Universidad Nacional Autónoma de México

FACULTAD DE INGENIERIA

INTERCOMUNICACION ENTRE LAS COMPUTADORAS

VAX 11-780, BURROUGHS 6800 Y APPLE II PLUS

Tesis Profesional

Que para obtener el Título de

INGENIERO EN COMPUTACION

Presentan

LAURA SANDOVAL MONTAÑO
EDUARDO SALOMON JALLATH CORIA
HECTOR ARMANDO LOPEZ PINEDA

Director : Act. Sergio Castro Resines

México, D.F. 1985

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CAPITULO 1 ANTECEDENTES

1.1 INTRODUCCION 1-1
1.2 TECNICAS DE TRANSMISION 1-2
1.3 TIPOS DE LINEAS 1-2
1.4 CAPACIDADES DE CANAL 1-3
1.5 MODOS DE TRANSMISION 1-3
1.6 TECNICAS DE CONEXION (SWITCHING) 1-3
1.7 TRANSMISION DE DATOS 1-4
1.8 REDES DE COMUNICACION 1-4
1.9 FORMAS DE TRANSMISION 1-6
1.9.1 CODIGO USASCII 1-6
1.9.2 CODIGO EBCDIC 1-6
1.9.3 TRANSMISION DE DATOS 1-6
1.9.4 TRANSMISION ASINCRONA 1-7
1.9.5 TRANSMISION SINCRONA 1-7
1.9.6 DISCIPLINA DE LINEA 1-8
1.10 PROTOCOLOS 1-10
1.10.1 PROTOCOLO BISYNC 1-11
1.10.2 PROTOCOLO HDLC 1-12
1.11 TIPOS DE MODULACION 1-13
1.11.1 MODULACION POR FRECUENCIA 1-14
1.11.2 MODULACION POR FASE 1-14
1.12 HARDWARE DE COMUNICACIONES 1-15
1.12.1 TERMINALES 1-15
1.12.2 MODEMS/DATA-SETS 1-15
1.12.3 MULTIPLEXORES 1-15
1.12.4 CONCENTRADORES 1-16
1.12.5 PROCESADORES FRONTALES 1-16

CAPITULO 2 SISTEMA VAX 11/780

2.1 CARACTERISTICAS GENERALES 2-1
2.2 PROCESADOR 2-2
2.3 INTERFACES DE CONTROL DE ENTRADA Y SALIDA 2-3
2.3.1 MASSBUSS 2-3
2.3.2 UNIBUS 2-3
2.4 PERIFERICOS 2-4
2.4.1 TERMINALES 2-5
2.4.2 MULTIPLEXOR ASINCRONO DZ11 2-6
2.5 SISTEMA OPERATIVO 2-9

CAPITULO 3 CARACTERISTICAS DEL SISTEMA B6800

3.1 HARDWARE 3-1
3.2 SOFTWARE 3-1
3.3 HARDWARE DE COMUNICACIONES 3-3
3.4 PROTOCOLOS 3-4
3.5 SOFTWARE DE COMUNICACION DE DATOS 3-5
3.5.1 FLUJO DE MENSAJES 3-5
3.5.2 NETWORK DEFINITION LANGUAGE 3-6
3.5.3 MESSAGE CONTROL SYSTEM (MCS) 3-7

CAPITULO 4 CONEXION DEL SISTEMA VAX 11/780 AL SISTEMA B6800

4.1 PLANTEAMIENTO DEL PROBLEMA 4-1
4.2 SOLUCION DEL PROBLEMA 4-2
4.3 FORMA DE OPERACION 4-5
4.4 MODO ALTERNO DE SOLUCION 4-6

CAPITULO 5 SISTEMA APPLE PLUS II

5.1 CARACTERISTICAS GENERALES 5-1
5.2 MICROPROCESADOR 6502 5-1
5.3 SISTEMA OPERATIVO APPLE PASCAL 5-2
5.4 SISTEMA OPERATIVO DOS 3,3 5-2

CAPITULO 6 CONEXION DEL SISTEMA APPLE II PLUS A OTROS SISTEMAS

6.1 INTRODUCCION 6-1
6.2 ANTECEDENTES 6-1
6.3 DESARROLLO 6-3
6.3.1 CONDICIONES FISICAS 6-3
6.3.2 CONEXION FISICA 6-4
6.3.3 REALIZACION DEL PROTOCOLO DE COMUNICACION . . . 6-5
6.4 PROCESO DEL PROGRAMA DE COMUNICACION 6-7

CAPITULO 7 CONCLUSIONES

APENDICE A INTERFAZ EIA RS232-C

A.1 INTRODUCCION A-1
A.1.1 RESUMEN A-1
A.1.2 CONFIGURACION DE LA INTERFAZ A-2
A.1.3 VELOCIDADES DE TRANSMISION A-2
A.1.4 SEÑAL COMUN DE TIERRA A-2
A.1.5 COMUNICACION SINCRONA Y NO SINCRONA A-2
A.1.6 TIPOS DE SERVICIO A-2
A.1.7 TIPO DE FUNCIONES A-3
A.1.8 MODOS DE OPERACION A-3
A.2 CARACTERISTICAS ELECTRICAS DE LAS SEÑALES A-4
A.2.1 CIRCUITO EQUIVALENTE A-4
A.2.2 CONSIDERACIONES DE SEGURIDAD A-5
A.2.3 DEFINICION DE LOS ESTADOS DE LA SEÑAL EN LOS CIRCUITOS DE INTERCAMBIO A-5
A.2.4 IMPEDANCIA TERMINAL A-6
A.2.5 PROTECCION CONTRA FALLAS A-6
A.2.6 VOLTAJES Y CORRIENTES DE CONTROL A-6
A.2.7 REQUERIMIENTOS DE LA REGION DE TRANSICION . . . A-7
A.3 CARACTERISTICAS MECANICAS DE LA INTERFAZ A-8
A.3.1 DEFINICION DE INTERFAZ A-8
A.3.2 IDENTIFICACION DE PIN A-8
A.4 DESCRIPCION FUNCIONAL DE LOS CIRCUITOS DE INTERCOMUNICACION A-10

A.4.1 GENERALIDADES A-10
A.4.2 CATEGORIAS A-10
A.4.3 CARACTERISTICAS DE LAS SEÑALES, GENERALIDADES A-10
A.4.4 CIRCUITOS DE INTERCOMUNICACION A-12

APENDICE B LINE-DRIVER MICOM MICRO 400

B.1 INTRODUCCION B-1
B.2 TRANSMISION DE DATOS B-1
B.3 EXTENSION DE LA INTERFAZ EIA B-2
B.4 APLICACIONES B-2

APENDICE C PROGRAMAS FUENTE DE VAX

APENDICE D PROGRAMA FUENTE DE BURROUGHS

APENDICE E PROGRAMAS FUENTE DE APPLE

APENDICE F BIBLIOGRAFIA

CAPITULO 1

ANTECEDENTES

El hombre ha tenido a través de los tiempos la necesidad de comunicarse a lugares remotos, siempre tratando de que esa comunicación sea lo más rápida posible. En la antigüedad las comunicaciones se efectuaban a través del correo o viajando, por lo que se dependía de la velocidad del hombre. Posteriormente vendrían el telégrafo y el teléfono, hasta llegar a los modernos sistemas de microondas que existen hoy en día.

Conforme fue aumentando la población en el mundo, así también con la formación de industrias y grandes corporaciones en los negocios, la necesidad de una comunicación más rápida se hizo patente, y fue entonces cuando se empezaron a utilizar las líneas telefónicas para transmisión de datos, y eventualmente las nuevas tecnologías de comunicaciones están involucradas también en la transmisión de datos.

En comparación con otros periféricos, los dispositivos de comunicación de datos requieren un manejo especial debido a que intervienen otros factores, como son: el alto porcentaje de error, la interacción con el ser humano y el costo de las comunicaciones.

1.1 INTRODUCCION

En los sistemas de comunicación en línea están involucrados varios dispositivos de características diferentes, como son: terminales, modems, concentradores, multiplexores, procesadores frontales, adaptadores de línea, controladores de línea y procesadores centrales. Por esto es importante tener en cuenta varios conceptos, los cuales se describen a continuación.

Es conveniente aclarar que en este trabajo se han dejado

varios términos técnicos con su nombre original en el idioma inglés. El motivo de esta política son dos razones: la primera, evitar confusiones con alguna probable mala traducción y, la segunda, tratar de apegarse lo más posible a los términos que comúnmente se manejan.

1.2 TECNICAS DE TRANSMISION

Tenemos dos técnicas básicas de transmisión, la digital y la analógica.

La transmisión digital consiste en pulsos discretos separados; tiene la característica de efectuarse en conexión directa (sin Modem), pero está limitada por la distorsión a unos 300 metros, aunque se logran mayores distancias en transmisiones a bajas velocidades.

La transmisión analógica es la transmisión continua de una señal variable, producida por la modulación con ondas senoidales , técnica usada para la transmisión de voz. Este sistema requiere de Modems para la transmisión de datos pero permite una mayor distancia entre las conexiones ya que en este sistema es factible la corrección electrónica de la señal cuando ésta es afectada por el ruido o la atenuación.

1.3 TIPOS DE LINEAS

Podemos definir dos tipos de líneas: privada y pública.

La línea privada puede a su vez ser directa o rentada.

La directa tiene que ser dentro de la propiedad, es decir, no invadir territorios públicos, por lo que, generalmente, son cortas. La rentada es una línea del servicio público, dedicada a nuestras necesidades sin límite en su uso, por lo que podemos establecer las condiciones de transmisión que queremos.

La línea pública está ejemplificada por el sistema de teléfonos; en este sistema tenemos que "marcar" el destino y se requiere de modems o de acopladores acústicos para la transmisión de datos. Tiene una gran interferencia por el ruido de los centros de conmutación, pero tiene la ventaja de que pueden escogerse varios destinos. Su uso es más barato que la línea rentada en el caso de usarse poco tiempo durante el día.

1.4 CAPACIDADES DE CANAL

Cuando son utilizadas las líneas telefónicas para transmisión de datos, ya sea privada o pública, la velocidad de transferencia de información varía, dependiendo del tipo de modulación que se use, teniendo una limitación de 2400 bauds. La modulación es necesaria para evitar los problemas que producen el ruido, la atenuación y los retrasos de la señal.

1.5 MODOS DE TRANSMISION

Existen 3 modos básicos de transmisión:

SIMPLEX - Transmisión en una sola dirección sin posibilidad de regreso.

HALF-DUPLEX - Transmisión en cualquier sentido, pero no en ambos simultáneamente. Se utilizan generalmente dos hilos para la conexión.

FULL-DUPLEX - Transmisión simultanea independiente en ambas direcciones. Se utiliza generalmente con cuatro hilos para su conexión.

1.6 TECNICAS DE CONEXION (SWITCHING)

Existen 3 técnicas básicas de conexión: por circuito, por mensaje y por paquete.

La conexión por circuito se logra, primeramente, estableciendo la ruta que va a seguir el flujo de datos. El canal es dedicado enteramente durante el tiempo de conexión. Este sistema se usa en la red telefónica.

La conexión por mensaje consiste en enviar los mensajes a través de una ruta predeterminada del punto de origen al punto de destino. Los mensajes son almacenados temporalmente y después enviados a cada punto de relevo. La conexión por circuito no está hecha con anterioridad a cada mensaje, pero es usada conforme ésta se vuelve disponible en cada enlace.

La conexión por paquete es similar a la de mensajes, sólo

que en este caso los mensajes son partidos en bloques de 1000 bits. Esto trae como consecuencia que los mensajes deban ser reagrupados en el punto de destino. Este sistema tiene una alta integridad y disponibilidad.

1.7 TRANSMISION DE DATOS

En la mayoría de las redes de comunicación de datos el dato que deseamos nosotros transmitir es representado por canal telefónico.

La forma más general de comunicaciones actualmente usada es la línea telefónica, debido a su relativo bajo costo y al hecho de que ya existe la facilidad establecida.

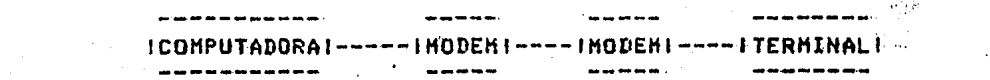
El uso de líneas telefónicas presenta un problema para la transmisión de datos digitales (ondas cuadradas) normalmente producidas por equipo electrónico de proceso de datos, porque dichas líneas fueron desarrolladas para la transmisión de la voz.

Las ondas sonoras, cuando son convertidas en energía eléctrica, se vuelven frecuencias que cambian continuamente (analógicas), las cuales son adecuadas para la transmisión sobre la red telefónica. Los datos digitales producidos por una computadora o terminal no pueden ser transmitidos directamente sobre la mayoría de las líneas telefónicas; y deben ser convertidos en frecuencias aceptables en los circuitos telefónicos.

Los DATA-SETS son dispositivos que efectúan el proceso de convertir las ondas cuadradas en señal analógica en frecuencias de banda de voz y la señal analógica en ondas cuadradas. Los DATA-SETS se conocen comunmente como MODEMS.

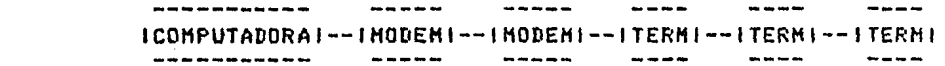
1.8 REDES DE COMUNICACION

Existen dos tipos principales de redes: punto a punto y multipunto. La siguiente figura ilustra la red punto a punto:

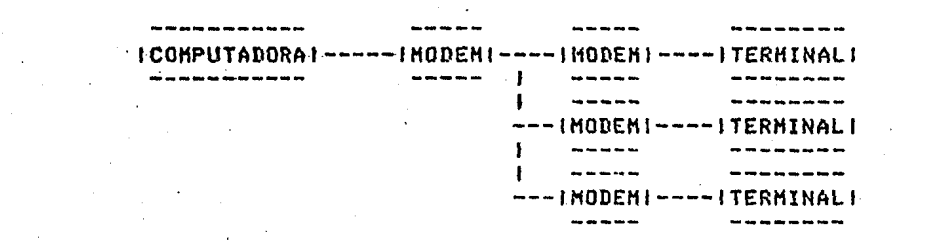


En la red punto a punto hay sólo dos estaciones en un determinado canal. Los datos son transmitidos y recibidos

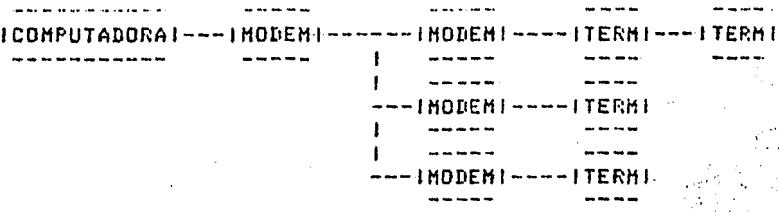
entre los dos. Mientras que en una red multipunto existen dos o más estaciones en un solo canal. En esta configuración, a una computadora le son conectadas un número determinado de terminales. La computadora controla la red, permitiendo sólo a una terminal transmitir a la vez y seleccionando cuál de las terminales debe recibir un mensaje especifico.



La figura anterior muestra una computadora y tres terminales conectadas via lineas telefónicas. Las tres terminales son conectadas en serie a un mismo modem. A esta forma se le llama concatenación. Todos los datos recibidos por el modem son pasados a la primera terminal, a través de ella a la segunda, y asi sucesivamente. A este tipo de red se le llama 'Concatenación Multi-punto'. La siguiente figura muestra la conexión de tres terminales a una computadora. En esta configuración cada terminal está conectada a un modem. Cuando un conjunto de éstos son conectados a un sólo canal telefónico, se le llama 'drop'. Estas conexiones se pueden efectuar en distancias cortas o largas entre los modems. Esta red recibe el nombre de 'multi-punto multi-drop'.



En redes de comunicaciones multi-punto muy grandes se usa tanto la concatenación como el 'multidrop'.



1.9 FORMAS DE TRANSMISION

Existen dos formas básicas de transmisión: síncrona y asíncrona. El formato y el método para el reconocimiento de caracteres en cada modo es diferente.

1.9.1 CODIGO USASCII

El "United States of America Standard Code for Information Interchange" (USASCII) es un código universal. El USASCII es un código de 7 bits, el cual tiene un rango de 128 símbolos o caracteres. Muchas computadoras usan USASCII como código interno. A este código se le conoce normalmente con el nombre de ASCII.

1.9.2 CODIGO EBCDIC

El código "Extended Binary Coded Decimal Interchange" (EBCDIC) utiliza un conjunto de 8 bits para la representación de caracteres. Este código es usado en sistemas de cómputo grandes, como IBM o Burroughs.

1.9.3 TRANSMISION DE DATOS

En la mayoría de las redes de comunicación de datos, el dato que nosotros deseamos transmitir está representado por los códigos ASCII ó EBCDIC. La combinación de unos y ceros, representando el caracter que es enviado, será transmitido bit por bit. En comunicación de datos, un uno es sinónimo de una "marca" y un cero es sinónimo de un "espacio". Cuando no se transmiten datos, las líneas de recepción y transmisión entre la terminal y el modem se mantienen siempre en la condición de "marca".

1.9.4 TRANSMISION ASINCRONA

Cuando se transmiten datos asincronamente, cada caracter es sincronizado individualmente. Esto se logra agregando a cada caracter un bit de inicio ('start bit') y un bit de fin ('stop bit').

El 'start bit' es siempre un espacio (cero) y el 'stop bit' es siempre una marca (uno). Cuando se usa el código ASCII se transmite también un bit de paridad con cada caracter, después del bit 7 de cada código. Esto hace un total de 10 bits por caracter. Cuando se usa el código EBCDIC no se transmite bit de paridad; por tanto, cada caracter está también formado de 10 bits. Esto se ilustra en la siguiente forma:

CARACTER	CODIGO ASCII	CODIGO TRANSMITIDO
	b1 b7	st b1 p stop
A	1000 001	0 1000 001 0 1
B	1100 001	0 1100 001 1 1

CARACTER	CODIGO EBCDIC	CODIGO TRANSMITIDO
	b1 b8	st b1 stop
A	1000 0011	0 1000 0011 1
B	1100 0011	0 1100 0011 1

La terminal receptora, una vez que identifica el 'start bit', supone que los ocho siguientes bits son un caracter válido. El siguiente bit transmitido debe ser un 'stop bit'. Cualquier desviación de esto produce un error en conjunto. En este tipo de transmisión el intervalo de tiempo que existe entre la tranmisión de los caracteres no es crítico. Sin embargo, en la práctica, cuando se transmiten caracteres en conjunto, el 'start bit' del siguiente caracter sigue al 'stop bit' del caracter inmediato anterior.

1.9.5 TRANSMISION SINCRONA

Cuando se transmite sincronamente, ya sea en código ASCII ó EBCDIC, el 'start bit' y el'stop bit' son eliminados y la paridad usada en ASCII es non. Esto da ocho bits por caracter, transmitiendo bit por bit y empezando con el bit uno. Como no existen ni 'start bit' ni 'stop bit' se usa otro método para sincronizar la terminal receptora, el cual consiste en poner cada mensaje precedido por tres o cuatro

caracteres de sincronía. La terminal receptora almacena cada bit recibido en un registro. Cuando un caracter de sincronía es detectado, entonces la terminal asume cada grupo de 8 bits como un caracter válido. Por tanto, una vez que se inició la transmisión de un mensaje, éste debe continuar sin interrupción el envío de los caracteres. Los caracteres de sincronía pueden ser usados como relleno si el siguiente dato no está listo para ser transmitido.

1.9.6 DISCIPLINA DE LINEA

En cualquier red de comunicación de datos, donde más de una estación es capaz de transmitir, se necesitan algunos tipos de disciplina de línea. El formato del control de secuencias y los mensajes de datos pueden variar de un fabricante a otro; sin embargo, todas las disciplinas de línea persiguen el mismo fin. Esto es, la disciplina de línea "controla" toda actividad, determinando qué terminal o terminales pueden transmitir o recibir en un tiempo determinado.

1.9.6.1 **POLL SEQUENCE** - El "poll sequence" es una secuencia de control, transmitida por el CPU, preguntado a una terminal especifica si está lista para recibir un mensaje. La terminal debe recibir su mensaje individual de "poll" antes de poder transmitir. Por ejemplo, en un sistema normal de Burroughs Multipunto, el "poll sequence" consiste siempre de 5 caracteres: EOT, AD1, AD2, POL, ENQ, transmitidos en ese orden (poll=p).

Cuando un caracter EOT es recibido por todas las terminales en línea, las pone en estado de recibir una secuencia de control. En ese instante todas las terminales esperan la recepción de los caracteres subsecuentes; el segundo caracter lo revisarán para determinar si coincide o no con su primera dirección programada (AD1); el tercer caracter recibido es revisado nuevamente con su dirección dos (AD2); el cuarto caracter de la secuencia es checado para determinar de qué tipo de cotrol se trata (en este caso "poll"). En este punto, todas las terminales, excepto aquella en la que coincidieron las dos direcciones AD1 y AD2 no harán nada hasta la recepción del siguiente EOT del CPU. La única terminal que identificó las direcciones AD1 y AD2 como suyas, revisará que el quinto caracter sea un ENQ y entonces responderá al "poll".

1.9.6.2 **RESPUESTA AL POLL** - La terminal tiene sólo dos respuestas válidas a su 'Poll'. Si no está lista para transmitir (la tecla de transmisión no ha sido oprimida) debe responder con un EOT, el programa continúa probablemente efectuando el 'Polling' de la siguiente terminal en la línea. Si la terminal está lista para transmitir y recibe su 'Poll', entonces responderá con la siguiente transmisión de caracteres: SOH, AD1, AD2, (XMN'S), STX, Texto, ETX y BCC. Los caracteres AD1 y AD2 corresponden a la dirección de la terminal. La transmisión de los números (XMN'S) es opcional; esto es de uno a tres caracteres y representan el número de mensajes de una terminal en particular o a una terminal en particular. El 'Block Check Character' (BCC) es una suma binaria sin acarreos de cada bit después del caracter SOH e incluyendo el caracter ETX (si el código de comunicación es ASCII, el BCC debe ser transmitido con su paridad correcta). En una red síncrona, cualquier caracter SYN usado como relleno en un mensaje no es calculado en el BCC.

Si el mensaje es recibido por el CPU sin error, entonces responderá con un ACK, la terminal responderá con un EOT y con esto se completa el mensaje. Si existe algún error en el mensaje, el CPU responderá con un NAK y la terminal retransmitirá todo el mensaje. Esto puede ocurrir un determinado número de veces y está determinado por el software del sistema. Después de alcanzado el límite de software el CPU transmitirá un EOT, dejando listas para recibir a todas las terminales y continuará con el programa.

1.9.6.3 **TIME OUT** - Después de hacer el 'Poll' a una terminal el CPU espera una respuesta. Si por cualquier razón la terminal no responde, el CPU la considera 'time-out'. Esto es un periodo controlado por software que el CPU esperará por una respuesta. Un periodo normal será de uno a tres segundos.

1.9.6.4 **SELECT MESSAGES** - Un 'select message' pregunta a una terminal si está lista para recibir un mensaje del centro de datos. Empieza con un EOT, seguido de la dirección de la terminal (AD1, AD2), seguido de un caracter de 'select' (sel=a), y terminado con un ENQ.

Una selección será reconocida por una terminal transmitiendo un caracter ACK si está lista para recibir. La selección de una terminal que no está lista para recibir provocará la transmisión de un caracter de NAK por la terminal. Esto producirá que el centro de datos envíe un EOT como caracter aislado o continuando su rutina de 'Poll' y/o 'select'. Los

mensajes de datos transmitidos por la terminal serán reconocidos por un caracter de ACK si fueron correctamente recibidos, o por un caracter de NAK si existió algún error.

La recepción en el centro de datos de un caracter ACK de una terminal, resultado de una recepción en la terminal de un mensaje correcto, producirá que el centro de datos restablezca el estado de control transmitiendo simplemente un caracter EOT o un EOT que sea el primer caracter de un "poll" o de un "select". Las otras terminales en la misma red serán ahora también afectadas por la rutina de "poll/select" que es reiniciada por el centro de datos. La recepción en el centro de datos de un caracter NAK de una terminal, resultado de la falla en la recepción en la terminal del mensaje transmitido después de la selección, provocará que el centro de datos retransmita el mensaje. La retransmisión producida por la recepción del caracter NAK se efectuará "n" veces hasta un límite determinado, superado éste provocará una transmisión de un EOT por el centro de datos, ya sea como un solo caracter o como parte de un "poll" o un "select".

La no recepción por el centro de datos de un ACK o un NAK de una terminal en un periodo especificado después de una selección producirá en el centro de datos un "time-out" y la transmisión de un caracter EOT como reselección o un nuevo "poll/select".

La falta de mensajes recibidos en la terminal del centro de datos, después de una selección, significará que la terminal transmitirá en ACK ó NAK. La ausencia de respuesta será detectada en el centro de datos y la terminal será reseleccionada.

1.10 PROTOCOLOS

Un protocolo de comunicación es una convención para la transmisión de datos; tiene las funciones de control, elaboración de formatos, sincronización y representación de datos. Los protocolos generalmente caen dentro de dos categorías:

1. Protocolos asíncronos: en donde los datos aparecen sucesivamente en el canal de comunicación en tiempos arbitrarios, sin que sean controlados por un reloj, pero sí gobernando los retardos relativos que se presentan entre dato y dato.

2. Protocolos síncronos: en donde cada dato está gobernado por un reloj maestro y aparece en un intervalo de tiempo específico.

Los protocolos asíncronos tratan cada caracter como un mensaje y aparecen arbitrariamente en el canal de comunicación; los bits de cada caracter; sin embargo, se transmiten a una velocidad fija; por lo tanto estos protocolos son síncronos por caracter y asíncronos entre caracteres, pero se llaman asíncronos debido a esta última característica. Los protocolos asíncronos tienden a ser más simples, más lentos y más ampliamente utilizados que los protocolos síncronos. A pesar de que su simplicidad es su atracción, los protocolos asíncronos tienen la desventaja de que en cada caracter transmitido emplea información de control, y esto hace que aumente el tamaño y en consecuencia disminuya la velocidad de transmisión. Esto no ocurre en los protocolos síncronos debido a que la información de control no se requiere por cada caracter.

Ejemplos de tres protocolos síncronos son:

1. BISYNC (Binary Synchronous Communications), un protocolo antiguo y obsoleto usado en equipos IBM.

2. DDCMP (Digital Data Communications Message Protocol), un protocolo usado principalmente en equipo DEC y

3. HDLC (High-level Data Link Control), un protocolo usado en muchos equipos síncronos nuevos con funciones aplicadas a la industria.

1.10.1 PROTOCOLO BISYNC

El protocolo BISYNC cuenta con uno o dos caracteres de sincronía para identificar la sincronización de caracter y de bloque. La idea es que el receptor establezca sincronización, buscando el patrón especial de sincronía, inspeccionando cada bit que aparezca en la interfaz. Supuestamente, antes de establecer la sincronización, el transmisor manda el patrón de sincronía continuamente, a la vez que monitorea su propio receptor para verificar si un patrón de sincronía es enviado sobre un canal de regreso. El envío constante de mensajes entre los dos transmisores confirma el uno al otro que están sincronizados.

SYNC	SOH	Encabezado	STX	Bloque	ETX	bits de chequeo	SYNC	SOH

La figura anterior es una estructura clásica de mensaje empleada por el protocolo BISYNC. Esta estructura se utiliza ampliamente por convención. La figura muestra que inmediatamente después de los caracteres de sincronía está un encabezado que inicia con un SOH (inicio de encabezado). El SOH y todos los demás caracteres que componen el encabezado son caracteres ASCII de control. Este encabezado se usa para propósitos de control y pueden contener una dirección fuente y otra destino, y un número de secuencia. En caso de haber error en la recepción de un bloque de datos, el receptor transmite un mensaje de regreso a la dirección fuente con el número de secuencia. La fuente entonces retransmite el mensaje. Debido a que cada mensaje tiene un número de secuencia, el receptor reconoce cuando un mensaje ha sido retransmitido. El bloque de datos comienza con un STX (inicio de texto) y finaliza con un ETX (fin de texto) o ETB (fin de transmisión de bloque), e inmediatamente después le siguen unos bits de verificación. Cada caracter del bloque puede también ser verificado por un bit de paridad si los caracteres están en código ASCII.

Con esta idea básica, los protocolos BISYNC parecen ser simples; sin embargo, hay problemas que complican al protocolo. La estructura del protocolo coloca un ETX y un EOB, si el bloque de datos (o información de control) contiene cualquiera de esos caracteres entre sus datos, los caracteres pueden ser mal interpretados.

En resumen, existen dos características independientes e importantes del protocolo BISYNC:

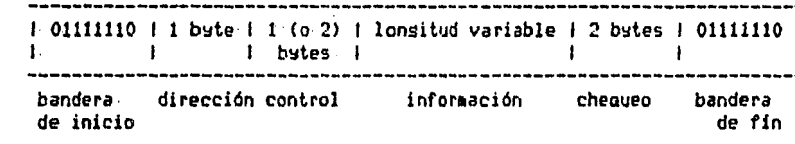
1. BISYNC depende de un patrón de sincronía para establecer sincronizacion de caracter.

2. BISYNC utiliza un protocolo de transmisión por bloque.

1.10.2 PROTOCOLO HDLC

El protocolo HDLC, como el BISYNC, usa un patrón especial para mantener la sincronización de caracter. Sin embargo, este patrón es único y nunca puede ocurrir en otro lugar que no sea al inicio de un bloque (el cual es también el inicio de un caracter). Por tanto, el receptor mantiene una búsqueda constante en el flujo de entrada, y automáticamente resincroniza al patrón siempre que éste aparezca. El patrón es la secuencia de bits 01111110 y es nombrada como "bandera"; esta bandera contiene seis unos consecutivos y no existe ningún dato que contenga esta secuencia. Para asegurar que la secuencia de sincronía no sea transmitida

como un dato, el transmisor emite cinco unos consecutivos (esto automáticamente inserta un falso 0)y posteriormente envía los bits que representan a los datos. El receptor revoca este proceso no tomando en cuenta el "0" que aparece después de una secuencia de cinco unos. No afecta la transmisión el estar insertando y borrando "ceros"; esto garantiza que el patrón de sincronía funcione como debe ser y ocurra la comunicación. El flujo de datos que entra al transmisor puede contener el patrón, y el receptor produce este patrón en su salida si el patrón está en el flujo de datos de entrada al transmisor.



La figura anterior muestra la estructura de un mensaje utilizado por el protocolo HDLC. El formato es muy similar al del BISYNC en lo que se refiere a que la información se transmite por bloque, pero los campos restantes están codificados en forma diferente. HDLC evita el uso de caracteres especiales para limitar los campos y simplifica enormemente el problema de la transmisión de datos que deben contener estos caracteres en la cadena. Un bloque HDLC empieza y termina con un patrón de sincronía que es llamado "bandera" en la figura. La dirección y la información de control aparece inmediatamente después de la bandera. Como en el protocolo BISYNC, la dirección y la información de control contienen una dirección fuente y una destino, y un número de secuencia. Cuando el bloque termina con una bandera, no es necesario tener un símbolo especial diferente para cerrar un bloque. La longitud de bloque necesaria no debe ser un número fijo de bits, debido a que la bandera se toma como un fin de bloque sin importar en donde ocurra en relación al inicio de un caracter. Cuando la bandera es detectada, los dieciseis bits precedentes son tratados como bits de verificación para determinar si hubo un error en la transmisión. El siguiente bloque puede usar la bandera final como bandera inicial y entonces directamente procesa la información de dirección y control del siguiente mensaje.

1.11 TIPOS DE MODULACION

Existen 3 formas básicas de modulación: por amplitud (ASK), por fase (PSK) y por frecuencia (FSK); siendo más utilizadas las dos últimas. Las formas de recepción y transmisión son diferentes para cada tipo.

1.11.1 MODULACION POR FRECUENCIA

El dato que nosotros deseamos transmitir no es más que una serie de bits (unos y ceros) que representan caracteres. Por lo que el modem tiene como salida dos diferentes frecuencias, una que represente un uno ("marca") y otra que represente un cero ("espacio"). El modem recorre su salida de una frecuencia a otra dependiendo de los datos digitales que vengan de un CPU o una terminal. A la llegada, el modem convierte las frecuencias a unos o ceros y los envia a la terminal o CPU receptores. En este tipo de modems las frecuencias usadas normalmente son 1200 hz para un uno ("marca") y 2200 hz para un cero ("espacio"). La terminal o CPU controlan la velocidad de transmisión.

1.11.2 MODULACION POR FASE

En este tipo de modulación en lugar de cambiar frecuencias como en FSK, en la modulación por fase se cambia la fase de la señal analógica. El dato será representado por los grados de recorrido en la fase (el cambio del inicio de un ciclo al inicio del siguiente). En la práctica, si se tiene un recorrido de fase de cero grados el dato es un cero, si el recorrido es de 180 grados el dato es un uno. La mayoría de los moduladores que utilizan PSK tienen la posibilidad de demodular y modular más de dos fases. En un modem que utiliza 4 diferentes recorridos de fase cada dato contiene dos bits de información. Ejemplo:

RECORRIDO DE FASE	VALOR DEL BIT
45 grados	00
5 grados	01
225 grados	10
315 grados	11

Algunos modems pueden usar 8 diferentes recorridos de fase, por lo que cada señal puede ahora tener tres bits de información. Con esto se puede tener que a una misma velocidad de transmisión (bauds/seg) mayor cantidad de información (bits/seg). Ejemplo:

	BAUD/SEG	BIT/SEG
1 bit	1800	1800
2 bits	1800	3600
3 bits	1800	5400

Normalmente la velocidad a la cual es transmitido el dato es

controlada por el modem. Pero algunas computadoras grandes proporcionan la señal de reloj en lugar de usar la de los modems.

1.12 HARDWARE DE COMUNICACIONES

A continuación se describen brevemente algunos de los elementos de hardware que están involucrados en los sistemas de comunicaciones de los sistemas de proceso.

1.12.1 TERMINALES

Una terminal es un instrumento de entrada y salida de información. Sirve como interfaz entre el ser humano y los sistemas de proceso.

Puede o no ser atendida por la computadora.

Las terminales locales pueden conectarse directamente al canal de Entrada y Salida y pueden operar como un sistema periférico en línea.

Las terminales remotas se conectan a la computadora a través de los sistemas de comunicaciones de datos.

1.12.2 MODEMS/DATA-SETS

Un modem es un dispositivo el cual convierte la señal digital a señal analógica y viceversa. La palabra modem se forma con las primeras 3 letras de las palabras MODulador y DEModulador.

1.12.3 MULTIPLEXORES

El multiplexor es un dispositivo para compartir una línea en la cual se tienen bandas de frecuencia o fracciones de tiempo en una base previamente determinada. Existen dos tipos :

1. Multiplexor en tiempo.
2. Multiplexor en frecuencia.

1.12.4 CONCENTRADORES

Un concentrador es una máquina para compartir las líneas de comunicación donde algunos canales de entrada comparten dinámicamente un canal de salida. La diferencia básica entre un multiplexor y un concentrador es que este último utiliza la línea de salida conforme a la demanda de las líneas de entrada.

1.12.5 PROCESADORES FRONTALES

Un procesador frontal es un procesador de propósito especial que tienen algunos sistemas, el cual se encarga de efectuar todas las operaciones relacionadas con los sistemas externos de comunicaciones, como son: terminales, impresoras remotas, 'Remote Job Entry', multiplexores, etcétera.

CAPITULO 2

SISTEMA VAX 11/780

2.1 CARACTERISTICAS GENERALES

El Sistema VAX 11/780 es un sistema funcional, confiable, que da facilidades de programación que normalmente sólo se encuentran en sistemas muy grandes. La familia de procesadores VAX tiene una arquitectura de 32 bits basada en la familia de minicomputadoras de 16 bits PDP-11. Los modos de direccionamiento y estructuras de stack son similares a los de la PDP-11 , pero la VAX proporciona 32 bits para direccionamiento, por lo que se tiene un espacio para direccionar un programa de cuatro gigabytes. También se cuenta con aritmética de 32 bits y rutas de datos para dar mayor velocidad en el proceso y permitir la concurrencia.

El conjunto de instrucciones de longitud variable y la variedad de tipos de datos nos dan una alta eficiencia en el manejo de bits. El conjunto de intrucciones tiene implantadas en hardware varias instrucciones de lenguajes de alto nivel y funciones del sistema operativo.

El sistema VAX es un sistema multiusuario que puede ser usado tanto para desarrollo de programas como para la ejecución de sistemas de aplicación. Es un sistema que se maneja por prioridades y eventos; la prioridad asignada y las actividades del proceso en el sistema determinan el nivel de servicio necesario. Los procesos en tiempo real reciben el sevicio de acuerdo a su prioridad y disponibilidad para ejecutarse, mientras el sistema administra el tiempo de CPU y la distribución de memoria residente para los procesos de ejecución normal.

El sistema VAX proporciona mecanismos de protección tanto en hardware como en software, asegurando la integridad de los datos y la disponibilidad del sistema. Dicha integridad se verifica con diagnósticos que corren en linea y con su propia detección de errores.

El sistema VAX es flexible y extendible. El sistema

operativo de memoria virtual da la oportunidad al programador de escribir grandes programas que pueden ejecutarse en configuraciones de poca o mucha memoria física. El lenguaje de comandos permite fácilmente a los usuarios modificar o exteder su propio repertorio de comandos.

2.2 PROCESADOR

El procesador VAX ejecuta intrucciones de longitud variable en modo nativo e intrucciones en modo compatible PDP-11. El procesador VAX incluye: administración integral de memoria, 16 registros de 32 bits, 32 niveles de prioridad de interrupciones, una consola inteligente, un reloj programable de tiempo real, un reloj de hora y fecha, memoria cache y almacenamiento de los diagnósticos de control.

El conjunto de instrucciones de VAX proporciona 32 bits de direccionamiento, permitiendo al procesador direccionar un espacio virtual hasta de 4.3 gigabytes. El hardware de administración de memoria tiene registros de mapeo usados por el sistema operativo, protección de páginas para modos de acceso y un buffer de traducción de direcciones que elimina accesos extras a memoria durante la traducción de dirección virtual a dirección física.

El procesador VAX también provee 16 registros de 32 bits que pueden ser usados para almacenamiento temporal o como acumuladores, registros de índice y registros base. Existen cuatro registros de signisficado especial, ellos son: el "Program Counter" y tres registros que son usados para dar una facilidad extra a las llamadas de rutinas o procedimientos. El procesador ofrece una variedad de modos de direccionamiento que usan los registros generales para identificar localidades de los operandos de las instrucciones, incluyendo también un modo de direccionamiento indexado para una capacidad de postindexado.

El conjunto de instrucciones tiene una alta eficiencia en el uso de bits. Incluye decimales, cadenas de caracteres, intrucciones de punto flotante, enteros, booleanos e intrucciones de campos de bits, en cualquier bit en memoria. La ejecución de intrucciones de punto flotante se mejora a través del acelerador de punto flotante.

2.3 INTERFACES DE CONTROL DE ENTRADA Y SALIDA

Los periféricos pueden ser conectados al bus de interconexión de memoria en dos formas: a través del MASSBUS, para discos de alta velocidad y/o unidades de cinta o a través de UNIBUS para una gran variedad de dispositivos, como: impresoras, discos, lectoras, terminales y ligas de comunicación con otros procesadores.

2.3.1 MASSBUSS

La interfaz para los periféricos del MASSBUS es el adaptador de MASSBUS, que realiza funciones de control, selección y almacenamiento temporal de datos. En un sistema VAX 11/780 se pueden conectar hasta cuatro adaptadores de MASSBUS.

Cada adaptador de MASSBUS tiene su propio mapa de traducción de direcciones. Además tiene un buffer de almacenamiento de 32 bytes. Los datos son agrupados en palabras de 64 bits (más paridad) para hacer un uso eficiente del ancho de banda de la interconexión a memoria.

2.3.2 UNIBUS

El UNIBUS es un bus asincrono bidireccional al cual se le conectan todos los dispositivos que no sean discos de alta velocidad o unidades de cinta; entre éstos se encuentran los periféricos de DIGITAL o los propios de los usuarios. El UNIBUS se conecta al memory-interconnect a través del adaptador del UNIBUS. El adaptador del UNIBUS realiza la selección de prioridades de los dispositivos conectados. Se pueden conectar hasta cuatro adaptadores del UNIBUS en el memory-interconnect.

El adaptador del UNIBUS provee accesos al procesador VAX-11/780 a los registros de periféricos de los dispositivos de UNIBUS, los requerimientos de tranferencias de datos y requerimientos de interrupciones a sus equivalentes del memory-interconnect, y viceversa, también son otorgados por éste. El traductor de direcciones del adaptador del UNIBUS traduce direcciones de 18 bits a direcciones del memory-interconect de 30 bits. El mapeo proporciona acceso directo al sistema de memoria para dispositivos periféricos. En algunos casos sólo esto constituye la transferencia.

Cualquier transferencia de acceso directo a memoria es manejada por una sola ruta directa de datos (se tienen hasta quince rutas más con buffer para los dispositivos que no son

el procesador). Cada 8 o 16 bits transferidos requieren de una transferencia de 32 bits al memory-interconnect. La velocidad máxima de transferencia para la ruta directa de datos es 500,000 bytes por segundo.

El adaptador del UNIBUS permite interrupción de programas concurrentes, y tranferencias directas o con buffer. El "throughput" de la ruta directa de datos más las 15 rutas de datos con buffer es de 13.3 millones de bytes por segundo.

2.4 PERIFERICOS

El sistema VAX proporciona una alta eficiencia en el funcionamiento de dispositivos de almacenamiento masivo para el uso de datos en línea, los equipos de unidades de grabación para proceso de datos, las interfaces para terminales para usuarios interactivos, las interfaces para acceso directo a memoria para usuarios en tiempo real, y para una interfaz en línea para comunicación entre procesos.

Los sistemas de almacenamiento masivo proveen una gran capacidad y una alta eficiencia. Cada adaptador de MASSBUS puede soportar hasta ocho manejadores de disco o siete manejadores de disco y un controlador de cinta magnética. Además, en un sistema de UNIBUS se pueden conectar hasta ocho manejadores de disco. El sistema operativo VAX/VMS tiene la capacidad de efectuar varios seeks a la vez en configuraciones con varios manejadores de disco, realiza varias transferencias simultáneas de bloque para entrada y salida, y permite al usuario el control de posición y del bloqueo.

Las lectoras de tarjetas e impresoras pueden ser dispositivos de entrada y salida declarados "spool" y administrados por las colas que controla el operador. Las series de impresoras LP11 y LA11 son de mediana velocidad (600 líneas por minuto). En este sistema se pueden conectar hasta cuatro impresoras LP11 y hasta 16 impresoras LA11.

El DMC11 y el DMP11 son controladores seriales síncronos de comunicaciones de línea, los cuales permiten una conexión entre procesadores usando el protocolo DDCMP (DIGITAL DATA COMMUNICATIONS MESSAGE PROTOCOL).

Para una alta eficiencia en la comunicación entre procesos, el sistema VAX-11/780 ofrece un multipuerto de memoria (MA780) y una interfaz de canal de alta velocidad (DR780). El DR780 puede ser usado para dispositivos que requieran velocidades de tranferencia hasta de 6.67 megabytes por segundo.

Todos los equipos periféricos están integrados con el software del sistema y están apoyados por diagnósticos y detección de errores en línea. Cada unidad tiene su propio sistema de verificación y corrección de errores. El software previene fallas de energía y tiene además algoritmos de recuperación de errores.

2.4.1 TERMINALES

Se pueden conectar al sitema VAX terminales interactivas. El manejador de terminales del sistema operativo permite manejo full-duplex para terminales de video y terminales impresoras.

Los programas pueden controlar la operación de las terminales a través del manejador de terminales, que además soporta varios modos de operación especial para las terminales. Un programa puede habilitar o deshabilitar varios modos a través de llamadas a los System Services: por ejemplo: eco, secuencias de escape, sincronización, etcétera.

Las entradas de una terminal son siempre independientes de una salida concurrente. A esta capacidad se le llama TYPE-AHEAD. Los datos transmitidos por una terminal son retenidos en un buffer hasta que un programa encole su requerimiento de lectura. En ese instante el dato es transferido al buffer del programa y regresado a la terminal (para hacer el eco). Si en ese instante lo que se está efectuando es una lectura, el eco y la tranferencia de datos son inmediatas.

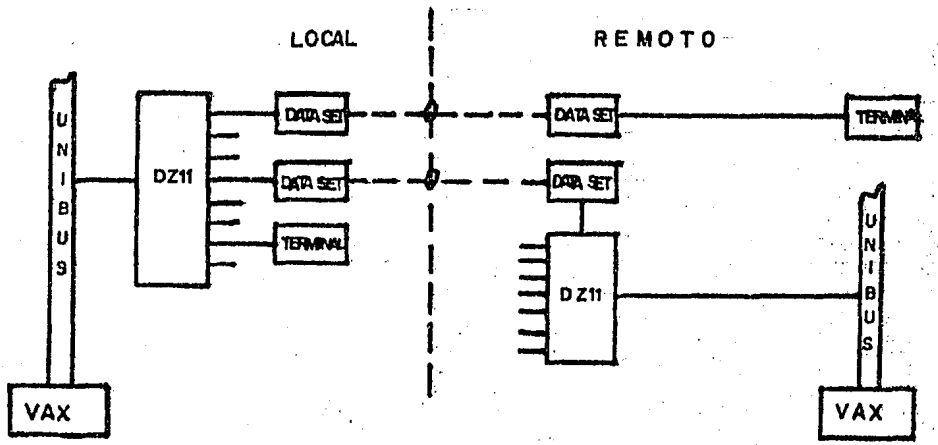
Una línea tecleada en una terminal es terminada por cualquier caracter especial; por ejemplo, la tecla RETURN. Un programa que lee de una terminal, puede opcionalmente especificar un terminador de línea especial para la lectura.

Las características de la terminal son establecidas inicialmente en la generación del sistema; sin embargo, existen comandos de operación de las terminales que permiten modificar en línea la característica de la terminal que está siendo usada. Por ejemplo, cambio en la velocidad de transmisión o cambio en el número de caracteres por línea de la terminal.

2.4.2 MULTIPLEXOR ASINCRONO DZ11

El DZ11 es un multiplexor asincrono que proporciona una interfaz entre el procesador VAX-11 y ocho líneas seriales asincronas. Puede ser usado en varias aplicaciones entre las que se encuentran: comunicación de procesos, tiempo compartido, transacciones y proceso en tiempo real. La operación local en terminales o computadoras es factible a velocidades hasta de 9600 bauds usando la interfaz EIA RS232-C o señalización con "loop" de corriente de zona. La operación remota, utilizando la red telefónica, es posible a través de las interfaces EIA RS232-C. Se proprociona también suficiente control sobre el modem para permitir operación de "DIAL-UP" (Auto respuesta) con modems de operación full-duplex como los modelos BELL 103 o 113 o equivalentes. (El control del DZ11 para el modem no soporta operación half-duplex y las señales secundarias de transmisión y recepción). También es posible la operación remota sobre líneas privadas para full-duplex punto a punto o full-duplex multipunto como estación de control.

La siguiente figura describe varias aplicaciones del DZ11.



El DZ11 tiene varias caracteristicas que permiten un control flexible de los parámetros, como son la velocidad de transmisión, longitud de caracter, número de "stop bits", tipo de paridad e interrupciones de la transmisión o

recepción. Las características adicionales incluyen el control limitado del modem, velocidad cero de recepción, almacenamiento de datos recibidos, módulo de entrada directa en los conectores SFC y líneas de 'turn-around'.

Cada módulo DZ11 proporciona ocho líneas seriales asincronas que hacen la interfaz a través de un panel de distribución de 16 líneas por lo que 2 módulos DZ11 pueden ser usados en un mismo panel.

2.4.2.1 **CONFIGURACIONES** - Existen seis diferentes configuraciones de DZ11 denominadas con una letra sufija (A-F). El DZ11-A y DZ11-B son opciones de dispositivos EIA con control parcial del modem. El DZ11-E es la combinación de DZ11-A y DZ11-B. El DZ11-C y DZ11-D son opciones con salida de 'loop' de corriente de 20 mA. El DZ11-F es la combinación de un DZ11-C y un DZ11-D.

2.4.2.2 **ESPECIFICACIONES GENERALES** -

- Modo de operación - FULL-DUPLEX
- Formato de Datos - asincrono, serie por bit, 1 'start bit' y 1, 1 1/2 ó 2 'stop bits' proporcionados por hardware bajo el control del programa.
- Tamaño de caracter - 5, 6, 7, u 8 BITS; seleccionable por programa (sin incluir bit de paridad).
- Paridad - puede ser: par, non o sin paridad.
- Velocidades de transmisión - 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200 y 9600.
- 'Breaks' - pueden ser generados y detectados en cada línea.
- 'Throughput' - 21,940 caracteres/segundo.

2.4.2.3 **SALIDAS** - En los DZ11-A, -D y -E cada línea proporciona niveles de voltaje y conectores del estandar EIA RS232-C. Las salidas apoyadas por esta opción son:

Pin 1 - tierra de protección

Pin 7 - tierra de la señal

Pin 2 - transmisión de datos

Pin 3 - recepción de datos

Pin 20 - datos de terminal disponible

Pin 22 - indicador de llamada

Pin 8 - detector de portadora

2.4.2.4 **ENTRADAS** - El UNIBUS es la entrada de toda interfaz DZ11.

2.4.2.5 **DISTANCIA** - La distancia recomendada para conectarse al DZ11 es 15 m a una velocidad de transmisión de 9600 bits/s con un cable BC05D o equivalente. Sin embargo, generalmente se puede trabajar a distancias mayores dependiendo del tipo de terminal, tipo de cable, velocidad de operación y el medio ambiente eléctrico. Comunicaciones a mayores distancias dependerán en mucho del ruido eléctrico existente. Por estas razones, DIGITAL no garantiza comunicación libre de error a una distancia mayor de 15 m. Sin embargo, las versiones EIA del DZ11 pueden ser conectadas a terminales locales a distancias mayores de 15 metros con resultados satisfactorios si la terminal está en el mismo edificio, en un medio ambiente como el de una oficina moderna. Si se usan cables trenzados con protección, la siguiente tabla puede servir como guía:

90m	-	9600 BAUD
300m	-	4800 BAUD
600m	-	2400 BAUD
900m	-	1200 BAUD
1500m	-	300 BAUD

2.5 SISTEMA OPERATIVO

VAX/VMS es un sistema operativo de propósito general para los sistemas VAX. Este sistema provee un medio ambiente confiable de alta funcionalidad para la ejecución concurrente multiusuario en tiempo compartido, batch, y aplicaciones de tiempo real.

El sistema operativo VAX/VMS ofrece:

- Administración de memoria virtual para la ejecución de grandes programas.
- Programación de prioridades por manejo de eventos.
- Memoria compartida, archivos, comunicación entre procesos con protección de datos basados en dueños y grupos de aplicación.
- System Services programados para control de procesos, subprocesos y comunicación entre ellos.

VAX/VMS tiene como características de administración de memoria: paginación, 'swapping', y protección y compartimiento para código y datos. La memoria es distribuida dinámicamente. Las aplicaciones pueden controlar la cantidad de memoria física asignada en la ejecución de un proceso, la protección de páginas y el 'swapping'. Estos controles pueden ser agregados después de la implantación de una aplicación.

VAX/VMS asigna el tiempo de CPU y memoria residente basado en prioridades. Por tanto, los procesos en tiempo real no tienen que competir con los procesos de baja prioridad para asignación de recursos. Los procesos con la misma prioridad se van rotando.

VAX/VMS permite aplicaciones en tiempo real para controlar la paginación de la memoria virtual y prioridad de ejecución. Las aplicaciones en tiempo real pueden eliminar recursos no utilizados para reducir la saturación del sistema. Los procesos en tiempo real no necesariamente tienen el privilegio de usar memoria protegida y/o estructuras de datos.

El sistema VAX/VMS tiene System Services para controlar procesos y la ejecución de los mismos, respuestas al control de tiempo real, control de asignación de recursos y obtención de información. Los System Services de control de procesos permiten la creación de subprocesos, así como de procesos independientes. Los procesos se pueden comunicar y sincronizar usando 'mailboxes', candados, areas compartidas de memoria, archivos compartidos o los 'Event Flag Clusters'

comunes.

Los diseñadores de aplicaciones pueden usar los mecanismos de protección y privilegio de VAX/VMS para implantar sistemas de seguridad y privacia especiales. El sistema VAX/VMS provee protección de acceso a memoria entre los procesos y dentro de los mismos. Cada proceso tiene su propio espacio de direccionamiento virtual independiente, que puede ser mapeado a páginas privadas o compartidas. Un proceso no puede utilizar una página privada de cualquier otro proceso. La protección de páginas compartidas de memoria y las facilidades para intercomunicación entre procesos, tales como 'mailboxes' y 'event flags', están basadas en los códigos de identificación de usuario (UIC) asignados individualmente a los programas y datos.

CAPITULO 3

CARACTERISTICAS DEL SISTEMA B6800

3.1 HARDWARE

La estructura básica del sistema B6800 consiste en una Unidad Central de Proceso (CPU), el Procesador de Entrada y Salida (IOP) y el Controlador de Memoria (MC). El CPU ejecuta instrucciones de programas, el IOP transfiere datos entre periféricos y memoria y el controlador de memoria maneja la transferencia de información entre memoria y CPU.

El controlador de memoria también provee una interfaz a memoria para dispositivos externos, como el procesador de comunicación de datos (DCP). Se pueden conectar hasta cuatro DCPs u otros dispositivos externos al controlador de memoria.

El procesador de entrada y salida maneja la transferencia de información entre periféricos y memoria; entre los periféricos se incluyen lectoras de tarjetas, impresoras de líneas, unidades de cinta magnética y unidades de disco.

El procesador central B6800 es una máquina de stack. Sus instrucciones son expresadas en sílabas de 8 bits y varía entre una y doce sílabas de longitud. El procesador central hace el fetch, el almacenamiento y las operaciones en palabras de 52 bits, formadas por 48 bits de datos, 3 bits de 'tag' para propósitos de control y un bit de paridad.

3.2 SOFTWARE

Todos los sistemas de software están escritos en Algol o en una extensión del mismo:

> Algol 60 con extensiones para entrada y salida, manejo de bits y caracteres. Todos los compiladores están escritos en Algol.

CARACTERISTICAS DEL SISTEMA B6800

El sistema operativo está escrito en NEWP, un lenguaje de alto nivel parecido a Algol.

El código objeto no puede ser modificado, por lo que pueden ser ejecutados concurrentemente dos o más veces, sin riesgo de modificar dicho código. Esto se logra a través del uso de los bits de 'tag'.

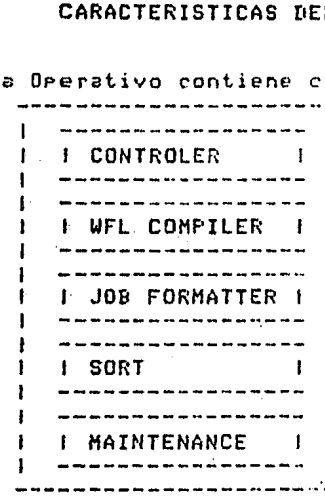
El sistema operativo se llama MCP (Master Control Program). Está compilado con NEWP y su código reside en disco. Los sistemas convencionales generalmente requieren un sistema operativo especial para cada configuración. Esto es, el sistema operativo depende de la configuración.

El MCP del sistema B6800 verifica el sistema de hardware para determinar la configuración. El MCP mantiene tablas de información, las cuales reflejan el estado actual de cada recurso de hadware, actualizando estas tablas con cada cambio de estado. Esto es hecho para recursos principales ('mainframe') y periféricos; por tanto, no es necesaria la generación del sistema, el MCP adapta automáticamente los cambios en el medio ambiente.

El MCP está formado de varias rutinas, las cuales manejan cosas como iniciación de todos los operandos de entrada y salida, manejo de memoria y administración de recursos. Sólo una pequeña parte del código (y las tablas) que manejan éstas y otras funciones residen en memoria; el resto del código del MCP está en disco y es pasado a memoria sólo cuando es necesitado. En este sistema no existen particiones fijas en memoria. Cada programa ocupará sólo la memoria necesaria para una ejecución eficiente, pero restringida por las otras demandas de memoria.

Las rutinas de entrada y salida son llamadas desde un programa de usuario del código del MCP cuando son requeridas. La parte del código del MCP cuando sea necesaria (por ejemplo: la lectora de la etiqueta de una cinta montada en una unidad) esto indica qué partes del MCP serán ejecutadas por los programas de usuario, mientras otras partes serán funciones independientes del sistema.

El Sistema Operativo contiene cinco módulos:



El CONTROLER es la interfaz entre el operador y el mundo exterior. El compilador WFL (WORK FLOW LANGUAGE) acepta tareas y crea conjuntos de tareas en un formato en disco listos para correr. El JOB-FORMATTER es un módulo usado para imprimir la salida de los trabajos. El SORT es un sistema para ordenar en disco, cinta y/o memoria, ligado dentro del MCP. MAINTENANCE es un sistema de pruebas en línea para ingenieros. Además en el MCP existen varias rutinas de utilería como las de administración de memoria.

3.3 HARDWARE DE COMUNICACIONES

En el flujo de información de una terminal al sistema central, los componentes del 'Data Communication Subsystem' manejan progresivamente grandes unidades de información.

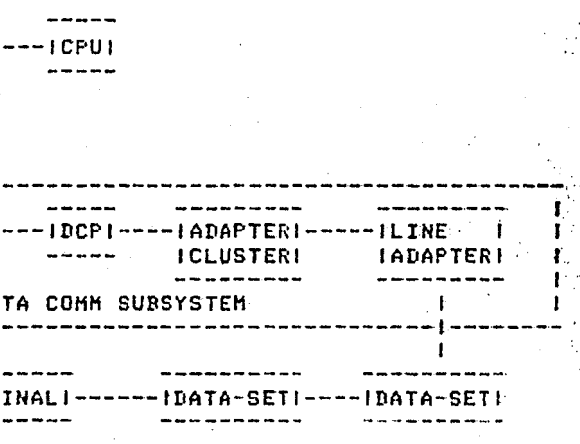
Cuando la información es enviada de una terminal al sistema central, la terminal transmite un caracter. Si la transmisión es en línea telefónica, los datos son modulados por un modem transmisor y demodulados en el modem receptor que está conectado a un adapter-cluster a través de un line-adapter.

El line-adapter compara las características eléctricas de la línea con las del adapter-cluster. El adapter-cluster acumula un caracter, le notifica al 'Data Communications Processor' (DCP) que el caracter está disponible, y transmite el caracter cuando el DCP lo requiere.

El DCP examina los caracteres de control asociados con cada transmisión para relacionarlo con su pertinente área en

memoria principal. Los caracteres del texto los acumula en palabras y pone cada palabra en el área apropiada de memoria principal. El DCP puede también verificar paridad y realizar traducción de caracteres; por ejemplo, de código ASCII a EBCDIC, o el inverso para los mensajes de salida.

El DCP puede ser también una interfaz directa al CPU para instrucciones de control. El DCP lee y escribe información a memoria principal a través del controlador de memoria.



Cada procesador B6800 puede manejar hasta 4 DCPs. Cada DCP puede tener hasta 16 adapter-clusters, a cada uno de los cuales se le pueden concectar hasta 16 line-adapters. Por tanto, a un solo procesador se le pueden conectar hasta 1,024 líneas, con varias terminales cada línea ("drop" o multipunto).

3.4 PROTOCOLOS

El sistema de comunicación de datos soporta la mayoría de los protocolos, entre los que se incluyen:

ASINCRONO

SINCRONO

BISYNC

BDLC/SDLC

3.5 SOFTWARE DE COMUNICACION DE DATOS

El software de comunicación de datos para la B6800 consiste de tres elementos física y funcionalmente separados:

1. NDL (Network Definition Language) para programar al DCP para el manejo físico de la línea.

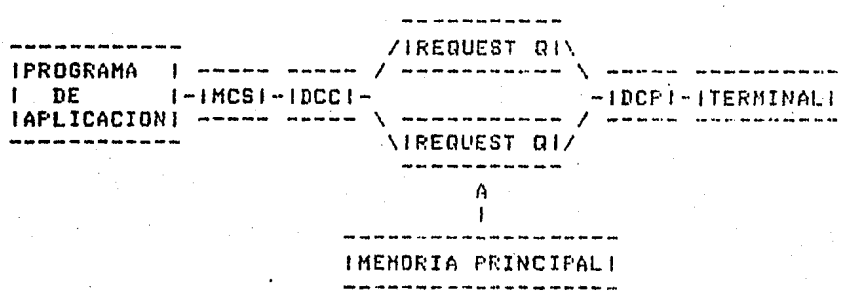
2. MCS (Message Control System) para manejar la distribución interna de los mensajes.

3. El programa de aplicación para tomar acciones basadas en el texto del mensaje

3.5.1 FLUJO DE MENSAJES

Los mensajes son manejados por caracteres por el DCP. El DCP ensambla caracteres dentro de palabras y agrega un 'Message Header' para la terminal originadora del mensaje. El DCP coloca los mensajes en memoria principal. El DCC (Data Communications Controler, que es parte del sistema operativo) examina los mensajes en la cola, inserta un apuntador en la cola apropiada del MCS que atiende esa terminal. El MCS analiza el encabezado y puede decidir pasar el mensaje a un programa de aplicación.

El siguiente diagrama describe el funcionamiento:



Algunas veces no es necesario que el MCS maneje el mensaje; en estos casos el MCS es ignorado; a este tipo de MCSs se

les llama no-participativos.

3.5.2 NETWORK DEFINITION LANGUAGE

El Network Definition Language (NDL) de la B6800 es un lenguaje descriptivo usado por los programadores de comunicaciones de datos para especificar la característica de una red. Cuando se tiene el programa compilado, se provee el código del DCP para la disciplina de línea requerida para los diferentes tipos de terminal (dispositivos de conexión, dispositivos de terminal, dispositivos poleados, etcétera); se describe también la red físicamente: las líneas dentro de la computadora, cómo están conectadas directamente, con modem, velocidad, etcétera; y qué estaciones están conectadas a cada línea.

Es conveniente hace notar la diferencia entre una estación lógica y una terminal física. Los programas de aplicación y los MCSs se refieren normalmente a estaciones mientras el DCP/NDL lo hacen a las terminales. Cada estación declarada en el NDL tiene una dirección física asociada con ella; también tiene una descripción de su conexión física, y una dirección lógica (Logical Station Number). Todo esto es asignado por el compilador de NDL. Los programas en NDL están organizados en ocho secciones:

1. La sección CONTROL permite el uso de una línea lógica a las estaciones asignadas a esa línea

2. La sección REQUEST contiene instrucciones de control para cada tipo de terminal. Indica también al DCP cómo marcar con caracteres especiales (SOH, start-of-header; ETX, end-of-text), cómo verificar paridad y qué acciones tomar con dichos caracteres especiales.

3. La sección MODEM define los tipos de data-set para cada tipo de línea. (síncrono, asíncrono, etcétera).

4. La sección TERMINAL define los tipos de terminales (por ejemplo: tamaño del buffer, screen, paridad, etcétera). También define qué data-set está asociado a cada terminal.

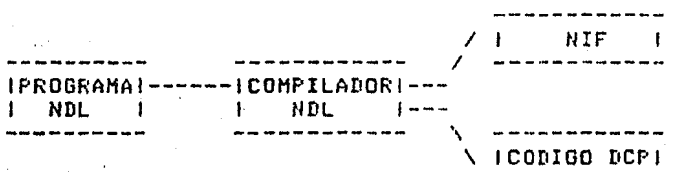
5. La sección STATION define cada uno de los puertos dentro del Data Comunication System (por ejemplo: nombre de la estación, tipo de terminal, entrada-salida disponible, nombre del MCS de control, caracter de control) para los mensajes de esa estación a su MCS respectivo.

6. La sección LINE define cada línea, localización física (DCP/cluster/adapter) y lista cada estación que se encuentra en la línea.

7. La sección DCP lista los DCPs que existen y el monto de memoria local para cada uno.

8. La sección FILE asocia las estaciones con nombres de archivos para el uso de programas de aplicación.

El compilador de NDL produce el código del DCP y un archivo de información de la red llamado "Network Information File" (NIF).



Varios conjuntos de NIF y códigos del DCP pueden existir en disco, pero sólo puede funcionar uno a la vez.

3.5.3 MESSAGE CONTROL SYSTEM (MCS)

Los MCSs son designados para proveer varios aspectos de control. Tienen la habilidad de:

- Controlar las estaciones declaradas a él (en NDL)
- Hacer que una estación pueda o no comunicarse.
- Aceptar entradas de una estación.
- Asignar o negar la entrada a una estación a un archivo.
- Conmutar mensajes.
- Reconfigurar la red en presencia de fallas.

Burroughs apoya varios MCSs para el sistema B6900 para manejar varios medios ambientes de comunicación comunes.

CANDE (Command AND Edit) es un MCS de tiempo compartido. Tiene la habilidad de crear archivos en disco, editarlos, compilar programas y ejecutarlos.

RJE (Remote Job Entry) da la capacidad de recibir trabajos de sistemas remotos en el sistema central y enviar su salida a dichos sistemas para su impresión.

DIAGNOSTIC MCS es usado para detectar problemas de líneas de comunicación de datos o DCPs. Se usa junto con un DLM (Data Line Monitor).

CAPITULO 4

CONEXION DEL SISTEMA VAX 11/780 AL SISTEMA B6800

4.1 PLANTEAMIENTO DEL PROBLEMA

El Centro de Cálculo de la Facultad de Ingeniería (CECAFI) desde su inicio ha sido usuario de los equipos Burroughs que ha tenido la Universidad, primero en instalaciones de lo que fue el Centro de Servicios de Computo (CSC), y actualmente en las que forman el Programa Universitario de Computo (PUC).

Este hecho motivó que el CECAFI desarrollara una gran cantidad de Sistemas Administrativos y Paquetes de Biblioteca en las computadoras de dichas instalaciones.

En el año de 1982 la Facultad de Ingeniería adquirió un equipo de cómputo marca DIGITAL. El equipo adquirido se instaló en el CECAFI y consistió en un sistema VAX 11/780, con la siguiente configuración:

1 Procesador central con opción de punto flotante y memoria cache de 8 K bytes.

2 Megabytes de memoria MOS.

1 Adaptador de Massbus.

1 Adaptador de Unibus.

1 Impresora de línea de 600 lpm.

1 Impresora graficadora (PLXY).

1 Sistema de consola.

3 Discos RP06 de 176 MB cada uno.

1 Unidad de cinta TU77.

3 Multiplexores asincronos de 8 puertos seriales DZ11.

20 Terminales VT100.

4 Terminales LA120.

Con la llegada de este equipo fue necesario el translado de algunos programas existentes en la computadora B7800 ubicada en el edificio del PUC, y los existentes en la computadora B6800 en las instalaciones de la Dirección de Cómputo para la Administración Académica (DCAA).

Por otro lado, algunos archivos, resultado de sistemas de control académico que se procesan en la VAX, es necesario transferirlos a la computadora B6800 de la DCAA, para entregarse a la Coordinación de la Administración Escolar (CAE), la cual utiliza esta computadora.

La transferencia de información entre ambas computadoras se venía realizando únicamente a través de cinta magnética. Teniendo en cuenta que existe únicamente, solamente una sola unidad de cinta en las instalaciones del CECAFI, este sistema era vulnerable en el momento de una falla en dicha unidad.

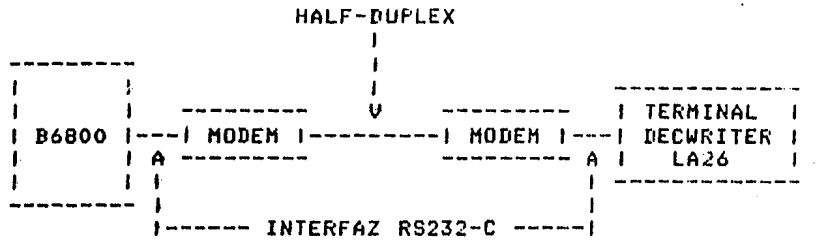
Por otro lado, el utilizar las cintas como único medio de traslado de la información requiere de mayor intervención humana en el trayecto de una instalación a otra, registro de las cintas, intervención de los operadores, etcétera; con lo que podemos concluir que este sistema no es del todo eficiente.

4.2 SOLUCION DEL PROBLEMA

Ante esta situación se planteó la necesidad de un sistema de comunicación rápido y flexible. El cual se contempló en una comunicación en línea entre las computadoras VAX 11/780 del CECAFI y los equipos Burroughs de la universidad, planteando, primeramente, la conexión con la computadora B6800 de la DCAA en el edificio del IIMAS.

La comunicación se llevaría a cabo a través de una línea telefónica entre ambas instalaciones. La línea utilizada ya existía y era usada para la comunicación de una terminal DECWRITER-LA26 instalada en el CECAFI, conectada con la computadora B6800. Este sistema de enlace existente se llevaba a cabo con una comunicación half-duplex entre dos modems Transdata MFX-1800, a una velocidad de transmisión de 300 bauds.

CONEXION DEL SISTEMA VAX 11/780 AL SISTEMA B6800



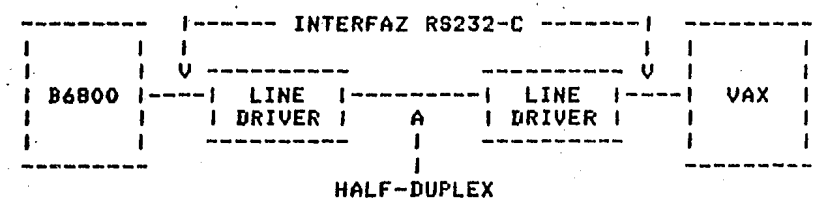
Esta instalación tenía dos problemas fundamentales:

1. La comunicación era HALF-DUPLEX, la cual requiere en los modems señales de control para transmisión y recepción y éstas no son apoyadas por la VAX.

2. Este tipo de modems limita la velocidad a un máximo de 1200 bauds, con lo que se requiere mucho tiempo de utilización de una línea en la transferencia de volúmenes grandes de información.

Para solucionar estos problemas se sustituyeron los modems existentes por unos line-drivers Micom Micro-401, los cuales no requieren señales de control para una operación HALF-DUPLEX y permiten un rango de velocidad de transmisión de 0 a 9600 bits por segundo.

Una vez instalados los line-drivers se procedió a hacer la interconexión directa de un puerto de la VAX al LINE-DRIVER donde se conectaba la terminal de la B6800 .



La forma de transmisión usada por la Burroughs en la terminal instalada en el CECAFI era asíncrona, misma que utiliza la VAX para el manejo de sus terminales, por lo que este nivel de comunicación no representó ningún problema en la conexión.

Establecida la conexión física, y no existiendo problema en la forma de transmisión para envío de caracteres, se

procedió a la elaboración del software en la VAX, el cual pudiera efectuar una comunicación práctica con Burroughs.

Así fue como se desarrollaron los programas, que consisten básicamente de 3 módulos:

1. Simulación de una terminal asincrona con el objeto de poder trabajar en CANDE o en algún otro MCS de Burroughs.

2. Transferencia de archivos de Burroughs a VAX.

3. Transferencia de archivos de VAX a Burroughs.

Los programas desarrollados en VAX tienen como característica importante su total independencia del sistema Burroughs, es decir, no es necesario que para efectuar la intercomunicación se encuentre corriendo algún programa especial en esta última.

La transmisión y recepción de caracteres no se podían manejar a través de instrucciones normales de lenguajes de alto nivel o ensamblador, por lo que fue necesario utilizar algunas rutinas del "Run Time/Library" y "System Services".

El lenguaje utilizado para desarrollar los programas fue VAX-11 FORTRAN, una versión de Fortran-77, el cual nos permite utilizar directamente las rutinas especiales antes mencionadas.

El primer módulo es un programa que permite trabajar en una sesión en B6800 desde una terminal de VAX. Puede efectuarse cualquier función o comando que normalmente se efectúe en una terminal de Burroughs. Tiene como única limitante que sólo puede estar trabajando un usuario de VAX a la vez, ya que sólo se cuenta con un puerto de interconexión.

El segundo módulo tiene como función transferir archivos de Burroughs a VAX. Su funcionamiento es el siguente: primeramente abrir un archivo en VAX; como segundo paso se manda un comando a Burroughs para quitar la opción de página (?-P) y otro para que liste el archivo sin secuencia (LIST;UN); de esta forma el archivo es recibido en la VAX registro por registro y almacenado en el archivo previamente abierto. Al finalizar la transferencia, se cierra el archivo en VAX y se manda un comando para restablecer la opción de página (?+P). Debido a las características de la terminal definidas en la Burroughs, el tamaño del registro de un archivo está limitado a 132 bytes como máximo.

El tercer módulo se encarga de realizar la transferencia de archivos de VAX a B6800. Funciona de la siguiente forma: primeramente se abre el archivo de la VAX que se va a

transmitir, después se manda un comando a Burroghs para que abra un archivo (MAKE) y otro que dé secuencia de cinta (TAPE SEQ); una vez hecho esto se procede a la transmisión del archivo en VAX a la B6800; terminado el archivo el programa en VAX espera un intervalo de 4 segundos para que la Burroughs cierre el archivo; finalmente se manda un comando para guardar en disco el archivo transmitido (SAVE).

4.3 FORMA DE OPERACION

El programa desarrollado en VAX puede correrse desde cualquier terminal, teniendo como condición que el puerto conectado a la B6800 sea el TTC6: (aunque éste es un parámetro que se puede modificar). Para correrlo se deben seguir los siguientes pasos:

1. Verificar que se encuentre establecida la conexión física entre las dos computadoras; es decir, que el puerto TTC6: de la VAX se encuentre conectado al LINE-DRIVER que comunica la terminal de la B6800 con el CECAFI.

2. Entrar en sesión en una terminal con una clave privilegiada, por ejemplo: SYSTEM u OPERADOR.

3. Verificar que el puerto TTC6: tenga las siguientes caracteristicas:

   1. Velocidad de transmisión y recepción 1200 bps.

   2. No eco.

   3. Width = 132.

   4. Device_type LA120.

   De no estar correcto alguno de los parámetros anteriores hay que corregirlo con el siguiente comando:

```
$ SET TERMINAL/PERMANENT/SPEED=1200/NOECHO/-
$_WIDTH=132/DEVICE_TYPE=LA120 TTC6:
```

4. Si no existió problema en la conexión física y en los parametros entonces ejecutar el archivo de comandos SYS$PROGRAMAS:COMUNICA.COM, de la siguiente forma:

CONEXION DEL SISTEMA VAX 11/780 AL SISTEMA B6800

$ @SYS$PROGRAMAS:COMUNICA.COM

Al ejecutarse saldrá en la pantalla las siguientes cuatro opciones:

1. TERMINAL EN CANDE

2. TRANSFERENCIA DE UN ARCHIVO DE VAX A B6800

3. TRANSFERENCIA DE UN ARCHIVO DE B6800 A VAX

4. SALIDA DEL SISTEMA

Teclee la opción según se desee, teniendo en cuenta que es necesario estar en sesión en B6800 en la clave deseada antes de hacer alguna transferencia de archivos entre las computadoras. Por esto se recomienda usar la opción 1 antes de la 2 ó la 3.

Para terminar la ejecución de la opción 1 y regresar al módulo principal teclee el comando "?VAX". Si se desea interrumpir por completo la ejecución del programa, teclee CTRL/Y.

Los listados de los programas desarrollados en el sistema VAX se encuentran en el Apéndice C.

4.4 MODO ALTERNO DE SOLUCION

Adicionalmente se trabajó en módulos de software para el computador B6800 de Burroughs, desarrollando así una solución alterna al problema de conexión entre los dos computadores.

Siguiendo las especificaciones básicas de enlace del apartado, 4.1 a través de la interfaz RS232-C, se analizaron diferentes formas de comunicación utilizando la misma línea de transmisión-recepción para dicha conexión.

Para implantar la conexión se debe proceder a modificar la red de comunicaciones del B6800, a través del NDL para el DCP, aplicando el protocolo correspondiente.

En el modo normal de operación se definen los parámetros lógicos necesarios para la conexión física de terminales; aunque también, es posible definir en el NDL un protocolo especial de comunicación para conectar diferentes equipos de comunicaciones, y así facilitar el manejo de teleproceso en el ambiente Burroughs.

CONEXION DEL SISTEMA VAX 11/780 AL SISTEMA B6800

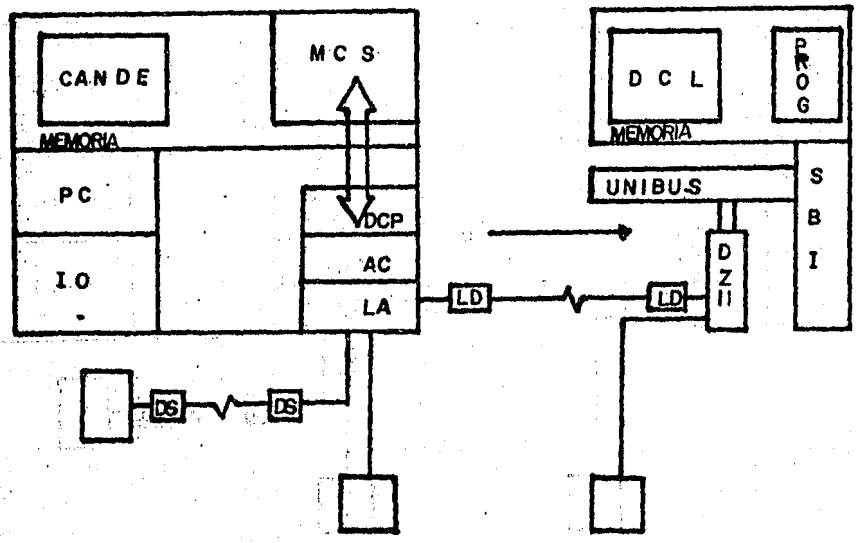
Asociado a cada terminal de un computador Burroughs existe un sistema de control, que se ejecuta en el procesador central, para servir como anfitrión de los usuarios de videos; también puede manejar un sin número de terminales al mismo tiempo. Este tipo de sistemas es llamado MCS (Message Control System).

La forma de trabajo de los MCS's es en memoria principal, éstos reciben la totalidad de información generada por las terminales; es decir, el control de la información es almacenada en colas de mensajes, cuyos elementos contienen información de la terminal que envía el mensaje.

Generalmente se programan dos colas de mensajes para recibir el flujo de información, una es usada para procesar los comandos normales de operación y la otra para procesar comandos de más prioridad, como es el caso de los comandos de control de CANDE.

El sistema debe estar escrito en DCALGOL, lenguaje diseñado para definir en el computador Burroughs lenguajes anfitriones, los cuales incluyen comandos para manejo de estaciones, líneas, medio ambiente, atributos de línea, reconfiguración y prueba.

En base a lo anterior se definió un MCS capaz de recibir la información de la línea declarada para el CECAFI y de transmitirla a la terminal solicitante de comunicación con el computador VAX de acuerdo a la siguiente figura.



CONEXION DEL SISTEMA VAX 11/780 AL SISTEMA B6800

El MCS generado tiene la capacidad de comunicarse con el computador VAX en forma de interfaz; es decir, toma la información de una terminal de Burroughs y la transmite a través de la línea declarada para el CECAFI. También éste tiene la capacidad de transferir el control a otro MCS. Es decir, se programó el código necesario para que una terminal asignada al MCS pudiera regresar a CANDE en cualquier momento. Asimismo, se generó la rutina de manejo de dos colas de mensajes, a través de las cuales se procesa la totalidad de información producida por la conexión. Por lo que,se seleccionarón dos tipos de comandos, los de control y los normales. Los primeros para manejo del flujo de mensajes, y los segundos para comandos prioritarios.

Una característica importante del MCS desarrollado es la facilidad de comunicación con el sistema operativo, ya que tiene la habilidad de recibir mensajes del sistema. Por ejemplo, terminar con el proceso al proporcionar un comando en la consola de operación.

Por otro lado, se trabajó en la detección de altas de terminales a este MCS. El control de una terminal es pasado a un MCS en diferentes formas, por lo que, el MCS implantado contiene el código necesario para determinar, el medio a través del cual una terminal se da de alta.

Las rutinas implantadas en el MCS son los siguientes:

1. Comunicación bilateral con terminales VAX.

2. Posibilidad de cambio de MCS hacia nuevos sistemas.

3. Atención a diversos usuarios del MCS.

En el apéndice D se presenta el código fuente del programa implantado como MCS.

Con el fin de mejorar este MCS, se describe a continuación las formas en las que se podría operar funcionalmente el sistema de control de mensajes.

1. Modo terminal

2. Transferencia de archivos de B6800 a VAX.

3. Tranferencia de archivos de VAX a Burroughs.

Para el primer punto se ideó cambiar las especificaciones del NDL para enviar los requisitos impuestos por VAX. Así en la secuencia de 'poll' del procesador de comunicaciones de Burroughs se procedería a manejar dichos requisitos presentando con esto la primera alternativa de solución al

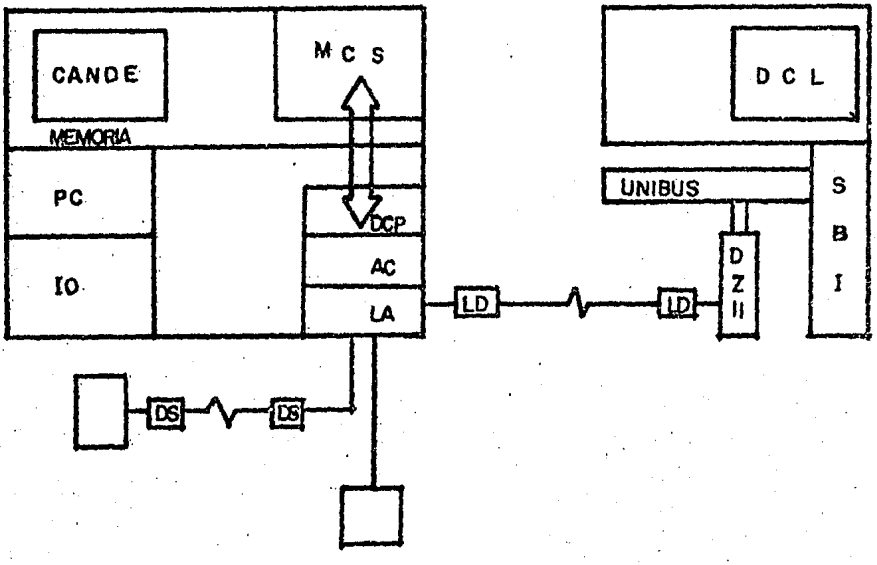
problema.

Las condiciones necesarias para esta alternativa se muestran a continuación:

1. Modificación a la red de comunicaciones del B6800.
2. Compilación de la red modificada.
3. Cancelación de la red sin modificaciones.
4. Instalación de la nueva red.

Como segunda alternativa se ideó comunicar al MCS con un programa corriendo en VAX que simulara DCL (DATA COMMUNICATION LANGUAGE) para evitar el cambio en la red de comunicación. El MCS estaría en comunicación directa con el programa en VAX.

Por motivos de acceso a la red del B6800 la mejor opción sería la segunda. El siguiente diagrama presenta el caso de la comunicación como terminal.



Para los siguientes dos puntos se deberá utilizar los mismos módulos de software tanto en VAX como en Burroughs de tal forma que se implanten los siguientes comandos para la transferencia de archivos.

CONEXION DEL SISTEMA VAX 11/780 AL SISTEMA B6800

1. TRANSFIERE Nomb-archivo-B6800 Nomb-archivo-VAX - Comando para efectuar la tranferencia de archivos de disco a disco en Burroughs a VAX.

2. RECIBE Nomb-archivo-VAX Nomb-archivo-B6800 - Comando utilizado para almacenar información en Burroughs.

Cabe hacer la observación de que el computador de Burroughs genera su propio sistema de detección y corrección de errores en el aspecto de comunicación; razón por la cual no se implantó un procedimiento de corrección en el sistema.

CAPITULO 5

SISTEMA APPLE PLUS II

5.1 CARACTERISTICAS GENERALES

El Sistema APPLE II Plus es una microcomputadora personal y una de las pioneras en su ramo. La unidad central de proceso está integrada por un microprocesador 6502 de 8 bits. Tiene 50k de memoria RAM : 2K para operaciones de entrada/salida y 48K para propósitos generales. El área de memoria ROM es de 14K: 2k se emplean para entrada/salida y en el resto se encuentran programas de manejo del monitor y de inicialización del sistema.

La unidad central maneja 8 conectores internos, situados en la parte posterior de la máquina, y se utilizan para insertar tarjetas que tienen funciones específicas, tales como el sistema APPLE Pascal, mejorar la visualización en pantalla de 40 a 80 columnas, comunicación con dispositivos externos (manejadores de disco flexible, impresoras, modems, etc.).

El sistema operativo regular es el DOS 3.3 (Disk Operating System) y como opcionales están el UCSD Pascal y el CP/M. Los lenguajes que maneja son: en versión estandar, Basic (Integer BASIC y Applesoft BASIC) y opcionales Fortran y Pascal, entre otros.

5.2 MICROPROCESADOR 6502

El microprocesador 6502 maneja 56 instrucciones de ensamblador, emplea 13 modos de direccionamiento, un acumulador, dos registros de índice de 16 bits (X y Y), un registro que apunta al stack (S) y un registro que nos indica el estado del procesador (P).

El microprocesador 6502 utiliza un bus de datos de 8 bits, paralelo y bidireccional, además un bus de direcciones de 16 bits; esto hace que tenga la capacidad de direccionar 64k

localidades de memoria; también utiliza un stack de 256 bytes fijos. La frecuencia del reloj es de 1.023 Mhz.

5.3 SISTEMA OPERATIVO APPLE PASCAL

El sistema operativo APPLE PASCAL está formado por un manejador de archivos de disco flexible, un editor de texto versátil para la escritura de programas y archivos, un compilador Pascal que convierte los programas en código ejecutable (P-code), un ensamblador 6502 que convierte rutinas en lenguaje ensamblador a lenguaje de máquina y un ligador que incluye rutinas externas al programa de aplicación.

Además del sistema operativo principal existen varios programas de utilería, los cuales permiten formatear discos, anexar rutinas al sistema de biblioteca y configurar el sistema para el manejo de terminales externas.

El sistema APPLE PASCAL es una versión del sistema UCSD Pascal, el cual es una implantación de Pascal como intérprete; esto es, el compilador emite código para un "Pseudo-Machine" o "P-Machine" el cual es emulado al tiempo de la ejecución por un programa escrito en el lenguaje de máquina de la computadora empleada.

Para la APPLE, este programa "P-machine" es el intérprete y está escrito en el lenguaje de máquina del microprocesador 6502; se encuentra en el archivo SYSTEM.APPLE.

El sistema operativo de APPLE PASCAL y varias utilerías están escritas en Pascal y se ejecutan con el mismo intérprete.

5.4 SISTEMA OPERATIVO DOS 3,3

El sistema operativo DOS (Disk Operating System) versión 3,3 tiene como función el controlar los manejadores de disco flexible y consiste de una tarjeta que por lo regular se inserta en el conector seis de la microcomputadora. Este sistema está formado por programas que automáticamente almacenan los archivos a disco, salvan y dan acceso información de ellos, entre otras cosas. Emplea comandos que se pueden utilizar desde programas escritos en Applesoft BAsic, Integer Basic o en lenguaje de máquina.

CAPITULO 6

CONEXION DEL SISTEMA APPLE II PLUS A OTROS SISTEMAS

6.1 INTRODUCCION

En los últimos años el mercado de microcomputadoras se ha incrementado enormemente, siendo hoy en día una herramienta muy útil en diversas aplicaciones; sin embargo, existen todavía una serie de trabajos que por la naturaleza de su tamaño, o el tiempo de respuesta requerido, no se pueden realizar en estos sistemas, y es necesario auxiliarse de computadoras de mayor capacidad.

La búsqueda de una situación versátil entre los sistemas de diferente tamaño motivaron la interconexión de las microcomputadoras existentes en el CECAFI a los equipos de tamaño mediano y grande que se tienen a disposición.

6.2 ANTECEDENTES

En el año de 1981 el Centro de Cálculo adquirió 15 microcomputadoras Radio-Schack y tres microcomputadoras APPLE II Plus, cada una de estas últimas con la siguiente configuración:

1 microprocesador 6502 con 48 KB.

2 manejadores de disco flexible con su controlador

1 tarjeta de expanción de memoria de 16 KB

Sistemas Operativos DOS 3.3 y APPLE Pascal

1 impresora Texas Instruments OMNI 800 y con ella la tarjeta paralelo

Dos de estas microcomputadoras se asignaron para dar servicio a los alumnos y profesores de la Facultad de Ingeniería, y la tercera para uso exclusivo de proyectos del

CONEXION DEL SISTEMA APPLE II PLUS A OTROS SISTEMAS

Centro.

A principios del año de 1984, la Facultad de Ingeniería adquiere el sistema Altos 986, una microcomputadora de 16 bits multiusuario; esto, aunado a la existencia del sistema VAX 11/780, hizo que se pensara en las microcomputadoras APPLE para otras aplicaciones diferentes a las que comúnmente se efectuaban con ella.

Por otro lado, con la adquisición del Sistema VAX 11/780, el Centro de Cálculo de la Facultad de Ingeniería tenía sólo 20 terminales VT100, como ya se mencionó anteriormnete; dichas terminales resultan apenas suficientes para dar servicio a profesores, alumnos y personal encargado del desarrollo y producción de sistemas; por lo que no se asignó ninguna terminal al personal de captura de dicho centro, y hubo la necesidad de seguir utilizando perforadoras de tarjetas para la captura de datos, lo que producía un gasto inútil por el uso de un producto caro y no reutilizable, como son las tarjetas.

Fue entonces cuando se planteó la posibilidad de utilizar los sistemas APPLE para disminuir en parte estos problemas; es decir, estas microcomputadoras se utilizarían como terminales y como dispositivos de captura en disco flexible.

El sistema APPLE PASCAL de las microcomputadoras APPLE emplean un editor de textos muy versátil, similar a uno de los editores del sistema VAX 11/780: el editor EDT; por lo que también facilita la captura de programas fuente de VAX que pueden ser posteriormente transmitidos para su rápida compilación y ejecución.

El editor EDT en VAX es un editor muy bueno, excelente en el manejo de caracteres, pero tiene la desventaja que consume muchos recursos, y en algunos casos, cuando el sistema está un poco saturado, el tiempo de respuesta no es tan rápido como podría ser si se trabajara con el microcomputador, ya que en este último caso se trata de un solo usuario.

En resumen; utilizando las microcomputadoras APPLE como tales para la creación de archivos y posteriormente trasmitirlos al sistema VAX 11/780 o al sistema B6800, se reduce el consumo de recursos y la dependencia a estos últimos, ya que se tiene la ventaja de no existir necesidad de estar conectados en todo momento al los equipos grandes, para poder hacer varias aplicaciones.

6.3 DESARROLLO

Para efectuar la intercomunicación entre dos computadoras se deben hacer consideraciones, tanto en la conexión física como en la realización de los protocolos de comunicación.

Las consideraciones que se deben tomar en la conexión física son:

- Observar el tipo de interfaz de comunicación con dispositivos externos que emplean las máquinas que se desean conectar.
- Verificar que estas interfaces sean compatibles para su conexión.

Las consideraciones que se deben tomar para la realización de protocolos de comunicación son:

- Analizar los protocolos de comunicación que emplean las computadoras que se quieren conectar.
- Investigar el código que utilizan éstas para la emisión y recepción de caracteres

6.3.1 CONDICIONES FISICAS

Tomando dichas consideraciones para la conexión física entre el sistema APPLE II Plus y el sistema VAX 11/780, se presentó el problema de que la microcomputadora APPLE sólo contaba con una interfaz paralela que se empleaba para la conexión de la impresora. La computadora VAX 11/780, en cambio, manejaba las señales de la interfaz serie RS232. Habiendo esta incompatibilidad se vio la necesidad de comprar una interfaz serie para APPLE que empleara las señales requeridas. Fue entonces cuando se adquirió la interfaz serie CA J13, descrita a continuación

6.3.1.1 INTERFAZ SERIE CA J13 - La interfaz serie CA J13 es una tarjeta propia para las microcomputadoras APPLE II y APPLE IIe. Provee dos formas de conexión serial: Terminal y Modem; la primera se usa cuando se requiere conectar terminales, impresoras, graficadores y cualquier otro equipo terminal de datos, y la segunda se usa para conectar modems, computadoras o cualquier otro equipo de comunicación de datos. Dependiendo de la aplicación, el cable interno de la tarjeta serie estará enchufado al conector Terminal o Modem.

CONEXION DEL SISTEMA APPLE II PLUS A OTROS SISTEMAS

Este interfaz es compatible con todos los sistemas operativos que emplea APPLE: DOS 3.3, Basic Integer y Applesoft, APPLE Pascal, etc. El rango de transmisión va desde 110 a 19,200 bits/seg; el cambio de dicha velocidad se efectúa por medio de microswitches localizados en la misma tarjeta, es decir se hace mecánicamente y no por programa.

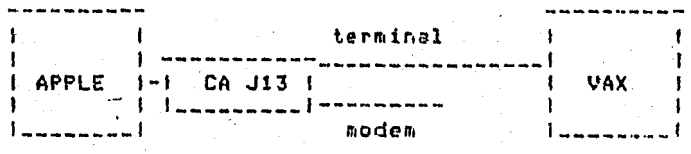
El corazón de esta tarjeta serie es el microcircuito 'Asynchronous Communications Interface Adapter' (ACIA) MC6850, el cual tiene como función la conversión del dato de paralelo a serie a la salida y de serie a paralelo a la entrada; además de controlar la entrada y salida del dato.

6.3.1.1.1 ACIA MC6850 - La interfaz asincrona MC6850 usa un bus del microprocesador estándar de la familia 6800 y produce o recibe la señales de control básicas de la interfaz RS232-C. Este dispositivo contiene cuatro registros que son accesibles por el microprocesador -un registro de control para comandos, un registro de estado, un registro de entrada y un registro de salida.

El registro de comandos se utiliza para configurar el modo de transmisión, o sea para indicar la paridad, el número de bits de 'stop' y armar o desarmar la salida de interrupción. La salida REQUEST TO SEND está representada por un solo bit en el registro de control cuyo valor se habilita por programa. El registro de estado refleja el estado presente de los registros de entrada y salida (si se encuentra dato o esta vacio respectivamente), de las señales de CLEAR TO SEND y CARRIER DETECT, y cuándo se requiere de interrupción. Los registros de entrada y salida se utilizan para almacenar el dato que se ha recibido o el que se enviará, respectivamente, por el canal de comunicación.

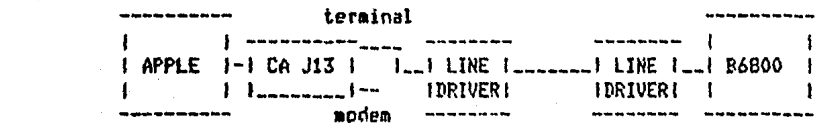
6.3.2 CONEXION FISICA

Habiendose adquirido la tarjeta serie se procedió a efectuar la conexión física. Se realizó en forma directa a un puerto de la VAX, por lo que se conectó el DB-25 a la salida 'terminal' de la tarjeta serie.



CONEXION DEL SISTEMA APPLE II PLUS A OTROS SISTEMAS

Para la conexión física con el sistema B6800 se emplearon los line-drives requeridos para la conexión de los sistemas VAX y B6800; como no fue necesario emplear las señales de control de modem, se conectó de DB-25 a la salida "terminal" de la tarjeta serie, de la misma forma que en el caso anterior.



6.3.3 REALIZACION DEL PROTOCOLO DE COMUNICACION

De las consideraciones que se deben tomar para la realización del protocolo de comunicación se observó que no existía ningún problema, ya que el protocolo que utilizaba el sistema VAX 11/780 era asíncrono, mismo que empleaba la interfaz serie CA J13 del sistema APPLE. También se tenía que para la emisión y recepción de datos se utilizaba el código ASCII en ambas máquinas. No habiendo problemas que solventar, se procedió a desarrollar los programas de comunicación.

Se desarrollaron tres programas que consisten básicamente en:

1. Simulación de una terminal asíncrona con el objeto de emplear los comandos del sistema VAX 11/780
2. Transferencia de archivos de VAX a APPLE
3. Transferencia de archivos de APPLE a VAX

Estos programas fueron desarrollados en APPLE, siendo totalmente transparentes para el sistema VAX 11/780. Para su desarrollo hubo la necesidad de emplear el lenguaje ensamblador 6502, ya que se tenía que estar interactuando con localidades de memoria RAM empleadas por la interfaz CA J13.

Como nuestro objetivo era el emplear el sistema APPLE Pascal por la versatilidad en el manejo de editor (como ya se mencionó),los programas de comunicación se escribieron en lenguaje Pascal con manejo de rutinas en ensamblador 6502.

CONEXION DEL SISTEMA APPLE II PLUS A OTROS SISTEMAS

El sistema operativo APPLE Pascal trata a los periféricos como dispositivos lógicos; para esto, los periféricos deben estar asociados a ciertos conectores de APPLE para que sean reconocidos. El sistema operativo reserva algunos nombres de archivos para nombrar tales dispositivos como el teclado, el video, la impresora y el modem. La tabla siguiente muestra la asignación de conectores para cada dispositivo.

Conector 1	Impresora
Conector 2	Dispositivo para propósitos generales de entrada/salida (modem)
Conector 3	Consola para entrada/salida (terminal)

Los nombres lógicos asociados a dichos dispositivos son:

PRINTER:	Impresora
REMIN:	Dispositivo remoto de entrada (modem)
REMOUT:	Dispositivo remoto de salida (modem)
CONSOLE:	Consola de entrada/salida (terminal)

Teniendo esta facilidad en el manejo de periféricos se decidió colocar la interfaz serie CA J13 en el conector dos y así, para enviar datos por dicho puerto, se hacia referencia a REMOUT: y para la recepción de datos a REMIN:. Sin embargo, el utilizar estos nombres lógicos no ayudó mucho a lograr la intercomunicación, ya que no había forma de estar preguntando si había dato que recibir o si el puerto estaba listo para emitir información. De aquí que se optó por emplear el lenguaje ensamblador por medio del cual se permitía conocer el estado del canal de comunicación.

La microcomputadora APPLE tiene dieciseis bytes de memoria RAM asociados a cada conector para realizar funciones de entrada y salida. Cada grupo de bytes inicia en la dirección C0B0+n0, donde n es el número de conector. En nuestro caso, la tarjeta serie CA J13 se conectó en el número dos; por tanto, su grupo de bytes inicia en la localidad C0A0.

Los registros empleados por el microcircuito ACIA MC6850 (localizado en la tarjeta serie) son accesibles escribiendo o leyendo los bytes asociados al conector. La siguiente tabla muestra las localidades de memoria RAM asociadas a los registros del ACIA MC6850.

Dirección	Registro
C08E+n0	Registro de estado. Este registro sólo de lectura se emplea para obtener el estado en tiempo real del ACIA 6850.
C08E+n0	Registro de control. Este registro sólo de escritura se emplea para especificar

varios modos de [illegible] del ACIA 6850.

C08F+n0 — Registro de datos. Este registro se emplea para leer y escribir el dato al ACIA 6850.

Los programas de comunicación desarrollados emplean constantemente dichas localidades de memoria RAM para la emisión y recepción de caracteres.

Como se señaló, los programas son transparentes para el sistema VAX 11/780; por tanto, el programa que realiza la simulacion de APPLE como terminal asincrona se puede emplear para la conexión con otro sistema que tenga las caracteristicas de comunicación del sistema VAX 11/780. Tomando esto en cuenta, se procedió a la conexión de APPLE al sistema Burroughs como terminal asincrona, utilizando el mismo programa, lográndose así la intercomunicación.

6.4 PROCESO DEL PROGRAMA DE COMUNICACION

El programa de comunicación debe estar en cualquier disco flexible del sistema APPLE Pascal. Para iniciar la comunicación, después de haber hecho la conexión física, se ejecutará el programa INTER.CODE el cual despliega lo siguiente en la pantalla:

INTERCOMUNICACION

(1) TERMINAL

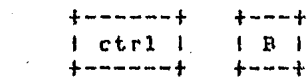
(2) TRANSFERENCIA DE ARCHIVOS DE APPLE A VAX

(3) TRANSFERENCIA DE ARCHIVOS DE VAX A APPLE

(4) FIN DE INTERCOMUNICACION

OPCION:

Si se escoge la opción 1, el sistema APPLE funcionará como una terminal asincrona del sistema VAX 11/780, Burroughs B-6800 o cualquier otro sistema que emplee protocolo asincrono. Para salir de esta opción, se deberá teclear :



Si se escoge la opción 2, se efectuará la transferencia de archivos del sistema APPLE al sistema VAX 11/780 de la siguiente manera: primeramente se pide el nombre COMPLETO del archivo en APPLE que se desea transferir al sistema VAX 11/780, posteriormente el nombre con que se quiere en VAX. De esta forma se empieza a transferir el archivo; al término de la transferencia se desplegará de nuevo la opciones.

La opción 3 efectuŕa la transferencia de archivos del sistema VAX 11/780 al sistema APPLE, pidiéndose lo siguiente: nombre COMPLETO del archivo en VAX que se requiere transferir, nombre con el que se quedará en APPLE. Y se efectúa la transferencia; de igual manera que en la opción 2, al término de la transferencia se desplegará el menú.

Esta opción tiene una limitante: el archivo que se transmitirá a APPLE no debe rebasar de los 20 Kbytes.

Por último, si se escoge la opción 4, terminará la comunicación y el sistema APPLE actuará como una microcomputadora.

Los listados de los programas desarrollados en el sistema Apple se encuentran en el Apéndice E.

CAPITULO 7

CONCLUSIONES

El presente trabajo fue desarrollado en tres diferentes equipos de procesamiento electrónico de datos, para obtener un panorama más amplio acerca de las perspectivas de comunicaciones que se tienen con los equipos disponibles.

Con objeto de mantener un nivel más adecuado en cuanto a interconexión de sistemas de cómputo, se instalaron los programas que solucionan los problemas fundamentales; sin embargo, consideramos que este trabajo servirá como guía para futuras aplicaciones que persigan objetivos similares.

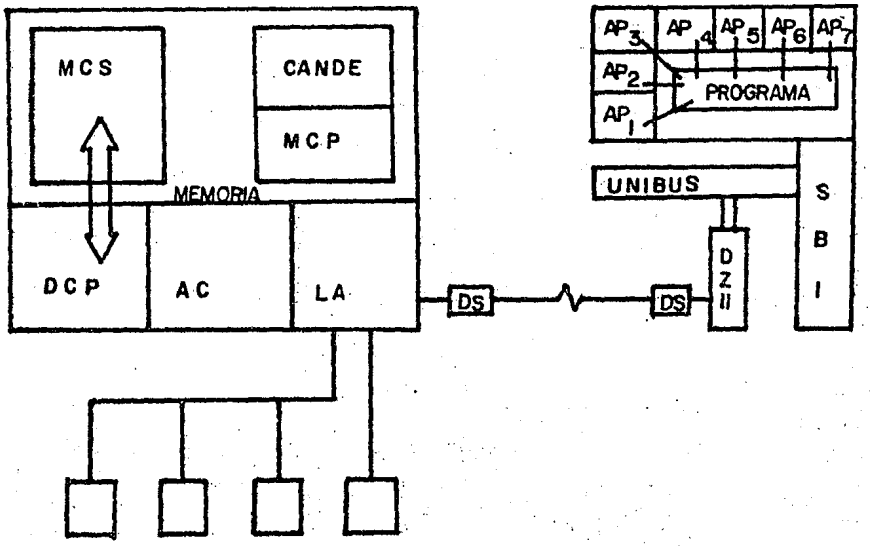
En el Sistema VAX se desarrollaron tres módulos principales, uno para modo terminal, y dos más para tranferencia de archivos con un Sistema Burroughs. Debido a que, como se anotó en el Capítulo 4, estos programas son independientes del software que se encuentre corriendo en Burroughs; su flexibilidad para la adaptación a otro sistema similar es considerable, ya que sólo se requiere de algunos cambios en los programas de transferencia de archivos en las instrucciones dependientes del sistema, como son: LIST, MAKE y SAVE, por sus equivalentes en el sistema correspondiente.

Por otro lado, este sistema tiene la limitación de trabajar solamente con un usuario a la vez. Esto se podría solucionar cambiando la definición de la terminal en la red de Burroughs para que reconociera la línea como la de un concentrador, de tal manera que la VAX efectuaría las funciones del concentrador pero a través de software.

Para el funcionamiento de este sistema, también sería necesario, desarrollar un programa especial en VAX, que controlara el envío de los mensajes de los diferentes usuarios a un MAILBOX y de ahí los enviara a Burroughs con una dirección de identificación. A la vez se tendría un sistema de recepción de mensajes de Burroughs a VAX para después enviarlos a la terminal correspondiente.

CONCLUSIONES

Por lo que toca al sistema Burroughs, en un análisis de posibilidades, se observó, que modificando el MCS propuesto es factible utilizar varias terminales de Burroughs sobre la misma línea de VAX, funcionando así como un concentrador. Para esto el MCS tendría que comunicarse con un programa especial corriendo en VAX, el cual pudiera atender los comandos enviados por las diferentes terminales de Burroughs enlazadas a través de este MCS.



Las condiciones técnicas que debería contener el MCS para esta opción son las siguientes:

1. Considerar un arreglo de información de terminales.
2. Generar colas de comandos provenientes de las terminales.
3. Mandar el comando con la información de la terminal que lo generó.
4. Generar un algoritmo de "time-slice" para mejorar el "troughput".

Es pertinente mencionar que es estrictamente necesaria la comunicación de este MCS con un programa corriendo en VAX para efectuar la simulación de sesiones en DCL, ya que en

CONCLUSIONES

VAX el sistema operativo atiende sólo una terminal por línea.

Por su parte, el software desarrollado en el sistema APPLE sirve para emular una terminal asíncrona y transferir archivos entre la APPLE y la VAX; por lo que, se puede utilizar como terminal asíncrona de cualquier sistema que así lo requiera. Además con cambios básicos en las instrucciones específicas del sistema, como son TYPE y CREATE, se podría lograr también la transferencia de archivos.

En las conexiones estudiadas, se observó que el comportamiento general de los sistemas de interconexión de computadores es similar en los equipos de procesamiento, cuando se trata de líneas asíncronas; de aquí se desprende que la forma más sencilla de comunicación es con el manejo de declaraciones asíncronas.

En lo que respecta al manejo de errores de transmisión de datos, es recomendable el uso de facilidades que ofrecen los proveedores de equipo de comunicación, ya que evita así el desarrollo de algoritmos para detección y corrección de errores, además de un "overhead" innecesario para el programa a diseñar.

APENDICE A

INTERFAZ EIA RS232-C

INTERFAZ ENTRE TERMINALES DE DATOS Y EQUIPO DE COMUNICACION DE DATOS EMPLEANDO INTERCAMBIO EN SERIE DE DATOS BINARIOS.

A.1 INTRODUCCION

A.1.1 RESUMEN

Este estándar es aplicable a la interconexión de equipos terminales de datos (dte), y equipos de comunicación de datos (dce) empleando intercambio serie de datos binarios. Se define:

SECCION A.2 - Características de señales eléctricas: Características de señales eléctricas y circuitos relacionados.

SECCION A.3 - Características mecánicas de la interfaz: Definición de las características mecánicas de la interfaz entre el equipo terminal de datos y el equipo de comunicaciones de datos.

SECCION A.4 - Descripción funcional de circuitos de intercambio: Descripción funcional del conjunto de datos, sincronización y circuitos de intercambio para uso de una interfaz digital entre el equipo terminal de datos y el equipo de comunicaciones de datos.

A.1.2 CONFIGURACION DE LA INTERFAZ

Este estándar incluye trece configuraciones especificas tratando de adecuarse a quince aplicaciones de sistemas definidos. Estas configuraciones son identificadas por su tipo, usando los caracteres alfabéticos de la 'a' a la 'm' y donde la configuración de circuitos de intercambio será indicada en cada caso por el proveedor.

A.1.3 VELOCIDADES DE TRANSMISION

Este estándar es aplicable para uso de velocidades de señalización entre cero y el límite superior de 20,000 bits por segundo.

A.1.4 SEÑAL COMUN DE TIERRA

Este estándar es aplicable para el intercambio de datos, sincronización y señales de control cuando son usados en conjunto con el equipo electrónico, cada una de los cuales tiene una señal de tierra, que puede ser interconectada en el punto de la interfaz. Esto no es aplicable cuando se requiere aislamiento de equipo entre los lados opuestos.

A.1.5 COMUNICACION SINCRONA Y NO SINCRONA

Este estándar se usa en sistemas de comunicaciones síncronos y asíncronos.

A.1.6 TIPOS DE SERVICIO

Este estándar se aplica a todos las clases de servicios de comunicaciones de datos, incluyendo:

1. Enlaces especiales o servicio de líneas privadas, en dos o cuatro cables. Consideraciones dadas para operaciones punto a punto o multipunto.

2. Servicio de red conmutada, en dos o cuatro cables. Consideraciones dadas para respuestas automáticas de llamadas; sin embargo, este estándar no incluye todos los intercambios requeridos de circuitos para una conexión automática a un origen. (Ver EIA estandar RS-366 "Interface between data terminal

equipment and automatic calling equipment for data communication").

A.1.7 TIPO DE FUNCIONES

El conjunto de datos puede incluir señales convertidoras de recepción y transmisión, así como funciones de control. Otras funciones, tales como generación de pulsos, control de errores, etcetera, pueden o no ser implantadas. El equipo que permite estas funciones adicionales puede ser incluido en el equipo terminal de datos o en el equipo de comunicación de datos, o puede ser implantado como una unidad separada interconectada entre los dos.

Cuando tales funciones son implantadas dentro del equipo terminal de datos o el equipo de comunicación de datos, esta interfaz estándar se aplicara sólo al intercambio de circuitos entre las dos clases de equipos.

Cuando funciones adicionales son implantadas en una unidad insertada entre el equipo terminal de datos y el equipo de comunicación de datos, este estandar es aplicable en ambos lados (la interfaz con el equipo terminal de datos y la interfaz con el equipo de comunicación de datos - ver sección A.3.1) de cada unidad por separado.

A.1.8 MODOS DE OPERACION

Este estándar se aplica a todos los modos de operación provistos bajo las diferentes interfaces estándar para configuraciones de sistemas de comunicaciones.

A.2 CARACTERISTICAS ELECTRICAS DE LAS SEÑALES

A.2.1 CIRCUITO EQUIVALENTE

La figura siguiente (circuito equivalente de intercambio), muestra los parámetros eléctricos que serán explicados en los siguientes párrafos de esta sección. El circuito equivalente mostrado en la figura 2.1 es aplicable a todo el intercabio de circuitos contemplado en la categoría (datos, sincronizacion, o control) a la que pertenecen. El circuito equivalente es independiente de si el manejador es conectado en el equipo de comunicación de datos y el conector en el equipo terminal de datos o viceversa.

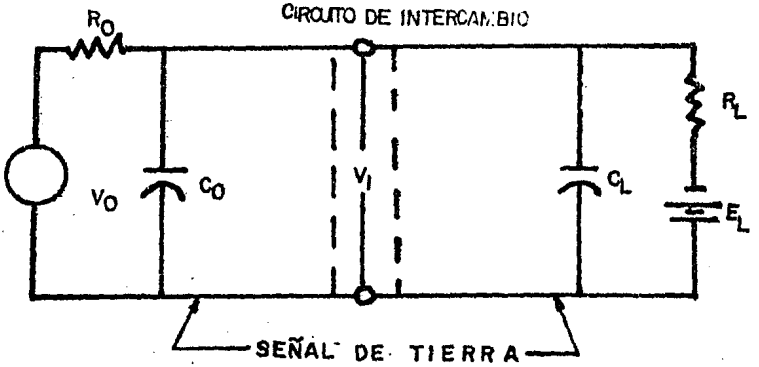


FIGURA 2.1 Circuito equivalente de intercambio

- Vo es el voltaje de control a circuito abierto.
- Ro es la resistencia de control interno (dc).
- Co es la capacitancia total efectiva asociada con el controlador, medida en el punto de la interfaz e incluyendo cualquier cable al punto de la interfaz.
- Vi es el voltaje en el punto de la interfaz.
- CL es la capacitancia total efectiva asociada con el controlador, medida en el punto de la interfaz e incluyendo cualquier cable al punto de la interfaz.
- RL es la resistencia terminal de carga (dc).
- EL es el voltaje terminal de circuito abierto (bias).

A.2.2 CONSIDERACIONES DE SEGURIDAD

El manejador en un circuito de intercambio debe ser diseñado para que permanezca en circuito abierto, un corto circuito entre los conductores produce que el circuito de intercambio en el cable de interconexión y cualquier otro conductor en ese cable, o cualquier carga pasiva no inductiva entre ese circuito de intercambio y cualquier otro circuito de intercambio incluyendo el Circuito AB (señal de tierra), sin daño significativo a él o al equipo asociado. La terminal en el circuito de intercambio debe ser diseñada contemplando cualquier señal dentro de los límites de 25 V especificados en la sección A.2.6.

A.2.3 DEFINICION DE LOS ESTADOS DE LA SENAL EN LOS CIRCUITOS DE INTERCAMBIO

Para circuitos de intercambio de datos, la señal será considerada en condiciones de marca cuando el voltaje (V1) en el circuito de intercambio, medido en el punto de la interfaz, es más negativa que menos tres voltios con respecto al circuito AB (señal de tierra). La señal será considerada en la condición de espacio cuando el voltaje (V1) sea más positivo de tres voltios con respecto al circuito AB. La región entre más tres voltios y menos tres voltios se define como la región de transición. El estado de la señal no tiene definición única cuando el voltaje (V1) está en esta región de transición. Durante la transmisión de datos, la condición de marca será usada para distinguir el estado 'UNO' y la condición de espacio será usada para distinguir el estado 'CERO'.

Para sicronización y control en circuitos de intercambio, la función será considerada ON cuando el voltaje (V1) en el circuito de intercambio es más positiva en tres voltios con respecto al circuito AB, y deberá ser considerada OFF cuando el voltaje (V1) es más negativo que menos tres voltios con respecto al circuito AB. La función no esta únicamente definida para voltajes en la región de transición entre tres voltios y menos tres voltios.

Notación	Voltaje de intercambio	
	Negativo	Positivo
Estado binario	1	0
Condición de la señal	Marca	Espacio
Función	OFF	ON

Estas especificaciones tampoco implican ni excluyen el uso de circuitos, los cuales utilizan técnicas de histéresis para aumentar su inmunidad al ruido. Sin embargo, los

requerimientos de la sección A.2.5 deben ser satisfechos.

A.2.4 IMPEDANCIA TERMINAL

La impedancia de carga (RL y CL) del lado del terminador de un circuito de intercambio tendrá una resistencia (RL) de no menos de 3000 ohms, medidos con un voltaje aplicado de 3 a 25 voltios en magnitud. La capacitancia efectiva derivada (CI) del lado del terminador de un circuito de intercambio, medida en el punto de interfaz, no excederá de 2500 pico-farads. La componenete reactiva de la impedancia de carga no será inductiva. El voltaje terminal de circuito abierto (EI) no excederá 2 voltios en magnitud.

A.2.5 PROTECCION CONTRA FALLAS

Los siguientes circuitos de intercambio, cuando se implementen deberán ser usados para detectar un corte de potencia en el equipo conectado a través de la interfaz, o la desconexión de los cables de interconexión.

Circuito CA (Request to Send)

Circuito CC (Data Set Ready)

Circuito CD (Data Terminal Ready)

Circuito SCA (Secondary Request to Send)

La impedancia de la fuente de potencia del lado del controlador de estos circuitos no debe ser menor de 300 OHMS, medidos con un voltaje aplicado no mayor de 2 volts en magnitud referidas al Circuito AB (Señal de Tierra). El terminador de estos circuitos deberá interpretar la condición de corte de potencia o la desconexión de los cables de interconexión como una condición de OFF.

A.2.6 VOLTAJES Y CORRIENTES DE CONTROL

El voltaje controlador de circuito abierto (Vo), con respecto al Circuito AB (Señal de Tierra) en cualquier circuito de intercambio, no deberá de exceder 25 voltios en magnitud. La fuente de impedancia (RO y CO) del lado del controlador de un circuito de intercambio, incluyendo

cualquier cable al punto de la interfaz no está especificado; sin embargo, la combinación de Vo y Ro deberá ser seleccionada en tal forma que un corto circuito entre dos conductores cualesquiera (incluyendo tierra) en los cables de interconexión no resultará en una corriente mayor a medio Amper. Adicionalmente, el diseño del controlador debe ser tal que, cuando la resistencia de carga terminal (RL) esté en un rango entre 3000 y 7000 Ohms, el voltaje de circuito abierto terminal es cero, el pontencial (V1) en el punto de la interfaz no será menor de 5 volts y no mayor de 15 volts en magnitud.

A.2.7 REQUERIMIENTOS DE LA REGION DE TRANSICION

Las características del intercambio de las señales transmitidas a través del punto de interfaz, fuera de interferencias externas, conformarán a las limitaciones especificadas en esta sección. Estas limitaciones serán satisfechas en el punto de la interfaz cuando el circuito receptor cumple con las condiciones dadas en la Sección A.2.4. Estas limitaciones se aplican a todas las señales de intercambio (Datos, Control y Sincronización) a menos que otra cosa se especifique.

A.3 CARACTERISTICAS MECANICAS DE LA INTERFAZ

A.3.1 DEFINICION DE INTERFAZ

La interfaz entre el equipo terminal de datos y el equipo de comunicación de datos debe estar ubicada en un punto tal donde se pueda conectar la señal de la interfaz entre los dos equipos. El conector hembra debe estar asociado(no necesariamente adherido físicamente) con el equipo de comunicación de datos y estar montado fijamente cerca del equipo terminal de datos. Es factible emplear una extensión de cable en el equipo de comunicación de datos. El equipo terminal de datos debe proveer una extensión de cable con un conector macho. Se recomienda el uso de cables cortos (aproximadamente de 15 metros); sin embargo, se permite usar cables más largos con tal de que la capacidad de carga resultante, medida en el punto donde se encuentra la interfaz e incluyendo la señal terminal, no exceda de 2500 picofaradios.

Cuando las funciones adicionales están dadas por una unidad independiente entre el equipo terminal de datos y el equipo de comunicación de datos, el conector hembra, como se indicó anteriormente, debe estar asociado con el lado de esta unidad en el cual se intercomunica con el equipo terminal de datos, mientras que la extensión de cable con el conector macho debe estar del lado en el cual se intercomunica con el equipo de comunicación de datos.

A.3.2 IDENTIFICACION DE PIN

La figura anterior indica la función asignada a cada pin

La asignación de los pines no definidos específicamente en la sección 4 se harán por mutuo acuerdo. De preferencia no deben utilizarse pero en caso de que se requieran pines adicionales se deben tomar extremas precauciones en su selección.

NUMERO DE PIN	CIRCUITO	NOMBRE
1	AA	PROTECTIVE GROUND
2	BA	TRANSMITTED DATA
3	BB	RECEIVED DATA
4	CA	REQUEST TO SEND
5	CB	CLEAR TO SEND
6	CC	DATA SET READY
7	AB	SIGNAL GROUND (COMMON RETURN)
8	CF	RECEIVED LINE SIGNAL DETECTOR
9	—	(RESERVED FOR DATA SET TESTING)
10	—	(RESERVED FOR DATA SET TESTING)
11		NO ASIGNADO
12	SCF	SEC. REC'D LINE SIG. DETECTOR
13	SCB	SEC. CLEAR TO SEND
14	SBA	SECONDARY TRANSMITTED DATA
15	DB	TRANSMISSION SIGNAL ELEMENT TIMING (DCE SOURCE)
16	SBB	SECONDARY RECEIVED DATA
17	DD	RECEIVER SIGNAL ELEMENT TIMING (DCE SOURCE)
18		NO ASIGNADO
19	SCA	SECONDARY REQUEST TO SEND
20	CD	DATA TERMINAL READY
21	CG	SIGNAL QUALITY DETECTOR
22	CE	RING INDICATOR
23	CH/CI	DATA SIGNAL RATE SELECTOR (DTE/DCE SOURCE)
24	DA	TRANSMIT SIGNAL ELEMENT TIMING (DTE SOURCE)
25		NO ASIGNADO

FIGURA 3.1

A.4 DESCRIPCION FUNCIONAL DE LOS CIRCUITOS DE INTERCOMUNICACION

A.4.1 GENERALIDADES

Esta sección define los circuitos de intercomunicación básicos, los cuales se dirigen, colectivamente, a todo el sistema.

Se pueden utilizar circuitos de intercomunicación adicionales no definidos en esta sección, o variaciones en las funciones de los circuitos si definidos.

A.4.2 CATEGORIAS

Los circuitos de intercomunicación entre el equipo terminal de datos y el equipo de comunicación de datos caen dentro de cuatro categorías.

Tierra o 'common return'

Circuitos de datos

Circuitos de control

Circuitos de sincronización

Una lista de circuitos mostrando la categoría y la identificación de C.C.I.T.T. equivalente, de acuerdo con la recomendación V.24, se presenta en la figura 4.1.

A.4.3 CARACTERISTICAS DE LAS SEÑALES, GENERALIDADES

Los circuitos que transfieren señales de datos a través de la interfaz deben mantener condiciones de marca (uno binario) o espacio (cero binario) para la duración nominal total de cada elemento de la señal.

En sistemas síncronos, usando equipo de comunicación de datos síncronos, se pueden recurrir a las tolerancias de distorsión especificadas en el RS-334. Las tolerancias de distorsión aceptables para equipo terminal de datos en sistemas síncronos y sistemas 'start-stop' (p. ej. asíncronos), usando equipo de comunicación de datos no

síncrono están bajo consideración para otro estándar de RS334.

Los circuitos que transfieren señales sincronizadas a través de la interfaz mantienen condiciones ON y OFF para periodos nominalmente iguales, comprendidos con tolerancias aceptables como las especificadas en RS-334.

CIRCUITO	CCITT	NOMBRE DEL CIRCUITO	TIERRA	DATOS		CONTROL		TIEMPO	
				DE	A	DE	A	DE	A
AA	101	PROTECTIVE GROUND	X						
AB	102	SIGNAL GROUND/COMMON RETURN	X						
BA	103	TRANSMITTED DATA			X				
BB	104	RECEIVED DATA		X					
CA	105	REQUEST TO SEND					X		
CB	106	CLEAR TO SEND				X			
CC	107	DATA SET READY				X			
CD	108.2	DATA TERMINAL READY					X		
CE	125	RING INDICATOR				X			
CF	109	RECEIVED LINE SIGNAL DETECTOR				X			
CG	110	SIGNAL QUALITY DETECTOR				X			
CH	111	DATA SIGNAL RATE SELECTOR(DTE)					X		
CI	112	DATA SIGNAL RATE SELECTOR(DCE)				X			
DA	113	TRANSMITTER SIGNAL ELEMENT TIMIING (DTE)							X
DB	114	TRANSMITTER SIGNAL ELEMENT TIMING (DCE)						X	
DD	115	RECEIVER SIGNAL ELEMENT TIMING (DCE)						X	
SBA	118	SECONDARY TRANSMITTED DATA			X				
SBB	119	SECONDARY RECEIVED DATA		X					
SCA	120	SECONDARY REQUEST TO SEND					X		
SCB	121	SECONDARY CLEAR TO SEND				X			
SCF	122	SECONDARY REC'D LINE SIGNAL DETECTOR				X			

FIGURA 4.1
CIRCUITOS POR CATEGORIAS

Durante periodos en donde la información sincronizada no está dada sobre un circuito de sincronía de intercomunicación, este circuito debe estar fijo en la condición de OFF. Las tolerancias sobre la relación entre datos y las señales sincronizadas asociadas deben estar de acuerdo con RS-334.

A.4.4 CIRCUITOS DE INTERCOMUNICACION

CIRCUITO AA - PROTECTIVE GROUND (C.C.I.T.T. 101)
Dirección: No aplicable.

Este conductor debe estar vinculado eléctricamente a la máquina. Además puede ser conectado a tierras externas requeridas para aplicar regulaciones.

CIRCUITO AB - SIGNAL GROUND OR COMMON RETURN (C.C.I.T.T. 102)
Dirección: No aplicable.

Este conductor establece la tierra de referencia común potencial para todos los circuitos de intercambio excepto, para el Circuito AA (Protective Ground). Con el equipo de comunicación de datos, este circuito se debe llevar a un punto en el cual sea posible conectarse al Circuito AA por medio de un alambre interno al equipo. Este alambre puede ser conectado o extraído a la instalación, ya que puede ser requerido para regulaciones aplicables conocidas o para minimizar la introducción de ruido dentro del circuito electrónico.

CIRCUITO BA - TRANSMITTED DATA (C.C.I.T.T. 103)
Dirección: Al equipo de comunicación de datos.

Las señales en este circuito están generadas por el equipo terminal de datos y son transferidos al convertidor de la señal local de transmisión para la emisión de datos al equipo terminal de datos remotos.

El equipo terminal de datos debe mantener el Circuito BA (Transmitted Data) en condición de marca durante intervalos entre caracteres o palabras, y en todo tiempo mientras no se estén transmitiendo datos.

En todo sistema, el equipo terminal de datos no debe transmitir datos a menos que una condición de ON esté presente en los siguientes cuatro circuitos:

1. Circuito CA (Request to Send)

2. Circuito CB (Clear to Send)

3. Circuito CC (Data Set Ready)

4. Circuito CD (Data Terminal Ready)

Todas las señales de datos que son transmitidos a través de la interfaz en el circuito BA (Transmitted Data) durante el tiempo en que se mantiene la condicion ON en todos los circuitos antes mencionados, deben ser transmitidas al canal de comunicación.

Ver la sección A.4.3, para características de las señales.

CIRCUITO BB - RECEIVED DATA (C.C.I.T.T. 104)
Dirección: Del equipo de comunicación de datos.

Las señales en este circuito son generadas por el convertidor de señal recibida en respuesta a las señales de datos recibidas desde el equipo terminal remoto de datos, vía convertidor de transmisión de señales remotas.

El Circuito BB (Received Data) debe estar mantenido en condición de UNO binario (de 'marca') cuando el Circuito CF (Received Line Signal Detector) esté en condición OFF.

En un canal half duplex, el Circuito BB debe mantener la condición de UNO binario (de marca) cuando el Circuito CA (Request to Send) esté en condición ON y por un breve intervalo después de la transición de ON a OFF del Circuito CA para permitir la terminación de la transmisión (ver Circuito BA - Transmitted Data) y la suspención de reflexiones de línea. Ver sección A.4.3 para características de la señal.

CIRCUITO CA - REQUEST TO SEND (C.C.I.T.T. 105)
Dirección: Al equipo de comunicación de datos.

Este circuito se usa para condicionar el equipo de comunicación de datos local para transmisión de datos y, en un canal half duplex, controlar la dirección de transmisión de datos del equipo de comunicación de datos local.

En canales de un sentido o duplex, la condición ON mantiene al equipo de comunicación de datos en modo de transmisión; la condición OFF lo mantiene en modo de no-transmisión.

En un canal half duplex, la condición ON mantiene al equipo de comunicación de datos en modo de transmisión e inhibe el modo receptor; la condición de OFF lo mantiene en modo receptor.

Una transición de OFF a ON instruye al equipo de comunicación de datos introducir el código de transmisión. El equipo de comunicación de datos responde tomando tantas acciones como necesite e indica la terminación de tales acciones cambiando en ON al Circuito CB (Clear to Send), indicando al equipo terminal de datos que los datos serán transferidos a través de la interfaz sobre el Circuito BA (Transmited Data).

Una transisión de ON a OFF instruye al equipo de comunicación de datos la terminación de la transmisión de todos los datos, los cuales fueron previamente transferidos a través de la interfaz sobre el Circuito BA y entonces asume el modo de no-transmisión o de receptor. El equipo de comunicación de datos responde a esta instrucción cambiando a OFF al Circuito CB (Clear to Send) cuando éste esté preparado para responder otra vez a una subsecuente condición ON del Circuito CA.

NOTA: Un modo no-transmisión no implica que todas las señales de línea han sido removidas desde el canal de comunicación.

Cuando el Circuito CA se cambia a OFF, no debe ser regresado a ON otra vez hasta que el Circuito CB ha sido cambiado a OFF por el equipo de comunicación de datos.

Se requiere una condición de ON en el Circuito CA e inclusive en el Circuito CB, Circuito CC (Data Set Ready) y, si fue incluido, el Circuito CD (Data Terminal Ready), siempre que el equipo terminal de datos los tranfiera a través de la interfaz sobre el circuito BA.

Se permite cambiar el Circuito CA a ON cuando el Circuito CB esta en OFF sin importar la condición de cualquier otro circuito de intercomunicación.

CIRCUITO CB - CLEAR TO SEND (C.C.I.T.T. 106)
Dirección: Del equipo de comunicación de datos.

Las señales en este circuito están generadas por el equipo de comunicación de datos para indicar si el conjunto de datos está listo para transmitirse.

La condición de ON junto con la condición de ON en los Circuitos CA, CC y, si fue incluído, CD, es una indicación para el equipo terminal de datos que las señales presentadas en el Circuito BA (Transmited Data) serán transmitidas al canal de comunicación.

La condición OFF es una indicación para el equipo terminal de datos que no debe transferir datos a través de la interfaz en el Circuito BA.

La condición ON del Circuito CB es una respuesta a la ocurrencia de condiciones simultáneas de ON del Circuito CC (Data Set Ready) y del Circuito (Request to Send), retrasada como sea necesario para que el equipo de comunicación de datos establezca un canal de comunicación (incluyendo la eliminación de "MARK HOLD" del circuito de intercomunicación Received Data del conjunto de datos remotos) al equipo terminal remoto de datos.

Si el Circuito CA (Request to send) no está incluido en el equipo de comunicación de datos con capacidad de transmisión, debe permanecer en condición ON y el Circuito CB en consecuencia deberá responder.

CIRCUITO CC -DATA SET READY (C.C.I.T.T. 107)
Dirección: Del equipo de comunicación de datos.
Las señales en este circuito son usadas para indicar el estado del conjunto local de datos

La condición ON en este circuito se presenta para indicar que:

1. El equipo local de comunicación de datos está conectado a un canal de comunicación ("OFF HOOK" en servicio conmutado),

2. El equipo local de comunicación de datos no está en prueba (local o remota), en charla (voz alternativa) o modo de sintonización.

3. El equipo local de comunicación de datos a terminado si:

   a) Cualquier función sincronizada requerida por el sistema de conmutación para completar el establecimiento de la llamada, y

   b) La transmisión de cualquier tono de respuesta en forma discreta, la duración de la cual está controlada únicamente por el conjunto de datos locales.

En donde el equipo local de comunicación de datos no transmite un tono de respuesta, o en donde la duración del tono de respuesta está controlado por alguna acción del conjunto de datos remotos, la condición ON se presenta tan pronto como todas las condiciones anteriores son satisfechas.

Este circuito debe ser usado solamente para indicar el estado del conjunto de datos locales. La condición ON no debe ser interpretada como una indicación de que el canal de comunicación ha sido establecido a una estación de datos

remotos o el estado de cualquier equipo de estación remota.

La condición OFF debe ser permanente y debe ser una indicación de que el equipo terminal de datos está ignorando las señales que aparecen en cualquier otro circuito de intercomunicación con la excepción del Circuito CE (Ring Indicator). La condición OFF no debe perjudicar la operación del Circuito CE o del Circuito CD (Data Terminal Ready).

Cuando la condición OFF ocurre durante el instante de una llamada antes de que el Circuito CD sea cambiado a OFF, el equipo terminal de datos debe interpretar esto como una pérdida o conexión abortada y tomar acción para terminar la llamada. Cualquier subsecuente condición ON en el circuito CC es considerada como una nueva llamada.

Cuando el conjunto de datos se usa juntamente con el Equipo Automático de Llamada (ACE), la transición de OFF a ON del Circuito CC no debe ser interpretada como una indicación de que éste ha renunciado al control del canal de comunicación para el conjunto de datos. La indicación para esto está dada por una carga apropiada en la interfaz del ACE (ver el estándar de RS-366).

NOTA:Se manda una atención si una llamada de dato se interrumpe por una comunicación de voz alterna; el circuito CC estará en condición OFF durante el tiempo en que la comunicación de voz esté en marcha. La transmisión o recepción de las señales requeridas de condición de canal de comunicación o del equipo de comunicación de datos en respuesta a la condición ON de intercambio del circuito CA (Request to Send) de la transmisión del equipo terminal de datos tomará lugar después de que el circuito CC llegue en ON, pero antes de la condición ON en el circuito CB (Clear to Send) o en el circuito CF (Received Line Signal Detector).

CIRCUITO CD -DATA TERMINAL READY (C.C.I.T.T. 108.2)
Dirección: Al equipo de comunicación de datos.

Se usan señales en este circuito para controlar el "switcheo" del equipo de comunicación de datos al canal de comunicación. La condición ON prepara al equipo de comunicación de datos a ser conectado al canal de comunicación y mantiene la conexión establecida para recursos externos (p.ej., llamada de origen manual, llamada de origen automático o respuesta manual).

Cuando la estación está equipada para respuesta automática de llamadas recibidas y se encuentra en modo de respuesta automática, la conexión a la línea ocurre solamente en respuesta a la combinación de una señal de timbre y la

condicion ON del circuito CD (Data Terminal Ready). Sin embargo, el equipo terminal de datos es normalmente permitido para presentar la condición ON en el circuito CD, siempre que esté listo para transmitir o recibir datos, excepto como se indicó anteriormente.

La condición OFF provoca al equipo de comunicación de datos a ser quitado del canal de comunicación, acatando la terminación de cualquier transmisión en proceso. Ver circuito BA (Transmitted Data). La condición OFF no imposibilitará la operación del circuito CE (Ring Indicator).

En aplicaciones de redes conmutadas, cuando el circuito CD se regresa a OFF, éste no regresará a ON de nuevo hasta que el circuito CC (Data Set Ready) se regrese a OFF por medio del equipo de comunicación de datos.

CIRCUITO CE -RING INDICATOR (C.C.I.T.T.125)
Dirección: Del equipo de comunicación de datos.

La condición ON de este circuito indica que una señal de timbre está siendo recibida en el canal de comunicación.

La condición OFF aparecerá aproximadamente a la vez con el segmento ON del ciclo de timbre (durante timbres) en el canal de comunicación.

La condición OFF será mantenida durante el segmento OFF del ciclo de timbre (entre 'timbres') y en todos los tiempos cuando el timbre no ha sido recibido. La operación de este circuito no será imposibilitado por la condición OFF en el circuito CD (Data Terminal Ready).

CIRCUITO CF -RECEIVED LINE SIGNAL DETECTOR (C.C.I.T.T. 109).
Dirección: Del equipo de comunicación de datos.

La condición ON en este circuito se presenta cuando el equipo de comunicación de datos está recibiendo una señal que reconoce ciertos criterios. Estos criterios los establece el fabricante del equipo de comunicación de datos.

La condición OFF indica que no se está recibiendo señal o que la señal recibida no es congruente para la demodulación.

La condición OFF del circuito CF(Received Line Signal Detector) provocará al circuito BB(Received Data) ser fijado a Uno Binario (condicion de marca).

Las indicaciones en este circuito ejercerán la operación

actual o pérdida de la señal con un determinado retardo de respaldo

En canales half-duplex, el circuito CF se mantiene en condición OFF siempre que el circuito CA (Request to Send) esté en condición ON y para un breve intervalo de tiempo, siguiendo la transición de ON a OFF del circuito CA. (Ver circuito BB.)

CIRCUITO CG -SIGNAL QUALITY DETECTOR (C.C.I.T.T. 110)
Dirección: Del equipo de comunicación de datos.

Se usan señales en este circuito para indicar si existe o no una alta probabilidad de error en el dato recibido.

Una condición ON se mantiene siempre que no haya razón para creer que un error ha ocurrido.

Una condición OFF indica que hay una alta probabilidad de un error. Esto puede, en algunas ocasiones, ser usado para llamar automáticamente a la retransmisión de una señal de dato previamente transmitida. Mejor dicho, la respuesta de este circuito será tal como se permita la identificación de elementos individuales de la señal cuestionable en el circuito BB (Received Data).

CIRCUITO CH - DATA SIGNAL RATE SELECTOR (DTE SOURCE)
(C.C.I.T.T. 111)
Dirección: Al equipo de comunicación de datos.

Las señales en este circuito se usan para seleccionar entre dos velocidades de señales de datos en el caso de conjuntos de datos síncronos de velocidad dual o de dos rangos de velocidades de señales de datos en el caso de conjuntos de datos no síncronos de rango dual.

Una condición ON seleccionará la velocidad de la señal de dato más alta o el rango de velocidades.

La velocidad de las señales de sincronización en el tiempo,si se incluyen en la interfaz, será controlada por este circuito apropiadamente.

CIRCUITO DA -TRANSMITTER SIGNAL ELEMENT TIMING (DTE SOURCE)
(C.C.I.T.T. 113)
Dirección: Al equipo de comunicación de datos.

Las señales en este circuito se usan para proveer la conversión de la señal de transmisión con el elemento de la señal de información sincronizada.

La transición de ON a OFF nominalmente indicará el centro de cada elemento de la señal en el circuito BA (Transmitted

Data). Cuando el circuito DA se incluye en el DTE, el DTE normalmente proveerá información sincronizada en el circuito siempre que DTE esté en condición POWER ON. Esto se permite para que el DTE sostenga la información sincronizada en el circuito por periodos cortos provistos por el circuito CA (Request to Send) estando en la condición OFF. (Por ejemplo, el mantener temporalmente la información sincronizada puede ser necesario en la ejecución de encuestas de mantenimiento con el DTE.)

CIRCUITO DB -TRANSMITTER SIGNAL ELEMENT TIMING (DCE SOURCE) (C.C.I.T.T. 114)
Dirección: Del equipo de comunicación de datos.

Las señales en este circuito se usan para proveer al equipo terminal de datos con el elemento de la señal de la información sincronizada. El equipo terminal de datos proveerá una señal de dato en el circuito BA (Transmitted Data), en el cual las transiciones entre elementos de la señal nominalmente ocurren en el tiempo de las transiciones de OFF a ON de la señal en el circuito DB. Cuando el circuito DB se incluye en el DCE, el DCE normalmente proveerá información sincronizada en este circuito siempre que el DCE esté en condición POWER ON. Esto se permite para que el DCE sostenga información sincronizada en este circuito por periodos cortos provistos por el circuito CC (Data Set Ready) estando en condición OFF. (Por ejemplo, el sostenimiento temporal de información sincronizada puede ser necesario en la ejecución de encuestas de mantenimiento con el DCE.)

CIRCUITO DD -RECEIVER SIGNAL ELEMENT TIMING (DCE SOURCE) (C.C.I.T.T. 115)
Dirección: Del equipo de comunicación de datos.

Las señales en este circuito se usan para proveer el equipo terminal de datos con el elemento de la señal de la información sincronizada recibida. La transición de ON a OFF nominalmente indicará el centro de cada elemento de la señal en el circuito BB (Received Data). La información sincronizada en el circuito DD será provista en todas las veces cuando el circuito CF (Received Line Signal Detector) esté en condición ON. Esto puede seguir de la transición ON a OFF del circuito CF (Ver seccion A.4.3).

CIRCUITO SBA -SECONDARY TRANSMITTED DATA (C.C.I.T.T. 118)
Dirección: Al equipo de comunicación de datos.

Este circuito es equivalente al circuito BA (Transmitted Data), excepto que éste se usa para transmitir datos vía canal secundario.

Las señales en este circuito son generadas por el equipo

terminal de datos y conectadas al canal local secundario transmitiendo la señal convertida para transmisión de datos al equipo terminal remoto de datos.

El equipo terminal de datos mantendrá el circuito SBA (Secondary Transmitted Data) en condición de marca durante intervalos entre caracteres o palabras y cada vez que los datos no estén siendo transmitidos.

En todos los sistemas, el equipo terminal de datos no transmitirá datos en el canal secundario a menos que una condición ON esté presente en todos los siguientes cuatro circuitos, cuando éstos se incluyen:

1. Circuito SCA -Secondary Request to Send

2. Circuito SCB -Secondary Clear to Send

3. Circuito CC -Data Set Ready

4. Circuito CD -Data terminal Ready.

Todas las señales de datos que son transmitidas a lo largo de la interfaz en el circuito de intercambio SBA en el momento en que las condiciones establecidas son satisfechas serán transmitidas al canal de comunicación. Ver sección A.4.3.

Cuando el canal secundario se usa sólo para circuito de seguridad o para interrumpir el flujo de datos en el canal primario (capacidad menor de 10 baud), el circuito SBA (Secundary Transmitted Data) está normalmente no probado, y el portador del canal cambia a ON u OFF por medio del significado del circuito SCA (Secondary Request to Send). El portador OFF es interpretado como una condición de "Interrupción".

CIRCUITO SBB -SECONDARY RECEIVED DATA (C.C.I.T.T. 119)
Dirección: Del equipo de comunicación de datos.

Este circuito es equivalente al circuito BB (Received Data) pero éste se usa para recibir datos en el canal secundario.

Cuando el canal secundario se usa solo para circuito de seguridad o para interrumpir el flujo de datos en el canal primario, el circuito SBB está normalmente probado. Ver circuito de intercambio SCF (Secondary Received Line Signal Detector).

CIRCUITO SCA -SECONDARY REQUEST TO SEND (C.C.I.T.T. 120)
Dirección: Al equipo de comunicación de datos.

Este circuito es equivalente al circuito CA (Request to

Send) excepto que éste requiere el establecimiento del canal secundario de datos en lugar del requerimiento del canal primario de datos.

Cuando el canal secundario de datos se usa como un canal de respaldo, la condición ON del circuito CA (Request to Send) deberá deshabilitar al circuito SCA y no será posible condicionar la señal de transmisión del canal secundario convertida para transmitir durante cualquier intervalo de tiempo cuando la señal transmitida del canal primario está así condicionada. Cuando las consideraciones del sistema dicten que uno u otro de los dos canales estarán en modo de transmisión a todo tiempo pero nunca ambos simultaneamente, esto puede ser acompañado por la aplicación permanente de la condición ON al circuito SCA (Secondary Request to Send) y controlando ambos, el canal primario y el secundario, en forma complementaria, a través del circuito CA (Request to Send). Alternativamente, en este caso, el circuito SCB necesita no ser implementado en la interfaz.

Cuando el canal secundario se usa sólo para circuito de seguridad o para interrumpir el flujo de datos en el canal primario de datos, el circuito SCA servirá para encender (ON) el canal secundario de portadora no modulada. La condición OFF del circuito SCA apagará (OFF) la portadora del canal secundario y de ese modo señalará una condición de interrupción en el final del canal de comunicación.

CIRCUITO SCB -SECONDARY CLEAR TO SEND (C.C.I.T.T. 121)
Dirección: Del equipo de comunicación de datos.

Este circuito es equivalente al circuito CB (Clear to Send), excepto que éste indica la disponibilidad del canal primario. Este circuito no está probado cuando el canal secundario se usa sólo como un circuito de seguridad o canal de interrupción.

CIRCUITO SCF -SECONDARY RECEIVED LINE SIGNAL DETECTOR (C.C.I.T.T. 122)
Dirección: Del equipo de comunicación de datos.

Este circuito es equivalente al circuito CF (Received Line Signal Detector) a excepción de que éste indica la recepción propia de la señal de la línea del canal secundario en lugar de la señal de la línea del canal primario.

Cuando el canal secundario se usa sólo como un circuito de seguridad o como un canal de interrupciones (ver circuito SCA -Secondary Request to Send), el circuito SCF deberá ser usado para indicar el estado del circuito de seguridad o para indicar la interrupción. La condición ON deberá indicar circuito de seguridad o una condición de no

interrupción. La condición OFF indicará falla de circuito (no seguridad) o la condición de interrupción.

APENDICE B

LINE-DRIVER MICOM MICRO 400

B.1 INTRODUCCION

El LINE-DRIVER local asincrono MICOM Micro 400 modelo 401 es un modem especializado, diseñado especialmente para transmisión de datos a corta distancia. Debido a su relativa simplicidad ofrece un ahorro considerable en el costo con respecto a modems convencionales.

Este LINE DRIVER es utilizado en líneas propias del usuario y permite una velocidad hasta de 19,200 BPS en operación FULL DUPLEX 4 hilos.

El MICRO 400 ofrece gabinete para instalación aislada ó en 'RACK' que permite tener hasta 16 módulos.

B.2 TRANSMISION DE DATOS

Las líneas privadas no tienen la misma limitación de ancho de banda que los circuitos telefónicos, por lo que no es necesario el uso de ecualizadores automáticos costosos para poder trabajar en varias velocidades de transmisión (9600, 4800, etcétera).

Los Datasets locales tienen dos grandes ventajas comparadas con los modems. La primera, es que son considerablemente más baratos y ofrecen transmisión de datos hasta de 19,200 BPS por el mismo precio que a 300 BPS. Dan por lo tanto una alta relación de bajo-costo y alta-velocidad. La segunda, pueden operar asincronamente aun a velocidades de 19,200 BPS; sin embargo, los modems comunes que trabajan en el rango de 2400 a 9600 BPS son síncronos. Por esta razón, los Datasets locales son ideales para usar terminales asincronas no inteligentes.

El DATASET local incluye filtrado para garantizar que la señal sea transmitida en la línea sin error.

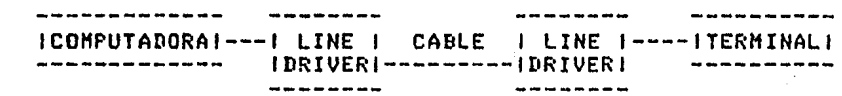
B.3 EXTENSION DE LA INTERFAZ EIA

Los LINE DRIVERS son normalmente usados para conectar terminales a una computadora a distancias de pocos cientos de metros.

La interfaz EIA RS232 especifica su operación libre de error en distancias no mayores a 15 m. Sin embargo, en la práctica, los cables de la Interfaz RS232 se usan en distancias mucho mayores, por lo que con el uso de los LINE-DRIVERS se garantiza la eliminación de problemas a operaciones en distancias mayores de 15 metros.

B.4 APLICACIONES

En el siguiente diagrama se muestra una aplicación del LINE-DRIVER.



APENDICE C

PROGRAMAS FUENTE DE VAX

```
$ INICIO:
$ W:== WRITE SYS$OUTPUT
$ W ""
$ W ""
$ W ""
$ W ""
$ W ""
$ W "            SISTEMA DE COMUNICACION VAX 11/780 - B 6800"
$ W ""
$ W ""
$ W ""
$ W " OPCIONES DEL SISTEMA :"
$ W ""
$ W ""
$ W "          1.- TERMINAL EN CANDE "
$ W ""
$ W "          2.- TRANFERENCIA DE UN ARCHIVO DE VAX A B6800 "
$ W ""
$ W "          3.- TRANFERENCIA DE UN ARCHIVO DE B6800 A VAX "
$ W ""
$ W "          4.- SALIDA DEL SISTEMA "
$ W ""
$ W ""
$ W ""
$ W ""
$ INQUIRE OPCION "DAME EL NUMERO DE LA OPCION"
$ BORRA
$ IF (OPCION .EQ. 1) THEN -
$     RUN CANDE.EXE
$ NOELSE1:
$ ENDIF1:
$ IF (OPCION .EQ. 2) THEN -
$     RUN TRANSVB.EXE
$ NOELSE2:
$ ENDIF2:
$ IF (OPCION .EQ. 3) THEN -
$     RUN TRANSBV.EXE
$ NOELSE3:
```

```
$ ENDIF3:
$ IF (OPCION .EQ. 4) THEN -
$     EXIT
$ NOELSE4:
$ ENDIF4:
$ GOTO INICIO
```

```
      program terminal

c     Programa para simular una terminal de B6800 en VAX a
c     traves del puerto ttc6:
c
c     Julio 1984                          Eduardo Jallath C.
c                                         Hector Lopez P.
c                                         Laura Sandoval M.

      implicit        none

      character       terminal_B6800*5 /'ttc6:'/,
     1                terminal_lec*2 /'tt'/,
     1                caracter*1,
     1                dato_entrada*1,
     1                line_feed*1,
     1                comando*80

      external        io$_readvblk,
     1                io$_writevblk,
     1                io$m_timed,
     1                ss$_timeout

      integer*2       term_chan_B6800,
     1                text_iosb(4),
     1                term_chan_lec,
     1                contador,
     1                posicion,
     1                i,
     1                long

      integer*4       terminador(2) ,
     1                sys$assign,
     1                sys$qiow,
     1                sys$qio,
     1                status,
     1                io_func

      logical         hay_mas_datos_de_B6800,
     1                si_leyo,
     1                fin_del_programa,
     1                hay_caracteres

      parameter       tiempo = 1

c     Se asigna el canal a B6800
      status = sys$assign (terminal_B6800,
     1                     term_chan_B6800,,)
      if (.not. status) call lib$stop (%val(status))

c     Se asigna el canal de la terminal del usuario VAX
      status = sys$assign (terminal_lec,
     1                     term_chan_lec,,)
```

```
      if (.not. status) call lib$stop (%val(status))

      terminador(1) = 0 !indica que se use la mascara
      terminador(2) = '00000000'x !sin terminador

      io_func = %loc(io$_readvblk) .or.
     1          %loc(io$m_timed)

      fin_del_programa = .false.
      do while ( .not. fin_del_programa)
c         Se lee el nombre del comando de la terminal VAX
          hay_caracteres = .true.
          contador = 1
          si_leyo = .true.
          do while (hay_caracteres)
              status = sys$qiow (,%val(term_chan_lec),
     1                   %val(io_func),
     1                   text_iosb,,,
     1                   %ref(comando(contador:contador)),
     1                   %val(1),,
     1                   %ref(terminador),,)
              if (.not.status) call lib$stop(%val(status))

              if (text_iosb(1).eq.%loc(ss$_timeout)) then
                  if (contador.eq.1) then
                      hay_caracteres = .false.
                      si_leyo = .false.
c                 noelse
                  endif
              else
                  caracter = comando(contador:contador)
c                 Pregunta si el caracter en un return
                  if (ichar(caracter).eq.13) then
                      hay_caracteres = .false.
                  else
c                     incrementa para seguir leyendo
                      contador = contador + 1
                  endif
              endif
          enddo

          if (si_leyo) then
              if ((comando(1:4) .eq. '?VAX') .OR.
                  (comando(1:4) .eq. '?vax')) then
                  fin_del_programa = .true.
              else
c                 Se manda el comando a la B6800
                  posicion = contador
                  comando(posicion:posicion) = char(13)!cr
                  status= sys$qiow(,%val(term_chan_B6800),
     1                                io$_writevblk,,,,
     1                                %ref(comando),
     1                                %val(posicion),,,,)
```

```
                if (.not. status)
     1          call lib$stop (%val(status))

c               Se lee el line-feed que manda despues de
c               recibir cada comando
                status= sys$qiow(,%val(term_chan_B6800),
     1                           io$_readvblk,
     1                           text_iosb,,,
     1                           %ref(line_feed),
     1                           %val(1),,
     1                           %ref(terminador),,,)
                if (.not. status)
     1          call lib$stop (%val(status))

            endif
c         noelse
          endif

c         Se lee si hay respuesta del comando
          hay_mas_datos_de_B6800 = .true.
          do while (hay_mas_datos_de_B6800)
              status = sys$qiow (, %val(term_chan_B6800),
     1                        %val(io_func),
     1                        text_iosb,,,
     1                        %ref(dato_entrada),
     1                        %val(1), %val(tiempo),
     1                        %ref(terminador),,)
              if (.not. status)
     1        call lib$stop (%val(status))

              if (text_iosb(1).eq.%loc(ss$_timeout)) then
                  hay_mas_datos_de_B6800 = .false.
              else
                  if ((ichar(dato_entrada).gt.31)
     1            .or.(ichar(dato_entrada).eq.10)
     1            .or.(ichar(dato_entrada).eq.13)) then
                      status = sys$qiow (,
     1                                %val(term_chan_lec),
     1                                io$_writevblk,,,,
     1                                %ref(dato_entrada),
     1                                %val(1),,,,,)
                      if (.not. status)
     1                call lib$stop (%val(status))
c                 noelse
                  endif
              endif
          enddo
      enddo
      end
```

PROGRAMAS FUENTE DE VAX

```
      program transfiere_VAX_B6800

c     Programa para tranferir archivos de VAX a B6800 por
c     el puerto ttc6:.
c
c     Para tranferir archivos debe primeramente estar en
c     una sesion de Cande en la clave donde desea se
c     tranfiera el archivo.
c
c     Julio 1984                          Eduardo Jallath C.
c                                         Hector Lopez P.
c                                         Laura Sandoval M.

      implicit        none

      character       terminal_B6800*5 /'ttc6:'/,
     1                terminal_lec*2 /'tt'/,
     1                nombre_archivo*80,
     1                prompt_pide*34
     1              /'dame el nombre del archivo en VAX:'/,
     1                linea*81,
     1                dato_entrada*80,
     1                dato_B6800*81,
     1                line_feed*1,
     1                ascii_time*23,
     1                ascii_interval*8 /'0 0:0:05'/


      external        io$_readvblk,
     1                io$_readprompt,
     1                io$_writevblk

      integer*2       term_chan_B6800,
     1                text_iosb(4),
     1                term_chan_lec,
     1                ichar,
     1                nombre_archivo_l,
     1                posicion

      integer*4       binary_interval(2),
     1                terminador_lf(2) ,
     1                sys$assign,
     1                sys$qiow,
     1                sys$qio,
     1                sys$bintim,
     1                sys$setimr,
     1                sys$waitfr,
     1                status


      write(6,13)
13    format(//,'  programa para transferir archivos de '
     1'VAX a B6800 ',//)
```

```
c       Se asigna el canal a B6800
        status = sys$assign (terminal_B6800,
        1                    term_chan_B6800,,)
        if (.not. status) call lib$stop (%val(status))

c       Se asigna el canal de la terminal del usuario VAX
        status = sys$assign (terminal_lec,
        1                    term_chan_lec,,)
        if (.not. status) call lib$stop (%val(status))

        terminador_lf(1) = 0 !indica que se use la mascara
        terminador_lf(2) = '00000400'x !mascara de line-feed

c       Se lee el nombre del archivo en VAX
c       de la terminal de VAX
        status = sys$qiow (, %val(term_chan_lec),
        1                    io$_readprompt,
        1                    text_iosb,,,
        1                    %ref(nombre_archivo),
        1                    %val(80),,,
        1                    %ref(prompt_pide),
        1                    %val(34))
        if (.not. status) call lib$stop (%val(status))
        nombre_archivo_l = text_iosb(2)

c       Se abre el archivo en VAX que se va a
c       transmitir a B6800
        nombre_archivo(nombre_archivo_l+1
        1              :nombre_archivo_l+1) = ' '!quita cr
        open(   unit = 2,
        1       file = nombre_archivo,
        1       status = 'old')

c       Se lee el nombre del archivo para B6800 de la
c       terminal de VAX
        prompt_pide = 'dame el nombre del archivo B6800:'
        status = sys$qiow (, %val(term_chan_lec),
        1                    io$_readprompt,
        1                    text_iosb,,,
        1                    %ref(nombre_archivo),
        1                    %val(80),,,
        1                    %ref(prompt_pide),
        1                    %val(33))
        if (.not. status) call lib$stop (%val(status))

        nombre_archivo_l = text_iosb(2)
        nombre_archivo(nombre_archivo_l+1:
        1              nombre_archivo_l+1) = ' '!quita cr
        dato_B6800 = '                                '

c       Se manda abrir un archivo en la B6800
        dato_B6800(1:5) = 'MAKE '
        dato_B6800(6:5+nombre_archivo_l) = nombre_archivo
```

```
      dato_B6800(6+nombre_archivo_l:6+nombre_archivo_l+13)
     1               = ' DATA;TAPE SEQ'
      posicion = 6+nombre_archivo_l+14
      dato_B6800(posicion:posicion) = char(13) !return
      status = sys$qiow (, %val(term_chan_B6800),
     1                        io$_writevblk,,,,
     1                        %ref(dato_B6800),
     1                        %val(posicion),,,,)
      if (.not. status) call lib$stop (%val(status))

c     Se lee el primer line-feed que manda despues de
c     recibir el comando
      status = sys$qiow (, %val(term_chan_B6800),
     1                        io$_readvblk,
     1                        text_iosb,,,
     1                        %ref(line_feed),
     1                        %val(1),,
     1                        %ref(terminador_lf),,)
      if (.not. status) call lib$stop (%val(status))

      dato_entrada(1:40) = '
      dato_entrada(41:80) = '
c     Se lee el '$WORKFILE VAX: DATA' de la B6800
      status = sys$qiow (, %val(term_chan_B6800),
     1                        io$_readvblk,
     1                        text_iosb,,,
     1                        %ref(dato_entrada),
     1                        %val(80),,
     1                        %ref(terminador_lf),,)
      if (.not. status) call lib$stop (%val(status))
      dato_entrada(text_iosb(2):text_iosb(2)) = ' '
      write(6,17)dato_entrada
17    format('$',a20)

      dato_entrada(1:40) = '
      dato_entrada(41:80) = '
c     Se lee el '$OK' de la B6800
      status = sys$qiow (, %val(term_chan_B6800),
     1                        io$_readvblk,
     1                        text_iosb,,,
     1                        %ref(dato_entrada),
     1                        %val(80),,
     1                        %ref(terminador_lf),,)
      if (.not. status) call lib$stop (%val(status))
      dato_entrada(text_iosb(2):text_iosb(2)) = '  '
      write(6,12)dato_entrada
12    format('$',a3)

      write(6,16)
16    format(/,8('*'),'archivo que se esta leyendo'
     1' de la VAX:',/)
c     Se lee el archivo completo hasta fin de archivo
      do while (.true.)
```

```
c               Se lee el archivo de VAX
                read(2,11,end = 99)linea
11              format(a80)
                write(6,14)linea
14              format('$',a79)
                linea(81:81) = char(13)
                dato_B6800(1:81) = linea(1:81)
c               Se manda la linea a B6800
                status = sys$qiow (, %val(term_chan_B6800),
     1                                  io$_writevblk,,,,
     1                                  %ref(dato_B6800),
     1                                  %val(81),,,,)
                if (.not. status)
     1              call lib$stop (%val(status))

c       nota: no manda line-feed la B6800 por el "tape seq"

        enddo
99      continue

c       Se lee el line-feed que manda despues de recibir
c       la ultima linea
        status = sys$qiow (, %val(term_chan_B6800),
        1                       io$_readvblk,
        1                       text_iosb,,,
        1                       %ref(line_feed),
        1                       %val(1),,
        1                       %ref(terminador_lf),,)
        if (.not. status) call lib$stop (%val(status))

c       Se lee el "$" que manda despues de recibir la ultima
c       linea
        status = sys$qiow (, %val(term_chan_B6800),
        1                       io$_readvblk,
        1                       text_iosb,,,
        1                       %ref(dato_entrada),
        1                       %val(80),,
        1                       %ref(terminador_lf),,)
        if (.not. status) call lib$stop (%val(status))

        write(6,21)
21      format(/,' P.f. espera a que cierre el archivo ',/)

c       Crea un intervalo de tiempo en binario de 5 seg.
        status = sys$bintim (ascii_interval,
        1                    binary_interval)
        if (.not. status) call lib$stop (%val(status))

c       Encola el requerimiento de tiempo
        status = sys$setimr (%val(1),
        1                    binary_interval,,)
        if (.not. status) call lib$stop (%val(status))
```

```
c       Espera se prenda el event flag
        status = sys$waitfr (%val(1))
        if (.not. status) call lib$stop (%val(status))

        dato_B6800 = '
c       Se manda un "SAVE" al archivo en la B6800
        dato_B6800(1:4) = 'SAVE'
        dato_B6800(5:5) = char(13) !return
        posicion = 5
        status = sys$qiow (, %val(term_chan_B6800),
     1                          io$_writevblk,,,,
     1                          %ref(dato_B6800),
     1                          %val(posicion),,,,)
        if (.not. status) call lib$stop (%val(status))

c       se lee el primer line-feed que manda despues de
c       recibir el comando save
        status = sys$qiow (, %val(term_chan_B6800),
     1                          io$_readvblk,
     1                          text_iosb,,,
     1                          %ref(line_feed),
     1                          %val(1),,
     1                          %ref(terminador_lf),,)
        if (.not. status) call lib$stop (%val(status))

        dato_entrada(1:40) = '
        dato_entrada(41:80) = '
c       Se lee el "$UPDATING" de la B6800
        status = sys$qiow (, %val(term_chan_B6800),
     1                          io$_readvblk,
     1                          text_iosb,,,
     1                          %ref(dato_entrada),
     1                          %val(80),,
     1                          %ref(terminador_lf),,)
        if (.not. status) call lib$stop (%val(status))
        dato_entrada(text_iosb(2):text_iosb(2)) = ' '
20      format('$',a10)
        write(6,20)dato_entrada

        dato_entrada(1:40) = '
        dato_entrada(41:80) = '
c       Se lee el "$workfile VAX saved" de la B6800
        status = sys$qiow (, %val(term_chan_B6800),
     1                          io$_readvblk,
     1                          text_iosb,,,
     1                          %ref(dato_entrada),
     1                          %val(80),,
     1                          %ref(terminador_lf),,)
        if (.not. status) call lib$stop (%val(status))
        dato_entrada(text_iosb(2):text_iosb(2)) = '  '
        write(6,19)dato_entrada
19      format('$',a22)
        write(6,22)
```

```
22      format(//,' ********** transledo concluido '
        1'**********',/)
        end
```

```
      program transfiere_B6800_VAX

c     Programa para tranferir archivos de B6800 a VAX por
c     el puerto tta6:
c
c     Para transferir archivos debera primeramente estar
c     conectado y en una sesion de conde.
c
c     Julio 1984                          Eduardo Jallath C.
c                                         Hector Lopez P.
c                                         Laura Sandoval M.

      implicit        none
      character       terminal_B6800*5 /'tta6:'/,
      1               terminal_lec*2 /'tt'/,
      1               nombre_archivo_B6800*80,
      1               nombre_archivo_VAX*80,
      1               line_feed*1,
      1               prompt_ride*36,
      1               dato_ent*97,
      1               dato_B6800*97

      external        io$_readvblk,
      1               io$_readprompt,
      1               io$_writevblk

      integer*2       term_chan_B6800,
      1               text_iosb(4),
      1               term_chan_lec,
      1               ichar,
      1               nombre_archivo_B6800_l,
      1               nombre_archivo_VAX_l,
      1               pos,
      1               i,
      1               contador,
      1               long
      integer*4       terminador(2) ,
      1               sys$assign,
      1               sys$qiow,
      1               sys$qio,
      1               status

      write(6,11)
11    format(/,'   Programa para transferencia de'
      1' archivos de B6800 a VAX',//)

c     Se asigna el canal a B6800
      status = sys$assign (terminal_B6800,
      1                    term_chan_B6800,,)
      if (.not. status) call lib$stop (%val(status))

c     Se asigna el canal de la terminal del usuario VAX
      status = sys$assign (terminal_lec, term_chan_lec,,)
```

```
      if (.not. status) call lib$stop (%val(status))

      terminador(1) = 0 !indica que se use la mascara
      terminador(2) =  '00000401'x !mascara del line-feed

c     Se lee el nombre del archivo en B6800 de la
c     terminal de VAX
      prompt_pide = 'dame el nombre del archivo en B6800:'
      status  =  sys$qiow (, %val(term_chan_lec),
     1                        io$_readprompt,
     1                        text_iosb,,,
     1                        %ref(nombre_archivo_B6800),
     1                        %val(80),,,
     1                        %ref(prompt_pide),%val(36))
      if (.not. status) call lib$stop (%val(status))

c     para meter el list:u y el return
      nombre_archivo_B6800_l = text_iosb(2)

      write(6,13)
13    format(//)
c     Se lee el nombre del archivo VAX de la terminal
c     del usuario en VAX
      prompt_pide = 'dame el nombre del archivo en VAX:'
      status  =  sys$qiow (, %val(term_chan_lec),
     1                        io$_readprompt,
     1                        text_iosb,,,
     1                        %ref(nombre_archivo_VAX),
     1                        %val(80),,,
     1                        %ref(prompt_pide),
     1                        %val(34))
      if (.not. status) call lib$stop (%val(status))

      nombre_archivo_VAX_l = text_iosb(2)
      nombre_archivo_VAX(nombre_archivo_VAX_l+1:
     1                   nombre_archivo_VAX_l+1) = ' '
c     Se abre el archivo en VAX donde quedaran los datos
c     que seran transmitidos de la B6800
      open(   unit = 2,
     1        file = nombre_archivo_VAX,
     1        status = 'new')

c     Se manda un "?-P" para que no mande .PAGE.
      pos = 4
      dato_B6800(1:3) = '?-P'
      dato_B6800(pos:pos) = char(13)
      status = sys$qiow (, %val(term_chan_B6800),
     1                     io$_writevblk,,,,
     1                     %ref(dato_B6800),
     1                     %val(pos),,,,)
      if (.not. status) call lib$stop (%val(status))

c     Se lee el primer line-feed
```

```
      status = sys$qiow (, %val(term_chan_B6800),
     1                        io$_readvblk,
     1                        text_iosb,,,
     1                        %ref(line_feed),
     1                        %val(1),,
     1                        %ref(terminador),,)
      if (.not. status) call lib$stop (%val(status))

      dato_ent(1:40) = '
      dato_ent(41:80) = '
c     Se lee el '.PAGE RESET' de la B6800
      status = sys$qiow (, %val(term_chan_B6800),
     1                        io$_readvblk,
     1                        text_iosb,,,
     1                        %ref(dato_ent),
     1                        %val(80),,
     1                        %ref(terminador),,)
      if (.not. status) call lib$stop (%val(status))
      long = text_iosb(2)-1
      write(6,17)dato_ent
17    format(//,' ',a80)

c     se manda listar el archivo de la B6800
      pos = nombre_archivo_B6800_1
      dato_B6800(1:7) = 'list:u '
      dato_B6800(8:8+pos-1) = nombre_archivo_B6800(1:pos)
      dato_B6800(8+pos:8+pos) = char(13) !return
      pos = 8+pos
      status = sys$qiow (, %val(term_chan_B6800),
     1                        io$_writevblk,,,,
     1                        %ref(dato_B6800),
     1                        %val(pos),,,,)
      if (.not. status) call lib$stop (%val(status))

      dato_ent(1:40) = '
      dato_ent(41:80) = '
c     Se lee el primer line-feed que manda antes de
c     listar el archivo
      status = sys$qiow (, %val(term_chan_B6800),
     1                        io$_readvblk,
     1                        text_iosb,,,
     1                        %ref(line_feed),
     1                        %val(1),,
     1                        %ref(terminador),,)
      if (.not. status) call lib$stop (%val(status))

      write(6,16)
16    format(' archivo de la B6800:',/)
      contador = 0
c     se lee el archivo completo hasta leer un '#'
      do while (dato_ent(1:2).ne.'# ')
              dato_ent(1:40)  = '
              dato_ent(41:80) = '
```

```
            status = sys$qiow (, %val(term_chan_B6800),
     1                          io$_readvblk,
     1                          text_iosb,,,
     1                          %ref(dato_ent),
     1                          %val(97),,
     1                          %ref(terminador),,)
            if(.not.status) call lib$stop (%val(status))
            long = text_iosb(2)
            contador = contador +1
            if (long .gt. 0) then
c                       para quitar el cr y lf
                        dato_ent(long:long+1)='  '
c                       hace el eco a la terminal de VAX
                        write(6,18)dato_ent
18                      format(' ',a80)
                        if (.not.((contador.eq.1).or.
     1                     (dato_ent(1:2).eq.'# '))) then
                           write(2,19)dato_ent
19                         format(a80)
                        !noelse
                        endif
            !noelse
            endif
      enddo
      contador = contador - 2
      if (contador.gt.0) then
         close(2)
         write(6,14)contador
14       format(/,' translado concluido con ',
     1  i5,' registros',/)
      else
         write(6,15)
15       format(/,' el archivo solicitado no existe')
      endif

c     Se manda un "?+P" para que regrese el .PAGE. normal
      pos = 4
      dato_B6800(1:3) = '?+P'
      dato_B6800(pos:pos) = char(13)
      status = sys$qiow (, %val(term_chan_B6800),
     1                          io$_writevblk,,,,
     1                          %ref(dato_B6800),
     1                          %val(pos),,,,)
      if (.not. status) call lib$stop (%val(status))

c     Se lee el line-feed
      status = sys$qiow (, %val(term_chan_B6800),
     1                          io$_readvblk,
     1                          text_iosb,,,
     1                          %ref(line_feed),
     1                          %val(1),,
     1                          %ref(terminador),,)
      if (.not. status) call lib$stop (%val(status))
```

```
dato_ent(1:40) = '
dato_ent(41:80) = '
se lee el ',PAGE SET' de la B6800
status = sys$qiow (, %val(term_chan_B6800),
1                          io$_readvblk,
1                          text_iosb,,,
1                          %ref(dato_ent),
1                          %val(80),,
1                          %ref(terminador),,)
if (.not. status) call lib$stop (%val(status))
long = text_iosb(2)-1
write(6,17)dato_ent
end
```

APENDICE D

PROGRAMA FUENTE DE BURROUGHS

```
begin

comment Este es un MCS basico para la comunicacion de
        mensajes entre las computadoras VAX 11/780 del
        Cecafi y la Burroughs 6800 de  la Direccion de
        Computo para la Administracion Academica del Puc

        Jul-1984                                Eduardo Jallath C.
                                                Hector Lopez P.
                                                Laura Sandoval M.
endcomment;


file impre(kind=disk,
           title="mcs/errors.",
           newfile=true,
           maxrecsize=14,
           protection=save );
message msg,
        errmsg;
queue prima,
      curra;
real rslt,
     lsnr,
     msgsize,
     waitindx;
array dumparray [0:5];
boolean eoj,
        err;
integer i;
define  classf  = msg[0].[47:8]#,
        typef   = msg[0].[47:8]#,
        terminal = 37#,
        terminalcecafi = 26#,
        noelse = #,
        endif  = #,
        lsnrf   = msg[0].[22:23]#,
        msgsz   = msg[2].[39:16]#,
```

```
        msgheader = <6(h12,x1)>,for i:=0 step 1 until 5 do
                    msg[i]#,
        movemsg = replace pointer(dumparray[0]) by pointer
                    ( msg[0] ) for 6 words #,
        textarea = msg[6]#;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                           %
%           procedure que maneja la current-queue           %
%                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

boolean procedure handlecurrentq;
   begin
        message msg,errmsg;
        remove(msg, currq);
        lsnr := lsnrf;
        case classf of
           begin
                0:  begin
                    if lsnr = terminal then
                       begin
                       lsnrf:= terminalcecafi;
                       lsnr := terminalcecafi;
                       end
                    else
                       if lsnr = terminalcecafi then
                          begin
                          lsnrf:= terminal;
                          lsnr := terminal;
                          end
                        noelse
                        endif
                    endif;
                    msg[0].[24:01]:=1;
                    msg[0]:= msg[0] & 33[47:8]; % eco
                    if handlecurrentq :=
                       (rslt:=dcwrite(msg)>63) then
                       movemsg;
                    end;
               14:  eoj := true;         % termina el program
        else:
           if msg[0].[47:8] > 0 then  % checa si hay error
               begin
               allocate(errmsg,6);     % ready la terminal
               errmsg[0] := lsnr & 37[47:8] & 1[39:16];
               if handlecurrentq :=
                 (rslt:=dcwrite(errmss) > 63) then
                  replace pointer(dumparray[0]) by pointer
                  (msg[0]) for 6 words
               noelse
               endif;
               if msg[1].[47:8] > 3 and
```

```
                    mss[1].[47:8]< 9 then
                    begin              % checa errores de linea
                    allocate(mss,8); % pone ready la linea
                    mss[0] := lsnr & 96[47:8];
                    if handlecurrenta:=
                       (rslt:=dcwrite(mss)>63) then
                       movemss
                    noelse
                    endif;
                    end
                 noelse
                 endif;
                 end
              noelse
              endif;
     end case;
end pro handlecurrenta;

%*************************************************************
%*****                                                    **
%*****       procedure que maneja la primary-queue        **
%*****                                                    **
%*************************************************************
boolean procedure handleprimaryq;
begin
file line(kind=printer);
array rmss[0:512],
      primamss[0:47];
real type;
message mss;
integer mssize;

%************** mensaje de inicio ****************
boolean procedure greetins(lsn);
value                      lsn;
integer                    lsn;
begin
    message mss;
    allocate(mss,18);
    mss[0]:= lsn & 32[47:8] & 1[27:01];
    replace pointer(mss[6]) by
            "bienvenidos al mcs comunicador";
    mssiz:=30;
    if greetins:=(rslt:=dcwrite(mss,curro )>63) then
       movemss
    noelse
    endif;
end;

%***********  procedure de transferencia ******************
boolean procedure transferin;
begin
    message mss;
```

```
        allocate (mss,8);
        mss[0]:= lsnr;
        typef := 37; % ready station
        mss[0].[39:16] := 1;  % rty sta
        if transferin :=(rslt:=dcwrite(mss) > 63) then
            movemss
        else
            begin
            allocate (mss,6);
            mss [0] := lsnr;
            typef:= 35;  % enable input
            if transferin :=(rslt:=dcwrite(mss) > 63)then
                movemss
            else
              .   transferin :=  greeting(lsnr)
            endif;
            end
        endif;
    end pro tranferin;

    %****  procedure de mensajes del sistema ******************
    boolean procedure smmsg;
        begin
        pointer p1;
        integer textlngth;
        ebcdic array emss[0:rmss[2].[39:16]+1 ];

        textlngth:=rmss[2].[39:16];
        scan p1 : pointer(rmss[6]) for textlngth;
        textlngth until neq ' ';
        replace emss[0] by p1 : p1 for textlngth;
        if emss[0] = 'quit' then
           smmsg:=true
        else
           display(emss[0])
        endif;
    end pro sysmsg;


    %***********  start of newstatactivity  ********************
    boolean procedure newstatactivity ( lsn );
    value                               lsn;
    integer                             lsn;
       begin
       define rmssz = rmss[2].[39:16] #;
       message msg;
       ebcdic array emsg [ 0: rmssz -1];
       pointer p1;
       integer textln,
               inx1,
               mcsnum;
       truthset stoppers (' .');
       case rmss [0].[39:16] of
```

```
begin
0: begin
   scan p1:pointer( rmsg[6] ) for
   textln:rmsgsz until neq ' ';
   if textln > 0 then
      begin
      replace emsg[0] by p1:p1 for textln;
      if size ( emsg ) > 4 then
          if emsg[0] = '?mcs' then
             begin
             allocate (msg,textln);
             msg[0]:= 0 & 2[47:8];
             scan p1:emsg[4] for textln:textln-4
             until neq ' ';
             replace pointer (msg[6]) by p1:p1 for
             textln:textln until in stoppers,
             ',' for 1;
             if newstatactivity:=
                (rslt:=dcwrite(msg)>63) then
                movemsg
          else
             begin
             inx1 := msg[6].[7:08];
             mcsnum := msg[ msg[ inx1 ].
                          [15:08] ].[23:08];
             allocate (msg,8);
             msg[0]:= lsn & 45[47:8] & 1[25:01];
             msg[6].[07:08]:= mcsnum;
             if newstatactivity:=
                (rslt:=dcwrite(msg)>63) then
                movemsg
             noelse
             endif;
             end
          endif;
          end
      noelse
      endif;
      end
  else
      begin
      allocate (msg, msgsize);
      replace pointer (msg[0]) by pointer (rmsg[0])
          for msgsize words;
      msg[0] := * & 33[47:08];
      if newstatactivity :=
         (rslt:=dcwrite (msg)>63) then
          movemsg
      noelse
      endif;
      end
  endif;
  end case 0 ;
```

```
      3: newstatactivity := greeti . . (lsn);
         else;
         end;
 end;% of newstationactivity


% Este es el inicio del codigo ejecutable del manejador de
% la cola primaria

mssize := remove(msg,prima);
lsnr := msg[0].[22:23];
type := typef;
if mssize > 512 then
   mssize:=512
noelse
endif;
replace pointer(rmsg[0]) by pointer (msg[0])
        for mssize words;
case type of
    begin
     1: handleprimarya := newstatactivity(lsnr);
    16: handleprimarya := transferin;
    17: eoJ := smmsg;
    99: handleprimarya := true;
  else:
       begin
       replace pointer(primamsg) by pointer(msg) for
       min (48,mssize);
       write(IMPRE,<" primarya else case    "> );
       write(IMPRE, <8(6( h12,x1),/)>,   primamsg[*]);
       handleprimarya := true;
       end;
end case;

end of handleprimarya    ;

% Este procedure se ejcuta para descontinuar el mcs
procedure abrt;
begin
   message msg;

   allocate(msg,7);
   write(IMPRE,<"mcs descontinuado ",f5.2,/,6(h12,x1)>,rslt,
         dumparray[*]);
   msg[0] := lsnr & 33[47:8] & 1[25:1];
   if rslt:= dcwrite(msg) str 63 then
      write(IMPRE,<"eoJ">)
   noelse
   endif;
   eoJ := true;
end pro abrt;
```

%%%

```
%                                                                      %
%                    programa principal                                %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
allocate (msg:6);
msg[0]:=0;
if rslt:= dcwrite(msg,prima) > 63 then
     begin
     movemsg;
     abrt
     end
noelse
endif;

while not eoj do
    begin
    waitindx := wait(prime.ainserteventscurra.ainsertevent);
    case waitindx of
         begin
         1: err := handleprimary;
         2: err := handlecurrent;
         end;
    if err then
       abrt
    noelse
    endif;
    end
endwhile;
end program .
```

APENDICE E

PROGRAMAS FUENTE DE APPLE

```
PROGRAM INTERCOM;

(*****************************************)
(*                                       *)
(*   PROTOCOLOS DE COMUNICACION PARA     *)
(*  LA INTERCONEXION DE LOS SISTEMAS     *)
(*  APPLE II PLUS Y VAX 11/780           *)
(*                                       *)
(*             EDUARDO S. JALLATH CORIA  *)
(*             HECTOR A. LOPEZ PINEDA    *)
(*             LAURA SANDOVAL MONTANO    *)
(*****************************************)

VAR
   OPCION  :INTEGER;
   AV      :CHAR;

(*****************************************)
(*                                       *)
(*  DECLARACION DE RUTINAS EXTERNAS      *)
(*  ESCRITAS EL LENGUAJE ENSAMBLADOR     *)
(*  DEL MICROPROCESADOR 6502             *)
(*                                       *)
(*****************************************)

PROCEDURE TERAS;EXTERNAL;

PROCEDURE EMCAR(CARACT:CHAR);EXTERNAL;

PROCEDURE ENTVAX;EXTERNAL;

(*****************************************)
(*                                       *)
(* RUTINA QUE TRANSFIERE ARCHIVOS DEL    *)
(* SISTEMA VAX 11/780 AL SISTEMA         *)
(* APPLE II PLUS                         *)
(*                                       *)
(*****************************************)
```

```
PROCEDURE VAXAP;
   VAR
      NOMAP,
      NOMVAX,
      ESCRIBE    :STRING;
      REGVAX     :ARRAY[0..100] OF CHAR;
      I,
      J          :INTEGER;
      SIGUE      :BOOLEAN;
      PUERENT,
      PUERSAL    :INTERACTIVE;
      CS,
      CQ,
      CAR,
      CR,
      CZ         :CHAR;
      ARCHAP     :TEXT;
      APPLE      :FILE OF CHAR;

   BEGIN
      CR:=CHR(13);
      CZ:=CHR(26);
      CS:=CHR(19);
      CQ:=CHR(17);
      RESET(PUERSAL,'REMOUT:');
      RESET(PUERENT,'REMIN:');
      (***** SE PIDE EL NOMBRE DEL ARCHIVO EN VAX *****)
      (*****       QUE SE TRANSFERIRA              *****)
      WRITE('NOMBRE DEL ARCHIVO A TRANSFERIR:');
      READLN(NOMVAX);
      WRITE('ARCHIVO EN APPLE:');
      READLN(NOMAP);
      REWRITE(APPLE,NOMAP);
      ESCRIBE:=CONCAT('TYPE ',NOMVAX,' ');
      WRITE(PUERSAL,ESCRIBE);
      WRITELN(ESCRIBE);
      EMCAR(CR);
      CAR:=' ';
      SIGUE:= TRUE;
      (***** SE LEE EL ARCHIVO DE VAX *****)
      WHILE SIGUE DO
         BEGIN
         WHILE (CAR<>CZ) AND (I<101) DO
            BEGIN
            READ(PUERENT,CAR);
            REGVAX[I]:=CAR;
            I:=I+1;
            END;
         (*ENDWHILE*)
         EMCAR(CS);
         WRITELN(' I = ',I);
         IF I>100 THEN
            BEGIN
```

```
            FOR J:=0 TO 100 DO
               BEGIN
               WRITE(REGVAX[J]);
               WRITE(APPLE,REGVAX[J]);
               END;
            (*ENDFOR*)
            I:=0;
            END
         ELSE
           (*** SI ES CTRL Z ES FIN DE ARCHIVO ***)
            IF CAR=CZ THEN
               BEGIN
               FOR J:=0 TO I-2 DO
                  BEGIN
                  WRITE(APPLE,REGVAX[J]);
                  END;
               (*ENDFOR*)
               CLOSE(APPLE);
               SIGUE:=FALSE;
               END;
            (*NOELSE*)
            (*ENDIF*)
         (*ENDIF*)
         ENCAR(CQ);
         END;
     (*ENDWHILE*)
   END;

(****************************************)
(*                                      *)
(*  RUTINA QUE TRANSFIERE ARCHIVOS DEL  *)
(*  SISTEMA APPLE II PLUS AL SISTEMA    *)
(*  VAX 11/780                          *)
(*                                      *)
(****************************************)

PROCEDURE APVAX;
   VAR
      PUERSAL,
      PUERENT    :INTERACTIVE;
      NOMBRE,
      CREA,
      LINEA,
      NOMVAX     :STRING;
      CZ,
      CR         :CHAR;
      ARCHAP     :TEXT;
      I          :INTEGER;

   BEGIN
      CR:=CHR(13);
      CZ:=CHR(26);
      RESET(PUERSAL,'REMOUT:');
```

```
      RESET(PUERENT,'REMIN:');
      (***** SE PIDE EL NOMBRE COMPLETO DEL ARCHIVO *****)
      (*****    DE APPLE, QUE SE TRANSFERIRA        *****)
      WRITE('ARCHIVO DE APPLE:');
      READLN(NOMBRE);
      RESET(ARCHAP,NOMBRE);
      WRITE('NOMBRE EN VAX:');
      READLN(NOMVAX);
      CREA:=CONCAT('CREATE ',NOMVAX,' ');
      WRITE(PUERSAL,CREA);
      WRITELN(CREA);
      EMCAR(CR);
      ENTVAX;
      WHILE NOT EOF(ARCHAP) DO
         BEGIN
         READLN(ARCHAP,LINEA);
         WRITE(PUERSAL,LINEA);
         WRITELN(OUTPUT,LINEA);
         EMCAR(CR);
         END;
      (*ENDWHILE*)
      EMCAR(CZ);
      WRITE('FIN DE TRANSFERENCIA');
   END;

(*************************************)
(*                                   *)
(*     PROGRAMA PRINCIPAL            *)
(*                                   *)
(*************************************)

BEGIN

   OPCION:=1;
   WHILE OPCION<>4 DO
      BEGIN
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN('       INTERCOMUNICACION');
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN('  (1) TERMINAL');
      WRITELN;
      WRITELN('  (2) TRANSFERENCIA DE ARCHIVOS DE ');
      WRITELN('      APPLE A VAX                   ');
      WRITELN;
      WRITELN('  (3) TRANSFERENCIA DE ARCHIVOS DE ');
      WRITELN('      VAX A APPLE                   ');
      WRITELN;
      WRITELN('  (4) FIN DE INTERCOMUNICACION');
      WRITELN;
```

```
        WRITELN;
        WRITELN;
        WRITELN;
        WRITE(' OPCION: ');
        READLN(OPCION);
        IF OPCION = 1 THEN
           TERAS
        ELSE
           IF OPCION = 2 THEN
              APVAX
           ELSE
              IF OPCION = 3 THEN
                 VAXAP;
              (*ENDIF*)
           (*ENDIF*)
        (*ENDIF*)
        END
END.
```

```
;----------------------------;
; PROGRAMA QUE EFECTUA LA    ;
; COMUNICACION INTERACTIVA   ;
; CON EL OTRO SISTEMA.       ;
;----------------------------;

   ;--------------------------;
   ;  RUTINA QUE ALMACENA UN  ;
   ;  ARGUMENTO DE 16 BITS    ;
   ;--------------------------;
   .MACRO DIREC
   PLA
   STA %1
   PLA
   STA %1+1
   .ENDM

   ;--------------------------;
   ; PROGRAMA PRINCIPAL       ;
   ;--------------------------;
   .PROC TERAS

YSALV   .EQU 35      ;ALMACENAMIENTO TEMPORAL DEL REGISTRO Y
ANCHPAN .EQU 21      ;ANCHO DE PANTALLA
INIPAN  .EQU 22      ;LINEA INICIAL DE LA PANTALLA
IZQPAN  .EQU 20      ;COLUMNA INICIAL DE LA PANTALLA
BASB    .EQU 28
BAS4    .EQU 29
BAS2B   .EQU 2A
BAS2A   .EQU 2B
FINPAN  .EQU 18      ;LINEA FINAL DE PANTALLA
CH      .EQU 24      ;CARACTERES HORIZONTALES ESCRITOS
CV      .EQU 25      ;LINEAS QUE SE HAN ESCRITO EN PANTALLA
PASCAL  .EQU 0

        DIREC PASCAL    ;SE ALMACENA LA DIRECCION DE REGRESO
        JSR INIVAR      ;SE INICIALIZAN LAS VARIABLES
        JSR BOPAN       ;SE BORRA LA PANTALLA
        JSR INIVAR      ;SE REINICIALIZAN LAS VARIABLES

INICIO  LDA 0C0AE       ;PREGUNTA SI HAY DATO RECIBIDO
        AND #01
        BEQ ABAJO       ;SI NO LO HAY SALTA A ABAJO
        JMP LEE

ABAJO   BIT 0C000       ;PREGUNTA SI SE TECLEO UN CARACTER
        BPL INICIO      ;SI NO ES ASI VA AL INICIO

        LDA 0C000
        BIT 0C010       ;SE LEE EL CARACTER TECLEADO
        STA 10
        JSR BORRA       ;INVESTIGA SI ES LA TECLA DE DELETE
        CMP #82
```

```
        BEQ REGRESA     ;SI SE TECLEO D REGRESA A PASCAL
        JSR EMCAR       ;LLAMADA A RUTINA QUE EMITE CARACTER
        JMP INICIO      ;SALTA A INICIO

INIVAR  LDA #28
        STA ANCHPAN
        LDA #00
        STA IZQPAN
        STA CH
        LDA #02
        STA INIPAN
        STA CV          ;RUTINA QUE INICIALIZA
        LDA #00         ;LAS VARIABLES
        STA BASB
        LDA #04
        STA BASA
        LDA #18
        STA FINPAN
        RTS

BORRA   CMP #95         ;RUTINA QUE CHECA SI SE
        BNE FINBO       ;TECLEO -> PARA BORRAR
        LDA #7F         ;CARACTERES
        STA 10
FINBO   RTS

REGRESA JSR INIVAR      ;SE INICAILIZAN LAS VARIABLES
        JMP REGRE       ;ANTES DE REGRESAR A PASCAL

LEE     LDA #0A0
        LDY CH          ;SE LEE EL DATO DEL PUERTO
        STA (BASB),Y    ;SE CHECA SI ES MAYUSCULA
        LDA 0C0AF       ;Y SE ESCRIBE EN PANTALLA
        JSR MAYUS
        ORA #80
        JSR ESCPAN
        LDA #60
        LDY CH
        STA (BASB),Y
        JMP ABAJO

EMCAR   LDA 0C0AE       ;SE EMITE EL CARACTER AL
        AND #02         ;CANAL DE COMUNICACION
        BEQ EMCAR
        LDA 10
        STA 0C0AF
        RTS

MAYUS   CMP #61         ;RUTINA QUE CONVIERTE
        BCC FINMA       ;LAS LETRAS MINUSCULAS
        CMP #7B         ;EN MAYUSCULAS
        BCS FINMA
        EOR #20
```

```
FINMA   RTS

BOPAN   LDA #00         ;RUTINA QUE BORRA LA
        STA CV          ;PANTALLA
SIGLIN  JSR VTABZ
        JSR CLREOL
        INC CV
        LDA CV
        CMP FINPAN
        BCC SIGLIN
        RTS

ESCPAN  STY YSALV       ;RUTINAS DE MANEJO DE
        PHA             ;PANTALLA
        JSR VIDSAL
        PLA
        LDY YSALV
        RTS

VIDSAL  CMP #0A0
        BCS ESCAVAN
        CMP #8A
        BEQ LF
        CMP #88
        BNE RTS1
        DEC CH
        BPL RTS1
        LDA ANCHPAN
        STA CH
        DEC CH
        LDA INIPAN
        CMP CV
        BCS RTS1
        DEC CV
        LDA CV
        JSR BASCALC
        ADC IZQPAN
        STA BASB
RTS1    RTS

ESCAVAN LDY CH
        STA (BASB),Y
        INC CH
        LDA CH
        CMP ANCHPAN
        BCS LF
        RTS

BASCALC PHA
        LSR A
        AND #03
        ORA #04
        STA BASA
```

```
          PLA
          AND #18
          BCC BSCLC2
          ADC #7F
BSCLC2    STA BASB
          ASL A
          ASL A
          ORA BASB
          STA BASB
          RTS

CR1       LDA #00
          STA CH
          RTS

LF        LDA #00
          STA CH
          INC CV
          LDA CV
          CMP FINPAN
          BCC VTABZ
          DEC CV
SCROLL    LDA INIPAN
          PHA
          JSR VTABZ
SCRL1     LDA BASB
          STA BAS2B
          LDA BASA
          STA BAS2A
          LDY ANCHPAN
          DEY
          PLA
          ADC #01
          CMP FINPAN
          BCS SCRL3
          PHA
          JSR VTABZ
SCRL2     LDA (BASB),Y
          STA (BAS2B),Y
          DEY
          BPL SCRL2
          BMI SCRL1
SCRL3     LDY #00
          JSR CLEOLZ
          BCS VTAB
CLREOL    LDY CH
CLEOLZ    LDA #0A0
CLEOL2    STA (BASB),Y
          INY
          CPY ANCHPAN
          BCC CLEOL2
          RTS
```

```
VTAB    LDA CV
VTABZ   JSR BASCALC
        ADC IZQPAN
        STA BASB
        RTS

REGRE   JSR BOPAN          ;RUTINA QUE EFECTUA EL REGRESO
        LDA PASCAL+1       ;AL PROGRAMA PASCAL, DESPUES
        PHA                ;DE HABER BORRADO LA PANTALLA
        LDA PASCAL
        PHA
        RTS

        .END
```

```
        ;----------------------------;
        ; PROGRAMA QUE EMITE POR EL ;
        ; PUERTO EL CARACTER QUE    ;
        ; RECIBE COMO PARAMETRO     ;
        ;----------------------------;

            ;--------------------------;
            ;  RUTINA QUE ALMACENA     ;
            ; UN ARGUMENTO DE 16 BITS ;
            ;--------------------------;
            .MACRO DIR
            PLA
            STA %1
            PLA
            STA %1+1
            .ENDM

            ;--------------------------;
            ; PROGRAMA PRINCIPAL       ;
            ;--------------------------;

            .PROC EMCAR,1

PASCAL      .EQU 0          ;SE ALMACENA LA
            DIR PASCAL      ;DIRECCION DE REGRESO
            PLA
            STA 10
EMITE       LDA OC0AE       ;SE EMITE ELCARACTER
            AND #02
            BEQ EMITE
            LDA 10
            STA OC0AF       ;SE OBTIENE LA
            LDA PASCAL +1   ;DIRECCION DE LA
            PHA             ;LLAMADA
            LDA PASCAL
            PHA
            RTS

            .END
```

```
;-------------------------------;
; PROGRAMA QUE ESPERA A QUE EL ;
; SISTEMA VAX/VMS ESTE LISTO   ;
; PARA QUE RECIBA UN ARCHIVO   ;
; DEL SISTEMA APPLE II PLUS    ;
;-------------------------------;

      ;-------------------------;
      ; RUTINA QUE ALMACENA UN ;
      ; ARGUMENTO DE 16 BITS   ;
      ;-------------------------;
      .MACRO DIREC
      PLA
      STA %1
      PLA
      STA %1+1
      .ENDM

      ;-------------------------;
      ; PROGRAMA PRINCIPAL     ;
      ;-------------------------;
      .PROC ENTVAX

PASCAL  .EQU 0          ;SE ALMACENA LA
        DIREC PASCAL    ;DIRECCION DE REGRESO

UNO     LDA OCOAE       ;SE ESPERA UN
        AND #01         ;"CR"
        BEQ UNO
        LDA OCOAF
        CMP #0D
        BEQ DOS
        JMP UNO

DOS     LDA OCOAE       ;SE ESPERA UN
        AND #01         ;"LINE FEED"
        BEQ DOS
        LDA OCOAF
        CMP #0A
        BEQ TRES
        JMP DOS

TRES    LDA OCOAE       ;SE ESPERA UN
        AND #01         ;"LINE FEED"
        BEQ TRES
        LDA OCOAF
        CMP #CD
        BEQ SAL
        JMP TRES

SAL     LDA PASCAL +1
        PHA
        LDA PASCAL
```

```
FIN
RTS

.END
```

APENDICE F

BIBLIOGRAFIA

1. Harold S. Stone
   Microcomputer Interfacing
   Edison Wesley Publishing Company
   1982

2. Apple Computer Inc.
   Apple II Reference Manual
   1979

3. Apple Computer Inc.
   Apple Pascal
   Language Reference Manual
   1979

4. Robert Mottola
   Asembly Language Programming
   For the Apple II
   Osborne/ Mc Graw-Hill

5. Apple Computer Inc.
   Apple II Serial Interface Card (A2I008)
   Instalation and Operating Manual

6. Computer Accessories
   J13 Serial Interface
   Operater's Manual
   1983

7. Digital Equipment Corporation
VAX/VMS
Volume 5A
System Services, I/O
System Service Reference Manual
1982

8. Digital Equipment Corporation
VAX/VMS
Volume 5B
System Services, I/O
I/O User's Guide
1982

9. Digital Equipment Corporation
VAX/VMS
Volume 6A
Run-Time Library
Run-Time Library User's Guide
1982

10. Digital Equipment Corporation
VAX/VMS
Volume 6A
Run-Time Library
Run-Time Library Reference Manual
1982

11. Digital Equipment Corporation
VAX/VMS
Command Language
User's Guide
1982

12. Digital Equipment Corporation
VAX/VMS
Command Language
Reference Manual
1982

13. Digital Equipment Corporation
VAX-11 Fortran
1982

14. Digital Equipment Corporation
Stuart Wecker
Networking, Tecnical Notes
Educational Services

15. Digital Equipment Corporation
VAX, Technical Summary

16. DYNA-TEST2000
Tech Control
Instruction Manual
Dynatech Data Systems

17. Burroughs Corporation
An Introduction to Burroughs
B 6800 Systems
1980

18. Boris Dobin Rosenthal
Interconexion entre dos computadoras B-6700
Tesis Profesional
Facultad de Ciencias, UNAM

19. Burroughs Corporation
DCALGOL
Reference Manual
1984

20. Burroughs Corporation
Network Definition Language
Reference Manual
1984

21. Burroughs Corporation
On-line System Concepts
1981

22. Tom Gabriele
(TAFT) Terminal Apple with File Transfer
'Byte', Junio 1982

23. Electronic Industries Association
RS-232-C
Interface Between Data Terminal Equipment and

BIBLIOGRAFIA

Data Communication Equipment Employing Serial Binary Data Interchange
Agosto 1969

24. Transdata, S.A.
Manual descriptivo MFX-1800
Version mesa TD-75
1981

25. Micom Micro400
Local Dataset
User's Manual
Febrero 1982

26. Motorola Semiconductors Products Inc.
Asynchronous Communications Interface Adapter (ACIA)
MC6850 Hoja de datos