	33H	23H
$4	5	34H	"	24H	34H	24H
%5	6	35H	"	25H	35H	25H
^6	7	36H	"	5EH	36H	1EH
&7	8	37H	"	26H	37H	26H
*8	9	38H	"	2AH	38H	2AH
(9	10	39H	"	28H	39H	28H
)0	11	30H	"	29H	30H	29H
| \	43	5CH	"	7CH	1CH	1CH

- fig. A.6 -

<,	51	2CH	"	3CH	2CH	3CH
>.	52	2EH	"	3EH	2EH	3EH
:;	39	3BH	"	3AH	3BH	3AH
?/	53	2FH	"	3FH	2FH	3FH
" '	40	27H	"	22H	27H	22H
{ [	26	5BH	"	7BH	1BH	1BH
}]	27	5DH	"	7DH	1DH	1DH
_ -	12	2DH	"	5FH	2DH	1FH
+=	13	3DH	"	2BH	3DH	2BH
←	14	08H	08H	08H	08H	08H
ESC	1	1BH	1BH	1BH	1BH	1BH
⇤	15	09H	09H	7FH	7FH	7FH
⇥						
SPACE	57	20H	20H	20H	20H	20H
↵	28	0DH	0DH	0DH	0DH	0DH
Scroll Lock →	70	15H	15H	03H	03H	03H

- fig. A.6 (cont.) -

APENDICE II

Apendice II: NORMAS DE COMUNICACION TIPICAS

En este apéndice trataré de explicar algunas normas o intentos de normalización que se han desarrollado en torno a la comunicación entre las partes de un microcomputador.

Para la comunicación entre el computador y sus periféricos (incluido el equipo terminal) consideraremos las siguientes recomendaciones:

a) para comunicaciones serie:

- RS-232 C

b) para comunicaciones en paralelo:

- CENTRONICS
- IEEE-488

También se abordará el tema de la normalización en cuanto a MODEMS o equipos de comunicación de datos se refiere, por consideralos como una parte muy importante de un circuito de transmisión de datos tipico.

1 - Norma de interconexión serie RS-232C.

Las normas RS-232 fueron definidas por la EIA (Electrical Industry Association) en cooperación con la Bell System, los fabricantes de ordenadores y los de MODEMS con el objeto de normalizar los circuitos de interconexión entre el equipo terminal de datos (ETD) y el equipo terminal del circuito de datos (ETCD).

En la actualidad la norma RS-232 C es la más usada en la comunicación serie entre ordenadores y sus periféricos. Sin embargo, tiene las limitaciones de separación entre el ETD y el ETCD (aproximadamente 15 metros) y de velocidad de transferencia de información (hasta 20 Kbits/s).

Las normas RS-232 cubren los tres aspectos siguientes de la comunicación entre el ETD y el ETCD: características eléctricas de las senales, características mecánicas de los conectores y descripción funcional de las senales usadas para realizar la comunicación. La letra C en RS-232 C indica la tercera y, por ahora, última revisión.

A lo largo de este apartado daremos solamente una descripción funcional de las senales, remitiendo al lector a la bibliografia indicada al final de este trabajo, para una información más detallada.

Dentro del conjunto de las senales podemos distinguir cuatro grandes grupos: de datos, de control, de temporización y las masas. En la figura A.7 se representa el número de la senal dentro del conector, el mnemónico, el sentido de conexión entre el ETD y el ETCD y una breve descripción de su significado.

Como norma general se establece que las senales de datos se consideran "uno lógico" cuando presentan un nivel de tensión entre -3 y -25 v., y "cero lógico" cuando presentan un nivel entre 3 y 25 v. En cuanto a las senales de control y temporización, se consideran en estado ON cuando presentan un nivel de tensión positivo, y en estado OFF cuando este nivel es negativo.

Veamos la descripción de las senales por grupos:

1.1 - senales de datos.

- BA. Transmisión de datos.- Es la senal usada para la transmisión de los datos entre el ETD y el ETCD. Para que el ETD transmita es necesario que las senales CA, CB, CC y CD estén en estado ON. Esta senal se conoce también como TXD.

Núm de señal	*Nemonico*	*Direccion*	*Breve descripcion*
1	AA		Señal de tierra
2	BA	Hacia ETCD	Transmisión de datos
3	BB	Hacia ETD	Recepción de datos
4	CA	Hacia ETCD	Petición de transmitir
5	CB	Hacia ETD	Preparado para transmitir
6	CC	Hacia ETD	Aparato de datos preparado
7	AB		Masa común de las señales
8	CF	Hacia ETD	Detector de señales de línea recibidas por el canal de datos
9	--	--	Reservada para la comprobacion de los datos
10	—	—	Reservada para la comprobacion de los datos
11			Sin asignacion
12	SCF	Hacia ETD	Detector de señales de línea recibidas por el canal de reserva de datos
13	SCB	Hacia ETD	Preparado el canal de reserva para transmitir
14	SBA	Hacia ETCD	Transmisión de datos por el canal de reserva
15	DB	Hacia ETD	Temporizacion para los elementos de señal en la transmisión
16	SBB	Hacia ETD	Recepción de datos por el canal de reserva
17	DD	Hacia ETD	Temporizacion para los elementos de señal en la recepción
18			Sin asignación
19	SCA	Hacia ETCD	Petición para transmitir por el canal de reserva
20	CD	Hacia ETCD	Terminal de datos preparado
21	CG	Hacia ETDD	Detector de la calidad de las señales de datos
22	CE	Hacia ETD	Indicador de llamada
23	CH/CI	Ambas	Selector de velocidad binaria con origen ETD (H) u origen ETCD (CI)
24	DA	Hacia ETCD	Temporización para los elementos de señal en la transmisión
25			Sin asignacion

- fig. A.7 -

- BB. Recepción de datos.- Es la senal usada para la transmisión de los datos entre el ETCD y el ETD. Se conoce también como RXD.

- SBA. Transmisión de datos para el canal de reserva.- Es equivalente a BA pero en el canal de reserva, que trabaja a velocidades menores.

- SBB. Recepción de datos para el canal de reserva.- Equivalente a BB pero en el canal de reserva.

1.2 - senales de control.

- CA. Petición de transmitir.- Esta senal es enviada desde el ETD hacia el ETCD para indicarle que quiere realizar una transmisión. Se conoce también como RTS.

- CB. Preparado para transmitir.- Es enviada desde el ETCD hacia el ETD indicando si el ETCD está preparado para realizar la transmisión por el canal de datos. Se conoce como CTS.

- CC. Aparato de datos preparado.- Esta senal es enviada desde el ETCD hacia el ETD indicando si el ETCD está preparado para funcionar. Se conoce como DSR.

- CD. Terminal de datos preparado.- Es enviada desde el ETD hacia el ETCD para indicar que está en disposición de establecer una comunicación. Se conoce como DTR.

- CE. Indicador de llamada.- Es enviada desde el ETCD hacia el ETD indicando si se está recibiendo una llamada o no. Se conoce como RI.

- CF. Detector de senales de linea recibidas por el canal de datos.- Es enviada desde el ETCD hacia el ETD indicando si las senales de linea recibidas están o no dentro de los límites especificados.

- CG. Detector de calidad en la senal de datos.- Esta senal va desde el ETCD hacia el ETD indicando si existe o no cierta probabilidad de error en los datos recibidos.

- CH. Selector de velocidad binaria.- Esta senal va desde el ETD hacia el ETCD y sirve para seleccionar una de las velocidades binarias de un ETCD sincrono o una de las dos gamas de velocidades de un ETCD asincrono.

- CI. Selector de velocidad binaria.- Esta senal sirve para la selección de la velocidad binaria en el ETD, en función de la misma en el ETCD.

- SCA. Petición para transmitir por el canal de reserva.

- SCB. Preparado el canal de reserva para transmitir.

- SCF. Detector de senales de linea recibidas por el canal de reserva.

1.3 - senales de temporización.

- DA. Temporización en la transmisión.- Indica el centro de cada bit a transmitir.

- DB. Temporización en la transmisión.

- DD. Temporización en la recepción.- Es equivalente a DA peor de cara al ETD.

1.4 - senales de masa.

- AA. Senal de tierra.- Tierra de protección.

- AB. Masa común de las senales. Se conoce como GND.

En la mayoria de los casos es suficiente la utilización de las siguientes senales: TXD, RXD, DTR, DSR, RTS, CTS y GND

Los términos empleados en la descripción de las senales anteriores han sido sacados de la recomendación V.24 del CCITT, que coincide en muchos aspectos con la RS-232 C.

2 - Interface de conexión en paralelo CENTRONICS.

Esta modalidad de interface permite ritmos de transferencia de hasta 75000 caracteres por segundo. En la figura A.8 se puede ver un cronograma que relaciona las distintas senales implicadas en función del tiempo.

La iniciativa en cuanto a la transmisión corresponde al controlador. Cuando éste desea iniciar la comunicación, establece los niveles apropiados en las lineas de datos y posteriormente activa la senal de validación.

Al finalizar la "validación", el periférico activa la senal de ocupado durante el tiempo preciso para ejecutar la función que le ha sido encomendada, por lo que su duración es muy variable, dependiendo especificamente de dicha función.

Al término de su función, el periférico comunica su situación de disponibiliodad para la aceptación de nuevos datos, mediante la activación de la senal de "aceptación", repitiendose el proceso.

3 - Norma para la interconexión digital de instrumentos programables IEEE-488 (ANSI-MC. 1.1 - 1975).

Este bus se utiliza fundamentalmente en la interconexión entre uno o varios procesadores y periféricos inteligentes. El hecho de que el protocolo de la transmisión sea asincrono hace que el bus pueda extenderse hasta 20 m., con cargas cada 2m. Su uso se extiende cada vez más, existiendo interfaces para toda clase de ordenadores, incluso personales.

El bus debe contar con un dispositivo que actue como controlador, encargado de gestionar las peticiones de uso del bus, ya sea para emitir mensajes (locutor) o para recibirlos (oyente).

Intel y Motorola han puesto a la venta sendas pastillas que realizan las funciones de control de la transferencia de datos en el bus (locutor y oyente (L, LE, T, TE)).

Intel, además, ha programado una pastilla UPI-41 para que se comporte como controlador del bus (C).

Esta norma, aprobada por el IEEE en 1975, fue ideada por HEWLETT-PACKARD para interconectar sus aparatos de medida programables.

Esta descripción del bus no pretende ser exhaustiva, pues ello requeriria un tomo entero, sino que se pretende aclarar los conceptos necesarios para comprender los fundamentos del funcionamiento del bus. En la bibliografia se puede encontrar una referencia que suministre información más completa y detallada.

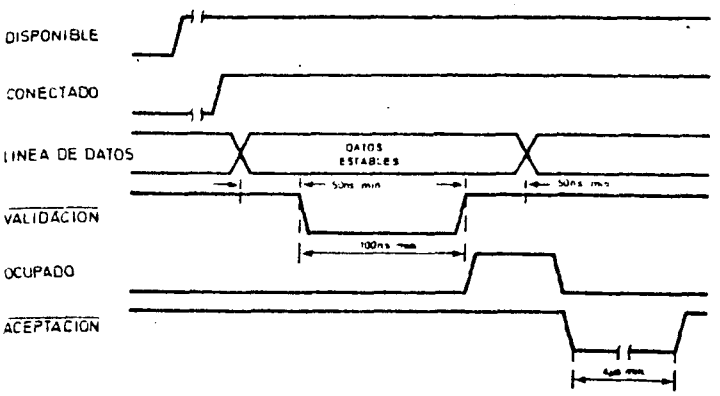
3.1 - senales eléctricas del bus.

El bus consta de 16 senales activas, divididas en tres grupos:

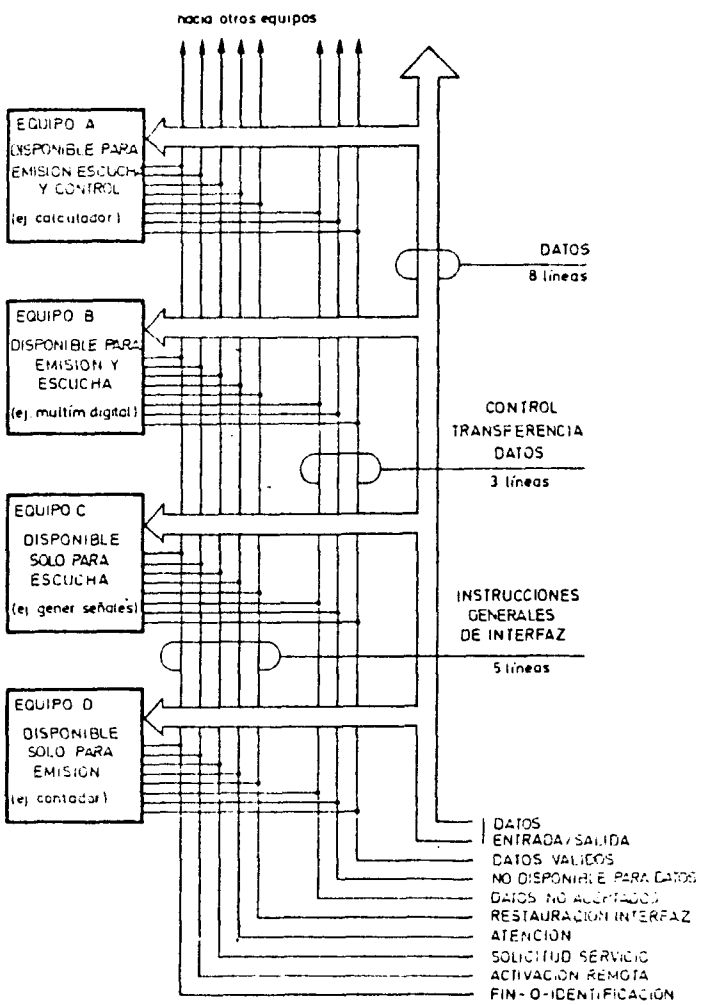
- bus de datos (8)
- bus de control de transferencias (3)
- bus de control general de la interconexión (5)

Estas senales pueden verse en la figura A.9.

3.1.a - bus de datos.- Un conjunto de ocho lineas de datos permiten transmitir por el bus, de octeto en octeto. Estas lineas se denominan DIO1-DIO8.



- fig. A.8 -



- fig. A.9 -

3.1.b - bus de control de transferencia.- Consta de tres senales, usadas para efectuar la transferencia de cada octeto entre el locutor y el oyente a través de las lineas DIOn. La denominación y descripción de estas senales es la siguiente:

- DAV (Data VAlid).- Es emitida por el locutor de la transferencia e indican que los datos son estables en el bus.

- NRFD (Not Ready For Data).- Es emitida por el oyente de la transferencia e indica que aun no está listo para recibir nuevos datos.

- NDAC (Not Data ACcepted).- Es emitida por el oyente e indica al locutor que debe mantener los datos en el bus, porque aun no han sido almacenados.

3.1.c - bus de control general de la interconexión.- Comprende cinco senales, que son empleadas para mantener un flujo ordenado de información a traves del bus. Son las siguientes:

- ATN (Attention).- Es empleada por el controlador del bus e indica a todos los demás dispositivos que se está enviando un mensaje de interés general.

- IFC (Interface Clear).- El controlador indica al resto de los dispositivos que deben volver al estado inicial o de reposo.

- SRQ (Service Request).- Los dispositivos no controladores usan esta linea para indicar al controlador su deseo de utilizar el bus para hacer una transferencia de datos.

- REN (Remote Enable).- El controlador indica a los dispositivos seleccionados que deben ignorar el control local, panel frontal o similar, para obedecer al control remoto recibido a través del bus.

- EOI (End Or Identify).- Puede ser activada por el dispositivo locutor o por el controlador. En el primer caso indica el fin de la transmisión de un bloque de datos. En el segundo, el controlador indica a los dispositivos que han pedido servicio que se identifiquen.

3.2 - especificaciones mecánicas.

El tipo de conector recomendado en la norma IEEE es de tipo trapezoidal de 24 contactos, transmitiendose las senales en lógica negativa (1=bajo<=0.8 v.; 0=alto>=2.0 v.).

3.3 - funciones a realizar por los dispositivos conectables al bus.

Para que un dispositivo sea capaz de interpretar una senal o conjunto de ellas es preciso que se le haya dotado de los circuitos necesarios para analizarlas y responder en función de ellas. Estos circuitos activarán o no unas senales, locales o globales, en función de las senales que interpreten.

Según esto, un dispoditivo se considera descompuesto en diez funciones distintas, cada una de las cuales es relativamente independiente de las demás y se puede analizar separadamente. El anàlisis detallado de cada una de estas funciones requeriria la transcripción literal de la información contenida en los folletos del IEEE, lo que nos apartaria del objetivo inicial de dar una visión lo más clara y resumida posible de las posibilidades de este bus.

Las funciones que puede realizar un dispositivo conectado al bus son las indicadas en la figura A.10.

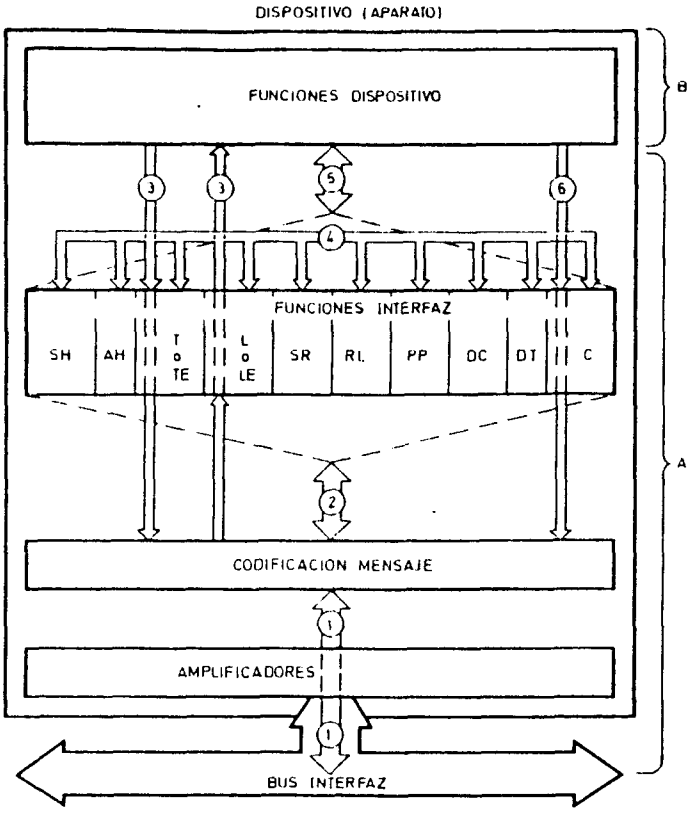
3.3.1 - Controlador (Controller=C).- El dispositivo que realiza las funciones de controlador es aquel encargado de inicializar todas las comunicaciones que se establezcan a través del bus.

En el bus debe haber siempre un dispositivo que reciba las peticiones de uso del bus, ya sean de un locutor o de un oyente. Este dispositivo analizará las peticiones de servicio, averiguando el o los dispositivos que las han realizado y autoriz7andoles el uso del bus por orden de prioridad, según los criterios con los que se haya programado el controlador.

El controlador tiene además la capacidad de enviar órdenes a todos los dispositivos conectados, modificando el estado interno de estos. En un momento determinado sólo puede haber un dispositivo ejerciendo las funciones de controlador.

3.3.2 - Locutor (Talker=T o TE).- La función de locutor permite a un dispositivo transmitir información a otros dispositivos conectados al bus. Un dispositivo con capacidad para realizar esta función sólo podrá ejercitarla cuando haya recibido del controlador la orden de transmitir un mensaje.

Todo dispositivo capaz de actuar como locutor tiene asignada una dirección, que le diferencia de los demás. Esta dirección se llama dirección de locutor y puede transmitirse en uno o dos octetos (5 a 10 bits). En el segundo caso la función de locutor se complica, llamándose entonces locutor extendido (Talker Extended = TE).



(a)

Función del interfaz	*Símbolo*	*Caminos empleados por la función*
Protocolo surtidor	SH	1, 2, 4, 5
Protocolo receptor	AH	1, 2, 4, 5
Locutor o locutor extendido	T o TE	1, 2, 3, 4, 5
Oyente u oyente extendido	L ó LE	1, 2, 3, 4, 5
Petición de servicio	SR	1, 2, 4, 5
Control remoto o local	RL	1, 2, 4, 5
Encuesta paralelo	PP	1, 2, 4, 5
Borrado del dispositivo	DC	1, 2, 4, 5
Disparo del dispositivo	DT	1, 2, 4, 5
Controlador	C	1, 2, 4, 5, 6

(b)

Descomposición funcional de un dispositivo conectado al bus IEEE-488 a) Descomposición en bloques del dispositivo completo A= Capacidad definidad por esta norma B = Capacidad definida por el diseñador 1 = Lineas del bus. 2 = Mensajes remotos del interfaz, de y hacia las funciones del interfaz. 3 = Mensajes dependientes del dispositivo, de y hacia las funciones del dispositivo. 4 = Relaciones entre los estados de las funciones del interfaz. 5 = Mensajes locales entre las funciones del dispositivo y las funciones del interfaz. (Los mensajes hacia las funciones del interfaz están definidos, los mensajes desde las funciones del interfaz dependen de las elecciones hechas por el diseñador). 6 = Mensajes remotos del interfaz enviados por las funciones del dispositivo que actúa como controlador. b) Repertorio de funciones del interfaz

- fig. A.10 -

3.3.3 - Oyente (Listener=L o LE).- La función de oyente capacita al dispositivo que la ejercita para recibir mensajes a través del bus. Estos mensajes serán transmitidos por el dispositivo que esté ejerciendo en ese momento la función de locutor.

3.3.4 - Protocolo Surtidor (Source Handshake = SH).- Esta función permite al dispositivo que la ejercita enviar un dato cualquiera por el bus. La función de locutor sólo capacita al dispositivo para adquirir el control del bus, como dispositivo emisor de mensajes. Se necesita esta función (SH) para controlar el protocolo de las senales que asegura la llegada del dato a los demás dispositivos.

3.3.5 - Protocolo Receptor (Acceptor Handshake = AH).- Esta función permite al dispositivo recibir los datos transmitidos por elbus.

3.3.6 - Petición de servicio (Service Request = SR).- Permite al dispositivo dotado con ella, solicitar asincronamente del controlador el uso de bus.

3.3.7 - Encuesta en paralelo (Parallel Poll = PP).- Esta función proporciona al dispositivo la capacidad de presentar un bit de estado al controlador sin haber sido direccionado como locutor.

3.3.8 - Borrado del dispositivo (Device Clear = DC).- Esta función permite inicializar el dispositivo, individualmente o como parte de un grupo de ellos.

3.3.9 - Disparo del dispositivo (Device Trigger = DT).- Permite poner en marcha las funciones básicas del dispositivo.

3.3.10 - Control remoto o local (Remote Local = RL).- Permite al dispositivo seleccionar entre dos fuentes de información de entrada. La función indica al dispositivo si debe usar la información recibida del panel frontal, local, o la recibida a través de la interconexión al bus, remota.

3.4 - mensajes transmitidos por el bus.

A continuación daré una breve descripción de los mensajes que pueden ser transmitidos por el bus, a título meramente informativo.

3.4.1 - Atención: ATN.

3.4.2 - Mandatos Universales: UCG (Universal Command Group).

- Limpiar dispositivo: DCL.
- Bloqueo local: LLO (Local Lockout).
- Desmontar la encuesta en paralelo: PPU (Parallel Poll Unconfigure).
- Inicio de encuesta serie: SPE (Serial Poll Enable).
- Fin de encuesta serie: SPD (Serial Poll Disable).
- Identificación: IDY (Identify).
- Limpiar la interconexión: IFC (Interface Clear).
- Validar control remoto: REN (Remote Enable).

3.4.3 - Mandatos selectivos: ACG (Addressed Command Group).

- Disparo de un grupo: GET (Group Execute Trigger).
- Pasar a local: GTL (Go to Local).

- Montar la encuesta paralelo: PPC (Parallel Poll Configure).

- Limpiar los dispositivos seleccionados: SDC (Selected Device Clear).

- Toma el control: TCT (Take Control).

3.4.4 - Direcciones: AD.

- Dirección de oyente: MLA (My Listen Address).

- No oir: UNL (Unlisten).

- Dirección de locutor: MTA (My Talker Address) u OTA (Other Talker Address).

- No hablar: UNT (Untalk).

3.4.5 - Mensajes Secundarios: SE.

- Dirección secundaria: MSA u OSA.

- Validación de encuesta paralelo: PPE (Parallel Poll Enable).

- Invalidar encuesta en paralelo: PPD (Parallel Poll Disable).

3.4.6 - Mensajes de estado: ST.

- Fin: END.

- Octeto de estado: STB o RQS.

- Petición de servicio: SRQ.

- Respuesta a encuesta paralelo: PPR.

3.4.7 - Mensajes de manipulación de datos: HS.

- Dato aceptado: DAC (Data Accepted).

- Dato válido: DAV (Data Valid).

- Listo para dato: RFD (Ready for Data).

3.4.8 - Mensajes dependientes del dispositivo: DD.

- Fin de una cadena de datos: EOS (End of String).

- Nulo: NUL.

Para una màs detallada informaciòn sobre el protocolo de interconexiòn definido por esta norma, se recomienda acudir a la bibliografia indicada al final del trabajo.

EQUIPOS TERMINALES DEL CIRCUITO DE DATOS.

Las funciones básicas de un ETCD (Equipo Terminal del Circuito de Datos) son las siguientes:

- dialogar con el ETD en el establecimiento, mantenimiento y terminación de una comunicación.

- transformar el mensaje de datos que recibe del ETD en una senal compatible con la linea de transmisión utilizada.

- reconvertir la senal recibida de la linea de transmisión en un mensaje de datos compatible con el ETD.

La inserción del ETD en el conjunto se realiza mediante dos interfaces normalizados:

- interface ETD/ETCD, a través del cual se realiza la función de dialogo.

- interface ETCD/linea, con unas caracteristicas impuestas por el tipo de linea y por la naturaleza de la senal a transmitir.

La conversión de senales se puede realizar de dos formas diferentes:

- por codificación.- El tren de datos recibido del ETD se transforma en otro compatible con la linea de transmisión. La transmisión se realiza en banda base y en los circuitos para transmisión sincrona a velocidades altas (a partir de 4800 bits/s) este proceso se denomina aleatorización, y precede a la modulación.

- por modulación.- El tren de datos entrante genera una señal analógica, compatible con la línea de transmisión, a base de modificar, en función de la señal de entrada, alguno de los parámetros que definen una onda sinusoidal pura de la forma:

$$A * COS (\omega t - \#)$$

Este proceso da lugar a tres sistemas básicos de modulación:

- de amplitud o ASK
- de frecuencia o FSK
- de fase o PSK

En cuanto a la rama de recepción, la reconversión de las señales procedentes de la línea de transmisión se realiza en el ETCD mediante uno o varios de los siguientes procesos:

- demodulación.- Es el proceso inverso a la modulación y consiste en reconstruir, a partir de la señal recibida, el tren de datos que la originó. Existen ciertos problemas de distorsión y sincronismo, aunque la demodulación puede ser coherente o no coherente, según que el receptor posea o no una referencia de la onda portadora con la cual ponerse en fase.

- recepción en banda base.- En este proceso, la señal recibida, distorsionada, se transforma en una sucesión de símbolos bien definidos.

- decodificación.- Es la operación inversa a la codificación.

Todos estos procesos se realizan en un único conjunto llamado MODEM, contracción de MOdulador-DEModulador.

Desde un punto de vista puramente técnico, no cabe duda

de que existe una infinidad de soluciones a la hora de disenar un MODEM, pero, con el fin de facilitar la instalación de circuitos standar, el CCITT ha normalizado una serie de MODEMs que cubren perfectamente la totalidad de las necesidades presentadas hasta hoy.

Esta normalización define y fija, para cada tipo de MODEM, una serie de caracteristicas de tal forma que pueden conectarse entre si MODEMs de diferentes constructores que han resuelto el problema con tecnologias muy distintas.

Como se puede observar en la figura A.11, el conjunto de MODEMs normalizados cubre, con algunos solapes, casi todos los valores asignados a los dos parámetros fundamentales que definen un tipo de MODEM:

- velocidad
- tipo de linea de transmisión.

MODO DE TRANSMISIÓN		ASÍNCRONA		ASÍNC./SÍNCR.		SÍNCRONA				
VELOCIDAD BITS/S / TIPO DE LÍNEA		≤ 200	≤ 300	≤ 600	≤ 1.200	2.400	4 800	9 600	19 200	48 72
RED AUTOMÁTICA CONMUTADA (2H)			V-20 V-21	V-22 V-23	V-26 bis	V-27 ter				
LÍNEA P. a P. CALIDAD M-1010	2H		V-21	V-22 V-23		V-27 bis				
	4H			V-23		V-27 bis				
LÍN. MULTIPUNTO CALIDAD (4H)	M-1010			V-23						
	M-1020					V-27 bis				
LÍNEA P. a P. CALIDAD M-1020 (4H)						V-26 V-27 V-27 bis				
							V-29			
GRUPO PRIMARIO										V-38
LÍNEA TELEGRÁFICA		Adaptador impulsos telegráficos								
PARES METÁLICOS		MODEMS EN BANDA BASE (NO NORMALIZADOS)								

- fig. A.11 -

APENDICE III

```
ASM80 MACBA0.ASM MOD85


ISIS-II 8080/8085 MACRO ASSEMBLER, V4.0        MODULE   PAGE    1


 LOC  OBJ          LINE       SOURCE STATEMENT

                      1 ;    ************************************************
                      2 ;    *                                              *
                      3 ;    *       SOFTWARE DEL TERMINAL MACBA-0:         *
                      4 ;    *       ------------------------------         *
                      5 ;    *                                              *
                      6 ;    *       INICIALIZACION EN ENSAMBLADOR          *
                      7 ;    *       -----------------------------          *
                      8 ;    *                                              *
                      9 ;    *     Esta parte del programa para el terminal *
                     10 ;    *  MACBA-0  se encarga  de la localizacion  de  *
                     11 ;    *  los  saltos a las  diferentes subrutinas de  *
                     12 ;    *  interrupcion soportables por el 8085.        *
                     13 ;    *                                              *
                     14 ;    ************************************************
                     15 ;
                     16 ;
                     17 ;
                     18 ;            PROGRAMA              TIPO DE INTERRUPCION
                     19 ;            --------              --------------------
                     20 ;
                     21 ;
                     22 ;
                     23          EXTRN   BEGIN1,INT65,INT55
0000                 24          ORG     0;                    * RST 0
0000 F3              25          DI
0001 C30000    E     26          JMP     BEGIN1
0003                 27          ORG     08H;                  * RST 1
0008 C9              28          RET
0010                 29          ORG     10H;                  * RST 2
0010 C9              30          RET
0018                 31          ORG     18H;                  * RST 3
0018 C9              32          RET
0020                 33          ORG     20H;                  * RST 4
0020 C9              34          RET
0024                 35          ORG     24H;                  * TRAP
0024 C9              36          RET
0028                 37          ORG     28H;                  * RST 5
0028 C9              38          RET
002C                 39          ORG     2CH;                  * RST 5.5
002C F3              40          DI
002D C34000          41          JMP     SUB55
0030                 42          ORG     30H;                  * RST 6
0030 C9              43          RET
0034                 44          ORG     34H;                  * RST 6.5
0034 F3              45          DI
0035 C34C00          46          JMP     SUB65
0038                 47          ORG     38H;                  * RST 7
0038 C9              48          RET
003C                 49          ORG     3CH;                  * RST 7.5
003C C9              50          RET
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

                     51
                     52
                     53
                     54 ;       ****************************************************
                     55 ;       *                                                  *
                     56 ;       *     Llamadas a las subrutinas de interrupcion    *
                     57 ;       *   realizadas en PL/M 80                          *
                     58 ;       *                                                  *
                     59 ;       ****************************************************
                     60 ;
                     61 ;
  0040               62          ORG     40H
  0040 E5            63 SUB55:   PUSH    H
  0041 D5            64          PUSH    D
  0042 C5            65          PUSH    B
  0043 F5            66          PUSH    PSW
  0044 CD0000    E   67          CALL    INT55
  0047 0E19          68          MVI     C,19H
  0049 C35800        69          JMP     BACK
  004C E5            70 SUB65:   PUSH    H
  004D D5            71          PUSH    D
  004E C5            72          PUSH    B
  004F F5            73          PUSH    PSW
  0050 CD0000    E   74          CALL    INT65
  0053 0E18          75          MVI     C,18H
  0055 C35800        76          JMP     BACK
  0058 79            77 BACK:    MOV     A,C
  0059 30            78          SIM
  005A F1            79          POP     PSW
  005B C1            80          POP     B
  005C D1            81          POP     D
  005D E1            82          POP     H
  005E FB            83          EI
  005F C9            84          RET
                     85          END

PUBLIC SYMBOLS


EXTERNAL SYMBOLS
BEGIN1 E 0000    INT55  E 0000    INT65  E 0000

USER SYMBOLS
BACK   A 0058    BEGIN1 E 0000    INT55  E 0000
INT65  E 0000    SUB55  A 0040    SUB65  A 004C

ASSEMBLY COMPLETE,   NO ERRORS
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TERMINAL
NO OBJECT MODULE REQUESTED
COMPILER INVOKED BY:  PLM80 MACBA.TER WORKFILES(:F0:,:F0:) NOOBJECT


            $TITLE('SOFTWARE DE SOPORTE DEL TERMINAL MACBA-0')
            $DATE(JUNIO-86)
            $PAGELENGTH(66)
            /**/
            /*******************************************************/
            /*                                                     */
            /*         SOFTWARE DEL TERMINAL MACBA-0:              */
            /*                                                     */
            /*       -Autor:                                       */
            /*              Manuel del Castillo Barrios            */
            /*       -Fecha:                                       */
            /*                    JUNIO - 1986                     */
            /*                                                     */
            /*       Este programa constituye el modulo principal  */
            /*  del  apartado software;  ya que incluye todas las  */
            /*  subrutinas  necesarias para  la  completa gestion  */
            /*  del sistema.                                       */
            /*                                                     */
            /*******************************************************/


  1         TERMINAL: DO;


                /*************************************/
                /*                                   */
                /*  DECLARACION DE MACROS GENERALES  */
                /*                                   */
                /*************************************/


  2   1         DECLARE   LIT   LITERALLY   'LITERALLY',
                            DEC       LIT       'DECLARE',
                            TRUE      LIT       '0FFH',
                            FALSE     LIT       '0',
                            FOREVER   LIT       'WHILE TRUE',
                        BOOLEAN  LIT       'BYTE';


                /*************************************/
                /*                                   */
                /*        DECLARACION   DE           */
                /*                                   */
                /* CONSTANTES RELACIONADAS CON EL PPI */
                /*                                   */
                /*************************************/


  3   1         DEC       CMD55     LIT       '13H',
                            MODE55    LIT       '0AFH',
                            SET55     LIT       '05H',
                            STATUS55  LIT       '12H',
                            KBPORT    LIT       '11H',
                            PRTPORT   LIT       '10H';
```

```
$EJECT


                /****************************************/
                /*                                      */
                /*        DECLARACION   DE              */
                /*                                      */
                /* CONSTANTES RELACIONADAS CON LA MUART */
                /*                                      */
                /****************************************/


4   1           DEC       MCMD1     LIT     '20H',
                          MCMD2     LIT     '21H',
                          MCMD3     LIT     '22H',
                          MMODE     LIT     '23H',
                          MP1CMD    LIT     '24H',
                          MSETINT   LIT     '25H',
                          MRESETINT LIT     '26H',
                          (MTX,MRX) LIT     '27H',
                          MPORT1    LIT     '28H',
                          (MP3ELL,MPORT2) LIT     '29H',
                          MTIM1     LIT     '2AH',
                          MTIM2     LIT     '2BH',
                          MTIM3     LIT     '2CH',
                          MTIM4     LIT     '2DH',
                          MTIM5     LIT     '2EH',
                          (MMODIF,MSTAT)  LIT     '2FH';
5   1           DEC       ENQ       LIT     '05H',
                          ACK       LIT     '06H',
                          BREAK     LIT     '0E4H',
                          BELL      LIT     '07H',
                          BELLMASK  LIT     '0EFH',
                          DSR       LIT     '02H',
                          RESETBELL LIT     '10H',
                          RBF       LIT     '40H',
                          TBE       LIT     '20H';
```

```
$EJECT


                /****************************************/
                /*                                      */
                /*        DECLARACION   DE              */
                /*                                      */
                /* CONSTANTES RELACIONADAS CON EL DMAC */
                /*                                      */
                /****************************************/


6   1           DEC       (DMAMOD,DMASTATUS) LIT '08H',
                          DMA0AD    LIT     '0',
                          DMA0TC    LIT     '1',
                          DMA1AD    LIT     '2',
                          DMA1TC    LIT     '3',
                          DMA2AD    LIT     '4',
                          DMA2TC    LIT     '5',
                          DMA3AD    LIT     '6',
                          DMA3TC    LIT     '7';



                /****************************************/
                /*                                      */
                /*         DECLARACION   DE             */
                /*                                      */
                /* CONSTANTES RELACIONADAS CON EL CRTC */
                /*                                      */
                /****************************************/


7   1           DEC     (CRTCMD,CRTSTATUS) LIT '31H',
                          CRTPARM   LIT     '30H',
                          TOPPANT   LIT     '1000H',
                          BOTPANT   LIT     '1EFFH',
                          ULTIMO    LIT     '1F00H',
                          CENTRO    LIT     '1730H',
                          CURBOT    LIT     '18H',
                          CR        LIT     '0DH',
                          LF        LIT     '0AH';
```

```
$EJECT

                /****************************************/
                /*                                      */
                /*         DECLARACION   DE             */
                /*                                      */
                /*       CONSTANTES    VARIAS           */
                /*                                      */
                /****************************************/


B   1           DEC       PRTACK            LIT '10H',
                          PRTBUSY           LIT '20H',
                          PRT$BUF$LEN       LIT '100',
                          LEN$LINEA         LIT '50H',
                          LOCAL$FLAG        LIT '0FH',
                          LINE$FLAG         LIT '0F0H',
                          NUM$FUNC          LIT '11',
                          NUM$ESK           LIT '19',
                          NUM$ESP           LIT '4',
                          PARAR             LIT '13H',
                          SEGUIR            LIT '11H';
9   1            DEC      JUEGODOS     LIT  '04H',
                          JUEGOUNO     LIT  '0B2H',
                          SUBRAYAR$ON  LIT  '20H',
                          SUBRAYAR$OFF LIT  '96H',
                          FLASH$ON     LIT  '02H',
                          FLASH$OFF    LIT  '0B4H',
                          INVERSE$ON   LIT  '10H',
                          INVERSE$OFF  LIT  '0A6H';
```

```
                $EJECT


                /*************************************/
                /*                                   */
                /*   CONSTANTES INCLUIDAS EN CODIGO  */
                /*                                   */
                /*************************************/


10   1          DEC       TABLA1(NUM$FUNC) BYTE DATA(
                          0AH /*LF*/,0CH /*FF*/,0DH /*CR*/,
                          08H /*BS*/,1BH /*ESC*/,03H /*BRK*/,
                          09H /*TAB*/,15H /*SCR*/,10H /*OPENPRT*/,
                          0FH /*CLOSEPRT*/,7FH /*RB*/);
11   1          DEC       TABLA2(NUM$ESK) BYTE DATA(
                          63H /*Esc c: INHIBIR TECLADO   */,
                          62H /*Esc b: HABILITAR TECLADO */,
                          42H /*Esc B: CURSOR ABAJO      */,
                          4AH /*Esc J: CLEAR DISPLAY     */,
                          4BH /*Esc K: CLEAR LINE        */,
                          41H /*Esc A: CURSOR ARRIBA     */,
                          43H /*Esc C: CURSOR DERECHA    */,
                          44H /*Esc D: CURSOR IZQUIERDA  */,
                          48H /*Esc H: HOME              */,
                          5AH /*Esc Z: SCROLL UP         */,
                          59H /*Esc Y: SCROLL DOWN       */,
                          58H /*Esc X: JUEGO CARAC. DOS  */,
                          57H /*Esc W: JUEGO CARAC. UNO  */,
                          56H /*Esc V: SUBRAYADO  ON     */,
                          55H /*Esc U: SUBRAYADO  OFF    */,
                          54H /*Esc T: FLASH   ON        */,
                          53H /*Esc S: FLASH   OFF       */,
                          52H /*Esc R: INVERSE   ON      */,
                          51H /*Esc Q: INVERSE   OFF     */);
12   1          DEC       TABLA3(NUM$ESP) BYTE DATA(
                          0AH /* LF */,0DH /* CR */,08H /* BS */,
                          09H /* TAB */);
13   1          DEC       TITULO(*) BYTE DATA('TERMINAL',1BH,'TMACBA0',1BH,'SPREPARADO'
                          ,0DH,0AH,'VERSION      1.8',0DH,0AH,'&');
14   1          DEC       MODEM(*) BYTE DATA(1BH,'T','CONECTAR EL MODEM',1BH,'S','&');
15   1          DEC       TEXLOC(*) BYTE DATA('LOCAL MODE','&');
16   1          DEC       TEXLIN(*) BYTE DATA('LINE  MODE','&');
```

```
$EJECT


                /**********************************/
                /*                                */
                /*       DECLARACION   DE         */
                /*                                */
                /*         VARIABLES              */
                /*                                */
                /**********************************/


17   1          DEC   (CURSX,CURSY,ESC,TEMP2) BYTE INITIAL (0,0,0,0),
                      (KBCHAR,USCHAR) BYTE INITIAL(0FFH,0FFH),
                      (PROTEK,MSTATUS,BAUD,TEMP) BYTE INITIAL (0,0,0,0),
                      (MPORT2$BUF,PRT$FLAG) BYTE INITIAL (0,0),
                  EFECTOS$ESPECIALES BYTE INITIAL (80H),
                      (LOCLIN) BYTE INITIAL (0FH),
                      (CURAD,LINEAD,TOPDIS) ADDRESS INITIAL (1000H,1000H,1000H),
                      CURCAR BASED CURAD BYTE ,
                      PRINCIPIO$LINEA BASED LINEAD BYTE,
                      (PRT$POINTER,TEMPAD) ADDRESS INITIAL (0,0),
                      BOTDIS ADDRESS INITIAL (1780H),
                      PRCHAR BASED PRT$POINTER BYTE,
                      BEG$PRT$BUFFER ADDRESS INITIAL (1F00H),
                  END$PRT$BUFFER ADDRESS INITIAL (1F63H);
18   1          DEC   (BLOCK1$AD,BLOCK1$TC,BLOCK2$AD,BLOCK2$TC) ADDRESS;
19   1          DEC   (SCFLAG,UFLAG,KBFLAG) BYTE INITIAL(0,0,0);
20   1          DEC   (TOPDOS,BOTDOS) ADDRESS;


        /*************************************************************************/
        /*                                                                       */
        /*              DECLARACION   DE   PROCEDIMIENTOS                        */
        /*                                                                       */
        /*************************************************************************/


        /*************************************************************/
        /*                                                           */
        /*  Procedimiento para  leer   la mascara de interrupcion    */
        /*                                                           */
        /*************************************************************/


21   1      R$MASK: PROCEDURE BYTE EXTERNAL;
22   2      END R$MASK;
```

```
                $EJECT

                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para colocar lamascara de interrupcion     */
                /*                                                           */
                /*************************************************************/


 23   1         S$MASK: PROCEDURE(MASK) EXTERNAL;
 24   2                 DECLARE MASK BYTE;
 25   2         END S$MASK;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento  para activar  el circuito encargado  de   */
                /*  producir el sonido de campanilla.                        */
                /*       (este circuito no esta incluido en el hardware)     */
                /*                                                           */
                /*************************************************************/


 26   1         BELL$SOUND: PROCEDURE;
 27   2                       MPORT2$BUF=(MPORT2$BUF AND BELL$MASK);
 28   2                       OUTPUT(MPBELL)=MPORT2$BUF;
 29   2                       MPORT2BUF=(MPORT2$BUF OR RESETBELL);
 30   2                       OUTPUT(MPBELL)=MPORT2$BUF;
 31   2         END BELL$SOUND;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento  auxiliar  para  la  deteccion  de la      */
                /*  llegada de un caracter desde el teclado.                 */
                /*                                                           */
                /*************************************************************/


 32   1         CARACTER$TECLADO: PROCEDURE BOOLEAN;
 33   2                           RETURN KBFLAG;
 34   2         END CARACTER$TECLADO;
```

```
                $EJECT

                /*************************************************************/
                /*                                                           */
                /*  Procedimiento auxiliar  para  la  deteccion  de  la      */
                /*  llegada de una caracter desde la UART.                   */
                /*                                                           */
                /*************************************************************/


35   1          CARACTER$UART: PROCEDURE BOOLEAN;
36   2                         RETURN UFLAG;
37   2          END CARACTER$UART;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para la transmision de un caracter a la    */
                /*  UART.                                                    */
                /*                                                           */
                /*************************************************************/


38   1          TRANSMITE$CARACTER: PROCEDURE;
39   2                              TEMP=(INPUT(MSTAT) AND TBE);
40   2                              DO WHILE TEMP=0;
41   3                                TEMP=(INPUT(MSTAT) AND TBE);
42   3                              END /* WHILE */;
43   2                              OUTPUT(MTX)=KBCHAR;
44   2          END TRANSMITE$CARACTER;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para pasar las coordenadas de cursor al    */
                /*  controlador de CRT.                                      */
                /*                                                           */
                /*************************************************************/


45   1          CARGAR$CURSOR: PROCEDURE;
46   2                         OUTPUT(CRTCMD)=80H;
47   2                         OUTPUT(CRTPARM)=CURSX;
48   2                         OUTPUT(CRTPARM)=CURSY;
49   2          END CARGAR$CURSOR;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para realizar el efecto de tabulacion.     */
                /*                                                           */
                /*************************************************************/


 50   1         TAB: PROCEDURE;
 51   2              TEMP=4;
 52   2              DO WHILE TEMP<4FH;
 53   3               IF CURSX<TEMP THEN DO;
 55   4                CURSX=TEMP;
 56   4                CALL CARGAR$CURSOR;
 57   4                TEMP=4FH;
 58   4               END /* DO IF */;
 59   3               ELSE TEMP=TEMP+5;
 60   3              END /* DO WHILE */;
 61   2         END TAB;
```

```
$EJECT


/*************************************************************/
/*                                                           */
/*  Procedimiento llamado por la interrupcion RST 6.5,       */
/*  que se encarga de reinicializar y recalcular los         */
/*  parametros que se han de pasar a los canales 2 y 3       */
/*  del controlador de DMA.                                  */
/*                                                           */
/*************************************************************/


62   1      INT65: PROCEDURE PUBLIC;
63   2             DEC NADA BYTE;
64   2             NADA=INPUT(CRTSTATUS);
65   2             OUTPUT(DMAMOD)=0;
66   2             OUTPUT(DMA2AD)=LOW(BLOCK1$AD);
67   2             OUTPUT(DMA2AD)=HIGH(BLOCK1$AD);
68   2             OUTPUT(DMA2TC)=LOW(BLOCK1$TC);
69   2             OUTPUT(DMA2TC)=HIGH(BLOCK1$TC);
70   2             OUTPUT(DMAMOD)=84H;
71   2             OUTPUT(DMA3AD)=LOW(BLOCK2$AD);
72   2             OUTPUT(DMA3AD)=HIGH(BLOCK2$AD);
73   2             OUTPUT(DMA3TC)=LOW(BLOCK2$TC);
74   2             OUTPUT(DMA3TC)=HIGH(BLOCK2$TC);
75   2             IF TOPDIS<=CENTRO THEN DO;
77   3               BLOCK1$AD=TOPDIS;
78   3               BLOCK1$TC=CENTRO-TOPDIS;
79   3               BLOCK2$AD=CENTRO+1;
80   3               BLOCK2$TC=BOTDIS-CENTRO+4FH;
81   3             END /* DO IF */;
82   2             ELSE DO;
83   3               BLOCK1$AD=TOPDIS;
84   3               BLOCK1$TC=(BOTPANT)-TOPDIS;
85   3               BLOCK2$AD=TOPPANT;
86   3               BLOCK2$TC=BOTDIS-TOPPANT+4FH;
87   3             END /* ELSE DO */;
88   2             BLOCK1$TC=BLOCK1$TC+8000H;
89   2             BLOCK2$TC=BLOCK2$TC+8000H;
90   2      END INT65;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento  llamado  por  la interrupcion RST 5.5;    */
                /*  que se encarga de la lectura del teclado.                */
                /*                                                           */
                /*************************************************************/


 91   1         INT55: PROCEDURE PUBLIC;
 92   2                 KBCHAR=(INPUT(KBPORT) AND 7FH);
 93   2                 IF PROTEK=TRUE THEN KBFLAG=FALSE;
 95   2                 ELSE KBFLAG=TRUE;
 96   2         END INT55;



                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para llenar de espacios (20h) la memoria   */
                /*  de pantalla.                                             */
                /*                                                           */
                /*************************************************************/


 97   1         BORRA$DISPLAY: PROCEDURE;
 98   2                        DO CURAD=TOPPANT TO (BOTPANT);
 99   3                          CURCAR=20H;
100   3                        END /* DO */;
101   2                        CURAD=TOPPANT;
102   2         END BORRA$DISPLAY;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para la deteccion de caracteres de         */
                /*  control.                                                 */
                /*                                                           */
                /*************************************************************/


103   1         NO$CONTROL$CHAR: PROCEDURE BOOLEAN;
104   2                          DO TEMP=0 TO NUM$ESP-1;
105   3                           IF KBCHAR=TABLA3(TEMP) THEN RETURN TRUE;
        107   3                                 END /* DO */;
        108   2                                 DO TEMP=0 TO NUM$FUNC-1;
        109   3                                  IF KBCHAR=TABLA1(TEMP) THEN RETURN FALSE;
111   3                          END /* DO */;
112   2                          IF ESC THEN RETURN FALSE;
114   2                          RETURN TRUE;
115   2         END NO$CONTROL$CHAR;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento de deteccion del modo de comunicacion:     */
                /*                    LOCAL  o  REMOTO                       */
                /*                                                           */
                /*************************************************************/


116   1         LINE: PROCEDURE BOOLEAN;
117   2               IF LOCLIN=LOCAL$FLAG THEN RETURN FALSE;
119   2               ELSE RETURN TRUE;
120   2         END LINE;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para el envio de un caracter de parada     */
                /*                                                           */
                /*************************************************************/


121   1         PARO: PROCEDURE;
122   2               TEMP2=KBCHAR;
123   2               KBCHAR=PARAR;
124   2               CALL TRANSMITE$CARACTER;
125   2               KBCHAR=TEMP2;
126   2         END PARO;
```

```
                $EJECT

                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para el  envio de un caracter de puesta    */
                /*  en marcha.                                               */
                /*                                                           */
                /*************************************************************/


127   1         MARCHA: PROCEDURE;
128   2                 TEMP2=KBCHAR;
129   2                 KBCHAR=SEGUIR;
130   2                 CALL TRANSMITE$CARACTER;
131   2                 KBCHAR=TEMP2;
132   2         END MARCHA;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para  calcular la direccion de comienzo    */
                /*  de la linea en curso.                                    */
                /*                                                           */
                /*************************************************************/


133   1         CALCULA$DIR: PROCEDURE ADDRESS;
134   2                      TEMPAD=TOPDIS+CURSY*LEN$LINEA;
135   2                      IF TEMPAD>=ULTIMO THEN TEMPAD=TEMPAD-ULTIMO+TOPPANT;
137   2                      RETURN TEMPAD;
138   2         END CALCULA$DIR;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para 'subir' la pantalla.                  */
                /*                                                           */
                /*************************************************************/


139   1         ROLLUP: PROCEDURE;
140   2                  TOPDOS=TOPDIS+LEN$LINEA;
141   2                  BOTDOS=BOTDIS+LEN$LINEA;
142   2                  IF TOPDOS>=ULTIMO THEN TOPDOS=TOPPANT;
144   2                  IF BOTDOS>=ULTIMO THEN BOTDOS=TOPPANT;
146   2                  DISABLE;
147   2                  TOPDIS=TOPDOS;
148   2                  BOTDIS=BOTDOS;
149   2                  ENABLE;
150   2         END ROLLUP;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para "bajar" la pantalla                   */
                /*                                                           */
                /*************************************************************/


151   1         ROLLDOWN: PROCEDURE;
152   2                   TOPDOS=TOPDIS-LEN$LINEA;
153   2                   BOTDOS=BOTDIS-LEN$LINEA;
154   2                   IF TOPDOS<TOPPANT THEN TOPDOS=ULTIMO-LEN$LINEA;
156   2                   IF BOTDOS<TOPPANT THEN BOTDOS=ULTIMO-LEN$LINEA;
158   2                   DISABLE;
159   2                   TOPDIS=TOPDOS;
160   2                   BOTDIS=BOTDOS;
161   2                   ENABLE;
162   2         END ROLLDOWN;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para borrar la linea en curso.             */
                /*                                                           */
                /*************************************************************/


163   1         CLLIN: PROCEDURE;
164   2                PRINCIPIO$LINEA=0F0H;
165   2         END CLLIN;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para leer la UART.                         */
                /*                                                           */
                /*************************************************************/


166   1         READ$UART: PROCEDURE;
167   2                     TEMP=(INPUT(MSTAT) AND RBF);
168   2                     IF TEMP=0 THEN UFLAG=FALSE;
169   2                     ELSE DO;
171   3                      USCHAR=(INPUT(MRX)AND 07FH);
172   3                      IF USCHAR=ENQ THEN DO;
174   4                       UFLAG=FALSE;
175   4                       CALL S$MASK(19H);
176   4                       TEMP2=KBCHAR;
177   4                       KBCHAR=ACK;
178   4                       CALL TRANSMITE$CARACTER;
179   4                       KBCHAR=TEMP2;
180   4                      END /* DO IF */;
181   3                      ELSE DO;
182   4                       IF USCHAR=07FH THEN UFLAG=FALSE;
184   4                       ELSE UFLAG=TRUE;
185   4                      END /* ELSE */;
186   3                     END /* ELSE DO */;
187   2         END READ$UART;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para ejecutar un linefeed.                 */
                /*                                                           */
                /*************************************************************/


188   1         LINEFEED: PROCEDURE;
189   2                   IF CURSY<CURBOT THEN CURSY=CURSY+1;
191   2                   ELSE DO;
192   3                    CALL ROLLUP;
193   3                    LINEAD=CALCULA$DIR;
194   3                    CALL CLLIN;
195   3                   END /* DO ELSE */;
196   2                   CALL CARGAR$CURSOR;
197   2         END LINEFEED;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para ejecutar un retorno de carro.         */
                /*                                                           */
                /*************************************************************/


198   1         CARRIAGE$RETURN: PROCEDURE;
199   2                          CURSX=0;
200   2                          CALL CARGAR$CURSOR;
201   2         END CARRIAGE$RETURN;



                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para mover el cursor hacia la izda.        */
                /*                                                           */
                /*************************************************************/


202   1         CURLEFT: PROCEDURE;
203   2                  IF CURSX>0 THEN DO;
205   3                   CURSX=CURSX-1;
206   3                   CALL CARGAR$CURSOR;
207   3                  END /* DO IF */;
208   2                  ELSE DO;
209   3                   IF CURSY>0 THEN DO;
211   4                    CURSY=CURSY-1;
212   4                    CURSX=4FH;
213   4                    CALL CARGAR$CURSOR;
214   4                   END /* DO */;
215   3                  END /* ELSE DO */;
216   2         END CURLEFT;


                /*************************************************************/
                /*                                                           */
                /*  Procediiento para borrar el caracter anterior.           */
                /*                                                           */
                /*************************************************************/


217   1         BACKSPACE: PROCEDURE;
218   2                    CALL CURLEFT;
219   2         END BACKSPACE;
```

```
$EJECT

/*************************************************************/
/*                                                           */
/*  Procedimiento para iniciar una secuencia 'escape'.       */
/*                                                           */
/*************************************************************/

220  1     ESCAPE: PROCEDURE;
221  2             ESC=TRUE;
222  2     END ESCAPE;

/*************************************************************/
/*                                                           */
/*  Procedimiento para enviar un caracter de 'break'.        */
/*                                                           */
/*************************************************************/

223  1     SEND$BRK: PROCEDURE;
224  2               OUTPUT(MCMD3)=0E4H;
225  2     END SEND$BRK;

/*************************************************************/
/*                                                           */
/*  Procedimiento para deshabilitar el teclado.              */
/*                                                           */
/*************************************************************/

226  1     INHIBEKB: PROCEDURE;
227  2               PROTEK=TRUE;
228  2     END INHIBEKB;

/*************************************************************/
/*                                                           */
/*  Procedimiento para habilitar el teclado.                 */
/*                                                           */
/*************************************************************/

229  1     HABILITAKB: PROCEDURE;
230  2                 PROTEK=FALSE;
231  2     END HABILITAKB;
```

```
$EJECT


/*************************************************************/
/*                                                           */
/*  Procedimiento para posicionar el cursor en la esquina    */
                /*  superior izquierda de la pantalla, al principio de la  */
                /*  memoria de pantalla.                                   */
                /*                                                         */
/*************************************************************/


232   1     HOME: PROCEDURE;
233   2           TOPDIS=TOPPANT;
234   2           BOTDIS=TOPPANT+0780H;
235   2           CURSX=0;
236   2           CURSY=0;
237   2           CALL CARGAR$CURSOR;
238   2     END HOME;
```

```
$EJECT


/*************************************************************/
/*                                                           */
/*  Procedimiento para enviar un caracter a la impresora.    */
/*                                                           */
/*************************************************************/


239   1     PRINTER: PROCEDURE(CHAR4);
240   2              DEC CHAR4 BYTE;
241   2              INICIA$BUFFER: PROCEDURE;
242   3                              END$PRT$BUFFER=BEG$PRT$BUFFER;
243   3                              PRT$POINTER=BEG$PRT$BUFFER;
244   3                              PRCHAR=20H;
245   3              END INICIA$BUFFER;
246   2              SEND$TO$PRT: PROCEDURE;
247   3                             DO WHILE (INPUT(STATUS55)AND PRTBUSY)=1;
248   4                             ;
249   4                             END /* DO WHILE */;
250   3                             OUTPUT(PRTPORT)=PRCHAR;
251   3                             DO WHILE (INPUT(STATUS55)AND PRTACK)=0;
252   4                             ;
253   4                             END /* DO WHILE */;
254   3              END SEND$TO$PRT;
255   2              VACIA$BUFFER: PROCEDURE;
256   3                            IF LINE THEN CALL PARO;
258   3                            DO TEMPAD=BEG$PRT$BUFFER TO END$PRT$BUFFER-1;
259   4                             PRT$POINTER=TEMPAD;
260   4                             CALL SEND$TO$PRT;
261   4                            END /* DO */;
262   3                            IF LINE THEN DO;
264   4                             PRCHAR=0AH;
265   4                             CALL SEND$TO$PRT;
266   4                             CALL CARRIAGE$RETURN;
267   4                             CALL MARCHA;
268   4                            END/* DO IF*/;
269   3             END VACIA$BUFFER;
270   2             IF END$PRT$BUFFER<(BEG$PRT$BUFFER+PRT$BUF$LEN-1) THEN DO;
272   3              PRCHAR=CHAR4;
273   3              PRT$POINTER=PRT$POINTER+1;
274   3              END$PRT$BUFFER=PRT$POINTER;
275   3              IF CHAR4=0DH THEN DO;
277   4               CALL VACIA$BUFFER;
278   4               CALL INICIA$BUFFER;
279   4              END /* DO */;
280   3             END /* DO */;
281   2             ELSE DO;
282   3              CALL VACIA$BUFFER;
283   3              CALL INICIA$BUFFER;
284   3             END /* ELSE DO */;
285   2     END PRINTER;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento  para  realizar  la  impresion  de  un     */
                /*  caracter; detectando si este va hacia la pantalla  o     */
                /*  hacia la impresora.                                      */
                /*                                                           */
                /*************************************************************/


286   1         PRINT: PROCEDURE(CHAR3);
287   2                DEC CHAR3 BYTE;
288   2                IF SCFLAG=FALSE THEN DO;
290   3                 CURAD=CALCULA$DIR+CURSX;
291   3                 TEMP=CURCAR;
292   3                 CURCAR=CHAR3;
293   3                 IF CURSX+1>=LEN$LINEA THEN DO;
295   4                  CALL LINEFEED;
296   4                  CALL CARRIAGE$RETURN;
297   4                 END /* DO IF */;
298   3                 ELSE DO;
299   4                  CURSX=CURSX+1;
300   4                  CURAD=CURAD+1;
301   4                  IF TEMP=0F0H THEN CURCAR=TEMP;
303   4                  CALL CARGAR$CURSOR;
304   4                 END /* ELSE */;
305   3                END /* DO IF */;
306   2                ELSE SCFLAG=FALSE;
307   2         END PRINT;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para mover el cursor hacia abajo.          */
                /*                                                           */
                /*************************************************************/


308   1         CURDOWN: PROCEDURE;
309   2                  IF CURSY=CURBOT THEN RETURN;
311   2                  CURSY=CURSY+1;
312   2                  CALL CARGAR$CURSOR;
313   2                  LINEAD=CALCULA$DIR;
314   2                  IF PRINCIPIO$LINEA=0F0H THEN CALL CLLIN;
316   2         END CURDOWN;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para mover el cursor hacia arriba.         */
                /*                                                           */
                /*************************************************************/


317   1         CURUP: PROCEDURE;
318   2                IF CURSY=0 THEN RETURN;
320   2                CURSY=CURSY-1;
321   2                CALL CARGAR$CURSOR;
322   2         END CURUP;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para mover el cursor hacia la derecha.     */
                /*                                                           */
                /*************************************************************/


323   1         CURRIGHT: PROCEDURE;
324   2                   IF CURSX<4FH THEN DO;
326   3                    CURSX=CURSX+1;
327   3                    CALL CARGAR$CURSOR;
328   3                   END /* DO IF */;
329   2                   ELSE DO;
330   3                    IF CURSY<CURBOT THEN CURSY=CURSY+1;
332   3                    CURSX=0;
333   3                    CALL CARGAR$CURSOR;
334   3                   END /* ELSE DO */;
335   2         END CURRIGHT;
```

```
                $EJECT

                /*************************************************************/
                /*                                                           */
                /*  Procedimiento  para  limpiar  la  pantalla  desde la     */
                /*  posicion del curso en adelante de forma rapida.          */
                /*                                                           */
                /*************************************************************/


336   1         CLS: PROCEDURE;
337   2              LINEAD=CALCULA$DIR;
338   2              DO CURAD=LINEAD+CURSX TO LINEAD+4FH;
339   3               CURCAR=20H;
340   3              END /* DO */;
341   2              CURSY=CURSY+1;
342   2              LINEAD=LINEAD+LEN$LINEA;
343   2              DO WHILE LINEAD<ULTIMO;
344   3               PRINCIPIO$LINEA=0F0H;
345   3               LINEAD=LINEAD+LEN$LINEA;
346   3              END /* DO WHILE */;
347   2              CURSY=CURSY-1;
348   2         END CLS;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para abrir el canal de impresora.          */
                /*                                                           */
                /*************************************************************/


349   1         OPEN$PRT: PROCEDURE;
350   2                   PRT$FLAG=TRUE;
351   2         END OPEN$PRT;



                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para cerrar el canal de la impresora.      */
                /*                                                           */
                /*************************************************************/


352   1         CLOSE$PRT: PROCEDURE;
353   2                    PRT$FLAG=FALSE;
354   2                    CALL PRINTER(LF);
355   2                    CALL PRINTER(CR);
356   2         END CLOSE$PRT;
```

```
$EJECT

/*************************************************************/
/*                                                           */
/*  Procedimiento para establecer la mascara de efectos      */
/*  especiales de video. Comienza alguno de los efectos.     */
/*                                                           */
/*************************************************************/


357  1   ON$EFEKTS: PROCEDURE(MASK);
358  2             DEC MASK BYTE;
359  2             EFECTOS$ESPECIALES=(EFECTOS$ESPECIALES OR MASK);
360  2             CALL PRINT(EFECTOS$ESPECIALES);
361  2   END ON$EFEKTS;


/*************************************************************/
/*                                                           */
/*  Procedimiento para resetear alguno de los efectos        */
/*  especiales.                                              */
/*                                                           */
/*************************************************************/


362  1   OFF$EFEKTS: PROCEDURE(MASK);
363  2               DEC MASK BYTE;
364  2               EFECTOS$ESPECIALES=(EFECTOS$ESPECIALES AND MASK);
365  2               CALL PRINT(EFECTOS$ESPECIALES);
366  2   END OFF$EFEKTS;


/*************************************************************/
/*                                                           */
/*  Procedimiento para enviar un caracter directamente       */
/*  desde la memoria de pantalla hacia elordenador.          */
/*                                                           */
/*************************************************************/


367  1   SCR: PROCEDURE;
368  2        IF LINE THEN DO;
370  3         SCFLAG=TRUE;
371  3         IF CURSX+1<LEN$LINEA THEN DO;
373  4          CURAD=CALCULA$DIR+CURSX;
374  4          KBCHAR=CURCAR;
375  4          CURSX=CURSX+1;
376  4          CALL TRANSMITE$CARACTER;
377  4         END /* DO */;
378  3        END /* DO IF */;
379  2        ELSE CURSX=CURSX+1;
380  2        CALL CARGAR$CURSOR;
     381  2   END SCR;
```

```
$EJECT

/*************************************************************/
/*                                                           */
/*  Procedimiento de decodificacion de caracteres.           */
/*                                                           */
/*************************************************************/


382  1     ACTUA: PROCEDURE(CHAR2);
383  2            DEC CHAR2 BYTE;
384  2            IF PRT$FLAG THEN CALL PRINTER(CHAR2);
386  2            IF ESC=FALSE THEN DO;
388  3             DO TEMP=0 TO NUM$FUNC-1;
389  4              IF TABLA1(TEMP)=CHAR2 THEN GOTO DECODE1;
391  4             END /* DO */;
392  3         IF CHAR2=BELL THEN CALL BELL$SOUND;
394  3             IF CHAR2<20H THEN RETURN;
396  3             ELSE CALL PRINT(CHAR2);
397  3             RETURN;
398  3     DECODE1: DO CASE TEMP;
399  4               CALL LINEFEED;
400  4               DO;
401  5                CALL HOME;
402  5                CALL CLS;
403  5               END /* DO */;
404  4               CALL CARRIAGE$RETURN;
405  4               CALL BACKSPACE;
406  4               CALL ESCAPE;
407  4               CALL SEND$BRK;
408  4               CALL TAB;
409  4               CALL SCR;
410  4               CALL OPENPRT;
411  4               CALL CLOSEPRT;
412  4               CALL BACKSPACE;
413  4              END /* DO CASE */;
414  3             END /* DO */;
415  2             ELSE DO;
416  3          ESC=FALSE;
417  3              DO TEMP=0 TO NUM$ESK-1;
418  4               IF TABLA2(TEMP)=CHAR2 THEN GOTO DECODE2;
420  4              END /* DO */;
421  3              RETURN;
```

```
                $EJECT
422   3         DECODE2: DO CASE TEMP;
423   4                    CALL INHIBEKB;
424   4                    CALL HABILITAKB;
425   4                    CALL CURDOWN;
426   4                    CALL CLS;
427   4                    CALL CLLIN;
428   4                    CALL CURUP;
429   4                    CALL CURRIGHT;
430   4                    CALL CURLEFT;
431   4                    CALL HOME;
432   4                    CALL ROLLUP;
433   4                    CALL ROLLDOWN;
434   4                    CALL ON$EFEKTS(JUEGODOS);
435   4                    CALL OFF$EFEKTS(JUEGOUNO);
436   4                    CALL ON$EFEKTS(SUBRAYAR$ON);
437   4                    CALL OFF$EFEKTS(SUBRAYAR$OFF);
438   4                    CALL ON$EFEKTS(FLASH$ON);
439   4                    CALL OFF$EFEKTS(FLASH$OFF);
440   4                    CALL ON$EFEKTS(INVERSE$ON);
441   4                    CALL OFF$EFEKTS(INVERSE$OFF);
442   4                   END /* DO CASE */;
443   3                  END /* ELSE */;
444   2         END ACTUA;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento para imprimir un texto.                    */
                /*                                                           */
                /*************************************************************/


445   1         WRITE: PROCEDURE(TEXTAD);
446   2                DEC TEXTAD ADDRESS,
                           CHAR5 BASED TEXTAD BYTE;
447   2                DO WHILE CHAR5<>'&';
448   3                 CALL ACTUA(CHAR5);
449   3                 TEXTAD=TEXTAD+1;
450   3                END /* DO WHILE */;
451   2         END WRITE;
```

```
                $EJECT


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento  para conmutar de modo local a remoto y    */
                /*  viceversa.                                               */
                /*                                                           */
                /*************************************************************/


452   1         CONMUTA:  PROCEDURE;
453   2               IF (INPUT(MPORT1) AND 10H)<>0 THEN DO;
455   3                IF LOCLIN=LOCAL$FLAG THEN DO;
457   4                 CALL WRITE(.TEXLIN);
458   4                 LOCLIN=LINE$FLAG;
459   4                END /* DO IF */;
460   3                ELSE DO;
461   4                 CALL WRITE(.TEXLOC);
462   4                 LOCLIN=LOCAL$FLAG;
463   4                END /* ELSE */;
464   3                CALL CARRIAGE$RETURN;
465   3                CALL LINEFEED;
466   3                DO WHILE (INPUT(MPORT1) AND 10H)<>0;
467   4                END /* DO WHILE */;
468   3               END /* DO IF */;
469   2         END CONMUTA;


                /*************************************************************/
                /*                                                           */
                /*  Procedimiento de comprobacion del estado de la linea.    */
                /*                                                           */
                /*************************************************************/


470   1         COMP$LINE: PROCEDURE;
471   2                     IF (INPUT(MPORT1) AND DSR)=2  THEN DO;
473   3                     CALL WRITE(.MODEM);
474   3                     CALL LINEFEED;
475   3                     CALL CARRIAGE$RETURN;
476   3                     DO WHILE (INPUT(MPORT1) AND DSR)=2;
477   4                      ;
478   4                     END /* DO WHILE */;
479   3                     END /* DO IF */;
480   2         END COMP$LINE;
```

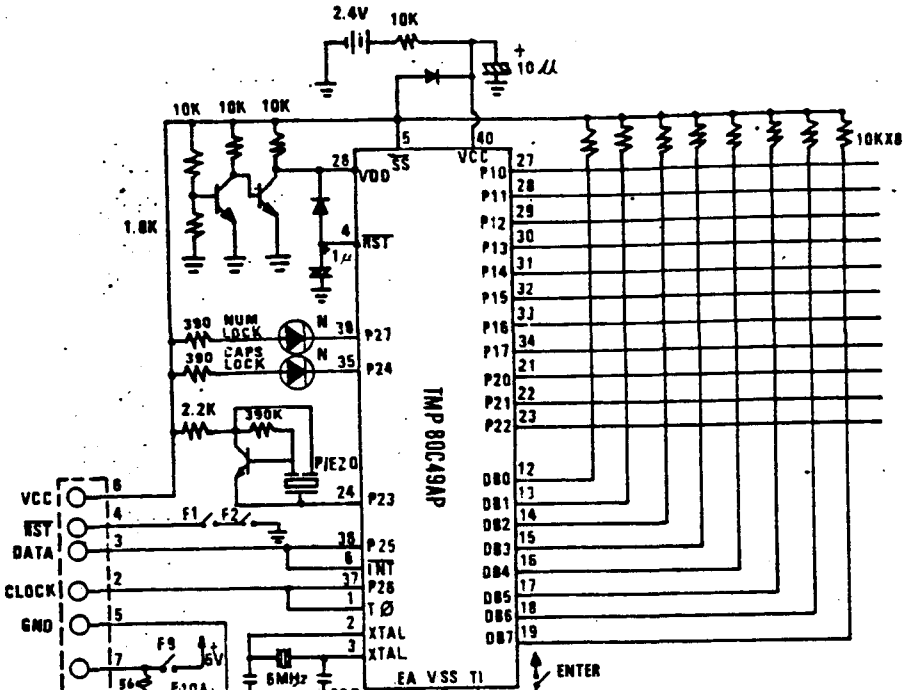
```
                $EJECT

                /*************************************************************/
                /*                                                           */
                /*  Procedimiento  de  llamada  a  la  subrutina  de         */
                /*  decodificacion.                                          */
                /*                                                           */
                /*************************************************************/


481   1         EJECUTA: PROCEDURE(CHAR);
482   2                  DEC CHAR BYTE;
483   2                  IF CHAR=0DH THEN CALL LINEFEED;
485   2                  CALL ACTUA(CHAR);
486   2         END EJECUTA;



                /*************************************************************/
                /*                                                           */
                /*              DECLARACION  DE  ETIQUETAS                   */
                /*                                                           */
                /*************************************************************/


487   1               DEC       BEGIN1    LABEL   PUBLIC;


                /*************************************************************/
                /*                                                           */
                /*      Comienzo del programa:                               */
                /*      Inicializacion de perifericos y variables.           */
                /*                                                           */
                /*************************************************************/


488   1         BEGIN1: CALL S$MASK(18H);
489   1         PIOC:   OUTPUT(CMD55)=MODE55;
490   1                 OUTPUT(CMD55)=SET55;
491   1         MUART:  OUTPUT(MCMD1)=24H;
492   1                 OUTPUT(MCMD3)=0E1H;
493   1                 OUTPUT(MMODE)=3AH;
494   1                 BAUD=INPUT(MPORT2) AND 0FH;
495   1                 IF BAUD<3 THEN BAUD=3;
497   1                 BAUD=BAUD+10H;
498   1                 OUTPUT(MCMD2)=BAUD;
499   1                 OUTPUT(MP1CMD)=01H;
500   1                 OUTPUT(MRESETINT)=0FFH;
501   1                 OUTPUT(MSETINT)=0H;
502   1                 OUTPUT(MMODIF)=03H;
```

```
                $EJECT
503   1         DMA:    OUTPUT(DMA2AD)=LOW(TOPDIS);
504   1                 OUTPUT(DMA2AD)=HIGH(TOPDIS);
505   1                 OUTPUT(DMA2TC)=0CFH;
506   1                 OUTPUT(DMA2TC)=87H;
507   1                 OUTPUT(DMA3AD)=LOW(TOPDIS);
508   1                 OUTPUT(DMA3AD)=HIGH(TOPDIS);
509   1                 OUTPUT(DMA3TC)=0CFH;
510   1                 OUTPUT(DMA3TC)=87H;
511   1                 OUTPUT(DMAMOD)=084H;
512   1         CRT:    OUTPUT(CRTCMD)=0;
513   1                 OUTPUT(CRTPARM)=4FH;
514   1                 OUTPUT(CRTPARM)=58H;
515   1                 OUTPUT(CRTPARM)=89H;
516   1                 OUTPUT(CRTPARM)=0DDH;
517   1                 CALL BORRA$DISPLAY;
518   1                 CALL CARGAR$CURSOR;
519   1                 OUTPUT(CRTCMD)=0E0H;
520   1                 OUTPUT(CRTCMD)=23H;
521   1                 CALL S$MASK(18H);
522   1                 USCHAR,KBCHAR=0FFH;
523   1                 PROTEK,ESC,CURSY,CURSX,SCFLAG,UFLAG,KBFLAG=0;
524   1                 PRT$FLAG,MPORT2$BUF,TEMP2,TEMP,BAUD,MSTATUS=0;
525   1                 EFECTOS$ESPECIALES=80H;
526   1                 LOCLIN=0FH;
527   1                 TOPDIS,LINEAD,CURAD=1000H;
528   1                 CURCAR=20H;
529   1                 TEMPAD,PRT$POINTER=0;
530   1                 BOTDIS=1780H;
531   1                 BEG$PRT$BUFFER=1F00H;
532   1                 END$PRT$BUFFER=1F63H;
533   1                 BLOCK2$AD,BLOCK1$AD=TOPDIS;
534   1                 BLOCK2$TC,BLOCK1$TC=87CFH;
535   1                 ENABLE;
```

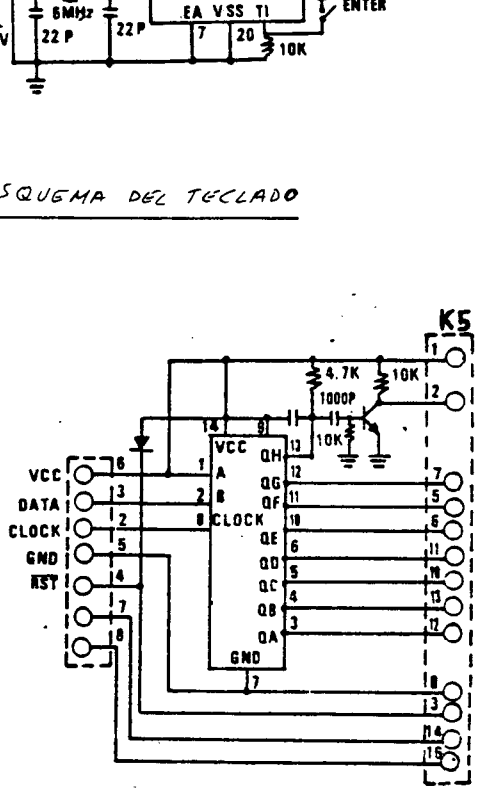
```
$EJECT

/*************************************************************/
/*                                                           */
/*                      BUCLE PRINCIPAL                      */
/*                                                           */
/*************************************************************/


536   1     BEGIN2: CALL LINEFEED;
537   1             CALL WRITE(.TITULO);
538   1             DO FOREVER;
539   2              CALL S$MASK(18H);
540   2              ENABLE;
541   2              CALL CONMUTA;
542   2              IF LINE THEN DO;
544   3               CALL COMP$LINE;
545   3               CALL READ$UART;
546   3               IF CARACTER$UART THEN DO;
548   4                UFLAG=FALSE;
549   4                CALL ACTUA(USCHAR);
550   4               END /* DO IF */;
551   3               ELSE DO;
552   4                CALL S$MASK(19H);
553   4                IF CARACTER$TECLADO THEN DO;
555   5                 IF NO$CONTROL$CHAR THEN CALL TRANSMITE$CARACTER;
557   5                 ELSE CALL EJECUTA(KBCHAR);
558   5                 KBFLAG=FALSE;
559   5                END /* DO */;
560   4               END /* ELSE */;
561   3              END /* DO IF  */;
562   2              ELSE DO;
563   3               CALL S$MASK(19H);
564   3               IF CARACTER$TECLADO THEN CALL EJECUTA(KBCHAR);
566   3               KBFLAG=FALSE;
567   3              END /* ELSE DO */;
568   2             END /* FOREVER  */;
569   1     END TERMINAL;


MODULE INFORMATION:

     CODE AREA SIZE     = 0A30H   2608D
     VARIABLE AREA SIZE = 0036H     54D
MAXIMUM STACK SIZE = 0010H     16D
1235 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

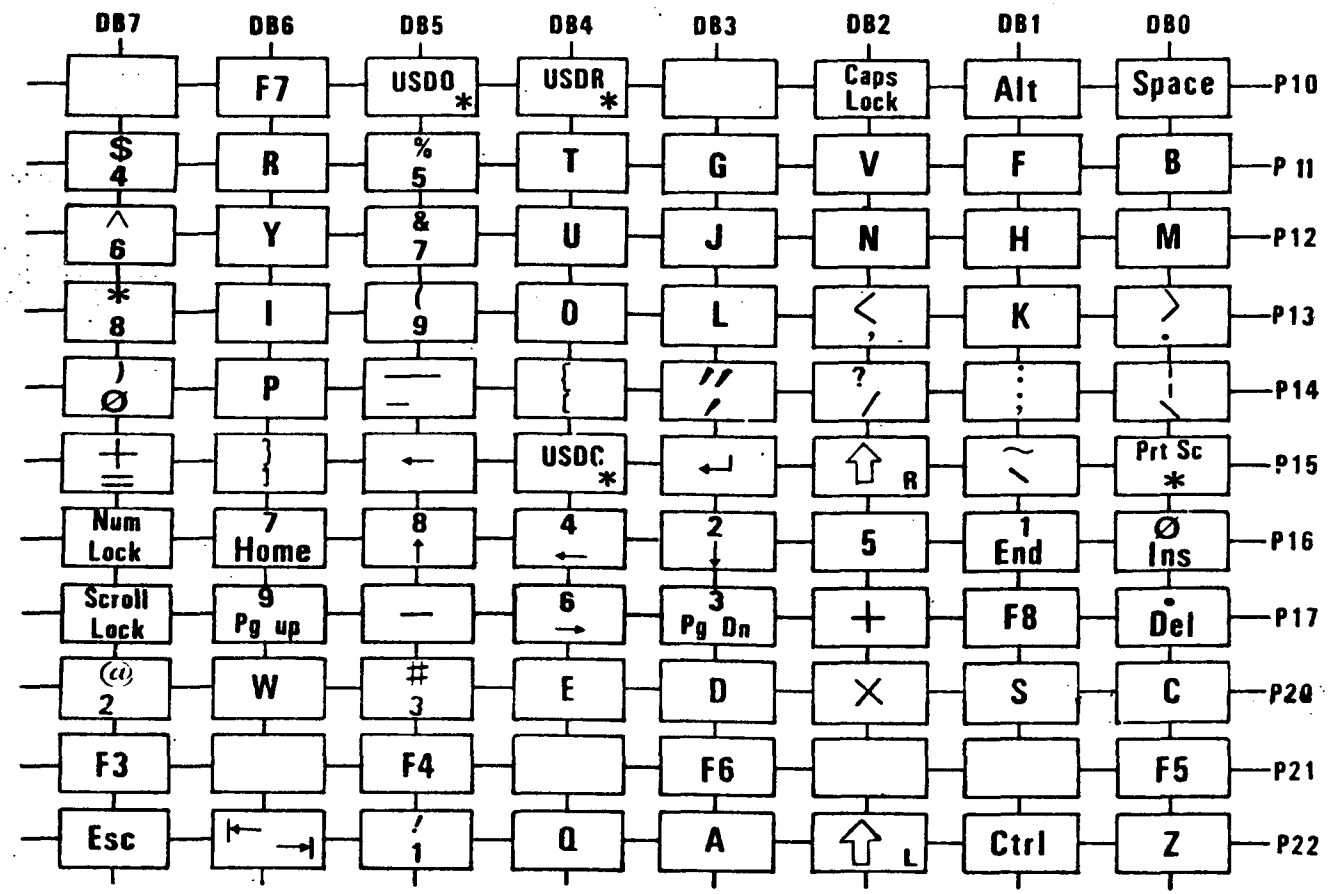
```
                ISIS-II OBJECT LINKER V3.0 INVOKED BY:
-LINK MACBA.OBJ,MACBA0.OBJ,PLM80.LIB TO MACBA.LNK PRINT(:F0:MACBA.LSK)&
**MAP

LINK MAP OF MODULE MACBA
WRITTEN TO FILE :F0:MACBA.LNK
MODULE IS A MAIN MODULE

SEGMENT INFORMATION:
START   STOP LENGTH REL NAME

                A84H  B  CODE
                 36H  B  DATA
                 10H  B  STACK
0000H  0003H     4H  A  ABSOLUTE
0008H  0008H     1H  A  ABSOLUTE
0010H  0010H     1H  A  ABSOLUTE
0018H  0018H     1H  A  ABSOLUTE
0020H  0020H     1H  A  ABSOLUTE
0024H  0024H     1H  A  ABSOLUTE
0028H  0028H     1H  A  ABSOLUTE
002CH  0030H     5H  A  ABSOLUTE
0034H  0038H     5H  A  ABSOLUTE
003CH  003CH     1H  A  ABSOLUTE
0040H  005FH    20H  A  ABSOLUTE

INPUT MODULES INCLUDED:
 :F0:MACBA.OBJ(TERMINAL)
 :F0:MACBA0.OBJ(MODULE)
 :F0:PLM80.LIB(@P0014)
 :F0:PLM80.LIB(@P0034)
 :F0:PLM80.LIB(@P0094)
 :F0:PLM80.LIB(@F0098)
 :F0:PLM80.LIB(@P0101)
 :F0:PLM80.LIB(@P0103)
 :F0:PLM80.LIB(@PRMSK)
 :F0:PLM80.LIB(@PSMSK)
                ISIS-II OBJECT LOCATER V3.0 INVOKED BY:
-LOCATE MACBA.LNK TO MACBA.V21 MAP PRINT(:F0:MACBA.LSC)&
**CODE(0060H) DATA(1F64H) STACK(1FE0H) MEMORY(2000H)

MEMORY MAP OF MODULE MACBA
READ FROM FILE :F0:MACBA.LNK
WRITTEN TO FILE :F0:MACBA.V21
MODULE START ADDRESS 00DEH

START   STOP LENGTH REL NAME

0000H  0003H     4H  A  ABSOLUTE
0008H  0008H     1H  A  ABSOLUTE
0010H  0010H     1H  A  ABSOLUTE
0018H  0018H     1H  A  ABSOLUTE
0020H  0020H     1H  A  ABSOLUTE
0024H  0024H     1H  A  ABSOLUTE
0028H  0028H     1H  A  ABSOLUTE
002CH  0030H     5H  A  ABSOLUTE
0034H  0038H     5H  A  ABSOLUTE
003CH  003CH     1H  A  ABSOLUTE
0040H  005FH    20H  A  ABSOLUTE
0060H  0AE3H   A84H  B  CODE
1F64H  1F99H    36H  B  DATA
1FE0H  1FFBH    1CH  B  STACK
2000H  F6BFH  D6C0H  B  MEMORY
```

APENDICE IV



ESQUEMA DEL TECLADO



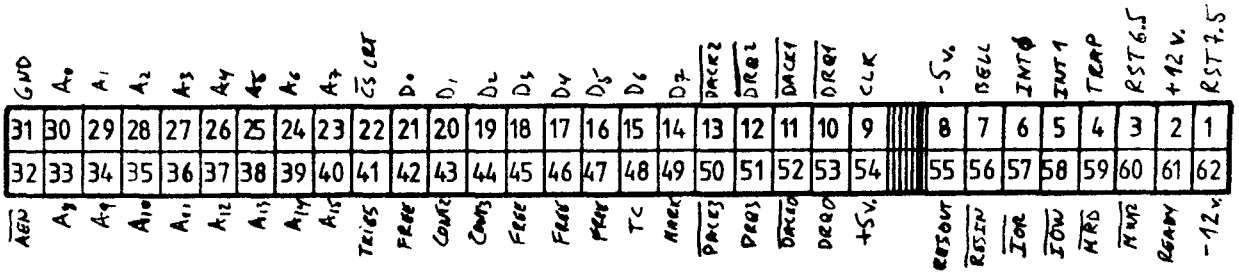


MATRIZ DE TECLAS

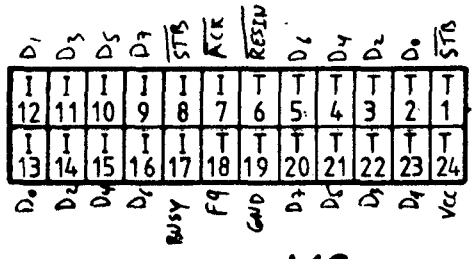
	Fecha	Nombre		E. U. P.
Dibujado	9-7-86	M. C. B.		
Comprob.				
Escala	TARJETA DE CONTROL DEL SISTEMA			Esquema 1

	Fecha	Nombre		
Dibujado	11-7-86	M.C.B.		E.U.P.
Comprob.				
Escala			TARJETA DE CONTROL CRT	Esquema 2

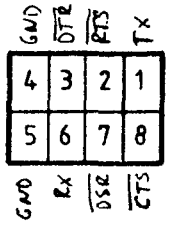
K1 bus del sistema



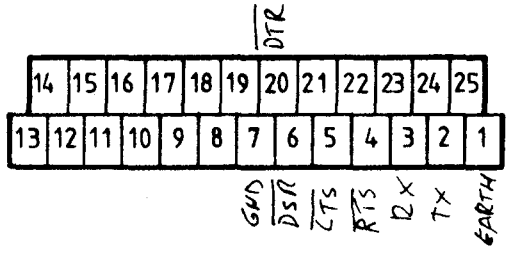
K2 teclado e impresora



K3 modem

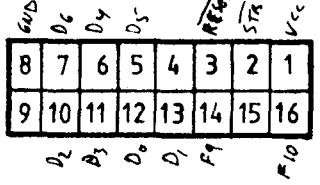


K6 modem (RS-232C)



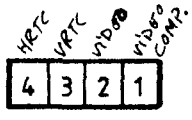
K4 term. libres

K5 placa adaptadora teclado



K7 especiales

K8 video



APENDICE V

- LISTA DE COMPONENTES:

- Resistencias: (ohmios, 1/4 de watio)

R1 a R13	--	2K7	R22	--	10K
R14	--	150	R23	--	2K2
R15	--	10K	R24	--	1K
R16-R17	--	330	R25	--	10K
R18	--	14K	R26	--	2K2
R19	--	4K7	R27	--	50
R20	--	4K7	R28	--	50
R21	--	2K2	P1,P2	--	50K lineal

- Condensadores:

C1 a C7	--	100 nF	C12	--	1 nF
C8	--	10 pF	C13	--	470pF
C9	--	1 nF	C14	--	100nF
C10	--	10K nF	C15	--	50 nF
C11	--	10 nF	C16	--	1K nF

- Circuitos integrados:

IC1	--	8085	IC17	--	1488
IC2	--	8256	IC18	--	7408
IC3	--	8257	IC19	--	7404
IC4	--	8255	IC20	--	74LS156
IC5	--	8275	IC21	--	2716
IC6	--	2716	IC22	--	74LS166
IC7	--	2716	IC23	--	74LS221
IC8	--	6116	IC24	--	74LS221
IC9	--	6116	IC25	--	74LS221
IC10	--	74LS244	IC26	--	74LS151
IC11	--	8212	IC27	--	7474
IC12	--	74LS245	IC28	--	74175
IC13	--	74LS156	IC29	--	74175
IC14	--	74LS156	IC30	--	74163
IC15	--	8212	IC31	--	7486
IC16	--	1489	IC32	--	7410

- Otros:

TR1 -- 2N2222 ; D1 -- 1N4148 ; M1aM4 - microinterruptores

XT1 -- cuarzo de 6.144 MHz.

XT2 -- cuarzo de 11.34 MHz.

APENDICE VI

BIBLIOGRAFIA

* COMPUTER PERIPHERALS FOR MINICOMPUTERS, MICROPROC. & PERSONAL COMPUTERS

Louis Hohenstein
McGraw-Hill - 1980

* MICROPROCESSOR AND PERIPHERAL HANDBOOK

INTEL Corporation - 1983

* INTERCONEXION DE PERIFERICOS A MICROPROCESADORES

Serie Mundo Electrònico
Marcombo - Boixareu Editores - 1983

* MICROPROCESSOR SYSTEM DESIGN CONCEPTS

Nikitas A. Alexandridis
Computer Science Press Inc.

* NOTAS DE APLICACION: AP-62, AP-32, AR-178

INTEL Corporation

* EIA STANDAR RS-232C

Electronic Industries Asociation - 1981

* ANSI/IEEE STANDAR DIGITAL INTERFACE FOR PROGRAMABLE INSTRUMENTATION (IEEE-488)

Institute of Electrical & Electronic Engineers - 1978

* MEMORY COMPONENTS HANDBOOK

INTEL Corporation - 1983

* FUNDAMENTOS DE TELEVISION

Otto Limann
Marcombo - Boixareu Editores - 1983

* MCS 80/85 FAMILY USER'S MANUAL

INTEL Corporation - 1979

* TTL DATA BOOK

Texas Instruments

* TELEINFORMATICA Y REDES DE COMPUTADORAS

Serie Mundo Electronico
Marcombo - Boixareu Editores

* LINEAR INTERFACE DATA BOOK

Fairchild - 1978