UNIVERSIDAD NACIONAL DE INGENIERIA

Facultad de Ingenieria Eléctrica y Electrónica

Diseño y Construcción de Adaptador de Comunicaciones entre Computador e Impresora

T E S I S

Para optar el Título Profesional de

INGENIERO ELECTRONICO

GUILLERMO LEON LO

PROMOCION 1982 - 1

Lima . Perú

1987

DISEÑO Y CONSTRUCCION DE ADAPTADOR DE COMUNICACIONES ENTRE COMPUTADOR E IMPRESORA

A mi madre

A la memoria de mi padre

INDICE

PROLOGO

Este trabajo trata del diseño de un adaptador de comunicaciones entre dos dispositivos que no son completamente compatibles. Dichos dispositivos son: un puerto serial asíncrono de un computador DATA GENERAL MV-8000 y una impresora QUME modelo SPRINT5.

La incompatibilidad radica en que ambos dispositivos controlan el flujo de información de manera distinta.

El puerto del computador trabaja con el protocolo X-ON/X-OFF para el control de tráfico de la data. La impresora QUME no trabaja con este protocolo, aunque es capaz de transmitir ambos caracteres de control si el usuario lo hace a través del teclado.

La impresora tiene un buffer(FIFO) de 221 caracteres y trabaja óptimamente si recibe bloques consecutivos de 110 caracteres terminados con el caracter de control ETX(End of Transmission Block). Por cada bloque recibido la lógica de la impresora devuelve el caracter ACK(Acknowledge), como reconocimiento e invitando la transmisión del siguiente bloque.

La impresora tiene las siguientes velocidades de transmisión: 110, 150, 300, 600 y 1200 BPS. Y como máxima velocidad de impresión 55 caracteres por se

gundo.

Considerando que un caracter está compuesto al menos de 8 bits de data, un bit de START y uno de STOP(no consideramos el bit de paridad), tenemos que cada caracter tiene 10 bits. Relacionando esta cantidad con las velocidades de transmisión antes mencionadas vemos que el buffer nunca se llenaría si se trabaja a 300 bps o menos.

Actualmente dichas impresoras están trabajando a 300 bps, por lo tanto nunca se alcanza la velocidad máxima de impresión.

El adaptador de comunicaciones en mención permitirá la conversión de protocolos de X-ON/X-OFF a ETX/ACK y viceversa.

Se sigue un procedimiento lógico en el diseño del adaptador:

-En el Capítulo I se especifica el problema; se hace una descripción de los procesadores de comunicaciones del computador, de la impresora y de las técnicas de control de flujo.

-Se diseña el sistema en el Capítulo II, considerándose por separado el hardware y el software, así también las rutinas de autoprueba. En cada caso se hace una pequeña introducción, los requerimientos se trasladan en una serie de pasos que permitirán enfocar el problema.

-La implementación y prueba del sistema completo se describe en el Capítulo III, haciéndose hincapié en el comcepto de emulación.

-Por último, se ha tratado de documentar adecuadamente, describiéndose cómo trabaja el sistema completo; y se explica cada parte del programa, - de manera que pueda hacerse fácilmente modificaciones en caso necesario.

Agradezco al Ing. Alberto Briceño Aranda, mi asesor, por su valiosa colaboración y aliento en la ejecución del presente tema.

CAPITULO I

ASPECTOS GENERALES

1.1 Introducción

Aquí se encontrará una descripción de las partes involucradas en este trabajo: los puertos asíncronos del computador, la impresora y los diferentes métodos usados para el control de flujo de la información.

1.2 El computador

1.2.1 Puerto asíncrono del computador DATA GENERAL MV-8000

Los periféricos lentos o de baja velocidad, como terminales interactivos, impresoras, lectoras, etc, se conectan al computador a través de puertos asíncronos de comunicación que están agrupados en 8 ó 16 para formar un controlador de comunicaciones, del cual podemos diferenciar los tres tipos siguientes:

-ATI: ASYNCHRONOUS TERMINAL INTERFACE o interface de terminales asíncronos.

-AMI: ASYNCHRONOUS MODEM INTERFACE o interface de modems asíncronos.

-IAC: INTELLIGENT ASYNCHRONOUS CONTROLLER o -

Controlador asíncrono inteligente.

De los tres el más versátil es el último, por la posibilidad de cambiar las características individuales de cada puerto en plena operación del sistema. No como con los ATI o AMI,que necesitan que el sistema se - reinicialice(en otras palabras se da un - "ABAJO" al sistema y luego un "ARRIBA" con - la nueva versión del sistema operativo que - contiene las nuevas características de los - puertos). Esta es una acción que detiene el trabajo del computador y que por lo tanto no puede ser hecho a menudo.

Cada uno de estos controladores puede manejar 8 ó 16 canales asíncronos, entendiéndose que cada canal corresponde a un terminal o modem.

1.2.2 Características funcionales de los puertos

Las características de cada uno de - dichos canales o puertos pueden ser programadas al generarse el primer sistema operativo, y luego cambiadas a voluntad con cada modificación de este último.

La programación de las características de los puertos se hace a través de cuatro palabras características(CHARACTERISTICS WORDS) y de una palabra de inicialización - (INICIALIZATION WORD). Algunas de las características programables son:

-Tipo de dispositivo conectado a dicho puer-

to: como pantalla, impresora,modem, télex, - etc.

-Número de caracteres por línea: 80 para pantallas, 132 para impresoras.

-Número de líneas por página: 24 en caso de - pantallas CRT.

-Velocidad de comunicación del dispositivo - con el controlador.

-Paridad.

-Longitud del caracter.

-Número de bits de STOP.

-Secuencia de teclas que producen BREAK o interrupción.

Otras características programables se encuentran en el anexo A.

1.3 La impresora

1.3.1 La impresora QUME SPRINT5

Esta es una impresora de calidad, unidireccional, que usa margarita; permite una variedad de formatos de papel, hojas continuas,tiene - control de espaciado o tabs, máxima velocidad de impresión de 55 caracteres por segundo. Posee un autotest que permite verificar la operatividad - de la misma.

1.3.2 Características operativas de la impresora

Cuando la impresora está operando a velocidades que son mayores que la velocidad real de

impresión, el dispositivo transmisor debe enviar - la data en ráfagas de menos de 224 caracteres cada una. Cuando se termina de procesar este bloque otro puede ser enviado. La lógica interna de la - impresora envía una señal al dispositivo transmisor avisándole que puede enviar el referido bloque. El dispositivo transmisor debe apendizar el caracter ASCII "ETX" al final de cada bloque. Cuando todos los caracteres precedentes al ETX son - procesados y la impresora descubre el ETX envía - un "ACK" al dispositivo transmisor.

La eficiencia es mayor si los bloques son divididos en grupos de 110 ó 111 caracteres seguidos de un ETX. La transmisión inicial a la impresora puede contener dos de tales grupos, los siguientes grupos serán enviados sólo si un ACK es recibido desde la printer. De esta manera la impresora puede continuar imprimiendo a la máxima - velocidad mientras que la estación transmisora - reacciona al requerimiento de un nuevo bloque a - través del ACK.

La principal consideración operativa en - un ambiente de comunicaciones, es que ambas estaciones: transmisora y receptora usen el mismo protocolo y convenciones.

Otra característica no mencionada en el - manual de la impresora, es que cuando su buffer está próximo a llenarse, la señal de control DTR se desactiva, hasta que pueda aceptar datos nuevamente, lo cual es indicado al reactivarse DTR.

Esto quiere decir que la impresora QUME es capaz de trabajar también con el método de control de flujo DTR-CTS, que se explica en el acápite 1.4.1.1.

1.3.3 Nota adicional sobre el buffer de la impresora

La impresora almacena los caracteres de entrada en un buffer llamado FIFO y los procesa- basada en la regla "el primero que entra es el - primero que sale". Cuando el FIFO está por llenarse se avisa a la estación transmisora de que esto va a suceder y se supone que se suspenderá- la transmisión de acuerdo al control de flujo.

Si la estación transmisora no detiene - transmisión, el FIFO continúa llenándose hasta el último caracter. Cuando la capacidad del FIFO es excedida, ocurre lo que se conoce como "OVER-FLOW DEL FIFO".

Si ocurre el "OVERFLOW DEL FIFO" la impresora empieza a descartar caracteres de entrada hasta que el FIFO se vacíe hasta cierto valor y recién continuará recibiendo caracteres.

1.3.4 Conector de la impresora

Dicho conector tiene las siguientes señales RS-232C:

1 Tierra del equipo
2 Dato transmitido
3 Dato recibido
4 REQUEST SO SEND (Requerimiento para transmitir)
5 CLEAR TO SEND (Listo para transmitir)

6 DATA SET READY (Modem listo)
7 Tierra de señalización
8 DATA CARRIER DETECT (Portadora detectada)
20 DATA TERMINAL READY (Terminal listo)

De las cuales son salidas : los pines 2, 4 y 20, y entradas: 3, 5 y 8.

1.4 Control de flujo o tráfico

Vamos a explicar este concepto con un ejemplo. Tomemos el caso de la conexión entre un computador y una impresora.

Examinando el manual de la impresora, programamos sus características para adecuarlas a las del computador. La única interrogante pendiente es qué velocidad usar. Parecería natural tomar la máxima, digamos 9600 BPS.

Enviamos un archivo a imprimir. El resultado: después de un par de líneas, hay más caracteres perdidos que impresos.

Lo que sucede es que se le está enviando información a una velocidad mayor de la que puede imprimir; su buffer no es suficiente para equilibrar la diferencia de velocidades, de aquí que muchos caracteres se pierdan.

Lo que necesitamos es un mecanismo o regla que detenga al computador cuando el buffer se llene y que haga que la transmisión se reinicie cuando el buffer se vacíe; este mecanismo o regla es comúnmente denominado control de flujo o tráfico.

Desafortunadamente no existe un método - único, como se muestra a continuación; por lo que dos dispositivos que usen diferentes métodos pueden no comunicarse efectivamente el uno con el otro.

1.4.1 Métodos usados para el control de tráfico

1.4.1.1 Uso de las señales RTS y CTS

-RTS (REQUEST TO SEND): Requerimiento para transmitir. Es la señal que se hace activa cuando el equipo terminal desea transmitir información.

-CTS (CLEAR TO SEND): Señal entregada por el modem indicándole al terminal que puede comenzar a transmitir.

En algunas aplicaciones de línea dedicada se unen el RTS de DTE con el CTS del DCE, de manera que el DTE puede usar RTS para activar la portadora del DCE; suele haber un retardo denominado delay entre RTS y CTS del orden - de los milisegundos para permitir que la portadora del modem se estabilice.

De la explicación dada arriba, podría pensarse que las señales RTS y CTS pudieran usarse para el control de tráfico; aunque frecuentemente es así, no se debería por muchas - razones. El problema es que no se permite que DCE desactive la señal CTS hasta que el DTE baje RTS. RTS y CTS son en realidad líneas de - "HANDSHAKE", ellos indican algo más que el hecho de que el DCE está lista para recibir data.

Las señales RTS y CTS fueron diseñadas para que el DTE controle el establecimiento del enlace a través del DCE. El terminal asume que mantendrá el enlace el tiempo que lo necesite y el DCE no puede arbitrariamente desactivar CTS cada vez que lo desee. Esto significa que CTS no puede ser propiamente usado como un indicador de flujo; el DCE (o el equipo que está simulando el DCE) no puede activar y desactivar CLEAR TO SEND para decirle al DTE que haga una pausa o reinicie la transmisión. En otras palabras dichas señales están para el propósito del control del enlace y no para el control de tráfico.

Supongamos que decidimos usar RTS y CTS de todas maneras (como lo hacen ciertos fabricantes). Si la impresora desactiva CLEAR TO SEND en medio de la transmisión de un caracter, qué sucede?, si el transmisor detiene la transmisión inmediatamente en medio del caracter, éste será dado por perdido. Si el transmisor espera hasta que termine el caracter y luego se detiene, puede que no haya espacio en el buffer para este último caracter. Debido a que esta posibilidad no está incluida en el Standard RS-232C; el éxito de usar las señales RTS y CTS no puede ser predecido sin estudiar cuidadosamentelos manuales de ambos equipos involucrados.

Algunos fabricantes evitan este con-

flicto usando otra señal de control: DTR(DATA TERMINAL READY) o terminal listo. Por ejemplo la impresora desactiva DTR cuando no puede aceptar más data y la activa otra vez cuando el buffer está casi vacío. Sin embargo nada garantiza que el computador reconocerá esta señal o lo interpretará correctamente ya que ninguna de estas líneas fueron diseñadas para el control de tráfico.

El gráfico 1.1 ilustra este método.

1.4.1.2 Uso de caracteres de control DC1 y DC3

Otro método es que la impresora envíe al computador por la línea 3 (RxD) algunos caracteres que haga que la transmisión se detenga y reinicie. El procesador de comunicaciones del computador reconoce estos caracteres especiales y controla su salida de acuerdo al caracter recibido. Los caracteres ASCII, DC1 y DC3 son usados frecuentemente y son llamados también X-ON y X-OFF respectivamente. En los teclados standard, estos caracteres corresponden a CONTROL-Q(reasumir) y CONTROL-S(pausa).

La lógica de la printer transmite CONTROL-S cuando su buffer está aproximadamente al 80% y CONTROL-Q cuando el buffer está a menos del 20%.

El gráfico 1.2 ilustra este método.

1.4.1.3 Uso de los caracteres ENQ/ACK/NACK

Este es propiamente un protocolo de

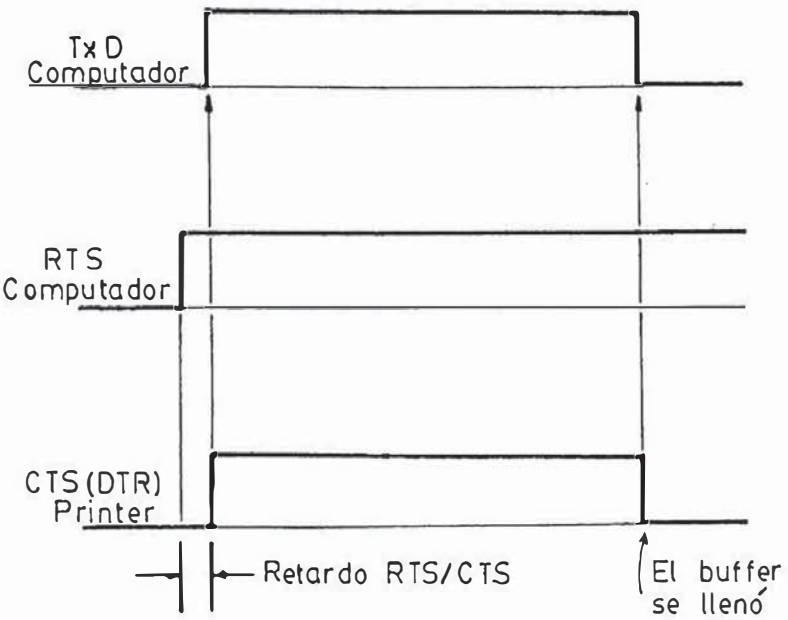


FIG. 1.1 USO DE RTS/CTS(DTR)

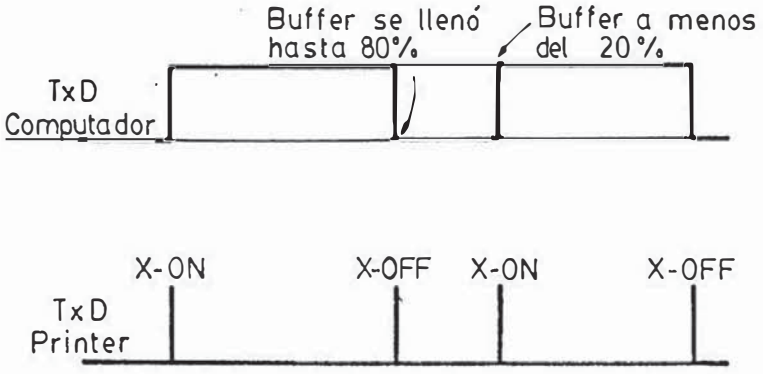


FIG. 1.2 USO DE X-ON/X-OFF

comunicaciones síncronas, donde los dispositivos transfierèn data en bloques que tienen al final unos bytes de chequeo de errores. Estos son usados por el dispositivo receptor para determinar si el bloque recibido está sin errores, si los hay se pedirá retransmisión.

-ENQ(ENQUIRY): Es el caracter que envía el transmisor interrogando al receptor si está listo para recibir o si recibió el bloque correctamente en caso que el receptor no responda con un ACK o un NACK.

-ACK(ACKNOWLEGMENT): reconocimiento o respuesta afirmativa, con este caracter el receptor responde al transmisor indicándole que está dispuesto a recibir información o que el bloque recibido está sin errores.

-NACK(NOT ACKNOWLEGMENT): reconocimiento o respuesta negativa. Con esto el receptor informa al transmisor que no está listo para recibir data o que el bloque recibido contenía errores y que poe lo tanto debe ser retransmitido.

La figura 1.3 ilustra gráficamente lo explicado, donde además tenemos:

1)Dispositivo 2 no está listo para recibir.
2)Dispositivo 2 listo para recibir.
3)Dispositivo 2 acepta bloque 1.
4)Dispositivo 2 rechaza bloque 2 y pide re-

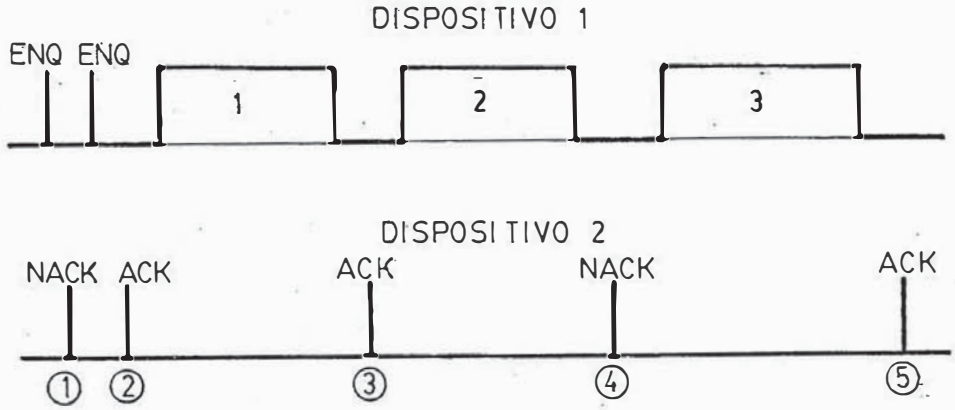


FIG. 1.3 USO DE ENQ/ACK/NACK

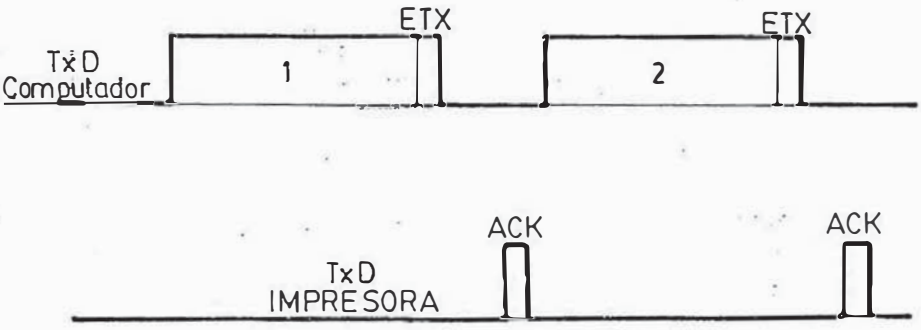


FIG. 1.4 USO DE ETX/ACK

transmisión.

5)Dispositivo 2 acepta bloque 2.

1.4.1.4 Uso de los caracteres ETX/ACK

Este método de control de tráfico es - muy particular:

-ETX: END OF TEXT (Fin de texto) es usado como delimitador de bloques de texto . En otras palabras es el último caracter en cada bloque.

-ACK: ACKNOWLEGMENT (Reconoci- - miento), cada vez que el receptor encuentra un caracter ETX, devuelve un ACK invitando a que - comience la transmisión del siguiente bloque; - sin embargo esto no significa que el bloque fue recibido sin errores como en el caso anterior - (Parágrafo 3).

El gráfico 1.4 ilustra este método.

1.5 Máxima velocidad de impresión

Este es un término que puede llevar a malas interpretaciones, por ejemplo si decimos que una impresora tiene como máxima velocidad de impresión 55 caracteres por segundo y que tenemos un documento que contiene 5500 caracteres podría pensarse que la impresión tomaría 100 segundos.

Esto no es necesariamente cierto por las - siguientes razones:

La velocidad de comunicación debe ser mayor que 55 caracteres por segundo o lo que es equi-

valente a 550 bits por segundo. Por supuesto se da que ambos, la impresora y el computador deben trabajar con el mismo método de control de tráfico para que no haya pérdida de información.

El documento en cuestión debe ser compacto en el sentido de que no haya muchos espacios en - blanco ni salto de páginas.

Vamos a explicar esta última parte:

Lo de los espacios en blanco, viene a que la impresora tiene que mover el carro de impresión - entre palabras sin imprimir nada, lo mismo sucede - con los saltos a la página siguiente en el que el - rodillo mueve el papel y no se imprime nada. Este - tiempo que le lleva al carro para ubicarse en la - próxima posición a imprimir es significativo en la velocidad de impresión. Esto es inevitable ya que - cualquier documento tiene palabras separadas por - espacios, líneas separadas por renglones, etc.

Nota: Los espacios entre palabras son tomados como caracteres, mas no los espacios entre líneas y entre páginas.

Por las razones antes mencionadas la impresión del documento de 5500 caracteres tomará más de 100 segundos.

Vayamos a un segundo ejemplo: digamos ahora que la velocidad de comunicaciones es de 300 BPS o su equivalente de 30 caracteres por segundo, si - la impresora puede imprimir a un máximo de 55 caracteres por segundo pues un documento de 28308 carac

teres, será impreso en 28308/(55x60)= 8.56 minutos.

Esto no es cierto porque la impresora sólo está recibiendo 30 caracteres por segundo.

Entonces el tiempo correcto sería:
28308/(30x60) ó sea 15.7 minutos.

Pues bien, mandamos a imprimir en dos oportunidades el mencionado archivo de 28308 caracteres que era un archivo tipo texto, con una media de - 40 líneas por página y 40 caracteres por línea.

Condiciones: en la primera oportunidad se hizo a las 14.57 con el computador cargado con 78 - procesos; resultado: la impresión tomó 21.04 minutos.

En la segunda oportunidad: a las 08.24, número de procesos corriéndose en el computador 48, el mismo archivo tomó 21.06 minutos.

Así tenemos una velocidad media de impresión de 22.4 caracteres por segundo.

El porqué de esta diferencia entre el tiempo estimado de 15.7 minutos y 21 minutos, ya fue explicado anteriormente.

Como tercera prueba se creó un archivo del siguiente tipo:

AAAAAA.....AAAAAA
BBBBBB.....BBBBBB
CCCCCC.....CCCCCC
.....
.....

Con 132 caracteres iguales en cada fila y

con un total de 12502 caracteres.

Aquí no tenemos espacios entre caracteres ni entre líneas, tampoco hay salto de página, el único tiempo en que no se imprime es cuando se hace retorno de carro, del fin de una línea al inicio de la otra.

Tiempo estimado: 12502/(30x60)= 6.9 minutos.

Tiempo real: 7.06 minutos, a las 15.50 horas con 75 procesos.

En esta oprtunidad sí se cumplió lo estimado porque se evitó todos los incovenientes antes mencionados.

Como última prueba tenemos un archivo del siguiente tipo:

20 filas de 1234567890123.....89012345678901234

Con 60 caracteres por fila, haciendo un total de 1220 caracteres(hay un caracter delimitador por cada línea).

Tiempo estimado: 1220/30= 46.9 segundos.

Tiempo real: 47.2 segundos, a las 8.00 AM con 48 procesos.

Conclusiones: Con archivos normales, que contienen espacios entre caracteres, líneas y páginas, la velocidad media de impresión es de 22.4 caracteres por segundo, para una velocidad de comunicación de 300 BPS. Se alcanza una velocidad media de 30 caracteres por segundo sólo cuando el archivo tiene la menor cantidad de blancos, como en los "archivos patrones" de la segunda y tercera prueba.

CAPITULO II

DISEÑO

Aquí se contempla el diseño del hardware y software del adaptador.

2.1. Diseño del hardware

2.1.1 Introducción

El adaptador de comunicaciones en mención necesita de un ente inteligente que gobierne el flujo de información entre dos puertos seriales.

Este dispositivo inteligente es un microprocesador que se comunica con cada uno de los puertos seriales.

Es necesario usar periféricos de entrada/salida seriales comúnmente llamados USART.

El programa que gobierne al adaptador estará grabada en una memoria no volátil ROM.

Para efectos de almacenaje temporal de información se dispone de una cantidad de memoria volátil RAM.

Como decodificador de I/O tenemos un decoder TTL de BCD a decimal.

Ya que casi toda la circuitería del

conversor trabaja con niveles TTL y los puertos externos de comunicaciones están bajo el standard RS-232C de la EIA, hacemos uso de conversores de TTL a RS-232C y viceversa.

Están presentes también en el diseño algunas puertas TTL para invertir niveles y agrupar señales de control lógicas.

Con un generador dual de baudios obtenemos las velocidades estables de transmisión para cada USART.

2.1.2 Requerimientos

El adaptador respecto al port del computador deberá ser capaz de comunicarse a cualquier velocidad standard de transmisión.

Trabajará con 7 bits de datos, paridad SPACE en transmisión, chequeará errores de overflow y framing en recepción.

Para mayor eficiencia, la velocidad del port asignado al computador, deberá ser mayor que 600 bps, ya que así la impresora trabajará a su máxima velocidad de impresión que es de 55 caracteres por segundo.

Respecto al port asignado a la printer, su velocidad será de 1200 bps, pudiendo ser 600 bps pero no inferior, por la razón arriba mencionada. Este port tendrá paridad SPACE, 7 bits de datos, no se chequea errores en recepción ya que el equipo estará primordialmente al lado de la printer y es improbable que ocurran

errores de comunicación, no como en el caso del - port hacia el computador que puede estar conectado al computador a través de modems y/o multiplexores.

Sobre la memoria: debe ser de un volumen tal que permita un tráfico fluido de la información, pero no tan grande como para que el equipo se convierta en un spooler, ya que ésta no es la finalidad del equipo.

Del CPU o microprocesador: El clock del CPU deberá ser lo suficientemente rápido de manera que permita que el equipo trabaje en "tiempo - real", es decir que pueda el CPU servir a ambos - USARTS sin que haya pérdida de data por overflow.

Del generador de baudios: Deberá proporcionar las velocidades standard de transmisión, - con buena estabilidad, gobernada preferentemente por un cristal.

Dichas velocidades deberán ser fácilmente cambiadas.

El adaptador debe ser transparente en - todo sentido:

-El computador debe creer que está - conectado a un dispositivo con el protocolo - X-ON/X-OFF de control de flujo.

-La printer debe pensar que el computador entiende su protocolo particular ETX/ACK.

-El usuario deberá manipular mínimamente el adaptador, a lo más tendrá que resetear-

lo para volverlo a sus condiciones iniciales, no existiendo otro control, salvo el de reseteo de errores de comunicaciones.

2.1.3 Concepción y diagrama de bloques

Uno de los USARTS es dedicado a servir al port asíncrono del computador, para esto se inicializa con parámetros compatibles con las características de dicho port, vale decir: número de bits de data, número de bits de stop, paridad, velocidad, etc.

El otro USART se dedica al servicio de la printer, adecuándose también a sus características.

El microprocesador actúa como un controlador regulando el flujo de información bidireccional entre el computador y la impresora.

Dicho sea de paso, el volumen de tráfico de información es diferente en uno y otro sentido, siendo mayor del computador a la impresora. En sentido contrario la data es mínima. La razón de esto se explica a continuación.

El usuario tiene pocas opciones disponibles en el teclado, ya que el port asignado está definido para impresora; por ejemplo, puede detener la impresión con CTRL-S, reasumirla con CRTL-Q o cancelarla con CRTL-O.

Aún en el caso de que port estuviese "habilitado" para trabajos interactivos, que se hacen normalmente con terminales CRT, la veloci-

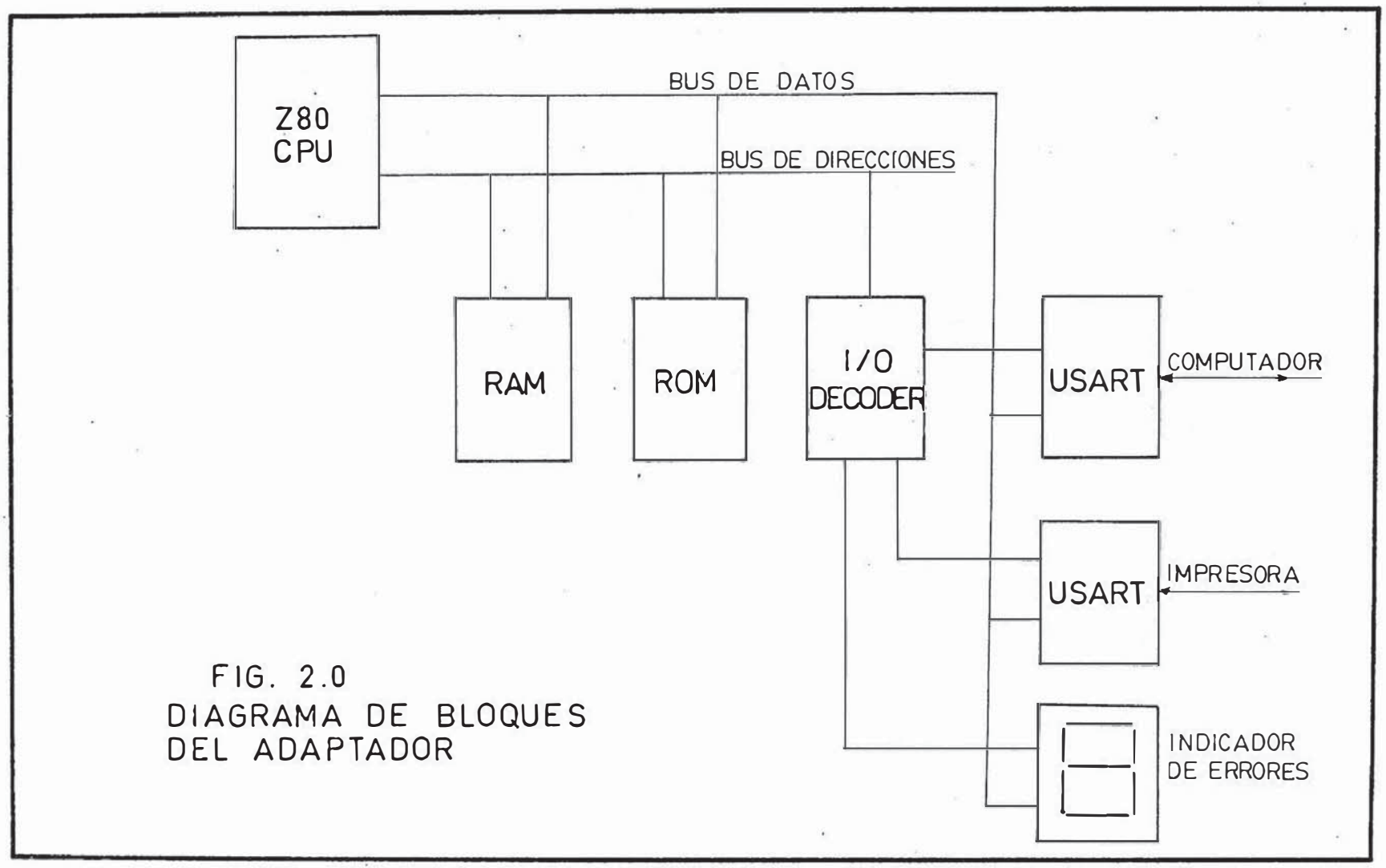


FIG. 2.0
DIAGRAMA DE BLOQUES
DEL ADAPTADOR

dad con la que tipea el usuario es menor que la - de transmisión del computador hacia la printer.

Por estas dos razones sólo se asigna un Kbyte de memoria de almacenaje temporal para la - data de computador a printer.

Este Kbyte de memoria equivale a aproximadamente un cuarto de página, considerándose - una página igual a 60 líneas por página, y 80 caracteres por línea, lo que hace un total de 4800 caracteres por página.

La razón por la que no escogemos más - memoria, es que este dispositivo no tiene las funciones de un "printer spooler", sino la de un adaptador de comunicaciones. Otra razón adicional, es que teniendo el usuario la capacidad de cancelar una impresión ya sea desde el teclado de la - impresora(con el comando CTRL-O) o a través de - la pantalla, es conveniente que el adaptador retenga la menor cantidad de información, así al - llegar la orden de cancelación, lo que va a ser - impreso será mínimo.

La figura 2.0 es un diagrama de bloques simplificado del adaptador.

2.1.4 Diseño final

El microprocesador:

Entre la variedad de microprocesadores de 8 bits disponibles en el mercado, como el 8080, 6800, 6502 y el Z80, se escogió este último por las siguientes razones:

-Bajo costo.

-Amplia información y bibliografía.

-El hecho de contar con un microcomputador SPECTRUM de SINCLAIR que justamente tiene como procesador un Z80A, y en el cual se tiene acceso a todos los buses de data, direcciones y de control. Tenemos también la facilidad de trabajar en lenguaje ASSEMBLY y usar un programa MONITOR para hacer el "DEBUGGING" de las rutinas.

La memoria RAM:

El requerimiento de memoria volátil es de una FIFO de un Kbyte, ya que éstos no son muy comerciales, optamos por escoger una RAM estática de un Kbyte representado por dos circuitos integrados 2114, cada uno de 1K por 4. La conversión de RAM a FIFO se hará mediante software. El hecho de que la RAM sea estática representa una facilidad en su manejo.

La memoria ROM:

La exigencia de 1Kbyte en memoria ROM es cubierta con el EPROM 2708 que tiene como único inconveniente el hecho de necesitar tres fuente: +5V, -5V y -12V.

Los USARTS:

Escogemos el 8251A de INTEL por ser uno de los más populares y de bajo costo en el mercado.

Entre sus características tenemos:

Número de bits por caracter programable entre 5, 6, 7 y 8; velocidad de transmisión en modo asíncrono hasta 64 Kbps.

Detección de errores de comunicación, velocidades del clock de 1, 16 y 64 veces la velocidad de transmisión. Número de bits de stop - programables. También puede trabajar en modo síncrono. Buffers de TX y RX de dos niveles.

La inicialización del 8251 requiere - solamente de 2 bytes.

Entre otros de los USARTS considerados está el Z80-SIO SERIAL INPUT/OUTPUT, que - tiene la ventaja de ser dual (dos USARTS en uno), tiene buffers de tres niveles (que sería - óptimo si las velocidades de transmisión fuesen altas). Otras de sus ventajas es que posee una - potente arquitectura de interrupción que le permite participar en cadenas de interrupción - "DAISY CHAIN" con prioridad.

La conexión al Z80-CPU es directa.

La principal desventaja del SIO es - que necesita de 6 palabras de inicialización, esta relativa complejidad se debe a que el SIO tiene varios modos de trabajo.

También tenemos el AMD9551 que es una versión mejorada del 8251. y el 2651 con el - AY-3-1015D de los cuales la información disponible es mínima.

Otra serie de USARTS está conformada por la familia COM2502/COM2017 de la STANDARD -

MICROSYSTEM CORPORATION. Tiene la desventaja de necesitar 2 fuentes -5V y -12V.

El "STATUS WORD" está en un registro accesible separadamente, es decir tiene pines - que indican error de paridad, framing o de over flow, de manera que requiere de lógica adicional para leer el "STATUS WORD".

La programación de la paridad, número de bits de data, chequeo de paridad, etc, se hace por hardware, que es una desventaja frente al 8251.

Es un IC de 40 pines comparados con 28 del 8251 que incluye en software muchas de - las funciones y controles del COM2502/COM2017.

El generador de baudios:

sobre el generador de - baudios tenemos para elegir:

-COM8046
-COM8116
-COM5016

Todos de características similares.

Optamos por el 8116.

El decodificador de direcciones para memoria:

Usamos un 7442, decodificador de BCD a decimal, para seleccionar la ROM y la RAM, dejando como opción de expansión de la RAM hasta 7Kbytes.

El decodificador de direcciones para I/O

Aunque solamente tenemos tres dispo-

sitivos de entrada escogemos el IC TTL 7442.

Data bus driver:

Aunque ni el bus de datos ni el bus de direcciones están excesivamente cargados para exceder la capacidad de corriente del Z80-CPU, es práctica usual usar driver para ellos.

En nuestro caso sólo usamos un buffer para el bus de datos: 74LS245.

2.1.5 Diagramas circuitales

Las figuras 2.1, 2.2 y 2.3 muestran - los circuitos objeto de este diseño.

La figura 2.1 corresponde al CPU, su sistema de clock y reset, el bus de datos y el decodificador de memoria y finalmente en la figura 2.3 se tienen los dispositivos de entrada/salida; la fuente de alimentación y su circuito de monitorado.

2.1.6 Teoría de operación del hardware

Como se explicó, el procesador Z80A (IC2) controla el funcionamiento de este adaptador de comunicaciones.

Se usa un clock de 3.579545 MHZ. El - sistema de reset combina el modo manual a través de un pushbutton; y el automático, por medio de la resistencia de 10 K, condensador de 47 uF, y el diodo 1N914, que funciona cada vez que se enciende el equipo o ante cortes de energía de duración mayores que algunos milisegundos.

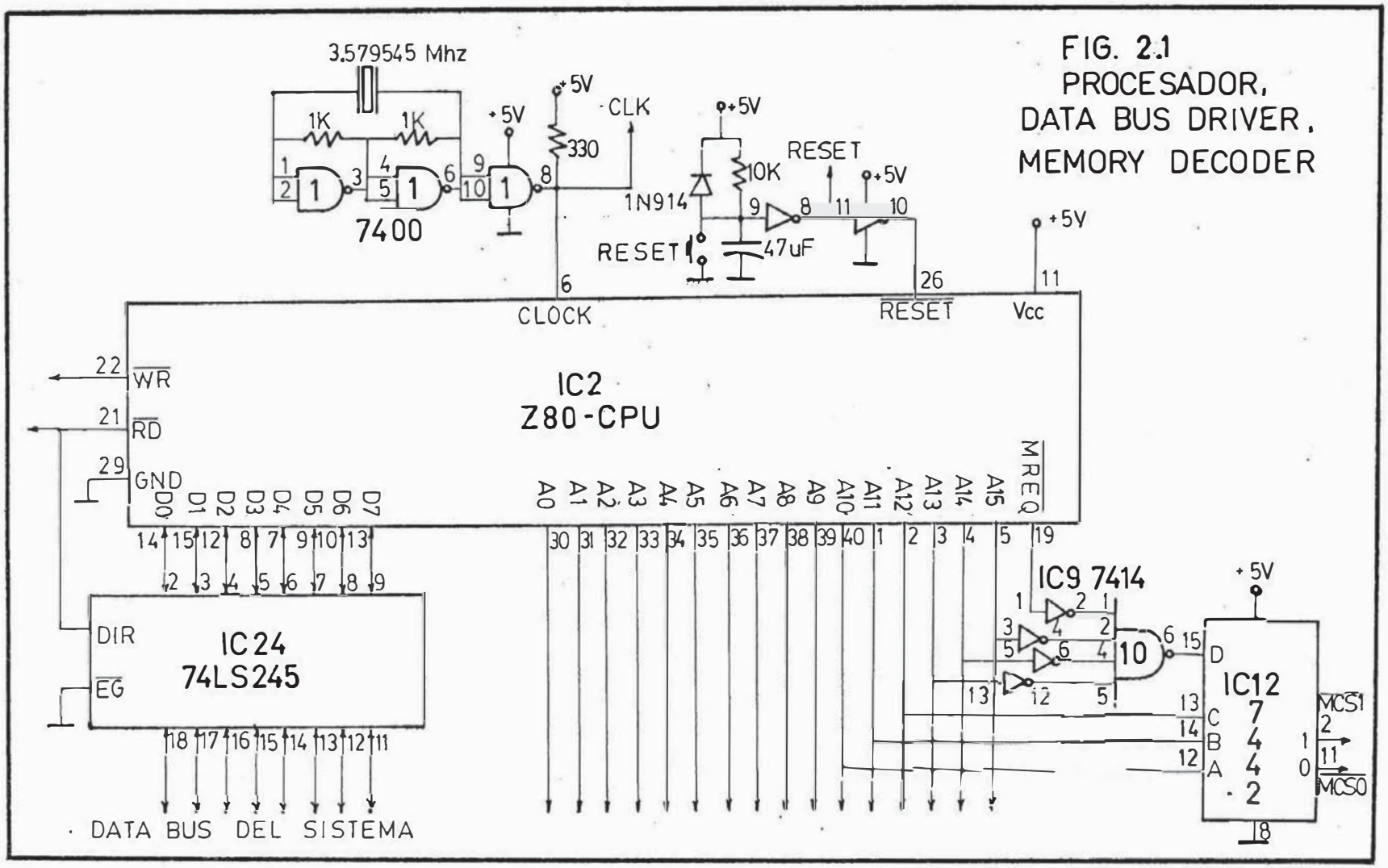


FIG. 2.1
PROCESADOR,
DATA BUS DRIVER,
MEMORY DECODER

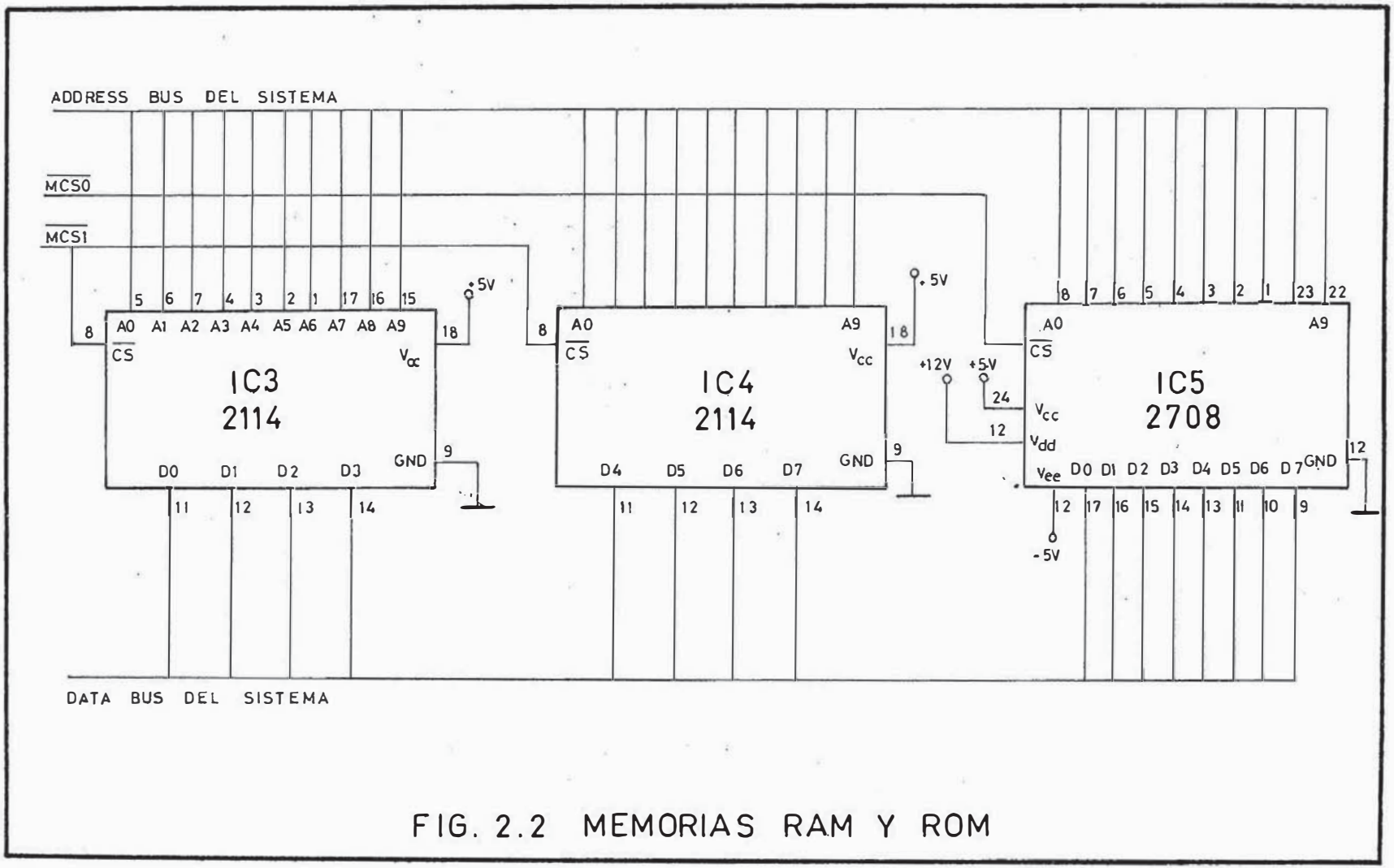


FIG. 2.2 MEMORIAS RAM Y ROM

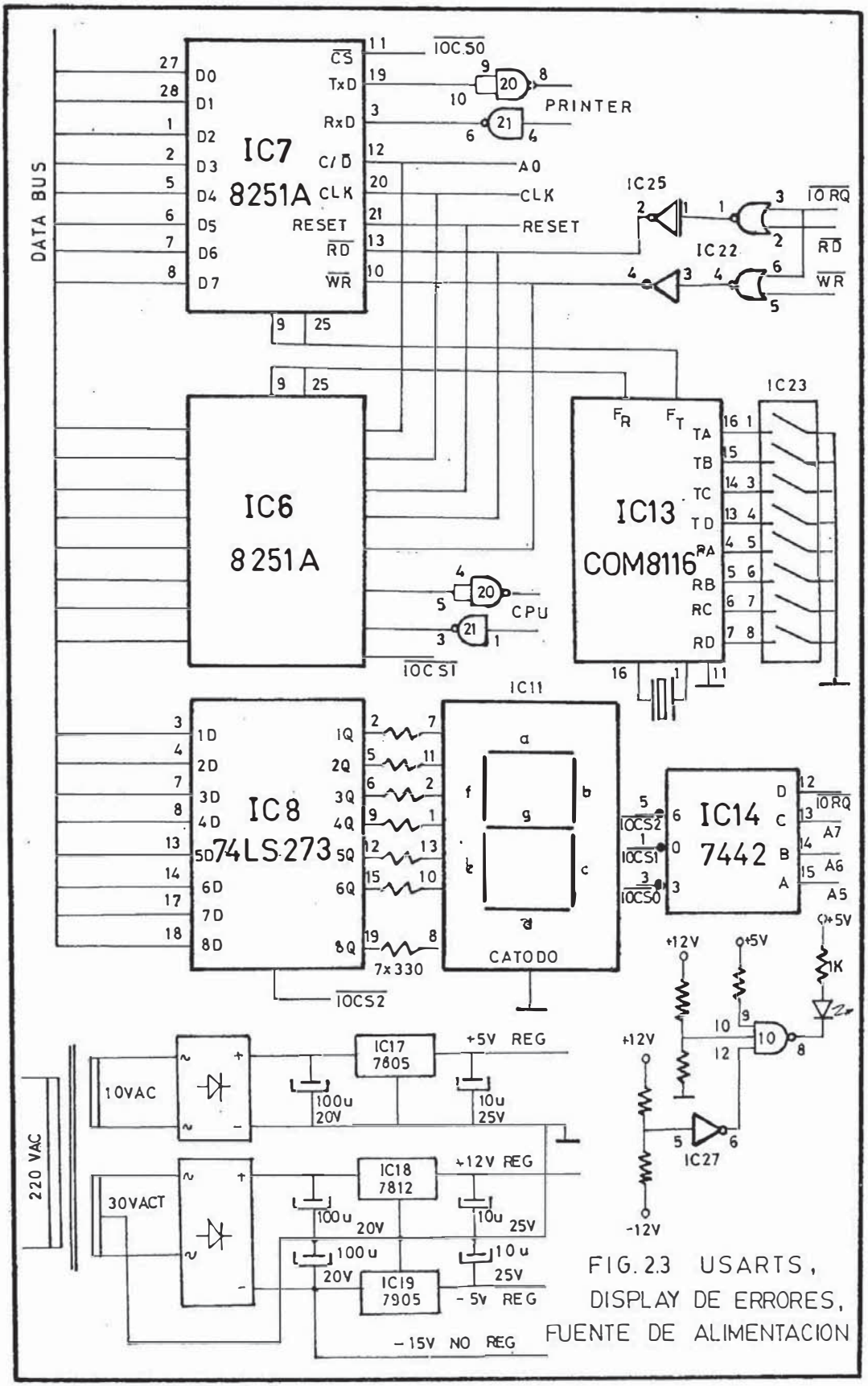


FIG. 2.3 USARTS, DISPLAY DE ERRORES, FUENTE DE ALIMENTACION

Se usa un data bus driver bidireccional 74LS245(IC24) controlada por la señal $\overline{RD}$. Para el bus de direcciones no se usa buffer.

Un 7442(IC12) es usado como decodificador de direcciones para las memorias, $\overline{MCS0}$ es la señal de habilitación para la ROM y $\overline{MCS1}$ para la RAM, se deja para futura expansión $\overline{MCS2}$ a $\overline{MCS7}$, que permitirá incrementar la memoria a 7 Kbytes.

Los dos 8251A (IC6 e IC7) son los USARTS trabajando en modo asíncrono, dedicado uno al servicio del computador y el otro a la impresora.

Los trasladores de nivel de RS-232C a TTL y viceversa son el MC1488(IC20) y MC1489(IC21) respectivamente.

Las señales de $\overline{RD}$ y $\overline{WR}$ para los USARTS se generan a partir de $\overline{IORQ}$, $\overline{RD}$ y $\overline{WR}$ del CPU por medio del 7402(IC22).

El 74LS273(IC8) y el display de siete segmentos(IC11) conforman el visualizador de errores del adaptador.

La relación entre la palabra que se envía al display y cada uno de los segmentos es como sigue:

BUS DE DATOS	D0	D1	D2	D3	D4	D5	D6	D7
SEGMENTOS	e	f	g	a	b	c	p	d

Hay otro 7442(IC14) que trabaja como decodificador para los dispositivos en entrada y salida.

Se usan las direcciones A7, A6 y A5 por lo que cada dispositivo no tiene una dirección única. Por ejemplo, el primer USART se identificaría como:

A7	A6	A5	A4	A3	A2	A1	A0
A7	A6	A5	X	X	X	X	1/0

Donde A7, A6 y A5 son los valores que escogemos, X significa que puede ser cualquier valor y A0 es (L/O) dependiendo si la transferencia de información es de datos o de palabras de control.

Así hemos escogido para el primer USART las siguientes direcciones:

0E para transferencia de datos.
0F para transferencia de palabras de control.

Para el segundo USART:

4E para datos.
4F para palabras de control.

Para el display de errores se toma la dirección 3E.

El IC16(COM116) y el IC23(DIP SWITCH) corresponden al generador de BAUDIOS, cuyas ve-

locidades de transmisión y recepción pueden ser seleccionados separadamente a través de los - switches, permitiendo así que cada USART trabaje a diferente velocidad.

La relación entre los switches y - las velocidades obtenibles es:

Pines del COM8116:	16	15	14	13	4	5	6	7
Función	TA	TB	TC	TD	RA	RB	RC	RD
Switch	1	2	3	4	5	6	7	8
1200 bps	ON	ON	ON	OFF	ON	ON	ON	OFF
2400 bps					OFF	OFF	ON	OFF
4800 bps					OFF	ON	OFF	OFF
9600 bps					ON	OFF	OFF	OFF

Donde ON corresponde a switch cerrado o pin puesto a tierra y OFF, switch abierto o pin no conectado(en "1" lógico). Para otras velocidades consultar las especificaciones del - COM8116.

IC3 e IC4(2 2114) conforman la RAM - del sistema, mientras que IC5(2708) es la ROM - del adaptador.

La fuente de +5 VDC se obtiene del - secundario de 10 VAC, rectificada y regulada con el IC12(LM7805).

La fuente de +12 VDC es regulada para la EPROM e IC20 y la de -5 VDC regulada para la misma EPROM. Mientras que la fuente de - -15 VDC no necesita ser regulada.

El clock para los USARTS 8251A es el

mismo del Z80 cumpliéndose así que debe ser al - menos treinta veces más rápido que la frecuencia del clock que se aplica a TxC o RxC, para propósitos de sincronización internas.

La señal de reset de los 8251A es en principio la misma que se aplica al Z80, ya que los USARTS necesitan ser reseteados antes de recibir las instrucciones de modo y comando que - programarán sus características de trabajo.

2.2 Diseño del software

.2.2.1 Introducción

Como mencionamos en el acápite 1.3.2 la eficiencia de la impresora puede incrementarse ya sea empaquetando la información en bloques de 110 caracteres o usando la señal DTR de la - impresora para controlar el tráfico. Optamos por la primera alternativa.

Tenemos básicamente cuatro rutinas:

-La de inicialización.
-Recepción de data del computador: RECEPC.
-Transmisión de data a la printer: TRANSM.
-Comparación de punteros: CONTl.

La rutina de inicialización se encarga de dar los parámetros iniciales a los USARTS 8251A, de la definición de variables y constantes, y de la definición de los límites del buffer etc.

La recepción de caracteres del computador, es transparente en el sentido de que no se analiza cada caracter, chequeándose solamente si hubieron errores de comunicación. También se encarga del pequeño flujo de datos de la impresora al computador, en este caso el tratamiento es "no transparente", porque se analiza todo caracter que la impresora envía al computador para ver si se trata de un caracter ce control con CTRL-S, CTRL-Q, ACK, etc.

La rutina de transmisión se encarga de la data que fluye del adaptador hacia la impresora.

Aunque los punteros de escritura y lectura del buffer se actualizan en las respectivas rutinas de recepción y transmisión; la comparación de ellos para ver que no se traslapen, se hace en la rutina CONT1. El buffer es una RAM, que en esta rutina simula ser una FIFO; esto se logra haciendo que los punteros vuelvan al principio del buffer después de llegar al final.

Al llenarse el buffer hasta 1000 caracteres se detiene la recepción, hasta que el buffer se vacíe a 110 caracteres y luego se reinicia la recepción, esto se hace con la finalidad de no perder caracteres y tener un tráfico fluido. Los valores de 1000 y 110 son escogidos por criterio y podrán ser cambiados de acuerdo a resultados experimentales.

2.2.2 Requerimientos de las rutinas

El requerimiento fundamental es que la atención a los USARTS por parte del microprocesador sea en "tiempo real", es decir que las rutinas sean lo suficientemente rápidas para que no se pierda información por overflow.

El programa de inicialización seteará todas las variables y constantes de manera que al iniciar el trabajo después del encendido o power up, no ocurran circunstancias que deriven en situaciones fuera de control, denominadas comúnmente "colgadas del sistema" (o system hang up).

Los programas de recepción y transmisión deberán manejar todas las situaciones de trabajo, tanto normales como anormales; en caso de situaciones anormales como errores de comunicaciones, se indicará en el display de errores y dejará que el sistema siga funcionando.

El programa de comparación de punteros es uno de los más importantes, porque debe manejar el buffer que es una memoria de acceso aleatorio y hacer que simule una memoria FIFO (first in first out).

2.2.3 Concepción y diagramas de flujo

2.2.3.1 La rutina de inicialización

En esta parte inicial de los programas deberán definirse todas las constantes que necesitará el programa para su funciona-

miento, como por ejemplo los límites del buffer, los diferentes caracteres de control utilizados, etc.

La figura 2.4 es el diagrama de flujo simplificado de la rutina de inicialización.

2.2.3.2 Rutina de recepción

Esta rutina sirve al USART conectado al puerto del computador. Como función básica tenemos que debe averiguar si se recibió algún caracter, examinando el "STATUS WORD" del USART; chequear si existieron errores de comunicaciones. El caracter recibido debe ser almacenado en memoria y los punteros deben ser actualizados.

También se encarga de la transmisión de caracteres hacia el computador previo análisis para ver si son caracteres de control.

La figura 2.5 bosqueja el posible diagrama de flujo.

2.2.3.3 La rutina de transmisión

Entre las funciones básicas de esta rutina, tenemos:

-Transmisión de datos hacia la impresora, en bloques delimitados por el caracter de control "ETX".

-Adecuarse a los varios casos que pueden ocurrir por la recepción de caracteres de control, el haberse transmitido un bloque, esperar un ACK, el haber transmitido la impre

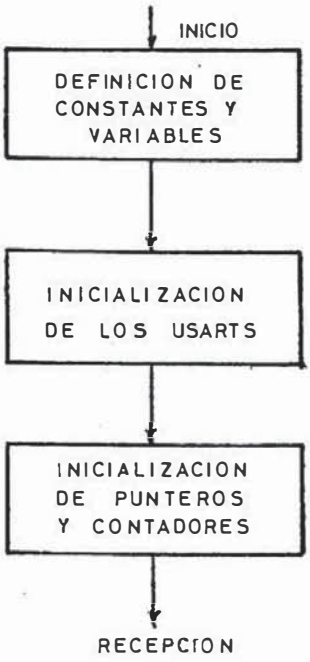


FIG. 2.4 INICIALIZACION

FIG. 2.5 RECEPCION

sora un CTRL-S, etc.

La figura 2.6 muestra un diagrama de flujo simplificado.

2.2.3.4 La rutina de comparación de punteros

Los punteros HL y DE indican las posiciones de memoria que ocupará el caracter que llega desde el computador y la que saldrá hacia la printer respectivamente.

Ambos punteros tienen como límites - 0400 y 07EF (1024 y 2031 en decimal), correspondientes a 1008 bytes de memoria.

Se tiene que chequear constantemente que los punteros no excedan los límites prescritos y que tampoco exista overflow del buffer.

Otra función importante es la de permitir un tráfico fluido de datos hacia la - printer haciendo que el computador transmita por ráfagas o bursts. Esto se obtiene usando extensivamente los caracteres de control - CTRL-S y CTRL-Q.

La figura 2.7 muestra un diagrama de flujo simplificado de esta rutina.

2.2.4 Diseño final de las rutinas

2.2.4.1 Rutina de inicialización

Bloque de inicialización

Se definen los códigos de los caracte

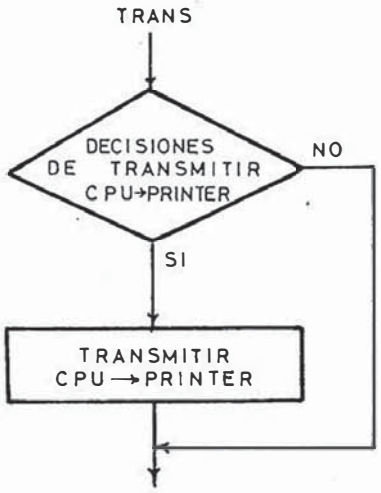


FIG. 2.6 TRANSMISION

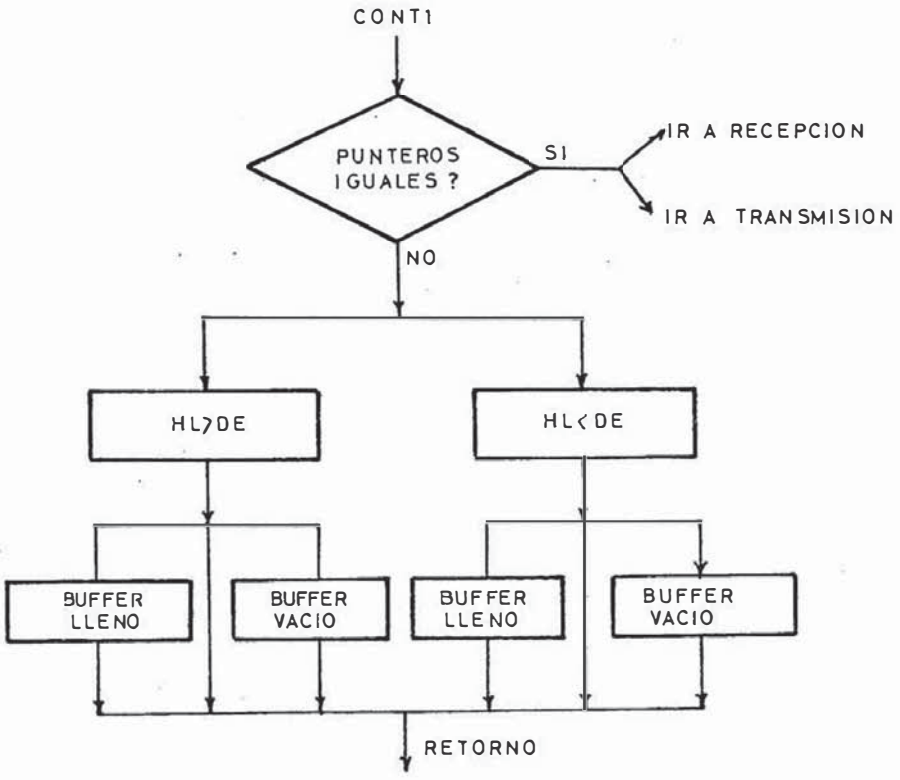


FIG. 2.7 COMPARACION DE PUNTEROS

res de control usados por la impresora y el - computador, también las constantes y variables del sistema. Asimismo se inicializan los - USARTS.

Caracteres de control

ETX: End of transmission block, HEX 83, es el último caracter de cada bloque de 110 caracteres.

CTRL-S: CONTROL S, o caracter de control - DC3, comúnmente denominado X-OFF. Al ser recibido este caracter por el ente transmisor, éste detiene la transmisión, HEX 93.

CTRL-Q: CONTROL Q, o caracter de control - DC1, comúnmente denominado X-ON, al - ser recibido se reinicia la transmisión. Es el opuesto a CTRL-S. HEX 91.

ACK: ACKNOWLEGMENT, HEX 86. Cada vez que - la printer recibe un bloque con un - ETX, devuelve el caracter ACK como - reconocimiento y aceptación, dando - por indicado que puede enviarse el - siguiente bloque.

Constantes del sistema

MAXDE: Valor máximo del registro par DE - (puntero de lectura del buffer); o también límite superior del buffer más uno=.07F0

Variables del sistema

HLTEMP: Dirección donde se almacena temporalmente el valor del par - HL.Valor = 07F2.

DETEMP: Dirección donde se almacena temporalmente el valor del registro - DE.Valor = 07F0.

TEMP: Dirección donde se guarda temporalmente el registro A(en este caso almacena el caracter transmitido - por la printer). Ver en la rutina TRANSM.Valor = 07F5.

IX: Guarda la dirección de la posición de memoria que se usa como FLAG.Valor = (07F4). El FLAG está definido - como:

- -Bit 0 = 0 cuando HL > DE y = 1 - cuando HL < DE.
- -Bit 1 = 0 cuando hay que transmitir dos bloques consecutivos.
- -Bit 2 = 1 cuando hay que esperar ACK desde la printer.
- -Bit 3 = 1 se recibió un CTRL-S - desde la printer.
- -Bit 4 = 1 adaptador envió CTRL-S.
- -Bit 5 = 1 hay que transmitir ETX hacia la printer.

SP: Puntero del STACK, valor inicial 07FF

```
IM 2
LD A,#02
LD I,A
EI
```

Se establece el modo de interrupción 2, cargamos el vector de interrupción con el valor hexadecimal 02 y luego habilitamos las interrupciones.

Inicialización de los USARTS 8251:

Se envían consecutivamente a los USART dos palabras, la primera de modo de trabajo y la segunda de comando.

La instrucción de modo tiene la siguiente estructura:

B7 y B6 definen el número de bits de STOP, 01 = 1 Bit, 10 = 1 y 1/2 Bits; 11 = 2 Bits.

B5 Even parity generation/check 1= Even 0 = Odd.

B4 Parity anable ; 1 = Enable,0=Disable

B3 y B2 Longitud del caracter; 00 = 5 Bits, 01 = 6 Bits, 10 = 7 Bits, 11 = 8 Bits.

B1 y B0 Factor de multiplicación de la velocidad 0C = Modo síncrono, 01 = (1X); 10 = (16X),11=(64X).

De la cual elegimos según requerimientos del sistema:

01001110 = HEX 4E

Paridad odd (SPACE)

7 Bits de dato.

1 Bit de stop.

De otro lado la palabra de comando es como sigue:

B7 = 1 entrar al modo de búsqueda de la palabra de sincronismo.
B6 = 1 resetear al 8251 y la próxima palabra que reciba será una - instrucción de modo.
B5 = 1 Activar RTS.
B4 = 1 resetear errores de comunicaciones.
B3 = 1 enviar el caracter BREAK.
B2 = 1 habilitar recepción.
B1 = 1 activar DTR.
B0 = 1 habilitar transmisión.

Asimismo tenemos: 00010101 = HEX 15

Con lo que : -Reseteamos errores.
-Habilitamos transmisión y - recepción.

Rutina START

Se inicializan punteros, contadores y - flags.

-Con LD HL,#0400
LD DE,#0400; inicializamos los punteros de lectura y - escritura con la posición inicial del buffer.

-Con LD B,0
LD C,0 , inicializamos B, que es - el contador de caracteres

Enviados a la printer y - toma valores de 0 a 110. El registro C es también inicializado a cero, cuando C = 2 se sabe que se - enviaron dos bloques consecutivos sin esperar ACK, a partir de este momento por cada bloque se recibe un ACK.
El registro C toma valores entre 0 y 3.

```
LD (IX), 0 ; la posición (IX) es -
           la del flag del sistema.
           Aquí se resetea el flag.
```

2.2.4.2 Rutina RECEPC

```
1)RECEPC    IN A,(#0F)
            BIT 1,A
            JR Z,CONTA
```

Chequeamos el STATUS WORD para averiguar si se recibió algún caracter, si no hubiese ningún - caracter, saltar a CONTA.

```
2)BIT 4,A
  CALL NZ,ERRORO
  BIT 5,A
  CALL NZ,ERRORF
```

Si se recibió algún caracter chequeamos errores de overflow y framing. Si existen errores saltar a la respectiva rutina de errores y -

regresar.

```
3)IN A,(#0E)
  LD (HL),A
```

Guardamos el caracter recibido en la posición de memoria indicada por HL.

```
4)INC HL
  LD (DETEMP),DE
  LD (HLTEMP),HL
  LD DE,MAXDE
```

Incrementamos el puntero HL, lo guardamos - junto con DE y cargamos DE con el límite superior del buffer.

```
5)XOR A
  SBC HL,DE
  JP M,MIN
```

Si HL es menor que DE saltar a MIN

```
6)LD HL,#400
  SET 0,(IX)
  JR MON
```

Resetear HL a valor inicial, setear flag 0 y saltar a MON.

```
7)MIN LD HL,(HLTEMP)
  MON LD DE,(DETEMP)
```

Restaurar HL y DE.

La parte que viene a continuación maneja la - data a ser transmitida del adaptador hacia el computador. La data de la impresora hacia el -

computador es analizada para ver si es uno de los caracteres de control ACK, CTRL-Q, CTRL-S. Si es uno de ellos se toma las acciones pertinentes. Si no lo es, simplemente lo transmite.

```
8)CONTA     IN A,(#4F)
            BIT 1,A
            JR Z,ANAL
```

Existe caracter desde la printer?, si no existe saltar a ANAL.

```
9)IN A,(#4E)
  LD (TEMP),A
  CP ACK
  JR NZ,Q
  RES 2,(IX)
  JR ANAL
```

Se almacena el caracter recibido provisionalmente en (TEMP).
Se compara con ACK, si no es igual saltar a - Q. Si es igual resetear flag 2 indicando que se puede reiniciar la transmisión hacia la - impresora. Luego saltar a ANAL.

```
10)Q        LD A,(TEMP)
            CP CTRLQ
            JR NZ,S
            LD A,CTRLQ
            CALL OQSTX
            RES 4,(IX)
            RES 3,(IX)
            JR ANAL
```

Restauramos A luego de la prueba anterior, y lo comparamos con CTRL-Q; si no es igual saltar a S. Si es igual enviar CTRL-Q al computador, resetear flags 3 y 4, y saltar a ANAL.

```
11)S        LD A,(TEMP)
            CP CTRLS
            JR NZ,ANY
            LD A,CTRLS
            CALL OQSTX
            SET 4,(IX)
            SET 3,(IX)
            JR ANAL
```

Restauramos nuevamente A y lo comparamos con - CTRL-S, si no es igual saltar a ANY.
Si es igual, enviar CTRL-S al computador, setear flags 3 y 4 y saltar a ANAL.

```
12)ANY      LD A,(TEMP)
            CALL OQSTX
   ANAL     CALL CONT1
```

Restauramos valor de A (que contiene un caracter que no es de control), lo enviamos al computador y finalmente llamamos a la rutina de - comparación de punteros CONT1.

2.2.4.3 Rutina de transmisión

```
1)TRANSM    BIT 1,(IX)
            JR Z,BYPASS
```

Si hay que transmitir 2 bloques saltar a - BYPASS.

```
                BIT 2,(IX)
                JR NZ,GAMMA
```

Si no se recibió ACK saltar a GAMMA.

```
2)BYPASS        BIT 3,(IX)
                JR NZ,GAMMA
```

Si se recibió un CTRL-S desde la printer, saltar a GAMMA.

```
                BIT 5,(IX)
                JR NZ,DELTA
```

Si hay que transmitir un ETX saltar a DELTA.

```
3)TX            IN A,(#4F)
                BIT 0,A
                JR Z,GAMMA
```

Está el buffer de transmisión vacío?. Si no lo está saltar a GAMMA.

```
                LD A,(DE)
                OUT (#4E),A
```

Enviar caracter a la printer.

```
                INC B
                LD A,B
                CP 110
                JR NZ,ALFA
```

Incrementamos contador de caracteres, si todavía no es igual a 110 saltar a ALFA.

```
4)DELTA         IN A,(#4F)
                BIT 0,A
                JR Z,OMEGA
```

Está el buffer de transmisión vacío?, si no - lo está saltar a OMEGA.

```
                LD A,ETX
                OUT (#4E),A
                RES 5,(IX)
                JR XX
```

Enviar caracter ETX a la printer, resetear - flag 5 y saltar a XX.

```
OMEGA           SET 5,(IX)
                JR GAMMA
```

Al no estar el buffer vacío, setear flag 5, - que será consultado al retorno a la rutina - TRANSM, antes de transmitir cualquier caracter.

```
XX              LD B,0
                SET 2,(IX)
```

Resetear contador de bytes y setear flag 2 - indicando que se espera un ACK.

```
5)              INC C
                LD A,C
                CP 2
                JR NZ,ALFA
                SET 1,(IX)
                LD C,3
```

El registro C es un contador auxiliar para determinar si se transmitieron 2 bloques.
Al terminar el segundo bloque, C = 2 y se setea flag 1 indicando que la transmisión será - por bloques individuales. Luego cargamos C -

con 3 (o cualquier otro valor), para evitar - de que C recicle a cero.

```
6)ALFA      INC DE
            LD (HLTEMP),HL
            LD HL,MAXDE
            DEC HL
            XOR A
            SBC HL,DE
            JP P,BETA
```

Alcanzó DE el límite superior del buffer?. si no ,saltar a BETA. Luego de cargar HL con MAXDE, decrementamos porque MAXDE es igual al límite superior del buffer más uno.

```
            LD DE,#0400
            RES 0,(IX)
```

Reinicializamos DE a 0400, el límite inferior del buffer. Reseteamos flag 0 indicando que - ahora DE > HL.

```
BETA        LD HL,(HLTEMP)

GAMMA       CALL CONT1
            JP RECEPC
```

Recuperamos HL, llamamos a la rutina de comparación de punteros y saltamos a recepción.

2.2.4.4 Rutina de comparación de punteros CONT1

```
1)CONT1     LD (HLTEMP),HL
            LD (DETEMP),DE
```

Almacenamos temporalmente HL y DE.

```
XOR A
SBC HL,DE
JR NZ,HLDEDIF
```

Comparamos punteros, si son diferentes saltar a HLDEDIF.

```
BIT 0,(IX)
POP HL
LD HL,(HLTEMP)
JR Z,RECEPC
JP TRANSM
```

Si los punteros son iguales, vemos si es necesario ir a recepción o a transmisión, consultando con flag 0. La instrucción POP HL - es"ciega" . Como vamos a salir de la subrutina CONTl sin usar la instrucción RET hay que descargar del STACK la dirección de retorno, si no se hace esto el STACK se llenaría de - direcciones de retorno.Al final restauramos HL.

La figura 2.8 explica los dos casos que se - presentan cuando los dos punteros son iguales.

```
2)HLDEDIF       JP M,INTERC
```

Si HL < DE saltar a INTERC para intercambiar los punteros.

```
BIT 4,(IX)
JR NZ,NOV
```

Si el conversor mandó un CTRL-S saltar a NOV.

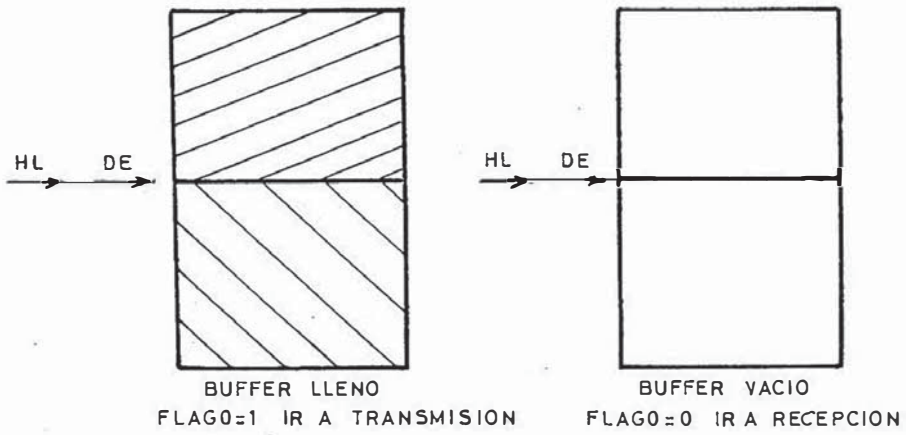


FIG. 2.8
CASOS EN QUE LOS PUNTEROS COINCIDEN

```
LD DE,1000
XOR A
SBC HL,DE
JP M,REST
```

Si el buffer no se llenó hasta 1000 caracteres, saltar a REST. (HL mantiene la diferencia de punteros hallada en la parte 1).

```
LD A,CTRLS
CALL OQSTX
SET 4,(IX)
JR REST
```

Si el buffer se llenó hasta 1000 caracteres, enviar un CTRL-S deteniendo al computador, - mediante la rutina OQSTX, setear flag 4 indi cando que el adaptador envió un CTRL-S, posteriormente saltar a REST.

3)NOV

```
LD DE,110
XOR A
SBC HL,DE
JP P,REST
```

Si la diferencia es mayor que 110 saltar a - REST.

```
LD A,CTRLQ
CALL OQSTX
RES 4,(IX)
JR REST
```

Si la diferencia es menor que 110 mandar un CTRL-Q, y resetear el flag 4.

```
4)INTERC        LD HL,(HLTEMP)
                EX DE,HL
```

Restaurar HL e intercambiar DE con HL para poder restar ya que DE > HL.

```
                BIT 4,(IX)
                JR NZ,CINZ
```

Si el conversor mandó un CTRL-S saltar a - CINC.

```
                XOR A
                SBC HL,DE
                LD DE,10
                XOR A
                SBC HL,DE
                JP P,REST
```

Si la diferencia de punteros es mayor que - 10 saltar a REST.

```
                LD A,CTRLS
                CALL OQSTX
                SET 4,(IX)
                JR REST
```

Si la parte vacía del buffer es menor que - 10 mandar un CTRL-S, setear flag 4 y saltar a REST.

```
5)CINC          XOR A
                SBC HL,DE
                LD DE,900
                XOR A
                SBC HL,DE
                JP N,REST
```

Después que el conversor envió un CTRL-S, - vemos si la cantidad de buffer vacía aumentó a 900. Si no aumentó saltar a REST.

```
LD A,CTRLQ
CALL OQSTX
RES 4,(IX)
```

Si aumentó a 900 mandar un CTRL-Q.

```
6)REST      LD HL,(HLTEMP)
            LD DE,(DETEMP)
            RET
```

Restaurar HL y DE; retornar.

2.2.4.5 La rutina OQSTX y la de errores

```
ERRORTX     LD A,#37
            JR MES

ERRORO      LD A,#BD
            JR MES

ERRORF      LD A,#OF

MES         OUT (#3E),A
            LD A,#15
            OUT (#OF),A
            RET
```

En cada caso cargamos el acumulador con el - código necesario para producir el mensaje deseado en el display y finalmente reseteamos - los errores del USART. Dichos códigos aparecerán en el display como:

ERRORTX: letra "H"

```
ERRORO:   Letra "O"
ERRORF:   letra "F"
OQSTX     PUSH AF
          IN A,(#OF)
          BIT O,A
          CALL Z,ERRORTX
          POP AF
          OUT (#OE),A
          RET
```

Guardamos el acumulador, vemos si el buffer de transmisión del USART está vacío, si no lo está llamar a la rutina de errores. En todo caso se envía el caracter al computador.

2.2.5 Diagramas de flujo

Las figuras 2.10, 2.11, 2.12 y 2.13 muestran los diagramas de flujo de inicialización, de recepción, de transmisión y de comparación de punteros.

2.2.6 Mapa de la memoria

La figura 2.14 es un diagrama simplificado del mapa de la memoria del adaptador.
Las posiciones entre 0000 y 03FF corresponden a la ROM; y entre 0400 y 07FF a la RAM; donde el STACK tiene un área reservada de 07F6 a 07FF. Las variables del sistema se almacenan entre 07FD y 07F5. El espacio restante corresponde a la "FIFO".

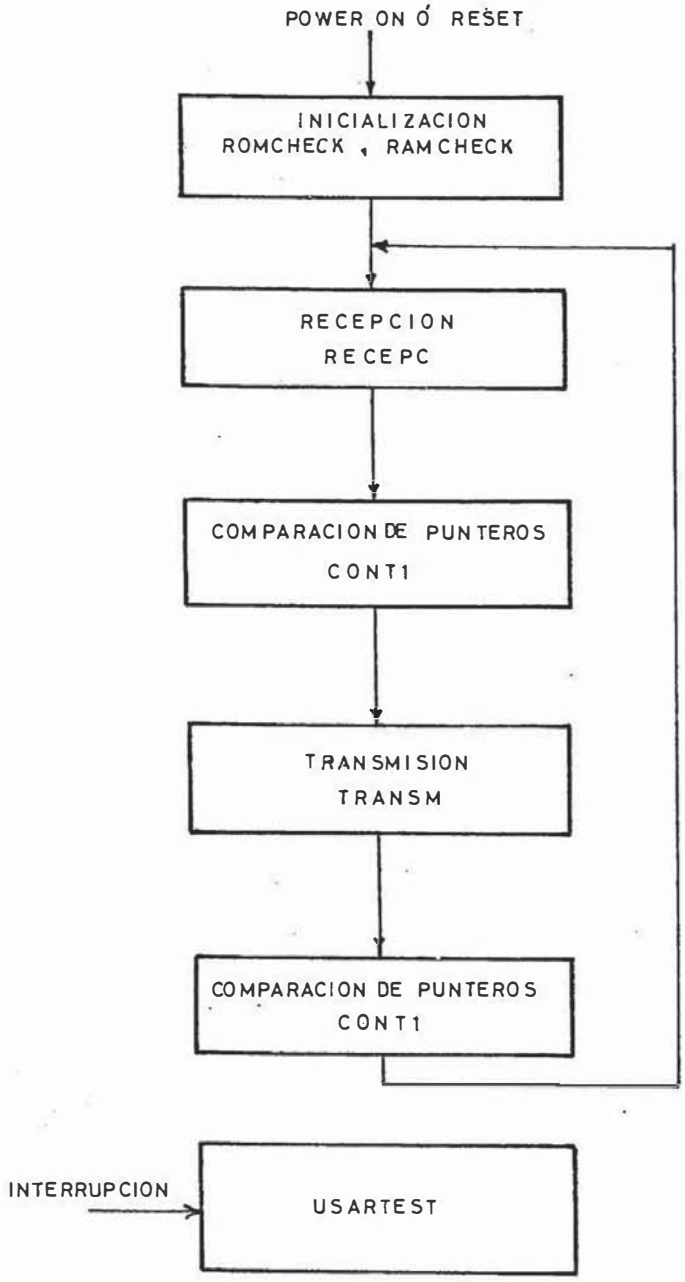


FIG. 2.9 RELACION ENTRE LAS DIFERENTES RUTINAS

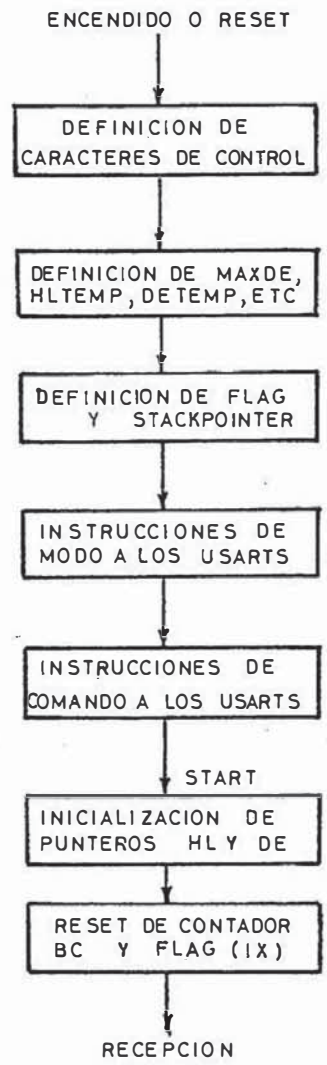


FIG. 2.10 DIAGRAMA DE FLUJO DETALLADO DE LA RUTINA DE INICIALIZACION

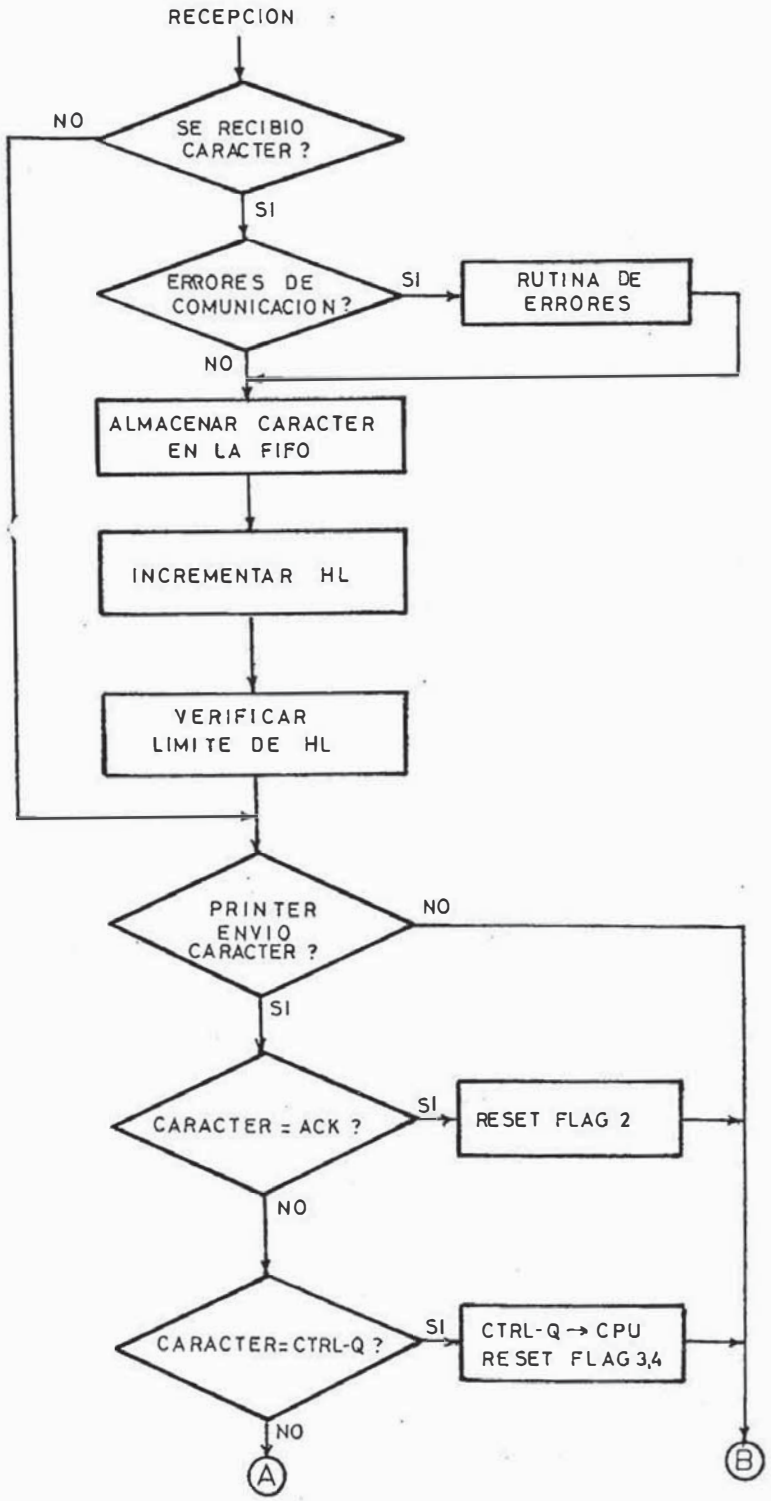


FIG. 2.11a DIAGRAMA DE FLUJO DETALLADO DE LA RUTINA DE RECEPCION

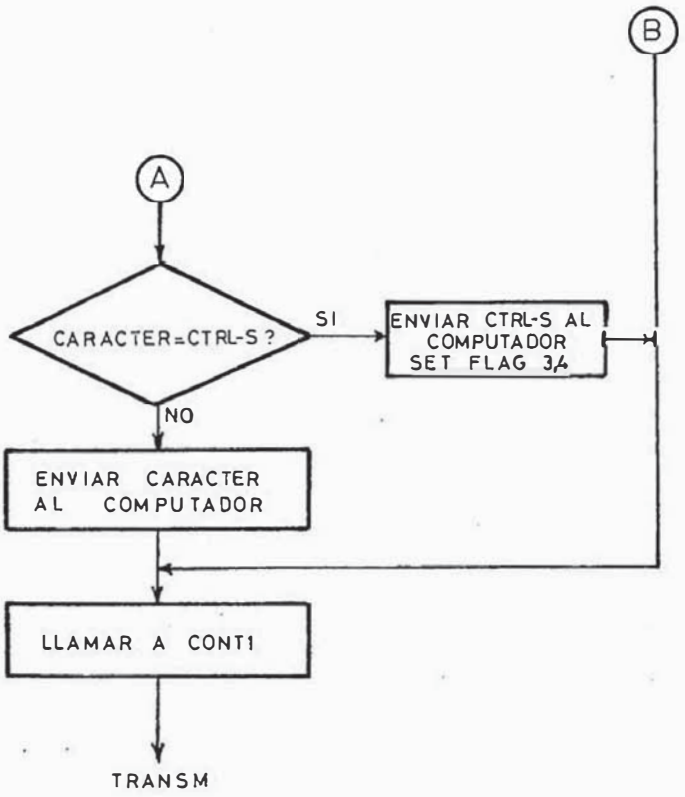


FIG. 2.11b DIAGRAMA DE FLUJO DETALLADO DE LA RUTINA DE RECEPCION

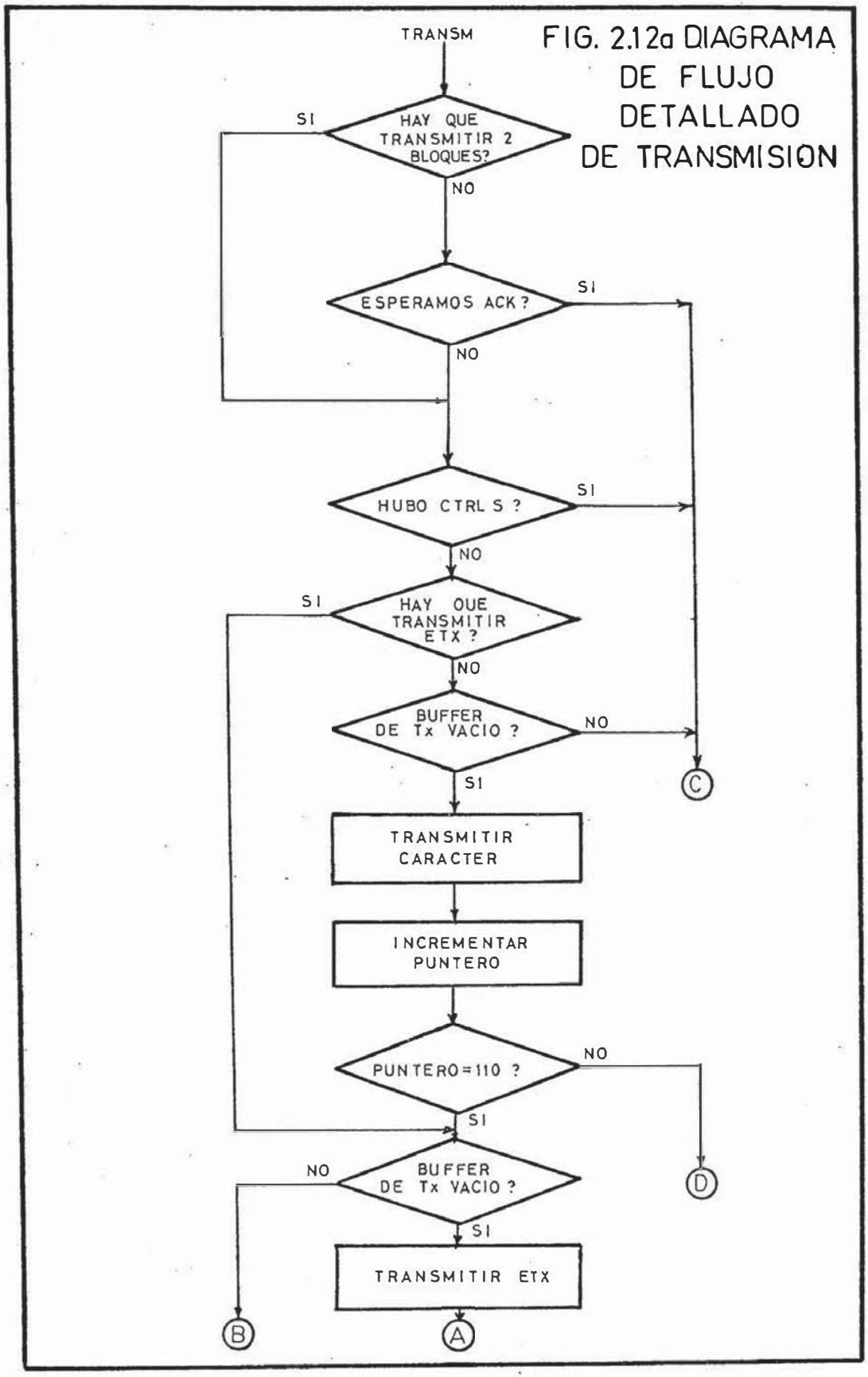
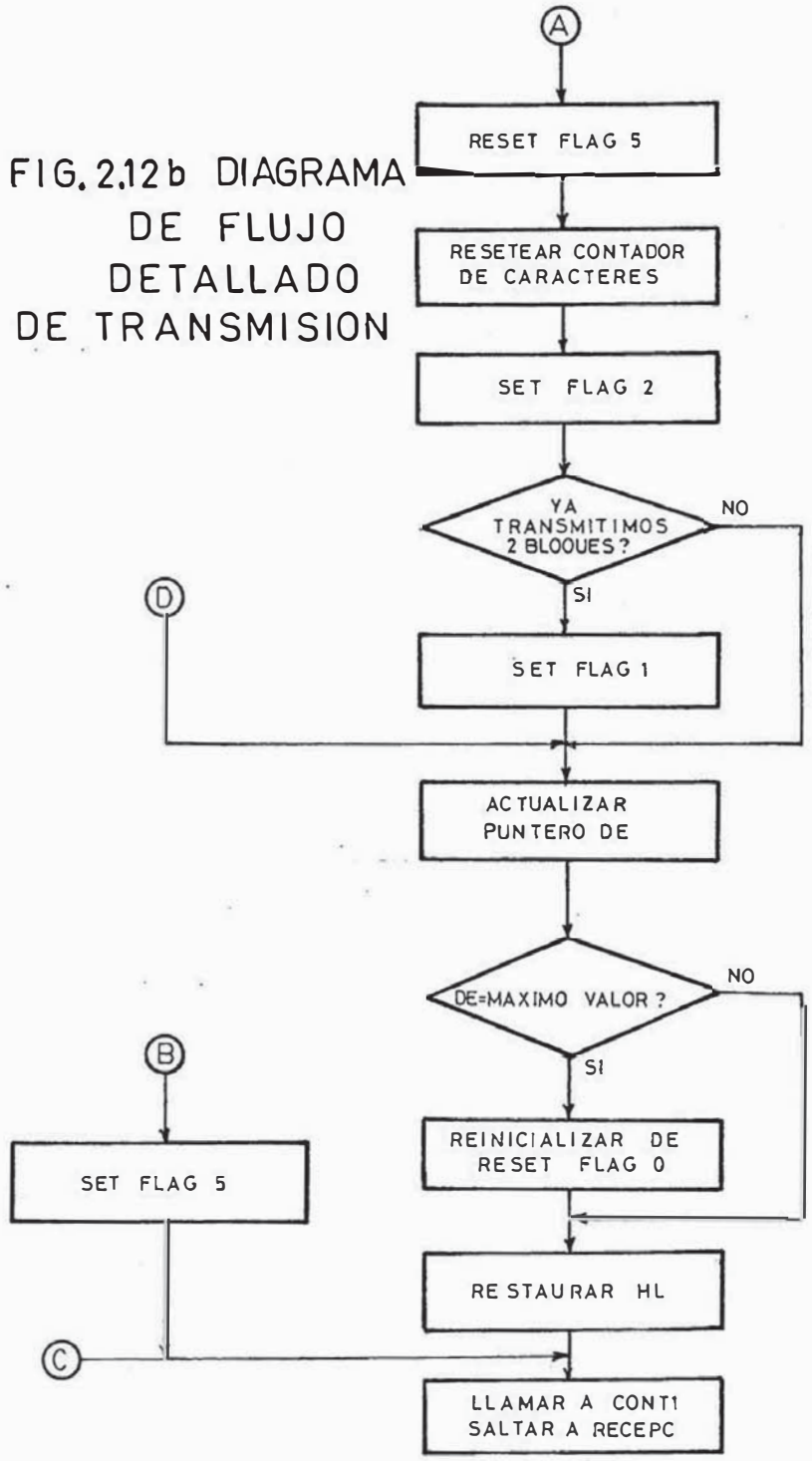


FIG. 2.12a DIAGRAMA DE FLUJO DETALLADO DE TRANSMISION

FIG. 2.12b DIAGRAMA DE FLUJO DETALLADO DE TRANSMISION



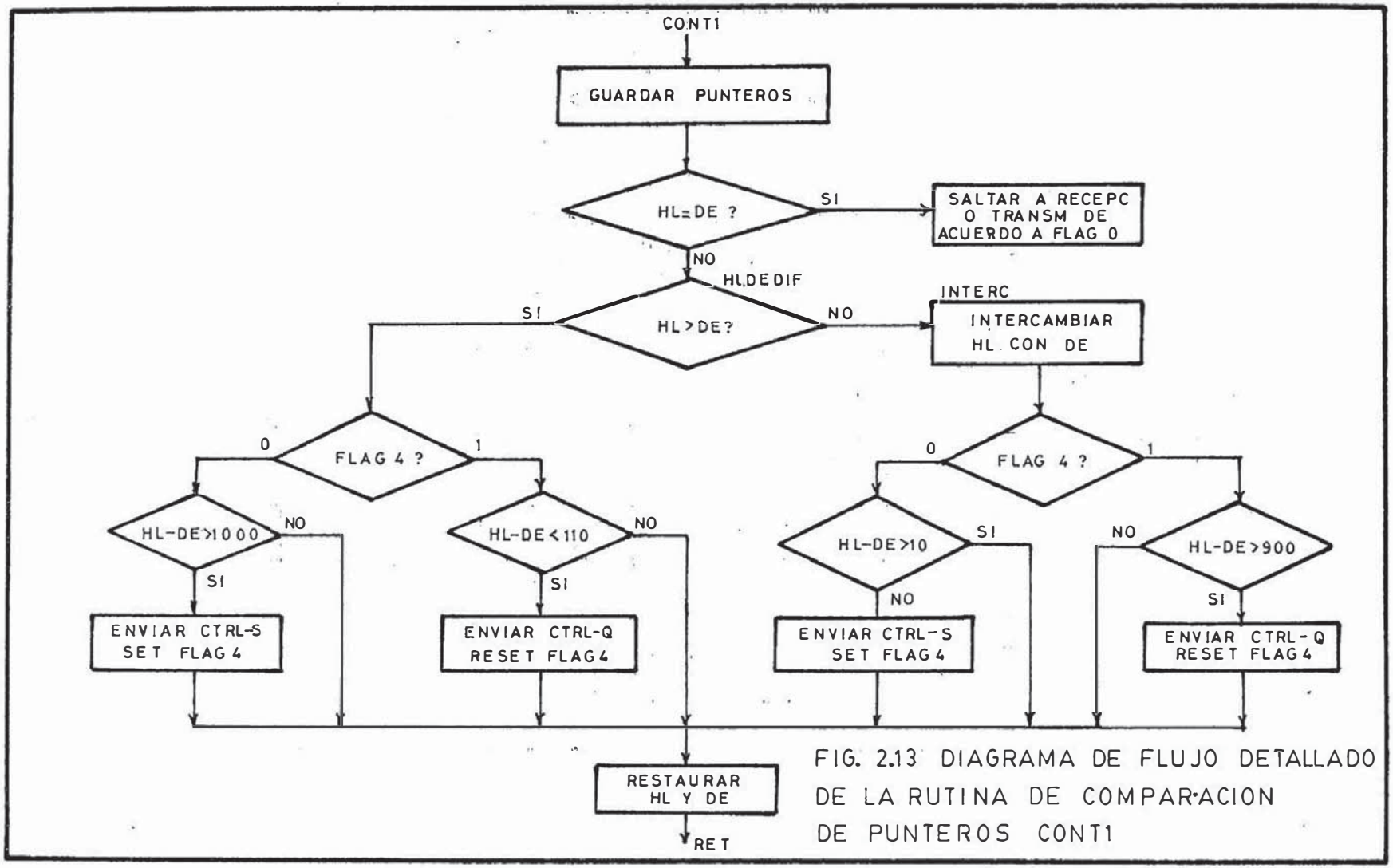


FIG. 2.13 DIAGRAMA DE FLUJO DETALLADO DE LA RUTINA DE COMPARACION DE PUNTEROS CONT1

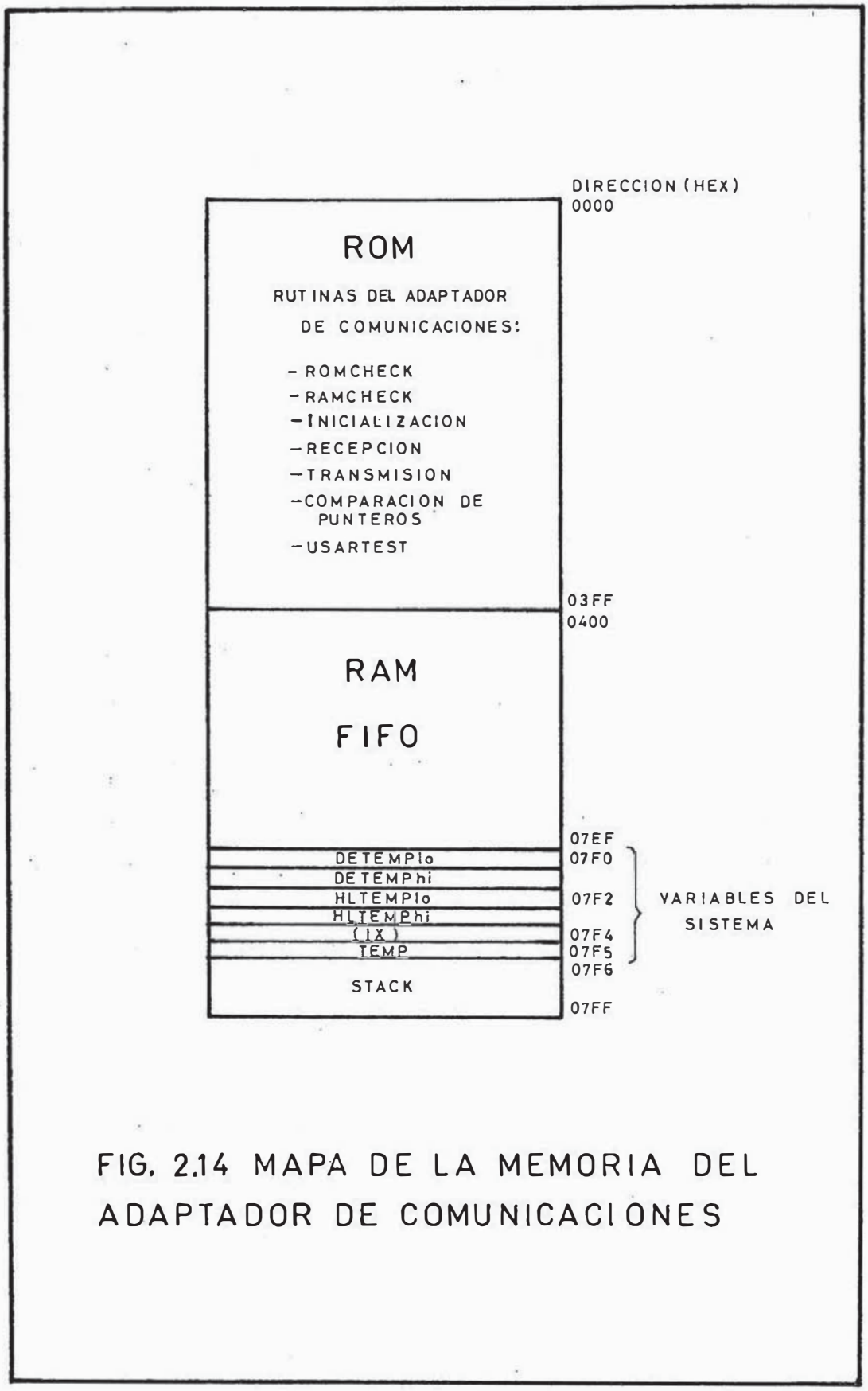


FIG. 2.14 MAPA DE LA MEMORIA DEL ADAPTADOR DE COMUNICACIONES

2.2.7 Cálculo del tiempo que toma cada rutina

Basado en las rutinas y de acuerdo a las especificaciones del procesador Z80 en cuanto a los números de períodos T del clock que toma para realizar cada instrucción, tenemos que:

A) Rutina de recepción

Se calculó el máximo tiempo que puede tomar esta rutina, que corresponde al caso cuando el computador envía un caracter con error de overflow y framing, se reinicializa HL a 0400 y la impresora envía un CTRL-S al computador. Esto equivale a 704T, donde T es el período del clock.

B) Rutina de transmisión

Aquí se considera el caso en que no es necesario enviar dos bloques, la impresora no espera ACK, no envió CTRL-S; aún no hay que enviar ETX, el buffer de transmisión está vacío; registro B igual a 110 caracteres; registro C diferente de 2 y hay que reinicializar DE; el resultado: 459T.

C) Rutina de comparación de punteros

Esta es la rutina donde hay más bifurcaciones. El máximo tiempo ocurre cuando los punteros son diferentes, HL es menor que DE, el adaptador no envió CTRL-S y recién se llena el buffer: 429T.

Si tomamos los tiempos máximos de

cada rutina tendremos:

(RECEPC) + (CONT1) + (TRANSM) + (CONT1) =

704T + 429T + 459T + 429T = 2021T

Este tiempo debe ser inferior al tiempo que un caracter enviado por el computador en llenar el buffer de recepción del - USART, para que no ocurra overflow, que es - de:

(# de bits/caracter) /velocidad de transmisión = (10 bits/caracter)/(1200 bits/seg) = 8.33 ms/caracter

De aquí deducimos que el clock - del Z80 puede ser tan bajo como:

2021T = 8.33 ms

Con lo que tenemos T = 4.12 us

o lo que es lo mismo f = 1/T = 242.6 Khz

Por ser comercial usaremos:

f = 3.579545 Mhz, que es la frecuencia de la subportadora de color en el sistema NTSC de televisión. Cristales de esta frecuencia se pueden encontrar fácilmente en el mercado.

2.2.8 Discusión sobre el buffer del adaptador

Ya sabemos que el buffer es tratado como una memoria FIFO(First In First Out), - lo que se trata a continuación es hasta cuánto se puede llenar el buffer sin que haya peligro

de overflow, por exceso de data que no pudiera entrar en memoria por estar ya llena.

Asumimos un valor de seguridad - de 8 a 10 caracteres, pudiendo darse cualquiera de las dos situaciones mostradas en los - gráficos de la figura 2.15.

Una vez que el espacio vacío en el buffer sea igual a 8 ó 10, el adaptador envía un CTRL-S al computador deteniendo el flujo de información.

Para evitar un constante control de flujo dejamos que el buffer se vacíe hasta que la cantidad de buffer ocupado sea un bloque de 110 caracteres.

2.2.9 Diagramas de tiempo

Los gráficos 2.16a y 2.16b, muestran cómo será teóricamente el comportamiento de los diferentes parámentros involucrados en este diseño, como son:

-Transmisión de datos del computador hacia el adaptador.
-Transmisión de datos del adaptador hacia la printer.
-Envío de la señal ACK por parte de la impresora.
-La variación de los flags 1 y 2.
-Comportamiento del buffer del adaptador y del buffer de la printer.

Es de notar que en estos gráficos

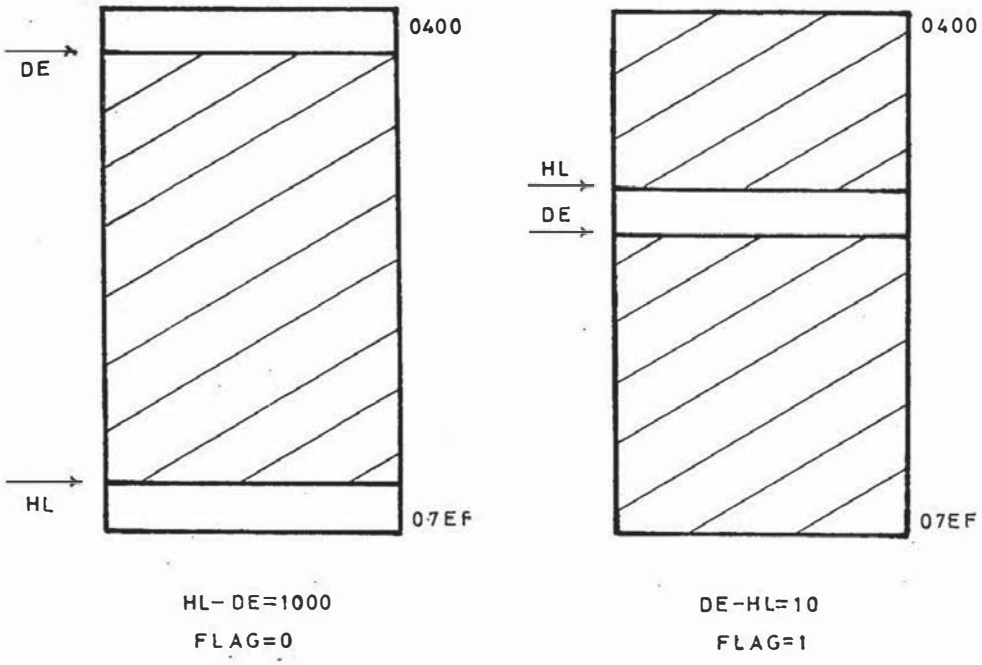


FIG. 2.15 LOS DOS CASOS EN QUE EL ADAPTADOR ENVIA CTRL-S AL COMPUTADOR PARA DETENER LA TRANSMISION

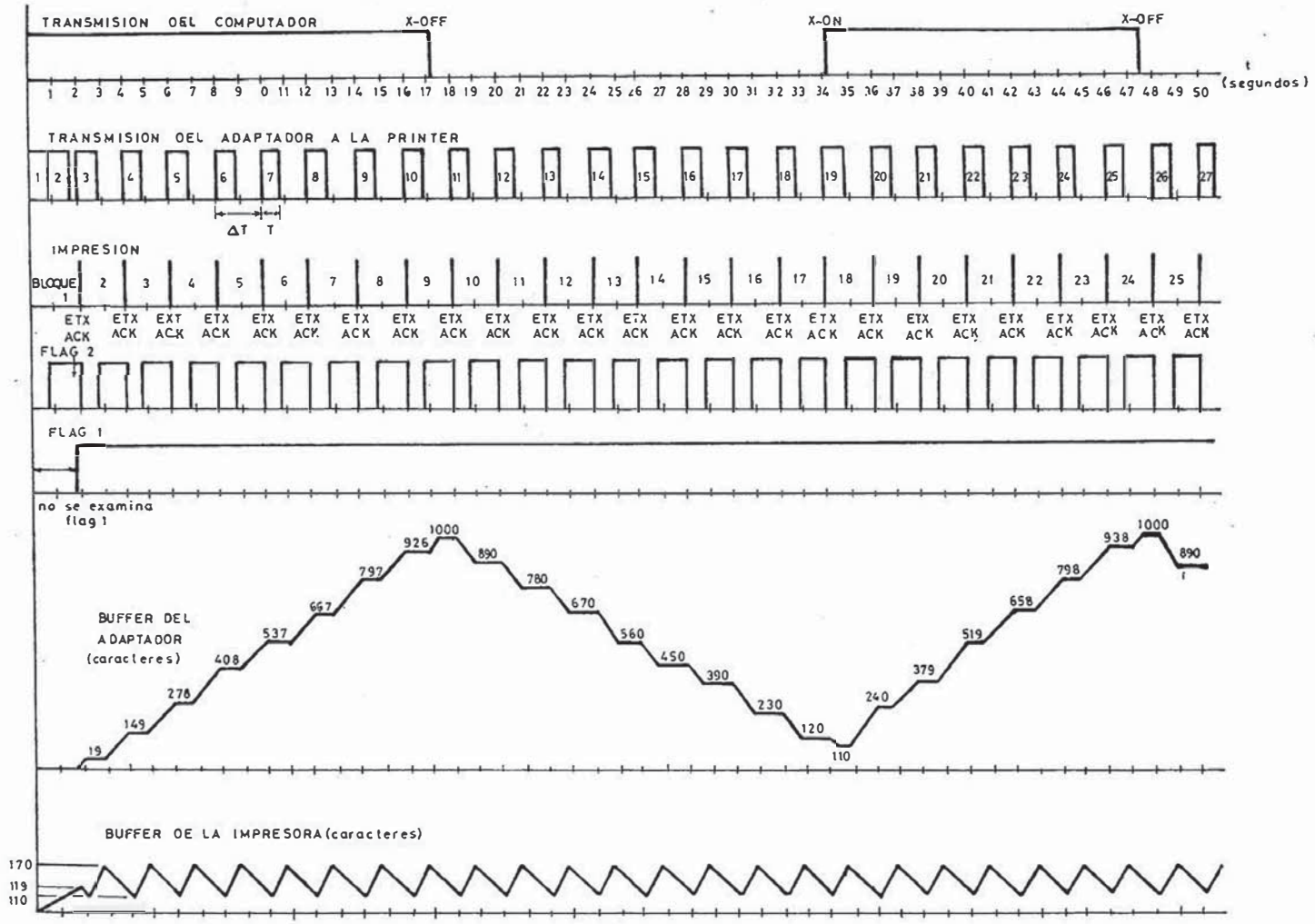


FIG. 2.16 COMPORTAMIENTO DE DIFERENTES PARAMETROS EN EL TIEMPO

se asume una velocidad de impresión constante - de 55 caracteres por segundo. Con archivos reales, la velocidad promedio de impresión es de- aproximadamente 40 caracteres por segundo. Esto sólo significaría que los diagramas se extenderían en el tiempo manteniéndose las relaciones entre los parámetros.

El tiempo tomado para transmitir - cada bloque está dado por:

T = (110 caracteres * 10 bits/caracter) (1/1200 bits/seg) = 11/12 segundos = 0.92 segundos.

Y el tiempo de impresión de cada - bloque:

ΔT = (110 caracteres/55 caracteres/segundo) = 2 segundos.

En los cálculos anteriores se - consideró que el computador transmite a 1200 bps. Si fuese 4800 bps por ejemplo, los tiempos T y ΔT seguirían iguales, lo que cambiaría serían los gráficos:

-Transmisión del computador: X-ON y X-OFF ocurren más frecuentemente.
-Buffer del adaptador: este gráfico se estrecha en el tiempo. Las pendientes serían distintas: 480 caracteres/segundo y 360 caracteres/ segundo en vez de 120 y 0 caracteres/segundo - respectivamente. Además el buffer alcanza los 100 caracteres en menor tiempo.

2.2.10 Comportamiento del buffer del adaptador a través del tiempo

Los gráficos 2.17a y 2.17b muestran cómo se va llenando el buffer del adaptador. Las zonas en blanco corresponden a áreas del buffer que no contienen información útil, es decir que ya fue procesada(imprimida), mientras que las zonas achuradas representan la información a imprimir.

T = 0 es el instante en que el computador empieza a enviar información a la impresora. Entre T = 0 y T = 16 el buffer se comienza a llenar, hasta que entre T = 16 y T = 18, el adaptador envía un CTRL-S deteniendo al computador; el puntero HL queda "congelado" en 76.

En T = 34.2 el buffer se vacía hasta 110 caracteres, se reinicia la transmisión del computador y el buffer empieza a llenarse nuevamente repitiéndose el ciclo en adelante.

Nota: HL es el puntero de llenado del buffer, siempre apunta a la próxima posición a ser escrita por el computador; mientras que DE es el puntero de " descarga" del buffer y apunta a la posición de lectura hacia la impresora.

2.3 Rutinas de selftest o autoprueba

2.3.1 Introducción

Tenemos básicamente tres rutinas

FIG. 2.17a COMPORTAMIENTO DEL BUFFER DEL ADAPTADOR A TRAVES DEL TIEMPO

FIG. 2.17b COMPORTAMIENTO DEL BUFFER DEL ADAPTADOR A TRAVES DEL TIEMPO

en software y un circuito que chequea las fuentes de alimentación. La primera de las rutinas es el chequeo de la ROM donde está el firmware. La segunda chequeará la integridad de la memoria RAM y la tercera que no es en sí una autoprueba sino que se activa mediante un pushbutton, chequea los dos USARTS.

El pushbutton está conectado al pin $\overline{INT}$ del Z80. al inicializar el sistema se escoge el modo de interrupción 2. El vector de interrupción es cargado con el valor hexadecimal 02; ya que no existe un dispositivo real - que realice la interrupción, la dirección de - interrupción se convierte en 02FF, que es la - dirección de inicio de la prueba de los Usarts: USARTEST, como se indica en la figura 2.18.

Las dos primeras rutinas: ROMCHECK y RAMCHECK, se corren en el encendido; si pasan OK, el sistema queda listo para operar; si no pasan, se muestra un mensaje de error y - se detiene.

2.3.2 Requerimientos

Sobre el chequeo de la ROM: debe verificarse todas las posiciones de la ROM. Lo ideal sería chequearlas contra otra copia de - los programas pero esto no es posible ya que en la ROM sólo hay una copia de ellos. Sin embargo lo que se hace garantiza probabilísticamente la integridad de la ROM.

Sobre la RAM: igualmente todas -

FIG. 2.18 RUTINAS DE SELFTEST

las posiciones de la RAM deberán ser verificadas una por una con alguna técnica simple y - segura.

Sobre el chequeo de los USARTS: debe garantizarse que cada uno de ellos transmita y reciba sin problemas y sin introducir - errores.

Sobre las fuentes de alimentación: el equipo tiene cuatro fuentes: +5V que es la principal, alimenta prácticamente a todos los circuitos integrados; y tres fuentes +12V,-12V y -5V.

Este no es un test en software - sino en hardware, aunque inicialmente se pensó usar un port de entrada para analizar las fuentes, se vió que era más sencillo hacerlo por - hardware. El circuito que chequee las fuentes deberá indicar ausencia de alimentación en la forma más sencilla posible.

2.3.3 Concepción de las rutinas de autoprueba

Se explica brevemente en qué consiste cada prueba y cómo se realizan.

2.3.3.1 Chequeo de la ROM

La idea en este caso es sumar una a una todas las posiciones útiles de la - memoria ROM y añadirle a la siguiente posición libre un valor de manera que la suma - final sea un valor conocido, como por ejemplo 00 en hexadecimal. Si la suma no da 00 -

significaría que alguna(s) posición(es) está(n) mala(s) en la ROM. Pero pudiera darse el caso remoto de que los errores se "compensarán" y la suma final sea también 00. Si la suma final no da 00, el equipo no debe trabajar y se detendrá mostrando el mensaje de error "A" en el display de siete segmentos.

La figura 2.19 puestra gráficamente lo explicado.

2.3.3.2 Chequeo de la RAM

Se prueba una a una todas las posiciones de la RAM de la siguiente manera:

1)Se toma una posición de memoria.
2)Se carga el contenido de dicha posición de memoria en el acumulador(por ejemplo BIT 0 = 0)
3)Se complementa A(BIT 0 = 1)
4)Se carga A en la misma posición de memoria.
5)Se vuelve a cargar A en la misma posición de memoria(BIT 0 = 1)
6)Se complementa A(BIT 0 = 0)
7)Se devuelve a memoria el contenido del acumulador.
8)Se compara A con el contenido de la memoria (BIT 0 = 0 ?)

Con lo que se probó dicha posición de memoria sin alterar su contenido, que sería una ventaja en otras condiciones

si se pudiera llamar esta rutina y no malograr la información almacenada, lo que no es nuestro caso ya que esta prueba se corre durante el POWER UP.

En el supuesto de que dicha posición haya estado mal, es decir que por alguna razón siempre está en 0; entonces en el paso 5 tendríamos BIT 0 =0; paso 6 BIT 0 = 1; paso 7, BIT 0 = 0 (ya que la posición está mala y no se puede escribir sobre ella), finalmente al comparar veríamos que no son iguales, con lo que detenemos la prueba.

2.3.3.3 Chequeo de los USARTS

Esta es una rutina de chequeo que necesita la intervención de un operador que haga un loop cruzado entre los USARTS. El loop cruzado se define como la unión de Tx de un USART con Rx del otro, como se muestra en la figura 2.21.

Se envía un caracter por un USART hacia el otro, este mismo caracter es enviado al primer USART, luego comparamos el caracter original con el que viajó a través de los dos USARTS, si son iguales probamos con el siguiente caracter; pero si son diferentes significa que alguno de los USARTS está trabajando mal.

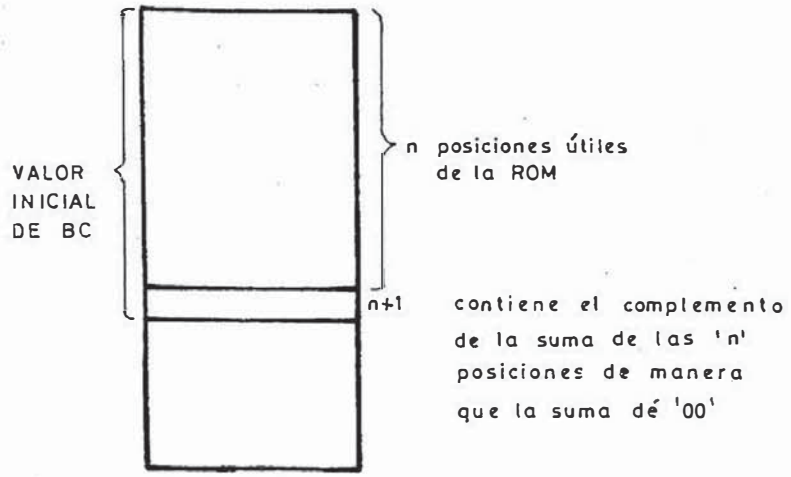


FIG. 2.19 CHEQUEO DE LA ROM

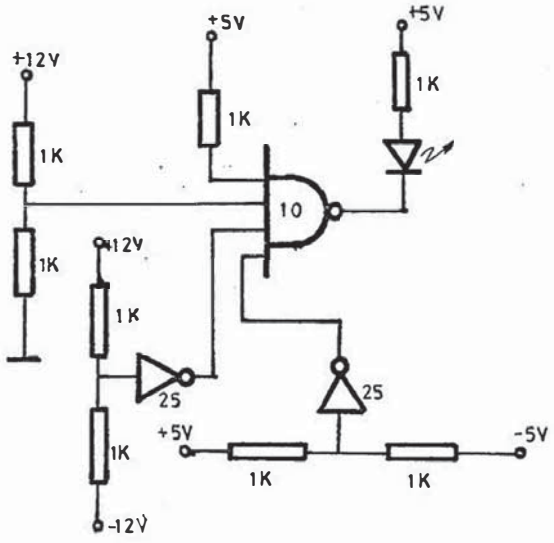


FIG. 2.20 MONITORADO DE FUENTES

FIG. 2.21 LOOP CRUZADO NECESARIO
PARA PRUEBA USARTEST

2.3.3.4 Circuito que monitora las fuentes de alimentación

Se usa un NAND TTL 7420 para analizae las cuatro fuentes, siendo la más importante la de +5V, ya que de esta fuente se alimentan la mayoría de los circuitos integrados que componen el adaptador.

La figura 2.20 muestra el circuito mencionado, donde tenemos:

LED de salida:	ON	Fuentes normales
	OFF	Falla en alguna fuente

Esta salida sería un equivalente al LED de POWER en otros equipos.

2.3.4 Diseño de las rutinas

2.3.4.1 Chequeo de la memoria ROM

```
ROMCHECK  LD HL,0          ;Posición inicial de
                           ;la ROM
          LD BC,770        ;# de bytes de ROM- 1
          XOR A            ;Clear CARRY FLAG
          ADD A,(HL)       ;Suma parcial de con-
                           ;tenido de memoria
          INC HL           ;Apuntar a próxima po
                           ;sición
          DEC BC           ;Decrementar contador
                           ;de bytes
          LD D,A           ;Guardar A
          LD A,C
          OR A,B           ;BC = 0 ?
```

```
                LD A,D          ;Restaurar A
                JR NZ,SUMA      ;Si hay más bytes ir
                                ;a SUMA
                CP 0            ;Comparar suma total
                                ;con 0
                JP Z,RAMCHECK   ;Si es cero ir  a
                                ;RAMCHECK
                LD A,#3F        ;Cargar acumulador con
                OUT(#3F),A      ;equivalente de mensa-
                                ;je A
                HALT            ;Detenerse
```

2.3.4.2 Chequeo de la memoria RAM

```
RAMCHECK        LD HL,#0400     ;Posición inicial RAM
                LD BC,1024      ;# de bytes a chequear
                LD A,(HL)       ;A <------ (HL)
                CPL             ;Complementar acumu-
                                ;lador
                LD (HL),A       ;(HL) <------ A
                LD A,(HL)       ;A <------ (HL)
                CPL             ;Complementar acumu-
                                ;lador
                LD (HL),A       ;(HL) <------ A
                CP (HL)         ;Comparar A con (HL)
                JP NZ,ERROR     ;Saltar en caso  de
                                ;error
                INC HL          ;Ir a la siguiente
                                ;posición
                DEC BC          ;Decrementar contador
                                ;de bytes
```

```
        LD A,C
        OR A,B          ;BC = 0 ?
        JR NZ,COMP      ;Si BC no es cero  ir
                        ;a COMP
        JP INICIO       ;Ir a las rutinas  de
                        ;trabajo
        LD A,#03        ;Cargar acumulador con
        OUT (#3F),A     ;el equivalente en sie
                        ;te segmentos de mensa
                        ;je "r".
        HALT            ;Detenerse
```

Notas:-El símbolo # significa hexadecimal
-El port 3F es el asignado al display de siete segmentos.

2.3.4.3 Rutina de chequeo de los USARTS

```
USARTEST LD B,32        ;Primer caracter ASCII
REC      IN A,(#4F)     ;Permanecer en el loop
         BIT 0,A        ;REC hasta que el bu-
         JR Z,REC       ;ffer de transm. esté
                        ;vacío
         LD A,B
         OUT (#4E),A    ;Caracter a port 4E
LOOP     IN A,(#0F)
         BIT 1,A        ;Se recibió caracter
         JR Z,LOOP      ;en port 0E?
         IN A,(#0E)     ;Caracter ⟶ A
         LD C,A         ;Guardar acumulador
RECI     IN A,(#0F)     ;Permanecer en el loop
```

```
        BIT 0,A         ;RECI hasta que el buffer
        JR Z,RECI       ;de transm.esté vacío
        LD A,C          ;Devolver el caracter
        OUT (#0E),A     ;al port 4E
LOOP2   IN A,(#4F)      ;Hay caracter de
        BIT 1,A         ;entrada en 4E
        JR Z,LOOP2
        IN A,(#4E)      ;Caracter ———→ A
        CP B            ;Caracter recibido =
                        ;caracter transmitido
        JR NZ,E1        ;En caso de error ir a E1
        INC B           ;Pasar al siguiente
                        ;caracter
        LD A,B
        CP 127          ;Ultimo caracter ASCII
        JR Z,USARTEST;Si es último ir a
                        ;USARTEST
        JR REC          ;Si no, ir a REC
E1      LD A,#B5        ;Equivalente a "U"
        OUT (#3F),A
        HALT            ;Detenerse
```

Nota: # significa hexadecimal.

Esta prueba es indefinida y se detiene ya sea reseteando el equipo o porque existe algún error producto del mal funcionamiento de alguno de los USARTS , lo que se mostrará en el display de siete segmentos con el mensaje "U".

2.3.5 Diagramas de flujo

Las figuras 2.22, 2.23 y 2.24 muestran respectivamente los diagramas de flujo de las rutinas de chequeo de la ROM, RAM y USARTS.

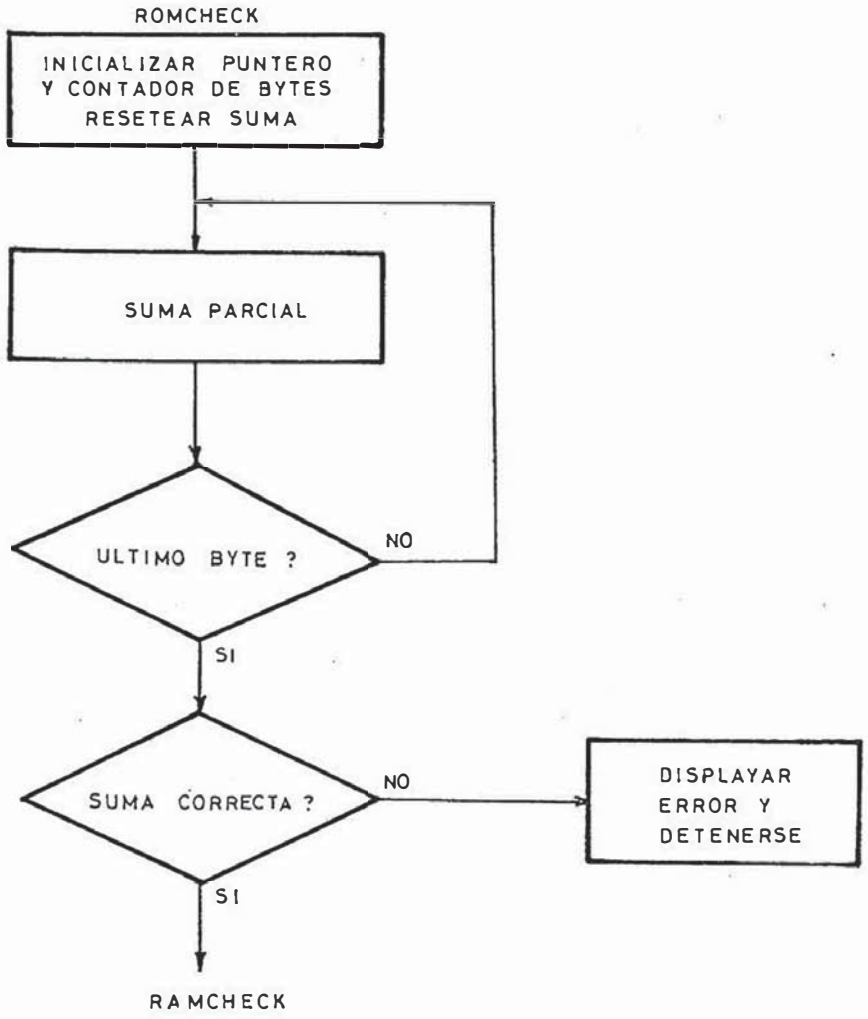


FIG. 2.22 DIAGRAMA DE FLUJO DE CHEQUEO DE LA ROM

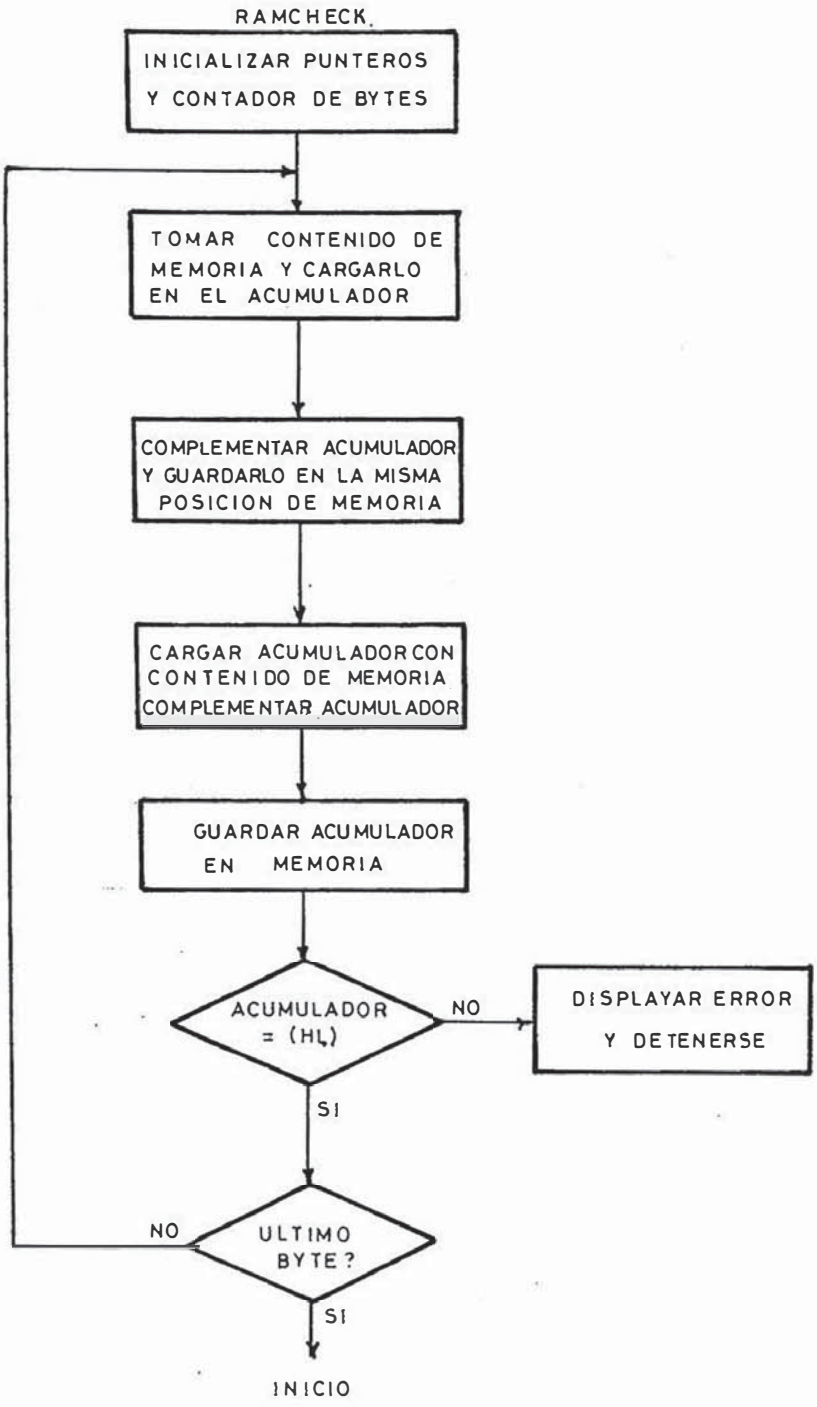


FIG. 2.23 DIAGRAMA DE FLUJO DE LA RUTINA DE CHEQUEO DE LA RAM

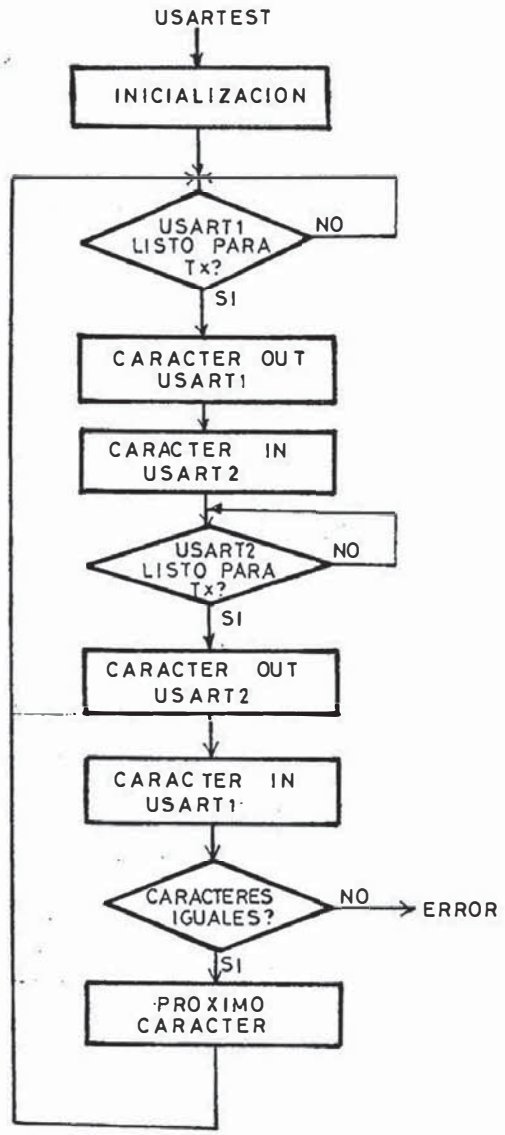


FIG. 2.24 DIAGRAMA DE FLUJO DE LA RUTINA DE CHEQUEO DE LOS USARTS

CAPITULO III

CONSTRUCCION

Construcción de prototipo de laboratorio

El prototipo de laboratorio se construyó alrededor de un microcomputador SPECTRUM de SINCLAIR, - cuyas características se detallan en el Anexo 5.7. Se - muestra también qué sector de la memoria se usó para la emulación; la distribución de circuitos integrados en - el prototipo; el diagrama circuital de la emulación; y finalmente las pruebas realizadas y los resultados obtenidos.

3.1 Emulación del CPU y de las memorias RAM y ROM.

El microcomputador SPECTRUM tiene la ventaja de permitir trabajar en lenguaje ASSEMBLY y así se evita el estar codificando cada instrucción en lenguaje de máquina; además se encuentra disponible a través de un conector de 56 pines, todas las señales del procesador, vale decir que se tiene acceso a los buses de datos, direcciones y de control del Z80 y otras señales propias del microcomputador.

En el prototipo el microcomputador reemplaza al microprocesador Z80, a la memoria RAM de 1K y a la ROM de 1K, comunicándose con el exterior a través del conector antes mencionado.

Para el bus de datos se usa un buffer - 74LS245.

Toda la circuitería restante permanece - prácticamente inalterada conforme a las figuras - 2.1, 2.2 y 2.3.

3.2 Mapa de memoria usada en la emulación

Ya que en el microcomputador SPECTRUM - los primeros 16K bytes corresponden a la ROM que contiene el sistema operativo más el intérprete - BASIC, tenemos que desplazar nuestros programas - hacia un sector de la memoria donde se pueda trabajar libremente. El área escogida es:

"ROM" : 7800 (30720) a 7BFF (31743)

"RAM" : 7C00 (31744) a 7FFF (32767)

Cada zona corresponde a 1K byte.

En la zona correspondiente a la ROM va - el programa ejecutable u objeto(en lenguaje de - máquina).

La "RAM" es el buffer del adaptador que es trabajada como una memoria FIFO. En las últimas posiciones de la RAM se encuentra el STACK, - aunque en el caso de la emulación se reservó 32 - bytes y no 10 como en el diseño original. La razón es que al existir un solo STACK y no dos (STACK - del usuario y STACK del sistema), al emsamblarse el programa fuente y ejecutarlo, el sistema operativo usa aproximadamente 16 posiciones.

Consiguientemente las variables del sistema quedarían en:

DETEMP : 7FDA
HLTEMP : 7FDC
IX : 7FDE
TEMP : 7FDF
SP : 7FFF
MAXDE : 7FDA

La figura 3.1 muestra la distribución de la memoria.

3.3 Distribución de los circuitos integrados en el prototipo

Todos los circuitos integrados se montaron sobre protoboards siguiendo la distribución indicada en la figura 3.2.

El cableado entre ICs se realizó con cable telefónico calibre AWG#24, la alimentación se llevó con cable AWG20, se usó condensadores de desacoplo de 0.1 uF en cada protoboard.

Se trató siempre de utilizar la menor distancia entre dos puntos a conectarse, pero sin pasar sobre los integrados para que el cableado se vea uniforme y se puedan hacer mediciones en los pines de los integrados.

3.4 Diagramas circuitales de la emulación

Las figuras 3.3, 3.4 y 3.5 muestran cómo se realizó la emulación, básicamente es el mismo circuito de las figuras 2.1, 2.2 y 2.3; pero

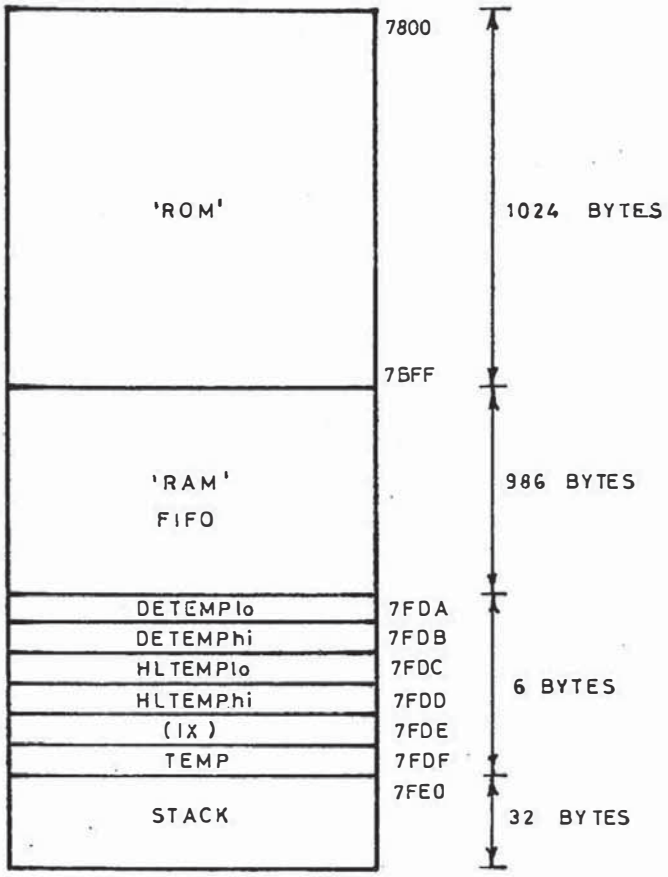


FIG. 3.1 MAPA DE LA MEMORIA USADA EN LA EMULACION

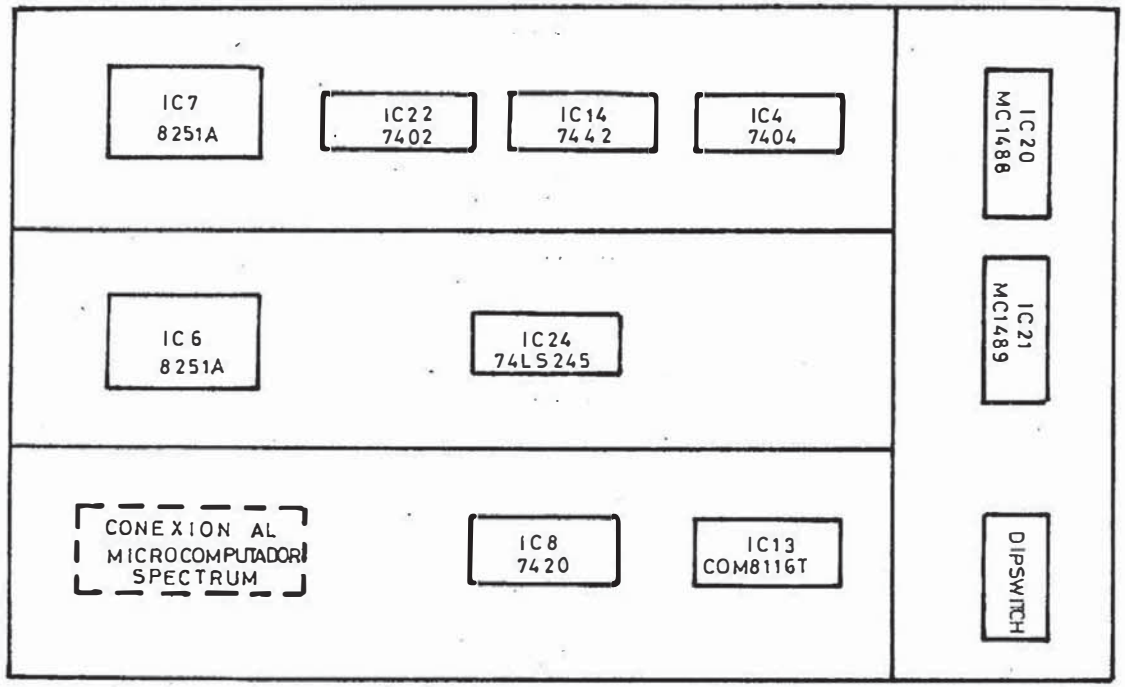


FIG. 3.2 DISTRIBUCION DE CIRCUITOS INTEGRADOS USADOS EN LA EMULACION

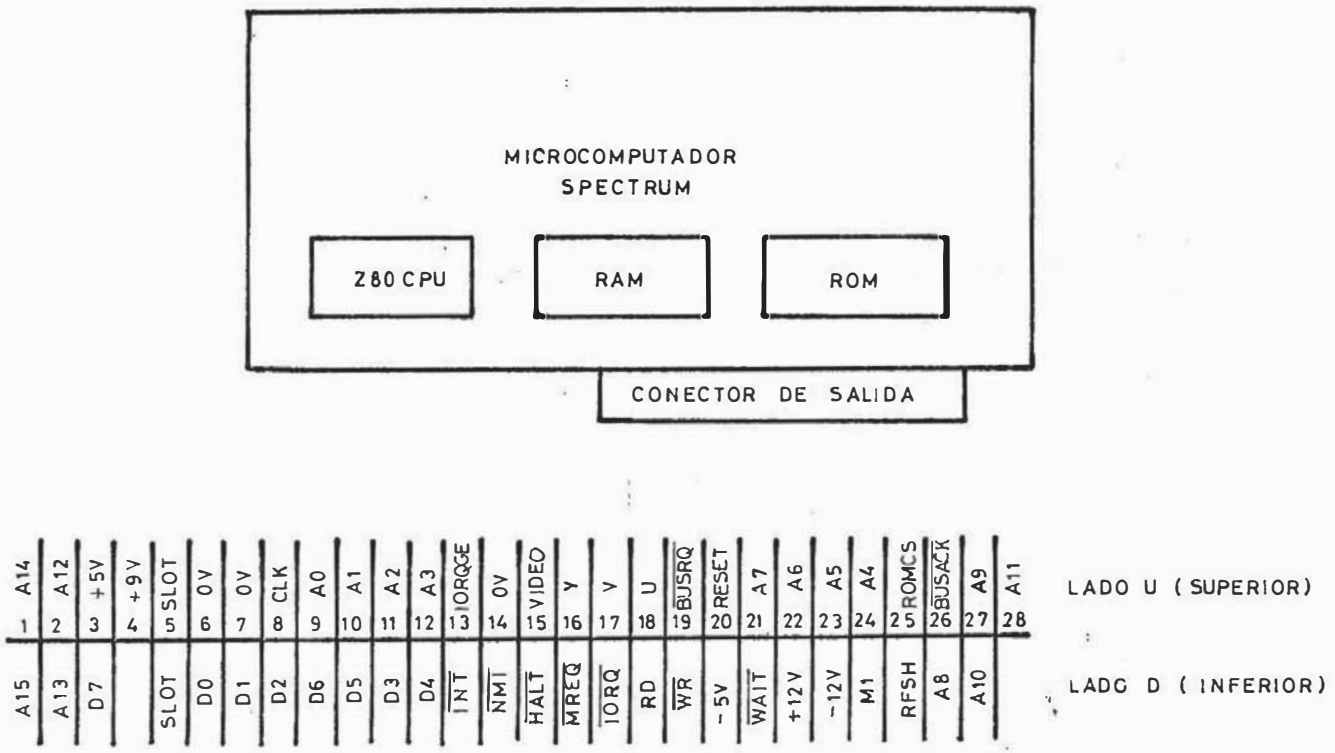


FIG. 3.3 EMULACION DEL CPU, RAM, ROM MEDIANTE EL MICROCOMPUTADOR SPECTRUM Y SU CONECTOR

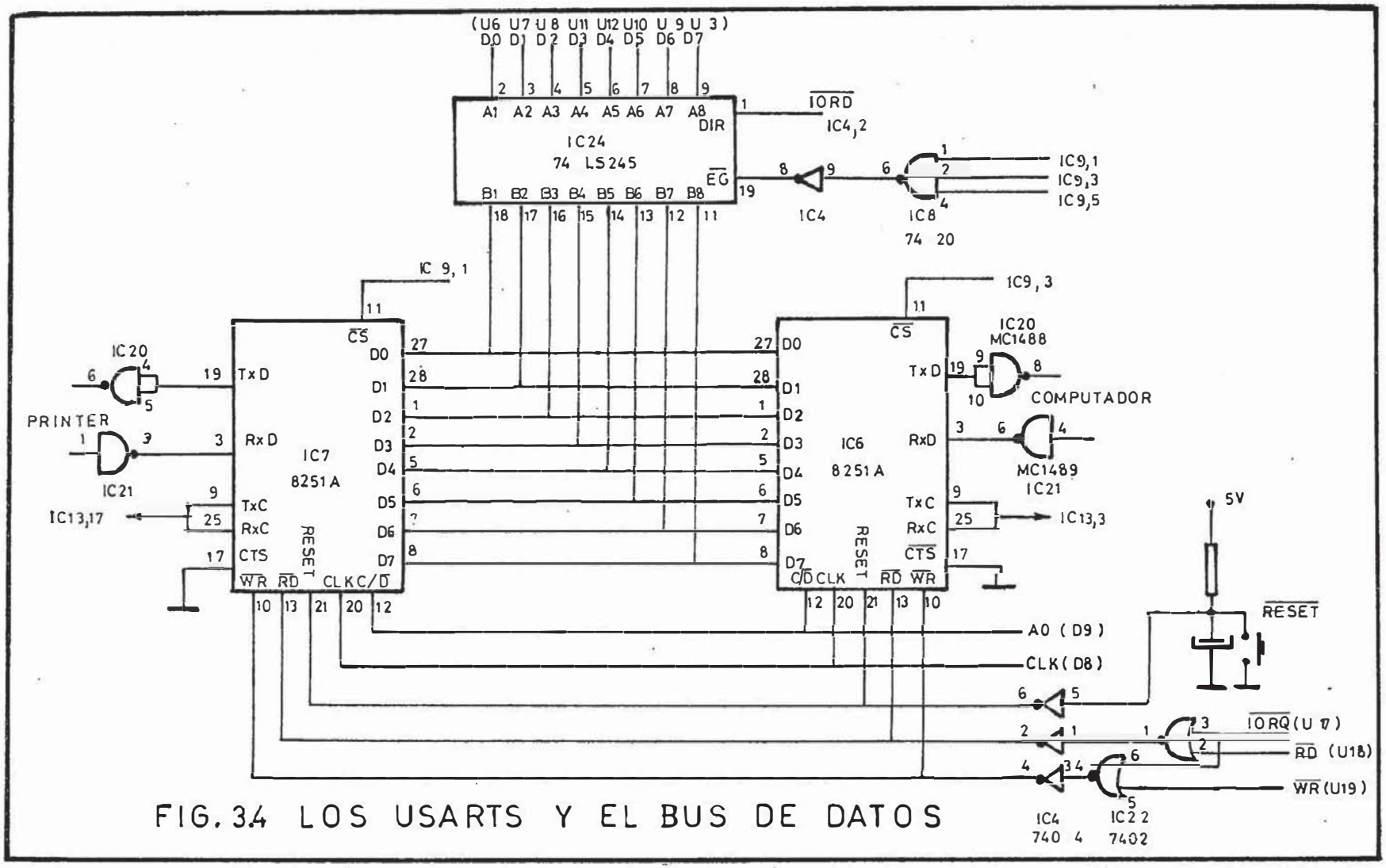


FIG. 3.4 LOS USARTS Y EL BUS DE DATOS

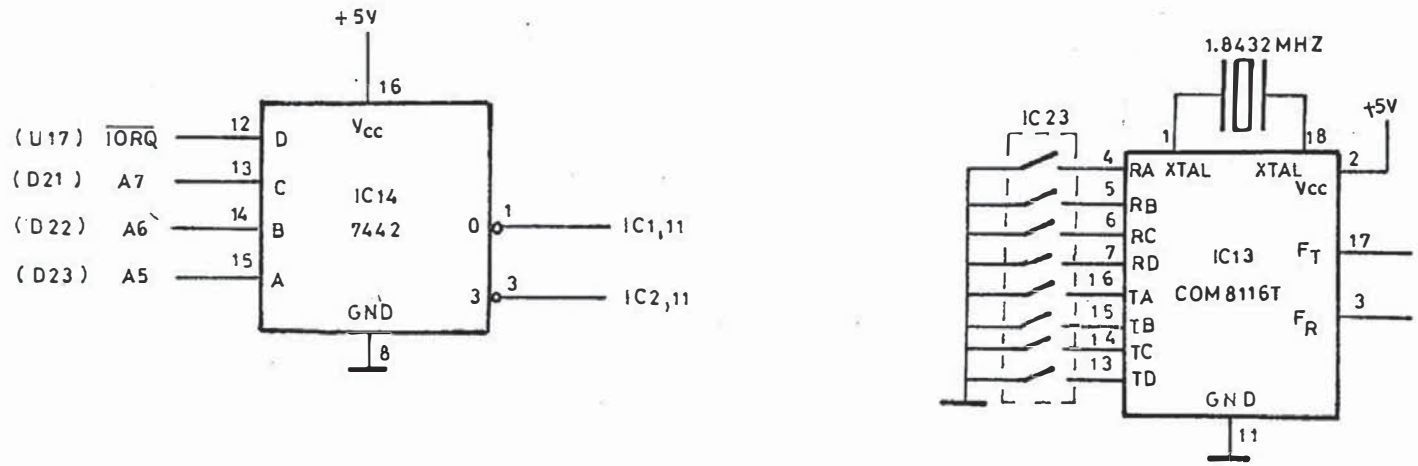


FIG. 3.5 GENERADOR DE BAUDIOS Y DECODIFICADOR DE I/O

con la diferencia que el CPU, la ROM y la RAM están dentro del microcomputador SPECTRUM.

En la figura 3.3 se especifica cada pin del conector con la nomenclatura utilizada para las otras figuras, por ejemplo U17 significa el pin 17 del lado superior del conector.

De la figura 3.4 podemos notar que el 74LS245 (DATA BUS DRIVER) es utilizado solamente para los USARTS y no para todo el sistema como en el diseño original, esto obliga a una pequeña variación en las señales de control DIR y $\overline{EG}$.

La conexión de los 8251 sigue igual, así también las señales $\overline{RD}$ y $\overline{WR}$ para ellos, pero el RESET aquí es independiente del microcomputador, porque en muchas ocasiones durante las pruebas se requería resetear los USARTS y no al microcomputador (ya que si se hace esto los programas se borraban de la memoria).

El resto de la circuitería permanece inalterado.

3.5 Pruebas y resultados

Con el circuito de las figuras 3.3.3.4 y 3.5 se hicieron una serie de pruebas, cuyos resultados describimos a continuación:

-Archivo de 9942 caracteres: AAAA....AAAA
BBBB....BBBB

Con 80 caracteres por fila.

Tiempo de impresión: 4 minutos.

Velocidad media: 41 caracteres/segundo.

-Archivo de 12502 caracteres: AAAA....AAAA
BBBB....BBBB

Con 132 caracteres por fila.
Tiempo de impresión: 5.09 minutos.
Velocidad media: 40 caracteres/segundo.

-Archivo de 1900 caracteres: WWWW....WWWW
WWWW....WWWW

Con 180 caracteres por fila.
Tiempo de impresión: 42 segundos.
Velocidad media: 45 caracteres/segundo.

-Archivo de 8497 caracteres, tipo texto.
Tiempo de impresión: 3:30 minutos.
Velocidad media: 40 caracteres/segundo.

-Archivo de 16821 caracteres, tipo texto, específicamente se trató del listado de los programas de este diseño.
Tiempo de impresión: 8.27 minutos.
Velocidad media de impresión: 33 caracteres/seg.

Adicionalmente se midió el tiempo que toma la impresora en imprimir una línea de 132 caracteres y 180 caracteres consecutivos. El número máximo de caracteres por línea es de 164, los caracteres restantes se imprimen sobre el último caracter (el 164avo).

La fila de 132 caracteres se imprime en 2.79 segundos. Esto implica una velocidad media de 50 caracteres/seg. Tiempo de retroceso del carro: aproximadamente 0.3 seg.

La fila de 180 caracteres toma 3.79 segundos, lo que da una velocidad media de 47 caracteres/segundo. El tiempo de retroceso es el mismo.

Con los resultados mencionados, llegamos a la siguiente conclusión:

-Velocidad media para archivos tipo texto: 33 caracteres/segundo.
-Velocidad media para archivos "patrones": 40 caracteres/segundo.

Comparados con las velocidades obtenidas a 300 bps, según acápite 1.5:

-Velocidad media para archivos tipo texto: 22 caracteres/segundo.
-Velocidad media para archivos"patrones" : 30 caracteres/segundo.

Obtenemos una mejora en la performance de la impresora de al menos 50% en archivos tipo texto y de 33% en archivos "patrones".

Durante las pruebas se comprobó que la velocidad de impresión depende mucho del tipo de archivo; la impresión es más lenta para aquellos archivos que contienen muchos espacios en blanco y saltos entre líneas.

De paso notamos que la impresora se aproxima a la máxima velocidad de impresión de 55 caracteres por segundo cuando procesa una línea ininterrumpida de caracteres que no contiene es-

pacios en blanco, y está recibiendo información a una razón de 1200 bps.

También se hicieron pruebas cambiando - la velocidad del USART conectado al computador, a 2400, 4800 y 9600 bps. Los resultados son satisfac torios. Se observa que los bursts que transmite el computador hacia el adaptador son más cortos en - duración por el natural incremento en la velocidad, que significa que el buffer se llena en menos tiem po. Lo que permanece constante es el lapso que - toma la printer en vaciar el buffer.

Por último se conectó el adaptador al computador a través de un par de modems asíncronos a 1200 bps, modelo VII222 de Racal-Vadic, simulándose así un ambiente de teleproceso. El adaptador funciona sin inconvenientes, manteniéndose las velocidades obtenidas en conexión directa.

3.6 Ilustraciones del proyecto

Presentamos una serie de fotografías - que dan una visión del proyecto.

La figura 3.6 corresponde a un computador MV-8000 con sus cinco gabinetes: el primero de la izquierda es el procesador, siguiéndole tres unidades de cinta y las unidades de disco.

La figura 3.7 es de una impresora QUME modelo - SPRINT 5 con teclado, sobre el cual trata este - proyecto.

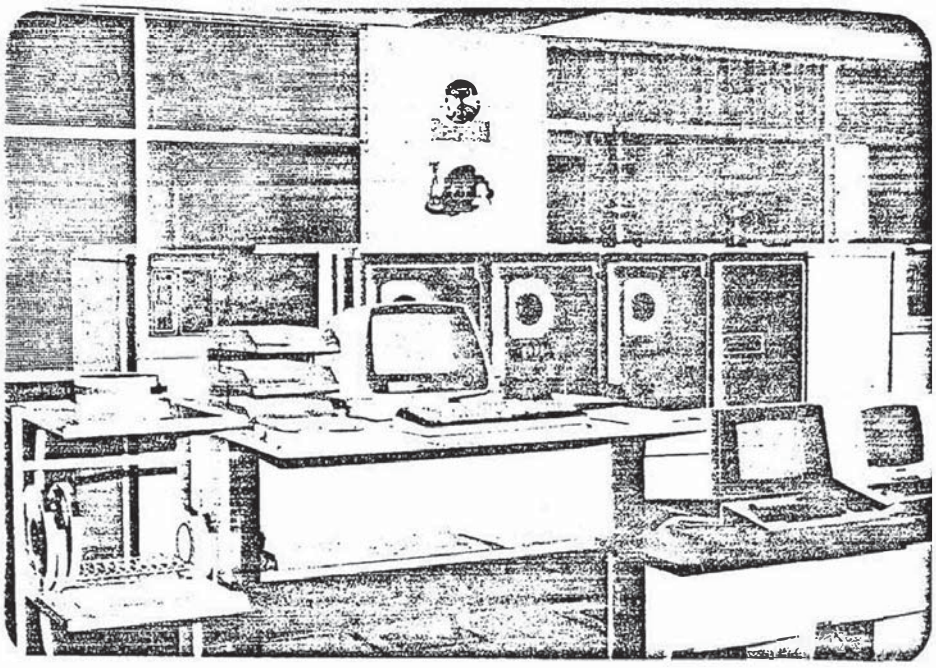


FIG. 3.6 COMPUTADOR DATA GENERAL MV 8000

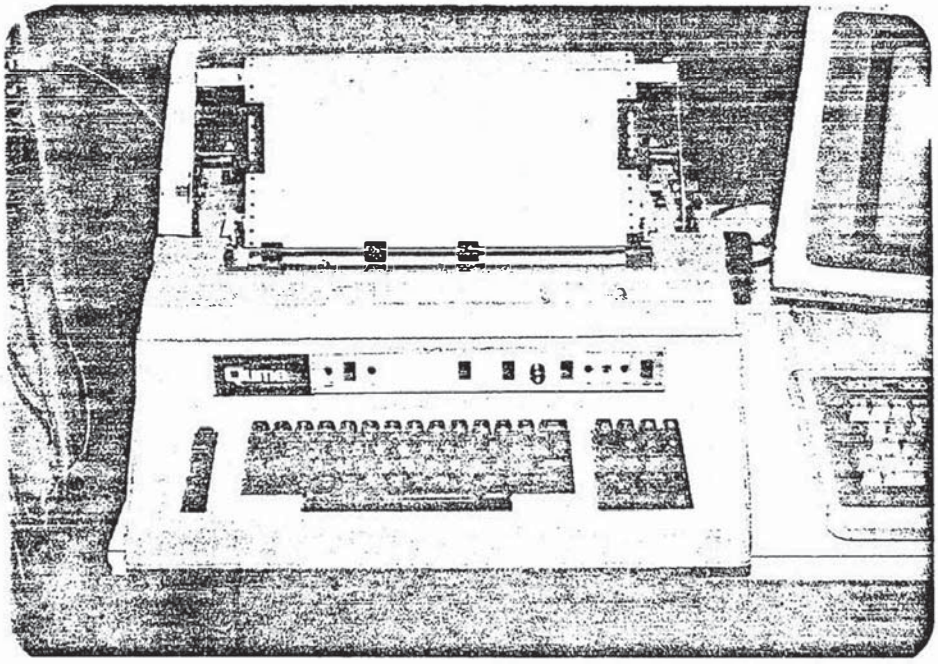


FIG. 3.7 IMPRESORA QUME SPRINT 5

La figura 3.8 es una fotografía del adaptador en - prototipo conectado al microcomputador SPECTRUM ; mientras que la figura 3.9 es del adaptador solamente, donde la distribución de los circuitos integrados va de acuerdo a la figura 3.2.

La figura 3.10 muestra los equipos auxiliares que se utilizaron en el proyecto, como el monitor de video donde se visualizan los programas, la fuente de alimentación separada; y el monitor RS-232C. Esporádicamente también se usaron: un osciloscopio, multímetro digital y una punta de prueba lógica.

La figura 3.11 es una fotografía de la pantalla de un terminal Visual V-300, que se usó para monitorar diferentes puntos. Este terminal tiene la particularidad de permitir la visualización de los - caracteres de control. En este caso se muestra la transmisión del computador, que corresponde a un archivo formado por líneas de un mismo caracter; - después de cada línea el computador envía los caracteres Cr y Lf(carriage return y line feed).

La figura 3.12 es la transmisión del adaptador, o la recepción de la impresora; se trata del mismo - archivo pero transmitido a través de un puerto programado para pantallas, por lo que solamente entrega Cr. Lo que hay que notar en esta fotografía es el caracter Etx (End of Transmission Block) que es apendizado por el adaptador.

La figura 3.13 es casi una réplica de la figura - 3.12 ya que la parte superior corresponde a la -

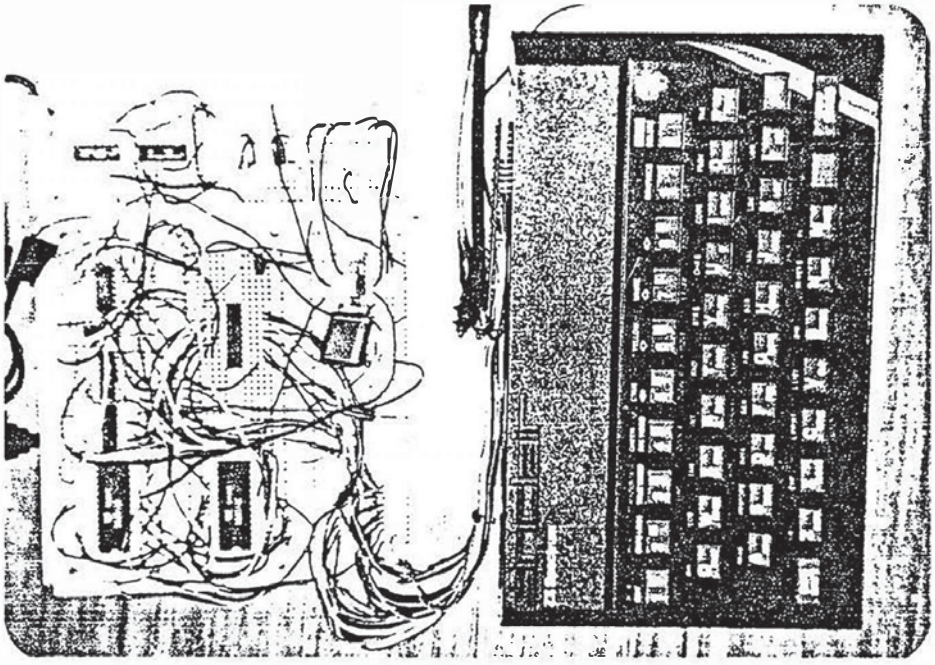


FIG. 3.8 PROTOTIPO Y MICROCOMPUTADOR SPECTRUM

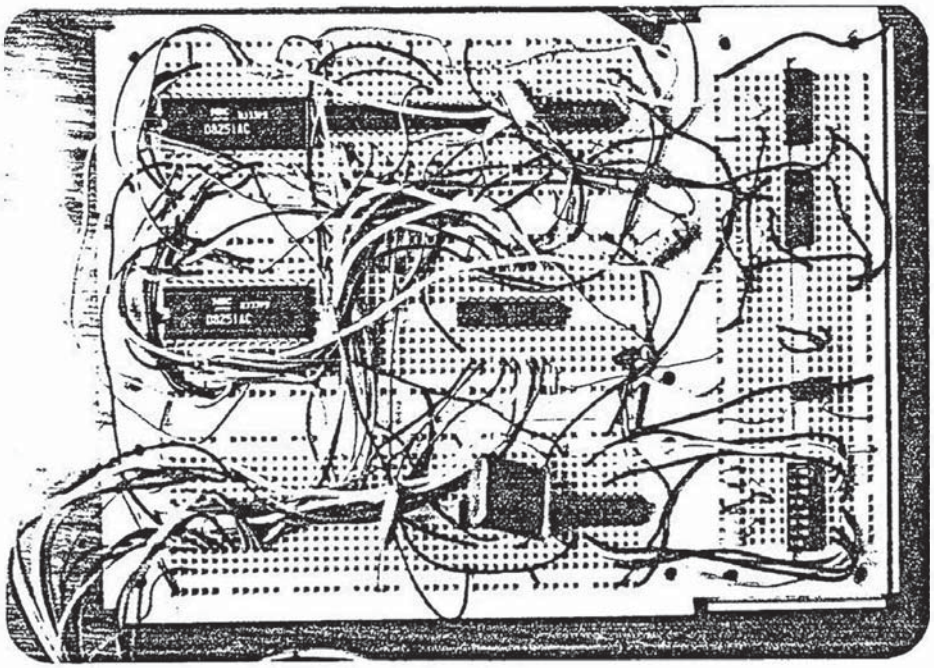


FIG. 3,9 PROTOTIPO

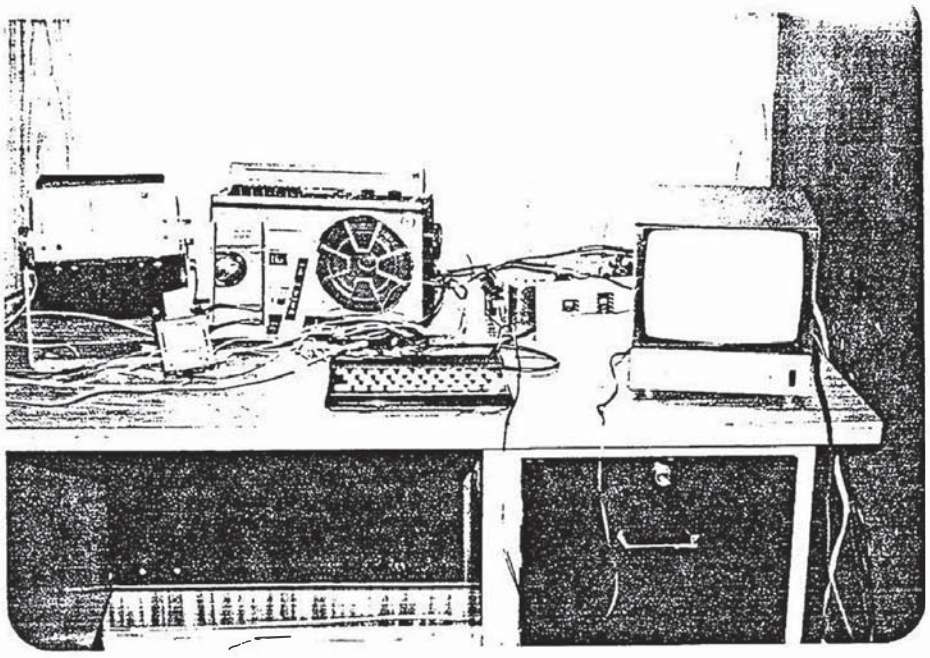


FIG. 3.10 EQUIPOS AUXILIARES

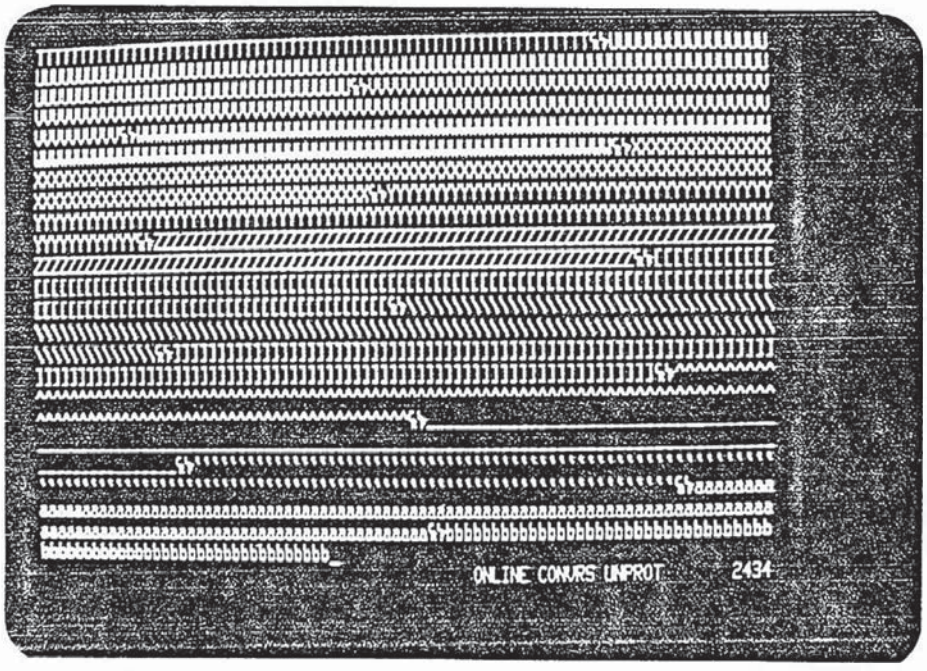


FIG. 3.11 TRANSMISION DEL COMPUTADOR

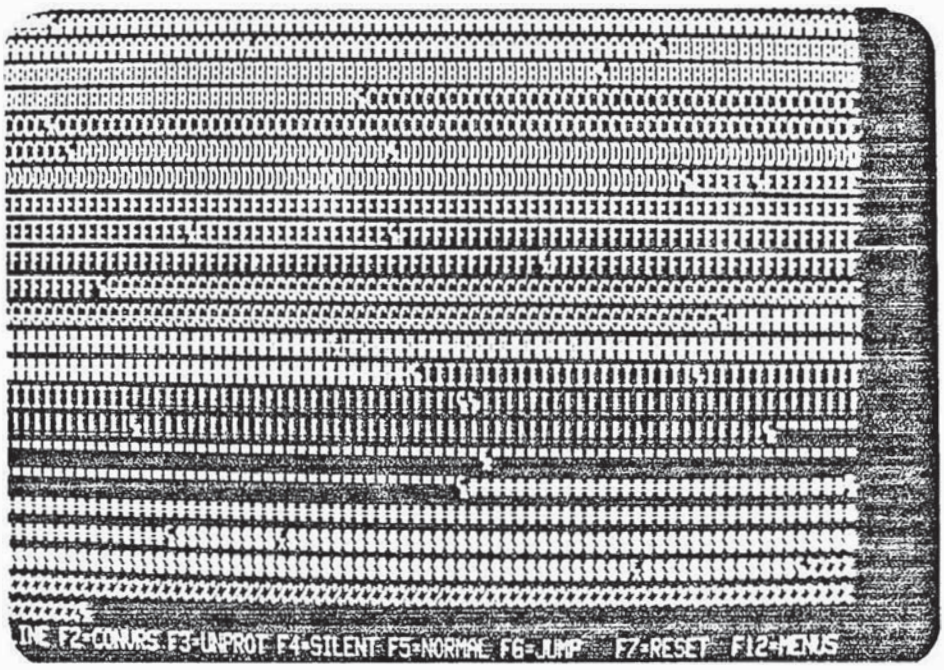


FIG. 3.12 TRANSMISION DEL ADAPTADOR

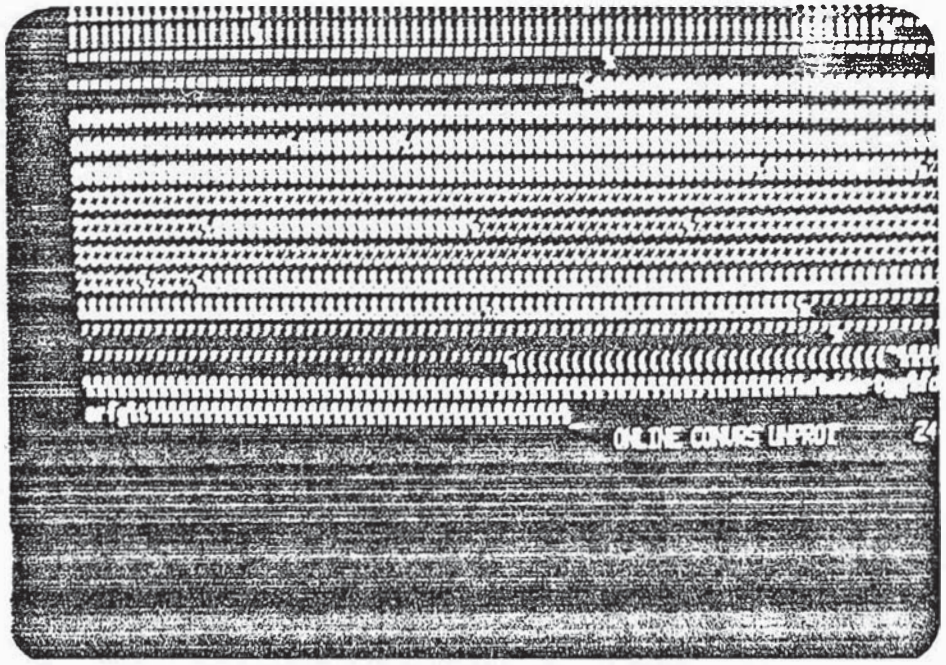


FIG 3.13 TRANSMISION Y RECEPCION DEL ADAPTADOR

transmisión del adaptador; excepto que las dos últimas líneas muestran la transmisión de la impresora. Se observa el caracter Ack(acknowledge), que es devuelto por la impresora por cada bloque que le envía el adaptador. Por la parte media se observa los caracteres D3 y D1(X-OFF y X-ON) que fue tipeado a través del teclado para detener la impresión y reasumirla (para acomodar el papel por ejemplo). Al final de la primera línea y al comienzo de la segunda, vemos otros caracteres tipeados desde teclado que no afectan el comportamiento del adaptador.

La figura 3.14 es lo que el adaptador le envía al computador; nuevamente tenemos D3 y D1; También hay otros caracteres que son ignorados por el computador.

La figura 3.15 muestra una vista de la pantalla del monitor de video, con el utilitario Zeus; el comando O dado inicialmente recupera el programa; Con L se listan las primeras 14 líneas del programa; se ensambla con A y se ejecuta con X. Inmediatamente siguen los mensajes de diagnóstico ROMOK y RAMOK. Luego tenemos una ráfaga de ERROR F indicando Framing Error, que se simuló enviándole al adaptador caracteres a una velocidad mitad de lo normal (1200 bps).

La figura 3.16 es otra vista del monitor, pero esta vez usando el utilitario Monitor48K. Se muestra las primeras 14 líneas del programa, con las posiciones de memoria, códigos de máquina y nemónicos.

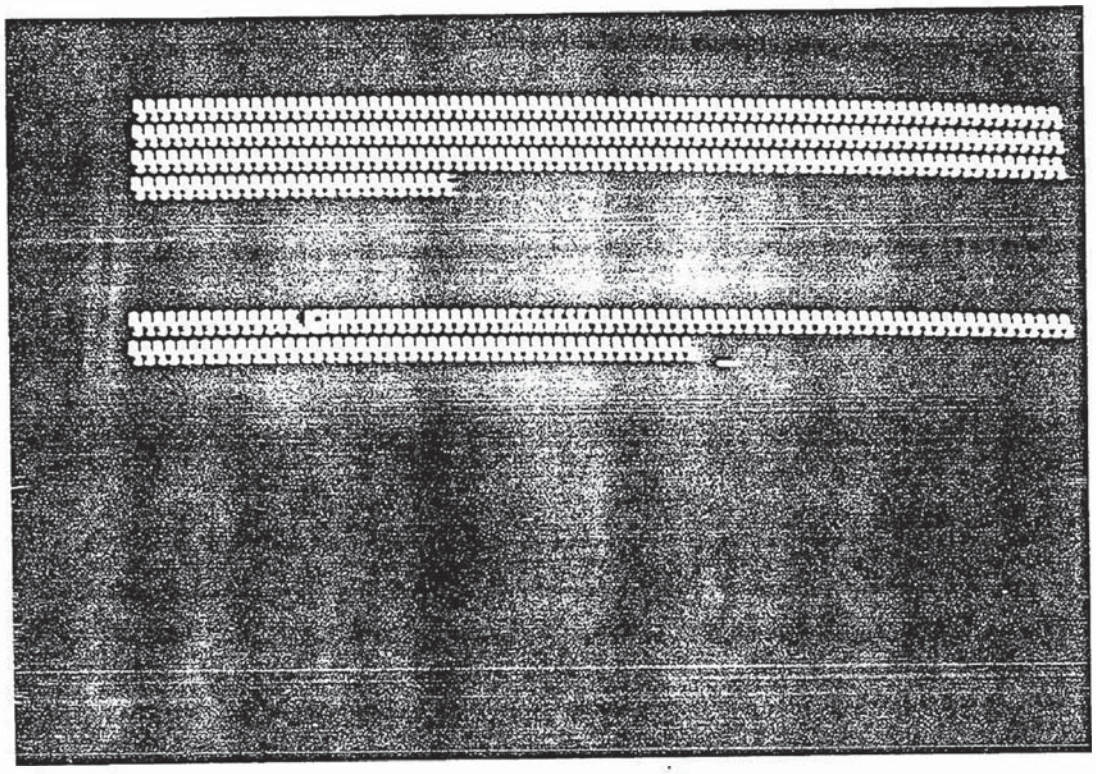


FIG. 3.14 RECEPCION DEL COMPUTADOR

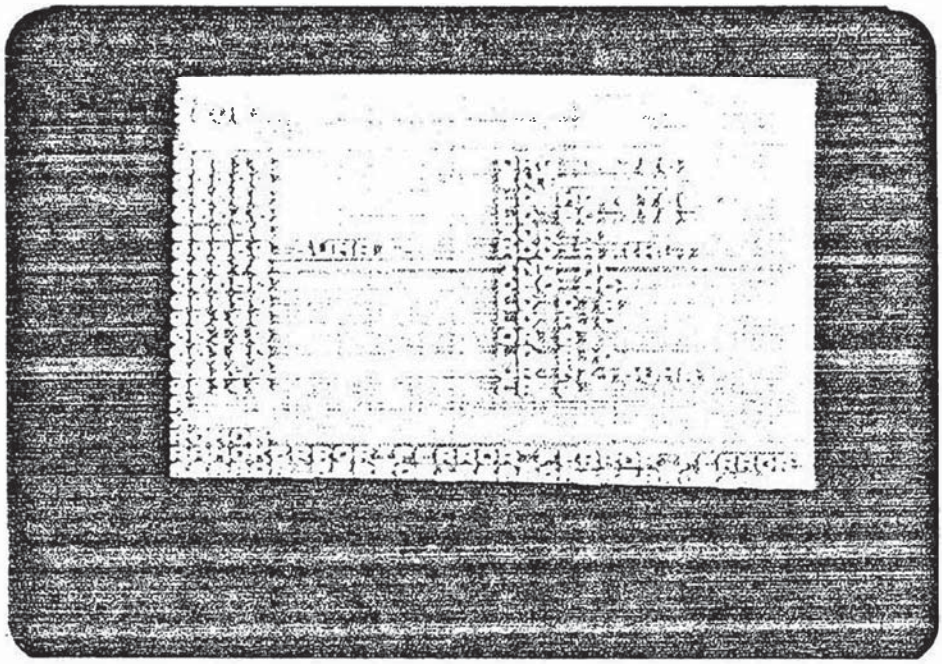


FIG. 3.15 USO DE UTILITARIO ZEUS

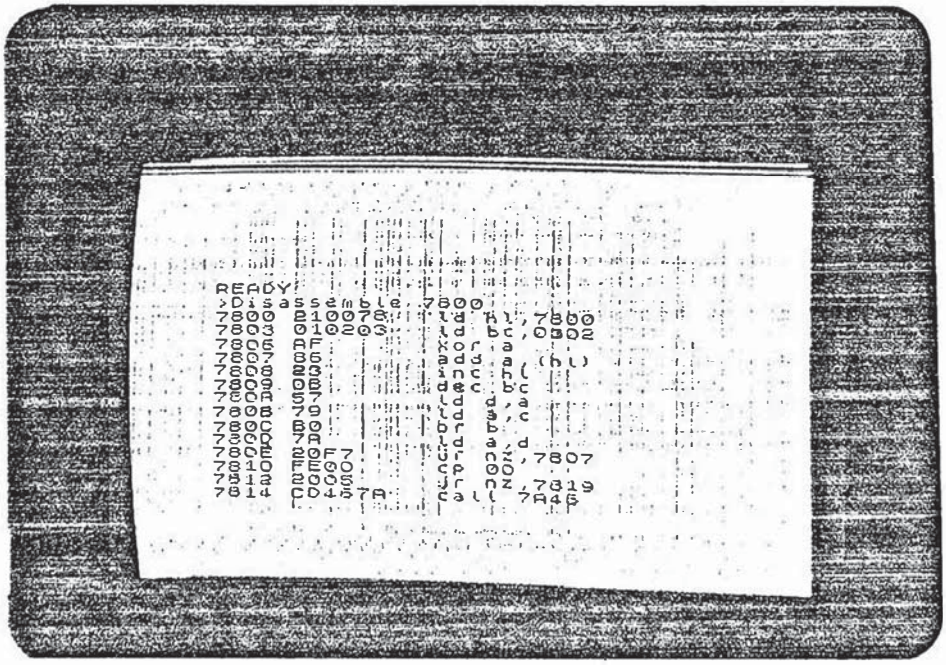


FIG. 3.16 USO DE UTILITARIO MONITOR48K

CONCLUSIONES

Se ha llegado hasta la construcción de un prototipo de laboratorio y se han hecho pruebas que conducen al mejoramiento del rendimiento de la impresora, alcanzándose mayores velocidades de impresión y reduciéndose por lo tanto el tiempo de impresión de los documentos.

El paso siguiente sería la construcción del equipo final, incluyendo el diseño del circuito impreso, el quemado de la EPROM con los programas descritos anteriormente.

De las dos alternativas posibles para implementar esté adaptador, se escogió el usar el empaquetado de la información en bloques de 110 caracteres apendizados por ETX. También hubiera sido posible utilizar la señal de control DTR.

Es de resaltar que este equipo es susceptible de mejoras tanto en software como en hardware. Por ejemplo podría lograrse la completa relocatabilidad de los programas, de manera que puedan ser llevados a cualquier parte de la memoria. Esto se haría cambiando instrucciones como JP M, LABEL por saltos relativos. Esto permitiría ensamblar los programas en una zona de memoria y correrlos en otra. También podemos minimizar circuitería, por ejemplo tratando los USARTS como posiciones de memoria, de este modo usaríamos un solo deco-

dificador 7442. Esta última opción no se usó durante la emulación ya que en el microcomputador Spectrum todas - las posiciones de memoria están ocupadas.

Sin embargo cualquier mejora como las mencionadas no produciría incrementos en la velocidad - de impresión de la impresora ya que ésta fue llevada a su límite.

La versatilidad del adaptador en cuanto a la velocidad con que puede trabajar con el computador queda comprobada, al poder conectarse a 1200, 2400, 4800 y 9600 bps.

Se ha logrado una transparencia casi - total, tanto para el computador como para la impresora, ya que el funcionamiento de ninguno de los dos se ha alterado. Inclusive el tiempo que el procesador de comunicaciones del computador dedica a la impresora disminuye, - ahora se comunica en bursts de 1200 bps u otra velocidad y no en forma continua a 300 bps.

El hardware usado en este diseño es - standard para aplicaciones de microprocesadores en ambiente de comunicaciones donde el uso de los USARTS es - extensivo. Es así que con una pequeña variación en los - circuitos, éstos podrían ser usados por ejemplo en:

-multiplexores estadísticos.
-compresores de datos.
-conversores de códigos.

En el caso de los multiplexores estadísticos, se asigna un USART a cada canal digital a multiplexar, y un USART trabajando en forma síncrona para -

la salida multiplexada. El uso de un microprocesador - permite la flexibilidad de sólo asignar ventanas de - tiempo a aquellos canales que están activos, aumentando así la eficiencia. El buffer del multiplexor se asigna también dinámicamente, de manera que en un instante de máximo tráfico, la suma de las velocidades de entrada - pueden exceder la velocidad de la salida multiplexada.

Para cada canal se asigna dinámicamente un área del buffer, normalmente una RAM, que manejada por el software, simula ser un FIFO; de aquí que - nuestro programa principal puede ser usado con algunas modificaciones.

Sólo faltaría agregar una rutina para la multiplicidad de punteros, ya que cada canal tendría su pareja de punteros de escritura y lectura; al atender a otro canal se cambia a un nuevo par de punteros, teniendo que guardarse los anteriores, para la oportunidad que se tenga que servir al mismo canal. También - se tiene que cuando el tráfico es muy pesado durante un instante largo, el multiplexor debe ser capaz de detener la transmisión de los canales para evitar la saturación del buffer, tal como ocurría con nuestro adaptador, que detiene al computador para evitar el overflow.

La aplicación de los compresores de - datos consiste en la utilización del microprocesador - para realizar la conversión de un alfabeto de longitud fija, digamos 7 bits ASCII, a otro alfabeto de longitud variable (menor en todo caso a la longitud inicial).

La codificación se basa en un estudio estadístico, en el que los caracteres que son usados -

más frecuentemente son representados con tres bits, los que se usan un poco menos con 4 bits y los pocos usados con 5 bits. De esta manera se logra una reducción en el tamaño del archivo a transmitir, que es la finalidad de la compresión, redundando en un menor tiempo de transmisión.

Este adaptador serviría para conectar la impresora QUME SPRINT5 a cualquier computador que tenga un port serial de comunicaciones con protocolo X-ON/X-OFF.

En el anexo H se muestra la lista de componentes y los costos aproximados.

El usuario operará este equipo en forma mínima y se podrá conectar localmente (la impresora está físicamente cerca del computador) o en una red de teleproceso (usando modems o multiplexores), pudiendo estar en cualquiera de los dos extremos, aunque preferentemente del lado de la printer.

Permitirá aumentar la velocidad de impresión, es decir si antes se imprimía a un máximo de 30 caracteres por segundo, ahora se hará a 55 caracteres por segundo.

Al ser ésta una aplicación específica y particular, es difícil encontrar en el mercado un equipo con las mismas funciones, aunque sí con algunas características similares, como por ejemplo el adaptador de comunicaciones programable BLACK BOX modelo J-CMF10 que convierte códigos X-ON/X-OFF a señales de control DTR y viceversa. Como otro ejemplo tenemos los

PRINTER SPOOLER que almacenan gran volumen de información a ser impresa aliviando algunas rutinas en el computador pero que no cumplen con el papel de adaptación requerida.

En cuanto al costo de nuestro adaptador se estima en alrededor de $100 mientras que el J-CMF10 supera los $500, siendo los PRINTER SPOOLER aún más costosos. Esta comparación es un poco forzada pues dichos equipos no cumplen las mismas funciones pero lo que se quiere dar a entender es que el costo del adaptador diseñado es significativamente menos.

Al resolver este problema en hardware estamos ahorrando memoria del computador, ya que una solución en software ocuparía espacio y crearía un proceso por cada impresora. La cantidad de memoria estimada para el programa que hiciese lo mismo que nuestro adaptador es de 4Kbytes.

BIBLIOGRAFIA

1) Build your own Z80 computer.
Steve Ciarcia

2) Z-80 Applicattions.
Coffron

3) How to generate and run AOS/VS on your ECLIPSE MV/Family Computer.
Data General

4) Microsystems Components Handbook.
Intel Corp.

5) Microprofessor MPF-1 User's Manual.
Multitech Industrial Corp.

6) Understanding Data Comunications.
Texas Instruments.

7) Z80 CPU Programmers Reference Guide.
Zilog Inc.

8) Zilog 1983/1984 Components Data Book.
Zilog Inc.