UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA AUTOMATICO Y SIMULTANEO PARA PRUEBA DE TERMINALES

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

MANUEL JOSUE ESCOBAR CRISTIANI

Director: Ing. Arturo E. Martínez Hernández

México, D. F. 1987

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

SISTEMA PARA PROBAR EL FUNCIONAMIENTO

DE TERMINALES TELEVIDEO MODELO:

921

922

924

9[illegible]

CONTENIDO

PREFACIO

En los capítulos siguientes se da una explicación del trabajo desarrollado, se indican los problemas que existen en la fabricación de terminales que originaron la necesidad de contar con un sistema que permita tener una mayor confiabilidad en la línea de producción; una descripción del grupo MEXEL (por ser esta la compañía donde se desarrollo este proyecto); de la computadora TS-802, por ser el centro del sistema probador, y de su dispositivo básico de comunicación que es un serializador de entrada/salida (ZRO-SIO); de la solución elegida; además se incluye el manual de operación del sistema, el cual está enfocado para servir de ayuda a la persona encargada de realizar las pruebas a las terminales. En los apéndices se incluye información para poder realizar las pruebas a otros modelos de terminales y a todo lo referente a la transportabilidad del software; se da un listado de los programas realizados, tanto en lenguaje Pascal como en ensamblador y finalmente se incluyen especificaciones de todos los circuitos integrados utilizados en el diseño de la tarjeta multiplexora, y se indica en que manual se puede obtener mayor información de los mismos.

CAPITULO 1

INTRODUCCION

1.1 ANTECEDENTES

El constante crecimiento de la industria de la computación en México ha ocasionado un incremento notable en el número y variedad de equipos en esta área que se fabrican en nuestro país, como pueden ser: terminales de video, impresoras, fuentes de alimentación, equipos de comunicación de datos (multiplexores, modems, etc.) y hasta las propias computadoras.

Existen, en la actualidad, problemas muy graves en el control de calidad de estos equipos, ya que los métodos convencionales de prueba se han vuelto inoperantes, por resultar ser lentos en comparación con el número de equipos que se pueden fabricar, y por ser manuales pueden ocurrir una mayor cantidad de fallas que no se logren detectar.

Por esto resulta necesario contar con sistemas automáticos de prueba para poder eliminar los equipos que no funcionen correctamente.

El presente trabajo es un sistema que se encarga de realizar pruebas de manera automática y en forma simultanea a terminales CRT de la marca Televideo, que son distribuidas en México por el Grupo MEXEL. El sistema fue realizado en la microcomputadora TS-802, que es la encargada (mediante una serie de rutinas en Pascal y en Ensamblador Z-80) de realizar la prueba a la terminal correspondiente (con la ayuda de una tarjeta multiplexora, diseñada como parte del sistema). A continuación se da una breve descripción de las funciones que desarrolla este grupo de empresas nacionales.

El Grupo Mexel es una compañía establecida en México que se encarga de la fabricación y distribución de equipo electrónico, fabricado tanto en el país como en el extranjero. Algunos de los equipos que produce son: de alta precisión para instrumentación electrónica, de medición, fuentes ininterrumpidas de poder, computadoras, y terminales de video para computadora.

El grupo MEXEL está formado por las siguientes empresas:
MEXEL, S.A. de C.V.
Mexicana de Electrónica Industrial,S.A.
Instrumentos Electrónicos Profesionales,S.A.
Industrias Televideo, S.A. de C.V.
Sistemas de Teleinformática, S.A. de C.V.
EMTEC, S.A. de C.V.
CPM Electronics, Inc.

1.2 PROBLEMA A SOLUCIONAR

En la planta de Industrias TELEVIDEO, S.A. de C.V. se hizo latente la necesidad de contar con un equipo de prueba automática de terminales, debido a que el constante crecimiento de esta empresa hacía más y más ineficiente el sistema tradicional de prueba, que consiste en que un grupo de operadores se encarga de realizar, durante un tiempo determinado, las pruebas pertinentes de forma totalmente manual, una de las principales deficiencias, de este sistema para probar terminales, consiste en que el tiempo de realización de las pruebas es muy alto, debido a que la velocidad con que se ejecutan queda condicionada a la habilidad del operador que las realice, esto implica, a su vez, la necesidad de contar con operadores altamente calificados, ademas podemos citar el hecho de que es más factible la ocurrencia de errores si el sistema de prueba es realizado manualmente y no por un sistema automático. Debido a esto el departamento de Ingeniería de MEXEL pensó en desarrollar este equipo automático de prueba de terminales, enfocado para cuatro modelos diferentes: TELEVIDEO921, TELEVIDEO922, TELEVIDEO924 Y TELEVIDEO955, estos son los modelos que en la actualidad tienen mayor demanda, como se cita en el apéndice A resulta sencillo incluir un nuevo modelo.

En resumen el problema existente era automatizar la prueba de terminales en la planta de Industrias TELEVIDEO. Al analizar el problema se llegó a la conclusión de que para obtener una solución sencilla, económica y práctica se utilizaría parte del equipo con que ya contaba MEXEL, lo que implicó la utilización de la microcomputadora TS802 con

todas las limitaciones indicadas en el capitulo tres, para implementar con ella el sistema de prueba de terminales. Con esto se logró reducir el costo total del proyecto, ya que se le daría uso a una microcomputadora que no estaba siendo utilizada, el costo se redujo exclusivamente al desarrollo del software necesario para la prueba y a la elaboración de una tarjeta multiplexora, diseñada e implementada como parte del proyecto, que sirve como interfaz entre la computadora y las terminales a probar.

Cabe mencionar que el sistema requerido fué enfocado a control de calidad y no a mantenimiento, es decir, se trataba de realizar un sistema que detectara fallas en las terminales de una manera práctica y eficaz, pero sin indicar la ubicación exacta de la falla, es decir las terminales que no pasaran esta prueba serían entregadas posteriormente a mantenimiento, en donde ya se cuenta con equipos que pueden ubicar con toda exactitud la falla, si éste existe, pero que únicamente pueden realizar la prueba a una sola terminal, ya que el microprocesador de la terminal es substituido por un emulador del sistema probador, a diferencia de este sistema que puede probar 256 terminales en forma "simultánea". De esta forma los dos sistemas de prueba se complementan de manera eficiente.

El sistema probador realiza una serie de pruebas a cada terminal, los resultados de cada una de ellas son desplegados en la pantalla de la microcomputadora TS-802 al terminar de realizarla a todas las terminales conectadas. El operador puede almacenar los resultados de las pruebas en un archivo en disco cada vez que el lo juzgue conveniente.

CAPITULO 2

MICROCOMPUTADORA TS-802

2.1 CONFIGURACION DEL HARDWARE.

2.1.1 INFORMACION GENERAL.

La microcomputadora Televideo TS-802 fué utilizada en el desarrollo de este trabajo como el dispositivo controlador, es la encargada de realizar las pruebas a las terminales mediante el uso de su puerto serie de comunicación, de su puerto paralelo interno (normalmente usado para conectar otra unidad de discos) y de una tarjeta direccionadora (diseñada especificamente para este trabajo), mediante una serie de programas en PASCAL y en ENSAMBLADOR. Es una microcomputadora compacta de una sola tableta la cual puede ser usada como una computadora, como estación de usuario, o como terminal conectada a un sistema multiusuario (ya sea con una Televideo TS-806 o con una Televideo TS-816).

Muchos dispositivos periféricos (impresoras, teletipos, modems,etc.) pueden ser usados con la TS-802, asegurando con esto la flexibilidad de esta microcomputadora.

Esta microcomputadora está formada, principalmente, por los siguientes componentes:

- Dos unidades de disco blando.
- Una tableta lógica TS-800.
- Una tableta "hermana" que contiene la lógica del controlador de disco Winchester.

- Una terminal con las capacidades del modelo 950 de televideo.
- Un microprocesador Z80A.
- Memoria RAM de 64K.
- 4K de EPROM para diagnósticos.

Las unidades de disco flexible usan discos de 5 1/4 de pulgada y doble densidad. Todos los puertos de entrada/salida operan a través del procesador interno (Z80A). La microcomputadora trabaja con el sistema operativo CPM.

2.1.2 PUERTOS DISPONIBLES.

En la parte trasera de la microcomputadora se encuentran tres puertos que permiten al usuario conectar dispositivos periféricos como se describe a continuación.

PUERTO	DESCRIPCION
P2 (izquierda)	Este puerto RS232C es para un dispositivo serie provisto por el usuario tal como una impresora o un modem. Este puerto viene configurado de fábrica para una impresora, pero puede modificarse para que en su lugar pueda ser conectado un modem.
P1 (derecha)	Este puerto RS232C puede ser configurado en cualquiera de 3 formas: (1) De fábrica viene configurado para ser usado con un modem. (2) Modificando conexiones internas de la microcomputadora se puede conectar la microcomputadora con una impresora serie. (3) Cambiando un dipswitch se puede usar la TS-802 como una terminal conectada con el sistema TS-806/816.

RS422

Es un puerto serie de alta velocidad que conecta la TS-802 a la TS-804 o TS-816 para usarla como estación de usuario.

Asimismo en la tableta de la microcomputadora se encuentran los puertos que corresponden a las unidades de disco flexible. Y en la tableta "hermana" (que es la encargada de manejar una unidad de disco adicional) se encuentra el puerto del Winchester drive.

2.2 CONFIGURACION DEL SOFTWARE.

2.2.1 SISTEMA OPERATIVO.

La microcomputadora TS-802 usa el sistema operativo CPM (Control Program for Microcomputers), la versión con la que trabajamos es la 2.2 desarrollado y distribuido por Digital Research.

2.2.2 LENGUAJES DE PROGRAMACION.

La microcomputadora TS-802 puede usar cualquier de los siguientes lenguajes de programación: BASIC, ALGOL, APL, "C", CBASIC, COBOL, FORTH, FORTRAN, MBASIC, PL/I, RM/COBOL, PASCAL y por supuesto ENSAMBLADOR Z80 y ENSAMBLADOR 8080.

2.3 LIMITACIONES

Por ser una microcomputadora "antigua" (apareció en 1981) resulta ser demasiado lenta en comparación con las computadoras personales actuales, utiliza como procesador el Z80 (que trabaja como máximo a 4 MB y es de 8 bits), el cual ya no se utiliza en las microcomputadoras, como tampoco el Z80-SIO que es su dispositivo de comunicación serie de entrada-salida, lo que limita la transportabilidad del sistema, ya que en caso de querer instalarlo en otra computadora se tendrían que cambiar las rutinas de manejo del controlador por las indicadas para el manejo del dispositivo que utilice la nueva computadora.

Es una microcomputadora, de uso muy poco frecuente en México, por lo que, en caso de que fallara se detendría la realización de las pruebas hasta que la falla sea corregida, lo que representaría retrasos en la producción.

CAPITULO 3

CONTROLADOR DE ENTRADA-SALIDA: Z80-SIO

3.1 INFORMACION GENERAL.

El Z80-SIO (SERIAL INPUT/OUTPUT) es un dispositivo controlador de entrada-salida en forma serie, diseñado para satisfacer una amplia variedad de requerimientos de comunicación de datos en sistemas de microcomputadoras, en el caso específico de la computadora TS-802 (que fue empleada en el desarrollo del trabajo) es el dispositivo que se encarga del control interno del puerto serie para comunicación externa. Su función básica es el control y la conversión de datos serie a paralelo y paralelo a serie.

El Z80-SIO es capaz de manejar protocolos orientados a Bytes en comunicación síncrona y asíncrona.

Las características principales de este dispositivo, que fueron utilizadas en el desarrollo del presente trabajo, son las siguientes:

-Dos canales full-duplex independientes.
-Rango de transmisión de datos:
 a) 0 a 550 [kbits/segundo] con un reloj de 2,5 [MHz].
 b) 0 a 880 [kbits/segundo] con un reloj de 4,0 [MHz].
-Sus características asíncronas son:
 a) 5, 6, 7 u 8 [bits/caracter].
 b) 1, 1,5 o 2 bits de paro.
 c) Paridad par, impar o no paridad
 d) Modos de reloj X1, X16, X32, X64.
 e) Generación y detección de Break.
 f) Detección de errores de estructura, paridad y overrun.

Nota:
1. Datos y características tomados de: "Thecnical Manual for Serial I/O Controller MK3884/MK3885"

-Ambos canales tienen entradas y salidas separadas para el control de Modems.

Existen otras muchas caracteristicas en este dispositivo, como todas las utilizadas para su operación sincrona, pero en el desarrollo del presente trabajo solo se programó para que trabajara en forma asincrona, por lo que sus otras caracteristicas no se detallan.

3.2 ARQUITECTURA

En la figura 1 se muestra la distribución de pines de este dispositivo, y la descripción del funcionamiento de los mismos se muestra en la tabla 1.

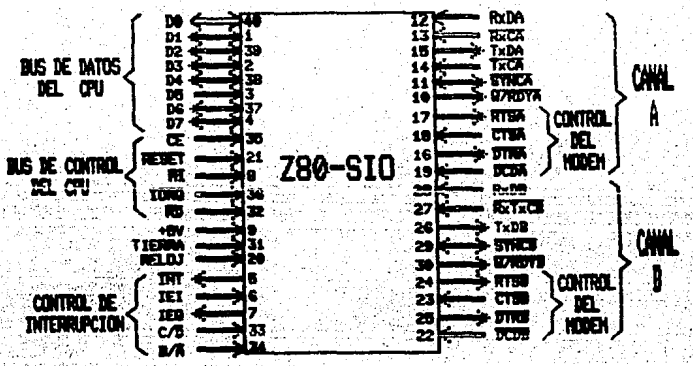


Figura 1

Tabla 1

D0-D7	Bus de datos del sistema (bidireccional tres estados).
B/$\overline{A}$	Selección del canal A o del B (entrada alta selecciona canal B).
C/$\overline{D}$	Selección para control o datos (entrada alta selecciona control).
$\overline{CE}$	Habilitación del chip (entrada activa bajo).
$\overline{M1}$	Ciclo de máquina, señal del Z80-CPU (entrada activa bajo).
$\overline{IORQ}$	Requerimiento de entrada/salida del Z80-CPU (entrada activa bajo).
$\overline{RD}$	Estado del ciclo de lectura del Z80-CPU (entrada activa bajo).
ϕ	Reloj del sistema (entrada).
$\overline{RESET}$	Reset (entrada activa bajo) deshabilita recepción y transmisión. TxDA y TxDB son forzadas a un nivel. El control del modem es forzado a a un nivel alto. Los registros de control deben ser reescritos después de que el SIO se ha "reseteado" y antes de que cualquier dato halla sido enviado o recibido. Todas las interrupciones son deshabilitadas.
IEI	Entrada para habilitar interrupción (entrada activa alto).
IEO	Salida para habilitar interrupción (salida activa alta). IEI y IEO de una conección "daisy-chain" son para control de prioridad de interrupción.
$\overline{INT}$	Requerimiento de interrupción (salida activa baja).
WAIT/READY A WAIT/READY B	Dos pines uno para cada canal. Pueden ser programadas para servir como líneas para usarse con un controlador DMA o como líneas de espera para sicronizar al Z80-CPU a la velocidad de datos del SIO.
CTSA, CTSB	Clear To Send (dos pines de entrada activa bajo). Cuando programados como Auto enables, Estas entradas habilitan la transmisión de sus respectivos canales. Si estos pines no son programados para habilitar transmisión, pueden ser usados como entradas de propósito general.
$\overline{DCDA}$, $\overline{DCDB}$	Data Carrier Detect (Dos entradas activas bajo). Su funcionamiento es similar al de las entradas CTS, excepto que estos pines son usados para habilitar recepción.
RxDA, RxDB	Recepción de datos (dos entradas activas alto).
TxDA, TxDB	Transmisión de datos (Dos salidas activas alto).
$\overline{RxCA}$, $\overline{RxCB}$	Reloj Receptor (entrada activa bajo).
$\overline{TxCA}$, $\overline{TxCB}$	Reloj Transmisor (entrada activa bajo).
$\overline{RTSA}$, $\overline{RTSB}$	Request To Send (salida activa bajo). Cuando el bit RTS es desactivado en modo asíncrono, este pin toma el valor de alto solo después de que ha terminado la transmisión.
$\overline{DTRA}$, $\overline{DTRB}$	Data Terminal Ready (salida activa bajo). Siguen el estado programado con el bit DTR.
$\overline{SYNCA}$, $\overline{SYNCB}$	Caracteres externos de sincronía, no utilizados en este trabajo.

La figura 2 muestra un diagrama de bloques de la arquitectura del Z80-SIO. Su estructura interna incluye un bus de interfaz con el Z80-CPU, lógica de interrupción y de control internos, además de dos canales full-duplex. La lógica de control de interrupción determina que canal y que dispositivo dentro del canal es el que tiene mayor prioridad para propósitos de interrupción automática utilizando vectores. La prioridad es fija, con el canal A asignado con más alta prioridad que el canal B y las interrupciones por recepción, transmisión y external/status teniendo la prioridad asignada en ese orden dentro de cada canal.

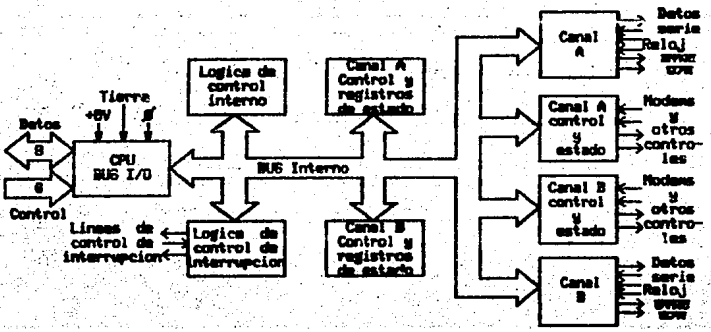


Figura 2

En la figura 3 se muestra la lógica de un canal en diagrama de bloques. Cada canal tiene cinco registros de control y tres registros de estado de ocho bits cada uno. El vector de interrupción es escrito en un registro de ocho bits en el canal B y puede también ser leido desde este canal. El receptor tiene tres registros buffer de ocho bits con un arreglo FIFO (primeras entradas primeras salidas), además de un registro de corrimiento de entrada de ocho bits. El transmisor tiene un solo registro buffer de ocho bits, además de un registro de corrimiento también de ocho bits. Los generadores/checadores CRC son registros de corrimiento de 16 bits con realimentación interna (programable) para dos diferentes códigos de CRC.

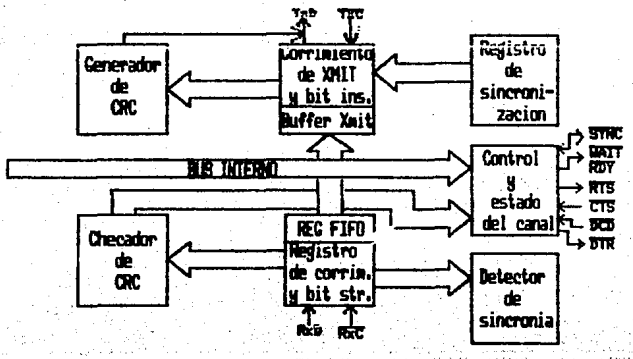


Figura 3

3.3 OPERACION ASINCRONA

El funcionamiento del SIO está determinado por el contenido de los registros de control de operación los que deben de ser programados antes de cualquier operación; algunos comandos y modos de operación pueden ser cambiados durante la operación, los registros de estado del dispositivo pueden ser leídos en cualquier tiempo.

En el modo asíncrono los puertos receptores son leídos y almacenados cuatro veces, es decir, hay tres registros de almacenamiento además del registro de corrimiento de entrada. Esto deja tiempo adicional para que el CPU atienda una interrupción al principio de una transferencia de datos en modo bloque de alta velocidad. Las banderas de error son también leídas y almacenadas cuatro veces y son cargadas al mismo tiempo que el caracter. Las banderas RECEIVER, OVERRUN Y PARITY ERROR no son "reseteadas" a menos que se utilize un comando ERROR RESET. Los errores END OF FRAME y CRC/FRAMING reflejan el estado del caracter actual en el buffer y no son reseteados por ERROR RESET. Así, cuando el estado de error es leído, reflejará un error en la palabra que está recibiendo el buffer en ese momento, además de cualquier error de paridad u overrun recibido desde el último comando de ERROR RESET. Para que haya correspondencia entre el estado de error y el contenido de los registros de recepción el registro de estado debe ser leído antes del dato. Esto es facilmente realizado si las interrupciones con vector son usadas dado que un vector especial de interrupción es generado para errores.

Si el estado es leído después del dato, el error de dato para la siguiente palabra de datos será también incluida si ha sido puesta en el buffer. Si las operaciones están siendo realizadas rapidamente de tal forma que el siguiente caracter aún no ha sido recibido, entonces el registro de estado continuará siendo válido. La excepción ocurre cuando el modo RECEIVE INTERRUPT ON FIRST CHARACTER ONLY es seleccionado. En este modo una interrupción especial mantendrá el dato de error y el caracter mismo (aún si es leído del buffer), hasta que se utilice un comando ERROR RESET.

Si se selecciona el modo INTERRUPT IN EVERY CHARACTER, el vector de interrupción será diferente si existen estados de error en el registro de estado. Si ocurriera un overrun del receptor a pesar del "buffereo" cuadruple, el caracter recibido más recientemente será cargado. El caracter precedente será perdido. Cuando el caracter que ha sido

escrito sobre otro caracter es leido, el bit de OVERFLOW será prendido y el vector de SPECIAL RECEIVE CONDITION respesará si STATUS AFFECTS VECTOR está habilitado.

Es posible usar el SIO en modo "Polled". Esto requiere el monitoreo del bit RECEIVE CHARACTER AVAILABLE para saber cuando lee un caracter. Este bit es "reseteado" automáticamente cuando todos los buffers de recepción estan vacios. El bit TRANSMIT BUFFER EMPTY es alto siempre y cuando el buffer de transmisión esté vacio. En operacion "Polled" dicho bit debe ser checado antes de escribir datos en el transmisor para prevenir la sobreescritura de datos.

La figura 4 muestra el formato con que son enviados o recibidos los datos en modo asincrono.

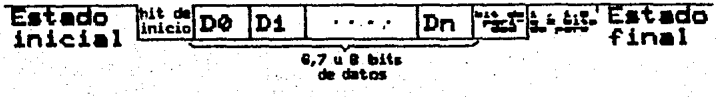


figura 4

3.3.1 TRANSMISION

Un caracter de datos enviado por el SIO será ensamblado como sigue en los modos asincronos:

Un estado inactivo (no están siendo enviados caracteres)es una marca (alto) a menos que haya sido programado un "break" en el registro de control, en cuyo caso, la linea permanecerá espaciada (bajo) hasta que el comando de SEND BREAK haya sido removido o el chip sea "reseteado".

La transmisión no puede empezar a menos que el bit TRANSMIT ENABLE esté "seteado". Si el bit de AUTO ENABLE es seleccionado, entonces CTS negada debe ser baja. Si es seleccionado el modo 5 bits/caracter entonces los bits no usados (D5, D6, D7) deben ser cero en cada byte de datos escrito en el SIO.

3.3.2 RECEPCION

La recepción asincrona se iniciará cuando el bit RECEIVER ENABLE sea "seteado". Si el bit de AUTO ENABLES es seleccionado, entonces DCD negada debe ser baja. Una condición baja en RxD indica un bit de inicio. Si el estado bajo persite por medio bit de tiempo, el bit de inicio se supone válido y el dato de entrada es entonces muestreado a la mitad de tiempo de cada bit hasta que es ensamblado todo el caracter.

Este método de detectar un bit de inicio mejora el rechazo de error cuando existen espigas de ruido en y a otro lado de la linea de marca. Si el modo reloj X1 es seleccionado, el bit de sincronización debe ser realizado externamente.

3.4 PROGRAMACION EN MODO ASINCRONO

Para programar el Z80-SIO los sitemas de software disponen de una serie de "comandos" que inicializan el modo básico de operación deseado y de otros para determinadas condiciones dentro del modo seleccionado, por ejemplo bits de paro, bits/caracter, caracter de sincronia, etc.

Cada uno de los canales del Z80-SIO contiene registros de comandos que pueden ser programados previamente a su operación.

3.4.1 REGISTROS DE ESCRITURA

En las figuras 5a, 5b, 5c se muestran todos los registros de escritura que contiene el SIO, los cuales pueden ser programados dependiendo del modo en que se utilice. La forma de programarlo se puede ver en el APENDICE B (programas en lenguaje ensamblador). Para nuestro caso se utilizará en modo Polled, con las características indicadas a continuación:

Características generales para transmisión y recepción:

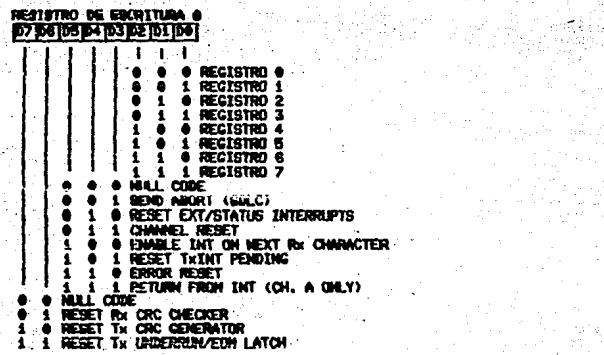
- Sin paridad.
- Un bit de paro.
- 1/32 de reloj.
- 8 bits por caracter.

Características de recepción:

- Data Carrier Detect activo.
- Clear To Send activo.
- Recepción activa.

Características de Transmisión:

- Terminal Ready activo.
- Request To send activo.
- Transmisión activa.


REGISTRO DE ESCRITURA 0
D7 D6 D5 D4 D3 D2 D1 D0
0 0 0 REGISTRO 0
0 0 1 REGISTRO 1
0 1 0 REGISTRO 2
0 1 1 REGISTRO 3
1 0 0 REGISTRO 4
1 0 1 REGISTRO 5
1 1 0 REGISTRO 6
1 1 1 REGISTRO 7
0 0 0 NULL CODE
0 0 1 SEND ABORT (SDLC)
0 1 0 RESET EXT/STATUS INTERRUPTS
0 1 1 CHANNEL RESET
1 0 0 ENABLE INT ON NEXT Rx CHARACTER
1 0 1 RESET TxINT PENDING
1 1 0 ERROR RESET
1 1 1 RETURN FROM INT (CH. A ONLY)
0 0 NULL CODE
0 1 RESET Rx CRC CHECKER
1 0 RESET Tx CRC GENERATOR
1 1 RESET Tx UNDERRUN/EOM LATCH

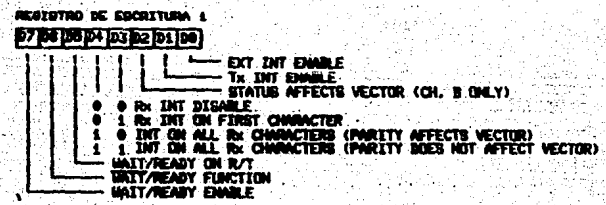
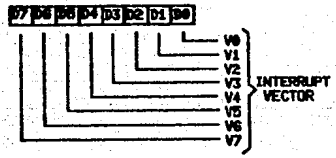
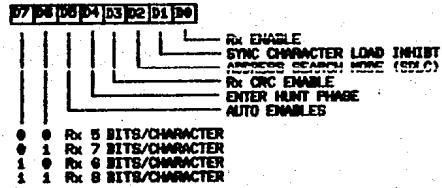

REGISTRO DE ESCRITURA 1
D7 D6 D5 D4 D3 D2 D1 D0
EXT INT ENABLE
Tx INT ENABLE
STATUS AFFECTS VECTOR (CH. B ONLY)
0 0 Rx INT DISABLE
0 1 Rx INT ON FIRST CHARACTER
1 0 INT ON ALL Rx CHARACTERS (PARITY AFFECTS VECTOR)
1 1 INT ON ALL Rx CHARACTERS (PARITY DOES NOT AFFECT VECTOR)
WAIT/READY ON R/T
WAIT/READY FUNCTION
WAIT/READY ENABLE

Figura 5a

REGISTRO DE ESCRITURA 2 (CHANNEL B ONLY)



REGISTRO DE ESCRITURA 3



REGISTRO DE ESCRITURA 4

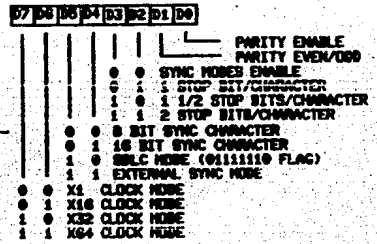
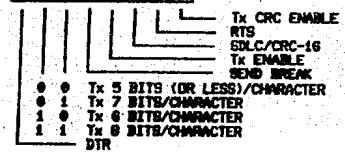
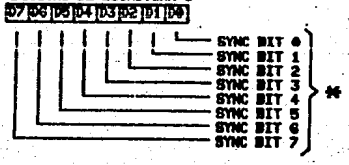


Figura 5b

REGISTRO DE ESCRITURA 5

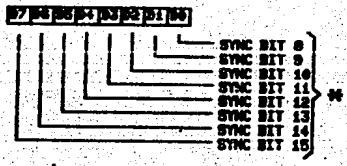


REGISTRO DE ESCRITURA 6



** Tambien SDLC campo de direccion

REGISTRO DE ESCRITURA 7



* Para SDLC debe ser programado como "01111110" para reconocimiento de bandera

Figura 5c

El primer registro a programar es el número cero. Mediante este registro se resetea el canal y se apunta a cualquier otro registro.

Una vez hecho lo anterior, se apunta al registro WR5, para nuestro propósito no se habilita interrupción, se coloca un bit de paro, y se divide el reloj de entrada por 32 (el porque se explica en el capitulo 3.5 referente al Z80-CTC).

Apuntando al registro WR5, habilitamos RTS, TxENABLE, Tx 8 BITS/CHARACTER y DTR. Con ello se habilita transmisión, demanda de envio, y demanda de recepción cuando los haya.

En el registro WR3, habilitamos RxENABLE,AUTO ENABLES, y Rx 8 BITS/CHARACTER, de esta manera se habilita recepción y las lineas de CTS y DCD se habilitarán automaticamente.

3.4.2 REGISTROS DE LECTURA

En las figuras 6a y 6b se muestran los registros de lectura del SIO

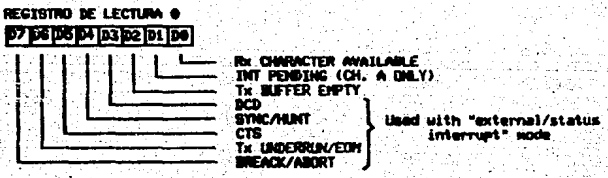


Figura 6a

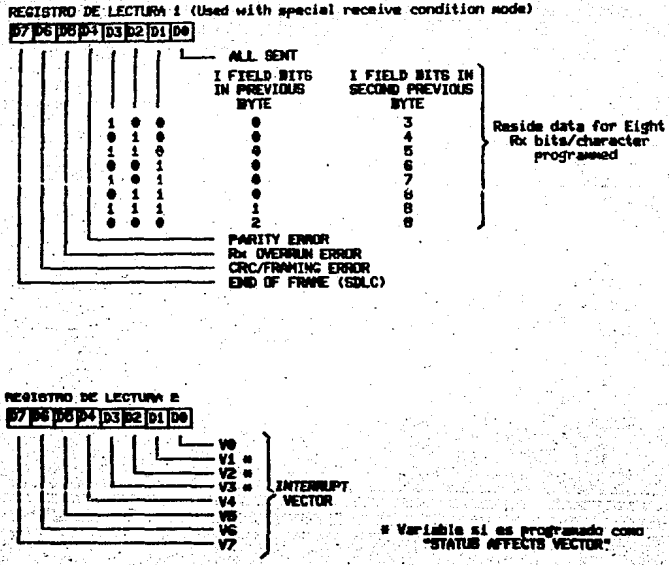

REGISTRO DE LECTURA 1 (Used with special receive condition mode)
D7 D6 D5 D4 D3 D2 D1 D0
ALL SENT
I FIELD BITS IN PREVIOUS BYTE
I FIELD BITS IN SECOND PREVIOUS BYTE
1 0 0 0 3
0 1 0 0 4
1 1 0 0 5
0 0 1 0 6
1 0 1 0 7
0 1 1 0 8
1 1 1 1 8
0 0 0 2 0
Reside data for Eight Rx bits/character programmed
PARITY ERROR
Rx OVERRUN ERROR
CRC/FRAMING ERROR
END OF FRAME (SDLC)
REGISTRO DE LECTURA 2
D7 D6 D5 D4 D3 D2 D1 D0
V0
V1 *
V2 *
V3 *
V4
V5
V6
V7
INTERRUPT VECTOR
* Variable si es programado como "STATUS AFFECTS VECTOR"

Figura 6b

Para el modo Polled solo requerimos de leer el registro RR0. De éste solo los bits 0 y 2 nos sirven. El bit 0 nos indica cuando hay caracter disponible para poder ser leido por el CPU y el bit 2 que muestra si el buffer de transmisión se encuentra vacio.

3.5 Z80-CTC COMO RELOJ DEL SIO

Dado que la configuración de la TS-802 implica que el reloj del SIO es el CTC, debemos programar a este dispositivo de manera que junto con el SIO nos den los baud rates requeridos para entablar comunicación con las terminales.

La programación es sencilla dado que solo son dos registros a programar, el primero indicará el modo de operar y el segundo será la constante de tiempo que no es más que el divisor de reloj del sistema.

De esta manera, programamos al CTC en modo contador, flanco negativo para decrementar el contador, la constante de tiempo sigue a la palabra de control y reinicialización automática cargando la constante de tiempo a la cuenta de cero.

La constante de tiempo será 1 para que el reloj del sistema, que ya fue dividido por 6.5, sea dividido entre él, dando 615 385 Hz los que entrarán al SIO y éste los dividirá por 32 proporcionando 19 200 baud/segundo, que son los requeridos.

CAPITULO 4

SOLUCION

4.1 SOLUCION.

La solución al problema planteado fue el diseño de un sistema multiplexor que permitiera conectar el número deseado de terminales a la computadora para la realización de las diferentes pruebas y que fuera transfiriendo la información a las distintas terminales de la manera más rápida posible de acuerdo a la prueba a realizar.

Para la realización de la prueba se vio la conveniencia de utilizar los siguientes puertos de la microcomputadora TS-802:

(P1) RS-232C de comunicaciones:

> Su finalidad es permitir el intercambio, entre la computadora y las terminales, de los datos y comandos necesarios para llevar a cabo la prueba.

(P3 PB) Controlador de disco winchester:

> Nos permite manejar los decodificadores que direccionan la terminal a probar. Este puerto se encuentra sobre una tableta que se denomina PIGGY BACK y que se localiza encima de la tableta lógica de la microcomputadora.

En la figura 7 se muestra un diagrama de bloques del sistema:

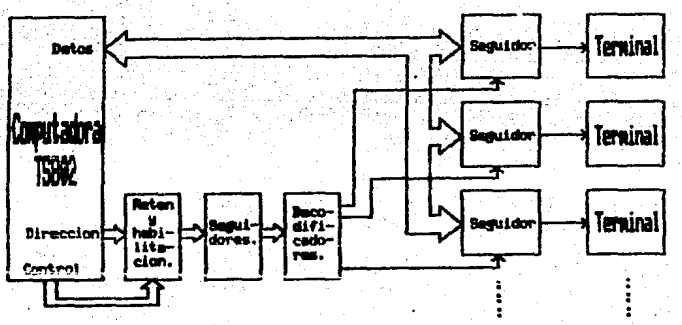


Figura 7

Su funcionamiento es el siguiente:

Inicialmente la computadora manda a traves del puerto del controlador de disco Winchester la dirección de la terminal a probar, a través del bus de datos y por el bus de control las señales de WWE (winchester write enable) y WCS (winchester chip select), haciendo uso de WCS se habilita el sistema de selección y con el WWE se habilita la recepción del dato y su escritura en un latch; dicha dirección es mantenida por medio del latch a la entrada de un bloque decodificador, el cual se encargará de seleccionar la terminal a la que se le realizará la prueba. La selección de la terminal se lleva a cabo al habilitar el seguidor correspondiente. Los cuatro bits mas significativos se encargan de seleccionar a uno de los 16 multiplexores que a la vez manejan a 16 terminales cada uno, y los cuatro menos significativos seleccionan el número de la terminal conectada al multiplexor seleccionado previamente. Ejemplo: si se manda el dato 01H significa que se selecciona la terminal uno conectada al multiplexor cero, un 1FH seleccionaría la terminal 16 del multiplexor número uno.

Una vez seleccionada la terminal se procede a realizar las diferentes pruebas que quiera el operador. La prueba de comunicación, es decir la prueba del puerto de comunicación de la terminal se realiza antes de ejecutar cualquier otra prueba para evitar que el sistema "se vaya a perder" debido a que no pueda entablar comunicación con la terminal direccionada, a las terminales que no pasen esta prueba no se les realizará ninguna otra.

Si se desea expandir el sistema basta repetir la etapa de salida a partir de los puntos suspensivos de las figuras 7 y 8.

4.1.1 HARDWARE

En la figura 8 se muestra el diagrama eléctrico del sistema multiplexor, y el significado de las líneas es el siguiente: los buses A y B corresponden al puerto uno de la computadora (puerto serie para comunicación), transmiten y reciben los datos correspondientes a las diversas pruebas; el bus C corresponde al puerto del Winchester disck controler (puerto paralelo para una unidad de disco adicional), usando las líneas 1 a 8 para indicar la dirección de la terminal a probar y las líneas 23 y 25 nos dan la presencia de una dirección válida (se usan como reloj de los flips-flops, para que la tarjeta tome una dirección nueva); los buses de la D a la J corresponden a los puertos de comunicación de las diversas terminales que estan siendo probadas.

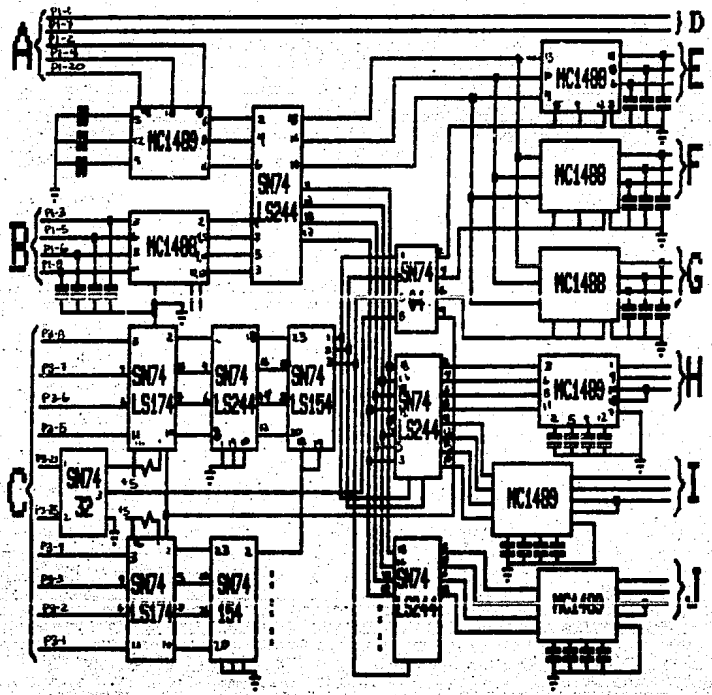


Figura 8

4.1.2 SOFTWARE

Para establecer el software se estudiaron las facilidades de la microcomputadora TS-802 y así se vió la conveniencia de realizar la prueba utilizando un programa principal en lenguaje PASCAL que llamara subrutinas en ensamblador y en PASCAL.

El lenguaje ensamblador se utilizó principalmente para manejo del SIO, tanto para configurarlo como para manejar el puerto RS232C, PASCAL se utilizó para desplegar información en pantalla, tanto de los distintos menús como de los resultados de las pruebas realizadas, para captura de datos y para grabar los resutados en disco.

Se utilizan dos rutinas externas, GRABATXT.PAS y GRT.ASM, para generar archivos con los resultados obtenidos en terminales que se encuentran funcionando normalmente, para comparar estos resultados con los obtenidos en las terminales a probar. Estos archivos contienen los datos leídos de la pantalla de una terminal y deben ser llamados de la siguiente manera: PANTALLA.XXX donde XXX es el modelo de la terminal a probar (921, 922, 924, 955), ya que con estos nombres son llamados por el programa principal.

El programa principal se subdivide en dos bloques: PROBADOR.PAS y TESTER.ASM, los que se encargan de realizar todas las pruebas a las terminales. Para generar la versión ejecutable estos dos programas ya compilados se deben ligar con las rutinas de PASCAL que se encuentran agrupadas en el archivo: PASLIB.ERL .

La figura 9 muestra el diagrama de flujo del sistema.

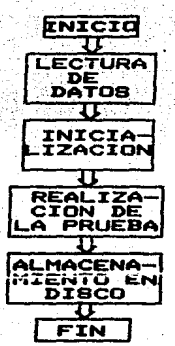

INICIO
LECTURA DE DATOS
INICIA-LIZACION
REALIZA-CION DE LA PRUEBA
ALMACENA-MIENTO EN DISCO
FIN

Figura 9

CAPITULO 5

MANUAL DE USO DEL SISTEMA

5.1 INFORMACION GENERAL DEL SISTEMA

El presente manual tiene como finalidad el servir de ayuda al operador encargado de utilizar el sistema PRUEBA DE TERMINALES.

El sistema físicamente consta de tres elementos: Microcomputadora TS-802, tarjeta de direccionamiento y discos de software, como se indica en la figura 10. Estos elementos siempre son necesarios para la correcta utilización del sistema.

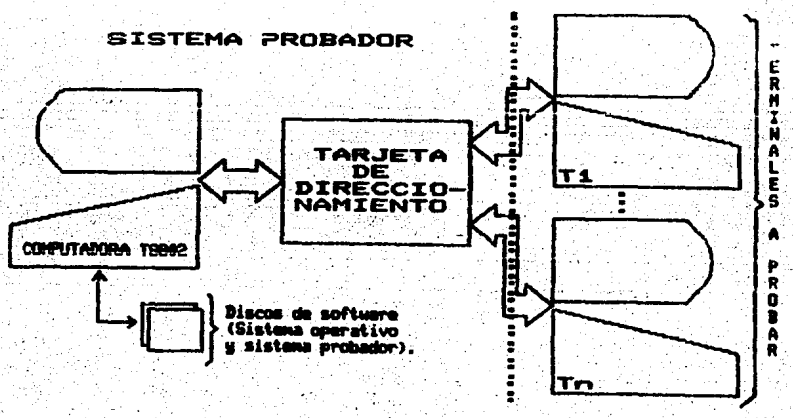


Figura 10

MICROCOMPUTADORA TS-802

Todos los programas realizados para el sistema fueron desarrollados para esta microcomputadora, por lo que únicamente se asegura su funcionamiento en esta máquina. Consta de un teclado, una pantalla de video y dos unidades de disco blando, la unidad superior es el drive "A" y la unidad inferior el "B".

TARJETA DE DIRECCIONAMIENTO

Esta tarjeta es el hardware necesario para poder probar más de una terminal. Consta básicamente de tres elementos: Conector RS232C de entrada, tarjeta lógica y conectores RS232C de salida.

El conector RS232C de entrada siempre debe ir conectado al puerto serie RS232C de la microcomputadora (ubicado en la parte posterior de la misma). Los conectores RS232C de salida deben ir conectados, uno a cada terminal bajo prueba, en su respectivo puerto serie RS232C Computer (DCE) (ubicado en la parte posterior de cada terminal).

La alimentación de la tarjeta de direccionamiento se realiza con una fuente que suministre los siguientes voltajes: +5, +12, -12 y tierra, cada voltaje se coloca en el conector marcado con el valor respectivo, como se indica en la figura 11.

Figura 11

DISCOS DE SOFTWARE

Es necesario contar con dos discos para poder correr el sistema, en el primero que debe estar en la unidad 'A' se debe tener el sistema operativo de la microcomputadora TS-802 y en el segundo, que debe estar siempre en la unidad 'B', es necesario contar por lo menos con los siguientes archivos: PROBADOR.COM, PANTALLA.921, PANTALLA.922, PANTALLA.924, PANTALLA.955.

Es recomendable, para poder realizar futuras modificaciones al sistema, contar con los siguientes archivos: PROBADOR.PAS, TESTER.ASM, PROBADOR.ERL, ENS.ERL, GRABATXT.PAS y GRT.ASM , en la unidad 'B' y L80, M80 en la unidad A y contar con un disco extra que contenga todo el compilador y el ligador de PASCAL que se puede colocar en la unidad "A".

El sistema genera como resultado de la prueba un archivo que contiene toda la información obtenida durante la prueba, el nombre de este archivo de resultados es dado por el operador, con el fin de poder almacenar el resultado de varias corridas.

5.2 INSTRUCTIVO PARA LA UTILIZACION DEL SISTEMA

El sistema consta de dos módulos principales: PROBADOR.PAS, que es el programa principal realizado en lenguaje ensamblador, el cual llama a una serie de rutinas en ensamblador, agrupadas en el archivo ENS.ASM.

El sistema esta hecho de tal forma que los datos son validados cuando son dados al sistema, por lo que en caso de existir algún error en los mismos este será indicado, y a continuación se volverán a leer los datos.

Para correr el sistema basta dar el comando B:PROBADOR una vez estando dentro de CPM. Primeramente el sistema pedirá los siguientes datos:

-El modelo de las terminales a probar:

-El numero de las terminales a probar (maximo 256):

Los modelos de terminales válidos son: 921, 922, 924, 955 .

El número de terminales a probar debe de ser un número entero mayor a 0 y menor a 255.

En caso de que cualquiera de estos valores sea distinto a los especificados, entonces existirá un error, el cual será indicado por el sistema y se tendrán que volver a dar los datos.

A continuación se desplegará sobre la pantalla la siguiente información:

INFORMACION PARA EL OPERADOR:
1.- Verificar todas las conexiones.
2.- Encienda las XXX terminales.
3.- Checque que el SET-UP de las terminales
tenga la siguiente configuración:

+ 1 + BAUD 19.2 + WORD 8 + PRTY NO + STOP 1 + COMM HDX+ PRTC X-ON +

4.- Checque que ninguna terminal quede en modo SET-UP.
5.- Cuando este listo teclee <RETURN>.

La línea de <SET-UP> de comunicación de cada una de las terminales debe de tener las características indicadas, de no ser así ésta debe de ser reconfigurada por el operador.

Al encender las terminales, se debe de teclear cualquier cosa en ellas, y verificar que aparescan los caracteres en pantalla para asegurar que estan correctamente conectadas, antes de continuar con la prueba, esta operación la debe de realizar el operador.

Al teclear <RETURN> inmediatamente se realiza la prueba de comunicación, la cual consiste en enviar una H a todas las terminales, en caso de que no aparezca este caracter en alguna de las terminales, o de que el sistema nos indique que alguna terminal está funcionando incorrectamente, esta puede ser cambiada por alguna otra, ya que a esta terminal no se le realizarán las demas pruebas.

Cuando se termine de checar a todas las terminales, el sistema preguntará si se quiere realizar nuevamente la prueba, aqui son válidas las siguientes respuestas: S si se quiere repetir y N si el operador ya esta conforme con el resultado de la prueba.

A continuación se desplegará el siguiente menú:

MENU DE PRUEBA:
1.- Video.
2.- Autoprueba.
3.- Comunicacion.

4.- Continua de quemado.
5.- Logica de video.
6.- Todas.
7.- Respaldar informacion.
8.- Consultar resultados.
9.- Salida del menu.

QUE PRUEBA QUIERES REALIZAR ??

Al dar cualquier número dentro de las opciones válidas (1, 2, 3, 4, 5, 6, 7, 8, 9) se realizará la prueba correspondiente. En caso de teclear cualquier otra cosa, el sistema lo indicará y volverá a pedir el dato.

Al terminarse de realizar alguna prueba el sistema indicará todos los errores encontrados y a que terminal corresponden, excepto en la de video, en este caso es el operador el que debe de indicar los errores encontrados de acuerdo a la siguiente tabla:

1.- No enciende el video.
2.- Vibra el video.
3.- Video fuera de cuadro.
4.- Video fuera de foco.
5.- Diferente tamaño en los caracteres desplegados.
6.- Mala intensidad.

Estos datos no son filtrados, ya que es posible que exista una falla no contemplada aún, por lo que se recomienda tener cuidado al teclear estos datos.

Todas las pruebas se pueden ejecutar más de una vez, si el operador así lo requiere.

La prueba continua de quemado es una prueba de larga duración, se recomienda que sea la última en correrse, y se puede dejar corriendo durante varias horas, tomando en cuenta que cada terminal requiere de uno a tres minutos para realizar esta prueba, dependiendo del modelo de terminal que se esté probando, multiplicamos este tiempo por el número de terminales y esto nos dará el tiempo necesario para ejecutar una sola vez esta prueba.

Cada vez que aparezca <RETURN> en la pantalla de la TS-802 se deberá de oprimir dicha tecla después de haber leido el texto desplegado en pantalla; esto se hace con el fin de que el operador pueda leer toda la información que salga en pantalla.

CAPITULO 6

CONCLUSIONES

En países en vías de desarrollo, como lo es México, no se cuenta siempre con todos los recursos económicos para poder resolver dentro de las industrias los problemas existentes con soluciones compradas. A veces aunque los recursos sobren no siempre una solución comprada resolvería sus problemas; por esto resulta importante que dentro de la misma industria se desarrollen este tipo de herramientas que resuelvan sus necesidades particulares y que además, como en este caso, si se tiene la suficiente visión de ingeniería se puedan utilizar aparatos o componentes aparentemente obsoletos o en desuso, que por alguna razón se encuentran fuera de servicio.

Trabajando conjuntamente en las áreas de Ingeniería Electrónica e Ingeniería en Computación se pueden obtener soluciones prácticas a bajo costo y en un corto período de desarrollo. Un ejemplo de estos trabajos se dió durante la elaboración de esta tesis aplicándose directamente al control de calidad de equipos electrónicos para reducir el total de horas hombre en trabajos rutinarios y monótonos, mejorando la confiabilidad de los equipos fabricados y disminuyendo el tiempo total de realización de las pruebas de funcionamiento.

APENDICE A

TRANSPORTABILIDAD Y CRECIMIENTO

CRECIMIENTO

Para realizar la prueba a un modelo diferente de terminal es necesario incluir nuevas rutinas en lenguaje ensamblador muy parecidas a las existentes para los otros cuatro modelos, cambiando unicamente las palabras de control de la terminal, ya que con estas se logra una determinada respuesta que el sistema evaluará, y son diferentes códigos para lograr una respuesta específica en distintos modelos. Por lo que solo se requiere introducir, dentro de las rutinas en ensamblador, una sección nueva para cada prueba a realizar llevando como etiqueta el nuevo modelo de terminal.

En cuanto al programa principal solo se requiere aumentar el número del modelo dentro del filtro de información que se realiza al leer el número del modelo de terminal que se va a probar, con esto se logra que el sistema crezca para un modelo nuevo de terminal.

TRANSPORTABILIDAD

Esta resulta ser un inconveniente del sistema, ya que si bien el programa principal en PASCAL puede ser transportado a otra computadora sin ningún problema, todas las rutinas en ENSAMBLADOR y la comunicación entre PASCAL y ENSAMBLADOR si sufren modificaciones.

Como el control de los dispositivos de I/O que manejan el puerto RS-232C de la computadora y su sistema de reloj para establecer la velocidad y todas las características de entrada - salida fueron diseñadas para manejar los circuitos

integrados específicos de la microcomputadora TS802 y estos varían al utilizar otras computadoras se requeriría cambiar todo lo referente a la inicialización y manejo de este para todas las señales de control y de datos, rutinas que se realizaron en ensamblador, además esta microcomputadora usa un CPU Z-80 de Zilog actualmente en desuso, por lo que el código en ensamblador se debe modificar para otro CPU.

APENDICE B

RUTINA PRINCIPAL EN PASCAL

```
PROGRAM PROBADOR(INPUT,OUTPUT);

(*************************************************************)
(**                                                         **)
(**                                                         **)
(**     SISTEMA PARA PROBAR EL FUNCIONAMIENTO DE LAS        **)
(**                                                         **)
(**                 TERMINALES TELEVIDEO                    **)
(**                                                         **)
(**                  MODELOS:   921,                        **)
(**                             922,                        **)
(**                             924 y                       **)
(**                             955                         **)
(**                                                         **)
(**      REALIZADO POR:                                     **)
(**                                                         **)
(**                  Manuel Josue Escobar Cristiani         **)
(**                                                         **)
(**      CON LA COLABORACION DE:                            **)
(**                                                         **)
(**                  Arturo Arvizu Mondragon                **)
(**                  Hector Mejia Vazquez                   **)
(**                                                         **)
(**                                                         **)
(**                                       noviembre 1985    **)
(**                                                         **)
(*************************************************************)

CONST
  MAX=256;

TYPE
  CHARFILE=FILE OF CHAR;
  I=0..256;
  VECTOR=ARRAY[I] OF INTEGER;

VAR
  ARCH
               :CHARFILE;
  NOMBRE
               :STRING;
  MOD_TERM,
  K,          (* Contador para provocar un retardo          *)
  NUM_TERM,
  REG,
  N
               :INTEGER;
  ESTO,       (* Indica en la prueba de comunicaciones:     *)
              (* ESTO[X] EL NUMERO DE LA TERMINAL QUE FALLO *)
```

```
                (*         X ES LA FALLA QUE OCURRIO:                  *)
                (*                  = 0 TERMINAL FUNCIONANDO           *)
                (*                  = 1 TERMINAL NO LISTA              *)
                (*                  = 2 TRANSMITE O RECIBE BASURA      *)
   EST1,        (* Indica en la prueba de ajuste de pantalla:          *)
                (*       X Indica el numero de terminal.               *)
                (*       EST1[X] falla ocurrida.                       *)
                (*          =1 no enciende el video.                   *)
                (*          =2 vibra el video.                         *)
                (*          =3 video fuera de cuadro.                  *)
                (*          =4 video fuera de foco.                    *)
                (*          =5 diferente tamaño de los.                *)
                (*             caracteres desplegados.                 *)
                (*          =6 mala intensidad.                        *)
   EST2,        (* Indica en la autoprueba:                            *)
                (*  EST2[X] = 0          prueba correcta.              *)
                (*          = 1          error en prueba.              *)
                (*  X : El numero de la terminal que fallo.            *)
   EST3,        (* Indica en la prueba continua de queseado:           *)
                (*     EST3[X] = 0     Prueba correcta.                *)
                (*             = 1     Error en prueba.                *)
                (*     X  indica el numero de la terminal.             *)
   EST4         (* Indica en la prueba de la logica de video:          *)
                (*     EST4[X] = 0     Prueba correcta.                *)
                (*             = 1     Error en prueba.                *)
                (*     X  indica el numero de la terminal.             *)
            :VECTOR;

(**************************************************************************)
(**                                                                      **)
(** Procedimientos externos realizados en lenguaje ensamblador           **)
(**                                                                      **)
(**************************************************************************)

EXTERNAL PROCEDURE ØISIOC;      (* Inicializacion del SIO y del CTC de *)
                                (* la computadora.                     *)
EXTERNAL PROCEDURE ØVIBR;       (* Envio de un patron de H's a las ter-*)
                                (* minales.                            *)
EXTERNAL PROCEDURE ØCOMUN;      (* Checa una correcta comunicacion en- *)
                                (* tre terminales y computadora.       *)
EXTERNAL PROCEDURE ØRECU;       (* Genera autoprueba y checa la panta- *)
                                (* talla de la terminal bajo prueba    *)
EXTERNAL PROCEDURE ØQUEMA;      (* Checa la memoria RAM de la terminal *)
                                (* que controla el despliegue de carac-*)
                                (* teres en pantalla.                  *)
EXTERNAL PROCEDURE ØLOG;        (* Checa la logica de video.           *)

PROCEDURE SACAMIN(VAR CONT      (* PUERTO DEL WINCHESTER.              *)
                        :INTEGER);
```

```
(**********************************************)
(**                                          **)
(**   DIRECCIONAMIENTO DE LA TERMINAL A PROBAR **)
(**                                          **)
(**********************************************)

VAR
        I
                :INTEGER;
BEGIN
I:=CONT;
INLINE($3A/I);
INLINE($D3/$48);
END;

PROCEDURE LEE_DATOS(VAR MOD_TERM,
                        NUM_TERM
                              :INTEGER);

(************************************************)
(**                                            **)
(** LECTURA DE LOS DATOS NECESARIOS PARA EFECTUAR **)
(**                LA PRUEBA                   **)
(**                                            **)
(************************************************)

VAR
       DAT_CORREC,            (* INDICA SI LOS DATOS FUERON CORRECTOS        *)
       CONTINUO               (* DETIENE EL PROCEDURE HASTA TECLEAR (RETURN) *)
              :CHAR;
       SALIDA                 (* =FALSE TERMINA EL PROCEDURE                 *)
              :BOOLEAN;

BEGIN
WRITE(CHR($1A),CHR($07),CHR($07),CHR($1B),CHR($3D),CHR($00),CHR($00));
WRITE(CHR($1B),CHR($29),CHR($1B),CHR($47),CHR($34));
WRITE('                                                                  ');
WRITELN('                                                                ');
WRITELN('                    SISTEMA DE PRUEBA DE TERMINALES             ');
WRITELN('                                                                ');
WRITELN('                      MODELO 921,922,924 Y 955                  ');
WRITELN('                                                                ');
WRITELN(CHR($1B),CHR($47),CHR($30),CHR($1B),CHR($28),CHR($1B),CHR($26));
WRITELN;
WRITELN;
WRITELN;
WRITELN;
WRITELN;
```

```
WRITELN;
WRITELN;
WRITELN;
WRITELN;
WRITELN;
WRITELN;
WRITE('                                                    <RETURN>');
READ(CONTINUO);
WRITE(CHR(#1A));
SALIDA:=TRUE;
WHILE SALIDA DO
      BEGIN
      WRITELN;
      WHILE SALIDA DO
            BEGIN
            WRITELN;
            WRITELN('DAME LOS SIGUIENTES DATOS: ');
            WRITE(' - EL MODELO DE LAS TERMINALES A PROBAR',
                  ' (921,922,924,955): ');
            READLN(MOD_TERM);
            WRITELN;
            WRITE(' - EL NUMERO DE TERMINALES A PROBAR (MAXIMO 255): ');
            READLN(NUM_TERM);
            WRITELN;
            WRITE('LOS DATOS SON CORRECTOS (S/N) ?? ');
            READLN(DAT_CORREC);
            WRITELN;
            IF DAT_CORREC = 'S' THEN
                  SALIDA:=FALSE
            ELSE
                  WRITELN(CHR(#1A))
            {#END IF#};
            END
      {#END DO#};
      IF (MOD_TERM <> 921) AND (MOD_TERM <> 922)
        AND (MOD_TERM <> 924) AND (MOD_TERM <> 955) THEN
            BEGIN
            WRITELN('     MODELO DE TERMINAL INVALIDO: ',MOD_TERM);
            SALIDA:=TRUE;
            END
      {#END IF#};
      IF (NUM_TERM < 1) OR (NUM_TERM > 255) THEN
            BEGIN
            WRITELN('     NUMERO INVALIDO DE TERMINALES: ',NUM_TERM);
            SALIDA:=TRUE;
            END
      {#END IF#};
      IF SALIDA = TRUE THEN
            BEGIN
```

```
                WRITELN;
                WRITELN;
                WRITELN;
                WRITE('                                                   <RETURN>');
                READ(CONTINUO);
                WRITE(CHR($1A));
                END
        (* END IF *);
        NUM_TERM:=NUM_TERM-1;
        END
 (*END DO*);
 END;

 (** FIN DE PROCEDURE LEE_DATOS **)


PROCEDURE INICIALIZACION(VAR MOD_TERM,          (* MODELO DE LAS TERMINALES A PROBAR       *)
                             NUM_TERM           (* NUMERO DE LAS TERMINALES A PROBAR       *)
                                          :INTEGER);

(****************************************)
(**                                   **)
(**    INICIALIZACION DE LA PRUEBA    **)
(**                                   **)
(****************************************)

VAR
     SIGUE              (* Detiene el procedure hasta teclear <RETURN> *)
               :CHAR;

BEGIN
WRITELN(CHR($1A));
WRITELN;
WRITELN;
WRITELN('     INFORMACION PARA EL OPERADOR:');
WRITELN('     1.- Verificar todas las conexiones.');
WRITELN('     2.- Encienda las ',NUM_TERM+1,' terminales');
WRITELN('     3.- Cheque que el SET-UP de las terminales');
WRITELN('         tenga la siguiente configuracion:');
WRITELN;
          WRITELN(CHR($1B),CHR($47),CHR($34),'1 + BAUD 19.2 + WORD  8  ',
                '+ PRTY  NO+ STOP  1 + COMM  FDX + XXXXXXXX ',CHR($1B),
                 CHR($47),CHR($30));
WRITELN;
WRITELN('     4.- Cheque que ninguna terminal quede en modo SET-UP');
WRITE('     5.- Cuando este listo teclee <RETURN>.');
READ(SIGUE);
WRITE(CHR($1A));
```

```
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITELN(CHR($13),CHR($47),CHR($30));
 WRITELN(CHR($1B),CHR($47),CHR($30));
 WRITE(CHR($1A));
 PISIOC;
 END;

 (** FIN DE PROCEDURE INICIALIZACION **)

 PROCEDURE PRUEBAO(VAR MOD_TERM,          (* Modelo de las terminales a probar *)
                       NUM_TERM           (* Numero de las terminales a probar *)
                                 :INTEGER);

 (*************************************)
 (**                                 **)
 (**   PRUEBA DEL PUERTO DE COMUNICACION DE   **)
 (**                                 **)
 (**        LAS TERMINALES A PROBAR          **)
 (**                                 **)
 (*************************************)
 VAR
    CONT,          (* DIRECCIONA UNA TERMINAL 1<=CONT<=256          *)
    AUX,           (* CONTADOR QUE INDICA HASTA 10 LECTURAS A CONSIO *)
    FALLAS,        (* CUENTA EL NUMERO DE TERMINALES QUE FALLO      *)
    CARACTER,      (* CARACTER RECIBIDO POR EL SIO                  *)
    ESTADO         (* INDICA DE LA TRANSMISION O LA RECEPCION SE    *)
                   (* PUEDEN REALIZAR                               *)
          :INTEGER;
    CONTINUO,      (* DETIENE EL PROCEDURE HASTA TECLEAR (RETURN)   *)
    RESP,          (* ='S' SE REPITE LA PRUEBA                      *)
    CAR            (* INDICA LA FALLA OCURRIDA                      *)
                   (*       =A : APAGADA O MAL CONECTADA            *)
                   (*       =B : RECIBE O TRANSMITE BASURA          *)
                   (*       =C : NO OCURRIO ERROR                   *)
          :CHAR;
 BEGIN
 RESP:='S';
 WHILE RESP = 'S' DO
       BEGIN
       FALLAS:=0;
       CONT:=0;
```

```
WHILE CONT <= NUM_TERM DO
        BEGIN
        ESTO[CONT]:=0;
        CONT:=CONT+1;
        END
(* END  DO *);
CONT:=0;
WHILE CONT <= NUM_TERM DO
        BEGIN
        SACAMIN(CONT);
        CASE MOD_TERM OF
                921:
                    BEGIN
                    INLINE($16/$01); (* LD D,01H *)
                    @COMUN;
                    INLINE($32/CARACTER);
                    END;
                922:
                    BEGIN
                    INLINE($16/$02); (* LD D,02H *)
                    @COMUN;
                    INLINE($32/CARACTER);
                    END;
                924:
                    BEGIN
                    INLINE($16/$04); (* LD D,04H *)
                    @COMUN;
                    INLINE($32/CARACTER);
                    END;
                955:
                    BEGIN
                    INLINE($16/$05); (* LD D,05H *)
                    @COMUN;
                    INLINE($32/CARACTER);
                    END;
        END (* CASE *);
        CAR:=CHR(CARACTER);
        IF CAR IN ['A','B','C','D'] THEN
                CASE CAR OF
                        'A':
                          BEGIN
                          FALLAS:=FALLAS+1;
                          ESTO[CONT]:=1;
                          END;
                        'B':
                          BEGIN
                          FALLAS:=FALLAS+1;
                          ESTO[CONT]:=2;
                          END;
```

```
                        'C':
                          (* PRUEBA CORRECTA *);
                        'D':
                          BEGIN
                          FALLAS:=FALLAS+1;
                          ESTO[CONT]:=3;
                          END;
                END (* CASE *)
        ELSE
                BEGIN
                FALLAS:=FALLAS+1;
                ESTO[CONT]:=3;
                END
        (* END IF *);
        CONT:=CONT+1;
        END
 (* END DO *);
 CONT:=0;
 AUX:=0;
 WRITELN(CHR($1A));
 IF FALLAS <> 0 THEN
        BEGIN
        WRITELN(CHR($1A));
        WRITELN;
        WRITELN;' FALLO COMUNICACION EN TERMINALES NUMERO:');
        WRITELN;
        END
 ELSE
        BEGIN
        WRITELN(CHR($1A));
        WRITELN;
        WRITELN;
        WRITELN;
        WRITELN;
        WRITELN('          =====)  PRUEBA CORRECTA  (===== ');
        WRITELN;
        WRITELN;
        WRITELN;
        WRITELN('                                                    (RETURN)');
        READ(CONTINUO);
        END
 (* END IF *);
WHILE CONT <= NUM_TERM DO
        BEGIN
        IF AUX = 6 THEN
                BEGIN
                AUX:=0;
                WRITELN('                                                    (RETURN)');
                READLN(CONTINUO);
```

```
                WRITELN(CHR($1A));
                WRITELN;
                WRITELN;
                END
        ELSE
                AUX:=AUX+1
        {$ END IF $};
        IF ESTO[CONT] IN [0,1,2,3] THEN
                CASE ESTO[CONT] OF
                        0:
                          {$ PRUEBA CORRECTA $};
                        1:
                          BEGIN
                          WRITELN('  NUMERO  ',CONT+1,' APAGADA O CABLES DESCONECTADOS');
                          WRITELN;
                          END;
                        2:
                          BEGIN
                          WRITELN('  NUMERO  ',CONT+1,' RECIBE BASURA');
                          WRITELN;
                          END;
                        3:
                          BEGIN
                          WRITELN('  NUMERO  ',CONT+1,' ERROR EN HARDWARE O TERMINAL APAGADA');
                          WRITELN;
                          END;
                END {$ CASE $}
        ELSE
                WRITELN('  ======> REALIZA NUEVAMENTE LA PRUEBA')
        {$ END IF $};
        CONT:=CONT+1;
        END
{$ END DO $};
WRITELN;
WRITELN;
IF FALLAS <> 0 THEN
        BEGIN
        WRITELN('  CHECAR QUE ESTEN PRENDIDAS, CONECTOR RS232-C Y CABLES');
        WRITE('                                                    <RETURN>');
        READLN(CONTINUO);
        END
{$ END IF $};
WRITELN;
WRITELN;
WRITELN(CHR($1A));
RESP:=' ';
WHILE (RESP <> 'S') AND (RESP <> 'N') DO
        BEGIN
        WRITE('  QUIERES QUE SE REPITA LA PRUEBA DE ',
```

```
                    'COMUNICACION (S/N) ? ');
                 READLN(RESP);
                 WRITELN;
                 IF (RESP <> 'S') AND (RESP <> 'N') THEN
                       WRITELN('   => Teclea unicamente S (si) o N (no)')
                 (* END IF *);
                 END
         (* END DO *);
         END
 (* END DO *);
 END
 (* END PROCEDURE PRUEBA0 *);

PROCEDURE PRUEBA1(VAR MOD_TERM,
                      NUM_TERM
                             :INTEGER);

(***********************************************)
(**                                           **)
(**      PRUEBA DE  AJUSTE DE PANTALLLA       **)
(**                                           **)
(***********************************************)

VAR
    CONT,               (*Direcciona terminal.                   *)
    FALLAS,             (*Indica el numero de fallas ocurridas.  *)
    AUX,                (*indica numero de terminal con error.   *)
    AUX2                (*Contador para contar 5 lecturas de error.*)
         :INTEGER;
    RESP,               (* =S realiza la prueba.                 *)
    CORREC,             (* Indica si los errores leidos son.     *)
                        (* correctos.                            *)
    SIGUE               (* Para el procedue hasta teclear (RETURN) *)
              :CHAR;
    NUM_INV             (* = TRUE numero invalido de terminales con*)
                        (*    error.                             *)
         :BOOLEAN;
BEGIN
WRITELN(CHR($1A));
RESP:='S';
WHILE RESP = 'S' DO
      BEGIN
      CONT:=0;
      WHILE CONT <= NUM_TERM DO
            BEGIN
            ESTI[CONT]:=0;
            CONT:=CONT+1;
            END
      (* END DO *);
```

```
FALLAS:=0;
CONT:=0;
WRITELN;
WRITELN;
WRITELN;
WRITELN;
WRITELN;
WRITELN('                          PRUEBA DE VIDEO');
WRITELN;
WRITELN('                          PATRON DE H''s');
WHILE CONT <= NUM_TERM DO
       BEGIN
       WHILE (ESTO[CONT]<>0) AND (CONT<=NUM_TERM) DO
              CONT:=CONT+1
       (* END DO *);
       IF CONT<=NUM_TERM THEN
              BEGIN
              SACAMIN(CONT);
              K:=1;
              WHILE K < 2000 DO
                     K:=K+1
              (* END DO *);
              CASE MOD_TERM OF
                     921:
                        BEGIN
                        INLINE($16/$01);          (* LD D,01H *)
                        @VIDH (* PATRON DE VIDEO *);
                        END;
                     922:
                        BEGIN
                        INLINE($16/$02);          (* LD D,02H *)
                        @VIDH (* PATRON DE VIDEO *);
                        END;
                     924:
                        BEGIN
                        INLINE($16/$04);          (* LD D,04H *)
                        @VIDH (* PATRON DE VIDEO *);
                        END;
                     955:
                        BEGIN
                        INLINE($16/$05);          (* LD D,05H *)
                        @VIDH (* PATRON DE VIDEO *);
                        END;
              END (* CASE *);
              CONT:=CONT+1;
              END
       (* END IF *);
       END
(* END DO *);
```

```
CONT:=0;
WRITELN;
WRITELN;
NUM_INV:=TRUE;
WHILE NUM_INV DO
        BEGIN
        WRITE('Cuantas terminales no se pudieron ajustar ?');
        READ(FALLAS);
        IF (FALLAS > NUM_TERM+1) OR (FALLAS < 0) THEN
                BEGIN
                WRITELN;
                WRITELN('INVALIDO NUMERO DE TERMINALES CON ERRROR: ',FALLAS)
                END
        ELSE
                NUM_INV:=FALSE
        (* END IF *);
        END
(* END DO *);
WRITELN(CHR($1A));
IF FALLAS <> 0 THEN
        BEGIN
        WRITELN;
        WRITELN;
        WRITELN('DAME LOS ',FALLAS,' NUMEROS DE LAS TERMINALES QUE FALLARON');
        WRITELN;
        WRITELN;
        END
(* END IF *);
AUX2:=0;
WHILE FALLAS > 0 DO
        BEGIN
        IF AUX2 = 6 THEN
                BEGIN
                AUX2:=0;
                WRITELN(CHR($1A));
                WRITELN;
                WRITELN;
                WRITELN('DAME LOS ',FALLAS,' NUMEROS DE LAS TERMINALES QUE FALLARON');
                WRITELN;
                WRITELN;
                END
        ELSE
                AUX2:=AUX2+1
        (* END IF *);
        WRITE('NUMERO DE TERMINAL: ');
        READLN(AUX);
        AUX:=AUX-1;
        WRITE('  NUMERO DE FALLA: ');
        READLN(EST[AUX]);
```

```
                WRITELN;
                FALLAS:=FALLAS-1;
                END
        (* END DO *);
        WRITE('                                                    <RETURN>');
        READLN(SIGUE);
        WRITELN(CHR($1A));
        RESP:=' ';
        WHILE (RESP <> 'S') AND (RESP <> 'N') DO
                BEGIN
                WRITE('  QUIERES QUE SE REPITA LA PRUEBA DE ',
                      'VIDEO (S/N) ? ');
                READLN(RESP);
                WRITELN;
                IF (RESP <> 'S') AND (RESP <> 'N') THEN
                        WRITELN('   => Teclea unicamente S (si) o N (no)')
                (* END IF *);
                END
        (* END DO *);
        END
(* END DO *);
END
(* END PROCEDURE PRUEBA1 *);


PROCEDURE PRUEBA2(VAR MOD_TERM,          (* Modelo de las terminales a probar     *)
                  NUM_TERM               (* Numero de las terminales a probar     *)
                                :INTEGER);

(*****************************************)
(**                                     **)
(**     REALIZACION DE LA AUTOPRUEBA    **)
(**                                     **)
(*****************************************)

CONST
        MAX=1650;       (* Numero maximo de caracteres en 1 pantalla.  *)

VAR
    F                   (* Nombre del archivo con el que se comparara  *)
                        (* la pantalla recibida.                       *)
                :TEXT;
    I,                  (* Cuenta de 0 hasta el numero maximo de carac *)
                        (* teres de una pantalla.                      *)
    CONT,               (* Direcciona una terminal 1<=CONT<=256.       *)
    AUX,                (* Para desplegar los resultados en bloques de *)
                        (* 6 fallas, como maximo, por pantalla.        *)
    FALLAS              (* Cuenta el numero de terminales que fallo.   *)
                :INTEGER;
```

```
    BUFF                ($ Almacena un caracter leido del archivo:     $)
                        ($           PANTALLA.XXX                      $)
                :BYTE;
    SIGUE,              ($ Detiene el procedure hasta teclear <RETURN> $)
    RESP,               ($ ='S' Se repite la prueba.                   $)
    VARIABLE            ($ Almacena el caracter recibido de la terminal$)
                :CHAR;
    ERROR               ($ Indica la existencia de un error.           $)
                :BOOLEAN;

BEGIN
WRITELN(CHR($1A));
CASE MOD_TERM OF
       921:
           ASSIGN(F,'PANTALLA.921');
       922:
           ASSIGN(F,'PANTALLA.922');
       924:
           ASSIGN(F,'PANTALLA.924');
       955:
           ASSIGN(F,'PANTALLA.955');
END ($ CASE $);
RESP:='S';
WHILE RESP='S' DO
       BEGIN
       CONT:=0;
       WHILE CONT <= NUM_TERM DO
              BEGIN
              EST2[CONT]:=0;
              CONT:=CONT+1;
              END
       ($ END DO $);
       CONT:=0;
       FALLAS:=0;
       WHILE CONT<= NUM_TERM DO
              BEGIN
              WRITELN;
              RESET(F);
              WRITELN;
              WRITELN;
              WRITELN(CHR($1A),'          $$$$$$$$$$  COMPARANDO ARCHIVO $$$$$$$$$$');
              WRITELN;
              WHILE (EST0[CONT]<>0) AND (CONT<=NUM_TERM) DO
                     CONT:=CONT+1
              ($ END DO $);
              IF CONT<=NUM_TERM THEN
                     BEGIN
                     SACAWIN(CONT);
                     CASE MOD_TERM OF
```

```
            921:
                BEGIN
                @ISIOC;
                INLINE($16/$01);    (* LD B,01H *)
                @RECU;
                END;
            922:
                BEGIN
                @ISIOC;
                INLINE($16/$02);    (* LD D,02H *)
                @RECU;
                END;
            924:
                BEGIN
                @ISIOC;
                INLINE($16/$04);    (* LD D,04H *)
                @RECU;
                END;
            955:
                BEGIN
                @ISIOC;
                INLINE($16/$05);    (* LD D,05H *)
                @RECU;
                END;
        END (* CASE *);
        FOR I:=0 TO MAX DO
            BEGIN
            INLINE($FD/$7E/00);     (* LD A,(IY+00)   *)
            INLINE($32/VARIABLE);   (* LD(VARIABLE),A *)
            INLINE($FD/$23);        (*    INC IY      *)
            READ(F,BUFF);
            IF BUFF<> VARIABLE THEN
                    BEGIN
                    WRITELN('ERROR');
                    ERROR:=TRUE;
                    END
            (* END IF *);
            END
        (* END FOR *);
        IF ERROR THEN
            BEGIN
            ESTZ[CONT]:=1;
            FALLAS:=FALLAS+1;
            END
        (* END IF *);
        CONT:=CONT+1;
        END
(* END IF *);
END.
```

```
(* END DO *);
IF FALLAS <> 0 THEN
      BEGIN
      WRITELN;
      WRITELN('     FALLO AUTOPRUEBA EN TERMINALES NUMERO: ');
      WRITELN;
      CONT:=0;
      AUX:=0;
      WHILE CONT <= NUM_TERM DO
            BEGIN
            IF AUX = 6 THEN
                  BEGIN
                  AUX:=0;
                  WRITE('                                                    <RETURN>');
                  READLN(SIGUE);
                  WRITELN(CHR($1A));
                  WRITELN;
                  WRITELN;
                  END
            ELSE
                  AUX:=AUX+1
            (* END IF *);
            IF EST2[CONT] <> 0 THEN
                  BEGIN
                  WRITELN;
                  WRITELN('  NUMERO  ',CONT+1,' FALLO AUTOPRUEBA');
                  END
            (* END IF *);
            CONT:=CONT+1;
            END
      (* END DO *);
      END
ELSE
      BEGIN
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN('                  =====) AUTOPRUEBA CORRECTA (=====');
      WRITE('                                                <RETURN>');
      READLN(SIGUE);
      END
(*END IF*);
WRITELN;
WRITELN;
WRITELN(CHR($1A));
RESP:=' ';
WHILE (RESP <> 'S') AND (RESP <> 'N') DO
      BEGIN
      WRITELN;
```

```
                WRITELN;
                WRITELN;
                WRITE('                    QUIERES QUE SE REPITA LA AUTOPRUEBA (S/N) ? ');
                READLN(RESP);
                WRITELN;
                IF (RESP <> 'S') AND (RESP <> 'N') THEN
                       WRITELN('                    => Teclea unicamente S (si) o N (no)')
                (* END IF *);
                END
         (* END DO *);
         END
  (* END DO *);
  END
  (* END PROCEDURE PRUEBA2 *);


  PROCEDURE PRUEBA3(VAR MOD_TERM,           (* Numero de las terminales a probar *)
                        NUM_TERM            (* Modelo de las terminales a probar *)
                                 :INTEGER);

  (**********************************************)
  (**                                          **)
  (**       PRUEBA CONTINUA DE QUEMADO         **)
  (**                                          **)
  (**********************************************)

  VAR
     CONT,               (*Direcciona terminal.                      *)
     FALLAS,             (*Indica el numero de fallas ocurridas.     *)
     AUX,                (*Variable para contar 5 lecturas de error.*)
     VARIABLE,           (*=0 No ocurrio falla en "quemado".         *)
     VECES               (*Numero de veces que se ejecutara la prue-*)
                         (*ba de quemado continua de terminales.     *)
          :INTEGER;
     CORREC              (* Indica si los errores leidos son         *)
                         (* correctos.                               *)
          :CHAR;
     NUM_INV             (* = TRUE error en numero de terminales con*)
                         (* error.                                   *)
        :BOOLEAN;
  BEGIN
  WRITELN(CHR($1A));
  FALLAS:=0;
  WRITE(' Cuantas veces quieres que se repita la prueba de quemado? ');
  READLN(VECES);
  WHILE VECES > 0 DO
        BEGIN
        FALLAS:=0;
        WRITELN(CHR($0A),CHR($0A),'                         PRUEBA CONTINUA');
```

```
WRITELN(CHR($0A),'                    DE QUEMADO DE TERMINALES');
CONT:=0;
WHILE CONT <= NUM_TERM DO
       BEGIN
       WHILE (ESTO[CONT]<>0) AND (CONT<=NUM_TERM) DO
              CONT:=CONT+1
       {$ END DO $};
       IF CONT<=NUM_TERM THEN
              BEGIN
              SACAWIN(CONT);
              CASE NOD_TERM OF
                     921:
                        BEGIN
                        INLINE($16/$01);          {$ LD D,01H $}
                        QQUEMA;
                        INLINE($32/VARIABLE);     {$ LD VARIABLE,A $}
                        END;
                     922:
                        BEGIN
                        INLINE($16/$02);          {$ LD D,02H $}
                        QQUEMA;
                        INLINE($32/VARIABLE);     {$ LD VARIABLE,A $}
                        END;
                     924:
                        BEGIN
                        INLINE($16/$04);          {$ LD D,04H $}
                        QQUEMA;
                        INLINE($32/VARIABLE);     {$ LD VARIABLE,A $}
                        END;
                     955:
                        BEGIN
                        INLINE($16/$05);          {$ LD D,05H $}
                        QQUEMA;
                        INLINE($32/VARIABLE);     {$ LD VARIABLE,A $}
                        END;
              END {$ CASE $};
              IF VARIABLE <> 0 THEN
                     BEGIN
                     EST3[CONT]:=VARIABLE;
                     FALLAS:=FALLAS+1;
                     END
              {$ END IF $};
              CONT:=CONT+1;
              END
       {$ END IF $};
       END
{$ END DO $};
CONT:=0;
WRITELN(CHR($1A));
```

```
        IF FALLAS <> 0 THEN
                BEGIN
                WRITELN(FALLAS,' Errores en prueba de quemado.' );
                WRITELN;
                WRITELN;
                END
        (* END IF *);
        AUX:=0;
        CONT:=0;
        WHILE CONT <= NUM_TERM DO
                BEGIN
                IF AUX = 6 THEN
                        BEGIN
                        AUX:=0;
                        WRITELN(CHR(#1A));
                        WRITELN;
                        WRITELN(FALLAS,' Error en prueba de quemado.');
                        WRITELN;
                        WRITELN;
                        END
                ELSE
                        AUX:=AUX+1
                (* END IF *);
                IF EST3[CONT] <> 0 THEN
                        WRITELN('FALLO TERMINAL NUMERO: ',CONT+1);
                WRITELN;
                CONT:=CONT+1;
                END
        (* END DO *);
        VECES:=VECES-1;
        END
(* END DO *);
END
(* END PROCEDURE PRUEBA3 *);


PROCEDURE PRUEBA4(VAR MOD_TERM,          (* Modelo de las terminales a probar      *)
                  NUM_TERM               (* Numero de las terminales a probar      *)
                                :INTEGER);

(*****************************************)
(**                                     **)
(**    PRUEBA DE LA LOGICA DE VIDEO     **)
(**                                     **)
(*****************************************)

VAR
   CONT,                 (* Direcciona una terminal 1<=CONT<=256.       *)
   AUX,                  (* Para desplegar los resultados en bloques de *)
```

```
                    (* 6 fallas, como maximo, por pantalla.        *)
    FALLAS,         (* Cuenta el numero de terminales que fallo.   *)
    VARIABLE        (* Almacena el caracter recibido de la terminal*)
                :INTEGER;
    SIGUE,          (* Detiene el procedure hasta teclear <RETURN> *)
    RESP            (* ='S' Se repite la prueba.                   *)
                :CHAR;
    ERROR           (* Indica la existencia de un error.           *)
                :BOOLEAN;

BEGIN
WRITELN(CHR($1A));
RESP:='S';
WHILE RESP='S' DO
        BEGIN
        CONT:=0;
        WHILE CONT <= NUM_TERM DO
                BEGIN
                EST4[CONT]:=0;
                CONT:=CONT+1;
                END
        (* END DO *);
        CONT:=0;
        FALLAS:=0;
        WHILE CONT<= NUM_TERM DO
                BEGIN
                WRITELN;
                WRITELN;
                WRITELN;
                WRITELN;
                WHILE (ESTO[CONT]<>0) AND (CONT<=NUM_TERM) DO
                        CONT:=CONT+1
                (* END DO *);
                IF CONT<=NUM_TERM THEN
                        BEGIN
                        SACAWIN(CONT);
                        CASE MOD_TERM OF
                                921:
                                    BEGIN
                                    INLINE($16/$01);          (* LD D,01H *)
                                    BLOS;
                                    INLINE($32/VARIABLE);     (* LD(VARIABLE),A *)
                                    WRITELN('....>>>>',VARIABLE);
                                    END;
                                922:
                                    BEGIN
                                    INLINE($16/$02);          (* LD D,02H *)
                                    BLOS;
                                    INLINE($32/VARIABLE);     (* LD(VARIABLE),A *)
```

```
                    END;
                924:
                    BEGIN
                    INLINE(016/004);          (* LD B,04H *)
                    BLOG;
                    INLINE(032/VARIABLE);     (* LD(VARIABLE),A *)
                    END;
                955:
                    BEGIN
                    INLINE(016/005);          (* LD B,05H *)
                    BLOG;
                    INLINE(032/VARIABLE);     (* LD(VARIABLE),A *)
                    END;
            END (* CASE *);
            IF  VARIABLE<>0 THEN
                    BEGIN
                    EST4[CONT]:=1;
                    FALLAS:=FALLAS+1;
                    END
            (* END IF *);
            END
        (* END IF *);
        CONT:=CONT+1;
        END
(* END DO *);
IF FALLAS <> 0 THEN
        BEGIN
        WRITELN;
        WRITELN('     FALLO LOGICA EN TERMINALES NUMERO: ');
        WRITELN;
        CONT:=0;
        AUX:=0;
        WHILE CONT <= NUM_TERM DO
            BEGIN
            IF AUX = 6 THEN
                    BEGIN
                    AUX:=0;
                    WRITE('                                                    <RETURN>');
                    READLN(SIGUE);
                    WRITELN(CHR(01A));
                    WRITELN;
                    WRITELN;
                    END
            ELSE
                    AUX:=AUX+1
            (* END IF *);
            IF EST4[CONT] <> 0 THEN
                    BEGIN
                    WRITELN;
```

```
                          WRITELN('  NUMERO  ',CONT+1,' FALLO LOGICA');
                          END
                     (* END IF *);
                     CONT:=CONT+1;
                     END
                (* END DO *);
                END
     ELSE
                BEGIN
                WRITELN;
                WRITELN;
                WRITELN;
                WRITELN('                    =====> LOGICA CORRECTA <=====');
                END
     (*END IF*);
     WRITELN;
     WRITELN;
     WRITE('                                                       <RETURN>');
     READLN(SIGUE);
     WRITELN(CHR($1A));
     RESP:=' ';
     WHILE (RESP <> 'S') AND (RESP <> 'N') DO
          BEGIN
          WRITELN;
          WRITELN;
          WRITELN;
          WRITE('               QUIERES QUE SE REPITA LA PRUEBA DE LOGICA DE VIDEO (S/N) ? ');
          READLN(RESP);
          WRITELN;
          IF (RESP <> 'S') AND (RESP <> 'N') THEN
               WRITELN('            => Teclea unicamente S (si) o N (no)')
          (* END IF *);
          END
     (* END DO *);
     END
(* END DO *);
END
(* END PROCEDURE PRUEBA4 *);


PROCEDURE PRUEBA(VAR MOD_TERM,          (* Modelo de las terminales a probar *)
                     NUM_TERM           (* Numero de las terminales a probar *)
                              :INTEGER);


(****************************************)
(**                                    **)
(**     REALIZACION DE LAS PRUEBAS     **)
(**                                    **)
(****************************************)
```

```
VAR
    NUM_PRUEBA          (* Numero de prueba a realizar (ver menu)      *)

              :INTEGER;
    SIGUE               (* Detiene el procedure hasta teclear (return) *)
              :CHAR;
BEGIN
PRUEBA0(MOD_TERM,NUM_TERM);
NUM_PRUEBA:=0;
WHILE NUM_PRUEBA <> 7 DO
       BEGIN
       WRITELN(CHR($1A));
       WRITELN;
       WRITELN(' MENU DE PRUEBA:');
       WRITELN('                1.- Video.');
       WRITELN('                 2.- Autoprueba.');
       WRITELN('                3.- Comunicacion.');
       WRITELN('                 4.- Continua de quemado.');
       WRITELN('                5.- Logica de video.');
       WRITELN('                 6.- Todas.');
       WRITELN('                7.- Salida del menu.');
       WRITELN;
       WRITELN;
       WRITELN;
       WRITE('QUE PRUEBA QUIERES REALIZAR ??');
       READLN(NUM_PRUEBA);
       WRITELN;
       IF NUM_PRUEBA IN [1,2,3,4,5,6,7] THEN
              CASE NUM_PRUEBA OF
                     1:
                       PRUEBA1(MOD_TERM,NUM_TERM); (*      Video      *)
                     2:
                       PRUEBA2(MOD_TERM,NUM_TERM); (*   Autoprueba    *)
                     3:
                       PRUEBA0(MOD_TERM,NUM_TERM); (*  Comunicacion   *)
                     4:
                       PRUEBA3(MOD_TERM,NUM_TERM); (*     Quemado     *)
                     5:
                       PRUEBA4(MOD_TERM,NUM_TERM); (* Logica de video *)
                     6:
                       BEGIN                       (*    Las cinco    *)
                       PRUEBA0(MOD_TERM,NUM_TERM);
                       PRUEBA1(MOD_TERM,NUM_TERM);
                       PRUEBA2(MOD_TERM,NUM_TERM);
                       PRUEBA3(MOD_TERM,NUM_TERM);
                       PRUEBA4(MOD_TERM,NUM_TERM);
                       END;
```

```
                        BEGIN
                        WRITELN;
                        WRITELN;
                        WRITELN('FIN DE PRUEBA-S');
                        K:=1;
                        WHILE K<20000 DO
                              K:=K+1
                        (* END DO *);
                        WRITELN(CHR(*10),*,CHR(*27),CHR(*1A));
                        END;
                END (* CASE *)
        ELSE
                BEGIN
                WRITELN('ERROR EN NUMERO DE : PRUEBA: ',NUM_PRUEBA);
                WRITE('                                                    (RETURN)');
                READ(SIGUE);
                END
        (*END IF*);
END (*DO*);
END;

(*END DE PROCEDURE PRUEBA*)

BEGIN
LEE_DATOS(MOD_TERM,NUM_TERM);
INICIALIZACION(MOD_TERM,NUM_TERM);
PRUEBA(MOD_TERM,NUM_TERM);
(*      WRITE('DAME EL NOMBRE DEL ARCHIVO A CREAR  CON LOS RESULTADOS DE LA PRUEBA: ');
READ(NOMBRE);
ASSIGN(ARCH,NOMBRE);
REWRITE(ARCH);
WRITELN('ESCRIBIENDO ARCHIVO');
REG:=0;
WHILE REG<256 DO
        BEGIN
        WRITELN(ARCH,EST0[REG]);
        WRITELN(ARCH,EST1[REG]);
        WRITELN(ARCH,EST2[REG]);
        WRITELN(ARCH,EST3[REG]);
        REG:=REG+1;
        END
(* END DO ;
CLOSE(F,N);
WRITELN('YA TERMINE');          *)
END.
```

APENDICE C

RUTINAS EN ENSAMBLADOR

```
        .Z80
        PUBLIC @ISIOC
        PUBLIC @COMUN
        PUBLIC @VIDM
        PUBLIC @RECU
        PUBLIC @QUEMA
        PUBLIC @LOG
@ISIOC:
        .Z80
;*************************************************
;*****                                       *****
;*****  VARIABLES PARA INICIALIZACION SIO    *****
;*****              Y CTC                    *****
;*****                                       *****
;*************************************************

DATOSIO EQU     20H     ;Puerto de datos canal A
COMSIO  EQU     22H     ;Puerto de comandos canal A
COMCTC  EQU     08H     ;Canal cero del CTC

;*************************************
;*****                         *******
;*****  INICIALIZACION DEL SIO *******
;***** PARA LECTURA Y ESCRITURA *******
;*****                         *******
;*************************************

        LD      HL,TBLSIO
        LD      C,COMSIO
        LD      B,7
        OTIR

;*************************************
;*****                         *******
;*****  INICIALIZACION CTC     *******
;*****                         *******
;*************************************

        LD      HL,TBLCTC
        LD      C,COMCTC
        LD      B,02H
        OTIR
        RET

;*************************************
;*****                         *******
;*****  LOCALIDADES AUXILIARES *******
;*****                         *******
;*************************************
```

```
TBLSIO:
        DB      18H     ;Apunta a WR0 y reset al canal
        DB      04H     ;Apunta a WR4
        DB      10000100B       ;Sin paridad, un bit de paro,a 1/32 del reloj
        DB      03H     ;Apunta a WR3
        DB      0E1H    ;Habilita recepcion 8 bits/caracter,habilita
                        ;Data Carrier Detect y Clear To Send, habilita
                        ;recepcion
        DB      05H     ;Apunta a WR5
        DB      0EAH    ;Habilita transmision 8 bits/caracter,Data
                        ;Terminal Ready,Request To Send y habilita
                        ;transmision

TBLCTC:
        DB      47H     ;Modo contador, flanco negativo decrementa el conta-
                        ;dor , la constante de tiempo sigue y se reinicializa
                        ;el contador cargando la constante a la cuenta
                        ;de cero
        DB      01H     ;constante de tiempo para lograr 19 200 bauds
                        ;  La computadora tiene un oscilador de 16 Mhertz,
                        ;esta frecuencia es dividida entre 4 por un contador
                        ;y luego entre 6.5 por otro mas. Esta ultima frecuen-
                        ;cia entra al CTC y su salida al SIO que la divide
                        ;finalmente para lograr los 19 200 bauds. Esto se a-
                        ;precia en los diagramas que siguen al programa.

;*****************************************************************

SECMEM:
        .Z80
; ***************************************************
; *****                                         *****
; *****  VARIABLES PARA MANEJO DE PANTALLA      *****
; *****                                         *****
; ***************************************************


BDOS    EQU     05H     ;Direccion de las rutinas de Basic disk operation
                        ;systems
ESC     EQU     1BH     ;Escape

; ***************************************************
; *****  VARIABLES PARA MANEJO DEL SIO          *****
; ***************************************************

DATOSIO EQU     20H     ;Puerto de datos canal A
COMSIO  EQU     22H     ;Puerto de comandos canal A
```

```
; ##########################################
; #####                              #####
; #####      PROGRAMA PRINCIPAL      #####
; #####                              #####
; ##########################################

;################RECUPERA DATO QUE MANDA LA RUTINA PRINCIPAL#################

        LD      A,D             ;Recupera el dato y
        LD      (TTPO),A                ;lo guarda. El dato nos da el tipo de terminal:
                                ;       1 = 921
                                ;       2 = 922
                                ;       4 = 924
                                ;       5 = 955

;########################################
;#      ESTA LA TERMINAL PRENDIDA?      #
;########################################
        LD      C,0FFH
ENCIEN:
        LD      B,0FFH
CICLUS:
        LD      A,1AH           ;ENVIA A TERMINAL CTRL Z, LETRA H E INDICA
        CALL    ENVIA3          ;QUE REGRESE LA LETRA CON EL ESC CORRESPON
                                ;DIENTE, ADEMAS DE MANDAR UNLOCK KEYBOARD:
                                ;ESC #
        LD      A,ESC
        CALL    ENVIA3
        LD      A,'#'
        CALL    ENVIA3
        LD      A,'H'
        CALL    ENVIA3

        LD      A,(TTPO)
        CP      5
        JP      Z,TT955
        CP      2
        JP      Z,TT922
        CP      4
        JP      Z,TT924
TT921:
        LD      A,ESC
        CALL    ENVIA3
        LD      A,'4'           ;La terminal 921 necesita ESC 4
        CALL    ENVIA3
        JP      PREN
TT922:
;                       No hay manual que nos lo diga
```

```
        JP      PREN
TT924:
        LD      A,ESC
        CALL    ENVIA3
        LD      A,'S'           ;La terminal 924 necesita ESC S 1
        CALL    ENVIA3
        LD      A,'1'
        CALL    ENVIA3
        JP      PREN
TT955:
        LD      A,ESC
        CALL    ENVIA3          ;La terminal 955 necesita ESC 4
        LD      A,'4'
        CALL    ENVIA3
PREN:
        IN      A,(COMSIO)
        AND     01H             ;HAY DATO DISPONIBLE?
        JP      NZ,SIGUE        ;SI LO HAY SALTA A LEERLO
        DJNZ    CICLO5
NOFUE:
        LD      A,41H           ;PERO SI NO HUBO DATO, PUEDE QUE TERMINAL
        RET                     ;ESTE APAGADA O NO TENGA CONECTOR, ETC.
SIGUE:
        IN      A,(DATOSIO)     ;PERO SI LO HAY...
        CP      'H'             ;LO COMPARA CON UNA H Y SI SON IGUALES
        JR      Z,CONTI         ;SALTA
        LD      B,C             ;DE LO CONTRARIO VA A "CACHAR" UNA H
        DEC     C
        DJNZ    ENCIEN
        LD      A,42H
        RET
CONTI:
        LD      A,43H           ;LA TERMINAL ESTA PRENDIDA Y SE ENTABLO
                                ;COMUNICACION
        RET
;$$$$$ ENVIO DE INFORMACION AL PUERTO DE DATOS DEL SIO $$$$$

ENVIA3:
        OUT     (DATOSIO),A
        PUSH    BC
        LD      B,0FFH
ENVIA4:
        DJNZ    ENVIA5
        JR      ENVIA6
ENVIA5:
        NOP
        IN      A,(COMSIO)
        AND     4               ;Ve si esta vacio el buffer, si no espera...
        JR      Z,ENVIA4
```

```
        POP     BC
        RET
ENVIA6:
        POP     BC              ;RECUPERA BC
        POP     BC              ;RECUPERA DIRECCION DE REGRESO DENTRO DE
                                ;@COMUN PARA QUE DE ESA MANERA NO PUEDA
                                ;A ELLA
        LD      A,44H           ;BANDERA PARA ERROR EN HARDWARE EN TERMINAL
                                ;O TERMINAL APAGADA
        RET
TTPO:
        DB      00
;###########################################################################

@VIDH:
        .Z80

; ##############################################
; #####                                    #####
; #####  VARIABLES PARA MANEJO DE PANTALLA #####
; #####                                    #####
; ##############################################


BDOS    EQU     05H     ;Direccion de las rutinas de Basic disk operation
                        ;systems
ESC     EQU     1BH     ;Escape
CR      EQU     0DH     ;Carriage return
LF      EQU     0AH     ;Line feed

; ##############################################
; #####  VARIABLES PARA MANEJO DEL SIO      #####
; ##############################################

DATOSIO EQU     20H     ;Puerto de datos canal A
COMSIO  EQU     22H     ;Puerto de comandos canal A

; #########################################
; #####                             #####
; #####       PROGRAMA PRINCIPAL    #####
; #####                             #####
; #########################################

;################RECUPERA DATO QUE MANDA LA RUTINA PRINCIPAL################

        LD      A,D             ;Recupera el dato y
        LD      (TIP),A         ;lo guarda. El dato nos da el tipo de terminal:
                                ;       1 = 921
                                ;       2 = 922
```

```
                        ;       4 = 924
                        ;       5 = 955

        LD      A,01AH  ;CTRL Z
        CALL    ENVIA
        LD      C,23
LOOP:
        LD      D,80
LOOP1:
        LD      A,'H'
        CALL    ENVIA   ;Envia una "H" hasta llenar 23 lineas de pantalla
        LD      B,D
        DEC     B
        DJNZ    LOOP1
        LD      B,C
        DEC     C
        DJNZ    LOOP
LOOP2:
        LD      B,78
LOOP3:
        LD      A,'H'
        CALL    ENVIA   ;Envia una "H" hasta llenar una linea
        LD      B,D
        DEC     B
        DJNZ    LOOP3

        LD      A,(TIP)
        CP      5
        JP      Z,T955
        CP      2
        JP      Z,T922
        CP      4
        JP      Z,T924
T921:
;                       La terminal 921 no tiene ESC para mandar el
;                       status line (?)
        JP      REGRESA
T922:
;                       No hay manual que nos lo diga
        JP      REGRESA
T924:
        LD      A,ESC
        CALL    ENVIA
        LD      A,'s'
        CALL    ENVIA
        LD      A,'2'
        CALL    ENVIA
        JP      REGRESA
T955:
```

```
        LD      A,ESC
        CALL    ENVIA
        LD      A,'I'
        CALL    ENVIA
        LD      A,'S'
        CALL    ENVIA
        LD      A,':'
        CALL    ENVIA
        LD      A,'1'
        CALL    ENVIA
        LD      A,'v'
        CALL    ENVIA
REGRESA:
        RET
; #####  ENVIO DE INFORMACION AL PUERTO DE DATOS DEL SIO  #####

ENVIA:
        OUT     (DATOSIO),A
ENVIA0:
        NOP
        IN      A,(COMSIO)
        AND     4               ;Ve si esta vacio el buffer, si no espera...
        JR      Z,ENVIA0
        RET
TIP:
        DB      00

;#######################################################################################

@RECU:
        .Z80
;       TITLE   'LECTURA DE LA PANTALLA DE LAS TERMINALES 921, 922 Y 955  A TRAVES DEL SIO'


;         Este programa es una subrutina que inicializa el SIO y el CTC
;       de manera de leer la pagina uno de memoria,que tambien es desplegada
;       por pantalla, a traves del SIO en modo Polled y escribir la informa-
;       cion en memoria RAM en la direccion de memoria apuntada por BUFFER.
;         El programa regresa a el programa principal con la direccion de
;       BUFFER, para que en el sea grabado en disco. El programa principal
;       esta en PASCAL MTPLUS.

;##################RECUPERA DATO QUE MANDA LA RUTINA PRINCIPAL##################

        LD      A,0             ;Recupera el dato y
        LD      (TIPO),A        ;lo guarda. El dato nos da el tipo de terminal:
                                ;       1 = 921
                                ;       2 = 922
                                ;       4 = 924
```

```
                                ;       5 = 955
; ************************************************
; *****                                    *****
; *****  VARIABLES PARA MANEJO DE PANTALLA *****
; *****                                    *****
; ************************************************

BDOS    EQU     05H     ;Direccion de las rutinas de Basic disk operation
                        ;systems
ESC     EQU     1BH     ;Escape
CR      EQU     0DH     ;Carriage return

; ************************************************
; *****  VARIABLES PARA MANEJO DEL SIO     *****
; ************************************************

DATOSIO EQU     20H     ;Puerto de datos canal A
COMSIO  EQU     22H     ;Puerto de comandos canal A

        IN      A,(DATOSIO)     ;Lee cuatro veces
        IN      A,(DATOSIO)     ;Para limpiar el SIO
        IN      A,(DATOSIO)
        IN      A,(DATOSIO)

; ************************************************
; *****                                  *****
; *****       PROGRAMA PRINCIPAL         *****
; *****                                  *****
; ************************************************
INICIO:
        LD      A,(TIPO)
        CP      5
        JP      Z,IN955
        CP      2
        JP      Z,IN922
        CP      4
        JP      Z,IN924
IN921:
        LD      A,ESC   ;Manda autoprueba, en la terminal se genera
        CALL    SEND    ;la pantalla
        LD      A,'V'
        CALL    SEND
        LD      A,ESC
        CALL    SEND
        LD      A,'7'   ;Manda que regrese pantalla ESC 7
        CALL    SEND
        JP      EMPEZAR
IN922:
```

```
        LD      A,ESC   ;Manda autoprueba, en la terminal se genera
        CALL    SEND    ;la pantalla
        LD      A,'V'
        CALL    SEND
        LD      A,ESC
        CALL    SEND
        LD      A,'7'   ;Manda que regrese pantalla ESC 7
        CALL    SEND
        JP      EMPEZAR
IN924:
        LD      A,ESC   ;Manda autoprueba, en la terminal se genera
        CALL    SEND    ;la pantalla
        LD      A,'V'
        CALL    SEND
        LD      A,ESC
        CALL    SEND
        LD      A,'5'   ;Manda que regrese pantalla con ESC S ?
        CALL    SEND
        LD      A,'?'
        CALL    SEND
        JP      EMPEZAR
IN955:
        LD      A,ESC   ;Manda autoprueba, en la terminal se genera
        CALL    SEND
        LD      A,'V'
        CALL    SEND
        LD      A,ESC
        CALL    SEND
        LD      A,'7'   ;Manda que regrese pantalla con ESC 7
        CALL    SEND
EMPEZAR:
        LD      IX,BUFFER       ;Guarda la direccion en el apuntador IX
ETI1:
        LD      B,0     ;Se van a admitir datos del SIO hasta que se
        CALL    RECEIVE ;encuentra una secuencia TVS, que son los ultimos
        CP      'T'     ;caracteres de la pantalla.
        JR      NZ,OTRO
        LD      B,1
        CALL    OTRO
        CALL    RECEIVE
        CP      'V'
        JR      NZ,OTRO
        CALL    RECEIVE
        CP      'S'
        JR      NZ,OTRO
        JR      PROCE   ;Si encontro secuencia salta
OTRO:
        LD      (IX+0),A        ;Si no ha encontrado secuencia se almacenan
        INC     IX              ;los datos en donde apunta IX
```

```
        LD      A,B
        CP      1
        JR      NZ,ETI1
        LD      B,0
        RET

PROCE:
        LD      HL,BUFFER
        LD      (GUAR),HL
        LD      IY,(GUAR)       ;Se carga en IY la direccion inicial
                                ;en donde se escribio la pantalla y regresa
        RET                     ;al programa principal.

; #####  ENVIO DE INFORMACION AL PUERTO DE DATOS DEL SIO  #####

SEND:
        OUT     (DATOSIO),A
SENDO:
        NOP
        IN      A,(CONSIO)
        AND     4               ;Ve si esta vacio el buffer, si no espera...
        JR      Z,SENDO
        RET

;#####  RECEPCION DE INFORMACION POR EL PUERTO DE DATOS DEL SIO  #####

RECEIVE:
        IN      A,(CONSIO)
        AND     1
        JR      Z,RECEIVE       ;Pregunta si hay datos en recepcion,si no lo
                                ;hay espera...
        IN      A,(DATOSIO)     ;de lo contrario lo lee
        LD      C,A             ;salva el dato
        LD      A,01H           ;y pregunta si no encontro bit de paro, si la

        OUT     (CONSIO),A      ;paridad es correcta o si no hubo error de
        IN      A,(CONSIO)      ;sobrecarrera, bits 6, 4 y 5 del registro de
        LD      B,70H           ;lectura de comandos 1 respectivamente.
        AND     B
        JR      Z,CORREC        ;si todo esta correcto salta
        LD      B,0C0H          ;de lo contrario..
        LD      DE,0FFFFH       ;espera a que se acabe de enviar pantalla y..
DELAY:
        DEC     E
        JP      NZ,DELAY
        DEC     D
        JP      NZ,DELAY
        DJNZ    DELAY
        JP      INICIO          ;..ejecuta de nuevo el programa
```

```
CORREC:
        LD      A,C             ;salva el dato y regresa
        RET

; ##############################################
; #####                                  #####
; #####      LOCALIDADES AUXILIARES      #####
; #####                                  #####
; ##############################################

GUAR:
        DW      0000H
TIPO:
        DB      00
BUFFER:
        DS      0800H   ;buffer de 2048 bytes

        RET
; ##############################################################################

ORGEN:
        .Z80

; ##############################################
; #####                                  #####
; #####  VARIABLES PARA MANEJO DEL SIO   #####
; #####           Y PANTALLA             #####
; ##############################################

DATOSIO EQU     20H     ;Puerto de datos canal A
COMSIO  EQU     22H     ;Puerto de comandos canal A

BDOS    EQU     05H     ;Direccion de las rutinas de Basic disk operation
                        ;systems
ESC     EQU     1BH     ;Escape
CR      EQU     0DH     ;Carriage return
LF      EQU     0AH     ;Line feed

; ##############################################
; #####                                  #####
; #####       PROGRAMA PRINCIPAL         #####
; #####                                  #####
; ##############################################

        LD      A,B             ;Recupera el dato y
        LD      (TPO),A         ;lo guarda. El dato nos da el tipo de terminal:
                                ;       1 = 921
                                ;       2 = 922
                                ;       4 = 924
```

```
;       5 = 955

        LD      L,22H
MAIN:
        LD      A,01AH  ;CTRL Z
        CALL    ENV
        LD      C,23
CICLO:
        LD      D,80
CICLO1:
        LD      A,L
        CALL    ENV     ;Envia un caracter hasta llenar 23 lineas de pantalla
        LD      B,D
        DEC     D
        DJNZ    CICLO1
        LD      B,C
        DEC     C
        DJNZ    CICLO
CICLO2:
        LD      D,79
CICLO3:
        LD      A,L
        CALL    ENV     ;Envia un caracter hasta llenar linea
        LD      B,D
        DEC     D
        DJNZ    CICLO3

        LD      A,(TPO)
        CP      4
        JP      Z,E924
;                       ESC 7 para que regrese la pantalla las terminales
;                       921,922 y 955
        LD      A,ESC
        CALL    ENV
        LD      A,'7'
        CALL    ENV
        JR      COMPARA
E924:
        LD      A,ESC
        CALL    ENV
        LD      A,'S'
        CALL    ENV
        LD      A,'7'
        CALL    ENV

;**********************COMPARA LO MANDADO CON LO RECIBIDO*****************

COMPARA:
        LD      H,L
```

```
        LD      C,23
CICLO4:
        LD      B,80
CICLO5:
        CALL    RECIBE
        CP      H
        JP      NZ,ERROR
        LD      B,D
        DEC     B
        DJNZ    CICLO5
        LD      B,C
        DEC     C
        DJNZ    CICLO4
        LD      D,79
CICLO6:
        CALL    RECIBE
        CP      H
        JP      NZ,ERROR
        LD      B,D
        DEC     B
        DJNZ    CICLO6
        LD      A,L
        INC     L
        INC     L
        CP      7EH
        JP      NZ,AGAIN
        LD      A,0
        JR      FIN
ERROR:
        LD      A,1
FIN:
        RET
; ***** ENVIO DE INFORMACION AL PUERTO DE DATOS DEL SIO *****

ENV:
        OUT     (DATOSIO),A
ENVO:
        NOP
        IN      A,(COMSIO)
        AND     4               ;Ve si esta vacio el buffer, si no espera...
        JR      Z,ENVO
        RET
RECIBE:
        IN      A,(COMSIO)
        AND     1
        JR      Z,RECIBE
        IN      A,(DATOSIO)
        CP      H
        JR      NZ,RECIBE
```

```
        RET
TPD:
        DB      00
@LOG:
        .Z80
;$$$$$$$$$$$$$$$$RECUPERA DATO QUE MANDA LA RUTINA PRINCIPAL$$$$$$$$$$$$$$$$

        LD      A,D             ;Recupera el dato y
        LD      (TPP),A         ;lo guarda. El dato nos da el tipo de terminal:
                                ;       1 = 921
                                ;       2 = 922
                                ;       4 = 924
                                ;       5 = 955

; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
; $$$$$                                     $$$$$
; $$$$$   VARIABLES PARA MANEJO DE PANTALLA $$$$$
; $$$$$                                     $$$$$
; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

BDOS    EQU     05H     ;Direccion de las rutinas de Basic disk operation
                        ;systems
ESC     EQU     1BH     ;Escape
CR      EQU     0DH     ;Carriage return

; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
; $$$$$  VARIABLES PARA MANEJO DEL SIO      $$$$$
; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

DATOSIO EQU     20H     ;Puerto de datos canal A
COMSIO  EQU     22H     ;Puerto de comandos canal A

        IN      A,(DATOSIO)     ;Lee cuatro veces
        IN      A,(DATOSIO)     ;Para limpiar el SIO
        IN      A,(DATOSIO)
        IN      A,(DATOSIO)

; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
; $$$$$                                  $$$$$
; $$$$$        PROGRAMA PRINCIPAL        $$$$$
; $$$$$                                  $$$$$
; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
INIO:
        LD      A,1AH
        CALL    SENDA
        LD      HL,TECLA
BQUEO:
        LD      A,(HL)
        INC     HL
```

```
        CP      0
        JR      Z,SGUE1
        CALL    SENDA
        JR      SGUE0
SGUE1:
        LD      A,(TPP)
        CP      5
        JP      Z,L0955
        CP      2
        JP      Z,L0922
        CP      4
        JP      Z,L0924
L0921:
        LD      A,ESC
        CALL    SENDA
        LD      A,'7'   ;Manda que regrese pantalla ESC 7
        CALL    SENDA
        JP      EPEZAR
L0922:
        LD      A,ESC
        CALL    SENDA
        LD      A,'7'   ;Manda que regrese pantalla ESC 7
        CALL    SENDA
        JP      EPEZAR
L0924:
        LD      A,ESC
        CALL    SENDA
        LD      A,'8'   ;Manda que regrese pantalla con ESC 8 ?
        CALL    SENDA
        LD      A,'7'
        CALL    SENDA
        JP      EPEZAR
L0955:
        LD      A,ESC
        CALL    SENDA
        LD      A,'7'   ;Manda que regrese pantalla con ESC 7
        CALL    SENDA
EPEZAR:
        LD      IX,BFER         ;Guardo la direccion en el apuntador IX
TII:
        LD      B,0     ;Se van a admitir datos del SIO hasta que se
        CALL    REIVE   ;encuentra una secuencia IVS, que son los ultimos
        CP      'T'     ;caracteres de la pantalla.
        JR      NZ,TRO
        LD      B,1
        CALL    TRO
        CALL    REIVE
        CP      'V'
        JR      NZ,TRO
```

```
        CALL    REIVE
        CP      'S'
        JR      NZ,TRD
        JR      PCE     ;Si encontro secuencia salta
TRD:
        LD      (IX+0),A        ;Si no ha encontrado secuencia se almacenan
        INC     IX              ;los datos en donde apunta IX
        LD      A,B
        CP      1
        JR      NZ,TII
        LD      B,0
        RET

PCE:
        LD      HL,BFER
        LD      IX,TECLA
PC1:
        LD      A,(IX)
        CP      54H
        JR      Z,OKEY
        LD      D,A
        LD      A,(HL)
        CP      D
        JR      NZ,FATAL
        INC     HL
        INC     IX
        JR      PC1
FATAL:
        LD      A,1             ;No paso la prueba
        JR      PC2
OKEY:
        LD      A,0             ;Paso la prueba
PC2:
        RET                     ;y regresa al programa principal.

; $$$$$  ENVIO DE INFORMACION AL PUERTO DE DATOS DEL SIO  $$$$$

SENDA:
        OUT     (DATOSIO),A
SENDA0:
        NOP
        IN      A,(CONSIO)
        AND     4               ;Ve si esta vacio el buffer, si no espera...
        JR      Z,SENDA0
        RET

;$$$$$  RECEPCION DE INFORMACION POR EL PUERTO DE DATOS DEL SIO  $$$$$

REIVE:
```

```
        IN      A,(CONSIO)
        AND     1
        JR      Z,REIVE         ;Pregunta si hay datos en recepcion,si no lo
                                ;hay espera...
        IN      A,(DATOSIO)     ;de lo contrario lo lee
        LD      C,A             ;salva el dato
        LD      A,01H           ;y pregunta si no encontro bit de paro, si la

        OUT     (CONSIO),A      ;paridad es correcta o si no hubo error de
        IN      A,(CONSIO)      ;sobrecarrera, bits 6, 4 y 5 del registro de
        LD      B,70H           ;lectura de comandos I respectivamente.
        AND     B
        JR      Z,CREC          ;si todo esta correcto salta
        LD      B,0C0H          ;de lo contrario..
        LD      DE,0FFFFH       ;espera a que se acabe de enviar pantalla y..
DAY:
        DEC     E
        JP      NZ,DAY
        DEC     D
        JP      NZ,DAY
        OUT     (CONSIO),A
        JP      INIO            ;..ejecuta de nuevo el programa
CREC:
        LD      A,C             ;salva el dato y regresa
        RET

; ##########################################################
; #####                                              #####
; #####      LOCALIDADES AUXILIARES                  #####
; #####                                              #####
; ##########################################################

BAR:
        DW      0000H
TPP:
        DB      00
BFER:
        DS      0100H   ;buffer de 512 bytes

        RET
TECLA:
        DB      18H,47H,30H,41H,42H
        DB      43H,44H,45H,46H,47H
        DB      30H,31H,32H,33H,34H
        DB      35H,36H,37H,38H,39H
        DB      18H,47H,31H,41H,42H
        DB      43H,44H,45H,46H,47H
        DB      30H,31H,32H,33H,34H
        DB      35H,36H,37H,38H,39H
```

```
DB      1BH,47H,32H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,33H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      0AH,0DH
DB      1BH,47H,34H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,35H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,36H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,37H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      0AH,0DH
DB      1BH,47H,38H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,39H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,3AH,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,3BH,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      0AH,0DH
DB      1BH,47H,3CH,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,3BH,41H,42H
DB      43H,44H,45H,46H,47H
```

```
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,3EH,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,3FH,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      0AH,0BH
DB      1BH,47H,28H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,29H,41H,42H
DB      43H,44H,45H,46H,47H
DB      30H,31H,32H,33H,34H
DB      35H,36H,37H,38H,39H
DB      1BH,47H,3EH,54H,45H
DB      4CH,45H,56H,49H,44H
DB      45H,4FH,20H,54H,56H
DB      53H,20H,39H,32H,31H
DB      1BH,47H,30H,00H
END
```

APENDICE D

PRINCIPALES COMPONENTES UTILIZADOS

Los circuitos integrados utilizados en la tarjeta direccionadora diseñada para la realización de las pruebas son los siguientes:

SN7404	Circuito inversor Y=A.
SN7432	Compuerta OR Y=A+B.
SN74LS154	Decodificador de 4 a 16 lineas.
SN74LS174	Flip-Flop D.
SN74LS244	Seguidor con salida 3 estados.
MC1488	Line Driver de +/-12 V a +5.0 V.
MC1489	Line Receiver de +5.0 V a +/-12 V.

Los primeros 5 circuitos se pueden consultar en un manual de circuitos TTL y los últimos 2 en un manual de circuitos para interfaz.

BIBLIOGRAFIA

- TS802H Installation and User's Guide.
  Televideo.

- CP/M Command Sumary.

- Pascal/MT+ Programmer's Guide.

- Pascal/MT+ Reference Manual.

- Technical Manual for Serial I/O Controller
  MK3884/MK3885

- Using the Z80-SIO in Asynchronous Communications
  Application Note. ZILOG.

- Logic Circuits and Microcomputer Systems.
  Claude A. Wintrowski and Charles H. House.
  Mc Graw-Hill.

- Designing with TTl Integrated Circuits.
  Texas Instruments Incorporated.
  Mc Graw-Hill