UNIVERSIDAD POLITECNICA DE CANARIAS.

Escuela Universitaria de Telecomunicaciones.

Proyecto fin de carrera.

TITULO : Spooler programable de comunicaciones serie / paralelo : SDC-85

ESPECIALIDAD : Equipos electrónicos.

AUTOR : Roberto F. Benítez Díaz.

TUTOR : Roberto Domínguez Rodríguez

Las Palmas de G.C. SEPTIEMBRE - 88

UNIVERSIDAD POLITECNICA DE CANARIAS

ESCUELA DE

INGENIERIA TECNICA DE TELECOMUNICACIONES.

Proyecto fin de carrera.

TITULO: Spooler programable de comunicaciones serie - paralelo: el SDC-85.

Autor :

Fdo: Roberto F. Benítez Díaz

Tutor :

Fdo: Roberto Domínguez Rodríguez

SEPTIEMBRE 1.988

INDICE.-

PROLOGO.-

Familiarización con el uP 8Ø85.-

Todo lo que en esta memoria se describe, gira en torno a un montaje de tamaño medio y bautizado con el nombre SDC-85.

Aunque para ciertos modos de funcionamiento (ya que tiene varios) no haría falta saber nada sobre lenguaje ensamblador ó compiladores, para programarlo, así como para entender en profundidad su hardware, es necesario estar bien familiarizado con la familia de circuitos integrados del microprocesador 8Ø85.

Contenido de la memoria.-

La memoria ha sido dividida en seis partes principales:

1. Introducción: aquí se explica el funcionamiento del SDC-85 y los procesos por los que pasó en su diseño.

2. Programa Monitor del SDC-85: aquí se explica el funcionamiento del programa Monitor escrito en PLM/8Ø y que permite introducir programas diseñados por el usuario.

3. Programa QILER1.PLM: aquí se muestra un programa que ha sido necesario diseñar para hacer funcionar al Sistema de Desarrollo MDS-221 como terminal debido a que había que comunicarlo con un compatible PC para la grabación en EPROM del programa Monitor del SDC-85.

4. Programa ASCBIN.BAS: aquí se muestra el programa escrito en GWBASIC para el compatible PC y que entre otras cosas convierte un fichero hexadecimal a fichero objeto a la vez que detecta errores de transmisión.

5. Hardware del SDC-85: aquí se presentan los esquemas de la circuitería utilizada en el SDC-85, así como los mapas de memoria y puertos del mismo.

6. Apéndices: por último se ha añadido como guía ó ayuda,

unos cuantos apéndices de materias supuestamente ya conocidas como : las comunicaciones RS-232c y Centronics, el Sistema operativo ISIS II, el emulador ICE-85, el Smartwork, el CREDIT y la información de los circuitos integrados.

Posibles aplicaciones.-

Quizás la mejor aplicación que se puede sacar del SDC-85 sea la de introducir programas diseñados por usuario en lenguaje ensamblador utilizando por ejemplo las rutinas del programa Monitor.

También es posible estar conectado a un ordenador con el cual se desarrolle el programa que luego, una vez depurado, sea grabado en EPROM y aplicado al SDC-85. Para ello no es necesario disponer de un Sistema de Desarrollo, aparato costoso por otro lado, sino hacer uso del programa Monitor que permite ejecutar el programa paso a paso si fuera necesario.

En cuanto a la ampliación física del SDC-85, si hubieran ganas para ello, se podrían concretar en:

- ampliación de memoria, cosa no absolutamente necesaria, pasando de 16Kbytes hasta 64Kbytes.

- creación de una tarjeta de control de video pantalla (gráfica ó de texto).

- ó la conexión a una unidad de disco (esto por ahora es inplanteable).

Medios utilizados.-

No se pueden olvidar los medios utilizados en la realización de este proyecto. En este caso se ha dispuesto de los siguientes:

- Sistema de D esarrollo MDS-221.

- Wisdom compatible PC con grabador de EPROMs.

- Impresora de texto TRS-8Ø
- Impresora gráfica.
- Terminal Hewlett Packard 2645.
- Fuente triple de alimentación PHILIPS.
- Osciloscopio Philips 2Ø Mhz.
- Frecuencímetro digital Hewlett Packard.

Tampoco se puede olvidar el software, aunque escaso, de los anteriores equipos, así como la tan deseada bibliografía.

Agradecimiento.-

Partiendo del hecho de que los medios anteriores pertenecen a la Escuela de Ingenieros Técnicos de Telecomunicaciones, hay que agradecer la colaboración prestada por su departamento de ordenadores, al profesor y tutor de este proyecto: Roberto Domínguez, a lœ maestros de laboratorio: Domingo Marrero y Manolo Chávez (laboratorio de placas) y a todos cuantos han apoyado física ó moralmente la finalización del mismo.

Por último y para acabar con este prólogo espero que esta memoria sirva para desvelar todos los secretos y misterios que el SDC-85 ha guardado durante cerca de un año que ha durado su realización.

CAPITULO 1:

Introducción.

Introducción y objetivo.-

Las siglas SDC-85: SDC-85 pretenden ser las siglas de Sistema de diseño de comunicaciones basada en la familia del microprocesador 8085 de Intel.

Similitud con el SDK-85: Su parecido en el nombre al conocido SDK-85 por los estudiantes de 4º de Telecomunicaciones, se debe también a su parecido en el funcionamiento. La principal diferencia está en que mientras el SDK-85 posee display y teclado propio, el SDC-85 necesita conectarse a un terminal y sea a través de su teclado y pantalla por donde se comunique con el usuario. Otra diferencia es la de disponer de dos canales serie RS-232c, un canal de salida paralelo (Centronics) y otro de entrada.

Como se explicará más adelante, también hay diferencias en cuanto al software ya que a los comandos del SDK-85 se le añaden gran parte de los comandos que existen en los monitores y depuradores de otros sistemas.

Breve historia del SDC-85: antes de que surgiera cualquier idea, se tenía pensado en algo totalmente práctico y con una circuitería basada en la familia del microprocesador 8085 de Intel, la cual es más fácil de manejar al disponer, aunque con diversos problemas de avería, del Sistema de Desarrollo de Intel MDS-221 así como de su información necesaria.

Teniendo en cuenta esto se pensó en hacer un Spooler de comunicaciones. Se entiende por Spooler un sistema que conectado a un ordenador es capaz de recibir y almacenar información de éste y enviar a un periférico lento tal como una impresora o un Plotter, de forma que el periférico no bloquee al ordenador por su lentitud y así éste pueda dedi-

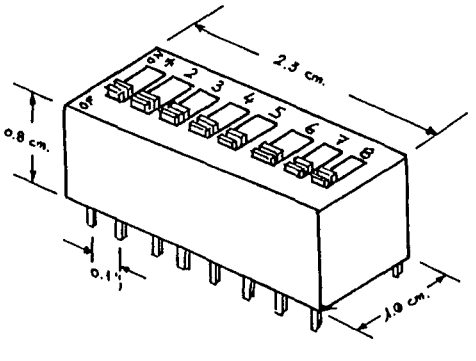
carse a otras tareas. Un buen Spooler tendría la posibilidad de configurarse, tanto en su entrada como en su salida, como serie ó paralelo, o dicho de otro modo RS-232 ó Centronics. En el resultado final, lo de Spooler es una posibilidad debido a la mayor versatilidad que se buscaba con este proyecto, aunque la idea está presente en todo el software escrito. De hecho el Spooler se puede activar ó no en el modo Monitor.

Luego se pensó, viendo que hay terminales que no están preparados para comunicarse con una impresora, que el SDC-85 que sí disponía de esta posibilidad sirviera de intermediario entre el terminal y la impresora. Esta configuración se la ha denominado modo "Terminal". Al final, por si lo anterior no fuera suficiente, se le dotó al software de un Monitor escrito en PLM/80 que le permite correr programas, tantoen tiempo real como paso a paso, y configurar el sistema.

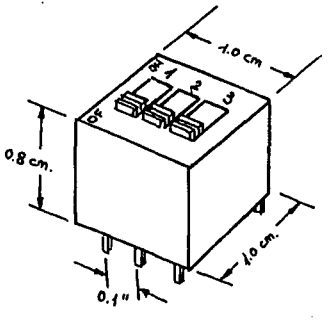
Mejora de la circuitería: nohay duda de que la circuitería mejora día a día debido a la mayor capacidad de integración. El único problema es su adquisición y no su coste económico ya que en este último sentido sería preferible haber utilizado un menor número de integrados de mayor capacidad. Tal es el caso de haber empleado memorias RAM y EPROM de 2 KBx8 (6116 y 2716), en lugar por ejemplo de 8 Kx8 (6186 y 2786) que en principio podría ser preferible en la práctica al ocupar cuatro veces menos.

Tipos de cnnfiguración.-

El SDC-85 se puede configurar e inicializar a través de sus dos micro-switch, uno de 3 microinterruptores, con los que se selecciona la velocidad de transmisión y recepción de los dos canales serie, y otro de 8 microinterruptores con los que se configura el sistema y la comunicación serie.



Microswitch de 8 microinterruptores.



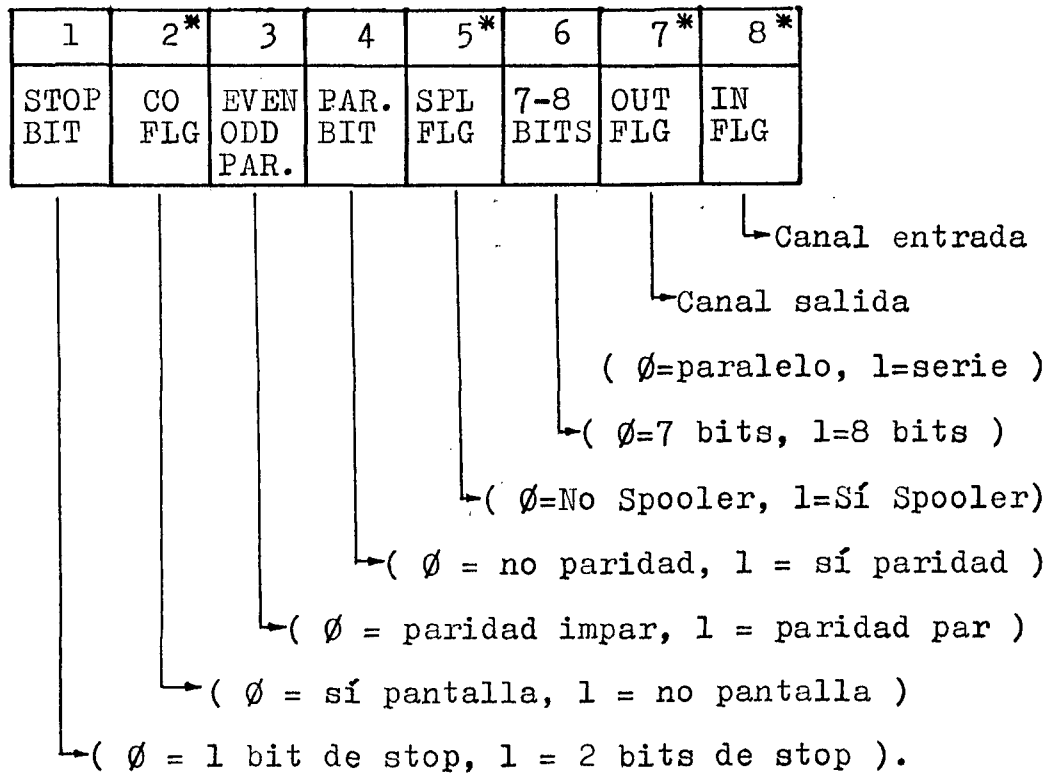
Microswitch de 3 microinterruptores.

Esto sólo tiene efecto después de un RESET hardware o RESET software (RST Ø), al ser al principio del programa cuando se leen los microswitch.

Conectar un microinterruptor en "ON" sería equivalente a poner en el puerto un "Ø" y un "OFF" sería poner un "1". La función y posición de varios de los bits de este microswitch coincide con el byte denominado "FLAG" en el programa Monitor.

A continuación se muestra la tabla de valores para la velocidad de transmisión y recepción serie, así como el esquema y nemónicos de los microinterruptores de configuración.

Microinterruptor	Valor de Port-bit
ON	" Ø "
OFF	" 1 "



Microswitch 1 2 3	Velocidad (baudios)
Ø Ø Ø	9.600
1 Ø Ø	4.800
Ø 1 Ø	2.400
1 1 Ø	1.200
Ø Ø 1	600
1 Ø 1	300
Ø 1 1	110
1 1 1	75

* El significado y posición coincide con el del byte FLAG en el programa Monitor, ya que configuran el sistema, al contrario que las demás que configuran o inicializan las dos USARTs.

- Bit 1 : Stop bit; selecciona la comunicación serie para uno (bit 1 = Ø) ó dos bits de stop (bit 1 = " 1 ").

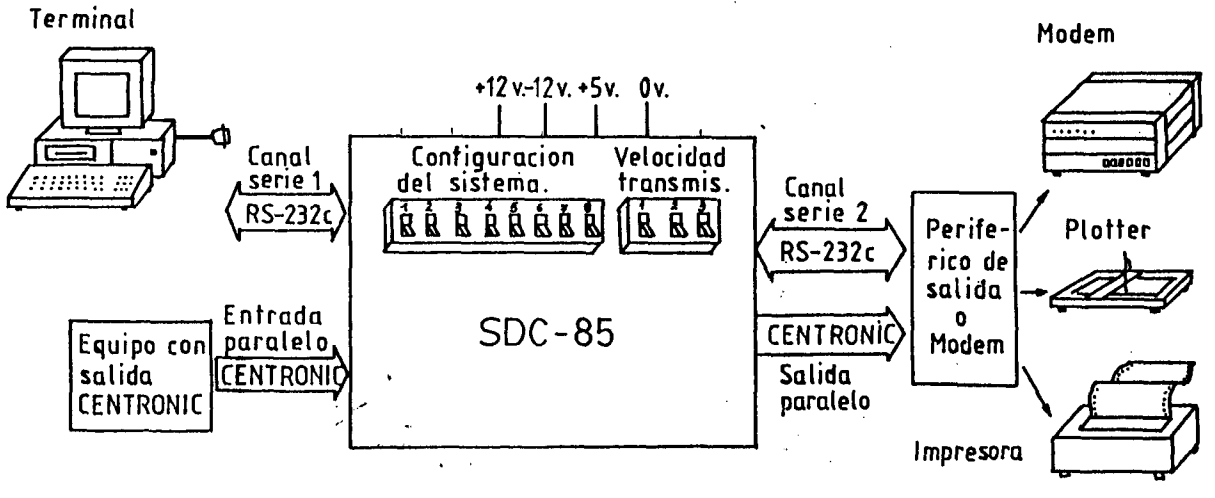
- Bit 2 : COFLG; Establece la posibilidad de utilizar la pantalla y el teclado como medio de comunicación con el

SDC-85 (bit 2 = " 1 ") ó el sistema se configura como Spoler, es decir, transmite al canal de salida (sea serie ó paralelo), lo que recibe por el canal de entrada (serie ó paralelo) sin salir por pantalla (bit 2 = " Ø ").

- Bit 3 : Paridad par / impar ; en caso de que la comunicación serie sea con paridad (bit 4 = " 1 "), se establece como par (bit 3 = " 1 ") ó impar (bit 3 = " Ø "), en casocontrario se ignora.
- Bit 4 : Paridad ; Asigna paridad (bit 4 = " 1 ") ó no (bit 4 = " Ø ") a la comunicación serie.
- Bit 5 : SPLFLG ; (Spooler flag). Activa el Spooler y su buffer de memoria (bit 5 = " 1 ") ó no (bit 5 = " Ø ").
- Bit 6 : 7 - 8 bits ; Si bit 6 = "Ø" entonces la comunicación serie es de 7 bits, en caso contrario (bit 6 = " 1 ") es de 8 bits.
- Bit 7 : OUTFLG ; asigna la salida de periférico serie (bit 7 = " 1 ") ó paralelo (bit 7 = " Ø ").
- Bit 8 : INFLG ; asigna la entrada de periférico serie (bit 8 = " 1 ") ó paralelo (bit 8 = " Ø ").

Modos de funcionamiento.-

Existen 3 modos de funcionamiento: el modo Monitor, el modo Terminal y el modo Spooler.



EQUIPOS CONECTABLES AL SDC-85

Modo Monitor: para entrar en este modo, después de un Reset, el microinterruptor nº 2 correspondiente a CO-FLG debe estar en "OFF" (bit 2 = " 1 "). Además debe estar conectado el terminal al conector DB-25 del canal serie 1, y ajustando la velocidad de transmisión en el SDC y en el terminal aparecerá el mensaje :

SDC-85 PREPARADO

*_

En este momento ya está dispuesto a ejecutar comandos del modo Monitor.

Modo Terminal: una vez dentro del modo Monitor, se entra en modo terminal pulsando CTRL - T. Este modo conecta un modem a través del canal serie 2 del SDC-85 al terminal (canal serie 1) y éste a una posible impresora (sa-

lida paralelo). Son permitidos sólo los controles que asignan los periféricos de entrada y salida como serie ó paralelo y activan el Spooler e impresora.

Modo Spooler: para entrar en este modo después de un Reset, el microinterruptor nº 2, correspondiente a COFLG debe estar en "ON" (bit 2 = "Ø"). Esto significa que no hay posibilidad de usar la pantalla como medio para visualizar la comunicación, que puede ser dependiendo de los microinterruptores ó bits de FLAG: "INFLG", "OUTFLG" y del bit de FLAG "OUTABLE":

INFLG	OUTFLG	ENTRADA	SALIDA
Ø	Ø	Paralelo	Paralelo
Ø	1	Paralelo	Serie
1	Ø	Serie	Paralelo
1	1	Serie	Serie

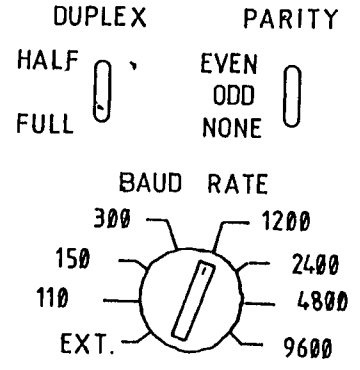
Este modo utiliza unas posiciones de memoria denominadas buffer de Spooler para almacenar temporalmente la comunicación y puede ser alterado por medio del comando Monitor: B posición inicial, posición final
que establecen el comienzo y el final del buffer en memoria. De lo contrario las posiciones por defecto son 1FØØ H a 1FFF H.

Otra forma de entrar en este modo estando en modo Monitor es haciendo un: G 909

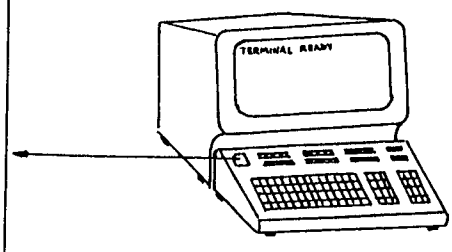
En modo Spooler no se permiten comandos de configuración como cambiar el tipo de canal de entrada ó canal de salida, ni ningún otro. Sólo está permitido salir con ESCAPE.

Característica del terminal: el programa Monitor del SDC-85 supone la conexión a un terminal de 8Ø columnas con velocidad de transmisión asíncrona. Tal es el caso del Ter

minal Hewlett Packard mod. 2645.



Interruptores de comunicacion HP.



TERMINAL HEWLET PACKARD 2645

CAPITULO 2 :

Programa Monitor del SDC-85

Comandos del Modo Monitor.-

Características generales de los comandos: los comandos están formados por una letra ASCII: A - Z, a - z y opcionalmente por 1, 2 ó 3 direcciones (2 bytes) ó por bytes.

Direcciones, son números hexadecimales entre ØØØØH y FFFF H. Bytes, son números entre ØØ H y FF H. Si se introdujeran más de 4 dígitos, sólo serían aceptados los 4 últimos (para una dirección).

ej. 12345 = 2345 H

Para un byte serían aceptados sólo los dos últimos dígitos. ej. 1234 = 34 H

Los caracteres hexadecimales son: Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9, a/A, b/B, c/C, d/D, e/E, f/F.

Los separadores válidos entre direcciones son: ","/"espacio"=" "/"return"=cr/"Line feed"=LF=ØAH/"=". El resto sería considerado como error y abortaría el comando mostrando el mensaje: *CMDERR (error en comando)
ó también: *PRMERR (error en parámetro)

- Comando B: asignación de posiciones de buffer.

formato: B direcc. inicial, direcc. final

ejemplo: *B3ØØØ,38ØØ

Al ejecutar este comando se cambian las posiciones de memoria a las que se les asigna el buffer del Spooler. Así después de este comando BUFLF = 3ØØØ H (posición más baja) y BUFHF = 38ØØ H (posición más alta). Las variables BUFHF (buffer high fijo) y BUFLF (buffer low fijo) son usadas en el programa monitor. Los valores iniciales, después de un Reset, son: BUFLF = 1FØØ H y BUFHF = 2ØØØ H

- Comando D: presentación de posiciones de memoria.

formato$_1$: D direcc. inicial, direcc. final

ejemplo: *DØ,13

```
ØØØØ: 22 D9 2Ø C3 73 ØØ ØØ ØØ
      22 D9 2Ø F5 AF 32 DC 2Ø
ØØ1Ø: El 22 D3 2Ø
```

Con este comando se presentan el contenido de las posiciones de memoria tanto ROM (memoria de sólo lectura) como RAM (memoria de lectura-escritura). En la parte izquierda, se presenta la dirección módulo 16 y a la derecha, los contenidos a partir de esa dirección.

formato$_2$: D dirección

ejemplo: *DØ

```
ØØØØ: 22 D9
```

En caso de introducir sólo una dirección, se presentan los contenidos de esa posición y la siguiente.

- Comando E: listado en ensamblador

formato: E direcc. inicial, direcc. final

ejemplo: *EØ,6

```
ØØØØ: 22 D9 ØØ        SHLD 2ØD9
ØØØ3: C3 73 ØØ        JMP  ØØ73
ØØØ6: ØØ              NOP
```

Con este comando se lista en lenguaje ensamblador del microprocesador 8085, el contenido de las posiciones de memoria empezando por la dirección inicial y hasta la dirección final.

La línea en ensamblador tiene el siguiente formato:

Direccion	:		Byte 1		Byte 2		Byte 3		Etiqueta						Operador					Operando						
Ø Ø Ø Ø	:		2	2	D	9	Ø	Ø							S	H	L	D		2	Ø	D	9			
0 1 2 3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

- Comando G: salto a programa usuario ("GO").

formato: G dirección

ejemplo: *G18ØØ

Con este comando se salta a la dirección en que el usuario tenga un programa que vaya a correr o comprobar. Para devolver el control al programa Monitor se tiene que ejecutar la instrucción "RST 1". Así permanecen intactos los registros del microprocesador y la dirección última del PC (contador de programa) antes del Restart 1 y la configuración del sistema. El siguiente programa puede servir de ejemplo:

```
18ØØ: 3E FF      MVI  A,ØFFH
18Ø2: 3D         DCR  A
18Ø3: C2 Ø2 18   JNZ  18Ø2H
18Ø6: CF         RST  1
```

- Comando H: entrada hexadecimal de memoria.

formato: H direcc.= bloque hexadec.<CTRL-Y>

ejemplo: *H18ØØ=3EFF3DC20018CF<CTRL-Y>

Con este comando se pueden introducir los contenidos de memoria a partir de una dirección y finaliza con el caracter Control-Y=19H como fin de comando. Así después de introducir el comando anterior la posición 18ØØ contiene 3E, 18Ø1H=FF, ... y 18Ø6H=CF. Al final del comando aparece en pantalla: *CHECK=<número check sum>

que es una suma de comprobación de los caracteres transmitidos. Este comando es muy útil pues conectado el SDC-85 a un ordenador con capacidad de almacenamiento en disco, puede volcar programas de éste en la RAM del sistema a gran velocidad y con la única limitación de la capacidad de memoria disponible. En este sentido se puede usar el sistema de desarrollo MDS-221 como ordenador y el QILER1, que se explicará más adelante, como programa que hace de él un terminal,

- Comando M: mover bloques de memoria.

formato: M direcc. inicial, direcc. final, cuenta

ejemplo: M18ØØ,181Ø,1Ø

Este comando mueve el bloque de memoria, que empieza en la "dirección inicial", acaba en la "dirección final" y cuya longitud la determinal el número de cuenta. Así en el ejemplo anterior se empezaría por mover el byte de 18ØØH a 181ØH, el 18Ø1H al 1811H, ... así hasta 1ØH (16d.) bytes.

- Comando P: ejecución de programas paso a paso.

formato: P direcc. inicial, direcc. final, cuenta

ejemplo: P18ØØ,Ø,3

```
                          SP   PC   A  F  B  C  D  E  H  L  I
18ØØ: ØØ       NOP        2Ø4Ø 18Ø1 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ Ø7
18Ø1: C3 ØØ 18 JMP 18ØØ  2Ø4Ø 18ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ Ø7
18ØØ: ØØ       NOP        2Ø4Ø 18Ø1 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ Ø7
```

Este comando ejecuta instrucciones paso a paso a partir de la dirección inicial y hasta que se salte a la posición final, ó se ejecute un número de instrucciones igual a "cuenta". A la vez que se va ejecutando van apareciendo en pantalla las instrucciones en lenguaje ensamblador. Para poder ver el estado de los registros en la parte derecha de la pantalla habría que pulsar CTRL-C. Aparecerá entonces el mensaje: *CTLFLG=ON

Para no listar los registros, volver a pulsar CTRL-C. Similarmente aparecerá entonces el mensaje:

*CTLFLG=OFF

- Comando p: entrada - salida por puerto.

Entrada por puertó:

formato: p nº de puerto

ejemplo: *pØØ

PORT(ØØ)=Ø8

Este comando visualiza el contenido de un puerto. Es equivalente a haber ejecutado la instrucción en ensamblador del microprocesador 8Ø85:

IN nº puerto

Salida por puerto:

formato: p nº de puerto=dato

ejemplo: *pØØ=FF

Este comando asigna el "dato" como salida del puerto. Es equivalente a haber ejecutado las intrucciones del microprocesador: MVI A, dato
OUT nº puerto

aunque sin haber modificado el valor de los registros.

- Comando R: ver contenido de registros.

formato: R

SP=2Ø4Ø PC=ØØØØ A=ØØ F=ØØ B=ØØ C=ØØ D=ØØ E=ØØ H=ØØ L=ØØ I=ØF

Este comando presenta el valor de cada uno de los registros del microprocsador 8Ø85:

SP = Stack Pointer (puntero del stack)

PC = Program Counter (contador de programa)

Registros A, F, B, C, D, E, H, L.

Registro I = máscara de interrupciones del programa.

Formato règ. I:

7	6	5	4	3	2	1	Ø
X	X	X	X	IE	M7.5	M6.5	M5.5

X = no significativo.

IE = Interrupt enable (interrupciones habilitadas).

M7.5 = máscara de interrupción 7.5

M6.5 = máscara de interrupción 6.5

M5.5 = máscara de interrupción 5.5

Los registros dobles como PC y SP se presentan en for-

mato de 4 dígitos hexadecimales (address) y el resto en formato de 2 dígitos hex. (byte).

- Comando RQ: cambiar contenido de los registros.

formato: RQ

SP=2Ø4Ø-_

Este comando permite cambiar el valor de los registros empezando por SP (stack pointer) y acabando por I (registro de interrupciones), en el mismo orden en que aparecen con el comando "R". Para ello basta con introducir el valor nuevo después del guión que lo separa del valor anterior. En caso de no querer cambiar ese registro pulsar "," ó " " y se presentará el valor del siguiente registro. Para abandonar el comando pulsar <return>.

- Comando S: substituir posiciones de memoria.

formato: S direcc., valor Ø, valor 1, ...< return>

ejemplo: *S18ØØ,ØØ-Ø1,ØØ-Ø2,ØØ-<cr>

Este comando presenta y cambia los valores de memoria a partir de una dirección. Para no asignar ningún valor en una posición basta con pulsar un separador como "," ó " ". (espacio).

- Comando T: cambiar valores de traza.

formato: T dirección 1, dirección 2

ejemplo: T181Ø,182Ø

Este comando cambia las direcciones entre las cuales se presenta en pantalla la ejecución paso a paso. Es decir, sólo se presentan las instrucciones cuyas posiciones iniciales (1er byte = código de operación) están dentro del intervalo de la traza. El resto se ejecuta pero no aparece en pantalla, con lo cual la velocidad de ejecución es mayor, a

demás de que se selecciona y comprueba la parte del programa a depurar.

CONTROLES DEL SISTEMA MONITOR DEL SDC-85

Se producen pulsando una tecla mientras se pulsa la de "CONTROL".

- Control-A: visualiza estado del sistema; bits de FLAG, buffer y traza.

INFLG=OFF OUTFLG=OFF CTLFLG=OFF SPLFLG=OFF BFFLG=ON
BEFLG=ON COFLG =ON OUTABL=OFF BUFFER=1FØØ,2ØØØ
TRACE=ØØØØ,FFFF

FLAG, es el byte del programa que está presente en todos los modos: monitor, terminal y spooler, y es el que determina prácticamente la configuración del sistema. Tiene el siguiente formato:

FLAG = .209F H.

7	6*	5	4	3*	2	1*	Ø*
OUT-ABLE	CO-FLG	BE-FLG	BF-FLG	SPL FLG	CTL FLG	OUT FLG	IN-FLG

OUTABLE: (output enable), habilita ó inhabilita la salida por impresora, es decir, indica si en el momento actual, la salida (modo monitor - terminal) va a impresora ó no, (Ø = inhabilitado, 1 = habilitado). El valor inicial es "Ø", o sea, inhabilitada la salida.

COFLG: (console output flag), indica si está activada ó no la consola de salida (pantalla). " Ø " = inhabilitada, " 1 " = habilitada. (Canal serie 1)

BEFLG: (buffer empty flag), indica si el buffer está ó no vacío, ("Ø" = buffer no vacío, "1" = buffer vacío). El estado inicial es "1", y su valor cambia al activar el

Spooler y no vaciar el buffer por estar el periférico de salida ocupado o no estar enchufado.

BFFLG:(buffer full flag) indica si el buffer está o no lleno, ("Ø" = buffer no lleno, "1" = buffer lleno). El valor inicial es "Ø".

SPLFLG*: (Spooler flag), indica si está activado ó no el Spooler. ("Ø"=OFF, "1"=ON).

CTLFLG: (control flag), indica si está activado ó no el control de pantalla. Sirve para presentar el estado de los registros cnn el comando "P", de ejecución paso a paso. ("Ø" = OFF, "1" = ON). El estado inicial es "Ø", o sea, no activado.

OUTFLG*: (output flag), indica si la salida es serie (RS-232) ó paralelo (Centronics).

paralelo: OUTFLG = Ø = OFF

serie: OUTFLG = 1 = ON

INFLG*: (input flag), indica si la entrada es serie (RS-232) ó paralelo (Centronics).

paralelo: INFLG = Ø = OFF

serie: INFLG = 1 = ON

* El valor inicial viene determinado por el microinterruptor del SDC-85.

- Control - C : activar / desactivar CTLFLG.

Si CTLFLG está activado, al pulsar este control, pasa a desactivado y se presenta el mensaje:

*CTLFLG=OFF

Por el contrario, si está desactivado, CTLFLG pasa a activado y se presenta el mensaje:

*CTLFLG=ON

- Control - D : activar / desactivar Spooler.

Si SPLFLG está activado, al pulsar este control, pasa a desactivado y presenta el mensaje:

*SPLFLG=OFF

Por el contrario si está desactivado, SPLFLG pasa a activado y se presenta el mensaje:

*SPLFLG=ON

Aún después de desactivado, lo que está en el Spooler, lo seguiría enviando, si lo hubiera, al periférico de salida activado.

- Control - I : asigna la entrada serie ó paralelo.

Si INFLG está en OFF (entrada paralelo), al pulsar este control, INFLG está en ON (entrada serie = canal 2) y se presenta el mensaje:

*INFLG=ON

Por el contrario, si está en ON, al pulsarlo, INFLG estará en OFF, y se presenta el mensaje:

*INFLG=OFF

- Control - O : asigna salida serie ó paralelo.

Si OUTFLG está en OFF (salida paralelo) al pulsar este control, OUTFLG estará en ON (salida asignada = canal serie 1) y se presenta el mensaje:

*OUTFLG=ON

Por el contrario si está en ON, al pulsarlo, OUTFLG estará en OFF y se presenta el mensaje:

*OUTFLG=OFF

- Control - P : activar periférico de salida.

Si OUTABL está desactivado, (inhabilitadas las salidas) al pulsar este control, OUTABL se activa y se presenta el mensaje: *OUTABL=ON

Por el contrario, si está activado, al pulsarlo OUTFLG se desactiva y se presenta el mensaje:

*OUTABL=OFF

- Control - Q : continuar la transmisión.

Este control reanuda la transmisión después de haber sido parada por Control-S.

- Control - S : parar la comunicación.

Este control para el envío de caracteres a la pantalla. Es efectivo en comandos de larga ejecución como lo son el comando "D" (presentar posiciones de memoria), el comando "E" (listado en lenguaje ensamblador) ó comando "P" (ejecución paso a paso). Para reanudar el envío hay que pulsar Control-Q y para abortar el comando pulsar <ESC>.

- Control - T : activar / desactivar el modo terminal.

Estando en modo Monitor, este control pasa al modo Terminal, el cual se configura conectando el canal serie 1 al terminal, el canal serie 2 al modem y la salida paralelo Centronics a la impresora.

El resto de los controles son también válidos en este modo. Para volver al modo monitor basta con volver a pulsar Control-T.

A continuación se presenta un trozo de sesión donde se compara y se hace uso del comando TRACE ("T") que selecciona el intervalo de posiciones de instrucción a presentar

con el comando paso a paso. El programa (posiciones 18ØØh a 18ØEh) es simplemente un uso de la rutina DELAY del programa Monitor.

```
 SUC-55 PREPARADO
*?
SP=207D PC=0C2A A=20 F=10 B=01 C=2A D=00 E=00 H=20 L=C4 I=07

*INFLG =OFF OUTFLG=OFF CTLFLG=OFF SPLFLG=OFF BFFLG =OFF BEFLG =ON  COFLG =ON
 OUTABL=OFF BUFFER=1F00,2000 TRACE=0000,FFFF RAMBUSY=0000
*S#
FRM-ERR
*R0
SP=207D-2040
*E1800,180F
1800: 0E 02         MVI  C,02
1802: 11 01 00      LXI  D ,0001
1805: CD F7 00      CALL 00F7
1808: 0D            DCR  C
1809: 79            MOV  A,C
180A: B7            ORA  A
180B: C2 02 18      JNZ  1802
180E: CF            RST  1

* CTLFLG=ON
*P1800,180E,FF
                                    SP   PC   A  F  B  C  D  E  H  L  I
1800: 0E 02         MVI  C,02       2040 1802 20 10 01 02 00 00 20 C4 07
1802: 11 01 00      LXI  D ,0001    2040 1805 20 10 01 02 00 01 20 C4 07
1805: CD F7 00      CALL 00F7       203E 00F7 20 10 01 02 00 01 20 C4 07
00F7: 1B            DCX  D          203E 00F8 20 10 01 02 00 00 20 C4 07
00F8: 7A            MOV  A,D        203E 00F9 00 10 01 02 00 00 20 C4 07
00F9: B3            ORA  E          203E 00FA 00 44 01 02 00 00 20 C4 07
00FA: C2 F7 00      JNZ  00F7       203E 00FD 00 44 01 02 00 00 20 C4 07
00FD: C9            RET             2040 1808 00 44 01 02 00 00 20 C4 07
1808: 0D            DCR  C          2040 1809 00 10 01 01 00 00 20 C4 07
1809: 79            MOV  A,C        2040 180A 01 10 01 01 00 00 20 C4 07
180A: B7            ORA  A          2040 180B 01 00 01 01 00 00 20 C4 07
180B: C2 02 18      JNZ  1802       2040 1802 01 00 01 01 00 00 20 C4 07
1802: 11 01 00      LXI  D ,0001    2040 1805 01 00 01 01 00 01 20 C4 07
1805: CD F7 00      CALL 00F7       203E 00F7 01 00 01 01 00 01 20 C4 07
00F7: 1B            DCX  D          203E 00F8 01 00 01 01 00 00 20 C4 07
00F8: 7A            MOV  A,D        203E 00F9 00 00 01 01 00 00 20 C4 07
00F9: B3            ORA  E          203E 00FA 00 44 01 01 00 00 20 C4 07
00FA: C2 F7 00      JNZ  00F7       203E 00FD 00 44 01 01 00 00 20 C4 07
00FD: C9            RET             2040 1808 00 44 01 01 00 00 20 C4 07
1808: 0D            DCR  C          2040 1809 00 54 01 00 00 00 20 C4 07
1809: 79            MOV  A,C        2040 180A 00 54 01 00 00 00 20 C4 07
180A: B7            ORA  A          2040 1808 00 44 01 00 00 00 20 C4 07
180B: C2 02 18      JNZ  1802       2040 180E 00 44 01 00 00 00 20 C4 07
*T1800,1810
*P1800,180E,FF
                                    SP   PC   A  F  B  C  D  E  H  L  I
1800: 0E 02         MVI  C,02       2040 1802 00 44 01 02 00 00 20 C4 07
1802: 11 01 00      LXI  D ,0001    2040 1805 00 44 01 02 00 01 20 C4 07
1805: CD F7 00      CALL 00F7       203E 00F7 00 44 01 02 00 01 20 C4 07
1808: 0D            DCR  C          2040 1809 00 10 01 01 00 00 20 C4 07
1809: 79            MOV  A,C        2040 180A 01 10 01 01 00 00 20 C4 07
180A: B7            ORA  A          2040 1808 01 00 01 01 00 00 20 C4 07
180B: C2 02 18      JNZ  1802       2040 1802 01 00 01 01 00 00 20 C4 07
1802: 11 01 00      LXI  D ,0001    2040 1805 01 00 01 01 00 01 20 C4 07
1805: CD F7 00      CALL 00F7       2040 1808 00 54 01 01 00 00 20 C4 07
1808: 0D            DCR  C          2040 1809 00 54 01 00 00 00 20 C4 07
1809: 79            MOV  A,C        2040 180A 00 54 01 00 00 00 20 C4 07
180A: B7            ORA  A          2040 1808 00 44 01 00 00 00 20 C4 07
180B: C2 02 18      JNZ  1802       2040 180E 00 44 01 00 00 00 20 C4 07
*
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE MONITOR
OBJECT MODULE PLACED IN MONITE.OBJ
COMPILER INVOKED BY:  PLM80 MONITE.PLM WORKFILES(:F0:,:F0:) NOPAGING PAGEWIDTH(100) DEBUG NOINTVECTO
                      -R


  1          MONITOR:    DO:
             /* ********************************************************** */
             /* Ejemplos del monitor en PLM/80 :                           */
             /* /,/ /<cr>/=/      separadores validos.                     */
             /* B1F00,2000        asignar zona buffer entre 1F00H y 2000H. */
             /* C0100             llamar a una subrutina. Comando CALL.    */
             /* D0100,0110        presenta desde la posic. 0100 a 0110H    */
             /* E0000,8010        listado ensamblador.                     */
             /* G4000             ir a la posicion 4000H. Comando GOTO.    */
             /* H6000=12 34 56 <ctrlY> entrada de bloques de memoria.      */
             /* M8000,8010,3      mover bloques de memoria de 3 posic.     */
             /* P8000,8010,20     paso a paso desde 8000H a 8010H o 20 pasos. */
             /* p00               ver contenido Puerto 0.                  */
             /* p00=FF            asignar valores de puertos.              */
             /* R<cr>             ver contenido de registros.              */
             /* R0                cambiar contenido de registros.          */
             /* S4000,2A-3F-40-12,50-34 <CR> Substituir posic. memoria.    */
             /* T1800,1803        asignar TRAZA entre posiciones 1800 a 1803. */
             /* CTRL-A            visualizar bits de FLAG.                 */
             /* CTRL-C            latch de control de Pantalla.            */
             /* CTRL-D            latch de activacion Spooler.             */
             /* CTRL-I            latch activacion periferico entrada.     */
             /* CTRL-O            latch activacion periferico salida.      */
             /* CTRL-P            activar periferico de salida.            */
             /* CTRL-Q            continua la presentacion en pantalla.    */
             /* CTRL-S            para la presentacion en pantalla.        */
             /* CTRL-T            pasar del modo MONITOR al modo TERMINAL. */
             /* ********************************************************** */
  2    1        DECLARE LIT LITERALLY 'LITERALLY';
  3    1        DECLARE DEC LIT 'DECLARE',  OUT LIT 'OUTPUT',   IN  LIT 'INPUT';
  4    1        DEC UD0     LIT '00H',  UD1     LIT '08H', /* Usart Data */
                    US0     LIT '01H',  US1     LIT '09H', /* Usart Status */
                    UC0     LIT '01H',  UC1     LIT '09H', /* Usart Command */
                    CMD55B  LIT '18H',  CMD55A  LIT '20H',
                    STS55B  LIT '18H',  STS55A  LIT '20H',
                    PA55B   LIT '19H',  PA55A   LIT '21H', /* LP port in */
                    PB55B   LIT '1AH',  PB55A   LIT '22H', /* LP port out */
                    PC55B   LIT '1BH',  PC55A   LIT '23H',
                    TML55B  LIT '1CH',  TML55A  LIT '24H', /* Timer Low */
                    TMH55B  LIT '1DH',  TMH55A  LIT '25H', /* Timer High */
                    TE      LIT '01H',  TRDY    LIT '01H',
                    DTR     LIT '02H',  RRDY    LIT '02H',
                    RE      LIT '04H',  TEMPTY  LIT '04H';/*Receiver Enable, Transm.empty*/
                                        PE      LIT '08H';/* Parity Error */
                    ER      LIT '10H',  OE      LIT '10H';/* Error Reset, Overrun Error */
                    RTS     LIT '20H',  FE      LIT '20H';/*Request to Send,Framing Error*/
                    IR      LIT '40H',  DSR     LIT '80H';/*Internal Reset,Data Set Ready*/
                    CTRLA   LIT '01H',  CTRLC   LIT '03H',  CTRLD   LIT '04H',
                    BEL     LIT '07H',  CTRLI   LIT '09H',  LF      LIT '0AH',
                    CR      LIT '0DH',  CTRLO   LIT '0FH',  CTRLP   LIT '10H',
                    CTRLQ   LIT '11H',  CTRLS   LIT '13H',  CTRLT   LIT '14H',
                    CTRLY   LIT '19H',  ESC     LIT '1BH',
                    FALSE   LIT '00H',  SPLFLG  LIT '08H',
                    TRUE    LIT '0FFH', COFLG   LIT '40H',
                    FALSO   LIT '0F0H', BEFLG   LIT '20H',
                    INFLG   LIT '01H',  BFFLG   LIT '10H',
                    OUTFLG  LIT '02H',  CTLFLG  LIT '04H';/* Control flag */
                    OUTABLE LIT '80H';/* Output enable */

             /* **********  LITERALES NIVEL 1 ******************** */
  5    1        DEC CISTS0  LIT '((IN(US0) AND RRDY)=0)';
  6    1        DEC COSTS0  LIT '((IN(US0) AND (TRDY OR TEMPTY) XOR (TRDY OR TEMPTY))<>0)';
  7    1        DEC MISTS0  LIT '((IN(US1) AND RRDY)=0)';
  8    1        DEC MOSTS0  LIT '((IN(US1) AND (TRDY OR TEMPTY) XOR (TRDY OR TEMPTY))<>0)';
  9    1        DEC LISTS0  LIT '((IN(STS55A) AND 02H)=0)';
 10    1        DEC LOSTS0  LIT '((IN(STS55A) AND 10H)<>0)';
 11    1        DEC CISTS1  LIT '((IN(US0) AND RRDY)<>0)';
 12    1        DEC COSTS1  LIT '((IN(US0) AND (TRDY OR TEMPTY) XOR (TRDY OR TEMPTY))=0)';
 13    1        DEC MISTS1  LIT '((IN(US1) AND RRDY)<>0)';
 14    1        DEC MOSTS1  LIT '((IN(US1) AND (TRDY OR TEMPTY) XOR (TRDY OR TEMPTY))=0)';
 15    1        DEC LISTS1  LIT '((IN(STS55A) AND 02H)<>0)';
 16    1        DEC LOSTS1  LIT '((IN(STS55A) AND 10H)=0)';
```

```
                /* **********  DECLARACION DE VARIABLES *************** */
17   1          DEC (SPLSV,PCLSV,RST13) ADDRESS EXTERNAL;
18   1          DEC (ASAV,FSAV,BSAV,CSAV,DSAV,ESAV,HSAV,LSAV,ISAV) BYTE EXTERNAL;
19   1          DEC START0 LABEL PUBLIC, NEMON(32) BYTE PUBLIC, FLAG BYTE PUBLIC;
20   1          DEC PC ADDRESS AT(.PCLSV), SP ADDRESS AT(.SPLSV);
21   1          DEC (BUFHV,BUFLV) ADDRESS PUBLIC;
22   1          DEC (BUFHF,BUFLF,DIREC,LONGI,POSMEM,RAMBUSY,TRACEH,TRACEL) ADDRESS;
23   1          DEC ADR(3) ADDRESS;
24   1          DEC (INKEY,DATHEX0,DATHEX1,N,PRMFLG) BYTE;
25   1          DEC MEMA BASED  POSMEM  ADDRESS;
26   1          DEC MEM8 BASED POSMEM BYTE, DIR8 BASED DIREC BYTE, CHR8FH BASED BUFHV BYTE;
27   1          DEC RGSTRA(*) ADDRESS DATA ('SPPC');
28   1          DEC RGADDA(*) ADDRESS DATA (.SPLSV,.PCLSV);
29   1          DEC RGSTR8(*) BYTE DATA ('AFBCDEHLI');
30   1          DEC RGADD8(*) ADDRESS DATA(.ASAV,.FSAV,.BSAV,.CSAV,.DSAV,.ESAV,.HSAV,.LSAV,
                .ISAV);
31   1          DEC MSKSTR(8) STRUCTURE (MASK BYTE, STRFLG(6) BYTE) DATA (INFLG,'INFLG ',
               OUTFLG,'OUTFLG',CTLFLG,'CTLFLG',SPLFLG,'SPLFLG',BFFLG,'BFFLG ',BEFLG,'BEFLG ',
               COFLG,'COFLG ',OUTABLE,'OUTABL');
                /* ***************** PROCEDIMIENTOS NIVEL 0     ********* */
                /* Programa usuario en ensamblador */
32   1     PRGUSR: PROCEDURE EXTERNAL;
33   2                 END PRGUSR;
34   1     SSTEP:  PROCEDURE EXTERNAL;
35   2             END SSTEP;
36   1     PORTIN: PROCEDURE (INPORT) BYTE EXTERNAL;
37   2             DEC INPORT BYTE;
38   2             END PORTIN;
39   1     PRTOUT: PROCEDURE (OUTPORT,DATO) EXTERNAL;
40   2                 DEC (OUTPORT,DATO) BYTE;
41   2                 END PRTOUT;
                /* Conversion ASC to HEX */
42   1     ASCHEX: PROCEDURE (CHAR) BYTE;
43   2             DEC CHAR BYTE;
44   2             IF CHAR<='9' AND CHAR>='0' THEN RETURN CHAR-'0';
46   2             IF CHAR<='F' AND CHAR >='A' THEN RETURN CHAR-'A'+0AH;
48   2             IF CHAR <='f' AND CHAR >='a' THEN RETURN CHAR-'a'+0AH;
50   2             RETURN FALSO;
51   2             END ASCHEX;
52   1     HEXASC: PROCEDURE (DATHEX) BYTE EXTERNAL;/* Conversion HEX a ASC */
53   2             DEC DATHEX BYTE;
54   2             END HEXASC;
                /* Programacion del timer del 8155 */
55   1     TMR55B: PROCEDURE (N);/* 9600,4800,2400,1200,600,300,150,110 */
56   2                 DEC BAUD(8) ADDRESS DATA (4005H,400AH,4014H,4028H,4050H,40A0H,
                    4140H,41B4H);
57   2                 DEC N BYTE;
58   2                 OUT(TMH55B)=HIGH(BAUD(N));
59   2                 OUT(TML55B)=LOW(BAUD(N));
60   2                 END TMR55B;
61   1     DECODE: PROCEDURE (DIREC) ADDRESS EXTERNAL;/*Listado Assembler de instrucc.*/
62   2             DEC DIREC ADDRESS;
63   2             END DECODE;
64   1     LO:     PROCEDURE (CHAR) ;/* List Output, salida por impresora */
65   2             DEC CHAR BYTE;
66   2             DO WHILE LOSTS0;
                    /* Port A buffer full */
67   3             END;
68   2             OUT (PB55A)=CHAR;
69   2             END LO;
70   1     MOLOSTS:    PROCEDURE BYTE;/* Status periferico de salida */
71   2                 IF (FLAG AND OUTFLG)=0 THEN RETURN LOSTS1;
73   2                 RETURN MOSTS1;
74   2                 END MOLOSTS;
75   1     MO:     PROCEDURE (CHAR);/* Modem Output */
76   2             DEC CHAR BYTE;
77   2             DO WHILE MOSTS0;
78   3             END;
79   2             OUT (UD1)=CHAR;
80   2             END MO;
                /* Param check: separadores validos: (',',' ',LF,CR,'=') */
81   1     PRMCHK: PROCEDURE BYTE;
82   2             IF INKEY=CR OR INKEY=',' OR INKEY=' ' OR INKEY=LF
                    OR INKEY ='=' THEN RETURN TRUE;
84   2             RETURN FALSE;
85   2             END PRMCHK;
```

```
                        /* ***************** PROCEDIMIENTOS NIVEL 1 ************************/
 86   1       MOLO:        PROCEDURE (CHAR);/* Salida por periferico de salida */
 87   2                    DECLARE CHAR BYTE;
 88   2                    IF (FLAG AND OUTFLG)=0 THEN CALL LO(CHAR);
 90   2                    ELSE CALL MO(CHAR);
 91   2                    END MOLO;
                        /* ***************** PROCEDIMIENTOS NIVEL 2 ************************/
 92   1       SPLOUT: PROCEDURE;/* Vaciado del Spooler */
 93   2               DEC CHRBUF0 BASED BUFLV BYTE;
 94   2               IF (FLAG AND BEFLG)=0 THEN /* Buffer no vacio */
 95   2                 IF MOLOSTS THEN DO;/* Periferico activado */
 97   3                   CALL MOLO(CHRBUF0);
 98   3                   IF (BUFLV:=BUFLV+1)=BUFHF THEN BUFLV=BUFLF;
                           /* Buffer vacio del todo */
100   3                   IF BUFLV=BUFHV THEN FLAG=FLAG AND NOT BFFLG OR BEFLG;
                           /* Buffer no lleno del todo */
102   3                   ELSE FLAG=FLAG AND NOT (BFFLG OR BEFLG);
103   3                 END;
104   2               END SPLOUT;
                      /************************** PROCEDIMIENTOS NIVEL 3 ********************** */
105   1       CI:     PROCEDURE BYTE;/* Console Input */
106   2               IF (IN(US0) AND RRDY)=0 THEN DO;
108   3                 OUT(UC0)=RTS OR RE OR DTR OR TE;
109   3                 DO WHILE CISTS0;
110   4                   CALL SPLOUT;
111   4                 END;
112   3               END;
113   2               OUT(UC0)=ER OR RE OR DTR OR TE;
114   2               RETURN IN(UD0);
115   2               END CI;
116   1       MI:     PROCEDURE BYTE;/* Modem Input */
117   2               IF (IN(US1) AND RRDY)=0 THEN DO;
119   3                 OUT(UC1)=RTS OR RE OR DTR OR TE;
120   3                 DO WHILE MISTS0;
121   4                   CALL SPLOUT;
122   4                 END;
123   3               END;
124   2               OUT(UC1)=ER OR RE OR DTR OR TE;
125   2               RETURN (IN(UD1));
126   2               END MI;
127   1       CO:     PROCEDURE (CHAR) ;/* Console Output */
128   2               DEC CHAR BYTE;
129   2               DO WHILE COSTS0;
130   3                 CALL SPLOUT;
131   3               END;
132   2               OUT(UD0)=CHAR;
133   2               END CO;
134   1       LI:     PROCEDURE BYTE;/* List Input, entrada impresora */
135   2               DO WHILE LISTS0;
136   3                 CALL SPLOUT; /* Port B buffer empty */
137   3               END;
138   2               RETURN IN(PA55A);
139   2               END LI;
                      /************************** PROCEDIMIENTOS NIVEL 4 ************************/
140   1       PO:     PROCEDURE(CHAR);/* Peripheral Output */
141   2               DEC CHAR BYTE;
142   2               IF (FLAG AND COFLG)<>0 THEN CALL CO(CHAR);
144   2               IF (FLAG AND OUTABLE)<>0 THEN DO;/* Periferico activado */
146   3                 IF (FLAG AND SPLFLG)<>0 THEN DO;/* Spooler activado */
148   4                   DO WHILE (FLAG AND BFFLG)<>0;
149   5                     CALL SPLOUT;/* Vaciar el Spooler se este esta lleno */
150   5                   END;
151   4                   CHRBFH=CHAR;/* Llenado del Spooler */
152   4                   IF (BUFHV:=BUFHV+1)=BUFHF THEN BUFHV=BUFLF;
                           /* Spooler lleno del todo */
154   4                   IF BUFHV=BUFLV THEN FLAG=FLAG AND NOT BEFLG OR BFFLG;
                           /* Spooler ni vacio ni lleno del todo */
156   4                   ELSE FLAG=FLAG AND NOT (BEFLG OR BFFLG);
157   4                 END;
158   3                 ELSE DO;/* Spooler no activado */
159   4                   DO WHILE (FLAG AND BEFLG)=0;
160   5                     CALL SPLOUT;
161   5                   END;
162   4                   CALL MOLO(CHAR);
163   4                 END;
164   3               END;
165   2               END PO;
```

```
166  1     ESPERA: PROCEDURE BYTE;/* Si es CTRL-S esperar a CTRL-@ (TRUE) o ESC (FALSE) */
167  2             IF CISTS1 THEN
168  2               IF (INKEY:=CI)= CTRLS THEN
169  2                 DO WHILE (INKEY:=CI)<> CTRL@;
170  3                   IF INKEY = ESC THEN RETURN FALSE;
172  3                 END;
173  2             RETURN TRUE;
174  2             END ESPERA;
                /* *************** PROCEDIMIENTOS NIVEL 5 ****************************** */
175  1     TI:     PROCEDURE BYTE;/* Terminal Input */
176  2             DEC CHAR BYTE;
177  2             CALL PO(CHAR:=CI);
178  2             RETURN CHAR;
179  2             END TI;
180  1     POBYTE: PROCEDURE (DATHEX@);/* Presentar un byte HEX a ASC por pantala */
181  2             DEC DATHEX@ BYTE;
182  2             CALL PO(HEXASC(ROR(DATHEX@,4)));
183  2             CALL PO(HEXASC(DATHEX@));
184  2             END POBYTE;
185  1     POCRLF: PROCEDURE;
186  2             CALL PO(CR);
187  2             CALL PO(LF);
188  2             END POCRLF;
189  1     POSTR:  PROCEDURE (DIREC);/* Presentacion de una cadena de caracteres */
190  2             DEC DIREC ADDRESS;
191  2             DEC LETRA BASED DIREC BYTE;
192  2             DO WHILE LETRA <> '&';
193  3               CALL PO(LETRA);
194  3               DIREC=DIREC+1;
195  3             END;
196  2             END POSTR;
                /****************** PROCEDIMIENTOS NIVEL 6 ********************/
197  1     POADR:  PROCEDURE (ADRHEX);/* Presentacion en pantalla de una direccion */
198  2             DEC ADRHEX ADDRESS;
199  2             CALL POBYTE (HIGH(ADRHEX));
200  2             CALL POBYTE (LOW(ADRHEX));
201  2             END POADR;
202  1     POASM:  PROCEDURE;/* Presentacion en pantalla de la cadena assembler */
203  2             DO POSMEM=.NEMON TO .NEMON+1FH;
204  3               CALL PO(MEMB);
205  3             END;
206  2             END POASM;
207  1     POREGB: PROCEDURE;/* Salida de registros byte por periferico */
208  2             POSMEM=RGADDB(N);
209  2              CALL PO(RGSTRB(N));
210  2              CALL PO('=');
211  2              CALL POBYTE(MEMB);
212  2             END POREGB;
213  1     STATUS: PROCEDURE (N);/* Procedimiento de visualizacion del bit N de FLAG */
214  2             DEC (N,M) BYTE;
215  2             DO M=@ TO 5;
216  3               CALL PO(MSKSTR(N).STRFLG(M));
217  3             END;
218  2             CALL POSTR(.('=0&'));
219  2             IF (FLAG AND MSKSTR(N).MASK)=@ THEN CALL POSTR(.('FF&'));
221  2             ELSE CALL POSTR(.('N &'));
222  2             END STATUS;
                /* ****************** PROCEDIMIENTOS NIVEL 7 ********************** */
223  1     POREGA: PROCEDURE;/* Presentacion de un registro por pantalla */
224  2             POSMEM=RGADDA(N);
225  2              CALL PO(HIGH(RGSTRA(N)));
226  2              CALL PO(LOW(RGSTRA(N)));
227  2              CALL PO('=');
228  2              CALL POADR(MEMA);
229  2             END POREGA;
                /* Recogida de 1 direccion por teclado u presentar separador */
230  1     PARAM@: PROCEDURE BYTE;
231  2             PRMFLG=FALSE;
                    /* TRUE separador valido */
                    /* FALSE resto de los casos */
232  2             IF (DATHEX@:=ASCHEX(INKEY:=TI))<>FALSO THEN DO;
234  3             PRMFLG=TRUE;
235  3             DIREC=DATHEX@;
236  3             DO WHILE (DATHEX@:=ASCHEX(INKEY:=TI)) <> FALSO;
237  4               DIREC=DIREC*10H+DATHEX@;
238  4             END;
239  3             END;
240  2             RETURN PRMCHK;
241  2             END PARAM@;
```

```
242   1     STSLTH: PROCEDURE(N);/* Status Latch */
243   2             DEC N BYTE;
244   2             FLAG=FLAG XOR MSKSTR(N).MASK;
245   2             CALL STATUS(N);
246   2             END STSLTH;
                    /* ********************* PROCEDIMIENTOS NIVEL 8 ************** */
247   1     EXPR2:  PROCEDURE BYTE;/* Recogida de 2 direcciones por teclado */
248   2             IF PARAM0 THEN DO;
250   3               ADR(0)=DIREC;
251   3               ADR(1)=(DIREC:=DIREC+1);
252   3               IF INKEY=CR THEN RETURN TRUE;
254   3               IF PARAM0 THEN DO;
256   4                 ADR(1)=DIREC;
257   4                 IF INKEY=CR THEN RETURN TRUE;
259   4               END;
260   3             END;
261   2             RETURN FALSE;/* Separador no valido */
262   2             END EXPR2;
263   1     PARAM1: PROCEDURE BYTE;/* Coger direccion y ver que la hay */
                    /* TRUE=PRMCHK valido y direcc. */
264   2             RETURN PARAM0 AND PRMFLG;
265   2             END PARAM1;
266   1     CONTROL:    PROCEDURE BYTE;/* Activar-desactivar bit de FLAG */
267   2             IF INKEY=CTRLP THEN CALL STSLTH(7);/* OUTABLE latch */
269   2             ELSE IF INKEY=CTRLC THEN CALL STSLTH(2);/* CTLFLG latch */
271   2             ELSE IF INKEY=CTRLD THEN CALL STSLTH(3);/* SPLFLG latch */
273   2             ELSE IF INKEY=CTRLI THEN CALL STSLTH(0);/* INFLAG latch */
275   2             ELSE IF INKEY=CTRLO THEN CALL STSLTH(1);/* OUTFLAG latch */
277   2             ELSE RETURN FALSE;
278   2             RETURN TRUE;
279   2             END CONTROL;
                    /* ********************* PROCEDIMIENTOS NIVEL 9 ********************* */
                    /* Recoge 3 direcciones */
280   1     EXPR3:  PROCEDURE BYTE;
281   2             IF (NOT EXPR2)AND(NOT PRMCHK) THEN RETURN FALSE;/*Separador no valido*/
283   2             ADR(2)=1;
284   2             IF INKEY<>CR THEN DO;
286   3               IF PARAM1 THEN ADR(2)=DIREC;
288   3               IF INKEY<>CR THEN RETURN FALSE;
290   3             END;
291   2             RETURN TRUE;
292   2             END EXPR3;
              /* ************************ COMIENZO REAL DEL PROGRAMA ********************** */
293   1     START0: OUT(CMD55A)=7AH;/* Pa=Input Port, Pb=Output Port, Pc=Control Port */
294   1             OUT(CMD55B)=00H;/* Pa=Input Port, Pb=Input Port, Pc=Input Port */
295   1             CALL TMR55B(ROR(IN(PC55B) AND 38H,3));
296   1             OUT(CMD55B)=0C0H;/* Comienzo Timer B */
297   1             OUT(UC0),OUT(UC1),OUT(UC0),OUT(UC1),OUT(UC0),OUT(UC1)=0;/* Cmd. Mode */
298   1             OUT(UC0),OUT(UC1)=40H; /* Reset Usarts */
299   1             OUT(UC0),OUT(UC1)=IN(PB55B) OR 4BH;/* Inicializar Usarts, *64 */
300   1             OUT(UC0),OUT(UC1)=RTS OR ER OR RE OR DTR OR TE;
301   1             BUFHF=2000H;
302   1             BUFLF,BUFLV,BUFHV=1F00H;
303   1             IF ((FLAG:=IN(PB55B) AND (COFLG OR INFLG OR OUTFLG OR SPLFLG) OR BEFLG)
                    AND COFLG)<>0 THEN DO;
305   2             TRACEL=0;
306   2             TRACEH=0FFFFH;
307   2             CALL POSTR(.(CR,LF,BEL,'SDC-85 PREPARADO&'));
308   2     START1: CALL POCRLF;
309   2             CALL PO('*');
310   2             IF (INKEY:=TI)='D' THEN /* Comando DISPLAY de posiciones de memoria */
311   2               IF EXPR2 THEN DO;
313   3                 DIREC=ADR(0);
314   3     D0:         CALL POCRLF;
315   3                 CALL POADR(DIREC);
316   3                 CALL PO(':');
317   3                 DO WHILE ESPERA;
318   4                   CALL PO(' ');
319   4                   CALL POBYTE(DIRB);
320   4                   IF (DIREC:=DIREC+1)>ADR(1) THEN GOTO START1;
322   4                   IF (DIREC AND 0FH)=0 THEN GOTO D0;
324   4                 END;
325   3               END;
326   2               ELSE GOTO POPRMERR;
```

```
327  2            ELSE IF INKEY='S' THEN DO;/* Comando SUBSTITUCION posiciones memoria */
329  3              IF PARAM1 THEN DO;
331  4                POSMEM=DIREC;
332  4                DO WHILE INKEY <> CR;
333  5                  CALL POBYTE (MEMB);
334  5                  CALL PO('-');
335  5                  IF PARAM1 THEN MEMB=LOW(DIREC);
337  5                  POSMEM=POSMEM+1;
338  5                END;
339  4              END;
340  3              ELSE GOTO POPRMERR;
341  3            END;
342  2            ELSE IF INKEY='G' THEN DO;/* Comando GO a una subrutina */
344  3                   IF PARAM1 THEN PC=DIREC;
346  3                   IF INKEY=CR THEN CALL PRGUSR;
348  3                   ELSE GOTO POPRMERR;
349  3                 END;
350  2            ELSE IF INKEY='C' THEN DO;
352  3                   IF PARAM1 THEN PC=DIREC;
354  3                   IF INKEY=CR THEN DO;
356  4                     POSMEM=(SPLSV:=SPLSV-2);
357  4                     MEMA=PCLSV;
358  4                     POSMEM=(SPLSV:=SPLSV-2);
359  4                     MEMA=.RST13;
360  4                     CALL PRGUSR;
361  4                   END;
362  3                   ELSE GOTO POPRMERR;
363  3                 END;
364  2            ELSE IF INKEY='H' THEN DO;/* Comando entrada HEXADECIMAL en memoria */
366  3              IF PARAM1 AND INKEY='=' THEN DO;
368  4                POSMEM=0;/* Reconida de 1 byte en hexadecimal */
369  4     H1:        DO WHILE (DATHEX1:=ASCHEX(INKEY:=TI)) <> FALSO ;
370  5                  IF (DATHEX0:=ASCHEX(INKEY:=TI)) <> FALSO THEN
371  5                    DATHEX1=ROR(DATHEX1,4) OR DATHEX0;
372  5                  ELSE IF NOT PRMCHK THEN GOTO H2;
                        POSMEM=POSMEM+(DIRB:=DATHEX1);
375  5                  DIREC=DIREC+1;
376  5                END;
377  4                IF PRMCHK THEN GOTO H1;
379  4     H2:        CALL POCRLF;
380  4                CALL POSTR(.('CHECK=&'));
381  4                CALL POADR (POSMEM);
382  4              END;
383  3            END;
384  2            ELSE IF INKEY='M' THEN /* Comando MOVER bloques de memoria */
385  2                 IF EXPR3 THEN CALL MOVE(ADR(2),ADR(0),ADR(1));
387  2                 ELSE GOTO POPRMERR;
388  2            ELSE IF INKEY='R' THEN /* Cambiar valores de registros */
389  2                  IF (INKEY:=TI)='Q' THEN DO ;
391  3                    CALL POCRLF;
392  3                    DO N=0 TO 1;
393  4                      CALL POREGA;
394  4                      CALL PO('-');
395  4                      IF PARAM1 THEN MEMA=DIREC;
397  4                      IF INKEY=CR THEN GOTO START1;
399  4                      IF NOT PRMCHK THEN GOTO POPRMERR;
401  4                    END;
402  3                    DO N=0 TO 8;
403  4                      CALL POREGB;
404  4                      CALL PO('-');
405  4                      IF PARAM1 THEN MEMB=LOW(DIREC);
407  4                      IF INKEY=CR THEN GOTO START1;
409  4                      IF NOT PRMCHK THEN GOTO POPRMERR;
411  4                    END;
412  3                  END;
413  2                  ELSE IF INKEY=CR THEN DO;/* Presentacion de registros */
415  3                         CALL POCRLF;
416  3                         DO N=0 TO 1;
417  4                           CALL POREGA;
418  4                           CALL PO(' ');
419  4                         END;
420  3                         DO N=0 TO 8;
421  4                           CALL POREGB;
422  4                           CALL PO(' ');
423  4                         END;
424  3                         CALL POCRLF;
425  3                       END;
426  2                       ELSE GOTO POCMDERR;
```

```
427  2          ELSE IF INKEY='E' THEN /* Listado en ensamblador */
428  2                 IF EXPR2 THEN DO;
430  3                   DIREC=ADR(0);
431  3                   CALL POCRLF;
432  3                   DO WHILE (DIREC:=DECODE(DIREC))<>ADR(1) AND ESPERA;
433  4                     CALL POASM;
434  4                     CALL POCRLF;
435  4                   END;
436  3                 END;
437  2                 ELSE GOTO POPRMERR;
                     /* Comando Paso a Paso */
438  2          ELSE IF INKEY='P' THEN DO;
440  3                 DIREC=PC;
441  3                 IF EXPR3 THEN DO;
443  4                   CALL POCRLF;
444  4                   IF (FLAG AND CTLFLG)<>0 THEN DO;
446  5                     DO N=0 TO 35;
447  6                       CALL PO(' ');
448  6                     END;
449  5                     CALL POSTR(.('SP   PC &'));
450  5                     DO N=0 TO 8;
451  6                       CALL POSTR(.('  &'));
452  6                       CALL PO(RGSTRB(N));
453  6                     END;
454  5                   END;
455  4                   PC=ADR(0);
456  4                   LONGI=ADR(2)+1;
457  4                   DO WHILE ((LONGI:=LONGI-1)<>0) AND (PC<>ADR(1)) AND ESPERA;
458  5                     IF ((POSMEM:=DECODE(PC))>=TRACEL)AND(POSMEM<=TRACEH)THEN DO;
460  6                       CALL POCRLF;
461  6                       CALL POASM;
462  6                       CALL SSTEP;
463  6                       IF (FLAG AND CTLFLG)<>0 THEN DO;
465  7                         CALL POSTR(.('    &'));
466  7                         DO N=0 TO 1;
467  8                           POSMEM=RGADDA(N);
468  8                           CALL POADR(MEMA);
469  8                           CALL PO(' ');
470  8                         END;
471  7                         DO N=0 TO 8;
472  8                           POSMEM=RGADDB(N);
473  8                           CALL POBYTE(MEMB);
474  8                           CALL PO(' ');
475  8                         END;
476  7                       END;
477  6                     END;
478  5                     ELSE CALL SSTEP;
479  5                   END;
480  4                 END;
481  3                 ELSE GOTO POPRMERR;
482  3               END;
483  2          ELSE IF INKEY=CTRLA THEN DO;
485  3               DO N=0 TO 7;
486  4                 CALL STATUS(N);
487  4                 IF N=6 THEN CALL POCRLF;
489  4                 CALL PO(' ');
490  4               END;
491  3               CALL POSTR(.('BUFFER=&'));/* Presenta posic. buffer fi.io */
492  3               CALL POADR(BUFLF);
493  3               CALL PO(',');
494  3               CALL POADR(BUFHF);
495  3               CALL POSTR(.(' TRACE=&'));
496  3               CALL POADR(TRACEL);
497  3               CALL PO(',');
498  3               CALL POADR(TRACEH);
499  3               IF BUFHV<BUFLV THEN RAMBUSY=(BUFHF-BUFLV)+(BUFHV-BUFLF);
501  3               ELSE RAMBUSY=BUFHV-BUFLV;
502  3               CALL POSTR(.(' RAMBUSY=&'));/* Presenta RAM ocupada por buffer */
503  3               CALL POADR(RAMBUSY);
504  3             END;
```

```
505   2             ELSE IF INKEY='p' THEN DO;/* Comando de salida/entrada puerto */
507   3               IF PARAM1 THEN IF INKEY=CR THEN DO;/* Entrada puerto */
510   4                 CALL POCRLF;
511   4                 CALL POSTR(.('PORT(&'));
512   4                 CALL POBYTE(LOW(DIREC));
513   4                 CALL POSTR(.(')=&'));
514   4                 CALL POBYTE(PORTIN(LOW(DIREC)));
515   4               END;
516   3               ELSE IF INKEY='=' THEN DO;/* Salida puerto */
518   4                     POSMEM=DIREC;
519   4                     IF PARAM1 THEN CALL PRTOUT(LOW(POSMEM),LOW(DIREC));
521   4                     ELSE GOTO POPRMERR;
522   4                   END;
523   3                   ELSE GOTO POPRMERR;
524   3               END;
525   2             ELSE IF INKEY='B' THEN /* Comando de asignacion de BUFFER */
526   2               IF EXPR2 THEN DO;
528   3                 BUFLF,BUFLV,BUFHV=ADR(0);
529   3                 BUFHF=ADR(1);
530   3               END;
531   2               ELSE GOTO POPRMERR;
532   2             ELSE IF INKEY='T' THEN /* Comando TRACE */
533   2               IF EXPR2 THEN DO;
535   3                 TRACEL=ADR(0);
536   3                 TRACEH=ADR(1);
537   3               END;
538   2               ELSE GOTO POPRMERR;
539   2             ELSE IF INKEY=CTRLT THEN DO;/* Modo TERMINAL */
541   3     TERMINAL:   IF MOSTS1 AND CISTS1 THEN
542   3                   IF (INKEY:=CI)=CTRLT THEN GOTO START1;
544   3                   ELSE IF NOT CONTROL THEN CALL MO(INKEY);
                        IF MISTS1 AND COSTS1 THEN CALL PO(MI);
548   3                 GOTO TERMINAL;
549   3               END;
550   2             ELSE IF NOT CONTROL THEN GOTO POCMDERR;
                    GOTO START1;
553   2     POCMDERR:   CALL POSTR(.(BEL,'#',CR,LF,'CMD-ERR','&'));
554   2                 GOTO START1;
555   2     POPRMERR:   CALL POSTR(.(BEL,'#',CR,LF,'PRM-ERR','&'));
556   2                 GOTO START1;
557   2             END;
558   1             ELSE IF ((FLAG:=FLAG OR OUTABLE) AND INFLG)=0 THEN DO;/* Spooler */
560   2     SP0:        IF (FLAG AND COFLG)<>0 AND CISTS1 THEN
561   2                   IF CI=ESC THEN GOTO START0;
563   2                 CALL PO(LI);
564   2                 GOTO SP0;
565   2               END;
                SP1:    IF (FLAG AND COFLG)<>0 AND CISTS1 THEN
567   1                   IF CI=ESC THEN GOTO START0;
569   1                 CALL PO(MI);
570   1                 GOTO SP1;
571   1             END MONITOR;
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 0DC0H   3520D
     VARIABLE AREA SIZE = 0050H     80D
     MAXIMUM STACK SIZE = 0012H     18D
     578 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
 .SID-II OBJECT LOCATER V3.0 INVOKED BY:
-LOCATE MONITE.LNK TO MONITE.LOC SYMBOLS CODE(113H) STACK(205BH) PUBLICS COLUMNS(3)&
**MAP PRINT(MONITE.MLC)

SYMBOL TABLE OF MODULE MONITE
READ FROM FILE :F0:MONITE.LNK
WRITTEN TO FILE :F0:MONITE.LOC

VALUE TYPE SYMBOL          VALUE TYPE SYMBOL          VALUE TYPE SYMBOL

00FEH PUB  PORTIN          00C5H PUB  PRGUSR         0103H PUB  PRTOUT
0003H PUB  RST13           009AH PUB  SSTEP          01F1H PUB  START0
1036H PUB  HEXASC          10FCH PUB  DECODE          143FH PUB  @P0014
1440H PUB  @P0015          1443H PUB  @P0016          1444H PUB  @P0017
144CH PUB  @P0025          144DH PUB  @P0026          1450H PUB  @P0027
1451H PUB  @P0028          1459H PUB  @P0034          145BH PUB  @P0035
146DH PUB  @P0071          146EH PUB  @P0072          1474H PUB  @P0094
1477H PUB  @P0095          147EH PUB  @P0098          1480H PUB  @P0099
1483H PUB  @P0100          148BH PUB  @P0101          148EH PUB  @P0102
1496H PUB  @P0103          1499H PUB  @P0104          207FH PUB  NEMON
209FH PUB  FLAG            20A0H PUB  BUFHV           20A2H PUB  BUFLV
20D4H PUB  ASAV            20D2H PUB  BSAV            20D1H PUB  CSAV
20D0H PUB  DSAV            20CFH PUB  ESAV            20D3H PUB  FSAV
20D4H PUB  HSAV            20DBH PUB  ISAV            20D9H PUB  LSAV
20D8H PUB  PCHSV           20D7H PUB  PCLSV           20D6H PUB  SPHSV
20D5H PUB  SPLSV
     MOD  MONITOR
2100H SYM  MEMORY          01F1H SYM  START0          207FH SYM  NEMON
209FH SYM  FLAG            20A0H SYM  BUFHV           20A2H SYM  BUFLV
20A4H SYM  BUFHF           20A6H SYM  BUFLF           20ABH SYM  DIREC
20A4H SYM  LONGI           20ACH SYM  POSMEM          20AEH SYM  RAMBUSY
20B0H SYM  TRACEH          2082H SYM  TRACEL          20B4H SYM  ADR
20BAH SYM  INKEY           20BBH SYM  DATHEX0         20BCH SYM  DATHEX1
20BDH SYM  N               20BEH SYM  PRMFLG          0113H SYM  RGSTRA
0117H SYM  RGADDA          0118H SYM  RGSTRB          0124H SYM  RGADDB
0136H SYM  MSXSTR          097AH SYM  ASCHEX          20BFH SYM  CHAR
09D6H SYM  TMR55B          20C0H SYM  N               016EH SYM  BAUD
09EEH SYM  LO              20C1H SYM  CHAR            0A04H SYM  MOLOSTS
0A24H SYM  MO              20C2H SYM  CHAR            0A3CH SYM  PRMCHK
0A7EH SYM  MOLO            20C3H SYM  CHAR            0A9EH SYM  SPLOUT
0AF0H SYM  CI              0B13H SYM  MI              0B36H SYM  CO
20C4H SYM  CHAR            0B51H SYM  LI              0B63H SYM  PO
20C5H SYM  CHAR            0BF7H SYM  ESPERA         0C27H SYM  TI
20C6H SYM  CHAR            0C35H SYM  POBYTE          20C7H SYM  DATHEX0
0C54H SYM  POCRLF          0C5FH SYM  POSTR           20C8H SYM  DIREC
0C80H SYM  POADR           20CAH SYM  ADRHEX          0C97H SYM  POASM
0CBEH SYM  POREGB          0CE9H SYM  STATUS          20CCH SYM  N
20CDH SYM  M               0D4FH SYM  POREGA          0D8EH SYM  PARAM0
0DDDH SYM  STSLTH          20CEH SYM  N               0DFFH SYM  EXPR2
0E39H SYM  P4RAM1          0E41H SYM  CONTROL         0E97H SYM  EXPR3
025FH SYM  START1          027FH SYM  D0              03B8H SYM  HI
041CH SYM  H2              0884H SYM  TERMINAL        08F4H SYM  POCMDERR
08FDH SYM  POPRMERR        0918H SYM  SP0             0948H SYM  SP1
     MOD  SAVREG
00F7H SYM  DEL0            00F4H SYM  DELAY           00FEH SYM  PORTIN
00C5H SYM  PRGUSR          0105H SYM  PRT0            0103H SYM  PRTOUT
00D8H SYM  PRUS0           00DCH SYM  PRUS1           0073H SYM  RST00
0008H SYM  RST10           0010H SYM  RST11           0081H SYM  RST12
0003H SYM  RST13           00AFH SYM  SS0             00B6H SYM  SS1
00B9H SYM  SS2             009AH SYM  SSTEP           0020H SYM  STS55A
0025H SYM  TMRHA           0024H SYM  TMRLA           0024H SYM  TRAP
20D4H SYM  ASAV            20D2H SYM  BSAV            20E2H SYM  CLDBHV
20E0H SYM  CLDBLV          20DFH SYM  CLDFLG          20EAH SYM  CODE
20D1H SYM  CSAV            20D0H SYM  DSAV            20CFH SYM  ESAV
20D3H SYM  FSAV            20DAH SYM  HSAV            20DBH SYM  ISAV
20D9H SYM  LSAV            20D8H SYM  PCHSV           20D7H SYM  PCLSV
20E4H SYM  RST55           20E7H SYM  RST65           20D6H SYM  SPHSV
20D5H SYM  SPLSV           20DDH SYM  SPMON           20DCH SYM  TEMP
     MOD  DASM85
2100H SYM  MEMORY          1036H SYM  HEXASC          20EDH SYM  DATHEX
1057H SYM  LIST            20EEH SYM  LSTPOSLNG       20EFH SYM  POSICION
20F1H SYM  N               20F2H SYM  Y               10A0H SYM  BYTEASC
20F3H SYM  DIRBYTE         20F5H SYM  POSLIN          10D8H SYM  ADDASC
20F6H SYM  DIRADD          20F8H SYM  POSLIN          10FCH SYM  DECODE
20F9H SYM  DIREC           0ED3H SYM  DECMAT0         0EE3H SYM  DECMAT1
0EEFH SYM  R               0EF7H SYM  R0              0EFFH SYM  R1
0F0BH SYM  CC              0F1BH SYM  NEM0            0F2DH SYM  NEM1
0F45H SYM  NEM2            0F5DH SYM  NEMMAT0         0FA5H SYM  NEMMAT1
1005H SYM  NEMMAT2         20FBH SYM  N               20FCH SYM  DECOD
20FDH SYM  CODIGOL         20FEH SYM  CODIGOH         20FFH SYM  POSNEM
```

```
MEMORY MAP OF MODULE MONITE
READ FROM FILE :F0:MONITE.LNK
WRITTEN TO FILE :F0:MONITE.LOC
MODULE START ADDRESS 01EEH

START   STOP LENGTH REL NAME

0000H  0005H     6H  A  ABSOLUTE
0008H  0021H    1AH  A  ABSOLUTE
0024H  002EH     BH  A  ABSOLUTE
0034H  0036H     3H  A  ABSOLUTE
003CH  0112H    D7H  A  ABSOLUTE
0113H  14A1H  138FH  B  CODE
205BH  207EH    24H  B  STACK
207FH  20FFH    81H  B  DATA
2100H  F6BFH  D5C0H  B  MEMORY


-ASM80 SAVREG.ASM MOD85 NOPAGING DEBUG


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


 LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    SAVREG
                      2          EXTRN   START0,BUFLV,BUFHV,FLAG
                      3          PUBLIC  SSTEP,PRGUSR,ESAV,DSAV,CSAV,BSAV,FSAV,ASAV
                      4          PUBLIC  SPLSV,SPHSV,PCLSV,PCHSV,LSAV,HSAV,ISAV
                      5          PUBLIC  PRTOUT,PORTIN,RST13
0025                  6 TMRHA    EQU     25H
0024                  7 TMRLA    EQU     24H
0020                  8 STS55A   EQU     20H
                      9          ;=========================================
0000                 10          ORG     0       ; RST 0. Reset del Sistema.
                     11          ;=========================================
0000 220A00    D     12          SHLD    LSAV
0003 C37300          13          JMP     RST00
                     14          ;=========================================
0008                 15          ORG     08      ; RST 1. Salto de RST 1.
                     16          ;=========================================
0008 220A00    D     17 RST13:   SHLD    LSAV
000B F5              18 RST10:   PUSH    PSW     ;Comenzar a guardar registros
000C AF              19          XRA     A       ;Inhabilitar interrupciones usuario
000D 320D00    D     20          STA     TEMP
0010 E1              21 RST11:   POP     H
0011 220400    D     22          SHLD    FSAV    ;Guardar reg. PSW
0014 E1              23          POP     H
0015 220800    D     24          SHLD    PCLSV   ;Guardar reg. PC
0018 210000          25          LXI     H,0
001B 39              26          DAD     SP
001C 220600    D     27          SHLD    SPLSV   ;Guardar reg. SP
001F C39100          28          JMP     RST12
                     29          ;=========================================================================
0024                 30          ORG     24H     ; Interrupcion TRAP = Regreso del programa usuario
                     31          ;=========================================================================
0024 F5              32 TRAP:    PUSH    PSW     ;Salto de Single Step
0025 3E7A            33          MVI     A,7AH
0027 D320            34          OUT     STS55A  ;Stop Timer
0029 C31000          35          JMP     RST11   ;
                     36          ;=========================
002C                 37          ORG     2CH     ; RST 5.5
                     38          ;=========================
002C C31500    D     39          JMP     RST55
                     40          ;=========================
0034                 41          ORG     34H     ; RST 6.6
                     42          ;=========================
0034 C31800    D     43          JMP     RST65
```

```
                        44          ;==========================
003C                    45          ORG     3CH     ; RST 7.5
                        46          ;==========================
003C E5                 47          PUSH    H       ; guardar registros
003D D5                 48          PUSH    D
003E F5                 49          PUSH    PSW
003F 3A0000     E       50          LDA     FLAG    ; Cambiar FLAG por CLD FLG
0042 47                 51          MOV     B,A
0043 3A1000     D       52          LDA     CLDFLG
0046 320000     E       53          STA     FLAG
0049 78                 54          MOV     A,B
004A 321000     D       55          STA     CLDFLG
004D 2A0000     E       56          LHLD    BUFLV   ; Cambiar BUFLV por CLDBLV
0050 EB                 57          XCHG
0051 2A1100     D       58          LHLD    CLDBLV
0054 220000     E       59          SHLD    BUFLV
0057 EB                 60          XCHG
0058 221100     D       61          SHLD    CLDBLV
005B 2A0300     E       62          LHLD    BUFHV   ; Cambiar BUFHV por CLDBHV
005E EB                 63          XCHG
005F 2A1300     D       64          LHLD    CLDBHV
0062 220000     E       65          SHLD    BUFHV
0065 EB                 66          XCHG
0066 221300     D       67          SHLD    CLDBHV
0069 CDF400             68          CALL    DELAY
006C 3E1B               69          MVI     A,1BH   ; Habilitar solo RST 7.5
006E 30                 70          SIM
006F F1                 71          POP     PSW     ;restaurar registros
0070 D1                 72          POP     D
0071 E1                 73          POP     H
0072 C9                 74          RET
0073 21FFFF     S       75 RST00:   LXI     H,STACK-1
0076 3600       E       76          MVI     M,HIGH START0
0078 2B                 77          DCX     H
0079 3600       E       78          MVI     M,LOW START0
007B 220E00     D       79          SHLD    SPMON
007E C30B00             80          JMP     RST10
00B1 210400     D       81 RST12:   LXI     H,BSAV+1
0084 F9                 82          SPHL
0085 C5                 83          PUSH    B       ;Guardar reg. BC
0086 D5                 84          PUSH    D       ;Guardar reg. DE
0087 20                 85          RIM             ;Guardar Byte de interrupciones
0088 E607               86          ANI     07H
008A 210D00     D       87          LXI     H,TEMP
008D B6                 88          ORA     M
008E 320C00     D       89          STA     ISAV
0091 2A0E00     D       90          LHLD    SPMON   ;Cargar Stack Monitor
0094 F9                 91          SPHL
0095 3E1B               92          MVI     A,1BH   ;Habilitar solo RST 7.5
0097 30                 93          SIM
0098 FB                 94          EI
0099 C9                 95          RET
                        96          ;==========================
                        97          ;Salto a programa usuario
                        98          ;==========================
009A 3A0C00     D       99 SSTEP:   LDA     ISAV    ; Guardar temporalmente enable interrupt
009D E608              100          ANI     08H
009F 320D00     D      101          STA     TEMP
00A2 2A0800     D      102          LHLD    PCLSV
00A5 7E                103          MOV     A,M
00A6 FEF3              104          CPI     (DI)    ; Instruccion 'DI'?
00A8 C2AF00            105          JNZ     SS0
00AB AF                106          XRA     A       ; Si, desabilitar interrupciones
00AC C3B600            107          JMP     SS1
00AF FEF3              108 SS0:     CPI     (EI)    ; Instruccion 'EI'?
00B1 C2B900            109          JNZ     SS2
00B4 3E08              110          MVI     A,08H   ; Si, habilitar interrupciones
00B6 320D00     D      111 SS1:     STA     TEMP
00B9 3E40              112 SS2:     MVI     A,40H
00BB D325              113          OUT     TMRHA
00BD 3EDB              114          MVI     A,219   ; Ciclos TRAP
00BF D324              115          OUT     TMRLA
00C1 3EFA              116          MVI     A,0FAH
00C3 D320              117          OUT     STS55A
```

```
                        118          ;Rutina que restaura registros usuario
00C5 3A0C00     D       119 PRGUSR:  LDA     ISAV    ; Activar mascara de interrupciones
00C8 47                 120          MOV     B,A
00C9 E607               121          ANI     07H
00CB F618               122          ORI     18H
00CD 30                 123          SIM
00CE 78                 124          MOV     A,B     ; Activar interr. si estaba activada por usuario
00CF E608               125          ANI     08H
00D1 C2D800             126          JNZ     PRUS0
00D4 F3                 127          DI
00D5 C3DC00             128          JMP     PRUS1
00D8 37                 129 PRUS0:   STC             ; Instrucciones de retardo
00D9 D2DC00             130          JNC     PRUS1
00DC 210000             131 PRUS1:   LXI     H,0
00DF 39                 132          DAD     SP
00E0 220E00     D       133          SHLD    SPMON   ; Guardar Stack Monitor
00E3 210000     D       134          LXI     H,ESAV
00E6 F9                 135          SPHL
00E7 D1                 136          POP     D       ;Cargar DE
00E8 C1                 137          POP     B       ;Cargar BC
00E9 F1                 138          POP     PSW     ;Cargar AF
00EA E1                 139          POP     H       ;
00EB F9                 140          SPHL            ;Cargar SP
00EC 2A0800     D       141          LHLD    PCLSV   ;
00EF E5                 142          PUSH    H       ;Cargar en Stack PC
00F0 2A0A00     D       143          LHLD    LSAV    ;Cargar HL
00F3 C9                 144          RET
                        145          ;Rutina Delay-retardo
00F4 11FFFF             146 DELAY:   LXI     D,0FFFFH
00F7 1B                 147 DEL0:    DCX     D
00F8 7A                 148          MOV     A,D
00F9 B3                 149          ORA     E
00FA C2F700             150          JNZ     DEL0;
00FD C9                 151          RET
00FE 2EDB               152 PORTIN:  MVI     L,0DBH  ; Entrada por puerto variable
0100 C30501             153          JMP     PRT0
0103 2ED3               154 PRTOUT:  MVI     L,0D3H  ; Salida por puerto variable
0105 61                 155 PRT0:    MOV     H,C
0106 221800     D       156          SHLD    CODE
0109 3EC9               157          MVI     A,0C9H
0108 321D00     D       158          STA     CODE+2
010E 7B                 159          MOV     A,E
010F CD1800     D       160          CALL    CODE
0112 C9                 161          RET
                        162          ;====================
                        163          ;Datos del programa
                        164          ;====================
                        165          DSEG
0000                    166 ESAV:    DS      1       ;Registro DE
0001                    167 DSAV:    DS      1       ;
0002                    168 CSAV:    DS      1       ;Registro BC
0003                    169 BSAV:    DS      1       ;
0004                    170 FSAV:    DS      1       ;Registro AF
0005                    171 ASAV:    DS      1       ;
0006                    172 SPLSV:   DS      1       ;Registro SP
0007                    173 SPHSV:   DS      1       ;
0008                    174 PCLSV:   DS      1       ;Registro PC
0009                    175 PCHSV:   DS      1       ;
000A                    176 LSAV:    DS      1       ;Registro HL
000B                    177 HSAV:    DS      1       ;
000C                    178 ISAV:    DS      1       ;Mascara Interrupciones
000D                    179 TEMP:    DS      1       ;Estado temporal de EI o DI, 0 o 1
000E                    180 SPMON:   DS      2       ;Stack Pointer regreso a Monitor
0010                    181 CLDFLG:  DS      1       ;Cold FLAG
0011                    182 CLDBLV:  DS      2       ;Cold Buffer Low variable
0013                    183 CLDBHV:  DS      2       ;Cold Buffer High variable
0015                    184 RST55:   DS      3       ;Salto de RST 5.5
0018                    185 RST65:   DS      3       ;Salto de RST 6.5
001B                    186 CODE:    DS      3       ;Rutina de IN - OUT
                        187          END
```

```
PUBLIC SYMBOLS
ASAV   D 0005    BSAV   D 0003    CSAV   D 0002    DSAV   D 0001    ESAV   D 0000    FSAV   D 0004    HSAV   D 000B
ISAV   D 000C    LSAV   D 000A    PCHSV  D 0009    PCLSV  D 0008    PORTIN A 00FE    PRGUSR A 00C5    PRTOUT A 0103
RST13  A 000B    SPHSV  D 0007    SPLSV  D 0006    SSTEP  A 009A

EXTERNAL SYMBOLS
BUFHV  E 0000    BUFLV  E 0000    FLAG   E 0000    START0 E 0000

USER SYMBOLS
ASAV   D 0005    BSAV   D 0003    BUFHV  E 0000    BUFLV  E 0000    CLDBHV D 0013    CLDBLV D 0011    CLDFLG D 0010
CODE   D 001B    CSAV   D 0002    DEL0   A 00F7    DELAY  A 00F4    DSAV   D 0001    ESAV   D 0000    FLAG   E 0000
FSAV   D 0004    HSAV   D 000B    ISAV   D 000C    LSAV   D 000A    PCHSV  D 0009    PCLSV  D 000B    PORTIN A 00FE
PRGUSR A 00C5    PRT0   A 0105    PRTOUT A 0103    PRUS0  A 00D8    PRUS1  A 00DC    RST00  A 0073    RST10  A 000B
RST11  A 0010    RST12  A 0081    RST13  A 000B    RST55  D 0015    RST65  D 0018    SPHSV  D 0007    SPLSV  D 0006
SPMON  D 000E    SS0    A 00AF    SS1    A 00B6    SS2    A 00B9    SSTEP  A 009A    START0 E 0000    STS55A A 0020
TEMP   D 000D    TMRHA  A 0025    TMRLA  A 0024    TRAP   A 0024

ASSEMBLY COMPLETE,   NO ERRORS
```

```
iSIS-II PL/M-80 V3.1 COMPILATION OF MODULE DASM85
OBJECT MODULE PLACED IN DASM85.OBJ
COMPILER INVOKED BY:  PLM80 DASM85.PLM WORKFILES(:F0:,:F0:) NOPAGING PAGEWIDTH(100) DEBUG NOINTVECTO
                      -R


  1          DASM85: DO;
  2    1         DECLARE LIT LITERALLY 'LITERALLY';
  3    1         DECLARE DEC LIT 'DECLARE';
  4    1         DEC NEMON(32) BYTE EXTERNAL;

                 /* Conversion HEX to ASC */
  5    1       HEXASC: PROCEDURE (DATHEX) BYTE PUBLIC;
  6    2                 DEC DATHEX BYTE;
  7    2                 IF (DATHEX:=DATHEX AND 0FH)>9 THEN RETURN DATHEX+'A'-0AH;
  9    2                 RETURN DATHEX+'0';
 10    2                 END HEXASC;

               /* Posiciona en cadena NEMON dada su posic. y long. y posic. de caract.*/
 11    1       LIST:     PROCEDURE (LSTPOSLNG,POSICION);
 12    2                 DEC POSICION ADDRESS;
 13    2                 DEC (N,Y,LSTPOSLNG) BYTE;
 14    2                 DEC LETRA BASED POSICION (1) BYTE;
 15    2                 Y=LSTPOSLNG AND 1FH;
 16    2                 DO N=0 TO (ROL(LSTPOSLNG,3) AND 07H)-1;
 17    3                   NEMON(Y+N)=LETRA(N);
 18    3                 END;
 19    2                 END LIST;

                         /* ********************** PROCEDIMIENTOS NIVEL 1 ****************** */

                         /* Convierte en ASCII y posiciona un byte en cadena NEMON */
 20    1       BYTEASC:     PROCEDURE (DIRBYTE,POSLIN);
 21    2                 DEC DIRBYTE ADDRESS;
 22    2                 DEC POSLIN BYTE;
 23    2                 DEC DATO BASED DIRBYTE BYTE;
 24    2                 NEMON(POSLIN)=HEXASC(ROR(DATO,4));
 25    2                 NEMON(POSLIN+1)=HEXASC(DATO);
 26    2                 END BYTEASC;

                         /* ********************** PROCEDIMIENTOS NIVEL 2 ****************** */

                         /* Convierte en ASCII y posicion una direc. en NEMON */
 27    1       ADDASC: PROCEDURE (DIRADD,POSLIN);
 28    2                 DEC DIRADD ADDRESS;
 29    2                 DEC POSLIN BYTE;
 30    2                 CALL BYTEASC(DIRADD+1,POSLIN);
 31    2                 CALL BYTEASC(DIRADD,POSLIN+2);
 32    2                 END ADDASC;

                         /* ********************** PROCEDIMIENTOS NIVEL 3 ****************** */

                         /* Listado en asembler de una instruccion */
 33    1       DECODE: PROCEDURE (DIREC) ADDRESS PUBLIC;
                         /* Declaraciones literales */
 34    2             DEC POSNEM0     LIT '14H',      POSNEM1     LIT '34H',
                       POSNEM12    LIT '55H',      POSNEM2     LIT '54H',
                       POSNEM3     LIT '74H',      POSNEM4     LIT '94H',
                       POSNEM6     LIT '0D4H',     POSOPR0     LIT '19H',
                       POSOPR02    LIT '59H',      POSOPR03    LIT '79H',
                       POSCOM1     LIT '1AH',      POSCOM2     LIT '1BH',
                       POSCOM3     LIT '1CH',      POSCOD0     LIT '06H',
                       POSCOD1     LIT '09H',      POSCOD2     LIT '0CH',
                       POSPNT      LIT '04H',      POSADD      LIT '00H';

                         /* Declaraciones de variables */
 35    2             DEC DECMAT0(16) BYTE DATA (20H,80H,01H,82H,28H,29H,4AH,21H,33H,83H,74H,04H,
                         75H,85H,42H,38H);
 36    2             DEC DECMAT1(6) STRUCTURE (X(2) BYTE) DATA (6AH,2BH,22H,63H,2CH,2DH,31H,20H,
                         00H,21H,30H,01H);
 37    2             DEC R(8) BYTE DATA ('BCDEHLMA');
 38    2             DEC R0(4) STRUCTURE (X(2) BYTE) DATA('B ','D ','H ','SP');
 39    2             DEC R1(4) STRUCTURE (X(3) BYTE) DATA ('B  ','D  ','H  ','PSW');
 40    2             DEC CC(8),STRUCTURE (L(2) BYTE) DATA ('NZ','Z ','NC','C ','PO',
                       'PE','P ','M ');
 41    2             DEC NEM0(6) STRUCTURE(X(3)BYTE) DATA('INR','DCR','MVI','R  ','J  ','C  ');
 42    2             DEC NEM1(6) STRUCTURE(X(4)BYTE) DATA('PUSH','POP ','LXI ','DAD ','INX ',
                         'DCX ');
 43    2             DEC NEM2(8) STRUCTURE(X(3)BYTE) DATA('ADD','ADC','SUB','SBB','ANA','XRA',
                         'ORA','CMP');
```

```
44  2       DEC NEMMAT0(24) STRUCTURE(X(3)BYTE) DATA ('NOP',' ? ',' ? ',' ? ','RIM',
                ' ? ','SIM',' ? ','RLC','RRC','RAL','RAR','DAA','CMA','STC','CMC',
                'ADI','ACI','SUI','SBI','ANI','XRI','ORI','CPI');

45  2       DEC NEMMAT1(16) STRUCTURE(X(6)BYTE) DATA('RET   ',' ?    ','PCHL  ',
                'SPHL  ','XTHL  ','XCHG  ','DI    ','EI    ','STAX B','LDAX B',
                'STAX D','LDAX D','SHLD  ','LHLD  ','STA   ','LDA   ');
46  2       DEC NEMMAT2(8) STRUCTURE(X(5)BYTE) DATA(60H,'JMP ',20H,' ?  ',40H,'OUT ',
                40H,'IN  ',60H,'CALL',20H,' ?  ',20H,' ?  ',20H,' ?  ');
47  2       DEC DIREC ADDRESS;/* Direccion de comienzo de ASM85 */
48  2       DEC CODEB BASED DIREC BYTE;
49  2       DEC CODEA BASED DIREC ADDRESS;
50  2       DEC (N,DECOD,CODIGOL,CODIGOH,POSNEM) BYTE;
51  2       CODIGOL=CODEB AND 07H;
52  2       CODIGOH=ROR(CODEB,3) AND 07H;
53  2       DO N=0 TO 31;
54  3         NEMON(N)=' ';
55  3       END;
56  2       CALL ADDASC(.DIREC,POSADD);
57  2       NEMON(POSPNT)=':';
58  2       CALL BYTEASC(DIREC,POSCOD0);
59  2       IF ((CODEB XOR 0C0H)AND 0C0H)<>0 AND (CODEB AND 0C0H)<>0 THEN
60  2         IF (CODEB AND 40H)=0 THEN DO; /* Operac. Aritm.-logica r */
62  3           CALL LIST(POSNEM3,.NEM2(CODIGOH));
63  3           NEMON(POSOPR0)=R(CODIGOL);
64  3         END;
65  2         ELSE IF CODEB=76H THEN CALL LIST(POSNEM3,.('HLT'));
67  2              ELSE DO; /* MOV r1,r2 */
68  3                CALL LIST(POSNEM3,.('MOV'));
69  3                NEMON(POSOPR0)=R(CODIGOH);
70  3                NEMON(POSCOM1)=',';
71  3                NEMON(POSCOM2)=R(CODIGOL);
72  3              END;
73  2       ELSE DO;
74  3         IF ((DECOD:=DECMAT0(CODIGOL OR (ROR(CODEB,3) AND 08H)))AND 60H)<>0 THEN
75  3           DO CASE ROR(DECOD,3) AND 03H;
76  4             DO;/* Grupo Nemonico */
77  5               CALL LIST (POSNEM3,.NEMMAT0(ROL(DECOD,3)AND 38H OR CODIGOH));
78  5               POSNEM=POSOPR0;
79  5             END;
80  4             DO;/* Registro r */
81  5               CALL LIST(POSNEM3,.NEM0(DECOD AND 07H));
82  5               NEMON(POSOPR0)=R(CODIGOH);
83  5               POSNEM=POSCOM2;
84  5             END;
85  4             DO;/* Condicion CC */
86  5               CALL LIST(POSNEM1,.NEM0(DECOD AND 07H));
87  5               CALL LIST(POSNEM12,.CC(CODIGOH));
88  5               POSNEM=POSOPR0;
89  5             END;
90  4             DO;/* RST n */
91  5               CALL LIST(POSNEM3,.('RST'));
92  5               NEMON(POSOPR0)='0' OR CODIGOH;
93  5             END;
94  4           END;
95  3           ELSE DO;/* Depende del 6 o 4 bit */
96  4             N=ROR(CODEB, 3 +(ROL(DECOD,1)AND 01H)) AND 03H;
97  4             IF ( (DECOD:=DECMAT1(DECOD AND 07H)
                    .X(ROR(CODEB,5-(ROL(DECOD,2)AND 02H))AND 01H))AND 60H)<>0 THEN
98  4               DO CASE ROR(DECOD,3) AND 03H;
99  5               DO;/* Mnemonico variable */
100 6                 CALL LIST (POSNEM6,.NEMMAT1(ROL(DECOD,2) AND 0CH OR N));
101 6                 POSNEM=POSOPR0;
102 6               END;
103 5               DO;/* Registro R0 */
104 6                 CALL LIST(POSNEM4,.NEM1(DECOD AND 07H));
105 6                 CALL LIST(POSOPR02,.R0(N));
106 6                 POSNEM=POSCOM3;
107 6               END;
108 5               DO;/* Registro R1 */
109 6                 CALL LIST(POSNEM4,.NEM1(DECOD AND 07H));
110 6                 CALL LIST (POSOPR03,.R1(N));
111 6               END;
112 5               ;
113 5             END;
114 4             ELSE DO; /* Un Mnemonico por codigo */
115 5               CALL LIST(POSNEM4,.NEMMAT2(ROL(DECOD,2) AND 04H OR N).X(1));
116 5               DECOD=NEMMAT2(ROL(DECOD,2) AND 04H OR N).X(0);
117 5               POSNEM=POSOPR0;
118 5             END;
119 4           END;
```

```
120   3             IF (DECOD AND 40H)<>0 THEN DO;/* 2 o 3 bytes */
122   4               IF POSNEM<>POSOPR0 THEN NEMON(POSNEM-1)=',';
124   4               CALL BYTEASC(DIREC:=DIREC+1,POSCOD1);
125   4               IF (DECOD AND 20H)<>0 THEN DO;/* 3 bytes */
127   5                 CALL ADDASC(DIREC,POSNEM);
128   5                 CALL BYTEASC(DIREC+1,POSCOD2);
129   5                 DIREC=DIREC+1;
130   5               END;
131   4               ELSE CALL BYTEASC(DIREC,POSNEM);
132   4             END;
133   3             END;
134   2             RETURN DIREC+1;
135   2             END DECODE;

136   1             END DASM85;


MODULE INFORMATION:

     CODE AREA SIZE     = 056CH   1388D
     VARIABLE AREA SIZE = 0013H     19D
     MAXIMUM STACK SIZE = 0006H      6D
     170 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
:0A0000[illegible]22DB20C37300A9
:10000B0022D920F5AF32DC20E122D320E122D720DB
:0A001800210000392 2D520C3810029
:0B00240000F53E7AD320C31000C3E42097
:0300340000C3E720FF
:10003C00E5D5F53A9F20473ADF20329F207B32DF12
:10004C00202AA220EB2AE02022A220EB22E0202A68
:10005C00A020EB2AE22022A020EB22E220CDF4000B
:10006C00E1B30F1D1E1C9217E2036012B36F122205
:10007C00DD20C300021D320F9C5D520E60721DCF8
:10008C00208632DB202ADD20F93E1B30FBC93ADBDF
:10009C00202E60832DC202AD7207EFEF3C2AF00AF68
:1000AC00C3B400FEFBC2B900E0832DC203E40D3F2
:1000BC00253EDBD3243EFAD3203ADB2047E607F675
:1000CC001B3078E608C2D800F3C3DC0037D2DC0065
:1000DC0021000039220D2021CF20F9D1C1F1E1F935
:1000EC002AD720E52AD920C911FFFF1B7AB3C2F702
:1000FC0000C92EDBC305012ED36122EA203EC93292
:10010C00EC207BCDEA20C9505343500520D7204159
:10011C004624234445484C49D420D320D220D12008
:10012C00D020CF20DA20D920DB2001494E464C4795
:10013C002002 4F5554464C47044354C464C470BFB
:10014C00535004C464C47104246464C4720204245A3
:10015C00464C4720402434F464C4720B04F55544116
:10016C00424C05400A401440284905040A040404189
:10017C00B4413D4F264646264E20260D0A075344D1
:10018C00432D3B3520505245504152414 4F26435F
:10019C0048454348 3D26535020202050432026200D9
:1001AC0020262020202026425546464552 3D26201A
:1001BC00545241434 53D262052414D425553593DE1
:1001CC00026504F5254282629 3D2607230D0A434D0D
:1001DC00442D455252260723 0D0A505 24D2D45529F
:1001EC005226317F20317F203E7AD3203E00D31817
:1001FC00DB1BE6380F0F0F4FCDD6093EC0D31B3E90
:10020C0000D301D309D301D309D301D3093E40D381
:10021C0001D309DB1AF64BD301D3093E37D301D3F3
:10022C0009210020224202100 1F22A62022A22086
:10023C0022A020DB1AE648F620329F20E640FE007F
:10024C00CA0909210000228220B220B020018701 0B
:10025C00CD5F0CCD540C0E2ACD630BCD270C32BACE
:10026C0020FE44C2D502CDFF0D1FD2CF022AB420EE
:10027C0022AB20CD540C2AAB20444DCD800C0E3A37
:10028C00CD630BCDF70B1FD2CC020E20CD630B2A06
:10029C00A8204ECD350C2AAB2023224B2011B62048
:1002AC00CD8E14D2B502C35F023E0F11AB20CD4CE7
:1002BC00143E00CD7414B5C2C902C37F02C38F02B1
:1002CC00C3D202C3FD08C3F1083ABA20FE53C220C0
:1002DC0003CD390E1FD21A032AAB2022AC203ABA19
:1002EC0020FE0DCA17032AAC204ECD350C0E2DCD99
:1002FC00630BCD390E1FD20D032AA8207D2AAC200A
:10030C00772AAC202322AC20C3EA02C31D03C3FD11
:10031C0008C3F1083ABA20FE47C24903CD390E1F73
:10032C00D235032AA8022D7203ABA20FE0DC24388
:10033C0003CDC500C34603C3FD08C3F1083ABA2078
:10034C00FE43C29703CD390E1FD25E032AA8202A8A
:10035C00D7203ABA20FE0DC291032AD5202B2B23BE
:10036C00D52022AC20E52AD720EBE17323722AD5C5
:10037C00202B2B22D52022AC2001080071237 0CD1C
:10038C00C500C39403C3FD08C3F1083ABA20FE4B44
:10039C00C23004CD390EF53ABA20D63DD6019FC1F4
:1003AC0048A11FD22D0421000022AC20CD270C32F5
:1003BC00EA204FCD7A0932BC20FEF0CA1204CD27EB
:1003CC00C32BA204FCD7A0932BB20FEF0CAEC03B6
:1003DC003ABC200F0F0F0F21BB20E62377C3F60387
:1003EC00CD3C0A1FDAF603C31C042AA8203ABC2011
:1003FC007711AC20CD3F14EB2B7323722AA8202344A
:10040C0022A820C3B803CD3C0A1FD21C04C3B803D6
:10041C00CD540C019B01CD5F0C2AAC20444DCD80FA
:10042C00CC3F1083ABA20FE4DC26004CD970E1FE2
:10043C00D25A0424AB20E52AB420444D2AB620EB1F
:10044C00E10A1203132B7CB5C24D04C35D04C3FD3A
:10045C0008C3F1083ABA20FE52C25E05CD270C3211
:10046C00BA20FE51C20705CD540C21BD2036003EEA
:10047C00121BD20BEDABF04CD4F0D2E2DCD630B77
:10048C00CD390E1FD29F042AAC20E52AA820EBE11F
:10049C007323723ABA20FE0DC2AA04C35F02CD3C8C
:1004AC00A2F1FD2B504C3FD083ABD203C32BD20033
:1004BC00C27B0421BD2036003E0B21BD20BEDA04DB
:1004CC0005CDBE0C0E2DCD630BCD390E1FD2E40421
:1004DC002AA8207D2AAC2077 3ABA20FE0DC2EF0460
:1004EC00C35F02CD3C0A2F1FD2FA04C3FD083ABDEC
:1004FC00203C328D20C2C404C35B053ABA20FE0D89
:10050C00C25805CD540C21BD2036003E0121BD20D2
:10051C00EDA3205CD4F0D0E20CD63033ABD203C1B
:10052C0032BD20C2170521BD2036003E0821BD205A
:10053C00EDA5205CDBE0C0E20CD630B3ABD203C6D
:10054C0032BD20C23705CD540CC35B05C3F408C3C0
:10055C00F1083ABA20FE45C2A605CDFF0D1FD2A068
:10056C0052AB42022A820CD540C2AA820444DCD15
:10057C00FC1022AB2011B620CD8E149F2FF5CDF79C
:10058C00BC148A11FD29D05CD970CCD540CC37641
:10059C00C3A305C3FD08C3F10B3ABA20FE50C237
:1005AC00D072AD72022A820CD970E1FD20707CDE2
:1005BC00540C3A9F20E604FE00CA160621BD2036D4
:1005CC003E2321BD20BEDAE5050E20CD630B3A9B
:1005DC00ED203C32BD20C2CD0501A201CD5F0C2156
:1005EC00BD2036003E0821BD20BEDA16060 1AB0147
:1005FC00CD5F0C2ABD2026000 11B01094ECD630BDB
:10060C003ABD203C32BD20C2F0052AB42022D720AE
:10061C002AB8202322AA202AAA202B22AA203E0074
:10062C00CD7414B5C6FF9F01B62011D720F5CD7E31
:10063C0014B5C6FF9FC14BA1F5CDF70BC14BA11F4A
:10064C00D204072AD720444DCDFC1022AC20EB213C
:10065C00B220CD99149F2F11B02001AC20F5CD7EB6
:10066C00149F2FC148A11FD2FE06CD540CCD970C60
:10067C00CD9A003A9F20E604FE00CAFB0601AE01AB
:10068C00CD5F0C21BD2036003E0121BD20BEDAC657
:10069C0062ABD2026000117012 9095E2356EB22EC
:1006AC00AC202AAC204E2346CD800C0E20CD630B03
:1006BC003ABD203C32BD20C29406213D2036003EFE
:1006CC00B218D20BEDAFB062ABD2026000124012C
:1006DC0029795E2356EB22AC202AAC204ECD350CDA
:1006EC00E20CD639B3ABD203C32BD20C2CB06C3DD
:1006FC0007CD9A00C32306C30A07C3FD08C3F143
:10070C0083ABA20FE01C2BE0721BD2036003E07C2
:10071C00218D20BEDA44072ABD204DCDE90C3ABDDF
:10072C0020FE06C23507CD540C0E20CD630B3ABD0E
:10073C00203C328D20C21A0701B301CD5F0C2AAEA2
:10074C0020444DCD800C0E2CCD639B2AA420444D9F
:10075C00CD800C01BB01CD5F0C2AB220444DCD8065
:10076C00C0E2CCD630B2AB020444DCD800C01A275
:10077C002011A020CD7E14D2A10701A22011A42008
:10078C00CD7E1401A62011A020E5CD7E14C1092236
:10079C00E4E20C3AD0701A22011A020CD7E1422AB45
:1007AC002001C301CD5F0C2AAE20444DCD800CC37B
:1007BC00F1083ABA20FE70C22E08CD390E1FD22B8A
:1007CC0083ABA20FE0DC2FB07CD540C01CD01CD69
:1007DC005F0C2AAB207D4FCD350C01D301CD5F00C9
:1007EC002AA8207D4FCDFE004FCD350CC32B083AE7
:1007FC008A20FE3DC228082AA82022AC20CD390EF2
:10080C001FD222082AAC207D2AAB20F57D5FC14882
:10081C00CD0301C32508C3FD08C32B08C3FD08C3C2
:10082C00F1083ABA20FE42C25808CDFF0D1FD25231
:10083C0082AB42022A62022A22022AB220FB
:10084C0022A420C35508C3FD08C3F1083ABA20FE00
:10085C0054C27C08CDFF0D1FD276082AB42022B2D8
:10086C00202AB62022B020C37908C3FD08C3F108A2
:10087C0034BA20FE14C2E608D809E605EE05D600FE
:10088C00D6019FF5DB01E602D600C6FF9FC148A149
:10089C001FD2BD08CDF00A32BA20FE14C2AE08C376
:1008AC005F02CD410E2F1FD2BD082ABA204DCD24B
:1008BC00AD809E602D600C6FF9FF5DB01E605EE72
:1008CC0005D600D6019FC148A11FD2E908CD130B5D
:1008DC004FCD630BC38408C3F108CD410E2F1FD23B
:1008EC00F108C3F40BC35F0201D601CD5F0CC35FEE
:1008FC000201E201CD5F0CC35F02C348093A9F209D
:10090C00F680329F20E601FE00C248093A9F20E69D
:10091C0040D600C6FF9FF5DB01E602D600C6FF9F5E
:10092C00C148A11FD23E09CDF00AFE1BC23E09C32D
:10093C00F101CD51034FCD6308C318093A9F20E643
:10094C0040D600C6FF9FF5DB01E602D600C6FF9F2E
:10095C00C148A11FD26E09CDF00AFE1BC26E09C39D
:10096C00F101CD130B4FCD630BC34809FB7621BFAF
:10097C0020713E3921BF20969F2FF57ED6309F2FB8
:10098C00C148A11FD299093ABF20D630C93E462191
:10099C00BF20969F2FF57ED6419F2FC148A11FD215
:1009AC00B6093ABF20D641C60AC93E6621BF209679
:1009BC009F2FF57ED6619F2FC148A11FD2D3093A34
:1009CC00BF20D661C60AC93EF0C921C020712AC019
:1009DC0026000016E012909237ED31D2B6E7DD3A9
:1009EC00CC921C12071DB20E610FE00CAFE09C320
:1009FC00F2093AC120D322C93A9F20E602FE00C276
:100A0C00180ADB20E610D600D6019FC9DB09E605E3
:100A1C00EE05D600D6019FC921C22071DB09E6057F
```

```
:100A2C00EE05FE00CA360AC3280A3AC220D308C90A
:100A3C003ABA20D60DD6019FF53ABA20D62CD6015B
:100A4C009FC148B1F53ABA20D620D6019FC148B112
:100A5C00F53ABA20D60AD6019FC14831F53ABA2068
:100A6C00D63DD6019FC148B11FD27B0A3EFFC93E7D
:100A7C0000C921C320713A9F20E602FE00C2960AEB
:100A8C002AC3204DCDEE09C39D0A2AC3204DCD2487
:100A9C00AC93A9F20E620FE00C2EF0ACD040A1FC5
:100AAC00D2FF0A2AA2204ECD7E0A2AA2202322A20D
:100ABC002011A420CD8E14B5C2CD0A2AA62022A2C4
:100ACC002001A02011A220CD7E14B5C2E70A3A9FC6
:100ADC0020E6EFF620329F20C3EF0A3A9F20E6CFA4
:100AEC00329F20C9DB01E602FE00C20C0B3E27D36D
:100AFC0031D801E602FE03C20C03CD9E0AC3FD0A0F
:100B0C003E17D301DB00C9DB09E602FE00C22F0B46
:100B1C003E27D309DB09E602FE00C22F0BCD9E0A4D
:100B2C00C3200B3E17D309DB08C921C42071DB019C
:100B3C00E605EE05FE00CA4B0BCD9E0AC33A083AF6
:100B4C00C420D300C9DB20E602FE00C2609BCD9EA0
:100B5C00AC35103DB21C921C52071A9F20E64005
:100B6C00FE00CA780B2AC5204DCD360B3A9F20E4E5
:100B7C0080FE00CAF60B3A9F20E608FE00CADF0B87
:100B8C003A9F20E610FE00CA9C0BCD9E0AC38C0B2C
:100B9C002AA0203AC520772AA0202322A02011A425
:100BAC0020CD8E14B5C2BA9B2AA62022A02001A2F9
:100BBC002011A020CD7E14B5C2D40B3A9F20E6DFC5
:100BCC00F610329F20C3DC0B3A9F20E6CF329F20D9
:100BDC00C3F6083A9F20E620FE00C2EF0BCD9E0A17
:100BEC00C3DF0B2AC5204DCD7E0AC9DB01E602FE10
:100BFC0000CA240CCDF00A32BA20FE13C2240CCD4C
:100C0C00F00A32BA20FE11CA240C3ABA20FE1BC2DA
:100C1C00210C3E00C9C3030C3EFFC9CDF00A32C6F5
:100C2C00204FCD630B3AC620C921C720713AC720BB
:100C3C00F0F0F0F4FCD36104FCD6303A2C7204D22
:100C4C00CD36104FCD630BC90E0DCD630B0E0ACDF7
:100C5C006303C921C920702B712ACB207EFE26CABD
:100C6C007F0C2AC8204ECD630B2AC8202322C82013
:100C7C00C3650CC921CB20702B712ACA207C4FCDA7
:100C8C00350C2ACA207D4FCD350CC9217F2022ACD2
:100C9C0020119E2021AC20CD9914DABD0C2AAC2059
:100CAC004ECD630B1101002AAC201922AC20D29D31
:100CBC00CC92ABD2026000124012905E2356E30C
:100CCC0022AC202ABD202600011801094ECD630B4E
:100CDC00E3DCD630B2AAC204ECD350CC921CC205A
:100CEC007121CD2036003E0521CD20BEDA210D111B
:100CFC007002ACC202603CD59140136010 9E52A1B
:100D0C00CD2026000101000 9C1094ECD630B21CD78
:100D1C0034C2F20C017E01CD5F0C1107002ACCED
:100D2C00202600CD59140136010 93A9F20A6FE0059
:100D3C00C2480D0181 01CD5F0CC34E0D018401CD64
:100D4C005F0CC92ABD2026000117012905E2356E14
:100D5C00E322AC202ABD2026000113012909237E99
:100D6C004FCD630B2ABD2026000113012909 6E4DBE
:100D7C00CD63030E3DCD630B2AAC204E2346CD80AC
:100D8C0009CC921BE2036003CD270C32BA204FCD7AAB
:100D9C00732BB20FEF0CAD90D21BE2036FF2ABB7A
:100DAC0020260022AB20CD270C32BA204FCD7A095C
:100DBC00328B20FEF0CAD90D2AAB2029292929E501
:100DCC002ABB2026000C10922AB20C3B20DCD3C0AA3
:100DDC00C921CE2071110700 2ACE202600CD59142E
:100DEC0001360109 3A9F20AE329F202ACE204DCDEC
:100DFC00E99CC9CD8E0D1FD2360E2AAB2022342 0A4
:100E0C002322AB2022B6203ABA20FE0DC2360E3EB6
:100E1C00FFC9CD8E0D1FD2360E2AAB2022B6203A3D
:100E2C00BA20FE0DC2360E3EFFC93E00C9CD8E0D66
:100E3C0021BE20A6C93ABA20FE10C2510E0E07CD13
:100E4C00DD0DC3940E3ABA20FE03C2610E0E02CD24
:100E5C00DD0DC3940E3ABA20FE04C2710E0E03CD02
:100E6C00DD0DC3940E3ABA20FE09C2B10E0E00CDE0
:100E7C00DD0DC3940E3ABA20FE0FC2910E0E01CDB9
:100E8C00DD0DC3940E3E00C93EFFC9CDFF0D2FF5FD
:100E9C00CD3C0A2FC148A11FD2AA0E3E00C9210188
:100EAC0022B8203ABA20FE0DCAD00ECD390E1F42
:100EBC00D2C50E2AAB2022B8203ABA20FE0DCAD0DC
:100ECC00E3E00C93EFFC9208001822B294A2133E9
:100EDC0037837404758542386A2322632C2D312000D3
:100EEC0021300142434445484C4D41422044204B66
:100EFC00205350422020442020482020505357 4E4D
:100F0C005A5A204E434320504F504550204D2049B3
:100F1C004E524443524D5649522020442020432 0E1
:100F2C00205055553485 04F50204C5B4920444144 79
:100F3C0020494E582044435820414444414443539 3
:100F4C0055425342424 14E415852414F5241434DFA
:100F5C00504E4F50203F20203F20203F2052494DE3
:100F6C00203F2053494D203F20524C43525243574
:100F7C00414C52415244414143 4D41535443434DE2
:100F8C0043414449414349535549534249414E49D0
:100F9C005852494F52494350495245542020202021
:100FAC003F2020202050434 84C2020535048 4C20B8
:100FBC0020585448 4C2020584348472020444920 6E
:100FCC0020202020454920202020535441582042 4C89
:100FDC004441582042535441582044 4C44415820D9
:100FEC0044534 84C4420204C484C4420205354 41FA
:100FFC00202020204C444120202060 4A4D502020 20AD
:10100C003F2020404F55542040494E2020604341 02
:10101C004C4C20203F2020202 03F2020202 03F200F
:10102C0020484C544D4F5652535421ED20713AEDFB
:10103C0020E60F32ED204F3E09B9D251103AED20B7
:10104C00C641D60AC93AED20C630C921F020722B10
:10105C007328713AEE20E61F32F22021F12036007C
:10106C003AEE20070707E6073D21F120BEDA9F1074
:10107C002AF120600EB2AEF20193AF120E521F2B3
:10108C0020864F0600217F2009C10A7721F12034E8
:10109C00C26C10C921F520732B702B712AF3207EA2
:1010AC000F0F0F0F4FCD36102AF52026000 17F2091
:1010BC0009772AF3204ECD3610F53AF5203C32F55F
:1010CC004F0600217F2009C14871C921F82073E7
:1010DC0023702B712AF62023444D2AFB20EBCDA03F
:1010EC00103AFB203C3C2AF6205F444DCDA010C9A4
:1010FC0021FA20702B712AF9203E07A632FD202AF6
:10110C00F9207E0F0F0FE60732FE2021FB203600 60
:10111C003E1F21FB20BEDA37112AFB2026000 17F5F
:10112C0020093620 21FB2034C21C111E0001F9209D
:10113C00CDDB1021B320363A2AF920444D1E06CDF5
:10114C00A0102AF9203EC0AEE6C0D600C6FF9FF51F
:10115C003EC0A6D600C6FF9FC148A11FD2D6112AF9
:10116C00F9203E40A6FE00C29B111103002AFE206E
:10117C002600CD591401450F09EB0E74CD57102ADA
:10118C00FD20260001EF0E097E329820C3D3112AD0
:10119C00F9207EFE76C2AF11112D100E74CD571082
:1011AC00C3D31111301 00E74CD57102AFE20260017
:1011BC0001EF0E097E3298202199 20362C2AFD2031
:1011CC002600097E329A20C33A142AF9207E0F0F8A
:1011DC00FE60321FD20B64F060021D30E097E3202
:1011EC00FC20E660FE00CAAC123AFC209F0F0FE6A2
:1011FC0034F060021A11209095E2356E3E93AFC4
:10120C002007070 7E63821FE20B66F26001103 00E1
:10121C00CD5914015D9F09E30E74CD571021FF2031
:10122C003619C3A9123AFC20E6076F26001103 00F9
:10123C00CD59140118F09EB0E74CD57102AFE204B
:10124C00260001EF0E097E32982021FF20361BC3A9
:10125C00A9123AFC20E6076F26001103 00CD5914A1
:10126C0011B0F09EB0E34CD57102AFE2026002946
:10127C001030F09EB0E35CD571021FF203619C36A
:10128C00A9121133100E74CD57103AFE20F63032DD
:10129C009820C3A9120A1231125E128E12C3D613F1
:1012AC002AF9203AFC2007E601C6034FCD6D14E45F
:1012BC00332FB203AFC20E6076F26002901E30EDF
:1012CC0009E52AF9203AFC200707E6024F3E059172
:1012DC004FCD6D14E6014F0600E1097E32FC20E68D
:1012EC0060FE00CA93133AFC200F0F0FE6034F0663
:1012FC002188130 9095E2356EBE93AFC20070705
:10130C00E60C21FB20B66F26001106 00CD59140196
:10131C00A50F09EB0ED4CD571021FF203619C39021
:10132C00133AFC20E6076F26002929012D0F09EB43
:10133C00E94CD57102AFB2026002901F70E09E3D3D
:10134C00E59CD571021FF20361CC390133AFC20A8
:10135C00E6076F26002929012D0F09EB0E94CD5786
:10136C0010110300 2AFB202600CD591401FF0E0991
:10137C00E30E79CD5710C39013C3901307132D1395
:1013BC0059138513C3D6133AFC200707E60421FB37
:1013CC0020366F26001105 00CD59140105100 0166
:1013AC0010009EB0E94CD57103AFC200707E60418
:1013BC0021F820366F26001105 00CD59140105 1034
:1013CC0097E32FC2021FF2036193AFC20E640FE33
:1013DC00CA3A143AFF20FE19CAF5133AFF203D11
:1013EC004F0600217F2009362C2AF9202322F920D0
:1013FC00444D1E09CDA0103AFC20E620FE00CA2E5A
:10140C00142AF920444D2AFF20EBCDD8102AF920BC
:10141C0023444D1E0CCDA0102AF9202322F920C301
:10142C003A142AF920444D2AFF20EBCDA0102AF9BA
:10143C0020023C9EB5F1600EB1AB56F131A8C67C952
:10144C00EB5F1600EB1AA56F131AA467C9444D2164
:10145C0000003E1029EB29EBD26B14093DC2601440
:10146C00C97E0F0DC26E14C95F16007B956F7A9CF6
:10147C00367C969604E23461A916F131A9867C96F32
:10148C0026001A956F131A9C67C95F16007B965F2E
:06149C007A239E57EBC904
:0001EE0110
```

CAPITULO 3:

Programa QUILER1.PLM

(Configuracion del MDS 221 como Terminal).

PROGRAMA QILER1.PLM.-

Este programa, escrito en lenguaje PLM/8Ø, residente en el Sistema de Desarrollo MDS-221, configura a éste como terminal. Su historia se remonta al QUILE6, programa escrito en lenguaje Ensamblador y que formó parte del proyecto de los alumnos Quintana y Ley Bosch.

Con el Qilerl se advierten varias ventajas sobre dicho predecesor como son:

1.- Se puede variar la velocidad de transmisión: 11Ø, 3ØØ, 6ØØ, 12ØØ, 24ØØ, 48ØØ, 96ØØ, 192ØØ. Frente a la velocidad fija del predecesor de 1200 baudios.

2.- El cierre del fichero (CTRL-C) es manual, frente al automático del anterior.

3.- Existe la posibilidad de enviar cualquier caracter, excepto CTRL-B. Con el anterior, ciertas teclas, llamadas de control, núnca podrían enviarse al ser utilizadas por el programa.

4.- Se puede cambiar el caracter de fin de fichero frente al fijo CTRL-Y del anterior.

5.- Capacidad de ser parado desde fuera a través de Control-S y Control-Q. Es decir, cuando el sistema de desarrollo recibe de fuera un Ctrl-S, no puede enviar ningún caracter hasta que reciba un Ctrl-Q.

Al ejecutar el programa QILER1.LOC, no están habilitados los caracteres de control, que se activan pulsando Control-B. Aparecerá entonces en pantalla el mensaje:

CONTROL = ON

En este momento se pueden usar los controles que a continuación se muestran. Para desactivar los caracteres de control basta con pulsar de nuevo CTRL-B. Aparecerá enton--

ces en la pantalla el mensaje:

CONTROL = OFF

- Controles del QILER1.PLM.-

Control - A: envío de un fichero.

Al pulsar este control aparecerá el mensaje:

NOMBRE DEL FICHERO A LEER =

Entonces se podrán introducir hasta 9 caracteres con el formato: XXXXXX.XXX

y que designarán el nombre del fichero a enviar. Se pueden borrar caracteres con RUBOUT. Si no se hubieran introducido caracteres ó se hubieran borrado, al pulsarlo abortará el control y aparecerá el mensaje:

WARNING: Apertura abortada.

De haber introducido un nombre correcto aparecería el mensaje: WARNING: Cerrado el fichero XXXXXX.XXX

Control - B: activar / desactivar controles.

Control - C : cierre del fichero.

Al pulsar este control se cierra el fichero abierto con CTRL-A. Aparece entonces el mensaje:

WARNING: Cerrado el fichero XXXXXX.XXX

Control - D : Reasignar caracter fin de fichero.

Al pulsar este control aparecerá en pantalla el mensaje:

Código fin de fichero = _

Entonces hay que pulsar la tecla que se quiere que sea el carácter de fin de fichero. Aparecerá entonces el mensaje: WARNING: Código fin de fichero = XX

donde XX es en hexadecimal el nuevo caracter de fin de fichero. El caracter por defecto de fin de fichero es Con-

trol-Y.

- Control - I : activar impresora.

Al pulsar este control, el canal de impresora queda abierto y se envía por ese canal el buffer cuando éste se llene. Para cerrar el canal basta pulsar Control-C, que además de la impresora cierra el fichero que estuviera abierto.

- Control - K : almacenar en disco.

Al pulsar este control aparece en pantalla el mensaje:

NOMBRE DEL FICHERO A CREAR = _

Siguiendo las normas de introducción del nombre del fichero de CTRL-A, después de introducido aparecerá el mensaje: WARNING: Abierto el fichero XXXXXX.XXX

Para cerrarlo, pulsar Control-C.

Ninguno de estos caracteres de control, estando activados, serán enviados por el canal serie, el resto,sin embargo,sí.

Bits del byte FLAG de QILER1:

7	6	5	4	3	2	1	ø
X	X	CSQ-FLG	PNT-FLG	EOB-FLG	DSK-FLG	CTL-FLG	IMP-FLG

IMPFLG: flag de activación de impresora.

CTLFLG: flag de control.

DSKFLG: flag de activación de disco.

EOBFLG: (end of block), falg de activación de fin de fichero.

PNTFLG: (point flag) flag de punto en nombre de fichero.

CSQFLG: flag de Control-S / Control-Q.

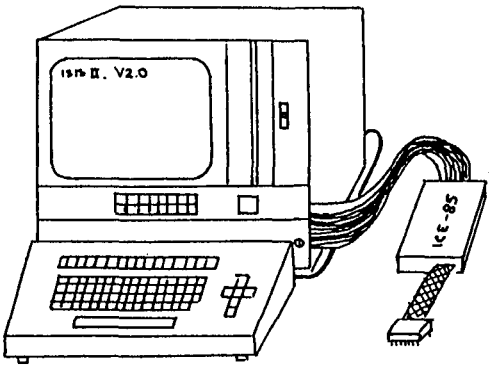
X : no significativo.

- HOME : estado break del terminal.

Al pulsar la tecla "HOME", el sistema de desarrollo se coloca en estado Break momentáneamente.

- ESCAPE : Volver al ISIS-II.

Al pulsar esta tecla, se da por terminado el programa QILERl y vuelve al ISIS-II.



Sistema de desarrollo INTEL MDS 221

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE CONEX
OBJECT MODULE PLACED IN QILER1.OBJ
COMPILER INVOKED BY:  PLM80 QILER1.PLM WORKFILES(:F0:,:F0:) NOPAGING PAGEWIDTH(100) DEBUG NOINTVECTO
                      -R


                /*************************************************************************/
                /*Este programa mantiene la comunicacion entre un sistema de desarrollo INTE-*/
                /*LLECT MDS 221 u un ordenador de la serie HP-3000. La comunicacion es: serie*/
                /* asincrona *64, velocidad de transmision 1200 baudios, 8 bits de longitud  */
                /* del caracter, sin paridad, u con 2 bits de stop.                          */
                /* Asimismo realiza la gestion u control de la impresora, u de la unidad de  */
                /* diskette del Sistema de Desarrollo. Para ello, se utilizan los siguientes */
                /* controles:                                                                */
                /*  -CTRL I:     Activar la impresora.                                       */
                /*  -CTRL K:     Activar disco Sistema Desarrollo.                           */
                /*  -CTRL A:     enviar fichero desde el Sistema de Desarrollo al ordenador. */
                /*  -HOME:       Enviar una condicion BREAK al ordenador.                    */
                /*  -ESC:        Abortar este programa u volver al sistema operativo ISIS II */
                /*  -CTRL-B:     Habilitar caracteres de control.                            */
                /*  -CTRL-C:     Cierre de fichero manual.                                   */
                /*  -CTRL-D:     Reasignar caracter fin de fichero.                          */
                /*  -CTRL-K:     Crear fichero de disco.                                     */
                /*  -CTRL-Q:     Continuar la comunicacion.                                  */
                /*  -CTRL-S:     Parar la comunicacion.                                      */
                /*  -CTRL-Y:     Fin de fichero.                                             */
                /*************************************************************************/

   1            CONEX:  DO;
   2    1           DECLARE LIT LITERALLY 'LITERALLY';
   3    1           DECLARE DEC LIT 'DECLARE';
   4    1           DEC (PBUFFER,I,WAFT,CAFT,RAFT,OAFT,ACCESS,STATUS,CONLEC)ADDRESS;
   5    1           DEC (FLAG,DATIN,INKEY,DATOUT,MFE) BYTE;
   6    1           DEC BAUDCNT(8) STRUCTURE (LSB BYTE, MSB BYTE) DATA(0AEH,12H,40H,0,20H,0,
                    10H,0,08H,0,04H,0,02H,0,01H,0);
   7    1           DEC CTRLA   LIT '01H',  CTRLB   LIT '02H',  CTRLC   LIT '03H',
                        CTRLD   LIT '04H',  ENQ     LIT '05H',  ACK     LIT '06H',
                        BEL     LIT '07H',  BS      LIT '08H',  CTRLI   LIT '09H',
                        LF      LIT '0AH',  CTRLK   LIT '0BH',  CR      LIT '0DH',
                        CTRLQ   LIT '11H',  CTRLS   LIT '13H',  ESC     LIT '1BH',
                        CTRLV   LIT '16H',  CTRLX   LIT '18H',  CTRLY   LIT '19H',
                        CTRLZ   LIT '1AH',
                        TAMBUF  LIT '1024', /* longitud en bytes del buffer */
                        HOME    LIT '1DH', /* Condicion BREAK a la USART */
                        RBOUT   LIT '7FH',  TRUE    LIT '0FFH', FALSE   LIT '00H',
                        USTCMD  LIT '0F7H', USTSTS  LIT '0F7H',
                        TMRCMD  LIT '0F3H', TMRCNT  LIT '0F1H',
                        RRDY    LIT '02H',  TRDY    LIT '01H',  TEMPTY  LIT '04H',
                        TE      LIT '01H',  DTR     LIT '02H',  RE      LIT '04H',
                        SBRK    LIT '08H',  RTS     LIT '20H',  ER      LIT '40H',
                        IMPFLG  LIT '01H',  CTLFLG  LIT '02H',  DSKFLG  LIT '04H',
                        EOBFLG  LIT '08H',  PNTFLG  LIT '10H',  CSQFLG  LIT '20H';
   8    1           DEC NFILE(10H) BYTE ,BUFFER(TAMBUF) BYTE;
                /* ********************** PROCEDIMIENTOS EXTERNOS ********************** */
                    /* Apertura de un fichero */
   9    1       OPEN:   PROCEDURE (AFTNPTR,FILE,ACCESS,MODE,STATUS) EXTERNAL;
  10    2               DEC (AFTNPTR,FILE,ACCESS,MODE,STATUS) ADDRESS;
  11    2               END OPEN;
                    /* Cierre de un fichero */
  12    1       CLOSE:  PROCEDURE (AFTN,STATUS) EXTERNAL;
  13    2               DEC (AFTN,STATUS) ADDRESS;
  14    2               END CLOSE;
                    /* Escritura en un fichero */
  15    1       WRITE:  PROCEDURE (AFTN,BUFFER,COUNT,STATUS) EXTERNAL;
  16    2               DEC (AFTN,BUFFER,COUNT,STATUS) ADDRESS;
  17    2               END WRITE;
                    /* Lectura de un fichero */
  18    1       READ:   PROCEDURE (AFTN,BUFFER,COUNT,ACTUAL,STATUS) EXTERNAL;
  19    2               DEC (AFTN,BUFFER,COUNT,ACTUAL,STATUS) ADDRESS;
  20    2               END READ;
                    /* Console Input */
  21    1       CI: PROCEDURE BYTE EXTERNAL;
  22    2           END CI;
                    /* Console Output */

  23    1       CO: PROCEDURE (CHAR) EXTERNAL;
  24    2           DEC CHAR BYTE;
  25    2           END CO;
```

```
26  1     IOSET:  PROCEDURE (CONFIG) EXTERNAL;
27  2             DEC CONFIG BYTE;
28  2             END IOSET;

29  1     LO: PROCEDURE (CHAR) EXTERNAL;
30  2         DEC CHAR BYTE;
31  2         END LO;

              /* Presentacion en pantalla del tipo de error del fichero */
32  1     ERROR:  PROCEDURE (ERRNUM) EXTERNAL;
33  2             DEC ERRNUM ADDRESS;
34  2             END ERROR;
          /* Detector de tecla; si se ha pulsado devuelve 0FFH si no 00H */
35  1     CSTS:   PROCEDURE BYTE EXTERNAL;
36  2             END CSTS;
          /* Salida hacia el sistema operativo ISIS */
37  1     EXIT:   PROCEDURE EXTERNAL;
38  2             END EXIT;

          /*Salida de cadena caracteres por pantalla */
        /*************** PROCEDIMIENTOS NIVEL 0  ************************************/

39  1     SO: PROCEDURE (DATOUT); /* Serial Output. Envio del caracter hacia la USART */
40  2         DEC DATOUT BYTE;
41  2         DO WHILE (INPUT(USTSTS) AND 05H XOR 05H)<>0;
          /* se espera a que la USART este preparada */
42  3         END;
43  2         OUTPUT (0F6H)=DATOUT;
44  2         END SO;

45  1     HEXASC: PROCEDURE (HEXBYTE) BYTE;/* Conversion Hexadecimal a ASCII */
46  2             DEC HEXBYTE BYTE;
47  2             IF (HEXBYTE:=HEXBYTE AND 0FH)>9 THEN RETURN HEXBYTE + 37H;
49  2             RETURN HEXBYTE+30H;
50  2             END HEXASC;

        /* ************************ PROCEDIMIENTOS NIVEL 1 ************************ */

51  1     COSTR:  PROCEDURE (DIREC);/* Presentacion de una cadena por pantalla */
52  2             DEC DIREC ADDRESS, LETRA BASED DIREC BYTE;
53  2             DO WHILE LETRA <>'&';
54  3               CALL CO(LETRA);
55  3               DIREC=DIREC+1;
56  3             END;
57  2             END COSTR;

58  1     CRLF:   PROCEDURE;
59  2             CALL CO(CR);
60  2             CALL CO(LF);
61  2             END;

62  1     COBYTE: PROCEDURE (HEXBYTE);
63  2             DEC HEXBYTE BYTE;
64  2             CALL CO(HEXASC(ROR(HEXBYTE,4)));
65  2             CALL CO(HEXASC(HEXBYTE));
66  2             END COBYTE;

67  1     DEL0:  PROCEDURE BYTE; /* retardo, TRUE=contesto el ordenador, FALSE=no */
68  2             I=1200H;
69  2             DO WHILE I<>0;
70  3               IF (INPUT(USTSTS) AND RRDY)<>0 THEN DO;
72  4                 IF (DATIN:=(INPUT(0F6H) AND 7FH))=ENQ THEN CALL SO (ACK);
74  4                 ELSE RETURN TRUE;
75  4               END;
76  3               ELSE I=I-1;
77  3             END;
78  2             RETURN FALSE;
79  2             END DEL0;
80  1     CEFICH: PROCEDURE; /* Cerrar fichero u presentar error si lo hubiera */
81  2             CALL CLOSE (CAFT,.STATUS);
82  2             IF STATUS<>0 THEN CALL ERROR (STATUS);
84  2             ELSE DO;
85  3               CALL COSTR(.(CR,LF,'WARNING: Cerrado fichero &'));
86  3               CALL COSTR(.NFILE);
87  3               CALL CRLF;
88  3             END;
89  2             END CEFICH;
```

```
 90   1      IMPRIM: PROCEDURE;/* Salida del Buffer por impresora u/o disco */
 91   2      DIOGD:  CALL SO(CTRLS); /* Se para la comunicacion. Se lee el ultimo caracter*/
 92   2              IF DEL0 THEN /* que entro en la USART cuando se paro la comunicacion.*/
 93   2                IF DATIN=CTRLS THEN DO;
 95   3                  FLAG=FLAG OR CSQFLG;
 96   3                  PBUFER=PBUFER-1;
 97   3                END;
 98   2                ELSE IF DATIN=CTRLQ THEN DO;
100   3                  FLAG=FLAG AND NOT CSQFLG;
101   3                  PBUFER=PBUFER-1;
102   3                END;
103   2                ELSE CALL CO(BUFFER(PBUFER));
104   2              IF (FLAG AND IMPFLG)<>0 THEN DO;/* Impresora activada */
106   3                I=0;
107   3                DO WHILE I<=PBUFER;
108   4                  CALL LO(BUFFER(I)); /* Salida del buffer por la impresora */
109   4                  I=I+1;
110   4                END ;
111   3              END;
                       /* Si esta activado el flag de disco entonces: */
112   2              IF (FLAG AND DSKFLG)<>0 THEN DO;/* Escribir en fichero escritura */
114   3                CALL WRITE(WAFT,.BUFFER,PBUFER+1,.STATUS);
115   3                IF STATUS<>0 THEN CALL ERROR (STATUS);
117   3                IF (FLAG AND EOBFLG)<>0 THEN CALL CEFICH;
119   3              END;
120   2              IF (FLAG AND EOBFLG)<>0 THEN FLAG=FLAG AND 0F2H;
122   2              CALL SO (CTRLQ); /* Se continua la comunicacion */
123   2          END IMPRIM;

              /* ********************** PROCEDIMIENTOS NIVEL 2 ************************** */

124   1      CEDISK: PROCEDURE ;/* Cierre fichero disco u reset del FLAG */
125   2              CALL CEFICH;
126   2              FLAG=FLAG AND NOT(EOBFLG OR DSKFLG);
127   2              END CEDISK;
128   1      DEL1:   PROCEDURE;/* Retardo u presenta en pantalla contestacion ordenador.*/
129   2              DO WHILE DEL0;
130   3                CALL CO(DATIN);
131   3              END;
132   2              END DEL1;
133   1      ABFICH: PROCEDURE (CADADD,ACCESS) BYTE; /* Lee el nombre del fichero u lo   */
134   2          DEC (CADADD,ACCESS) ADDRESS; /* almacena en las posiciones de NFILE */
135   2          CALL COSTR(CADADD);/* TRUE=se abre el fichero ;FALSE= error apertura */
136   2          DO I=0 TO 0EH;
137   3            NFILE(I)=' ';
138   3          END;
139   2          NFILE(0FH)='&';
140   2          FLAG=FLAG AND 0EFH;/* Reset bit de punto */
141   2          DO I=0 TO 0DH;/* Colocar nombre de fichero en NFILE */
142   3      ABFIC2: IF (INKEY:=CI AND 7FH)=RBOUT THEN
143   3                IF I>0 THEN DO;/* Reset bit de punto */
145   4                  IF NFILE(I:=I-1)='.' THEN FLAG=FLAG AND NOT PNTFLG;
147   4                  CALL CO(BS);/* Borrar ultimo caracter */
148   4                  CALL CO(NFILE(I):=' ');
149   4                  CALL CO(BS);
150   4                  GOTO ABFIC2;
151   4                END;
152   3                ELSE DO;
153   4                  CALL COSTR(.(CR,LF,'WARNING: Apertura abortada',CR,LF,'&'));
154   4                  RETURN FALSE;
155   4                END;
156   3              ELSE IF INKEY=CR THEN GOTO SALT1;
158   3              ELSE IF INKEY='.' THEN FLAG=FLAG OR PNTFLG;
160   3              ELSE IF I>5 AND (FLAG AND 10H)=0 THEN DO;
162   4                CALL CO(BEL);
163   4                GOTO ABFIC2;
164   4              END;
                     IF INKEY>20H THEN CALL CO(NFILE(I):=INKEY);
167   3              ELSE DO;
168   4                CALL CO(BEL);
169   4                GOTO ABFIC2;
170   4              END;
171   3          END;
172   2      SALT1:  CALL CRLF;
                  /* .OAFT= Numero de fichero; .NFILE= Nombre del fichero; */
                  /* ACCESS=1 (read); 2 (write); 3 (update); .STATUS= Numero error */
173   2          CALL OPEN (.OAFT,.NFILE,ACCESS,0,.STATUS);
174   2          IF STATUS <>0 THEN DO;
176   3            CALL ERROR (STATUS);
177   3            RETURN FALSE;
178   3          END;
```

```
179   2          CALL COSTR(.(CR,LF,'WARNING: Abierto fichero &'));
180   2          CALL COSTR(.NFILE);
181   2          CALL CRLF;
182   2          IF ACCESS=1 THEN RAFT,CAFT=0AFT;
184   2          ELSE IF ACCESS=2 THEN WAFT,CAFT=0AFT;
               RETURN TRUE;
187   2              END ABFICH;
              /*****************************************************************************/
                        /* PRINCIPIO DEL PROGRAMA */
              /*****************************************************************************/

188   1        PRINCIP:CALL IOSET(81H);
189   1            OUTPUT(USTCMD)=40H; /* Se resetea la USART */
190   1            OUTPUT(USTCMD)=0CFH; /* Programacion de la USART del canal 2 */
191   1            CALL COSTR(.(1BH,48H,1BH,4AH,'* PROGRAMA DE COMUNICACION SERIE PREPARADO.',
                   '(C) MAYO 88 . * * * * ',0DH,0AH,'&'));/* Presenta encabezamiento. */
192   1            CALL COSTR(.(CR,LF,'VELOCIDAD: 0=110, 1=300, 2=600, 3=1200, 4=2400,',
                   ' 5=4800, 6=9600, 7=19200',CR,LF,'&'));
193   1            DO WHILE (INKEY:=CI)<'0' OR INKEY>'7' ;
194   2              CALL CO(BEL);
195   2            END;
196   1            OUTPUT(TMRCMD)=076H; /* Se programa el Timer1 para trabajar en modo 3 */
197   1            OUTPUT(TMRCNT)=BAUDCNT(INKEY-'0').LSB;/* con una frecuencia variable */
198   1            OUTPUT(TMRCNT)=BAUDCNT(INKEY-'0').MSB;
199   1            OUTPUT(USTCMD)=DTR OR RE OR TE;
200   1            FLAG=0;
201   1            MFF=CTRLY;
202   1            CALL COSTR(.(CR,LF,'LINEA DSR - &'));
203   1            IF (INPUT(USTSTS) AND 80H)=0 THEN CALL COSTR(.('DES&'));
205   1            CALL COSTR(.('ACTIVADA',CR,LF,'&'));
206   1            OUTPUT(USTCMD)=RTS OR RTS OR RE OR TE;
207   1            CALL COSTR(.('LINEAS DTR Y RTS ACTIVADAS', CR, LF,'&'));
208   1        INDBUF: PBUFER=0; /* Inicializacion del puntero del buffer */
209   1        BUCPRI: IF CSTS<>0 THEN  /* si no se ha pulsado una tecla salta a RESPUE */
210   1                  IF (INKEY:=CI)=CTRLB THEN DO;
212   2                    CALL COSTR(.(CR,LF,'CONTROL = &'));
213   2                    IF((FLAG:=FLAG XOR CTLFLG)AND CTLFLG)<>0 THEN CALL COSTR(.('ON&'));
215   2                    ELSE CALL COSTR(.('OFF&'));
216   2                    CALL CRLF;
217   2                  END;
218   1                  ELSE IF (FLAG AND CTLFLG)<>0 THEN DO;
220   2                  IF INKEY=HOME THEN DO; /* Si es la tecla HOME se pone a la */
222   3                    OUTPUT (USTCMD)=RTS OR SBRK OR RE OR DTR OR TE;/*USART en BREAK */
223   3                    CALL DEL1;
224   3                    OUTPUT(USTCMD)=ER OR RTS OR RE OR DTR OR TE;
225   3                    CALL CEDISK;
226   3                  END;
227   2                  ELSE IF INKEY=CTRLI THEN FLAG=FLAG OR 01H;/* activa flag impresora */
229   2                ELSE IF INKEY=CTRLK THEN DO;/* se comprueba si esta activado el flag */
231   3                  IF (FLAG AND 04H)=0 THEN DO;/* de disco. Si no lo esta se activa y */
233   4                    FLAG=FLAG OR 04H; /* abre el nuevo fichero escritura. */
234   4                    IF NOT ABFICH(.(0DH,0AH,'NOMBRE DEL FICHERO A CREAR = &'),2) THEN
235   4                      GOTO BUCPRI;
236   4                  END;
237   3                END;
238   2                ELSE IF INKEY=ESC THEN CALL EXIT; /* Retorna al ISIS */
                   /* Si es CTL A se envia un fichero desde el Sist.Des. al ordenador */
240   2                ELSE IF INKEY=CTRLA THEN DO;/*Enviar fich. del Sist.Des. a ordenador */
                    /* Se abre el fichero de solo lectura */
242   3                  IF NOT ABFICH (.(0DH,0AH,'NOMBRE DEL FICHERO A LEER = &'),1) THEN
243   3                    GOTO BUCPRI;
244   3                  CALL SO (CR); /* Se manda un <CR> automatico a la USART */
245   3                  CALL DEL1; /* Volcar en el buffer el fichero */
246   3        LECFIC:   CALL READ (RAFT,.BUFFER,TAMBUF,.CONLEC,.STATUS);
247   3                  IF STATUS<>0 THEN DO;
249   4                    CALL ERROR (STATUS);
250   4                    CALL EXIT;
251   4                  END;
252   3                  PBUFER=0;
253   3        LAZPRI:   IF CSTS<>0 THEN  /* Si no se ha pulsado una tecla ir a LECDAT */
254   3        SDT:        IF (INKEY:=CI)=HOME THEN DO; /* Si es HOME entonces: */
256   4                      OUTPUT(USTCMD)=RTS OR SBRK OR RE OR DTR OR TE;/* USART en BREAK*/
257   4                      CALL DEL1;
258   4                      OUTPUT (USTCMD)=ER OR RTS OR RE OR DTR OR TE;
259   4                      CALL CEDISK; /* Cierra el fichero abierto */
260   4                      GOTO INDBUF; /* Vuelve a inicializar el puntero del buffer */
261   4                    END;
262   3                    ELSE IF INKEY=ESC THEN DO; /* Si la tecla pulsada es ESCAPE */
264   4                      CALL CEDISK; /* entonces: Cierra el fichero abierto */
265   4                      CALL EXIT; /* Vuelve al ISIS */
266   4                    END;
```

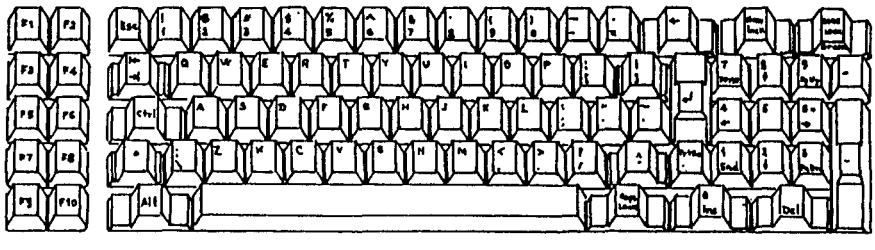
```
                        IF (INPUT(USTSTS) AND 02H)<>0 THEN DO;
                    /* Si el ordenador ha contestado entonces: */
269   4       LECDAT:     IF ((DATIN:=INPUT(0F6H)) AND 7FH)=ENQ THEN CALL SO(ACK);
                    /* Si es ENQUIRY, se responde con ACKNOWLEDGE */
271   4                   ELSE IF DATIN=CTRLS THEN DO;
273   5                     FLAG=FLAG OR CSQFLG;
274   5                     CALL COSTR(.('WAIT',1FH,1FH,1FH,1FH,'&'));
275   5                   END;/* Si no esta el transmisor en pausa entonces transmitir */
276   4                   ELSE IF DATIN=CTRLQ THEN FLAG=FLAG XOR NOT CSQFLG;
278   4                   ELSE CALL CO(DATIN);
279   4                 END; /* Si no ha contestado entonces :*/
280   3               ELSE IF(INPUT(USTSTS)AND 05H XOR 05H)=0 AND(FLAG AND CSQFLG)=0 THEN DO;
                    /* Si la USART no esta preparada ir a LAZPRI */
282   4                 IF CONLEC=0 THEN DO;/* Si el num. de bytes leidos es cero: */
284   5                   CALL CEDISK; /* se cierra el fichero */
285   5                   CALL SO(MFF);/* envia al ordenador marca de fin de fich.=CTL Y */
286   5                   GOTO INDBUF;/* inicializar el buffer y al bucle principal */
287   5                 END;
288   4                 CALL SO(BUFFER(PBUFER));/* envia un caract. del buffer por USART */
289   4                 IF BUFFER(PBUFER)=LF THEN CALL DEL1;/*Si es un <LF> llama a DEL1*/
291   4                 IF CONLEC-1=PBUFER THEN DO; /* Si PBUFFER = CONLEC entonces: */
293   5                   CALL DEL1;/* retraso para volver a leer el siguiente bloque */
294   5                   GOTO LECFIC; /* Se incrementa el puntero del buffer */
295   5                 END;
296   4       SIGCAR:     PBUFER=PBUFER+1;
297   4                 END;
                        GOTO LAZPRI;
299   3               END;
300   2               ELSE IF INKEY = CTRLC THEN DO;/* Cerrar fichero */
302   3                 FLAG=FLAG OR EOBFLG;/* Set bit ultimo bloque */
303   3                 CALL IMPRIM;
304   3                 CALL COSTR(.(CR,LF,'WARNING: Cerrado el fichero',CR,LF,'&'));
305   3               END;/* Substituir su codigo por backspace */
306   2               ELSE IF INKEY=RBOUT THEN CALL SO(BS); /* Si es RBOUT */
308   2               ELSE IF INKEY=CTRLD THEN DO;/* reasignar codigo fin fichero */
310   3                 CALL COSTR(.(CR,LF,BEL,'Codigo fin fichero=&'));
311   3                 MFF=CI;
312   3                 CALL COSTR(.(CR,LF,BEL,'WARNING: Codigo fin de fichero = &'));
313   3                 CALL COBYTE(MFF);
314   3                 CALL CRLF;
315   3               END;
316   2               ELSE IF (FLAG AND CSQFLG)=0 THEN CALL SO(INKEY);
                      END;
319   1               ELSE IF (FLAG AND CSQFLG)=0 THEN CALL SO (INKEY);
                        /* Si el ordenador no ha contestado se salta a BUCPRI */
                  RESPUE: IF (INPUT(USTSTS) AND RRDY)=0 THEN GOTO BUCPRI;
                      /* Si es ENQUIRY responde ACKNOWLEDGE */
323   1           SPUSAR: IF (DATIN:=(INPUT(0F6H)AND 7FH))=ENQ THEN CALL SO (ACK);
325   1             ELSE DO;/* Se saca el dato por la pantalla */
326   2               IF DATIN=CTRLS THEN DO;
328   3                 FLAG=FLAG OR CSQFLG;
329   3                 CALL COSTR(.('WAIT',1FH,1FH,1FH,1FH,'&'));
330   3               END;
331   2               ELSE IF DATIN=CTRLQ THEN FLAG=FLAG AND NOT CSQFLG;
333   2               ELSE CALL CO(DATIN); /* Si no hay ningun flag activado se vuelve al */
                        /* bucle principal. Si no, se mete el dato en el buffer */
334   2               IF (FLAG AND 05H)<>0 THEN DO;
336   3                   BUFFER(PBUFER)=DATIN;/*Volver a empezar si el buffer no esta lleno */
337   3                   IF (PBUFER:=PBUFER +1)<TAMBUF-1 THEN GOTO BUCPRI;
339   3           SALTQ:  CALL IMPRIM;/*Sacarlo por periferico si cabe */
340   3                   GOTO INDBUF;
341   3                 END;
342   2             END;
343   1             GOTO BUCPRI;
344   1             END CONEX;


MODULE INFORMATION:

     CODE AREA SIZE     = 08FAH   2298D
     VARIABLE AREA SIZE = 0430H   1072D
     MAXIMUM STACK SIZE = 000AH     10D
     378 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
 ISIS-II OBJECT LINKER V3.0 INVOKED BY:
-LINK QILER1.OBJ, SYSTEM.LIB, PLM80.LIB TO QILER1.LNK MAP PRINT(QILER1.MLK)

LINK MAP OF MODULE QILER1
WRITTEN TO FILE :F0:QILER1.LNK
MODULE IS A MAIN MODULE

SEGMENT INFORMATION:
START   STOP LENGTH REL NAME

                9BFH  B  CODE
                458H  B  DATA
                 1EH  B  STACK

INPUT MODULES INCLUDED:
 :F0:QILER1.OBJ(CONEX)
 :F0:SYSTEM.LIB(CI)
 :F0:SYSTEM.LIB(CLOSE)
 :F0:SYSTEM.LIB(CO)
 :F0:SYSTEM.LIB(CSTS)
 :F0:SYSTEM.LIB(EXIT)
 :F0:SYSTEM.LIB(IOSET)
 :F0:SYSTEM.LIB(LO)
 :F0:SYSTEM.LIB(OPEN)
 :F0:SYSTEM.LIB(READ)
 :F0:SYSTEM.LIB(WRITE)
 :F0:SYSTEM.LIB(ERROR)
 :F0:SYSTEM.LIB(ISIS)
 :F0:PLM80.LIB(@P0094)
 :F0:PLM80.LIB(@P0098)
 :F0:PLM80.LIB(@P0101)
 :F0:PLM80.LIB(@P0103)


 ISIS-II OBJECT LOCATER V3.0 INVOKED BY:
-LOCATE QILER1.LNK TO QILER1.LOC SYMBOLS CODE(4000H) PUBLICS COLUMNS(3) MAP PRINT(QILER1.MLC)

SYMBOL TABLE OF MODULE QILER1
READ FROM FILE :F0:QILER1.LNK
WRITTEN TO FILE :F0:QILER1.LOC

 VALUE TYPE SYMBOL            VALUE TYPE SYMBOL            VALUE TYPE SYMBOL

 F803H PUB  CI                F809H PUB  CO                F812H PUB  CSTS
 F818H PUB  IOSET             F80FH PUB  LO                0040H PUB  ISIS
 48FAH PUB  CLOSE             490DH PUB  EXIT              491CH PUB  OPEN
 4940H PUB  READ              4964H PUB  WRITE             4983H PUB  ERROR
 4991H PUB  @P0094            4994H PUB  @P0095            499BH PUB  @P0098
 499DH PUB  @P0099            49A0H PUB  @P0100            49A8H PUB  @P0101
 49ABH PUB  @P0102            49B3H PUB  @P0103            49B6H PUB  @P0104
       MOD  CONEX
 4E41H SYM  MEMORY            49E9H SYM  PBUFER            49EBH SYM  I
 49EDH SYM  WAFT              49EFH SYM  CAFT              49F1H SYM  RAFT
 49F3H SYM  OAFT              49F5H SYM  ACCESS            49F7H SYM  STATUS
 49F9H SYM  CONLEC            49FBH SYM  FLAG              49FCH SYM  DATIN
 49FDH SYM  INKEY             49FEH SYM  DATOUT            49FFH SYM  MFF
 4000H SYM  BAUDCNT           4A00H SYM  NFILE             4A10H SYM  BUFFER
 4532H SYM  SO                4E10H SYM  DATOUT            459AH SYM  HEXASC
 4E11H SYM  HEXBYTE           45B9H SYM  COSTR             4E12H SYM  DIREC
 45D4H SYM  CRLF              45E5H SYM  COBYTE            4E14H SYM  HEXBYTE
 4604H SYM  DEL0              4646H SYM  CEFICH            4678H SYM  IMPRIM
 4678H SYM  DIOGD             474BH SYM  CEDISK            4757H SYM  DEL1
 4769H SYM  ABFICH            4E15H SYM  CADADD            4E17H SYM  ACCESS
 47C0H SYM  ABFIC2            488CH SYM  SALT1             41FAH SYM  PRINCIP
 4283H SYM  INDBUF            4289H SYM  BUCPRI            4357H SYM  LECFIC
 4389H SYM  LAZPRI            4391H SYM  SDT               43C7H SYM  LECDAT
 4476H SYM  SIGCAR            44FBH SYM  RESPUE            4507H SYM  SPUSAR
 4577H SYM  SALT0

 MEMORY MAP OF MODULE QILER1
 READ FROM FILE :F0:QILER1.LNK
 WRITTEN TO FILE :F0:QILER1.LOC
 MODULE START ADDRESS 41F7H

 START   STOP LENGTH REL NAME

 4000H  49BEH   9BFH  B  CODE
 49BFH  49E8H    2AH  B  STACK
 49E9H  4E40H   458H  B  DATA
 4E41H  F68FH  A87FH  B  MEMORY
```

CAPITULO 4:

Programa ASCBIN.BAS

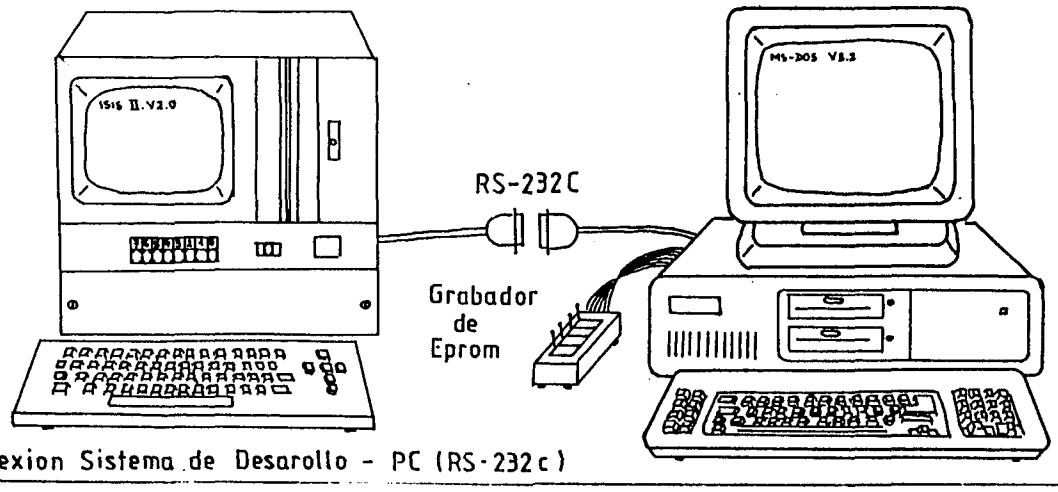
(Conversion de ficheros hexadecimales a binarios.)



Teclado del IBM/PC

PROGRAMA ASCBIN.BAS.- (Detección de errores y conversión de fichero hexadecimal a binario).

El programa ASCBIN.BAS fué escrito en GWBASIC en un WISDOM PC con monitor en color para poder grabar el programa Monitor del SDC-85 en tres EPROMs de 2Kbx8 (2716).



Conexion Sistema de Desarollo - PC (RS-232c)

En realidad, el programa se encarga de recibir caracteres hexadecimales enviados por el Sistema de Desarrollo, comprobar si ha habido errores en transmisión y pasar el fichero a formato binario.

Al principio del programa se pregunta:

BUFTOP=_

que es el tamaño del buffer interno del programa y que debido a que se inicializa con ØØH, cuanto mayor sea este número, más tardará en salir el siguiente mensaje:

COMANDO=_

que da pié a introducir comandos de ejecución. Por tanto, una longitud aceptable es 1/2 kbyte = 512. Esto significa que a efectos de memorización el PC sólo dispone de 512 bytes.

Comandos del programa ASCBIN.BAS:

- Comando T: transmisión / recepción.

Este comando configura al PC como terminal inicializando el canal serie 2 con 8 bits/palabra, transmisión asíncrona, sin paridad y 2 bits de stop. Después de pulsar este comando aparece en pantalla el mensaje:

VELOCIDAD=_

En este momento se puede introducir la velocidad que se desee: 11Ø, 3ØØ, 6ØØ, 12ØØ, 24ØØ, 48ØØ y 96ØØ.

Para activar ficheros, pulsar CTRL - E. Aparece entonces el mensaje: ARCHIVO=_

Una vez introducido el nombre del archivo a enviar ó recibir, aparece el mensaje:

(T)ransmite ó (R)ecive ?

Si se pulsa "T", se lee el fichero del PC y se envía por el canal serie. Si se pulsa "R", se escribe lo que se recibe por el canal en el fichero. Se tiene por caracter fin de fichero el CTRL-Z, que cierra el fichero automáticamente. También lo cierra pulsando CTRL-E de nuevo.

- Comando A: conversión fichero hexadecimal a fichero binario.

formato: A

Al ejecutar este comando aparecen los siguientes mensajes: FICHERO ASC-1 ?

FICHERO ASC-2 ?

Después de introducir los nombres del fichero ASCII y del fichero binario a escribir, empiezan a aparecer en pantalla, las posiciones y valores del disco.

Al final se presentan el número de errores ó caracteres no hexadecimales encontrados en el fichero.

- Comando a: conversión de buffer hexadecimal a fichero binario.

formato: a direcc. inicial, direcc. final

ejemplo: aØ,FF

Al ejecutar este comando se presenta el mensaje:

FICHERO ASCII=_

Después de introducir el nombre del fichero binario a escribir, se convierte el bloque hexadecimal del buffer desde la dirección inicial hasta la dirección final y se envía al fichero.

- Comando C: comparar ficheros.

formato: C

Al ejecutar este comando aparecen en pantalla los mensajes:

FICHERO 1 A COMPARAR = _

FICHERO 2 A COMPARAR = _

Una vez introducidos los nombres de los ficheros hexadecimales, aparecen las posiciones reales dentro del disco. Al final se indican el número de errores y la longitud del fichero de menor longitud.

- Comando D: presentar posiciones de buffer de memoria.

formato: D direcc. inicial, direcc. final.

ejemplo: DØ,FF

- Comando L: cargar fichero en buffer de memoria.

formato: L direcc. inicial, direcc. final

ejemplo: LØ,FF

Al ejecutar este comando, aparece el mensaje:

FICHERO A CARGAR = _

Una vez introducido el nombre del fichero a cargar, lee el primer caracter y lo guarda en la dirección inicial, a-

símismo los siguientes hasta el byte de dirección final. Si el fichero es menor que el bloque de memoria, se carga entonces todo el fichero. Si nó, carga sólo los primeros bytes.

- Comando B: comparación y binarización de dos ficheros h.

  formato: B

  Al ejecutar este comando aparecen los mensajes:

  FICHERO 1 A COMPARAR = _

  FICHERO 2 A COMPARAR = _

  FICHERO BINARIO A GUARDAR = _

  que son los nombres del fichero hexadecimal enviado 2 veces, y el nombre del fichero binario resultado de la conversión. Al final se presentan los números de:

  X / X ERRORES / CARACTERES ESCRITOS

  X / X ERRORES EN FICHEROS A/B HEXADECIMALES.

- Comando 1: limpiar fichero hexadecimal.

  formato: 1

  Al ejecutarlo aparecen los mensajes:

  FICHERO ASCII SUCIO = _

  FICHERO ASCII LIMPIO = _

  Una vez introducidos los nombres, se guarda en el fichero limpio los caraçteres hexadecimales del fichero sucio (Ø-9, A-F), el resto es considerado error y se muestran al final de la ejecución del comando.

- Comando M: mover memoria.

  formato: M direcc. inicial, direcc. final, cuenta

  ejemplo: MØ,FF,1Ø

  direcc. inicial: posición del primer byte del bloque a mover.

direcc. final: posición final del primer byte del bloque a mover.

cuenta: número de bytes a mover.

- Comando S: substituir posiciones de memoria.

formato: S dirección inicial, ...

ejemplo: SØØ,ØØ-Ø1,ØØ-Ø2 <return>

direcc. inicial: dirección del primer byte a substituir. Para no variar el byte actual, pulsar ",".

- Comando s: guardar memoria en disco (save).

formato: s direcc. inicial, direcc. final

ejemplo: sØØ,FF

direcc. inicial: 1ª dirección del buffer a guardar en disco.

direcc. final: última dirección del buffer a guardar en disco.

RECEPCION Y CONVERSION DE UN FICHERO HEXADECIMAL.-

A continuación se muestra una forma de recibir un fichero hexadecimal 2 veces y convertirlo a fichero objeto, listo para ser grabado en EPROMs.

```
>MODE COM2:96ØØ,N,8,2
>COPY COM2: MONITE.HØ          (se envía el fichero)
>COPY COM2: MONITE.H1          (se envía de nuevo)
>GWBASIC                       (llamada del intérprete)
 LOAD "ASCBIN"
 RUN
 BUFTOP=512                    (512 bytes para buffer)
 COMANDO=1
 FICHERO ASCII SUCIO = MONITE.HØ
 FICHERO ASCII LIMPIO = MONITE.HØØ
```

```
COMANDO=1
FICHERO ASCII SUCIO = MONITE.H1
FICHERO ASCII LIMPIO = MONITE.H1Ø
COMANDO=C
FICHERO 1 A COMPARAR = MONITE.HØØ
FICHERO 2 A COMPARAR = MONITE.H1Ø
COMANDO=A
FICHERO ASC-1 ?  MONITE.HØØ
FICHERO ASC-2 ?  MONITE.HØ1
```

Este procedimiento es evidentemente largo, por lo que en el último momento, un nuevo comando, el "B" (binariza-- ción) hace la comparación de dos ficheros hexadecimales y su conversión a binario. Indicando al final el número de errores. En caso de haber Ø/X errores/caracteres escritos, la conversión sería buena.

Así se podrían resumir los comandos anteriores:

```
COMANDO=B
FICHERO 1 A COMPARAR = MONITE.HØ
FICHERO 2 A COMPARAR = MONITE.H1
FICHERO BINARIO A GUARDAR = MONITE.BIN
Ø/5281 ERRORES/CARACTERES ESCRITOS.
```

```
10 CLS
20 ON ERR GOTO 2870
30 SCREEN 0,0:COLOR 7,0
35 REM ================================
36 REM  COMIENZO DEL PROGRAMA ASCBIN.
37 REM ================================
40 INPUT "BUFTOP=";BUFTOP
50 DIM D$(BUFTOP)
150 BUFPTRO=0:PRINT:PRINT"DIMENSIONANDO MEMORIA : ";
160 FOR N=0 TO INT((BUFTOP-1)/1024)
170   BUFPTR1=1024*(N+1):IF BUFPTR1>BUFTOP THEN BUFPTR1=BUFTOP
180   FOR M=BUFPTRO TO BUFPTR1
190     D$(M)=CHR$(0)
200   NEXT M
210   BUFPTRO=M:PRINT HEX$(N+1)+"K ";
220 NEXT N
250 LOCATE ,,1:PRINT:PRINT "COMANDO=";
260 I$=INKEY$: IF I$="" THEN 260
270 IF I$=CHR$(27) THEN END
280 PRINT I$;
300 IF I$<> "T" THEN 700
310 KEY OFF:CLS:CLOSE
320 FALSE=0:TRUE=NOT FALSE:CTLS$=CHR$(19):CTLQ$=CHR$(17):CTLE$=CHR$(5)
330 PRINT"PROGRAMA ASYNC. TTY"
340 LINE INPUT "VELOCIDAD=",V$
350 OPEN "COM2:"+V$+",N,8,2" AS #1
360 PAUSE=FALSE
370 I$=INKEY$: IF I$="" THEN 390
380 IF I$=CTLE$ THEN 440
385 IF I$<>CHR$(27) THEN PRINT #1,I$;ELSE 250
390 IF EOF(1) THEN 370
400 IF LOC(1)>128 THEN PRINT #1,CTLS$;:PAUSE=TRUE
410 PRINT INPUT$(LOC(1),#1);
420 IF (PAUSE AND EOF(1)) THEN PRINT #1,CTLQ$;:PAUSE=FALSE
430 GOTO 370
440 LINE INPUT"ARCHIVO=";DSKFIL$
450 LINE INPUT"(T)RANSMITE O (R)ECIBE: ";I$
460 IF I$<>"R" THEN 570
470 OPEN "O",#2,DSKFIL$
480 I$=INKEY$: IF I$="" THEN 500
490 IF I$=CTLE$ OR I$=CHR$(27) THEN 620
500 IF EOF(1) THEN 480
510 IF LOC(1)>128 THEN PAUSE=TRUE:PRINT#1,CTLS$;
520 I$=INPUT$(LOC(1),#1):PRINT#2,I$;:PRINT I$;
530 IF PAUSE AND EOF(1) THEN PRINT#1,CTLQ$;
540 GOTO 480
570 IF I$<>"T" THEN 450
575 OPEN "I",#2,DSKFIL$
580 WHILE NOT EOF(2) AND INKEY$<>CTLE$
590   I$=INPUT$(1,#2):PRINT#1,I$;
600 WEND
610 PRINT #1,CHR$(28);
620 CLOSE#2:PRINT:PRINT"WARNING: Cerrado fichero ";DSKFIL$
630 GOTO 370
700 IF I$<>"B" THEN 980
710 PRINT:INPUT "FICHERO 1 A COMPARAR=";F1$
720 INPUT "FICHERO 2 A COMPARAR=";F2$
730 INPUT "FICHERO BINARIO A GUARDAR = ";F3$
740 OPEN "I",#1,F1$
750 OPEN "I",#2,F2$
760 OPEN "O",#3,F3$
770 N=0:E=0:E1=0:E2=0:B$=""
780 WHILE NOT(EOF(1) OR EOF(2) OR INKEY$=CHR$(27))
790   GOSUB 2840:M$=""
800   GOSUB 2900:M$=I$
810   IF EOF(1) OR EOF(2) THEN 830
820   GOSUB 2900:M$=M$+I$
830   PRINT #3,CHR$(VAL("&h"+M$));
840   N=N+1
850 WEND
860 CLOSE #1,#2,#3
870 PRINT:PRINT E;"/";N;"= ERRORES / CARACTERES ESCRITOS."
880 PRINT E1;"/";E2;"= ERRORES EN A/B."
890 GOTO 250
```

```
975 REM ==============================
976 REM  COMANDO DISPLAY.
977 REM ==============================
980 IF I$<>"D"THEN 1130
990   GOSUB 2730:IF I$=CHR$(27) THEN 250
1020   GOSUB 2810
1040   WHILE DIR1 <= DIR AND INKEY$<>CHR$(27)
1060     GOSUB 2750
1100   WEND
1110   GOSUB 3650
1120 GOTO 250
1125 REM ==============================
1126 REM  COMANDO SALVAR MEMORIA.
1127 REM ==============================
1130 IF I$<>"s" THEN 1250
1140   GOSUB 2730: IF I$=CHR$(27) THEN 250
1160   INPUT "NOMBRE DEL FICHERO A GUARDAR = ";F1$
1170   OPEN "O",#1,F1$
1190   WHILE DIR1<= DIR
1200     PRINT #1,D$(DIR1);
1210     GOSUB 2750
1220   WEND
1230   CLOSE #1
1240 GOTO 250
1245 REM ==============================
1246 REM  COMANDO MOVER MEMORIA.
1247 REM ==============================
1250 IF I$<>"M" THEN 1350
1260   GOSUB 2720
1265   N=0:GOSUB 2810
1270   WHILE N<DIR AND INKEY$<>CHR$(27)
1275     D$(DIR1)=D$(DIR2)
1280     GOSUB 2750
1290     DIR2=DIR2+1:N=N+1
1310   WEND
1340 GOTO 250
1350 IF I$<>"S" THEN 1442
1360   GOSUB 2590: IF I$<>"," THEN 250
1370   DIR1 = DIR
1390   WHILE I$=","
1400     PRINT RIGHT$("0"+HEX$(ASC(D$(DIR1))),2);"-";
1410     GOSUB 2710:IF I$<>CHR$(27) AND N<>0 THEN D$(DIR1)=CHR$(DIR)
1420     DIR1=DIR1+1:IF DIR1=8*INT(DIR1/8) THEN PRINT:PRINT RIGHT$("000"+HEX$(DI
R1),4);": ";
1430   WEND
1435 GOTO 250
1439 REM ==============================
1440 REM  COMANDO FILE ASC A FILE BIN
1441 REM ==============================
1442 IF I$<>"A" THEN 1510
1446   PRINT:INPUT"FICHERO ASC-1";F1$:IF F1$=CHR$(34)+CHR$(27)+CHR$(34) THEN 250
1447   INPUT"FICHERO ASC-2";F2$:IF F2$=CHR$(34)+CHR$(27)+CHR$(34) THEN 250
1448   OPEN "I",#1,F1$:OPEN "O",#2,F2$
1450   IF DIR1>0 THEN I$=INPUT$(DIR1,#1)
1452   E=0:N=0:B$=""
1453   WHILE (INKEY$<>CHR$(27)) AND NOT EOF(1)
1455     I$=INPUT$(1,#1): GOSUB 2490
1460     IF F=0 THEN E=E+1:PRINT CHR$(7);: GOTO 1490
1465     C=B:IF NOT EOF(1) THEN I$=INPUT$(1,#1) ELSE 1490
1470     GOSUB 2490:IF F=0 THEN E=E+1:PRINT CHR$(7); ELSE C=C*16+B
1480     GOSUB 2840:PRINT #2,CHR$(C);:PRINT RIGHT$("0"+HEX$(C),2);
1482     N=N+1
1490   WEND
1495   CLOSE#1,#2:GOSUB 3650 :PRINT E;" ERRORES."
1500 GOTO 250
```

```
1505 REM ============================
1506 REM  COMANDO MEMORIA ASC A FILE BIN.
1507 REM ============================
1510 IF I$<>"a" THEN 1660
1520   GOSUB 2730:IF I$=CHR$(27) THEN 250
1523 INPUT"FICHERO ASCII = ";F1$
1526 OPEN"O",#1,F1$
1530   E=0: N=0:B$="":O=0
1540   WHILE (O<DIR) AND (INKEY$<>CHR$(27))
1550     I$=D$(DIR1): GOSUB 2490
1560      IF F=0 THEN E=E+1: GOTO 1610
1570      C=B:DIR1=DIR1+1:O=O+1:I$=D$(DIR1):GOSUB 2490
1580      IF F=0 THEN E=E+1: ELSE C=C*16+B
1590      PRINT #1,CHR$(C);:GOSUB 2840
1600      PRINT " ";RIGHT$("0"+HEX$(C),2);:B$=B$+CHR$(C)
1610      DIR1=DIR1+1:N=N+1
1620   WEND
1630   CLOSE#1
1640   PRINT E;" ERRORES."
1650 GOTO 250
1655 REM ============================
1656 REM  COMANDO LOAD.
1657 REM ============================
1660 IF I$<>"L"THEN 1750
1670   GOSUB 2730:IF I$=CHR$(27) THEN 250
1680   INPUT "FICHERO A CARGAR = ";F1$
1690   OPEN "I",#1,F1$
1695   N=0:GOSUB 2810
1700   WHILE NOT EOF(1) AND N<=DIR
1710    D$(DIR1)=INPUT$(1,#1):GOSUB 2750:N=N+1:GOSUB 2790
1720   WEND
1725 GOSUB 3650
1730 CLOSE#1:PRINT N;" CARACTERES CARGADOS."
1740 GOTO 250
1745 REM ============================
1746 REM  COMANDO COMPARAR FICHEROS.
1747 REM ============================
1750 IF I$<>"C" THEN 1860
1755 PRINT
1760 INPUT"FICHERO 1 A COMPARAR = ";F1$
1770 INPUT"FICHERO 2 A COMPARAR = ";F2$
1780 OPEN "I",#1,F1$: OPEN "I",#2,F2$
1790 N=0:E=0:COLOR 7,0:PRINT
1800 WHILE NOT (EOF(1) OR EOF(2) OR INKEY$=CHR$(27))
1805 GOSUB 2840: I$=INPUT$(1,#1)
1810   IF I$=INPUT$(1,#2) THEN PRINT RIGHT$("0"+HEX$(ASC(I$)),2); ELSE E=E+1:COL
OR 23,0:PRINT RIGHT$("0"+HEX$(ASC(I$)),2);:COLOR 7,0
1820   N=N+1
1830 WEND
1840 PRINT: PRINT E;"/";N;"=ERRORES/CARACT. LEIDOS.":CLOSE #1,#2
1850 GOTO 250
1860 IF I$<>"1" THEN 250
1870 PRINT:INPUT "FICHERO ASCII SUCIO = ";F1$
1880 INPUT "FICHERO ASCII LIMPIO = ";F2$
1890 OPEN "I",#1,F1$:OPEN "O",#2,F2$
1900 N=0:E=0:B$=""
1910 WHILE NOT EOF(1) AND INKEY$<>CHR$(27)
1920   GOSUB 2840:I$=INPUT$(1,#1):GOSUB 2490:N=N+1
1930   IF F THEN PRINT RIGHT$("0"+HEX$(ASC(I$)),2);:PRINT #2,I$;:GOTO 1960
1940   COLOR 0,7:PRINT CHR$(7);RIGHT$("0"+HEX$(ASC(I$)),2);:COLOR 7,0:E=E+1
1960 WEND
1970 CLOSE #1,#2:GOSUB 3650:PRINT E;"/";N;"= ERRORES / CARACTERES LEIDOS."
1980 GOTO 250
1990 GOTO 250
2470 REM ============================
2475 REM  RUTINA CONVERSION ASC - BIN.
2480 REM ============================
2490 IF I$<"0" THEN F=0:RETURN
2500 IF I$<="9" THEN B=ASC(I$)-48:F=-1:RETURN
2510 IF I$<"A" THEN F=0:RETURN
2520 IF I$<="F" THEN B=ASC(I$)-55:F=-1:RETURN
2530 IF I$<"a" THEN F=0:RETURN
2540 IF I$<="f" THEN B=ASC(I$)-87:F=-1:RETURN
2550 F=0:RETURN
```

```
2560 REM ============================
2570 REM RUTINA LECTURA DIRECCION HEX.
2580 REM ============================
2590 GOSUB 2620:DIR=VAL("&h"+RIGHT$(DIR$,4)):IF DIR<0 THEN DIR=DIR+65536!
2600 IF DIR>BUFTOP THEN PRINT CHR$(7);
2610 RETURN
2620 DIR$="":I$=INKEY$:N=0
2630 WHILE I$<>"," AND I$<>CHR$(13) AND I$<>CHR$(27)
2650   GOSUB 2490:IF F THEN PRINT I$;:N=N+1:DIR$=DIR$+I$
2670   I$=INKEY$
2680 WEND
2690 PRINT I$;
2700 RETURN
2705 REM ================================
2706 REM  RUTINAS RECOGIDA DE DIRECCIONES
2707 REM ================================
2710 GOSUB 2620:DIR=VAL("&H"+RIGHT$(DIR$,2)):RETURN
2720 GOSUB 2590:IF I$<>CHR$(27) THEN DIR2=DIR ELSE RETURN
2730 GOSUB 2590:IF I$<>CHR$(27) THEN DIR1=DIR ELSE RETURN
2740 GOSUB 2590:RETURN
2750 IF DIR1=16*INT(DIR1/16) THEN GOSUB 3650:PRINT RIGHT$("000"+HEX$(DIR1),4);":
";:B$=""
2760 PRINT " ";RIGHT$("0"+HEX$(ASC(D$(DIR1))),2);
2770 B$ = B$ + D$(DIR1):DIR1= DIR1+ 1
2780 RETURN
2786 REM  PRESENTACION K BYTES
2790 IF DIR1=1024*INT(DIR1/1024) THEN PRINT DIR1/1024;"K ";
2800 RETURN
2810 B$="":IF DIR1<>16*INT(DIR1/16) THEN PRINT RIGHT$("000"+HEX$(DIR1),4);":";
2820 RETURN
2830 END
2836 REM PRESENTACION DIRECC. N
2840 IF N<>16*INT(N/16) THEN RETURN
2850 GOSUB 3650:PRINT RIGHT$("000"+HEX$(N),4);":";
2860 RETURN
2870 PRINT "ERROR";ERR;" EN LINEA ";ERL
2880 END
2900 I$=INPUT$(1,#1):GOSUB 2490: IF NOT F THEN IF NOT EOF(1) THEN E1=E1+1:GOTO 2
900 ELSE RETURN
2910 J$=I$
2920 I$=INPUT$(1,#2):GOSUB 2490:IF NOT F THEN IF NOT EOF(2) THEN E2=E2+1:GOTO 29
20 ELSE RETURN
2930 IF J$=I$ THEN PRINT I$;: RETURN
2940 E=E+1: COLOR 23,0:PRINT "?";:COLOR 7,0:GOTO 2900
3620 REM ================================
3630 REM  RUTINA PRESENTACION DE B$
3640 REM ================================
3650 PRINT TAB(56);
3660 FOR M=1 TO LEN (B$)
3670 D=ASC(MID$(B$,M,1))
3680 IF D<32 THEN COLOR 0,7:PRINT CHR$(D+64);:GOTO 3705
3690 IF D<127 THEN COLOR 7,0:PRINT CHR$(D);:GOTO 3710
3700 IF D<160 THEN COLOR 16,7:PRINT CHR$(D-64);ELSE COLOR 23,0:PRINT CHR$(D-128)
;
3705 COLOR 7,0
3710 NEXT M
3720 PRINT:B$=""
3730 RETURN
```

CAPITULO 5:

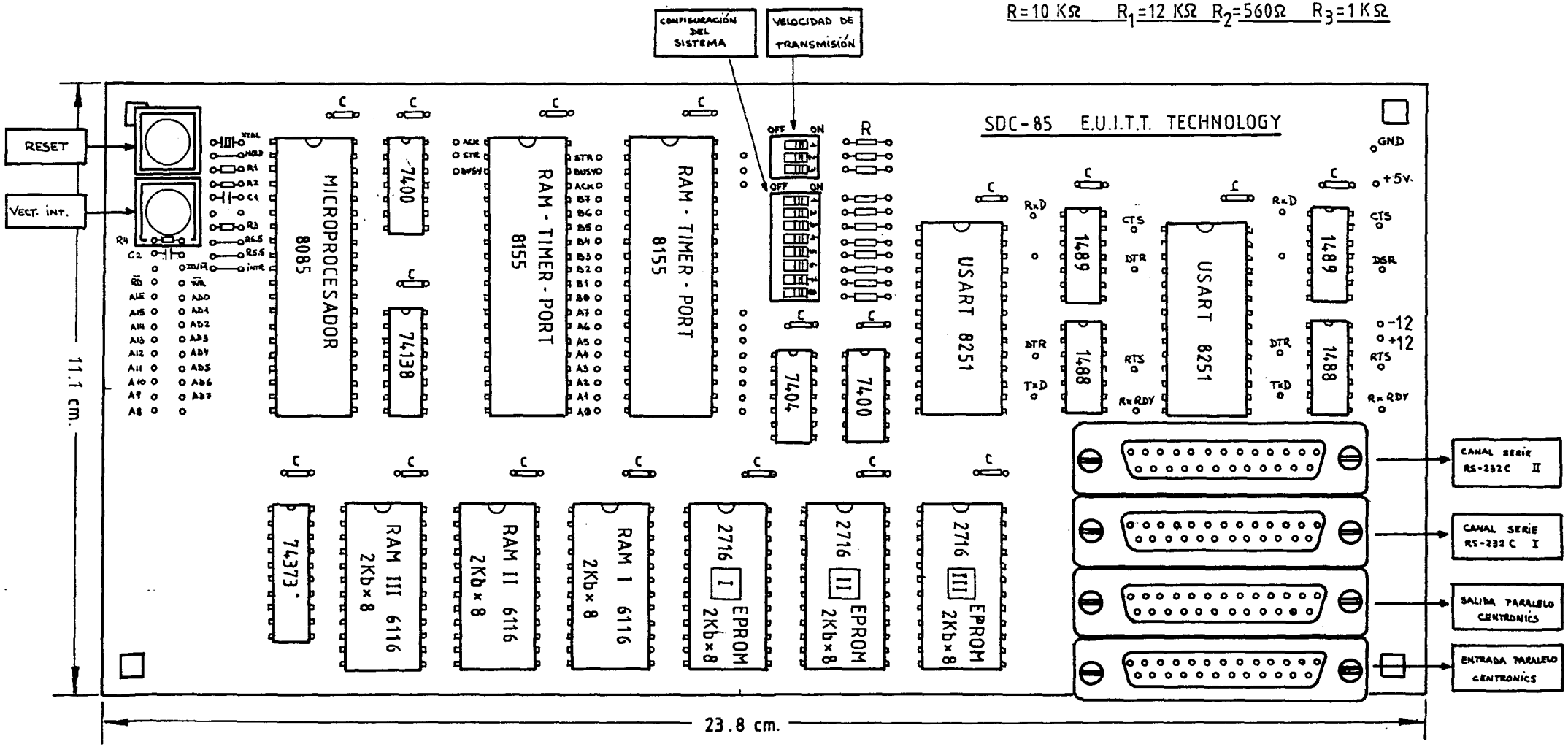
HARDWARE DEL SDC-85

SDC - 85

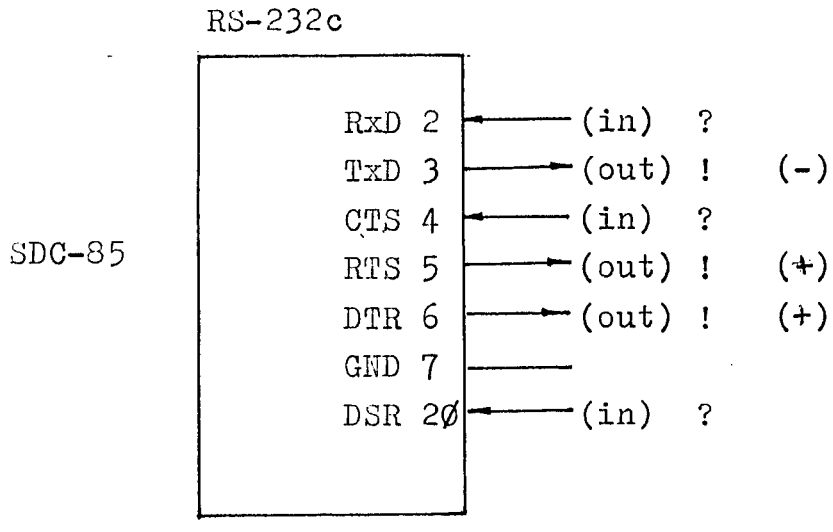
Vista superior. Tamaño real.

C=0.1 uF C_1=1uF C_2=1 uF R_4=220Ω

R=10 KΩ R_1=12 KΩ R_2=560Ω R_3=1 KΩ



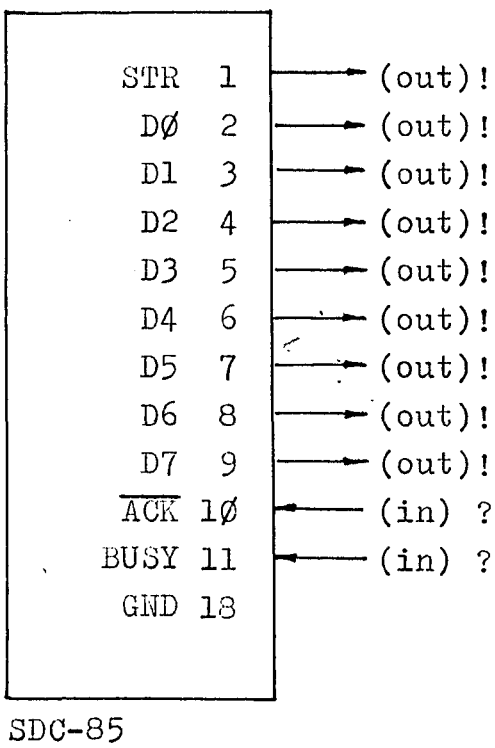
Configuración serie (canales 1 y 2) del SDC-85: (DB-25)



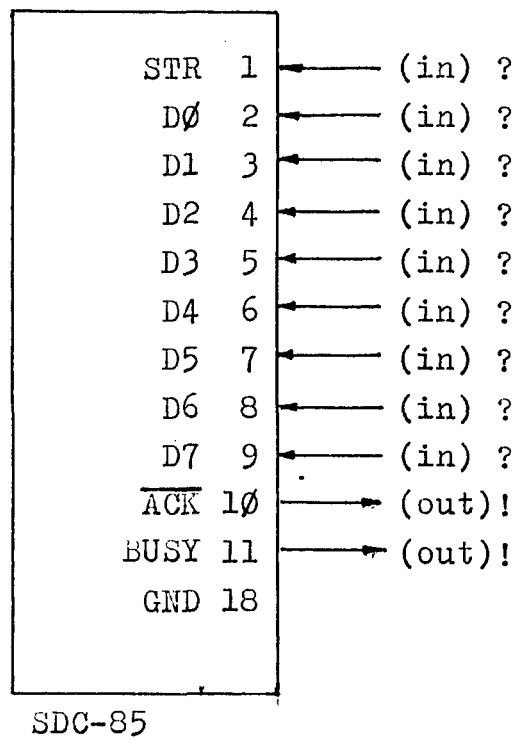
Configuración paralelo (entrada y salida) del SDC-85:

(DB-25)

salida Centronics



entrada Centronics



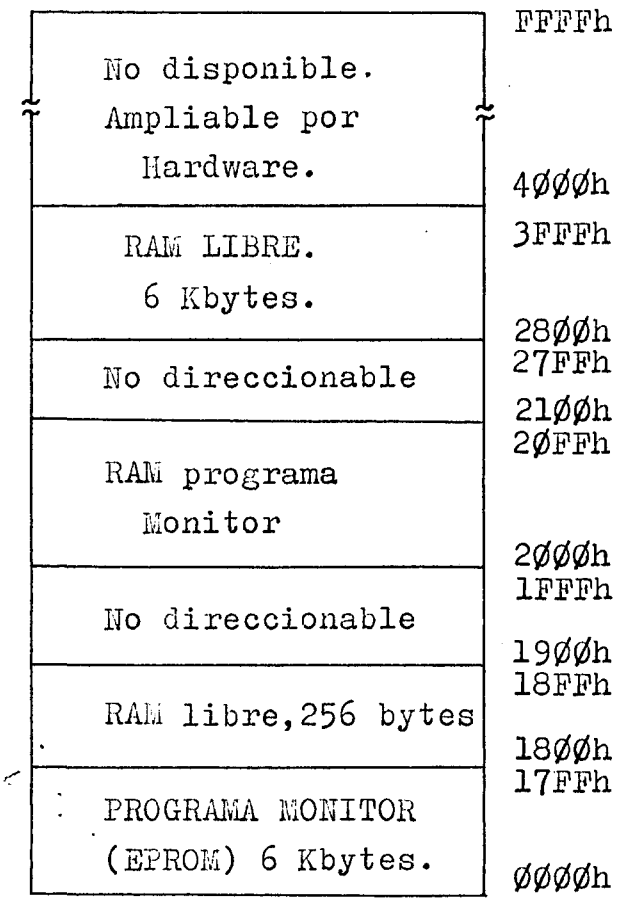
Descripción y funcionamiento del SDC-85:

Alimentación: el SDC-85 se alimenta con +5v. y GND para la circuitería TTL y +12v. y -12v. para la comunicación RS-232c.

Capacidad y direccionamiento de memoria.

De los 64 Kbytes que se pueden direccionar, el SDC-85 utiliza los primeros 16 Kbytes, de los cuales 6 Kb. son ROM (memoria de sólo lectura) y 6 1/2 Kb. son RAM (memoria lectura-escritura) y se distribuyen así:

Mapa de memoria del SDC-85:



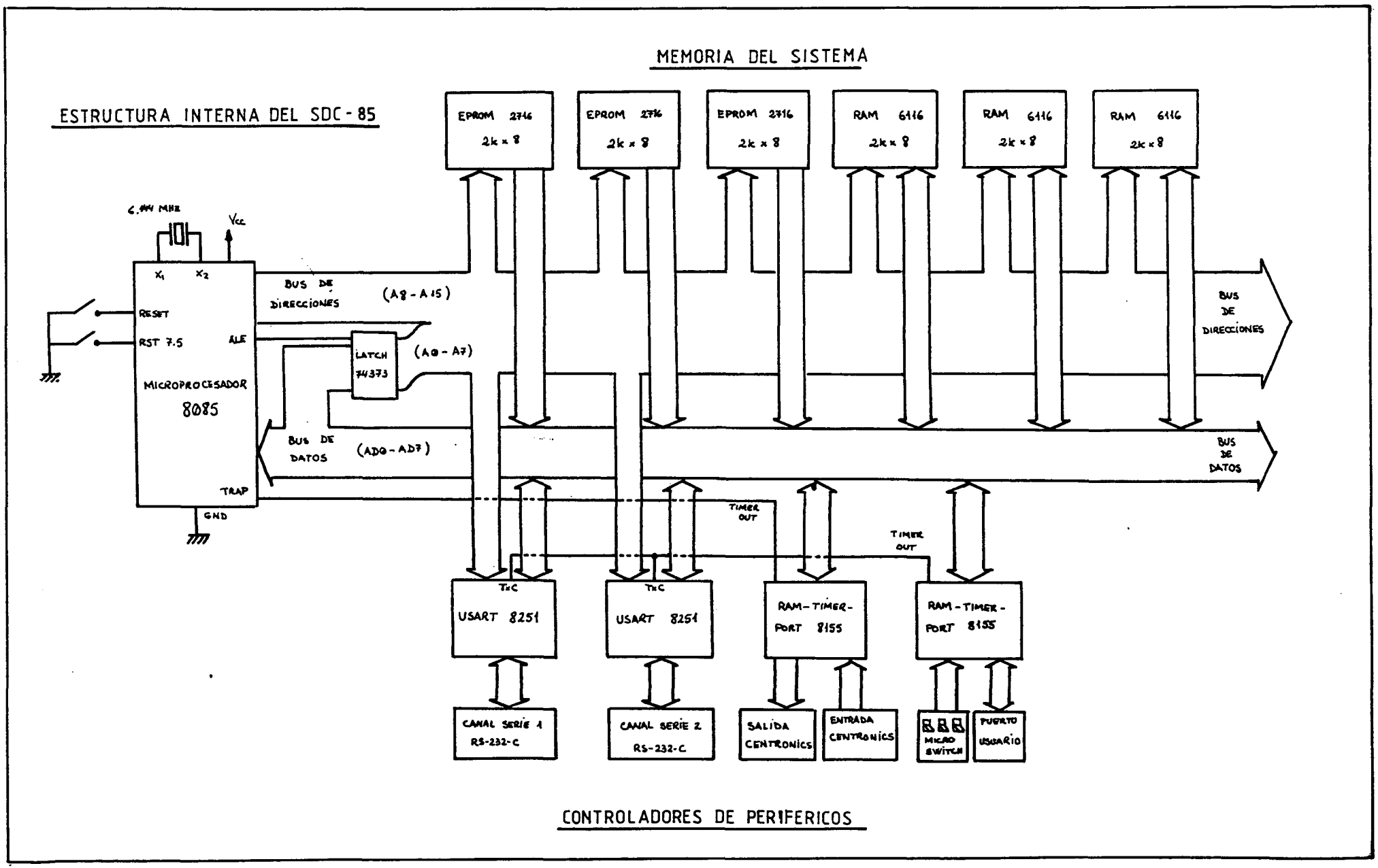
Bloque	Tipo de memoria
ØØØØh - Ø7FFh	2Kb. EPROM 2716
Ø8ØØh - ØFFFh	2Kb. EPROM 2716
1ØØØh - 17FFh	2Kb. EPROM 2716
18ØØh - 18FFh	1/2 Kb. RAM libre (8155)
2ØØØh - 2Ø5Ah	RAM libre
2Ø5Ah - 2Ø7Eh	Stack Monitor (24 b.)
2Ø7Fh - 2ØFFh	Data Monitor (144 b.)
28ØØh - 2FFFh	2 Kb. RAM libre 6116.
3ØØØh - 37FFh	2 Kb. RAM libre 6116.
38ØØh - 3FFFh	2 Kb. RAM libre 6116.
4000h - FFFFh	Sin asignar.

Direccionamiento de puertos.

Al igual que la memoria, de los FFh (256d.) posibles puertos, se direccionan sólo los 4Øh (64d.) primeros, los cuales se distribuyen así:

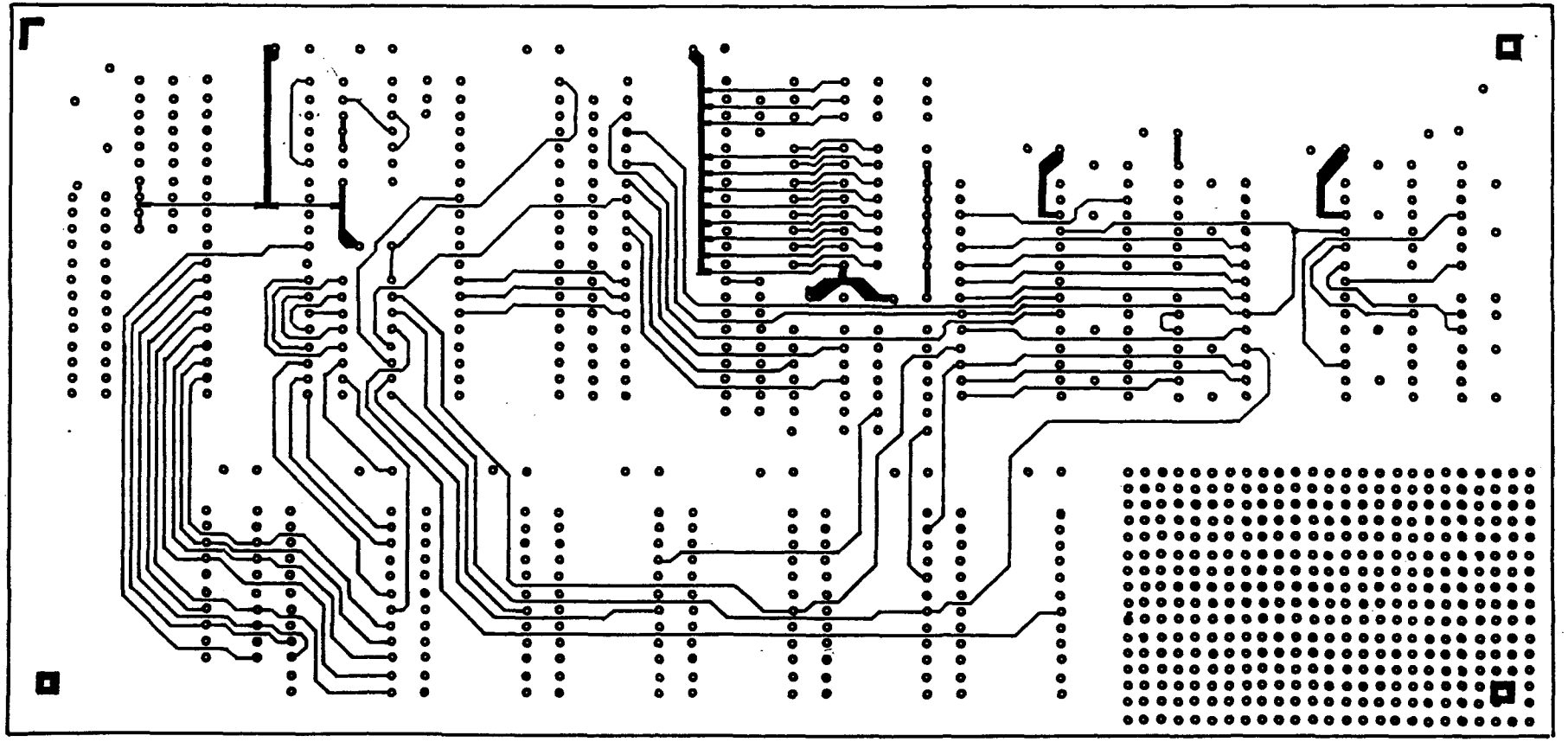
Puerto	
ØØh	Puerto de datos de la USART Ø.
Ø1h	Puerto de comando/status de la USART Ø.
Ø8h	Puerto de datos de la USART 1.
Ø9h	Puerto de comando/status de USART 1.
18h	Puerto de comando/status del 8155B.
19h	Puerto A libre, del 8155B.
1Ah	Puerto B de microswitch de configuración.
1Bh	Puerto C de microswitch de veloc. transm.
1Ch	Timer low, Reloj x 64 de la USART Ø - 1
1Dh	Timer high,
2Øh	Puerto de comando/status del 8155A.
21h	Puerto A :LP: entrada al 8155A.
22h	Puerto B :LP: salida del 8155A.
23h	Puerto C :LP: control del 8155A.
24h	Timer low, trap del uP.8Ø85.
25h	Timer high,
3Ø-FFh	sin asignar.

A continuación se muestra el diagrama de bloques de la estructura interna del SDC-85.

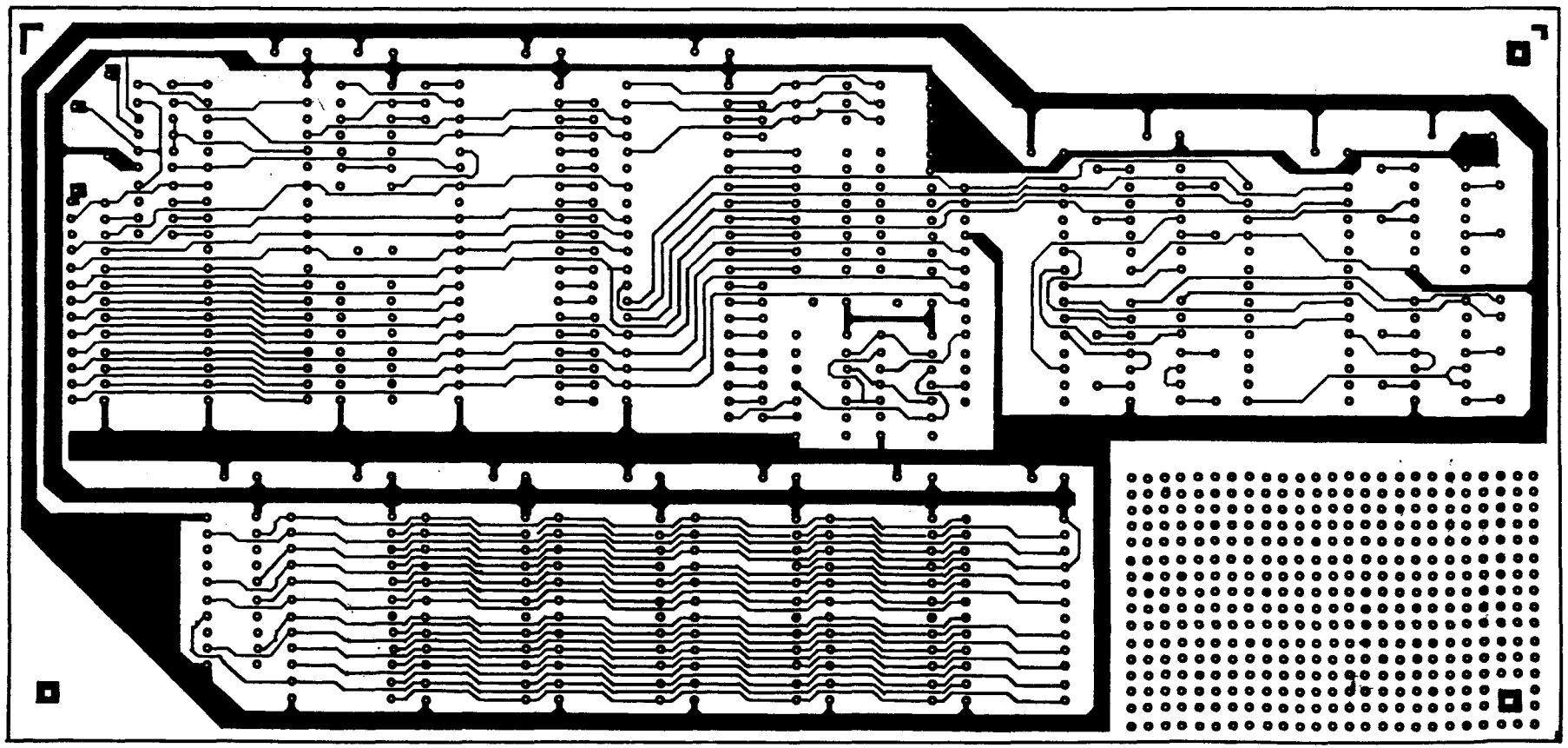


AUTOR: Roberto F. Benítez Díaz			
TITULO: Diagrama esquemático del SDC-85.		FECHA: 29/08/88	
TAMAÑO:	DEPT.	DIBUJO NUMERO:	REV. A

checkplot v1.0 r2 29 Jun 1988 12:31:07
file: b:sdc85.k0 upper layer
approx. size: 9.10 by 4.25 in. holes: 1035



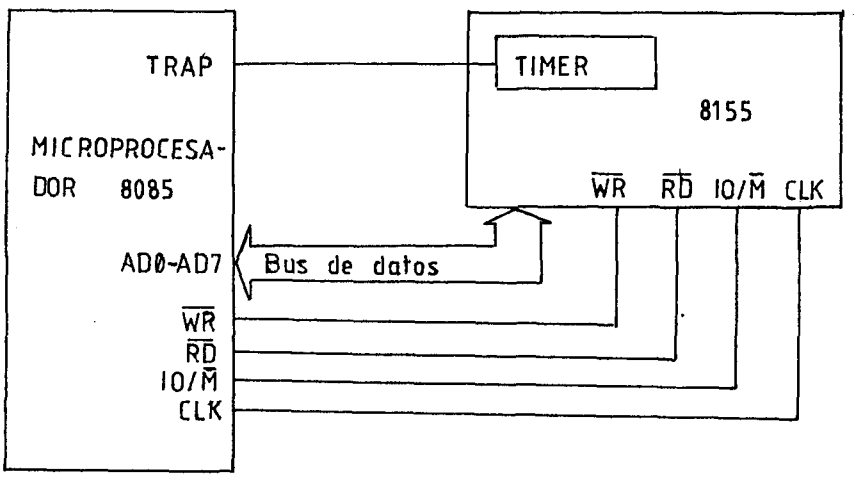
checkplot v1.0 r2 29 Jun 1988 12:26:40
file: b:sdc85.k0 lower layer
approx. size: 9.10 by 4.25 in. holes: 1035



Sistema de interrupciones del SDC-85.-

Interrupción.	Direc. salto	Prioridad.	Ejecución.	tipo de disparo
RESET	ØØh	1	Reset del sistema	nivel bajo
TRAP	24h	2	Paso a paso	sub. y alto
RST 7.5	3Ch	3	Reinicialización	flanco sub.
RST 6.5	34h	4	JMP 2ØE7	nivel alto
RST 5.5	2Ch	5	JMP 2ØE4	nivel alto
INTR	-	6	No utilizada	nivel alto

TRAP: es utilizada para la ejecución paso a paso del programa Monitor, a través de su conexión con el timer del 8155 .



RST 7.5: está conectada a un pulsador, el cual solicita la interrupción. En el programa Monitor se usa para intercambiar los valores :

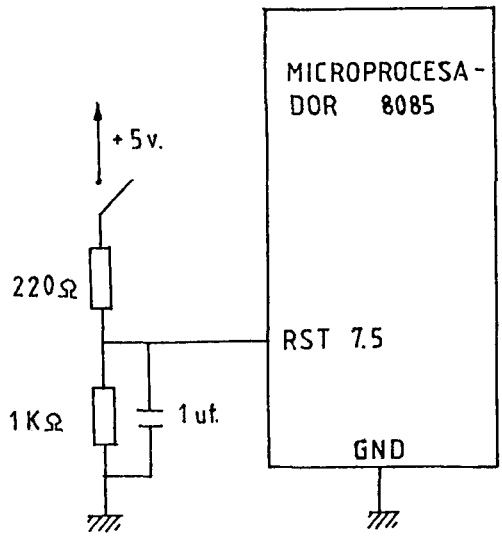
FLAG ⟷ CLDFLG
BUFLV ⟷ CLDBLV
BUFHV ⟷ CLDBHV
} Buffer de Spooler

Esto sirve para congelar momentáneamente la actividad del SDC-85. Para ello sería necesario haber reseteado el sistema pulsando:

RESET - RST 7.5 - RESET

De lo contrario el resultado sería impredecible al te-

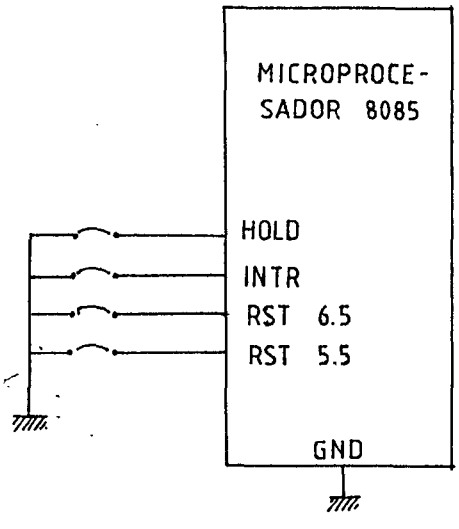
los nuevos valores, números aleatorios.



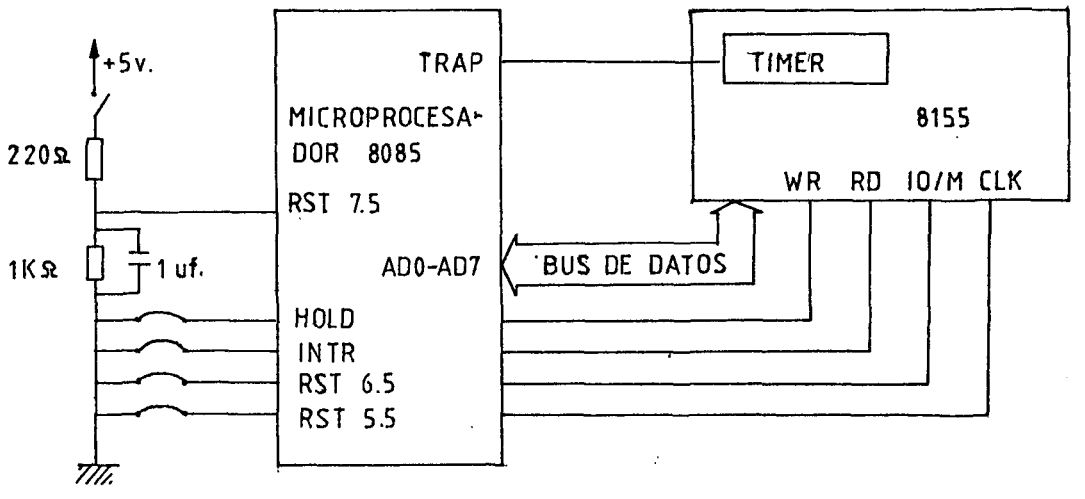
HOLD, INTR, RST 6.5, RST 5.5: están puenteados a Ø v. (inhabilitadas) y su uso está a disposición del usuario, colocando en memoria la instrucción y dirección de salto:

RST 6.5 2ØE7h - 2ØE9h

RST 5.5 2ØE4h - 2ØE6h



En resumen el aspecto de las interrupciones anteriores juntas es el siguiente:



Configuraciones RS-232c y Centronics de diversos terminales y periféricos.-

RS - 232 c

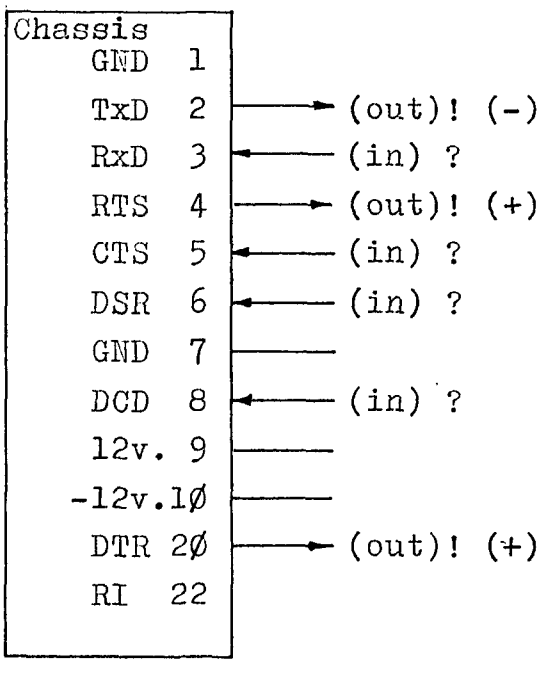
Sistema de desarrollo MDS-221 (DCE)

Pro.GND	1		
TxD	2	→	(out)! (-)
RxD	3	←	(in) ?
RTS	4	→	(out)! (+)
CTS	5	←	(in) ?
DSR	6	←	(in) ?
GND	7	—	
TxCLK	15	—	
RxCLK	17	—	
DTR	2Ø	→	(out)! (+)

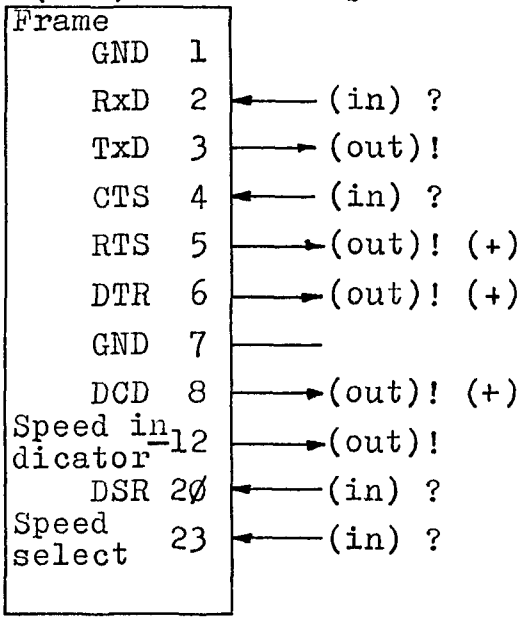
Terminal HP-2645 (DCE)

TxD	2		(out)! (-)
RxD	3	←	(in) ?
RTS	4	→	(out)! (+)
CTS	5	←	(in) ?
DSR	6	←	(in) ?
GND	7	—	
DCD	8	←	(in) ?
	11	→	(out)! (-)
DTR	2Ø	→	(out)! (+)

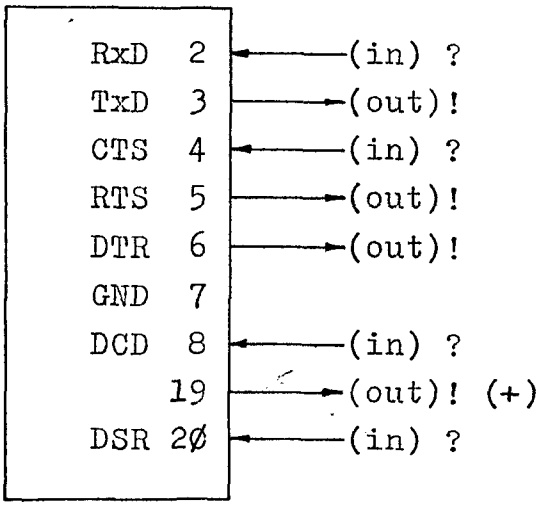
Wisdom PC/XT: (DCE)



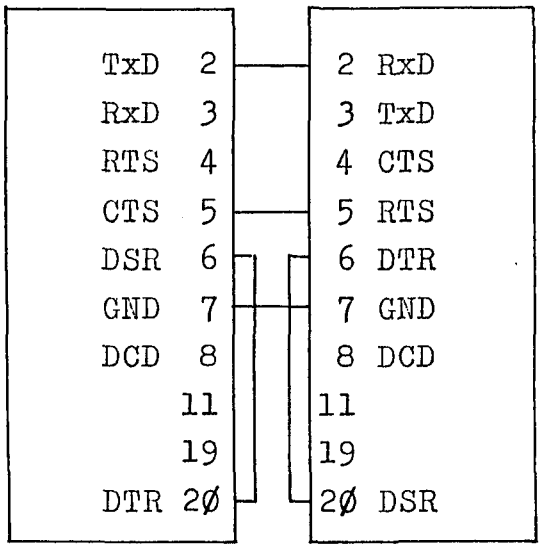
Terminal Televideo:
(DTE) - canal impresora -



Plotter DXY-1Ø1:
(DTE)

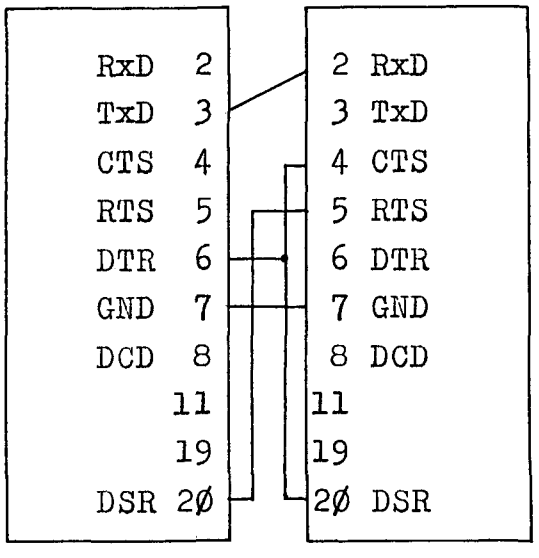


Televideo - Plotter DXY
(canal orden.)



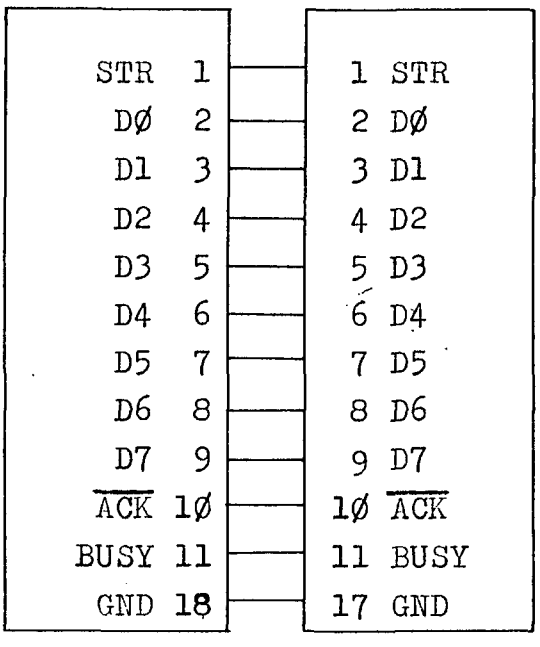
RS-232c

Televideo - Plotter DXY
(canal impresora)



Rs-232c

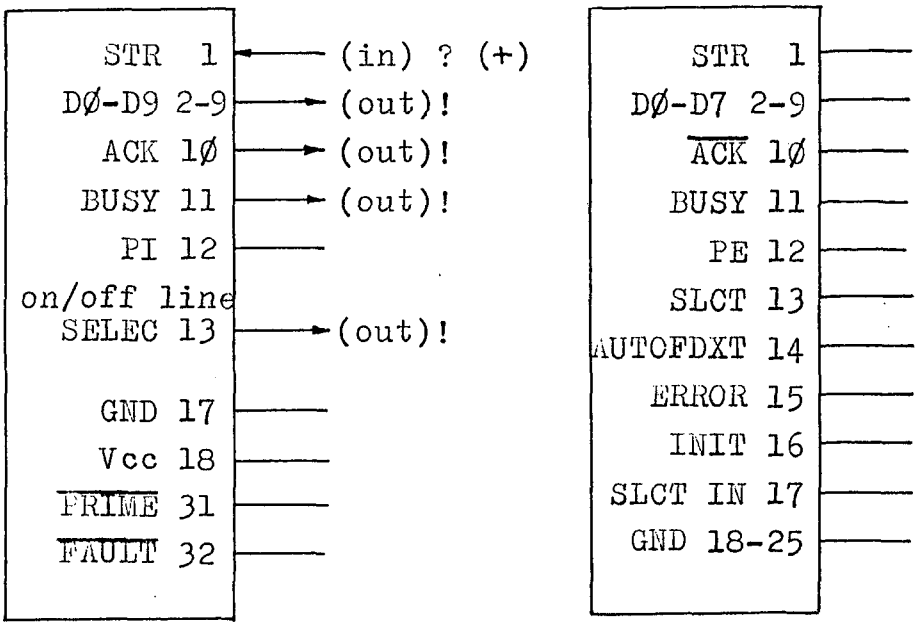
SDC-85 - Impresora TRS-8Ø



(DB-25) (Centronics)

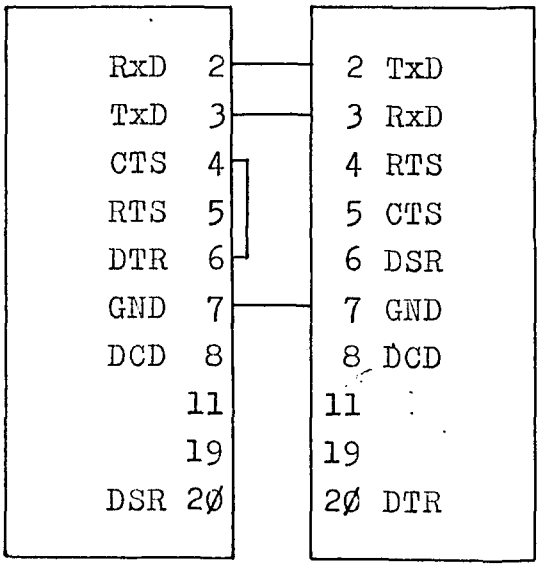
CENTRONICS

Impresora TRS-8Ø (entrada) — Ordenador Wisdom PC/XT (salida)



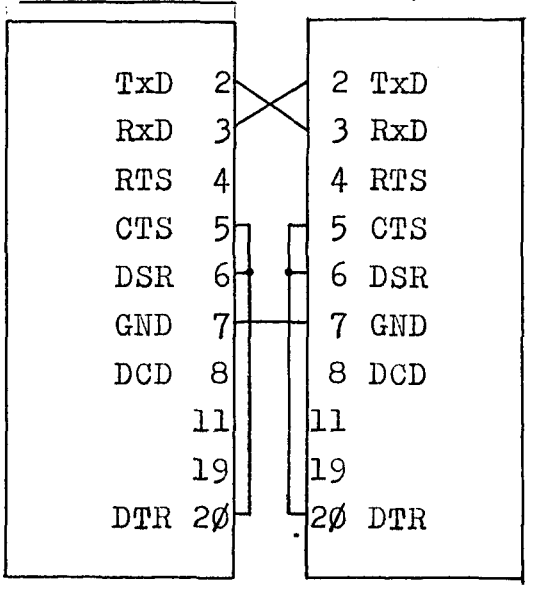
Conexiones mínimas entre terminales y periféricos.-

SDC-85 - Terminal HP



RS-232c

Sist.Desarr. - PC/XT



RS-232c

Componentes del SDC-85.-

A continuación se muestran los componentes electrónicos del montaje del SDC-85:

- Cantidad -	Componente	
1	8Ø85	Microprocesador
2	8251	USART
2	1488	Conversor de tensión
2	1489	Conversor de tensión
2	74ØØ	Puertas Nand.
1	74138	Decodificador
1	74Ø4	Puertas Not
3	2716	EPROM 2 Kbytes x 8.
2	6116	RAM 2Kbytes x 8.
4	DB-25	Conectores RS-232.
4	espadines	
12	separadores	
2	pulsadores	
1	uswitch - 8 microinterruptores	
1	uswitch - 3 microinterruptores	
15	resistencias 1/4 w.	
3	zócalos	40 pins.
2	"	28 "
6	"	24 "
7	"	14 "
1	Placa de circuito impreso.	
18	condensadores	0.1 uF.
2	condensadores	1 uF.

Esta lista está valorada actualmente en unas 19.ØØØ Ptas.

APENDICE A:

Sistema operativo ISIS II

SISTEMA OPERATIVO ISIS II.-

Este es el sistema operativo del Sistema de Desarrollo Intellec Series II: MDS-221 de Intel.

Formato de nombre de fichero / periférico:

:aparato:nombre del fichero.extensión

- :aparato: — 2 caracteres alfanuméricos preestablecidos
- nombre del fichero — 1-6 caracteres alfanuméricos.
- extensión — 1-3 caract. alfanum.

ejemplo: :FØ:MONITE.HEX

Nombres de periféricos del sistema.

:FØ: hasta :F9: unidades de disco (drive).
:TI: Teclado de teletipo (teletypewritter input).
:TO: Impresora de teletipo (teletypewritter output).
:TP: Perforadora de papel (teletypewritter punch).
:TR: Lectora de cinta de papel (teletypewritter reader).
:VI: Teclado del video terminal (video terminal input).
:VO: Pantalla del video terminal (video terminal output).
:HP: Perforadora a alta veloc. (high speed paper tape punch)
:HR: Lectora de alta veloc. (high speed paper tape reader)
:LP: Impresora paralelo (line printer).
:CI: Consola de entrada genérica (console input).
:CO: Consola de salida genérica (console output).
:BB: Aparato inexistente (byte bucket).

Edición de línea: 122 caracteres=longitud máxima.

RUBOUT: borra el caracter precedente.
Control-P: pulsada antes de un carácter de edición, permite su entrada en la línea de edición.
Control-R: presenta el contenido actual de la línea de edición.
Control-X: borra el contenido entero de la línea de edición. Presenta "#" y <CR>.
Control-Z: introduce fin de fichero al sistema. No cambia la pantalla. Borra la línea de edición.

Presentación de error:

ERROR nnn USER PC mmmm

- nnn — nº de error
- mmmm — número de contador de programa cuando ocurrió el error.

ERROR 24: STATUS=ØØnn nº de error.
D=x T=yyy S=zzz
dirección del sector
dirección de pista.
número de unidad de disco.

Organización del disco.-

Disco flexible: 77 pistas/disco.
Simple densidad: 26 bloques/pista.
Doble densidad: 52 bloques/pista. 128 bytes/bloque

Disco duro: 8ØØ pistas; 36 sectores/pista;
128 bytes/sector.

Contenido de directorio de disco:

2ØØ entradas (disco flexible)
992 entradas (disco duro).

* Nombre del fichero.
* Número de bloques por fichero.
* Número de bytes por fichero.
* Atributos.

Bloques: número de bloques dado el número de bytes =

B = INT (63 * (N/128)/62) +1
| └ número de bytes.
└ número de bloques.

Atributos:

Invisible: el fichero no es listado en directorio.
Write protected: el fichero está protegido contra escritura.
Format: es copiado en el nuevo disco al formatear con IDISK ó FORMAT.
System: es copiado en el nuevo disco al formatear con IDISK ó FORMAT especificando la opción "S".

Generación de nombres de ficheros:

*: equivale a cualquier cadena de caracteres.
Nombre.* : ficheros de un nombre y cualquier extensión.
*.extensión : ficheros de una extensión y cualquier nombre.
. : cualquier fichero.
ejemplo: AB*.HEX
? : equivale a un carácter.
ejemplo: A??.HEX

Comandos: sintaxis: <comando><parámetro>

Comandos de mantenimiento de disco:

IDISK - comando de formateo de disco.

sintaxis: IDISK <unidad><etiqueta>[<opciones>]

<unidad>: número de unidad de disco.

<etiqueta>: nombre designado al disco.

<opciones>: S = disco del sistema.
P = parada en sistemas de unidad simple.
FROM n = número de unidad con ficheros de formato.

ejemplo: IDISK :FØ:SYSTEM.PLM S
SYSTEM DISK
LOAD OUTPUT DISK, THEN TYPE (CR).
LOAD SYSTEM DISK, THEN TYPE (CR).

FORMAT - comando de formateo de disco.

sintaxis: FORMAT <unidad><etiqueta>[<opciones>]

<unidad>: número de unidad de disco

<etiqueta>: nombre designado al disco.

<opciones>: A = copia todos los ficheros en disco nuevo.
S = copia sólo los de atributo System.
FROM n = unidad con los ficheros necesarios en formateo.

ejemplo: FORMAT :FØ:JUGO.ØØØ

FIXMAP - mapa de sectores malos de disco duro.

sintaxis: FIXMAP <unidad>

<unidad>: nº de unidad de disco donde va a operar:
ISIS II MAP FIXER Vx.y

comandos:

(M) Mark <direcc. disco>: cambiar estado de sector de bueno a malo.

(F) Free <direcc. disco>: cambiar estado de sector de malo a bueno.

(L) List [<nombre de fichero>] : listar sectores malos.

(C) Count : lista número de sectores malos.

(R) Record: registra los cambios hechos por Mark y Free.

(Q) Quit: salir al ISIS II sin guardar cambios.

(E) Exit: registrar cambios y salir al ISIS II.

<direcc. disco>: <pista><sector>[T]
(Ø-199) (1 - 144) grupo de 36 sectores

Comandos de ejecución de programas.-

Nombre del fichero = directa ejecución del programa.

DEBUG - ejecución de programas bajo monitor.

sintaxis: DEBUG <nombre de programa>[<parámetros>]

<nombre de programa>: nombre del fichero ejecutable.

<parámetros>: parámetros normales del programa.

G8 : salir del Monitor al Sistema Operativo ISIS II.

ejemplo: DEBUG LIST FILE.TXT.

SUBMIT - ejecución no interactiva de programa.

sintaxis: SUBMIT <nombre>[.<extensión>][(<parámetro Ø>, ... ,<parámetro 9>)].

<nombre>[.<extensión>]: fichero conteniendo la secuencia de definición. Si se omite la extensión se supone ".CSD".

<parámetro n>: valor del parámetro formal.

%n : parámetros formales (Ø≤n≤9).

Control-P = caracter literalizante. Por ejemplo, delante de "%" no lo interpreta como parámetro formal.

Control-E = devuelve el control al teclado del Sistema de Desarrollo y viceversa.

ejemplo: PLM8Ø.CSD :

```
PLM8Ø :F1:%Ø.%1 DEBUG XREF PRINT (:F3:%Ø.LST) DATE %2
SUBMIT PLM8Ø(PROG1,SRC,'9 SEPT 78')
```

Comandos de control de ficheros.-

DIR - listado de directorio de disco.

sintaxis: DIR [FOR <fichero>][TO <fichero de listado>] [<opciones>]

<fichero>: fichero o grupo de ficheros (* ó ?) cuya entrada de directorio es listada.

<fichero de listado>: fichero que contendrá el listado.

<opciones>: Ø-9 unidad de disco (:FØ:,...,:F9:) a listar.

I: lista todos los ficheros, incluso los invisibles.

F : (fast). Lista sólo el nombre del fichero.

O : lista con formato de simple columna.

Z : lista sectores usados / disponibles.

P : parada para introducir disco.
LOAD SOURCE DISK, THEN TYPE (CR).
LOAD SYSTEM DISK, THEN TYPE (CR).

ejemplo: DIR

```
DIRECTORY OF name.ext
NAME  .EXT  BLKS  LENGTH ATTR  NAME  .EXT BLKS LENGTH ATTR
PROGA .HEX    75    9263  W     SUMS         51   6357
                        126
936/2002 blocks used.
ejemplo: DIR I FOR ISIS.* TO :LP:
```

COPY - copiar un fichero.

sintaxis: COPY <fichero entrada1>[,...,<fichero entrada n>] TO <fichero salida>[<opciones>]

<fichero entrada>: fichero de lectura a copiar.

<fichero salida>: fichero a ser creado ó recreado, ó periférico de salida. Si ya existe:

```
<fichero de salida> FILE ALREADY EXISTS.
DELETE ?
```

y si está protegido contra escritura :

```
<fichero de salida>, WRITE PROTECTED.
```

<opciones>: U (update). Evita el mensaje "ALREADY EXISTS", el fichero se reabre y se amplía en vez de borrarse.

S Copia sólo los de los atributos System.

N Copia los que no tengan atributo System ni Format.

P Pausa. Parada para cambiar de disco.
LOAD SOURCE DISK, THEN TYPE (CR).
LOAD OUTPUT DISK, THEN TYPE (CR).
LOAD SYSTEM DISK, THEN TYPE (CR).

Q (Query). Pregunta antes de copiar.
COPY <fichero entrada> TO <fichero salida> ?

C Asigna a <fichero salida> los atributos de <fichero de entrada>

B (Brief). Evita el mensaje "ALREADY EXISTS", borra el fichero anterior y lo recrea con los nuevos datos.

ejemplos:

```
COPY CHAP1,CHAP2,CHAP3 TO BOOK.
APPENDED :FØ:CHAP1 TO :FØ:BOOK
APPENDED :FØ:CHAP2 TO :FØ:BOOK
APPENDED :FØ:CHAP3 TO :FØ:BOOK
COPY BOOK TO :LP:
COPY *.* TO :FØ:
```

HDCOPY - Copia de pistas de disco duro.

sintaxis: HDCOPY <unidad 1> TO <unidad 2>|BACKUP

<unidad 1>:número de unidad fuente de la copia (Ø - 1).

<unidad 2>:número de unidad destino de la copia (Ø - 1).

BACKUP : copia disco unidad 1 a Ø y viceversa.

Errores de disco:

DISK-ERROR-UNABLE TO READ FROM SOURCE DISK ON DRIVE N.
LOGICAL ADDRESS (ttt,sss), STATUS = nnnn.
DISK ERROR UNABLE TO WRITE TO DESTINATION DISK ON DRIVE N
LOGICAL ADDRESS (ttt,sss), STATUS = nnnn.

ejemplo: HDCOPY 1 TO Ø
HDCOPY BACKUP

DELETE - borrar un fichero del disco.

sintaxis: DELETE <fichero1>[Q][,...,<fichero n>][P]

<fichero n>: nombre de los ficheros a borrar.

Q : pregunta para confirmar el borrado.

Errores : fichero, NO SUCH FILE.
fichero, WRITE PROTECTED.

ejemplo: DELETE CHAP?.*

RENAME - renombrar un fichero de disco.

sintaxis: RENAME <nombre anterior> TO <nombre nuevo>

<nombre anterior>: fichero existente a renombrar sin atributo "WRITE PROTECTED" ni "FORMAT".

<nombre nuevo>: nombre a ser asignado. Si ya existe pregunta si lo borra ó no:

<nombre nuevo>, ALREADY EXISTS, DELETE?

ejemplo: RENAME TEXT.BAK TO TEXT.OLD

ATTRIB - cambiar / presentar atributos de fichero.

sintaxis: ATTRIB <fichero>[<lista de atributos>][Q]

<fichero>: nombre del fichero cuyos atributos van a ser cambiados.

<lista de atributos>:

IØ-I1: Reset ó set del atributo "invisible".
WØ-W1: Reset ó set de atributo "Write protected".
FØ-F1: Reset ó set de atributo "Format".
SØ-S1: Reset ó set del atributo "System".

Q : espera a confirmar el cambio de atributos.

<fichero>, MODIFY ATTRIBUTES ?

ejemplo: ATTRIB PROGA.* W1

- Comandos de conversión de códigos.-

BINOBJ - conversión módulo objeto binario a absoluto.

sintaxis: BINOBJ <fichero bin> TO <fichero abs>

<fichero bin>: fichero conteniendo formato binario absoluto.

<fichero abs>: fichero conteniendo módulo objeto absoluto ejecutable bajo ISIS II.

ejemplo: BINOBJ PROGA.BIN TO PROGA.OBJ

HEXOBJ - conversión código hexadecimal a objeto absoluto

sintaxis: HEXOBJ <fichero hex> TO <fichero absoluto> [START(dirección)]

<fichero hex>: fichero de código objeto en formato hexadecimal.

<fichero abs>: fichero conteniendo módulo objeto absoluto para ser ejecutado por ISIS II.

START(dirección): dirección de la 1ª instrucción a ser ejecutada.

ejemplo: HEXOBJ PRIME.HEX TO PRIME OBJ START (32ØØH)

OBJHEX - conversión código objeto absoluto a hexadecimal. sintaxis: OBJHEX <fichero abs> TO <fichero hex>

<fichero abs>: fichero con módulo objeto absoluto.

<fichero hex>: fichero con conversión hexadecimal.

EDITOR DE TEXTOS:

Control-C: aborto de comando.

Control-R: impresión de línea actual.

Control-X: cancela la línea actual.

Rubout: corregir letras (ilegible)

sintaxis: EDIT <FICHERO 1>[TO <fichero 2>]

<fichero 1>: fichero a editar.

<fichero 2>: nombre del fichero ó periférico resultado.de la edición. Si no es aportado se guarda en <fichero 1>.

ejemplo: EDIT :HR: TO :F1:ASSY.SRC

B - Comando de principio de texto (Begin).

sintaxis: B$$

Z - Comando de fin de texto

sintaxis: Z$$

L - Comando de salto de n líneas.

sintaxis: [n]L$$ ejemplo: -5L$$

C - Comando de salto de n caracteres.

sintaxis: [n]C$$ ejemplo: -1ØØC$$

F - Comando de búsqueda.
sintaxis: F texto $$ ejemplo: FLOOP1$$

Comandos de texto.-

I - Comando de inserción.
sintaxis: I texto $$ ejemplo: IK$$

S - Comando de substitución.
sintaxis: S viejo texto [$ nuevo texto] $$
ejemplo: SJMP$CALL$$

D - Comando de borrado de n caracteres.
sintaxis: [n]D$$ ejemplo: -1ØD$$

K - Comando de eliminación de n líneas.
sintaxis: [n]K$$ ejemplo: ØL4K$$

Presentando un fichero.-

T - Comando de presentación de n líneas
sintaxis: [n]T$$ ejemplo: B5ØØT$$

Terminando una sesión.-

E - Comando de salida del editor y salvar edición.
sintaxis: E$$

Q - Comando de salida al ISIS II.
sintaxis: Q$$

W - Comando de escritura en disco de n líneas.
sintaxis: [n]W$$ ejemplo: 25W$$

Léyendo datos del disco.-

A - Comando de apéndice de 5Ø líneas.
sintaxis: A$$ ejemplo: AAA$$ (15Ø líneas)

Determinando espacio de memoria disponible.-

M - Comando de memoria.
nnnn - CHARACTER(S) AVAILABLE IN WORKSPACE.
sintaxis: M$$.

Iteraciones de comandos: (<>)
ejemplos: B1ØØ<SX1$LOOPCNT$$>$$
1ØØ<3C2<5CD>L>$$

- LINK: Comando de combinación y suma de módulos objeto.

sintaxis: LINK <lista entrada> TO <fichero salida> [<controles>]

<lista entrada>: 2 posibles;

<fichero>[(<módulo 1>, ..., <módulo n>)]

PUBLICS (<fichero 1>, ...,<fichero n>)

<fichero salida>:fichero objeto absoluto resultante.

<controles>:

MAP : se produce el mapa de la memoria.

NAME(<nombre de módulo>): asigna nombre al módulo resultante.

PRINT (<fichero>): especifica el fichero con el mapa del lincado. Por defecto es :CO:.

ejemplo: LINK FILNAM.EXT, PROG.LIB (TRIG), PROG.LIB &
**PUBLICS(TANSTA.AFL) TO SHIP.WRK &
**NAME(CELESTIGATION) MAP

LOCATE - comando de localización de ficheros objeto.

sintaxis: LOCATE <fichero entrada>[TO <fichero salida>] [<controles>]

<fichero entrada>: fichero con el código objeto relocalizable.

<fichero salida>: fichero con el módulo objeto absoluto.

<controles>: una ó mas palabras clave de control:

MAP: el mapa de memoria es listado en periférico de listado: B = byte relocalizable.
P = página relocalizable.
I = en página relocalizable.
A = absoluto

COLUMNS(n) : n=1,2 ó 3. Número de columnas en la tabla de símbolos.

PRINT(fichero): fichero de listado de LOCATE.

SYMBOLS: lista los símbolos locales del módulo.

LINES: los números de línea y nombre de los módulos de entrada son incluidos en la tabla de símbolos.

PUBLICS: los símbolos públicos son incluidos en la tabla de símbolos.

PURGE: nº de línea, símbolos locales, nombres de módulo y símbolos públicos son borrados del módulo final.

ORDER(secuencia de segmentos): establece el orden de los segmentos. Por defecto es: CODE, STACK, DATA y MEMORY.

NAME(nombre): nombre del módulo final.

RESTARTØ: coloca una instrucción JMP en posiciones Ø, 1 y 2 en módulo absoluto.

START(dirección): dirección de la 1ª instrucción a ejecutar en el segmento de código.

STACKSIZE(valor): valor en bytes de la longitud del stack.

ejemplo: LOCATE ORDER(DATA, STACK) CODE (6ØØØH)

LIB - Comando librería (122 caracteres/línea)

Comandos de LIB:

CREATE : crear un fichero librería.
sintaxis: CREATE <fichero>

ADD : sumar módulos a una librería.
sintaxis: ADD <fichero>[(<módulo>,...)]TO <fichero librería>

<fichero>: nombre del fichero librería ó fichero con un módulo objeto.

<módulo>: módulos del fichero librería a sumar.

<fichero librería>: fichero modificado por la adición.

DELETE: borrar módulos de una librería.
sintaxis: DELETE <librería>(<módulo>, ...)

<módulo>: módulo objeto a borrar de la <librería>

LIST : lista módulos de librería y sus símbolos públicos.
sintaxis: LIST <librería>[(<módulo>,...)][TO <fichero de listado>][PUBLICS]

<librería>: librería cuyos módulos van a ser listados.

<fichero listado>: fichero donde va a parar el listado. Por defecto es :CO:.

PUBLICS: son listados los símbolos públicos.

EXIT: retorno al ISIS II.

ejemplo:
```
-LIB
 ISIS II LIBRARIAN Vx.y
*CREATE FOO.LIB
*ADD SIN, OBJ, COS.OBJ, TO FOO.LIB
*LIST FOO.LIB
   SINE
   COSINE
*EXIT
```

RUTINAS DEL SISTEMA:

Rutinas de entrada / salida de fichero.-

OPEN - Inicialización de fichero para operación entrada/salida.

READ - Transferencia de datos de fichero a memoria.

WRITE - Transferencia de datos de memoria a fichero.

SEEK - Marcador de posición de fichero de disco.

RESCAN - Posicionar el marcador a principio de línea.

CLOSE - Terminar operaciones de entrada/salida en fichero.

SPATH - Obtener información de fichero.

Mantenimiento de directorio de disco.-

DELETE- Borrar un fichero del directorio de disco.

RENAME - Cambiar nombre de fichero de disco.

ATTRIB - Cambiar atributos de fichero de disco.

Reasignación de consola y salida de mensaje de error.

CONSOL -Cambio de aparato de consola

WHOCOM - Determinar fichero asignado como consola.

ERROR - Salida de mensaje de error por consola.

Ejecución de programas.

LOAD - Cargar un fichero de código ejecutable y transferencia de control.

EXIT - Terminar programa y retornar al ISIS II.

Rutinas de interface I/O del Monitor.-

CI - Rutina de entrada de consola (console input)

CO - rutina de salida de consola (console output)

RI - rutina de entrada de lectura (Reader input)

PO - rutina de salida de perforadora (Punch output).

LO - rutina de salida de listado (list output)

UI - rutina entrada grabador EPROM.

UO - rutina de salida grabador de EPROM.

Rutinas de Status del sistema.-

CSTS - rutina de status de consola de entrada.

UPPS - rutina de status de grabador de EPROM.

IODEF - rutina de definición de entrada/salida.

IOCHK - rutina de chequeo de configuración I/O.

IOSET - rutina de activación de configuración I/O.
MEMCHK - rutina de chequeo de tamaño de RAM.

PROGRAMA MONITOR.-

Entrada de comandos : <comando>[<parámetro>]<CR>
Comandos:
A - comando de asignación (assign).
sintaxis: A<aparato lógico>=<aparato físico>
<aparato lógico>: C=console=(T,C,B,1) <aparato físico>
R=reader= (T,P,1-2)
P=punch = (T,P,1-2)
L=list = (T,C,L,1)
(T=teletipo, p=lectora papel, C=pantalla, L=impresora, B=batch, 1-2= definido por usuario).
Q - comando de status (Query)
sintaxis: Q<cr>
D - comando de presentación de memoria (display)
sintaxis: D<dirección baja>,<direcc.alta><cr>
F - comando de relleno (fill).
sintaxis: F<dir.baja>,<dir.alta>,<constante><cr>
M - comando de movimiento de memoria.
sintaxis: M<dir.inicial>,<dir.final>,<dir.destino>
S - comando de substitución de memoria.
sintaxis: S<dirección>,[<byte>][,<byte>]...<cr>
X - comando de registros (presentación)
sintaxis: X<cr>
X - comando de registros (modificación).
sintaxis: X<registro>,[<dato>][,<dato>][...]<cr>
Comandos de entrada / salida a cinta de papel.
R - comando lectura.
sintaxis: R<bias><cr>
W - comando escritura.
sintaxis: W<direc.inicial>,<direcc.final><cr>
E - comando fin de fichero
sintaxis: E<punto de entrada><cr>
N - comando nulo
sintaxis: N<cr>
Comandos de ejecución:
G - comando de ejecución.
sintaxis: G[<dir.inicial>][,<punto corte 1>][,punto

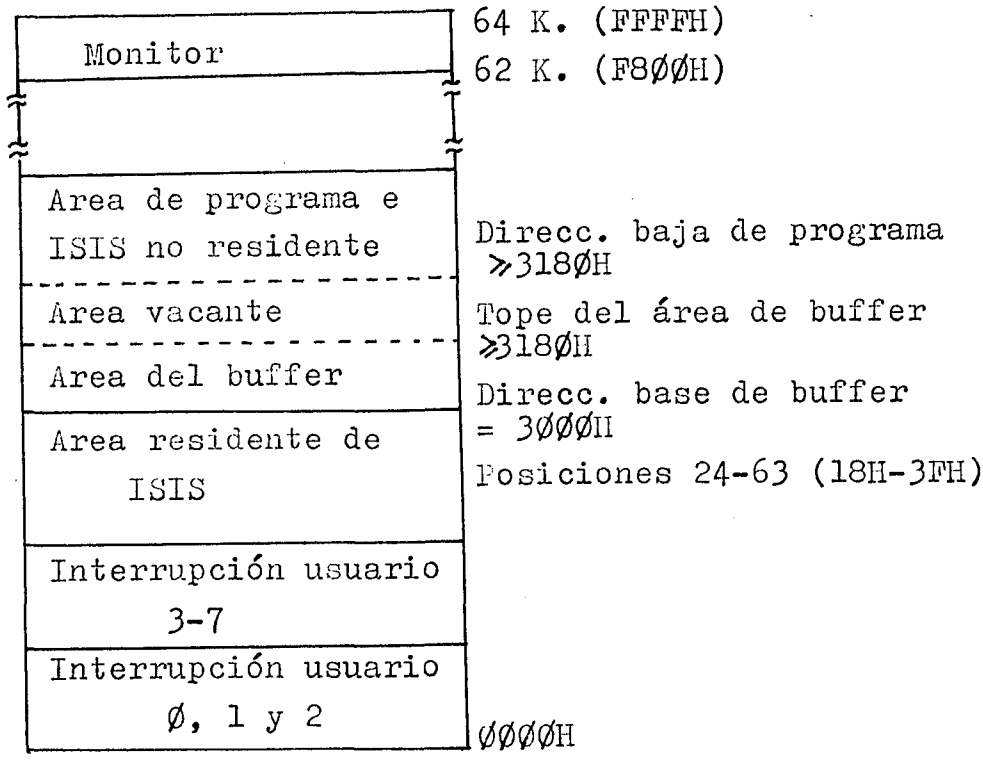
corte 2>]<cr>

Comando de utilidad.-

H - comando hexadecimal (suma y resta).

sintaxis: H<número 1>,<número 2><cr>

Organización y localización de la memoria.



Códigos de control del Terminal Intel.

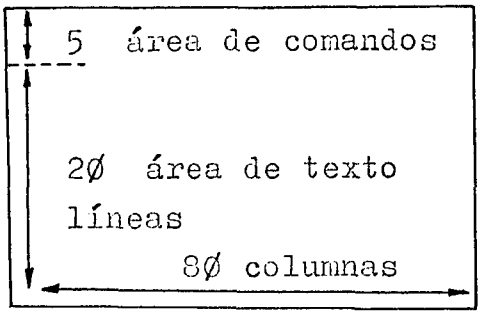
Función cursor	Tecla de cursor código entrada	Mov. cursor CRT código salida
Abajo ↓	1Ch, Ø	1Bh, 42h
Home	1Dh, Ø	1Bh, 48h
Izquierda ←	1Fh, Ø	1Bh, 44h
Derecha →	14h, Ø	1Bh, 43h
Arriba ↑	1Eh, Ø	1Bh, 41h

Función CRT	Código entrada
Limpiar pantalla	1Bh, 45h
Limpiar resto de pantalla	1Bh, 4Ah
Limpiar línea	1Bh, 4Bh
Carácter blanco	2Øh

APENDICE B:

Editor de texto: CREDIT.

CREDIT - Editor de textos del Sistema de Desarrollo Intellec series II, MDS-221.



Display: 25 líneas x 8Ø columnas.

Carácter literalizante = "\"

Entrada hexadecimal : ↑B 41 42 43 44 ↑B

Tags: TT (top of text). Principio de fichero.
TE (text end). Fin de fichero.
TB (text begin). Principio de fichero en memoria.
TZ Fin de fichero en memoria.

Buffer de texto = 2Ø kbytes.

CREDIT.MAC = fichero de ejecución de comandos CREDIT y macrodefiniciones.

sintaxis: CREDIT fichero 1[TO fichero 2][MACRO|(fichero comando)]|NOMACRO]

EX - (exit) comando de salida y salvar edición.
sintaxis: EX [fichero]

EQ - (Quit) comando de salida.
sintaxis: EQ

Funciones de edición de pantalla (zona de texto)

(↑A) - Añadir texto. sintaxis: ↑A texto ↑A
(↑C) - Añadir carácter. sintaxis: ↑Cx
(↑Z) - Borrar texto. sintaxis: ↑Z<movim.cursor>↑Z
(↑D) - Borrar carácter. sintaxis: ↑D
(↑V) - Posicionar cursor en 3ª línea de zona de texto.
sintaxis: ↑V
(↑N) - Próxima página. sintaxis: ↑N
(↑P) - Página previa. sintaxis: ↑P

Edición de línea de comandos.

sintaxis: *comando; comando; comando; &
*comando <cr>

sintaxis: *comando [argumento]

H - Comando de ayuda (help). sintaxis: H

Comandos de posicionamiento de cursor.-

L - salto de líneas. sintaxis:L[número de líneas]

J - salto de caracteres. sintaxis:J[nº caracteres/tag]

Comandos de tag.-

TS - Definición de tag. sintaxis: TS n , $\emptyset \leq n \leq 9$

TD - Borrar tag. sintaxis: TD n , $\emptyset \leq n \leq 9$

Comandos de texto.-

P - presentación de líneas a partir del cursor.
sintaxis: P[n|tag]

PH - Presentación hexadecimal de n caracteres.
sintaxis: PH[n|tag]

I - Inserción de texto. sintaxis: I/texto/

DL - Borrar líneas. sintaxis: DL[nº de líneas]

DC - Borrar caracteres. sintaxis:DC[nº de caract./tag]

XM - Mover texto. sintaxis:XM tag,{tag/±nº de líneas}

XC - Copiar texto. sintaxis:XC tag,{tag/±nº de líneas}

Comandos de búsqueda.-

F - encontrar texto.sintaxis:F/cadena/[tag|número]

S - substituir texto.
sintaxis: S/texto viejo/texto nuevo/[número|tag]

SQ - Substituir y confirmar.
sintaxis: SQ/texto viejo/texto nuevo/[número|tag]

Macrofacilidades.-

MS - (Macroset). Definición de función macro.
sintaxis: MS nombre/texto/ %=parámetro

MF - Macro función. Llamada a una función macro.
sintaxis: MF nombre[(parámetro[,... parámetro]]
↑F nombre (modo texto)

MD - Borrado de una función macro ó todas.
sintaxis: MD {nombre|*}

?M - Presentación de funciones macro.
sintaxis: ?M

Comandos de ejecución condicional:

QU - (query user). Recoger respuesta de usuario.
sintaxis: QU

QT - Condición usuario afirmativa.
sintaxis: QT;[<] comando[>]

QF - Condición usuario negativa.
sintaxis: QF;[<] comando[>]

YT - ejecución si flag es afirmativo.
sintaxis: YT;[<]comando[>]

YF - ejecución si flag es falso.
sintaxis: YF;[<] comando[>]

EL - comando de salida de lazo. sintaxis: EL

U - comando de salida de mensaje a usuario.
sintaxis: U/texto/

G - comando cargar fichero ejecutable CREDIT/
sintaxis: G nombre del fichero

OR - comando abrir para leer fichero.
sintaxis: OR fichero

OW - comando abrir para escribir fichero.
sintaxis: OW fichero

B - comando comienzo fichero sintaxis: B

R - comando leer fichero - número de líneas.
sintaxis: R[n]

W - comando escritura en fichero - n líneas.
sintaxis: W[n]

CR, CW - cerrar fichero de lectura, escritura.
sintaxis: CR sintaxis: CW

Comandos de alteración.-

A - Comando de alteración de configuración.
sintaxis: A código = valor
L = término de línea
V = número de líneas por pantalla.
S = T - "NOT FOUND" suprimido.
F - "NOT FOUND" no suprimido.
T = tab = Ø a 79
B = caracter break (ESC).
C = carácter no imprimible (↑).

AF - comando de alteración de función.
sintaxis: AF código = valor

?A - comando de presentación de características de edición. sintaxis: ?A

APENDICE E :

ICE-85 : In Circuit Emulator.

ICE-85: In circuit Emulator 8Ø85 Family.

Caracteres alfabéticos:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

Caracteres numéricos:

Ø123456789 (ABCDEF : caracteres hexadecimales).

Caracteres especiales: +-<=>$'&).(;*/#

Números de sentencias:

sintaxis: nombre de módulo # decimal-1Ø

Constantes numéricas:

sintaxis: digito(s) [base]

base: Y(binario), Q(octal), T(decimal), H(hexadecimal), K(decimal x 1024).

Operadores:

Tipo	Nombre de cla e	Operadores
relacional	rel - op.	=,<,>,<=,>=,<>
suma	plus - op.	+,-,(binario y unitario)
multipl.	mult.- op.	*,/,MOD

Puntuación:

Tipo	Nombre de clase	Caracteres de puntuac.
puntuación	punct. - op.	'&.,;()$ CR LF SP

COMANDOS.-

Comando ICE85: sintaxis: :unidad:ICE85

EXIT - Comando salida. sintaxis: EXIT

LOAD - Comando cargar módulo objeto.
sintaxis: LOAD :unidad:fichero { NOCODE / NOSYMBOL / NOLINE }

SAVE - Comando salvar módulo objeto.
sintaxis: SAVE :unidad:fichero { NOCODE/partición / NOSYMBOL / NOLINE }

LIST - Comando listado de sesión.
sintaxis: LIST:aparato:
LIST:unidad:fichero

Comandos de base de números.-

SUFFIX - Comando de establecer o presentar base de entrada. sintaxis: SUFFIX (presentación)
SUFFIX=|Y/Q/T/H|

BASE - Comando de establecer ó presentar base de salida.
sintaxis: BASE
BASE = | Y/Q/T/H/ASCII |

EVALUATE - Comando evaluación.
sintaxis: EVALUATE número.

Comandos de mapeado de memoria y puertos I/O.

MAP - Comando de modo mapeo.
sintaxis: MAP | MEMORY | = | SHARED / UNSHARED |

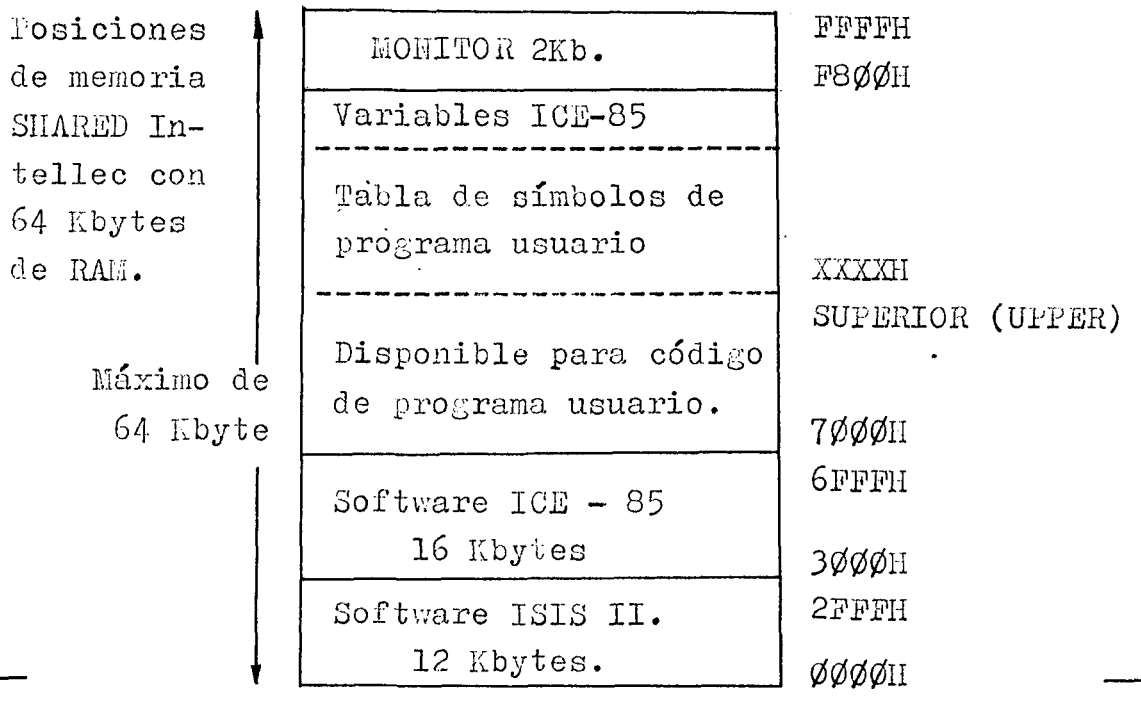
MAP - Comando mapeo
sintaxis: MAP [MEMORY] (partición bloque) =
= | GUARDED / USER [NOVERIFY] / INTELLEC Bloque [NOVERIFY] |

[partición de bloque]:
- nombre de bloque TO nombre de bloque -1
- nombre de bloque TO dirección alta
- nombre de bloque LENGTH nº de direcciones

MAP - Comando de mapeo de puertos I/O.
sintaxis: MAP IO (segmento de partición) = | GUARDED / USER / INTELLEC |

(segmento de partición):
- nombre segmento TO nombre segmento - 1
- nombre segmento TO puerto alto.
- nombre segmento LENGTH Nº de puertos

MAP - Comando presentación de status de mapeo.
sintaxis: MAP [MEMORY] {bloque de partición}
sintaxis: MAP IO [segmento de partición]



RESET MAP - Comando de reset de mapeo.
sintaxis: RESET MAP

Comandos de registros hardware.-
Registros 8 bits: RA, RB, RC, RD, RE, RF, RH, RL.
Registros pares: RBC, RDE, RHL, SP, PC.
Flag de estado: CY, PY, ACY, Z, SN.
Bits de interrupción: M5, M6, M7, IE, I7, SID, SOD.
Registros de estado: OPCODE, CAUSE, PPC, PSW, UPPER, BUFFERSIZE, TIMER, HTIMER.
Líneas de sincronización externa: SYØ, SY1.
Otras líneas externas: MATCH Ø, MATCH 1, EMUL, GND.

Comandos de presentación registros status y procesador.
sintaxis: nombre de registro
nombre de registro par
nombre de flag
nombre de i-bit
REGISTER
registro de status

Comando de asignación de registro.-
registro procesador = valor

RESET - Comando de Reset Hardware.-
sintaxis: RESET | HARDWARE |
| I7 |

ENABLE/DISABLE TIMEOUT - Comando de habilitación de errores. Sintaxis: | ENABLE | TIMEOUT
| DISABLE |

Comandos de contenido de memoria y puertos.-
Dirección de tipo memoria = BYTE, WORD, IBYTE, IWORD.
Partición de tipo memoria: dirección TO dirección
dirección LENGTH nº de bytes (BYTE ó IBYTE)
direcc. LENGTH nº de palabras(WORD ó IWORD)

Comandos de presentación de contenido de memoria y puerto
sintaxis: partición de tipo de memoria.
PORT número de puerto

Comando de asignación de memoria.-
sintaxis: partición de memoria = nuevo contenido [,nuevo contenido] ...

Comandos de asignación de puertos.-

sintaxis: PORT número de puerto = nuevo contenido

* Comandos de tablas de símbolos y números de línea.

Comandos de presentación de tabla de símbolos.-

sintaxis: referencia simbólica { ..módulo.ref.simbólica
SYMBOL { .referencia simbólica
referencia de sentencia (# nº de sentencia)

DEFINE - comando de definición de símbolos

sintaxis: DEFINE ref.simbólica = dirección / valor

Comando de cambio de símbolo.-

sintaxis: ref. simbólica = dirección / valor

REMOVE - Comando de borrado de símbolos.-

sintaxis: REMOVE ref. simbólica [, ref. simbólica]...

* Comandos de grupos de canales.-

Canales de usuario, del 8Ø85 uP. y grupos del sistema.

Grupo del sistema	Números de canales	Señales 8Ø85	Interpretación	Base de traza
UØ	1-8		Canales prueba usuario	H
U1	9-16		Canales de prueba usuario	H
U2	17-18		Canales de prueba usuario	H
DMUX	19	ALE	1:ADØ-7=DATA, Ø:ADØ-7=Dir.	Y
ADDR	2Ø-35	ADØ-7,	(DMUX=Ø) Dir.menos signif.	H
		A8-A15	Dirección más significativa	
DATA	2Ø-27	ADØ-7	Líneas de datos.(DMUX=1)	H
ADDRL	2Ø-27	ADØ-7	Direcc. menos significativa	H
ADDRH	28-35	A8-A15	Direcc. más significativa	H
STS	36	SØ	Acción IO/$\overline{M}$ S1 SØ Mnemon.	
	37	S1	HALT Ø Ø Ø H	
	38	IO/$\overline{M}$	WRITTEN Ø Ø 1 W	
			READ Ø 1 Ø R	
			EXECUTED Ø 1 1 E	
			OUTPUT 1 Ø 1 O	
			INPUT 1 1 Ø. I	
SD	39	SOD	Salida serie	Y
	4Ø	SID	Entrada serie	Y
RW	41	WR	Línea WR (escritura)	Y
	42	RD	Línea RD (lectura)	
MTH	43	Match Ø		Y
		Match 1		

DEFINE GROUP - Comando de definición de grupo.
sintaxis: DEFINE GROUP grupo = lista canales [IN base de grupo]
GROUP: Comando de presentación de grupo.
sintaxis: GROUP [lista de grupos]
GROUP - Comando de cambio de grupo.
sintaxis: GROUP grupo = lista canales [IN base de grupo]
REMOVE GROUP: Comando de borrado de grupo.
sintaxis: REMOVE GROUP [lista de grupos]

Comandos de control de emulación en tiempo real.-
Condiciones de parada seleccionada.-

Nemónico de condición	Grupos de canales del sistema			
	DMUX	ADDR	DATA	STS
HALT	XY	XXXXH	XXH	ØØØY
[LOCATION] dirección	Ø	Dirección	XXH	XXXY
[LOCATION]direcc.,status	Ø	Dirección	XXH	status
[LOCATION]masc. de direcc.	Ø	Masc. direc.	XXH	XXXY
[LOCATION]masc.dir.,status	Ø	Masc. direc.	XXH	status
VALUE data	1	XXXXH	data	XXXY
VALUE data status	1	XXXXH	data	status
VALUE máscara de data	1	XXXXH	msc.dat.	XXXY
VALUE masc. de data status	1	XXXXH	msc.dat.	status

registro de parada = {BRØ, BR1} | máscara numérica | ON lista de canales

GO - Comando de ejecución.
sintaxis: GO [FROM dirección][condiciones de parada]
GO [FROM dirección][TILL condición [OR condición][OR SYØ]]
GO REGISTER - Comando de definición del registro GO.
sintaxis: GR = condición de parada
GR = TILL condición [OR condición][OR SYØ]
Presentación de registros emulación.-
sintaxis: GR
registro de parada [lista de canales]
Comandos de definición de registro de parada
sintaxis: registro parada = condición
lista de canales de registro parada = | máscara numérica |
RESET - Comando de Reset de registro de parada
sintaxis: RESET registro parada [lista de canales]

ENABLE / DISABLE SYOUT - Comando habilitación SYØ

sintaxis: ENABLE SYØ OUT
DISABLE SYØ OUT

* Comandos de control de trazas.-

sintaxis: TRACE = | FRAME
CYCLE
INSTRUCTION |

MOVE, OLDEST, NEWEST - comandos de movimiento de puntero de traza. sintaxis: MOVE [[+/-]decimal]
OLDEST
NEWEST

PRINT - Comando de presentación de la traza

sintaxis: PRINT ALL
PRINT [[+/-]decimal]

Comando de definición del registro cualificador.-

sintaxis: lista de canales del Reg. Cualif.= | máscara
numérica |

RESET - Comando de Reset del registro de cualificación

sintaxis: RESET reg. cual. [lista de canales]

Comando de presentación de controles de traza.-

sintaxis: TRACE
Registro cualif. [lista de canales]

ENABLE / DISABLE - comando de habilitación de traza

sintaxis: | ENABLE
DISABLE | | SY1 IN
SY1 OUT
STOPTRACE |

Comandos de control de emulación paso a paso.-

Comandos de definición del registro de condición.-

sintaxis: registro de condición = expresión dondicional
(CRØ, CR1, CR2, CR3)

SR - comando de definición de registros.-

sintaxis: (1) SR = FOREVER

(2) SR = [COUNT expr-1Ø][TILL reg-cond] | AND
OR | reg-cond]...&3]

(3) SR = [COUNT expr-1Ø][TILL expr-cond[| AND
OR | cond-expr]..&3]

STEP - comando paso a paso.

sintaxis: (1) STEP [FROM dirección][FOREVER]

(2) STEP[FROM dirección][COUNT expr-1Ø][TILL reg-cond
[| AND
OR | cond-reg]... &3]

(3) STEP[FROM dirección][COUNT expr-1Ø][TILL exp-cond
[| AND
OR | exp-cond]... &3]

Comando presentación de registros STEP.

sintaxis: SR
registro condicional (CRØ,CR1,CR2,CR3)

ENABLE/DISABLE DUMP - comando de volcado paso a paso

sintaxis: ENABLE DUMP { partición / CALL / JUMP / RETURN }

DISABLE DUMP

Comandos CALL externos.

sintaxis: | CALL / ICALL | dirección [(dirección[,dirección])]

EXECUTE :unidad:fichero[(dirección[,direcc])]

APENDICE H:

Comunicaciones RS-232.

COMUNICACION RS-232c:

Introducción histórica.-

Los datos han sido transmitidos entre los continentes desde 1.866. De hecho, los puntos y rayas del código Morse son los precursores de los ceros y unos usados hoy en la comunicación de datos del ordenador.

Cuando los ordenadores llegaron a escena, la tecnología del teleimpresor y teletipo estaba casi madura y bien definida (aunque primitiva); por tanto es fácil ver porqué fueron naturalmente adoptados como dispositivos de entrada /salida.

Pronto se hizo económicamente deseable para los usuarios el acceso a ordenadores de localidades remotas. Distancias cortas - una treintena de metros, quizá en el mismo edificio - podrían salvarse mediante la adición de cables extra. Pero la atracción del acceso a distancias remotas se hicieron notar y los ingenieros de ordenadores empezarn ansiosamente a echarle el ojo a las líneas de teléfono. Después de todo, la compañía telefónica ya tenía cables extendidos a casi todas partes ... mejor incluso, eran baratos de alquilar.

Pero los datos del ordenador no pueden introducirse directamente dentro de las líneas de teléfono. Se requiere un dispositivo para adaptarlos: el modem.

Según la actividad en el campo de las telecomunicaciones aumentó rápidamente, muchas clases diferentes de equipos empezaron a aparecer. El sistema Bell vió en ellos poco que le gustase y mucho que podría comprometer y complicar el servicio de las comunicaciones al público. Las compañías telefónicas prohibieron consecuentemente la conexión de la mayoría de estos dispositivos.

El modelo de la conexión RS-232c.-

La situación pedía un modelo a gritos, que no tardó en llegar. En 1.969, la EIA (Asociación de Industrias Electrónicas), los laboratorios Bell y los fabricantes de equipos de comunicaciones formularon cooperativamente y emitieron el EIA RS-232, que casi inmediatamente experimentó revisiones menores, convirtiéndose en la RS-232c. Un modelo similar fué emitido por la Organización Internacional de mode-

los Comité Consultivo Internacional sobre Telefonía y Telegrafía (CCITT). Para que los usuarios de microordenadores entiendan este modelo heredado se debe recalcar que la conexión RS-232c fué desarrollada para un único propósito, establecida formalmente por su título:

Conexión entre un Equipo Terminal de Datos empleando un intercambio de datos binarios en serie.

Cada palabra del título es significativa: describe la conexión entre un terminal (Equipo Terminal de Datos ó DTE) a un modem (Equipo de Comunicación de Datos ó DCE) para la transmisión de datos en serie.

El documento consta de cuatro partes.

*Características de la señal eléctrica: esto describe el "lado" eléctrico que presentará la conexión y que requerirá del mundo externo. Se definen aquí los niveles de voltaje que representarán al "Ø" y "1" lógicos.

* Características mecánicas de la conexión (conectores). Esta sección establece que la conexión debe consistir en una clavija y un receptáculo y que el receptáculo estará en el DCE. Se especifican la asignación de números a las patillas, pero debería notarse que el propio conector no ha sido especificado. El familiar conector en forma de "D", el DB-25, que ahora es casi un sinónimo de las conexiones en serie, se deriva de otro organismo de modelos, la Organización Internacional de Modelos ó ISO.

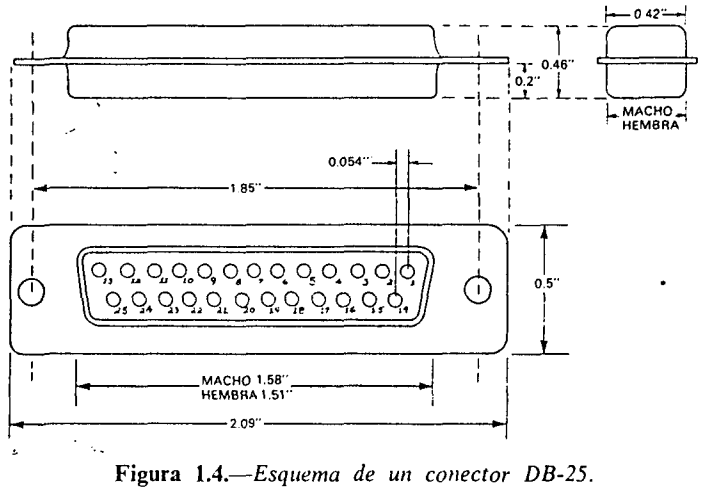


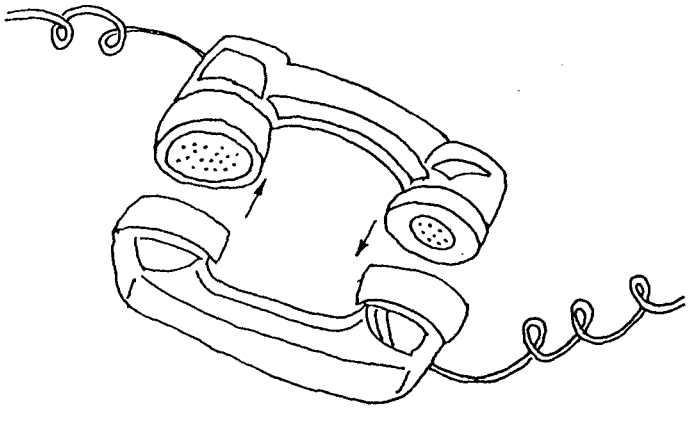
Figura 1.4.—*Esquema de un conector DB-25.*

* Descripción funcional de circuitos de intercambio: esta sección define y da nombres aa las funciones de las señales eléctricas a usar. Así por ejemplo, se asignan los DATOS TRANSMITIDOS a la patilla 2. Hay 21 definiciones similares, pero sólo unas cuantas son relevantes a los microordenadores.

* Conexiones modelo para configuraciones seleccionadas de sistemas de comunicación: son ejemplos de tipos comunes de conexión entre modem y terminal.

- Acoplamiento (handshaking).-

El acoplamiento es el modo en que se regula el flujo de datos a través de la conexión.



Hay dos tipos de acoplamiento: acoplamiento por software y acoplamiento por hardware.

El acoplamiento por software existe cuando un dispositivo controla a otro por medio del contenido de los datos. por ejemplo, una manera de controlar a una impresora es hacer que el ordenador le envíe sus caracteres línea a línea. Al final de cada línea, el ordenador pone un carácter que dice a la impresora: "Este es el final de la línea ... estoy esperando tu señal para enviarte la siquiente línea." La impresora acepta la línea, la imprime y envía de vuelta un carácter al ordenador con el significado: "Estoy lista para otra línea".

Por el contrario, con el acoplamiento por hardware una impresora puede realmente forzar al ordenador a pararse en el envío de caracteres cambiando simplemente el voltaje en un cable. La desventaja es que sólo puede usarse cuando los dispositivos pueden conectarse físicamente a través de un cable. Esto hace que no se pueda usar con los modems.

El carácter insertado por el ordenador es usualmente el carácter ASCII "END OF TEXT" (fin de texto) = CTRL-D = Ø3H ó ETX. Cuando el dispositivo receptor está listo para otro montón de datos, el carácter que envía de vuelta al ordenador es el caracter ASCII "ACKNOWLEDGE" (reconocimiento = CTRL-F = Ø6H) ó ACK. Por consiguiente esta forma de acoplamiento por software se conoce como protocolo ETX/ACK. Existen otros protocolos como ENQ/ACK, CTRL-S/CTRL-Q y XON/XOFF

Modos de configuración:

Existen dos modos de configuración: DTE (equipo terminal de datos) y DCE (equipo de comunicación de datos).

En general un DCE es un equipo que pasa datos mientras un DTE es un destino real de datos. Un ejemplo de DCE es el modem. Ejemplos del DTE son el terminal de pantalla (y teclado) y la impresora. Puesto que puede realizar ambos, un ordenador no encaja en ninguna de estas dos categorías, aunque su configuración más corriente es la de DTE.

Diagrama de bloques.-

(Terminal de pantalla) DTE				DCE (Modem)
Transmisión de datos	2	TxD	RxD	2 Receiver data
Recepción de datos	3	RxD	TxD	3 Transmissor data
Petición de envío	4	RTS	CTS	4 Clear to send.
Borrado para enviar	5	CTS	RTS	5 Request to send.
Conjunto de datos listo	6	DSR	DTR	6 Data terminal ready
Circuito común	7	GND	GND	7
Detector de portadora	8	DCD	DCD	8 Data carrier detect
Terminal de datos listo	2Ø	DTR	DSR	2Ø Data set ready
(conector macho)				(conector hembra)

Funciones de las patillas:

Patilla 1: Masa de protección. También llamada "masa de la carcasa". Se usa para prevenir descargas eléctricas en caso de que falle la corriente.

Patilla 2: Transmisión de datos. (TxD = transmitter data). Transmite datos del DTE al DCE.

Patilla 3: Recepción de datos (RxD = Receiver data). Recibe datos del DCE.

Patilla 4: Petición de envío. (RTS = Request to send). Salida de propósito general. Sirve para activar el CTS del DCE y así hacer que no se pare la comunicación.

Patilla 5: Borrado para enviar. (CTS = clear to send). Entrada de propósito especial. Se usa para deshabilitar el transmisor al estar el DCE ocupado.

Patilla 6: Conjunto de datos listo. (DSR = data set ready). Entrada de propósito general para comunicar al DTE que el DCE está encendido y listo para funcionar.

Patilla 7: Circuito común. Punto de referencia para todos los voltajes de la conexión. Obligatorio.

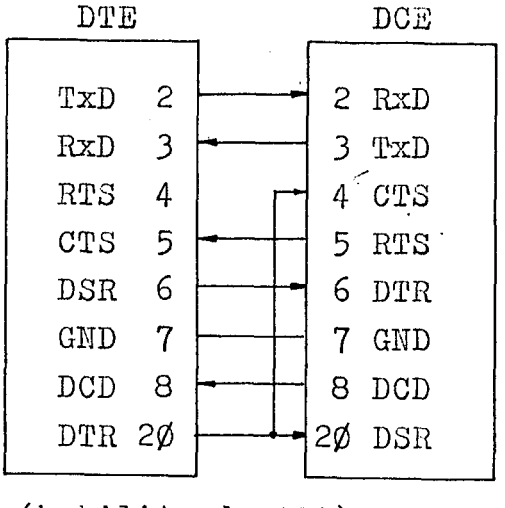
Patilla 8: Detector de portadora (DCD = data carrier detect). Sus usos varían pero en un DTE se usa frecuentemente para deshabilitar la recepción de datos.

Patilla 2Ø: Terminal de datos listo. (DTR = data terminal ready). Salida de propósito general. Generalmente usada para comunicar al DCE que el DTE está encendido y listo para funcionar.

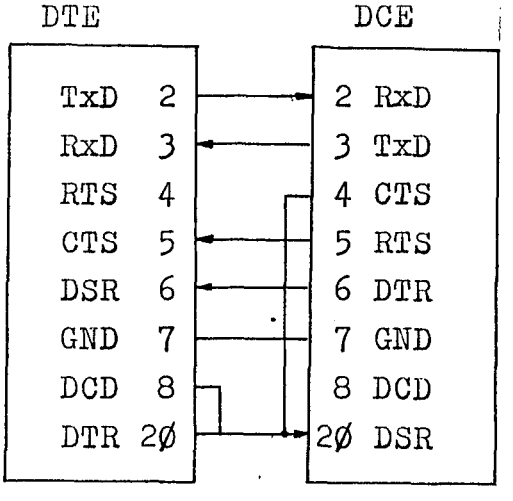
Hay otras patillas, pero cuando se emplean, usualmente proporcionan características opcionales o secundarias únicas para ese fabricante.

Configuraciones de transmisión trucada.-

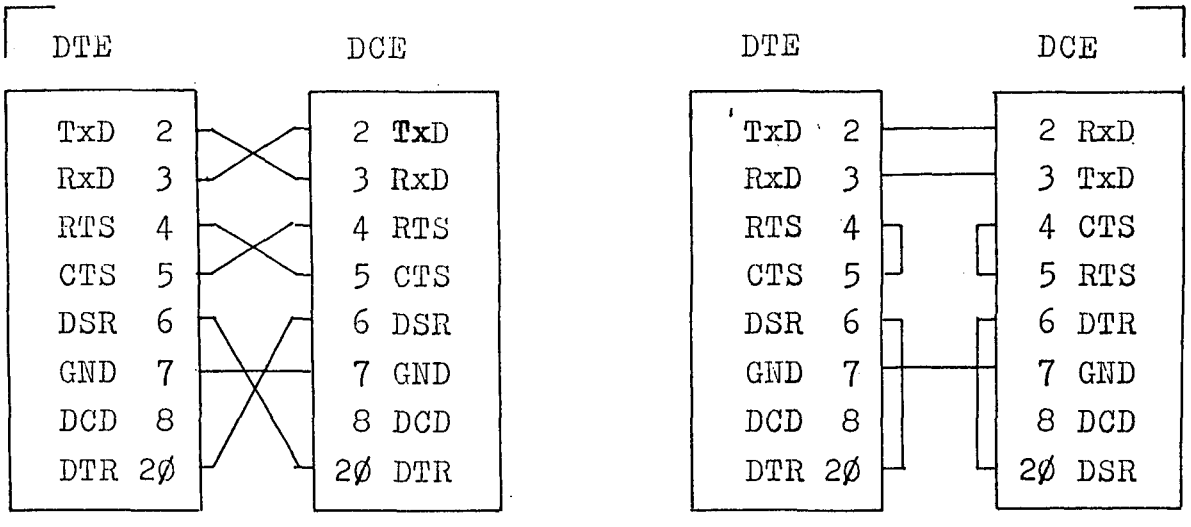
Puede ser necesario trucar la comunicación al no tener uno ó ambos equipos las señales necesarias para el acoplamiento. O aún teniéndolas no es necesaria su conexión directa.



(habilitando CTS)



(habilitando CTS y DCD)



(conversión de sexo) (modelo de 3 cables)

Niveles lógicos.-

Como regla general, las USART operan con la misma fuente de alimentación de 5 voltios que otros circuitos integrados dentro del ordenador. La conexión RS-232c, sin embargo, define su propio y nuevo entorno eléctrico. Son voltajes que oscilan entre +25v. a -25 voltios bajo ciertas circunstancias.

Un tipo de circuito integrado, el exitador de línea RS-232c, traduce los voltajes de salida de la UART a aquellos requeridos en las normas de conexión RS-232c, mientras que otro, el receptor de línea RS-232c convierte los voltajes RS-232c a niveles requeridos por la circuitería de la UART.

Definiciones lógicas:

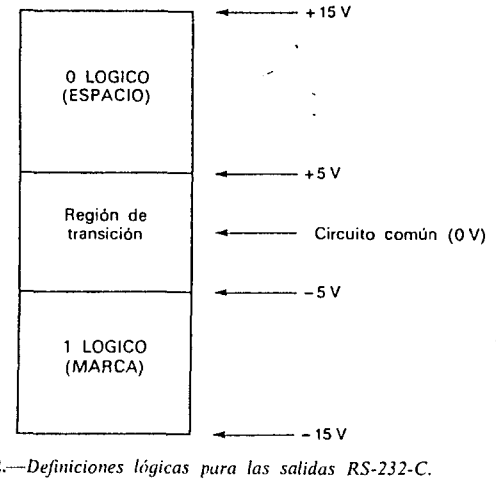


Figura 5.2.—*Definiciones lógicas para las salidas RS-232-C.*

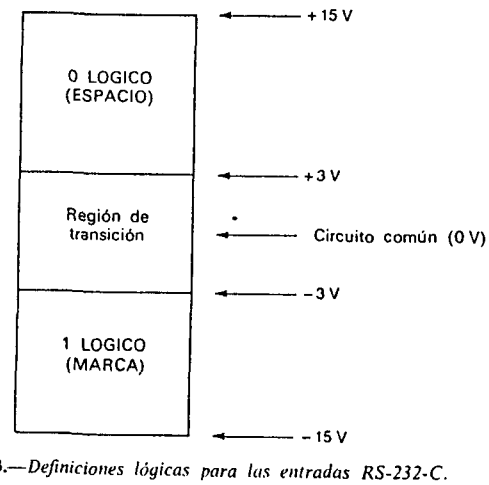


Figura 5.3.—*Definiciones lógicas para las entradas RS-232-C.*

Margen de ruidos.-

Se conoce como margen de ruidos del circuito a la diferencia entre las definiciones para los voltajes mínimos permisibles de la región de transición.

Al aumentar la velocidad de transmisión, las señales de datos se vuelven susceptibles de pérdidas de voltaje causadas por la capacidad e inductancia en el cable. Estas pérdidas, conocidas como efectos de alta frecuencia, aumentan con la longitud del cable.

La EIA limita la capacidad total del cable a 2500 picofaradios. Como un valor medio para un cable es entre 12Ø y 15Ø picofaradios por metro, unos 15Ø metros es la mayor distancia que puede tener teóricamente un cable.

Marca y espacio.-

A causa de la confusión engendrada por eso de la lógica invertida de las normas, la literatura sobre la materia se ha vuelto inconsistente y confusa. La tabla siguiente resume algunos de los términos en uso.

Ø lógico	1 lógico
espacio	marca
encendido	apagado
empezar	parar
falso	verdadero
positivo	negativo
bajo	alto
perforación	no perforación
ON	OFF

Orden de la transmisión de bits.-

Los datos se transmiten al revés. El bit menos significativo se transmite antes, seguido de los otros en orden inverso a su significancia.

UART.- (Universal Asynchronous Receiver Transmitter). Virtualmente se lleva a cabo el proceso completo de E/S en serie mediante un único circuito integrado, conocido genéricamente como UART. La UART se conoce como un dispositivo de servicio debido a que releva al procesador del tedioso trabajo de Entrada / salida en serie.

Fundamentos.- Funcionalmente la UART reune una sección de transmisor para convertir un octeto de 8 bits en una co

rriente en serie de 8 bits y una sección de RECEPTOR que reconvierte una corriente entrante de bits en un octeto de datos. Además hay una sección de CONTROL y ESTADO que, entre otras cosas recibe el estado lógico de varias patillas de entrada, y cuando un programa la llama, cambia los niveles lógicos de varias patillas de salida.

A continuación se muestran estas tres secciones.

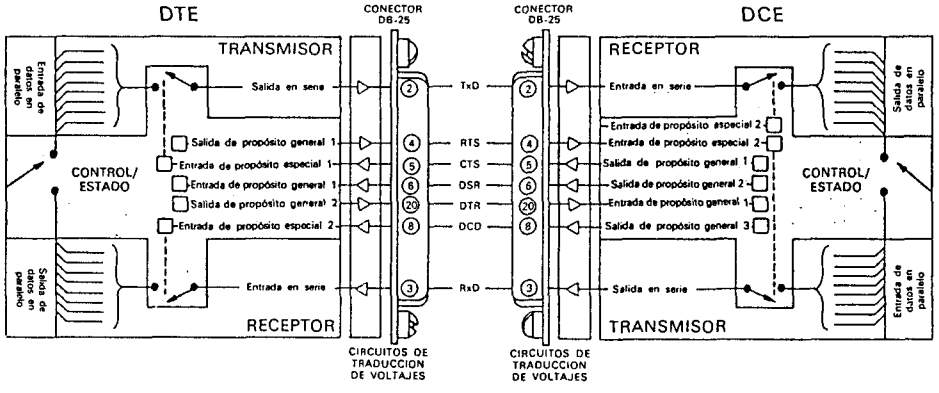


Figura 5.1.—*Otro nivel aísla la UART de los altos voltajes del RS-232-C.*

La parte de arriba del diagrama muestra la sección del transmisor. Aquí, datos en paralelo provenientes de la izquierda llegan al transmisor por las líneas del bus de datos. A la derecha salen los bits de datos en serie. La parte de abajo del diagrama muestra la sección del receptor. Los bits de entrada llegan al receptor desde la derecha y son convertidos en datos en paralelo y luego enviados por las líneas del bus de datos de la derecha.

Tanto la sección del receptor como la del transmisor utilizan las mismas líneas de datos.

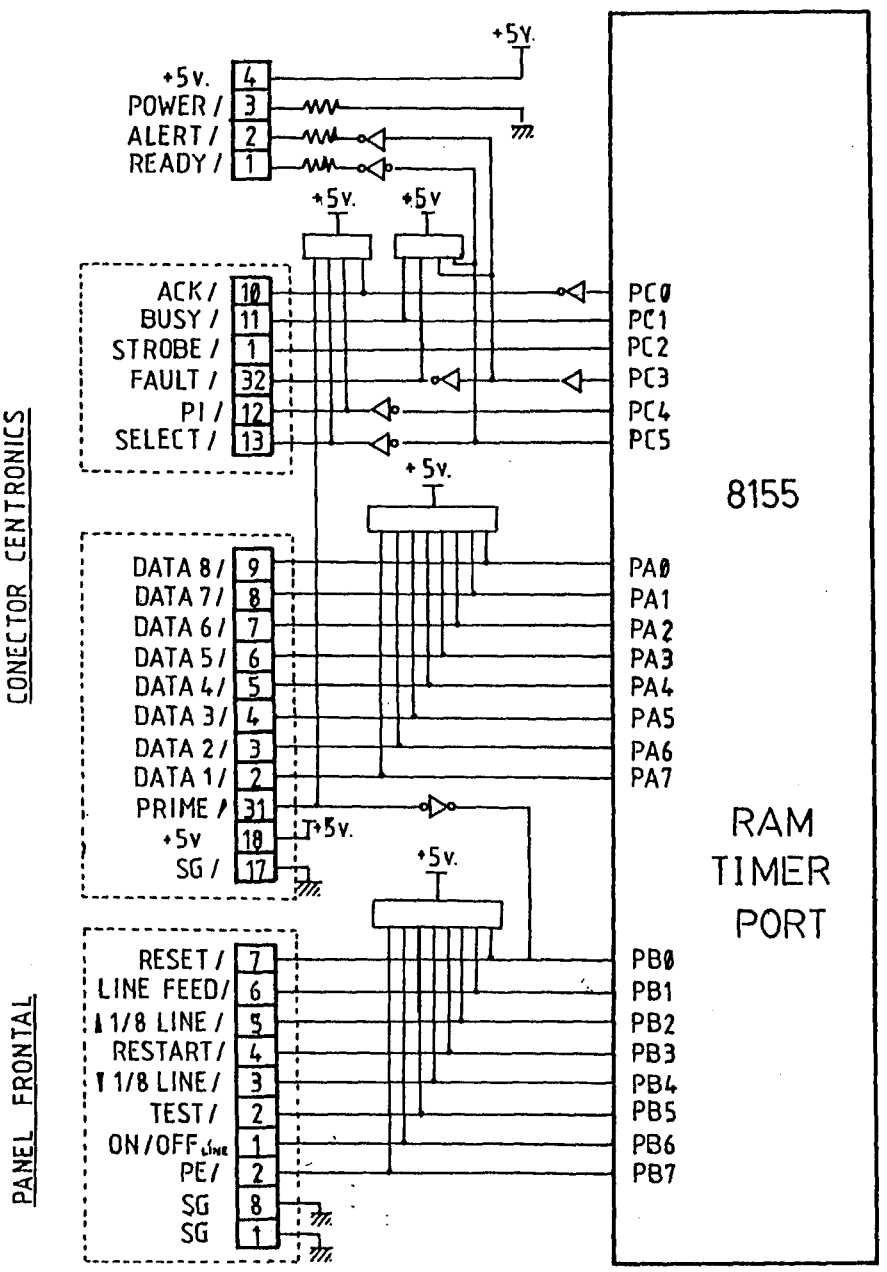
APENDICE I :

Comunicaciones

Centronics.

CONFIGURACION PARALELO CENTRONICS.-

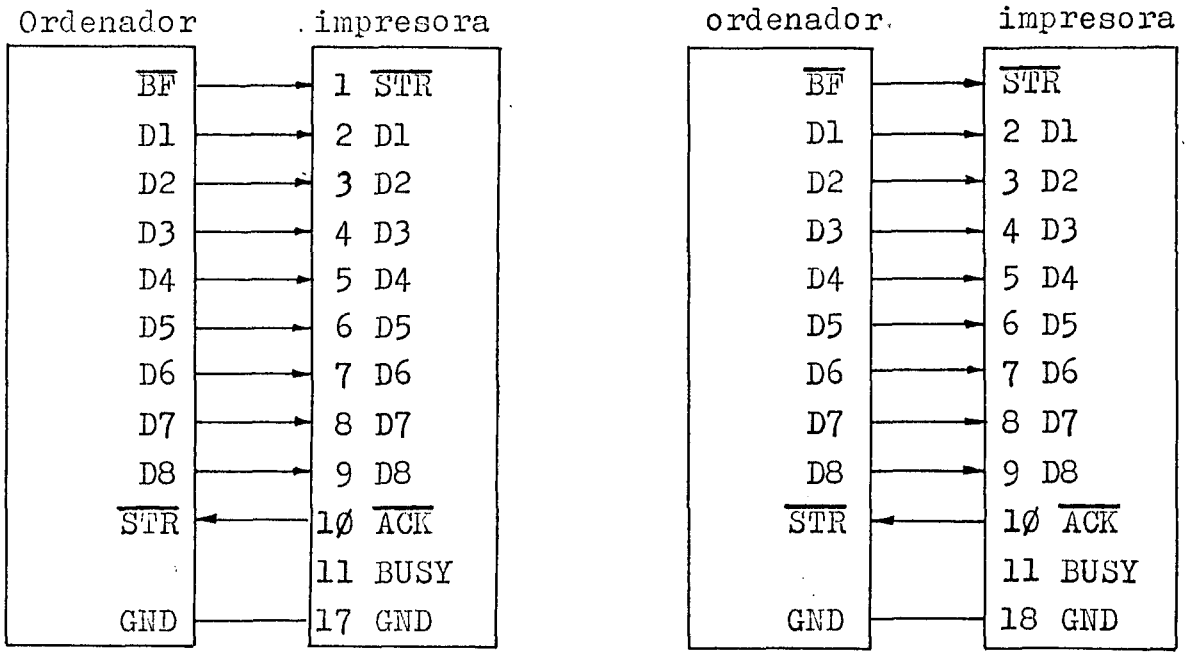
Debido a la escasez de información de la que se parte, hablaremos en esta sección de la configuración Centronics de la impresora TRS-8Ø. De ella es el siguiente montaje:



Configuración del 8155 en la impresora TRS-80 como entrada de datos paralelo CENTRONICS. (Strobed input)

Centronics, es el modelo de transmisión paralelo más corriente. Debido a que tiene el inconveniente de la unidireccionalidad su uso está casi restringido a conexiones con impresoras locales donde la velocidad de transmisión es mucho mayor que la serie ó RS-232c.

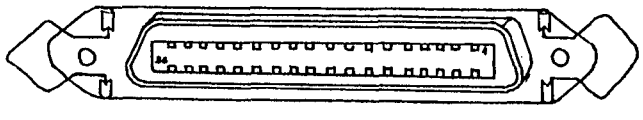
A continuación se muestran las patillas más importantes del modelo Centronics y usadas en el SDC-85.



(conector de 36 pins) (conector DB-25)

Conectores.-

Para esta comunicación se suele emplear dos tipos de conectores: el DB-25, el mismo que para la transmisión serie y el Centronics de 36 patillas.



Conector CENTRONICS hembra, 36 pins.

Del esquema de la impresora y del funcionamiento del 8155 se deduce el acoplamiento entre el ordenador y la impresora.

Funciones de las patillas.-

1. $\overline{\text{STR}}$ (strobe): entrada de propósito general activa a nivel bajo. Con esta patilla el ordenador indica a la impresora que los bits D1-D8 (2-9) son válidos y listos para ser imprimidos.

2 - 9. D1 - D8 : (data bits). Entrada de propósito general de caracteres de 8 bits.

1Ø. $\overline{\text{ACK}}$ (acknowledge=reconocimiento): salida de propósito especial, activa a nivel bajo. Con esta patilla la impresora indica al ordenador que ha recibido el dato.

11. BUSY (ocupado): salida de proósito especial activa a nivel alto. Con esta patilla la impresora indica al ordenador que está ocupada y no puede aceptar más datos aún.

17. GND (tierra): punto de referencia de voltajes.

Pueden usarse más patillas pero no afectan a la comunicación.

Hardware del acoplamiento con el 8155.-

Como es lógico, la conexión ha de hacerse por hardware, es decir, que sea un integrado periférico, el 8155, quien se encargue, en este caso, de observar el acoplamiento (activación de las patillas) y no el microprocesador con su programa.

Pero aquí surge un problema. Desde el punto de vista de transmisión, el 8155 no tiene medios por sí solo (sin lógica exterior) de establecer el acoplamiento. Sin embargo desde el punto de vista del receptor sí, como lo demuestra el móntaje anterior de la impresora. Esto se debe a lo siguiente:

- el 8155, en el modo de salida (en el esquema de la impresora es de entrada) dispone de una entrada especial, $\overline{\text{STR}}$ activa a nivel bajo y una salida especial, BF (buffer full). $\overline{\text{STR}}$ al ser activa a nivel bajo, no puede conectarse directamente a $\overline{\text{ACK}}$ del tranmisor ó a través de una puerta NOT a BUSY. Si así fuera, cuando la impresora estuviera ocupada (BUSY = "1", $\overline{\text{ACK}}$ = "Ø"), cosa que puede durar varias décimas de segundo, estaría indicando al $\overline{\text{STR}}$ del ordenador que se ha recibido el dato aún cuando no lo hubiera hecho.

Ante este problema, se diseña el diagrama de tiempos teórico que debe cumplir el nuevo circuito:

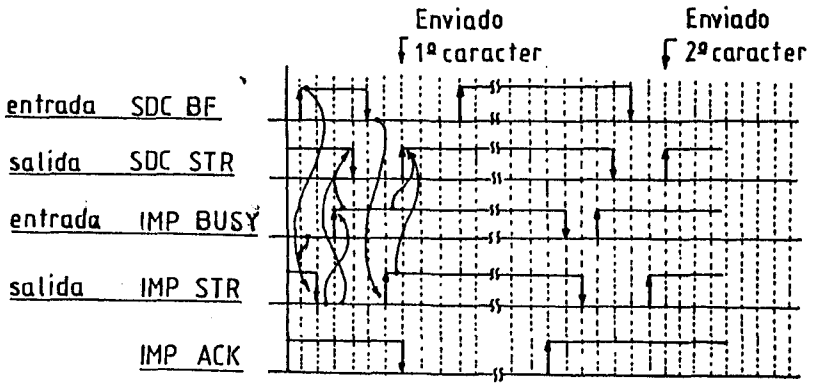


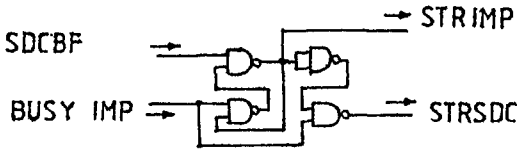
Diagrama de tiempos del control de salida CENTRONICS.

$$STRIMP = \overline{BF \cdot \overline{BUSY} + \overline{STRIMP} \cdot BF} = \overline{BF \cdot (\overline{BUSY} + \overline{STRIMP})} =$$

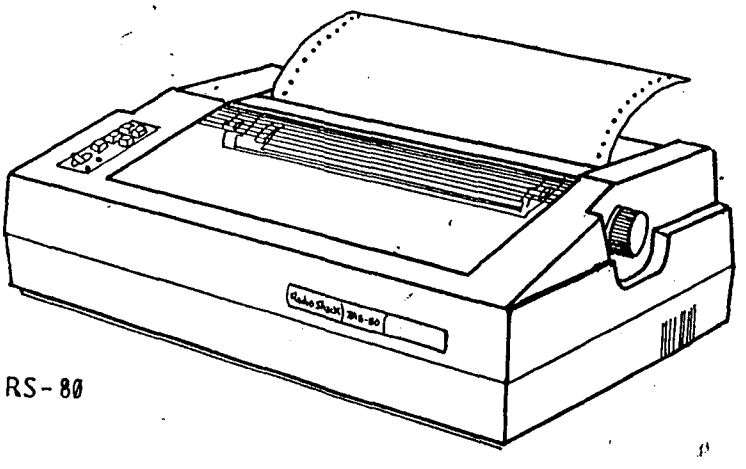
$$= \overline{BF \cdot (\overline{BUSY \cdot STRIMP})}$$

$$STRSDC = \overline{\overline{STRIMP} \cdot BUSY}$$

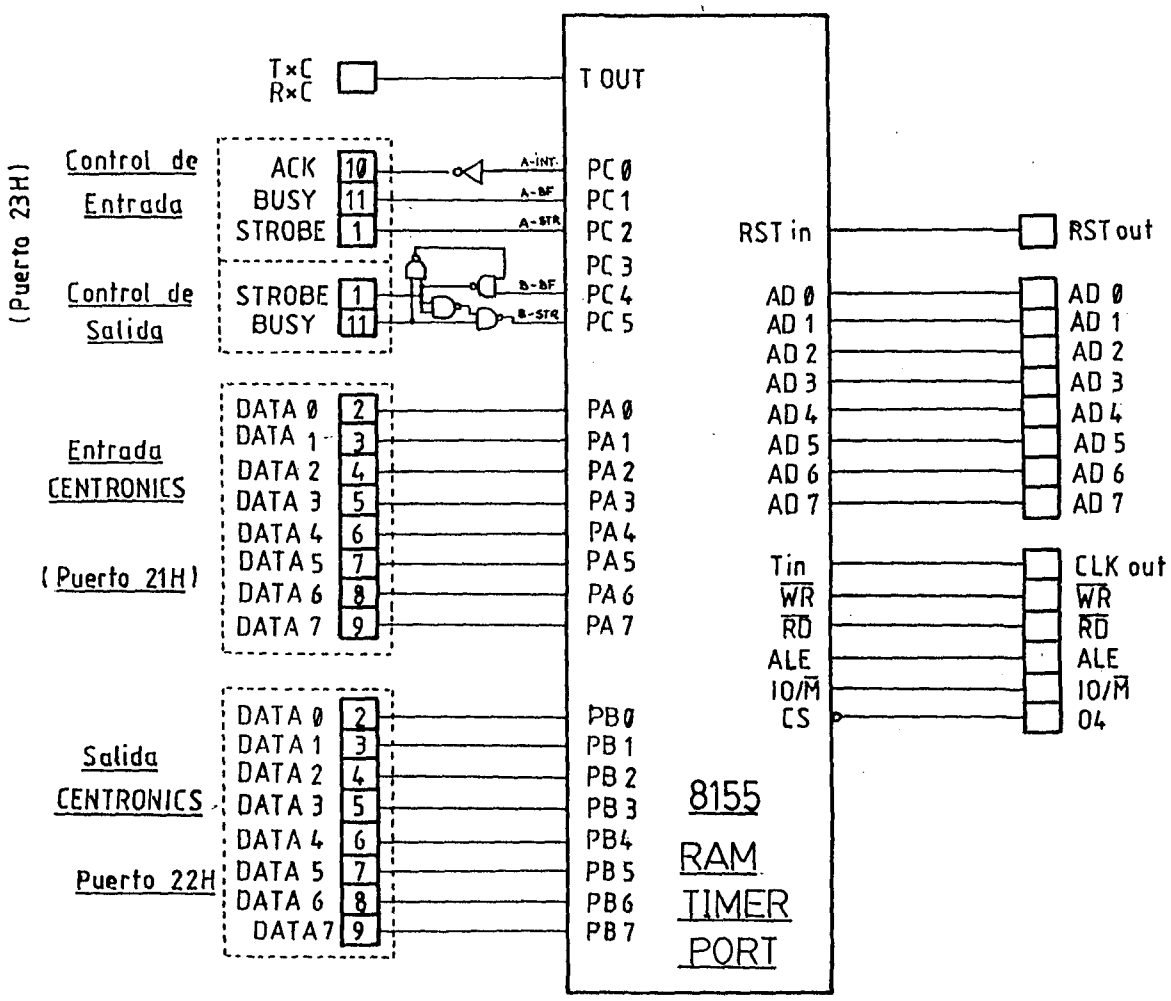
El resultado del análisis es el siguiente montaje:



Para finalizar, en la página siguiente se muestra la configuración que tiene el 8155 dentro del montaje del SDC-85 como controlador de tanto entrada como salida de datos Centronics (paralelo).



Impresora TRS-80

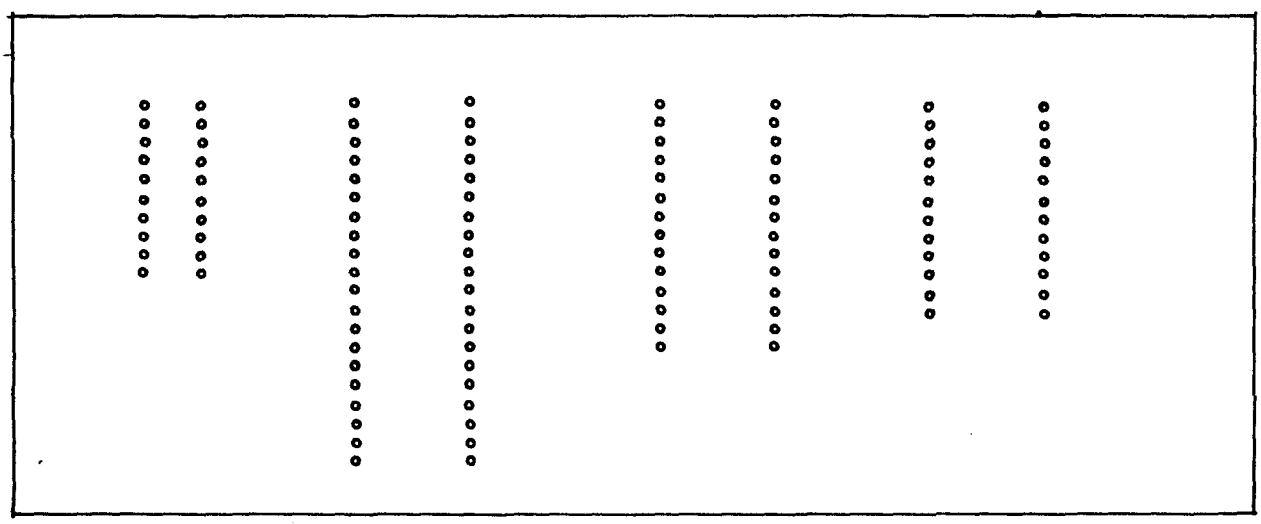


Configuración entrada / salida CENTRONICS del SDC-85.

APENDICE J:

Smartwork.

(Diseño por ordenador de circuito impreso.)



DISEÑO CON SMARTWORK.-

El Smartwork es un programa de Wintek Personal Compu-- ter Software para el diseño de circuitos impresos.

Para su uso, en un ordenador compatible IBM PC, es necesario disponer de los siguientes ficheros:

NOLOK.EXE
EDIT .EXE (edición)
PLOT .EXE (impresión)
SMART.STA
DEMO .PCB (demostración)

Si se dispone de "ratón" (Microsoft Mouse), para instalar su software teclear:

A>MOUSE

Para ello es necesario disponer del fichero MOUSE.COM en el diskete DOS.

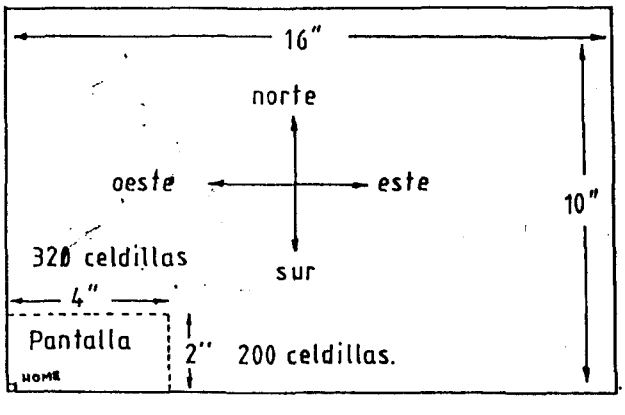
Inicialización: para editar un fichero de circuito impreso teclear:

A>NOLOK
A>EDIT

Entonces aparecerá en la esquina inferior izquierda un agujero en forma de "□" ó "o".

Especificaciones: el tamaño máximo de la placa que el Smartwork puede manejar es de 1Ø"x16" (25'4cm.x4Ø'64cm.). De él aparece en pantalla 2" x 4" (5'Ø8cm.x 1Ø'16cm).

Para minimizar defectos, el Smartwork fuerza ciertos espaciamientos mínimos de traza y anchos de pistas fijos



Tamaño relativo pantalla/total.

ancho de línea fina = 12 mils. (Ø.Ø12")
ancho de línea gruesa = 5Ø mils. (Ø.Ø5Ø")
diámetro agujero = 62 mils. (Ø.Ø62")
espaciamiento mínimo = 19 mils. (Ø.Ø19")

$1mil = 1Ø^{-3}$ pulgada = $2'54\ 1Ø^{-3}$ cm.

Hardware requerido.-

- IBM Personal Computer corriendo PC DOS versión (2.Ø) ó posteriores.
- 192 Kbytes de memoria central (RAM).

Comandos.-

Para introducir comandos pulsar <Enter> ó <Return>. Aparecerá entonces en la parte inferior izaquierda de la pantalla: >COMAND>

Con <ESC> se puede abortar la introducción del comando y con <Backspace> se pueden rectificar caracteres erróneos.

Un error en la sintaxis producirá un mensaje como:

UNKNOWN COMMAND

COMMAND> QUIT: salida del Smartwork.

Discard current worksapace (y/n) ?

Con este comando se sale del programa Smartwork al sistema operativo MS-DOS sin guardar la edición.

COMMAND> LOAD B:demo.pcb : Cargar fichero

Discard current workspace (y/n) ?

Con este comando se carga el fichero a editar y aparece entonces en pantalla la parte inferior izquierda de la cara inferior (lower) del fichero.

COMMAND> SAVE B: sdc85.kit : Guarda fichero.

Guarda en disco el fichero editado.

COMMAND> CLEAR : borra la edición actual.

COMMAND> SIP d n : dibuja hilera de agujeros.

dirección de la hilera ← (n(orte), s(ur), e(ste), w(oeste)) → número de agujeros

COMMAND> DIP d n :dibuja doble hilera de agujeros.

dirección ← (n(orte), s(ur), e(ste), w(oeste)) → número de agujeros

COMMAND> CLEAVE d : desplaza el dibujo

dirección ← (n(orte), s(ur), e(ste), w(oeste)) →

Este comando espacía ó desplaza (amplía) en una dirección (norte, sur, este y oeste) el dibujo de la placa a partir de la posición del cursor. Contrariamente, no hay forma de volver a la placa original, es decir, no hay forma de eliminar (reducir) en una dirección la placa.

Teclas de función:

Tecla	Función
F1	: marcar ruta de pista.
F2	: borrar pista.
F3	: situar agujero por dos caras (●)
F4	: borrar agujero.
F5	: ensanchar pista (▬▬)
F6	: adelgazar pista (——)
F7	: situar celdilla gruesa (■)
F8	: repetir ruta de pista.
ALT-F1	: establece una de dos configuraciones: 1. simple cara, alta resolución, pantalla B/N. 2. doble cara, media resolución, pantalla color
ALT-F2	: selecciona la intensidad.
ALT-F3	: establece uno de dos juegos de colores. 1. rojo, verde, amarillo. 2. magenta, cian, blanco.
ALT-F4	: selecciona el color background azul y negro.
ALT-F5	: selecciona la opción de 2 ó 3 colores.
ALT-F6	: establece el color activo.

	Grupo 1	Grupo 2	Grupo 3	Grupo 4
cara activa :	rojo	verde	magenta	cian
cara opuesta :	verde	rojo	cian	magenta
puntos coincid.:	amarillo	amarillo	blanco	blanco

ALT-F7	: establece el tamaño de ventana; visualiza los 10" x 16" de la placa entera en lugar de los 2" x 4" de la pantalla normal.
Backspace	: borra el último trozo de pista trazado.
CTRL-Break	: interrumpe la ruta.
PG UP/DN	: cambio de cara activa (superior/inferior)
Teclas de flecha	: ↑,↓,→,←
Shift + tecla de flecha	: avance rápido.
HOME	: situar el cursor en la esquina inferior izq.

Impresión por impresora y plotter.-

A>PLOT

Este fichero ejecutable está preparado para funcionar en una impresora EPSON FX-1ØØ ó MX-1ØØ y en un plotter DMP-41.

Input file containing the board layout :

Output device or filename :lptl:

(coml:, com2:, aux:, prn: ó lpt2:)

Primero espera a que se introduzca el fichero a imprimir y luego el periférico de salida (por defecto es lptl:).

Se puede elegir entre 2 escalas de impresión: 1x ó normal y 2x ó doble (de mayor definición). En cuanto al grosor de las puntas de la plumilla del plotter hay dos opciones:

Ø.6 mm = 12 mil. línea fina.
19 mil. espaciamiento.

Ø.7 mm = 14 mil. línea gruesa
17 mil. espaciamiento

APENDICE K:

Componentes del SDC-85.

8085A/8085A-2
SINGLE CHIP 8-BIT N-CHANNEL MICROPROCESSORS

- Single +5V Power Supply
- 100% Software Compatible with 8080A
- 1.3 μs Instruction Cycle (8085A); 0.8 μs (8085A-2)
- On-Chip Clock Generator (with External Crystal, LC or RC Network)
- On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control
- Four Vectored Interrupt Inputs (One is non-Maskable) Plus an 8080A-compatible interrupt
- Serial In/Serial Out Port
- Decimal, Binary and Double Precision Arithmetic
- Direct Addressing Capability to 64k Bytes of Memory

The Intel® 8085A is a complete 8 bit parallel Central Processing Unit (CPU). Its instruction set is 100% software compatible with the 8080A microprocessor, and it is designed to improve the present 8080A's performance by higher system speed. Its high level of system integration allows a minimum system of three IC's (8085A CPU, 8156 RAM/IO, and 8355/8755A ROM/PROM/IO) while maintaining total system expandability. The 8085A-2 is a faster version of the 8085A.

The 8085A incorporates all of the features that the 8224 clock generator and 8228 system controller provided for the 8080A, thereby offering a high level of system integration.

The 8085A uses a multiplexed data bus. The address is split between the 8 bit address bus and the 8 bit data bus. The on-chip address latches of 8155/8156/8355/8755A memory products allow a direct interface with the 8085A.

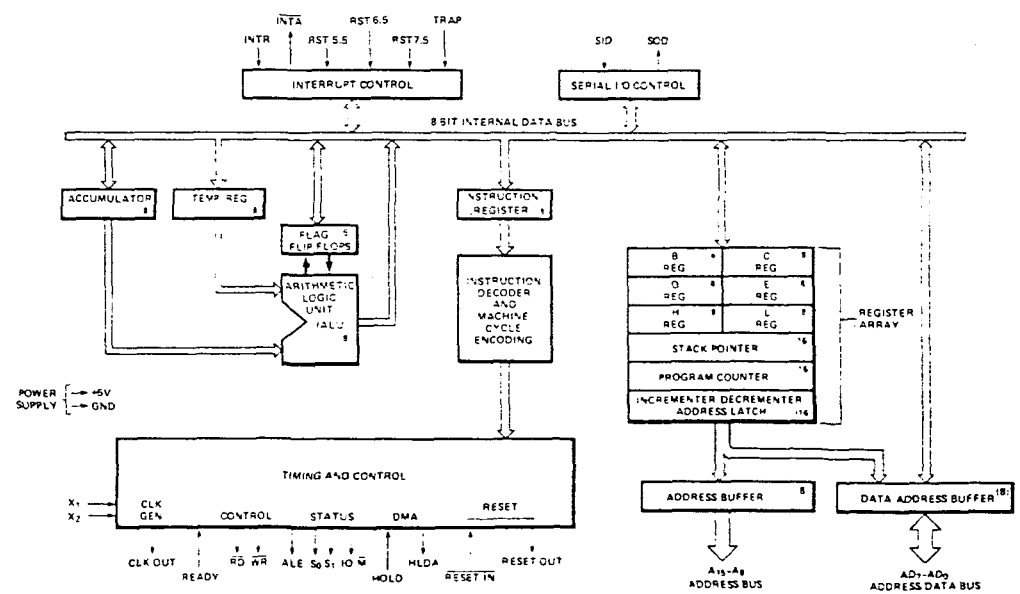


Figure 1. 8085A CPU Functional Block Diagram

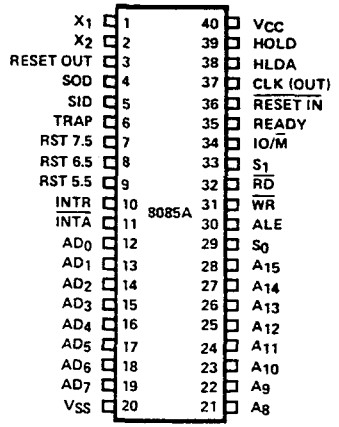


Figure 2. 8085A Pinout Diagram

8085A FUNCTIONAL PIN DEFINITION

The following describes the function of each pin:

Symbol	Function
A_8–A_{15} (Output, 3-state)	Address Bus: The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET.
AD_{0-7} (Input/Output, 3-state)	Multiplexed Address/Data Bus: Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state of a machine cycle. It then becomes the data bus during the second and third clock cycles.
ALE (Output)	Address Latch Enable: It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.
S_0, S_1, and IO/$\overline{M}$ (Output)	Machine cycle status:

IO/$\overline{M}$	S_1	S_0	Status
0	0	1	Memory write
0	1	0	Memory read
1	0	1	I/O write
1	1	0	I/O read
0	1	1	Opcode fetch
1	1	1	Interrupt Acknowledge
*	0	0	Halt
*	X	X	Hold
*	X	X	Reset

* = 3-state (high impedance)
X = unspecified

Symbol	Function
	S_1 can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$, S_0 and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.
$\overline{RD}$ (Output, 3-state)	READ control: A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer, 3-stated during Hold and Halt modes and during RESET.
$\overline{WR}$ (Output, 3-state)	WRITE control: A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET.
READY (Input)	If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the cpu will wait an integral number of clock cycles for READY to go high before completing the read or write cycle.
HOLD (Input)	HOLD indicates that another master is requesting the use of the address and data buses. The cpu, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data, $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.
HLDA (Output)	HOLD ACKNOWLEDGE: Indicates that the cpu has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The cpu takes the bus one half clock cycle after HLDA goes low.
INTR (Input)	INTERRUPT REQUEST: is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

8085A FUNCTIONAL PIN DESCRIPTION (Continued)

Symbol	Function
$\overline{INTA}$ (Output)	INTERRUPT ACKNOWLEDGE: Is used instead of (and has the same timing as) $\overline{RD}$ during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.
RST 5.5 RST 6.5 RST 7.5 (Inputs)	RESTART INTERRUPTS: These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupts is ordered as shown in Table 1. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.
TRAP (Input)	Trap interrupt is a nonmaskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. See Table 1.
$\overline{RESET\ IN}$ (Input)	Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. $\overline{RESET\ IN}$ is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay. The cpu is held in the reset condition as long as $\overline{RESET\ IN}$ is applied.
RESET OUT (Output)	Indicates cpu is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
X_1, X_2 (Input)	X_1 and X_2 are connected to a crystal, LC, or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
CLK (Output)	Clock Output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.
SID (Input)	Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
SOD (Output)	Serial output data line. The output SOD is set or reset as specified by the SIM instruction.
V_{CC}	+5 volt supply.
V_{SS}	Ground Reference.

TABLE 1. INTERRUPT PRIORITY, RESTART ADDRESS, AND SENSITIVITY

Name	Priority	Address Branched To (1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled
RST 7.5	2	3CH	Rising edge latched
RST 6.5	3	34H	High level until sampled.
RST 5.5	4	2CH	High level until sampled.
INTR	5	See Note 2	High level until sampled.

NOTES:
1. The processor pushes the PC on the stack before branching to the indicated address.
2. The address branched to depends on the instruction provided to the cpu when the interrupt is acknowledged.

FUNCTIONAL DESCRIPTION

The 8085A is a complete 8-bit parallel central processor. It is designed with N-channel depletion loads and requires a single +5 volt supply. Its basic clock speed is 3 MHz (8085A) or 5 MHz (8085A-2), thus improving on the present 8080A's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The cpu (8085A), a RAM/IO (8156), and a ROM or EPROM/IO chip (8355 or 8755A).

The 8085A has twelve addressable 8-bit registers. Four of them can function only as two 16-bit register pairs. Six others can be used interchangeably as 8-bit registers or as 16-bit register pairs. The 8085A register set is as follows:

Mnemonic	Register	Contents
ACC or A	Accumulator	8 bits
PC	Program Counter	16-bit address
BC,DE,HL	General-Purpose Registers; data pointer (HL)	8 bits x 6 or 16 bits x 3
SP	Stack Pointer	16-bit address
Flags or F	Flag Register	5 flags (8-bit space)

The 8085A uses a multiplexed Data Bus. The address is split between the higher 8-bit Address Bus and the lower 8-bit Address/Data Bus. During the first T state (clock cycle) of a machine cycle the low order address is sent out on the Address/Data bus. These lower 8 bits may be latched externally by the Address Latch Enable signal (ALE). During the rest of the machine cycle the data bus is used for memory or I/O data.

The 8085A provides $\overline{RD}$, $\overline{WR}$, S_0, S_1, and IO/$\overline{M}$ signals for bus control. An Interrupt Acknowledge signal ($\overline{INTA}$) is also provided. HOLD and all Interrupts are synchronized with the processor's internal clock. The 8085A also provides Serial Input Data (SID) and Serial Output Data (SOD) lines for simple serial interface.

In addition to these features, the 8085A has three maskable, vector interrupt pins and one nonmaskable TRAP interrupt.

INTERRUPT AND SERIAL I/O

The 8085A has 5 interrupt inputs: INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080A INT. Each of the three RESTART inputs, 5.5, 6.5, and 7.5, has a programmable mask. TRAP is also a RESTART interrupt but it is nonmaskable.

The three maskable interrupts cause the internal execution of RESTART (saving the program counter in the stack and branching to the RESTART address if the interrupts are enabled and if the interrupt mask is not set. The nonmaskable TRAP causes the internal execution of a RESTART vector independent of the state of the interrupt enable or masks. (See Table 1.

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are *high level-sensitive* like INTR and INT on the 8080 and are recognized with the same timing as INTR. RST 7.5 is *rising edge-sensitive*.

For RST 7.5, only a pulse is required to set an internal flip-flop which generates the internal interrupt request. (See Section 5.2.7. The RST 7.5 request flip-flop remains set until the request is serviced. Then it is reset automatically. This flip-flop may also be reset by using the SIM instruction or by issuing a $\overline{RESET\ IN}$ to the 8085A. The RST 7.5 internal flip-flop will be set by a pulse on the RST 7.5 pin even when the RST 7.5 interrupt is masked out.

The status of the three RST interrupt masks can only be affected by the SIM instruction and $\overline{RESET\ IN}$. (See SIM, Chapter 5.

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP — highest priority, RST 7.5, RST 6.5, RST 5.5, INTR — lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt an RST 7.5 routine if the interrupts are re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic events such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both *edge and level sensitive*. The TRAP input must go high and remain high until it is acknowledged. It will not be recognized again until it goes low, then high again. This avoids any false triggering due to noise or logic glitches. Figure 3 illustrates the TRAP interrupt request circuitry within the 8085A. Note that the servicing of any interrupt (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) disables all future interrupts (except TRAPs until an EI instruction is executed.

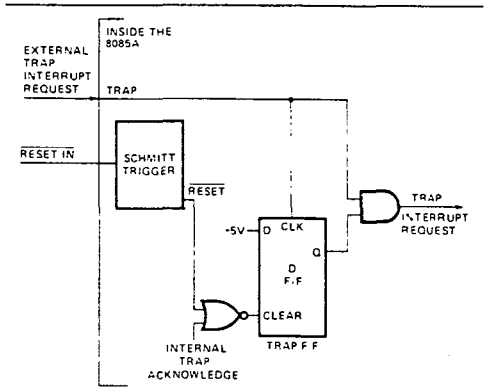


Figure 3. TRAP and $\overline{RESET\ IN}$ Circuit

The TRAP interrupt is special in that it disables interrupts, but preserves the previous interrupt enable status. Performing the first RIM instruction following a TRAP interrupt allows you to determine whether interrupts were enabled or disabled prior to the TRAP. All subsequent RIM instructions provide current interrupt enable status. Performing a RIM instruction following INTR, or RST 5.5–7.5 will provide current Interrupt Enable status, revealing that Interrupts are disabled. See the description of the RIM instruction in Chapter 5.

The serial I/O system is also controlled by the RIM and SIM instructions. SID is read by RIM, and SIM sets the SOD data.

DRIVING THE X_1 AND X_2 INPUTS

You may drive the clock inputs of the 8085A or 8085A-2 with a crystal, an LC tuned circuit, an RC network, or an external clock source. The driving frequency must be at least 1 MHz, and must be twice the desired internal clock frequency; hence, the 8085A is operated with a 6 MHz crystal (for 3 MHz clock), and the 8085A-2 can be operated with a 10 MHz crystal (for 5 MHz clock). If a crystal is used, it must have the following characteristics:

Parallel resonance at twice the clock frequency desired
C_L (load capacitance) ≤ 30 pf
C_S (shunt capacitance) ≤ 7 pf
R_S (equivalent shunt resistance) ≤ 75 Ohms
Drive level: 10 mW
Frequency tolerance: ±.005% (suggested)

Note the use of the 20pF capacitor between X_2 and ground. This capacitor is required with crystal frequencies below 4 MHz to assure oscillator startup at the correct frequency. A parallel-resonant LC circuit may be used as the frequency-determining network for the 8085A, providing that its frequency tolerance of approximately ±10% is acceptable. The components are chosen from the formula:

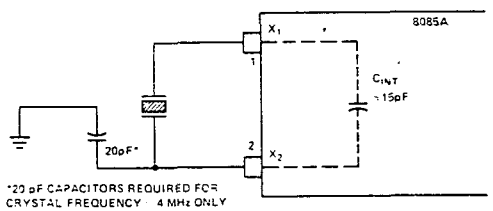
$$f = \frac{1}{2\pi\sqrt{L\,C_{ext} + C_{int}}}$$

To minimize variations in frequency, it is recommended that you choose a value for C_{ext} that is at least twice that of C_{int}, or 30 pF. The use of an LC circuit is not recommended for frequencies higher than approximately 5 MHz.

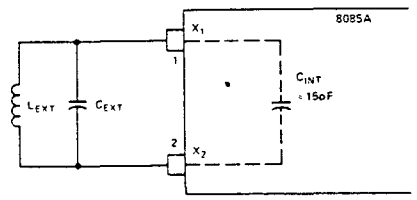
An RC circuit may be used as the frequency-determining network for the 8085A if maintaining a precise clock frequency is of no importance. Variations in the on-chip timing generation can cause a wide variation in frequency when using the RC mode. Its advantage is its low component cost. The driving frequency generated by the circuit shown is approximately 3 MHz. It is not recommended that frequencies greatly higher or lower than this be attempted.

Figure 4 shows the recommended clock driver circuits. Note in D and E that pullup resistors are required to assure that the high level voltage of the input is at least 4 V.

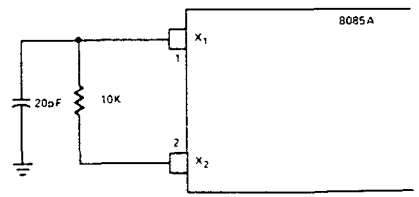
For driving frequencies up to and including 6 MHz you may supply the driving signal to X_1 and leave X_2 open-circuited (Figue 4D). If the driving frequency is from 6 MHz to 10 MHz, stability of the clock generator will be improved by driving both X_1 and X_2 with a push-pull source (Figure 4E). To prevent self-oscillation of the 8085A, be sure that X_2 is not coupled back to X_1 through the driving circuit.



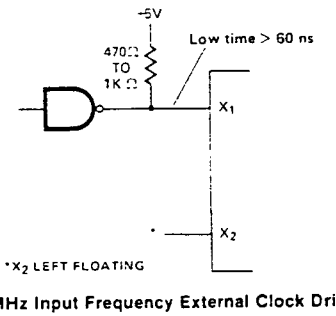
A. Quartz Crystal Clock Driver



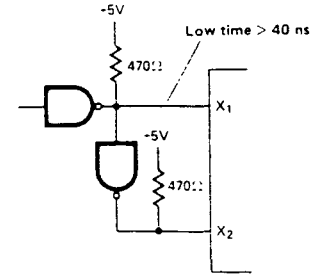
B. LC Tuned Circuit Clock Driver



C. RC Circuit Clock Driver



D. 1-6 MHz Input Frequency External Clock Driver Circuit



E. 1-10 MHz Input Frequency External Clock Driver Circuit

Figure 4. Clock Driver Circuits

GENERATING AN 8085A WAIT STATE

If your system requirements are such that slow memories or peripheral devices are being used, the circuit shown in Figure 5 may be used to insert one WAIT state in each 8085A machine cycle

The D flip-flops should be chosen so that

- CLK is rising edge-triggered
- CLEAR is low-level active.

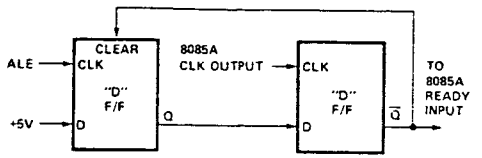


Figure 5. Generation of a Wait State for 8085A CPU

As in the 8080, the READY line is used to extend the read and write pulse lengths so that the 8085A can be used with slow memory. HOLD causes the cpu to relinquish the bus when it is through with it by floating the Address and Data Buses.

SYSTEM INTERFACE

The 8085A family includes memory components, which are directly compatible to the 8085A cpu. For example, a system consisting of the three chips, 8085A, 8156, and 8355 will have the following features:

- 2K Bytes ROM
- 256 Bytes RAM
- 1 Timer/Counter
- 4 8-bit I/O Ports
- 1 6-bit I/O Port
- 4 Interrupt Levels
- Serial In/Serial Out Ports

This minimum system, using the standard I/O technique is as shown in Figure 6.

In addition to standard I/O, the memory mapped I/O offers an efficient I/O addressing technique. With this technique, an area of memory address space is assigned for I/O address, thereby, using the memory address for I/O manipulation. Figure 7 shows the system configuration of Memory Mapped I/O using 8085A.

The 8085A cpu can also interface with the standard memory that does *not* have the multiplexed address/data bus. It will require a simple 8212 (8-bit latch) as shown in Figure 8.

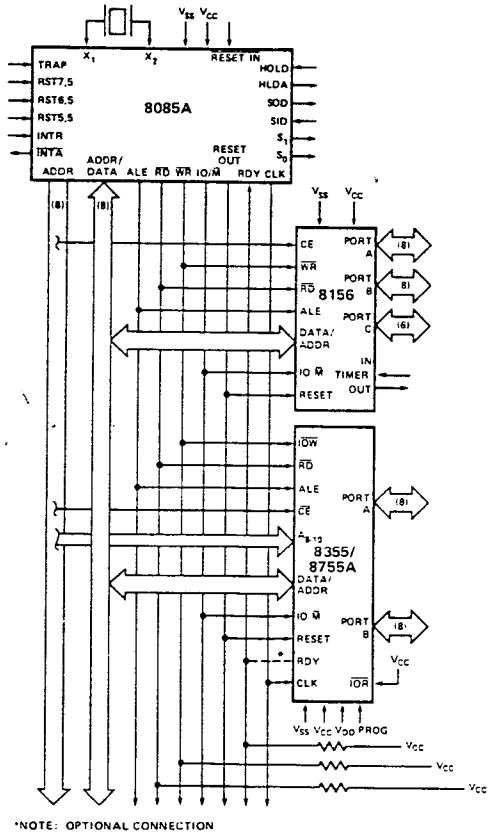


Figure 6. 8085A Minimum System (Standard I/O Technique)

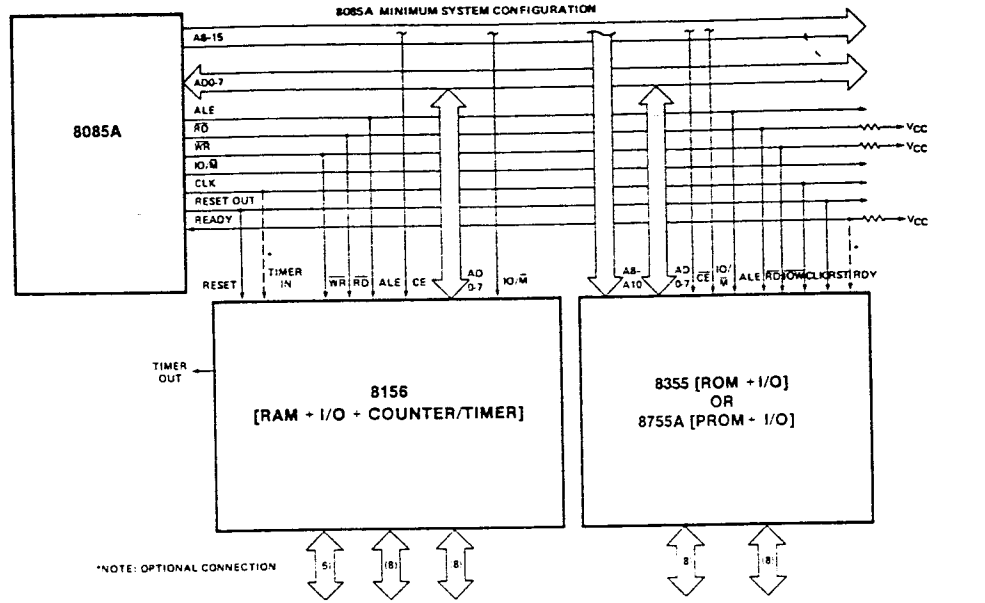


Figure 7. MCS-85™ Minimum System (Memory Mapped I/O)

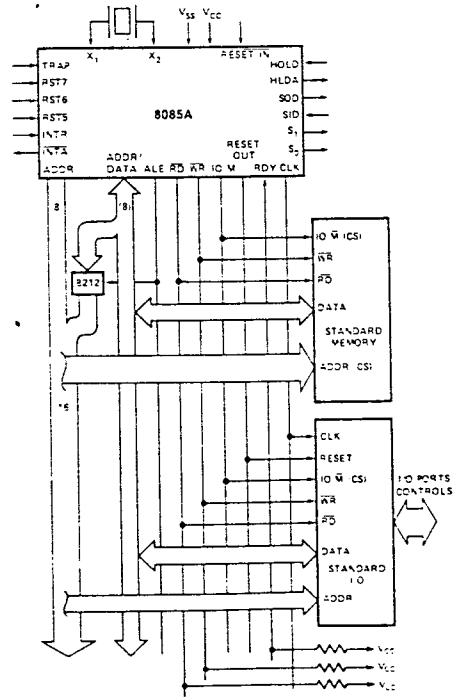


Figure 8. MCS-85™ System (Using Standard Memories)

BASIC SYSTEM TIMING

The 8085A has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. Figure 9 shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both the upper and lower half of the address.

There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines ($IO/\overline{M}$, S_1, S_0) and the three control signals ($\overline{RD}$, $\overline{WR}$, and $\overline{INTA}$). (See Table 2.) The status lines can be used as advanced controls (for device selection, for example), since they become active at the T_1 state, at the outset of each machine cycle. Control lines $\overline{RD}$ and $\overline{WR}$ become active later, at the time when the transfer of data is to take place, so are used as command lines.

A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of $\overline{READY}$ or HOLD inputs). Any T state must be one of ten possible states, shown in Table 3.

TABLE 2. 8085A MACHINE CYCLE CHART

MACHINE CYCLE		STATUS			CONTROL		
		IO/M	S1	S0	RD	WR	INTA
OPCODE FETCH	(OF)	0	1	1	0	1	1
MEMORY READ	(MR)	0	1	0	0	1	1
MEMORY WRITE	(MW)	0	0	1	1	0	1
I/O READ	(IOR)	1	1	0	0	1	1
I/O WRITE	(IOW)	1	0	1	1	0	1
ACKNOWLEDGE OF INTR	(INA)	1	1	1	1	1	0
BUS IDLE	(BI): DAD	0	1	0	1	1	1
	ACK. OF RST,TRAP	1	1	1	1	1	1
	HALT	TS	0	0	TS	TS	1

TABLE 3. 8085A MACHINE STATE CHART

Machine State	Status & Buses				Control			
	S1,S0	IO/M	A_8-A_{15}	AD_0-AD_7	RD,WR	INTA	ALE	
T_1	X	X	X	X	1	1	1*	
T_2	X	X	X	X	X	X	0	
T_{WAIT}	X	X	X	X	X	X	0	
T_3	X	X	X	X	X	X	0	
T_4	1	0†	X	TS	1	1	0	
T_5	1	0†	X	TS	1	1	0	
T_6	1	0†	X	TS	1	1	0	
T_{RESET}	X	TS	TS	TS	TS	1	0	
T_{HALT}	0	TS	TS	TS	TS	1	0	
T_{HOLD}	X	TS	TS	TS	TS	1	0	

0 = Logic "0" TS = High Impedance
1 = Logic "1" X = Unspecified

* ALE not generated during 2nd and 3rd machine cycles of DAD instruction
† IO/M = 1 during T_4-T_6 of INA machine cycle

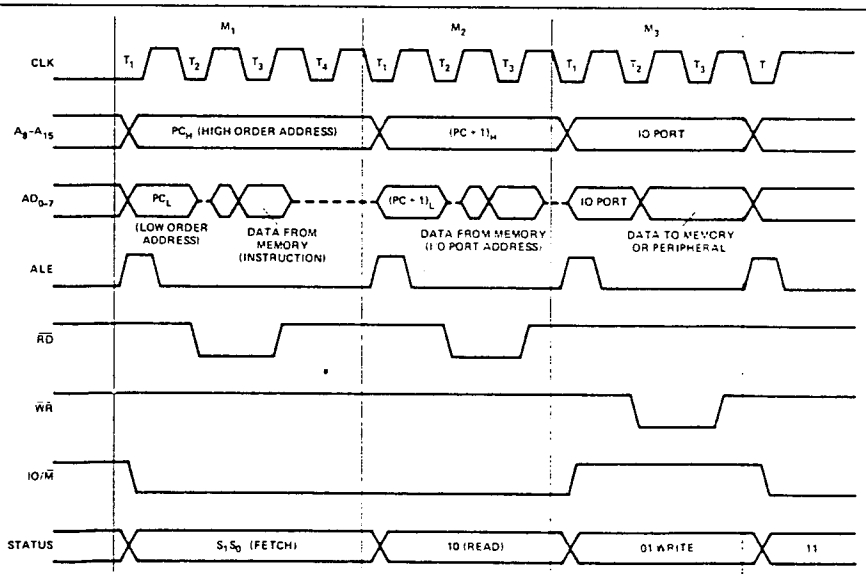


Figure 9. 8085A Basic System Timing

TABLE 4. ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . . −0.5V to +7V
Power Dissipation . . . . . 1.5 Watt

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

TABLE 5. D.C. CHARACTERISTICS

(T_A = 0°C to 70°C; V_{CC} = 5V ±5%; V_{SS} = 0V; unless otherwise specified)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400μA
I_{CC}	Power Supply Current		170	mA	
I_{IL}	Input Leakage		±10	μA	V_{in} = V_{CC}
I_{LO}	Output Leakage		±10	μA	0.45V ≤ V_{out} ≤ V_{CC}
V_{ILR}	Input Low Level, RESET	−0.5	+0.8	V	
V_{IHR}	Input High Level, RESET	2.4	V_{CC} +0.5	V	
V_{HY}	Hysteresis, RESET	0.25		V	

TABLE 6. A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$; $V_{CC} = 5V \pm 5\%$; $V_{SS} = 0V$

Symbol	Parameter	8085A[2]		8085A-2[2] (Preliminary)		Units
		Min.	Max.	Min.	Max.	
t_{CYC}	CLK Cycle Period	320	2000	200	2000	ns
t_1	CLK Low Time (Standard CLK Loading)	80		40		ns
t_2	CLK High Time (Standard CLK Loading)	120		70		ns
t_r, t_f	CLK Rise and Fall Time		30		30	ns
t_{XKR}	X_1 Rising to CLK Rising	30	120	30	100	ns
t_{XKF}	X_1 Rising to CLK Falling	30	150	30	110	ns
t_{AC}	A_{8-15} Valid to Leading Edge of Control[1]	270		115		ns
t_{ACL}	A_{0-7} Valid to Leading Edge of Control	240		115		ns
t_{AD}	A_{0-15} Valid to Valid Data In		575		350	ns
t_{AFR}	Address Float After Leading Edge of $\overline{READ}$ ($\overline{INTA}$)		0		0	ns
t_{AL}	A_{8-15} Valid Before Trailing Edge of ALE[1]	115		50		ns
t_{ALL}	A_{0-7} Valid Before Trailing Edge of ALE	90		50		ns
t_{ARY}	READY Valid from Address Valid		220		100	ns
t_{CA}	Address (A_{8-15}) Valid After Control	120		60		ns
t_{CC}	Width of Control Low ($\overline{RD}$, $\overline{WR}$, $\overline{INTA}$) Edge of ALE	400		230		ns
t_{CL}	Trailing Edge of Control to Leading Edge of ALE	50		25		ns
t_{DW}	Data Valid to Trailing Edge of $\overline{WRITE}$	420		230		ns
t_{HABE}	HLDA to Bus Enable		210		150	ns
t_{HABF}	Bus Float After HLDA		210		150	ns
t_{HACK}	HLDA Valid to Trailing Edge of CLK	110		40		ns
t_{HDH}	HOLD Hold Time	0		0		ns
t_{HDS}	HOLD Setup Time to Trailing Edge of CLK	170		120		ns
t_{INH}	INTR Hold Time	0		0		ns
t_{INS}	INTR, RST, and TRAP Setup Time to Falling Edge of CLK	160		150		ns
t_{LA}	Address Hold Time After ALE	100		50		ns
t_{LC}	Trailing Edge of ALE to Leading Edge of Control	130		60		ns
t_{LCK}	ALE Low During CLK High	100		50		ns
t_{LDR}	ALE to Valid Data During Read		460		270	ns
t_{LDW}	ALE to Valid Data During Write		200		120	ns
t_{LL}	ALE Width	140		80		ns
t_{LRY}	ALE to READY Stable		110		30	ns

Table 6. A.C. Characteristics (Cont.)

Symbol	Parameter	8085A[2]		8085A-2[2] (Preliminary)		Units
		Min.	Max.	Min.	Max.	
t_{RAE}	Trailing Edge of $\overline{READ}$ to Re-Enabling of Address	150		90		ns
t_{RD}	$\overline{READ}$ (or $\overline{INTA}$) to Valid Data		300		150	ns
t_{RV}	Control Trailing Edge to Leading Edge of Next Control	400		220		ns
t_{RDH}	Data Hold Time After $\overline{READ}$ $\overline{INTA}$[7]	0		0		ns
t_{RYH}	READY Hold Time	0		0		ns
t_{RYS}	READY Setup Time to Leading Edge of CLK	110		100		ns
t_{WD}	Data Valid After Trailing Edge of $\overline{WRITE}$	100		60		ns
t_{WDL}	LEADING Edge of WRITE to Data Valid		40		20	ns

Notes:
1. A_8-A_{15} address Specs apply to IO/$\overline{M}$, S_0, and S_1 except A_8-A_{15} are undefined during T_4-T_6 of OF cycle whereas IO/$\overline{M}$, S_0, and S_1 are stable.
2. Test conditions: $t_{CYC} = 320$ ns (8085A)/200 ns (8085A-2); $C_L = 150$ pF
3. For all output timing where $C_L = 150$ pF use the following correction factors:
   $25\text{ pF} \leq C_L < 150\text{ pF}$: -0.10 ns/pF
   $150\text{ pF} < C_L \leq 300\text{ pF}$: $+0.30$ ns/pF
4. Output timings are measured with purely capacitive load
5. All timings are measured at output voltage $V_L = 0.8V$, $V_H = 2.0V$, and 1.5V with 20 ns rise and fall time on inputs.
6. To calculate timing specifications at other values of t_{CYC} use Table 7
7. Data hold time is guaranteed under all loading conditions

Input Waveform for A.C. Tests:

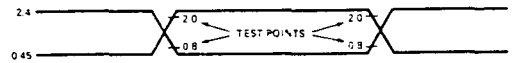


TABLE 7. BUS TIMING SPECIFICATION AS A T_{CYC} DEPENDENT

8085A				8085A-2 (Preliminary)			
t_{AL}	–	(1/2) T - 45	MIN	t_{AL}	–	(1/2) T - 50	MIN
t_{LA}	–	(1/2) T - 60	MIN	t_{LA}	–	(1/2) T - 50	MIN
t_{LL}	–	(1/2) T - 20	MIN	t_{LL}	–	(1/2) T - 20	MIN
t_{LCK}	–	(1/2) T - 60	MIN	t_{LCK}	–	(1/2) T - 50	MIN
t_{LC}	–	(1/2) T - 30	MIN	t_{LC}	–	(1/2) T - 40	MIN
t_{AD}	–	(5/2 + N) T - 225	MAX	t_{AD}	–	(5/2 + N) T - 150	MAX
t_{RD}	–	(3/2 + N) T - 180	MAX	t_{RD}	–	(3/2 + N) T - 150	MAX
t_{RAE}	–	(1/2) T - 10	MIN	t_{RAE}	–	(1/2) T - 10	MIN
t_{CA}	–	(1/2) T - 40	MIN	t_{CA}	–	(1/2) T - 40	MIN
t_{DW}	–	(3/2 + N) T - 60	MIN	t_{DW}	–	(3/2 + N) T - 70	MIN
t_{WD}	–	(1/2) T - 60	MIN	t_{WD}	–	(1/2) T - 40	MIN
t_{CC}	–	(3/2 + N) T - 80	MIN	t_{CC}	–	(3/2 + N) T - 70	MIN
t_{CL}	–	(1/2) T - 110	MIN	t_{CL}	–	(1/2) T - 75	MIN
t_{ARY}	–	(3/2) T - 260	MAX	t_{ARY}	–	(3/2) T - 200	MAX
t_{HACK}	–	(1/2) T - 50	MIN	t_{HACK}	–	(1/2) T - 60	MIN
t_{HABF}	–	(1/2) T + 50	MAX	t_{HABF}	–	(1/2) T + 50	MAX
t_{HABE}	–	(1/2) T + 50	MAX	t_{HABE}	–	(1/2) T + 50	MAX
t_{AC}	–	(2/2) T - 50	MIN	t_{AC}	–	(2/2) T - 85	MIN
t_1	–	(1/2) T - 80	MIN	t_1	–	(1/2) T - 60	MIN
t_2	–	(1/2) T - 40	MIN	t_2	–	(1/2) T - 30	MIN
t_{RV}	–	(3/2) T - 80	MIN	t_{RV}	–	(3/2) T - 80	MIN
t_{LDR}	–	(4/2) T - 180	MAX	t_{LDR}	–	(4/2) T - 130	MAX

NOTE: N is equal to the total WAIT states.
$T = t_{CYC}$

NOTE: N is equal to the total WAIT states.
$T = t_{CYC}$

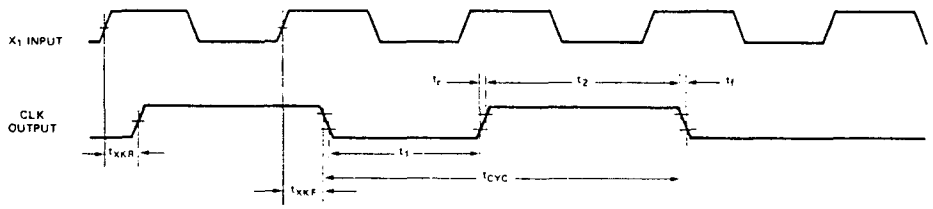
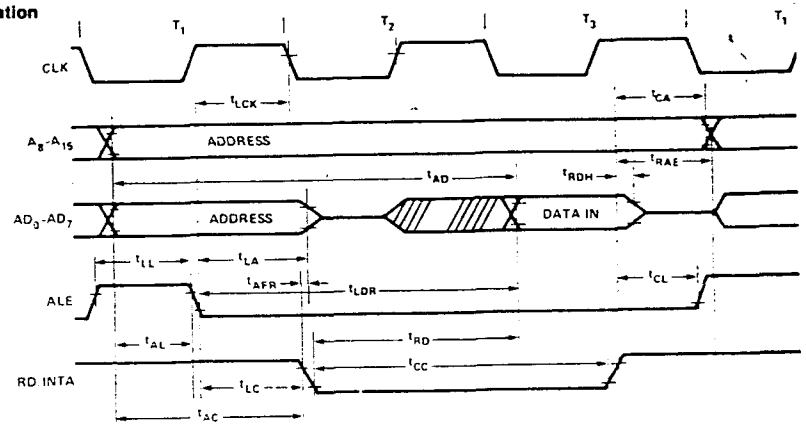
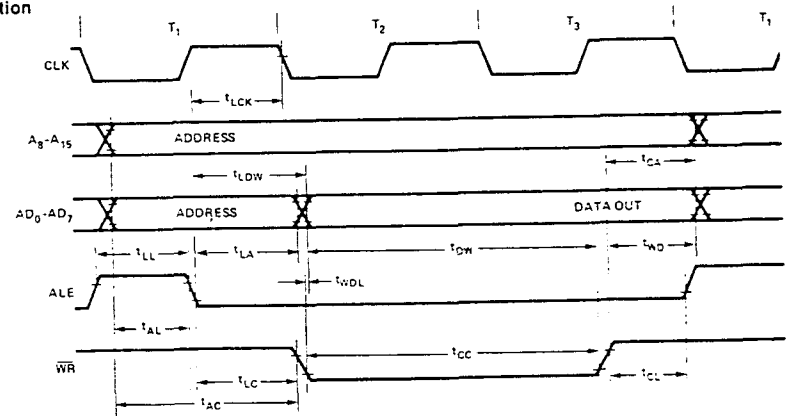


Figure 10. Clock Timing Waveform

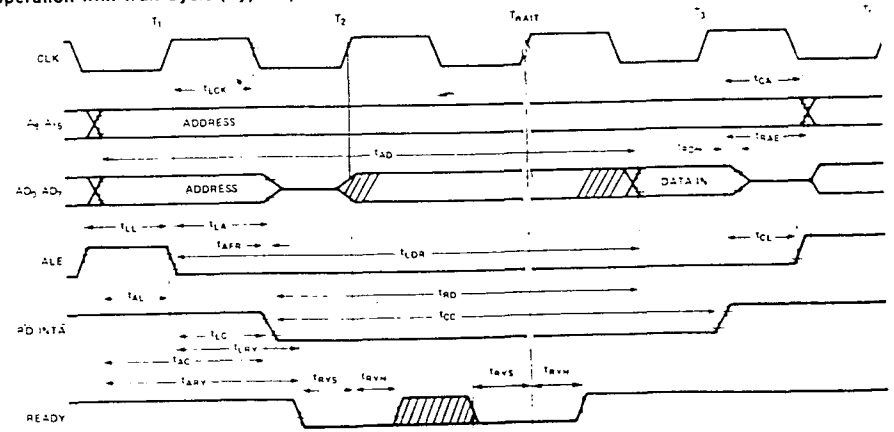
Read Operation



Write Operation



Read operation with Wait Cycle (Typical) — same READY timing applies to WRITE operation.



NOTE 1: READY must remain stable during setup and hold times

Figure 11. 8085A Bus Timing, With and Without Wait

Hold Operation

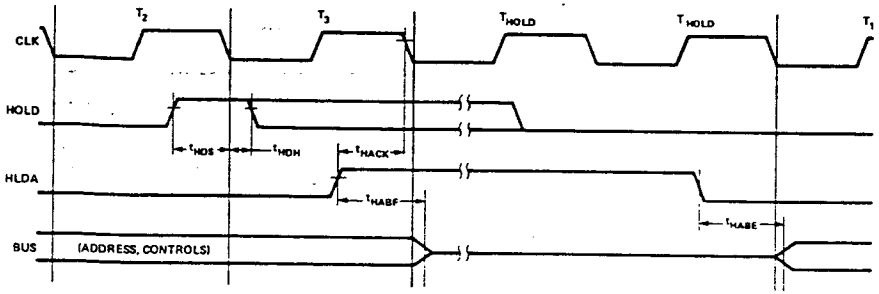


Figure 12. 8085A Hold Timing.

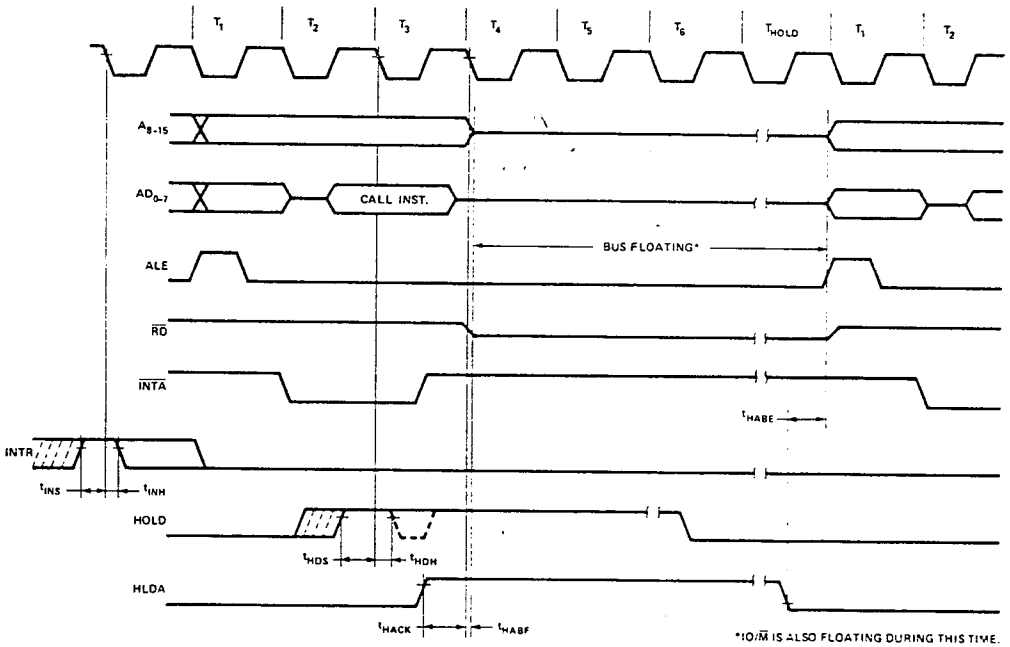


Figure 13. 8085A Interrupt and Hold Timing

8085A INSTRUCTION SET SUMMARY BY FUNCTIONAL GROUPING

Table 6-1

Mnemonic	Description	Instruction Code (1) D7	D6	D5	D4	D3	D2	D1	D0	Page
MOVE, LOAD, AND STORE										
MOV r1 r2	Move register to register	0	1	D	D	D	S	S	S	5-4
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	5-4
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	5-4
MVI r	Move immediate register	0	0	D	D	D	1	1	0	5-4
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	5-4
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	5-5
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	5-5
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	5-5
STAX B	Store A indirect	0	0	0	0	0	0	1	0	5-5
STAX D	Store A indirect	0	0	0	1	0	0	1	0	5-6
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	5-5
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	5-5
STA	Store A direct	0	0	1	1	1	0	1	0	5-5
LDA	Load A direct	0	0	1	1	1	0	1	0	5-5
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	5-5
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	5-5
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	5-6
STACK OPS										
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	5-15
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	5-15
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	5-15
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	5-15
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	5-15
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	5-15
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	5-15
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	5-15
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	5-16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5-16
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	5-5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5-9
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5-9
JUMP										
JMP	Jump unconditional	1	1	0	0	0	0	1	1	5-13
JC	Jump on carry	1	1	0	1	1	0	1	0	5-13
JNC	Jump on no carry	1	1	0	1	0	0	1	0	5-13
JZ	Jump on zero	1	1	0	0	1	0	1	0	5-13
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	5-13
JP	Jump on positive	1	1	1	1	0	0	1	0	5-13
JM	Jump on minus	1	1	1	1	1	0	1	0	5-13
JPE	Jump on parity even	1	1	1	0	1	0	1	0	5-13
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	5-13
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5-15
CALL										
CALL	Call unconditional	1	1	0	0	1	1	0	1	5-13
CC	Call on carry	1	1	0	1	1	1	0	0	5-14
CNC	Call on no carry	1	1	0	1	0	1	0	0	5-14
CZ	Call on zero	1	1	0	0	1	1	0	0	5-14
CNZ	Call on no zero	1	1	0	0	0	1	0	0	5-14
CP	Call on positive	1	1	1	1	0	1	0	0	5-14
CM	Call on minus	1	1	1	1	1	1	0	0	5-14
CPE	Call on parity even	1	1	1	0	1	1	0	0	5-14
CPO	Call on parity odd	1	1	1	0	0	1	0	0	5-14
RETURN										
RET	Return	1	1	0	0	1	0	0	1	5-14
RC	Return on carry	1	1	0	1	1	0	0	0	5-14
RNC	Return on no carry	1	1	0	1	0	0	0	0	5-14
RZ	Return on zero	1	1	0	0	1	0	0	0	5-14
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5-14
RP	Return on positive	1	1	1	1	0	0	0	0	5-14
RM	Return on minus	1	1	1	1	1	0	0	0	5-14
RPE	Return on parity even	1	1	1	0	1	0	0	0	5-14
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5-14
RESTART										
RST	Restart	1	1	A	A	A	1	1	1	5-14
INPUT/OUTPUT										
IN	Input	1	1	0	1	1	0	1	1	5-15
OUT	Output	1	1	0	1	0	0	1	1	5-15
INCREMENT AND DECREMENT										
INR r	Increment register	0	0	D	D	D	1	0	0	5-8
DCR r	Decrement register	0	0	D	D	D	1	0	1	5-8
INR M	Increment memory	0	0	1	1	0	1	0	0	5-8
DCR M	Decrement memory	0	0	1	1	0	1	0	1	5-8
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5-9
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5-9
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5-9
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5-9
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5-9
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5-9
ADD										
ADD r	Add register to A	1	0	0	0	0	S	S	S	5-6
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	5-7
ADD M	Add memory to A	1	0	0	0	0	1	1	0	5-6
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	5-7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	5-6
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	5-7
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	5-9
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	5-9
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	5-9
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	5-9
SUBTRACT										
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	5-7
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	5-7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	5-7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	5-8
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	5-7

8085A INSTRUCTION SET SUMMARY (Cont'd)
Table 6-1

Mnemonic	Description	Instruction Code (1) D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Page
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	5-8
LOGICAL										
ANA r	And register with A	1	0	1	0	0	S	S	S	5-9
XRA r	Exclusive OR register with A	1	0	1	0	1	S	S	S	5-10
ORA r	OR register with A	1	0	1	1	0	S	S	S	5-10
CMP r	Compare register with A	1	0	1	1	1	S	S	S	5-11
ANA M	And memory with A	1	0	1	0	0	1	1	0	5-10
XRA M	Exclusive OR memory with A	1	0	1	0	1	1	1	0	5-10
ORA M	OR memory with A	1	0	1	1	0	1	1	0	5-11
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	5-11
ANI	And immediate with A	1	1	1	0	0	1	1	0	5-10
XRI	Exclusive OR immediate with A	1	1	1	0	1	1	1	0	5-10
ORI	OR immediate with A	1	1	1	1	0	1	1	0	5-11
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	5-11
ROTATE										
RLC	Rotate A left	0	0	0	0	0	1	1	1	5-11
RRC	Rotate A right	0	0	0	0	1	1	1	1	5-12
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	5-12
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	5-12
SPECIALS										
CMA	Complement A	0	0	1	0	1	1	1	1	5-12
STC	Set carry	0	0	1	1	0	1	1	1	5-12
CMC	Complement carry	0	0	1	1	1	1	1	1	5-12
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	5-9
CONTROL										
EI	Enable Interrupts	1	1	1	1	1	0	1	1	5-17
DI	Disable Interrupt	1	1	1	1	0	0	1	1	5-17
NOP	No-operation	0	0	0	0	0	0	0	0	5-17
HLT	Halt	0	1	1	1	0	1	1	0	5-17
NEW 8085A INSTRUCTIONS										
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	5-17
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	5-18

NOTES: 1. DDD or SSS: B 000, C 001, D 010, E011, H 100, L 101, Memory 110, A 111.
2. Two possible cycle times, (6/12) indicate instruction cycles dependent on condition flags.

intel®

8155/8156/8155-2/8156-2
2048 BIT STATIC MOS RAM WITH I/O PORTS AND TIMER

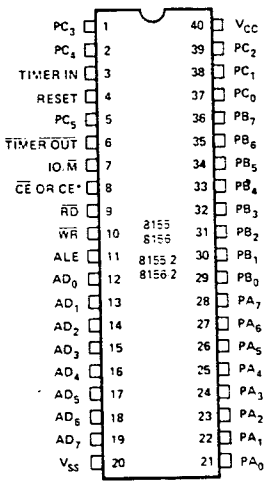
- 256 Word x 8 Bits
- Single +5V Power Supply
- Completely Static Operation
- Internal Address Latch
- 2 Programmable 8 Bit I/O Ports
- 1 Programmable 6-Bit I/O Port
- Programmable 14-Bit Binary Counter/ Timer
- Compatible with 8085A and 8088 CPU
- Multiplexed Address and Data Bus
- 40 Pin DIP

The 8155 and 89156 are RAM and I/O chips to be used in the 8085A and 8088 microprocessor systems. The RAM portion is designed with 2048 static cells organized as 256 x 8. They have a maximum access time of 400 ns to permit use with no wait states in 8085A CPU. The 8155-2 and 8156-2 have maximum access times of 330 ns for use with the 8085A-2 and the full speed 5 MHz 8088 CPU.

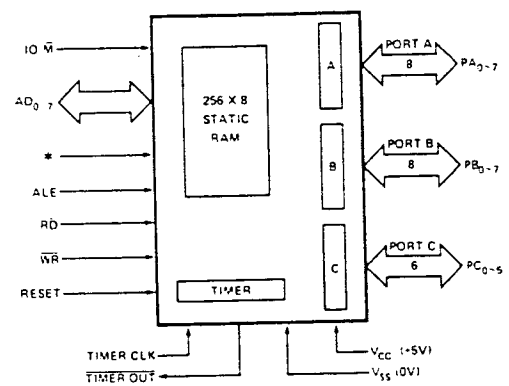
The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is jalso included on chip to provide either a square wave or terminal count pulse for the CPU system depending on timer mode.

PIN CONFIGURATION



BLOCK DIAGRAM



* 8155 8155-2 = $\overline{CE}$, 8156 8156-2 = CE

8155/8156 PIN FUNCTIONS

Symbol	Function
RESET (input)	Pulse provided by the 8085A to initialize the system (connect to 8085A RESET OUT). Input high on this line resets the chip and initializes the three I/O ports to input mode. The width of RESET pulse should typically be two 8085A clock cycle times.
AD_{0-7} (input)	3-state Address/Data lines that interface with the CPU lower 8-bit Address/Data Bus. The 8-bit address is latched into the address latch inside the 8155/56 on the falling edge of ALE. The address can be either for the memory section or the I/O section depending on the IO/$\overline{M}$ input. The 8-bit data is either written into the chip or read from the chip, depending on the $\overline{WR}$ or $\overline{RD}$ input signal.
CE or $\overline{CE}$ (input)	Chip Enable: On the 8155, this pin is $\overline{CE}$ and is ACTIVE LOW. On the 8156, this pin is CE and is ACTIVE HIGH.
$\overline{RD}$ (input)	Read control: Input low on this line with the Chip Enable active enables and AD_{0-7} buffers. If IO/$\overline{M}$ pin is low, the RAM content will be read out to the AD bus. Otherwise the content of the selected I/O port or command/status registers will be read to the AD bus.
$\overline{WR}$ (input)	Write control: Input low on this line with the Chip Enable active causes the data on the Address/Data bus to be written to the RAM or I/O ports and command/status register depending on IO/$\overline{M}$.
ALE (input)	Address Latch Enable: This control signal latches both the address on the AD_{0-7} lines and the state of the Chip Enable and IO/$\overline{M}$ into the chip at the falling edge of ALE.
IO/$\overline{M}$ (input)	Selects memory if low and I/O and command/status registers if high.
PA_{0-7} (8) (input/output)	These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
PB_{0-7} (8) (input/output)	These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
PC_{0-5} 6 (input/output)	These 6 pins can function as either input port, output port, or as control signals for PA and PB. Programming is done through the command register. When PC_{0-5} are used as control signals, they will provide the following: PC_0 — A INTR (Port A Interrupt) PC_1 — $\overline{ABF}$ Port A Buffer Full) PC_2 — $\overline{A\ STB}$ Port A Strobe) PC_3 — B INTR (Port B Interrupt) PC_4 — $\overline{B\ BF}$ Port B Buffer Full) PC_5 — B STB (Port B Strobe)
TIMER IN (input)	Input to the counter-timer.
$\overline{TIMER\ OUT}$ (output)	Timer output. This output can be either a square wave or a pulse depending on the timer mode.
V_{CC}	+5 volt supply.
V_{SS}	Ground Reference.

DESCRIPTION

The 8155/8156 contains the following:

- 2k Bit Static RAM organized as 256 x 8
- Two 8-bit I/O ports (PA & PB) and one 6-bit I/O port (PC)
- 14-bit timer-counter

The IO/$\overline{M}$ (IO/Memory Select) pin selects either the five registers (Command, Status, PA_{0-7}, PB_{0-7}, PC_{0-5}) or the memory (RAM) portion. (See Figure 1.)

The 8-bit address on the Address/Data lines, Chip Enable input CE or $\overline{CE}$, and IO/$\overline{M}$ are all latched on-chip at the falling edge of ALE. (See Figure 2.)

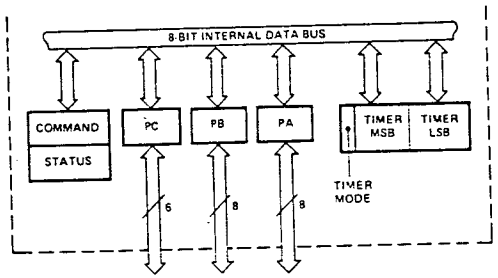
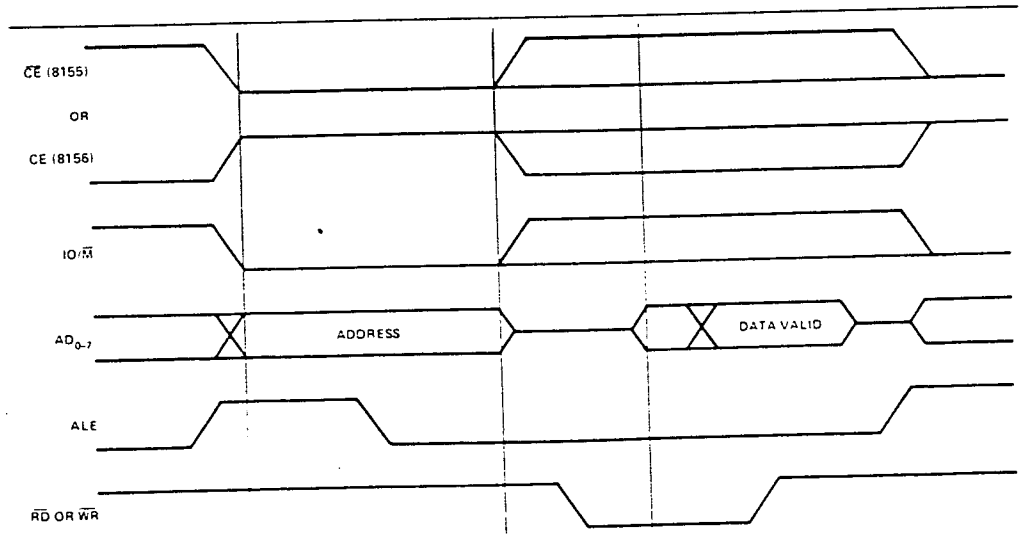


Figure 1. 8155/8156 Internal Registers



NOTE: FOR DETAILED TIMING INFORMATION, SEE FIGURE 12 AND A.C. CHARACTERISTICS.

Figure 2. 8155/8156 On-Board Memory Read/Write Cycle

PROGRAMMING OF THE COMMAND REGISTER

The command register consists of eight latches. Four bits (0-3) define the mode of the ports, two bits (4-5) enable or disable the interrupt from port C when it acts as control port, and the last two bits (6-7) are for the timer.

The command register contents can be altered at any time by using the I/O address XXXXX000 during a WRITE operation with the Chip Enable active and IO/$\overline{M}$ = 1. The meaning of each bit of the command byte is defined in Figure 3. The contents of the command register may never be read.

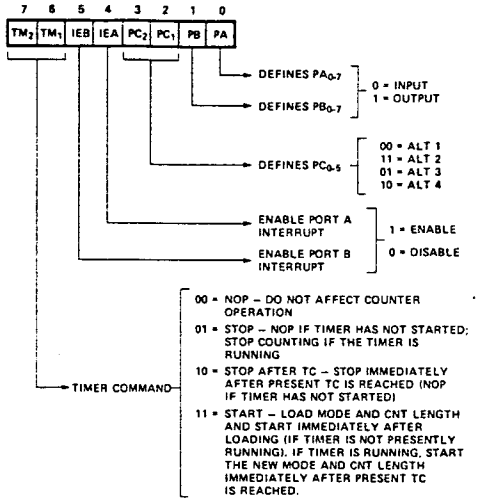


Figure 3. Command Register Bit Assignment

READING THE STATUS REGISTER

The status register consists of seven latches, one for each bit; six (0-5) for the status of the ports and one (6) for the status of the timer.

The status of the timer and the I/O section can be polled by reading the Status Register (Address XXXXX000). Status word format is shown in Figure 4. Note that you may never write to the status register since the command register shares the same I/O address and the command register is selected when a write to that address is issued.

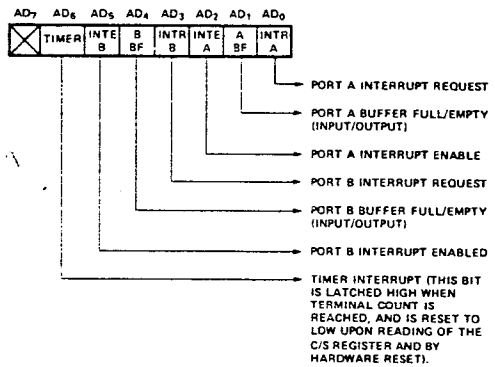


Figure 4. Status Register Bit Assignment

INPUT/OUTPUT SECTION

The I/O section of the 8155/8156 consists of five registers: (See Figure 5.)

- **Command/Status Register (C/S)** — Both registers are assigned the address XXXXX000. The C/S address serves the dual purpose.

  When the C/S registers are selected during WRITE operation, a command is written into the command register. The contents of this register are *not* accessible through the pins.

  When the C/S (XXXXX000) is selected during a READ operation, the status information of the I/O ports and the timer becomes available on the AD_{0-7} lines.

- **PA Register** — This register can be programmed to be either input or output ports depending on the status of the contents of the C/S Register. Also depending on the command, this port can operate in either the basic mode or the strobed mode See timing diagram . The I/O pins assigned in relation to this register are PA_{0-7}. The address of this register is XXXXX001.

- **PB Register** — This register functions the same as PA Register. The I/O pins assigned are PB_{0-7}. The address of this register is XXXXX010.

- **PC Register** — This register has the address XXXXX011 and contains only 6 bits. The 6 bits can be programmed to be either input ports. output ports or as control signals for PA and PB by properly programming the AD_2 and AD_3 bits of the C/S register.

  When PC_{0-5} is used as a control port. 3 bits are assigned for Port A and 3 for Port B. The first bit is an interrupt that the 8155 sends out. The second is an output signal indicating whether the buffer is full or empty, and the third is an input pin to accept a strobe for the strobed input mode. See Table 1.

When the 'C' port is programmed to either ALT3 or ALT4, the control signals for PA and PB are initialized as follows:

CONTROL	INPUT MODE	OUTPUT MODE
BF	Low	Low
INTR	Low	High
STB	Input Control	Input Control

I/O ADDRESS*								SELECTION
A7	A6	A5	A4	A3	A2	A1	A0	
X	X	X	X	X	0	0	0	Interval Command/Status Register
X	X	X	X	X	0	0	1	General Purpose I/O Port A
X	X	X	X	X	0	1	0	General Purpose I/O Port B
X	X	X	X	X	0	1	1	Port C — General Purpose I/O or Control
X	X	X	X	X	1	0	0	Low-Order 8 bits of Timer Count
X	X	X	X	X	1	0	1	High 6 bits of Timer Count and 2 bits of Timer Mode

X: Don't Care
*: I/O Address must be qualified by CE = 1 8156 or $\overline{CE}$ = 0 8155 and IO/$\overline{M}$ = 1 in order to select the appropriate register.

Figure 5. I/O port and Timer Addressing Scheme

Figure 6 shows how I/O PORTS A and B are structured within the 8155 and 8156:

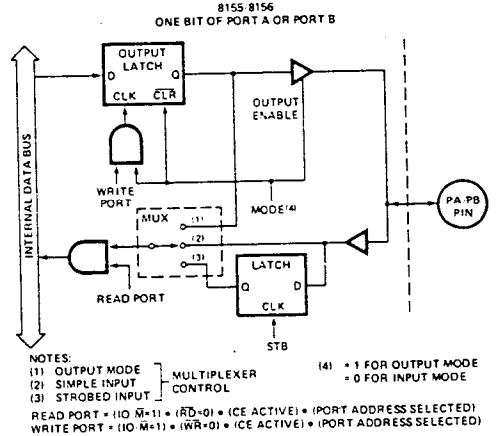


Figure 6. 8155/8156 Port Functions

TABLE 1. TABLE OF PORT CONTROL ASSIGNMENT.

Pin	ALT 1	ALT 2	ALT 3	ALT 4	
PC0	Input Port	Output Port	A INTR (Port A Interrupt)	A INTR (Port A Interrupt)	→
PC1	Input Port	Output Port	A BF (Port A Buffer Full)	A BF (Port A Buffer Full)	→
PC2	Input Port	Output Port	A $\overline{STB}$ (Port A Strobe)	A $\overline{STB}$ (Port A Strobe)	←
PC3	Input Port	Output Port	Output Port	B INTR (Port B Interrupt)	→
PC4	Input Port	Output Port	Output Port	B BF (Port B Buffer Full)	→
PC5	Input Port	Output Port	Output Port	B $\overline{STB}$ (Port B Strobe)	←

Note in the diagram that when the I/O ports are programmed to be output ports, the contents of the output ports can still be read by a READ operation when appropriately addressed.

The outputs of the 8155/8156 are "glitch-free" meaning that you can write a "1" to a bit position that was previously "1" and the level at the output pin will not change.

Note also that the output latch is cleared when the port enters the input mode. The output latch cannot be loaded by writing to the port if the port is in the input mode. The result is that each time a port mode is changed from input to output, the output pins will go low. When the 8155/56 is RESET, the output latches are all cleared and all 3 ports enter the input mode.

When in the ALT 1 or ALT 2 modes, the bits of PORT C are structured like the diagram above in the simple input or output mode, respectively.

Reading from an input port with nothing connected to the pins will provide unpredictable results.

Figure 7 shows how the 8155/8156 I/O ports might be configured in a typical MCS-85 system.

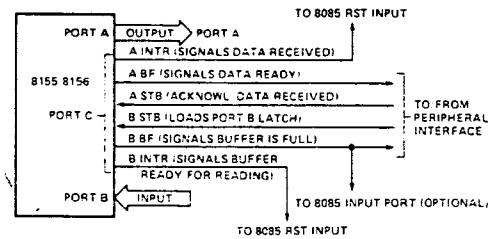
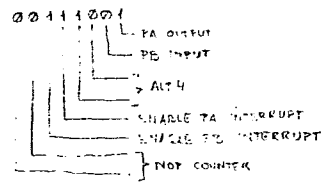


Figure 7. Example: Command Register = 00111001



TIMER SECTION

The timer is a 14-bit down-counter that counts the TIMER IN pulses and provides either a square wave or pulse when terminal count (TC) is reached.

The timer has the I/O address XXXXX100 for the low order byte of the register and the I/O address XXXXX101 for the high order byte of the register. (See Figure 5).

To program the timer, the COUNT LENGTH REG is loaded first, one byte at a time, by selecting the timer addresses. Bits 0-13 of the high order count register will specify the length of the next count and bits 14-15 of the high order register will specify the timer output mode (see Figure 8). The value loaded into the count length register can have any value from 2H through 3FFH in Bits 0-13.

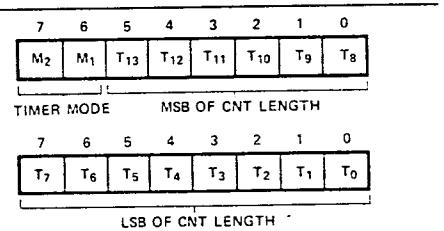


Figure 8. Timer Format

There are four modes to choose from: M2 and M1 define the timer mode, as shown in Figure 9.

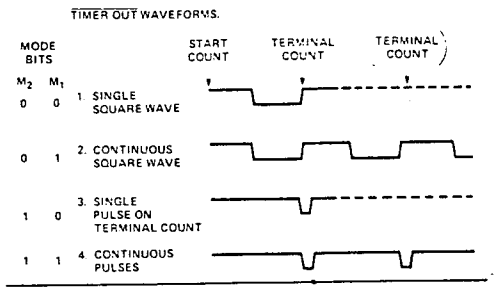


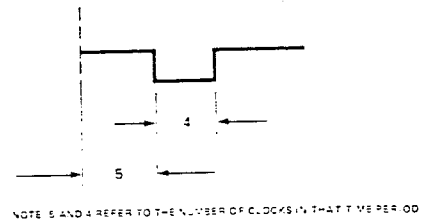
Figure 9. Timer Modes

Bits 6-7 TM_2 and TM_1 of command register contents are used to start and stop the counter. There are four commands to choose from:

TM_2	TM_1	
0	0	NOP — Do not affect counter operation.
0	1	STOP — NOP if timer has not started; stop counting if the timer is running.
1	0	STOP AFTER TC — Stop immediately after present TC is reached. NOP if timer has not started
1	1	START — Load mode and CNT length and start immediately after loading. If timer is not presently running. If timer is running, start the new mode and CNT length immediately after present TC is reached.

Note that while the counter is counting, you may load a new count and mode into the count length registers. Before the new count and mode will be used by the counter, you must issue a START command to the counter. This applies even though you may only want to change the count and use the previous mode.

In case of an odd-numbered count, the first half-cycle of the squarewave output, which is high, is one count longer than the second low half-cycle, as shown in Figure 10.



NOTE: 5 AND 4 REFER TO THE NUMBER OF CLOCKS IN THAT TIME PERIOD

Figure 10. Asymmetrical Square-Wave Output Resulting from Count of 9

The counter in the 8155 is not initialized to any particular mode or count when hardware RESET occurs, but RESET does *stop* the counting. Therefore, counting cannot begin following RESET until a START command is issued via the C/S register.

Please note that the timer circuit on the 8155 8156 chip is designed to be a square-wave timer, not an event counter. To achieve this, it counts down by twos twice in completing one cycle. Thus, its registers do not contain values directly representing the number of TIMER IN pulses received. You cannot load an initial value of 1 into the count register and cause the timer to operate, as its terminal count value is 10 binary or 2 decimal. For the detection of single pulses, it is suggested that one of the hardware interrupt pins on the 8085A be used. After the timer has started counting down, the values residing in the count registers can be used to calculate the actual number of TIMER IN pulses required to complete the timer cycle if desired. To obtain the remaining count, perform the following operations in order:

1. Stop the count
2. Read in the 16-bit value from the count length registers
3. Reset the upper two mode bits
4. Reset the carry and rotate right one position all 16 bits *through carry*
5. If carry is set, add 1/2 of the full original count. 1/2 full count — 1 if full count is odd.

Note: If you started with an odd count and you read the count length register before the third count pulse occurs, you will not be able to discern whether one or two counts has occurred. Regardless of this, the 8155/56 always counts out the right number of pulses in generating the $\overline{TIMER\ OUT}$ waveforms.

EXAMPLE PROGRAM

Following is an actual sequence of program steps that adjusts the 8155/56 count register contents to obtain the count, extracted from Intel® Application Note AP38. "Application Techniques for the Intel 8085A Bus." First store the value of the full original count in register HL of the 8085A. Then stop the count to avoid getting an incorrect count value. Then sample the timer-counter, storing the lower-order byte of the current count register in register C and the higher-order count byte in register B. Then, call the following 8080A/8085A subroutine:

HL = original Count

```
ADJUST, 78    MOV A,B     ;Load accumulator with upper half
                          ; of count.
      E63F    ANI 3F      ;Reset upper 2 bits and clear carry.
      1F      RAR         ;Rotate right through carry.
      47      MOV B,A     ;Store shifted value back in B.
      79      MOV A,C     ;Load accumulator with lower half.
      1F      RAR         ;Rotate right through carry.
      4F      MOV C,A     ;Store lower byte in C.
      DØ      RNC         ;If in 2nd half of count, return.
                          ;If in 1st half, go on.
      3F      CMC         ;Clear carry.
      7C      MOV A,H     ;Divide full count by 2.  If HL
                          ;is odd, disregard remainder.
      1F      RAR
      67      MOV H,A
      7D      MOV A,L
      1F      RAR
      6F      MOV L,A     ; HL ← HL/2
      09      DAD B       ;Double-precision add HL and BC.
      44      MOV B,H     ;Store results back in BC.
      4D      MOV C,L
      C9      RET         ;Return.
```

B= C=

1 1ª mitad
Ø 2ª mitad ⇒
nº de ciclos que faltan

After executing the subroutine, BC will contain the remaining count in the current count cycle.

8085A MINIMUM SYSTEM CONFIGURATION

Figure 11a shows a minimum system using three chips, containing:

- 256 Bytes RAM
- 2K Bytes ROM
- 38 I/O Pins
- 1 Interval Timer
- 4 Interrupt Levels

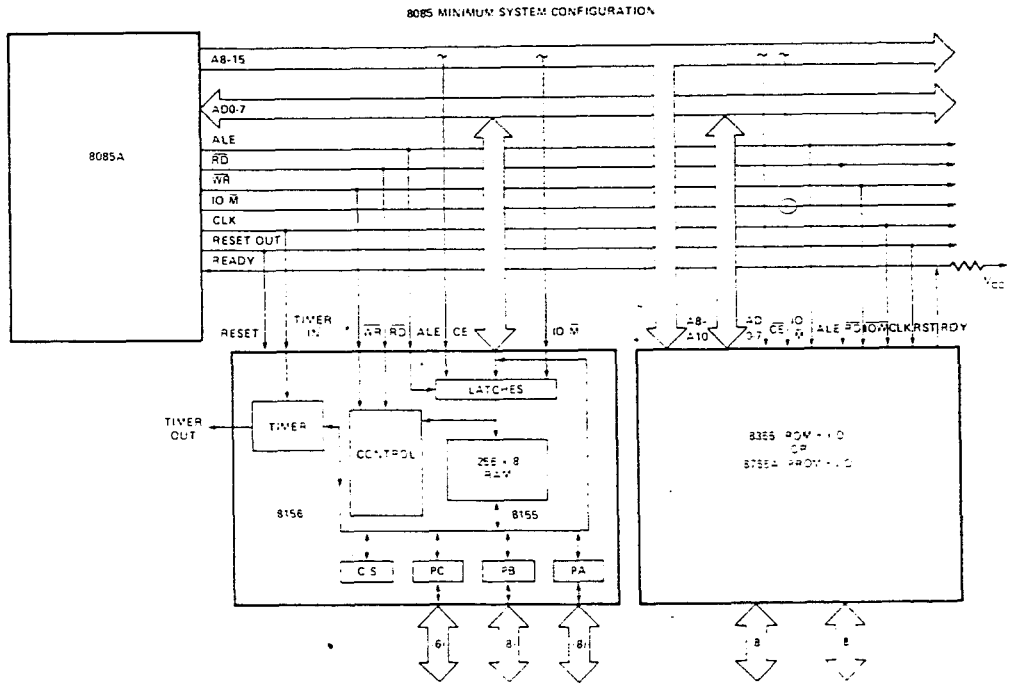


Figure 11a. 8085A Minimum System Configuration. (Memory Mapped I/O)

8088 FIVE CHIP SYSTEM

Figure 11b shows a five chip system containing:

- 1.25K Bytes RAM
- 2K Bytes ROM
- 38 I/O Pins
- 1 Interval Timer
- 2 Interrupt Levels

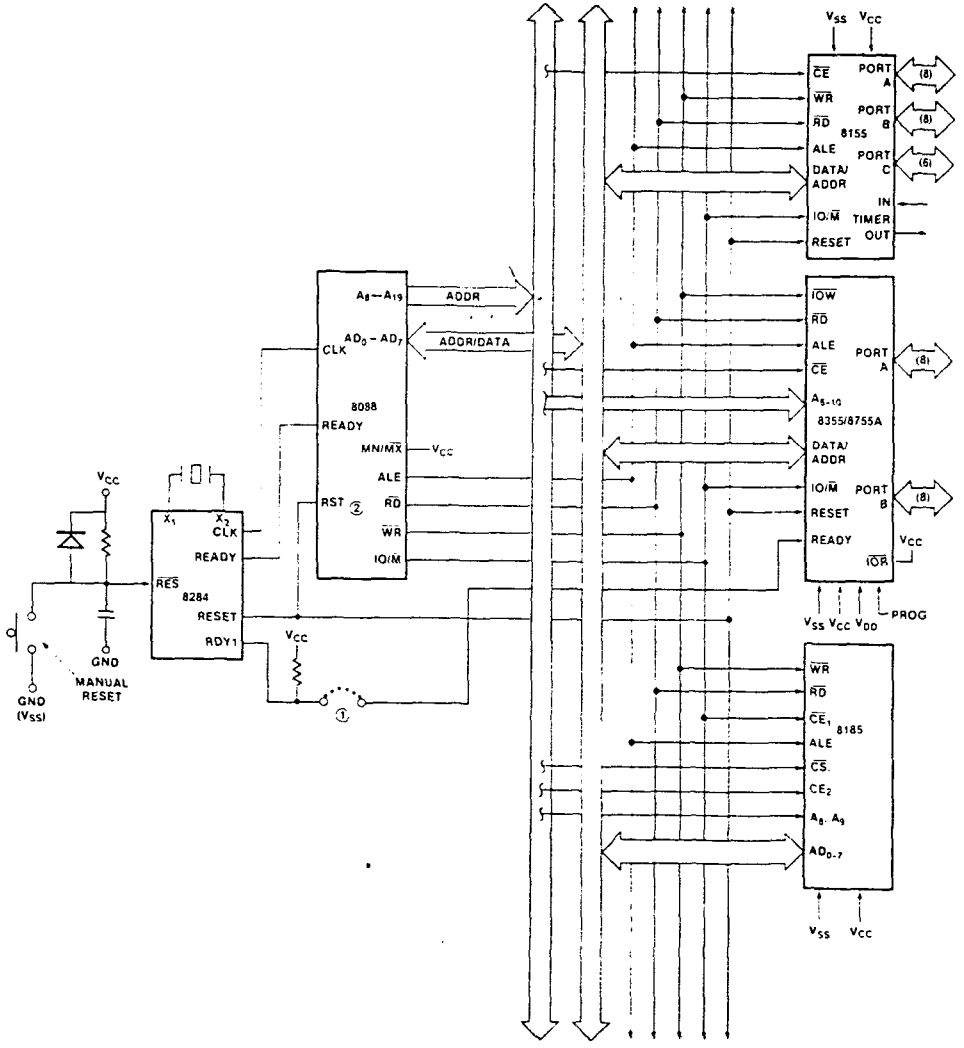


Figure 11b. 8088 Five Chip System Configuration

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	0°C to +70°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin With Respect to Ground	-0.5V to +7V
Power Dissipation	1.5W

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ± 5%)

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} to 0V
I_{LO}	Output Leakage Current		± 10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	V_{CC} Supply Current		180	mA	
I_{IL}(CE)	Chip Enable Leakage 8155 8156		 +100 -100	 μA μA	V_{IN} = V_{CC} to 0V

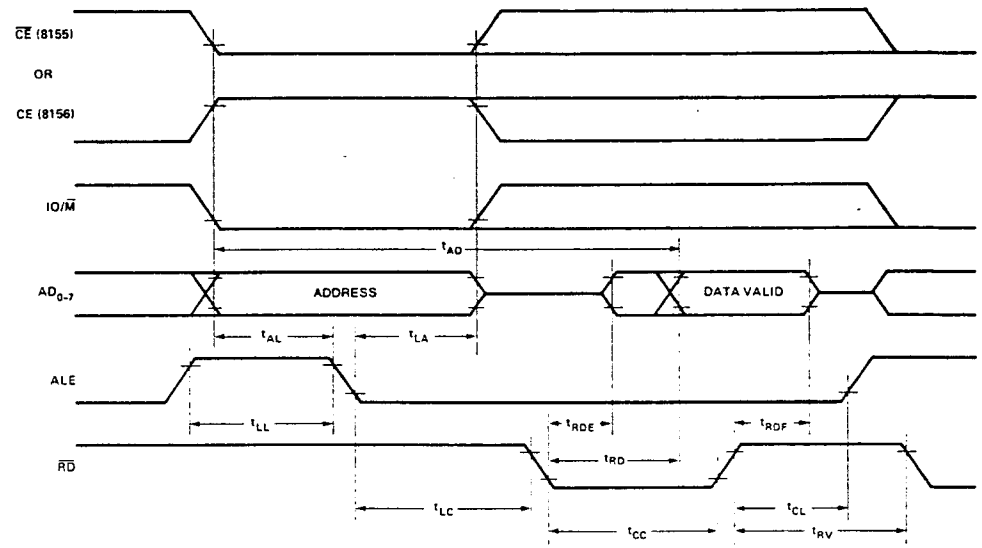
A.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ± 5%)

SYMBOL	PARAMETER	8155/8156		8155-2/8156-2 (Preliminary)		UNITS
		MIN.	MAX.	MIN.	MAX.	
t_{AL}	Address to Latch Set Up Time	50		30		ns
t_{LA}	Address Hold Time after Latch	80		30		ns
t_{LC}	Latch to READ/WRITE Control	100		40		ns
t_{RD}	Valid Data Out Delay from READ Control		170		140	ns
t_{AD}	Address Stable to Data Out Valid		400		330	ns
t_{LL}	Latch Enable Width	100		70		ns
t_{RDF}	Data Bus Float After READ	0	100	0	80	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		10		ns
t_{CC}	READ/WRITE Control Width	250		200		ns
t_{DW}	Data In to WRITE Set Up Time	150		100		ns
t_{WD}	Data In Hold Time After WRITE	0		0		ns
t_{RV}	Recovery Time Between Controls	300		200		ns
t_{WP}	WRITE to Port Output		400		300	ns
t_{PR}	Port Input Setup Time	70		50		ns
t_{RP}	Port Input Hold Time	50		10		ns
t_{SBF}	Strobe to Buffer Full		400		300	ns
t_{SS}	Strobe Width	200		150		ns
t_{RBE}	READ to Buffer Empty		400		300	ns
t_{SI}	Strobe to INTR On		400		300	ns
t_{RDI}	READ to INTR Off		400		300	ns
t_{PSS}	Port Setup Time to Strobe Strobe	50		0		ns
t_{PHS}	Port Hold Time After Strobe	120		100		ns
t_{SBE}	Strobe to Buffer Empty		400		300	ns
t_{WBF}	WRITE to Buffer Full		400		300	ns
t_{WI}	WRITE to INTR Off		400		300	ns
t_{TL}	TIMER-IN to $\overline{\text{TIMER-OUT}}$ Low		400		300	ns
t_{TH}	TIMER-IN to $\overline{\text{TIMER-OUT}}$ High		400		300	ns
t_{RDE}	Data Bus Enable from READ Control	10		10		ns
t_1	TIMER-IN Low Time	80		40		ns
t_2	TIMER-IN High Time	120		70		ns

Input Waveform for A.C. Tests:

WAVEFORMS

a. Read Cycle



b. Write Cycle

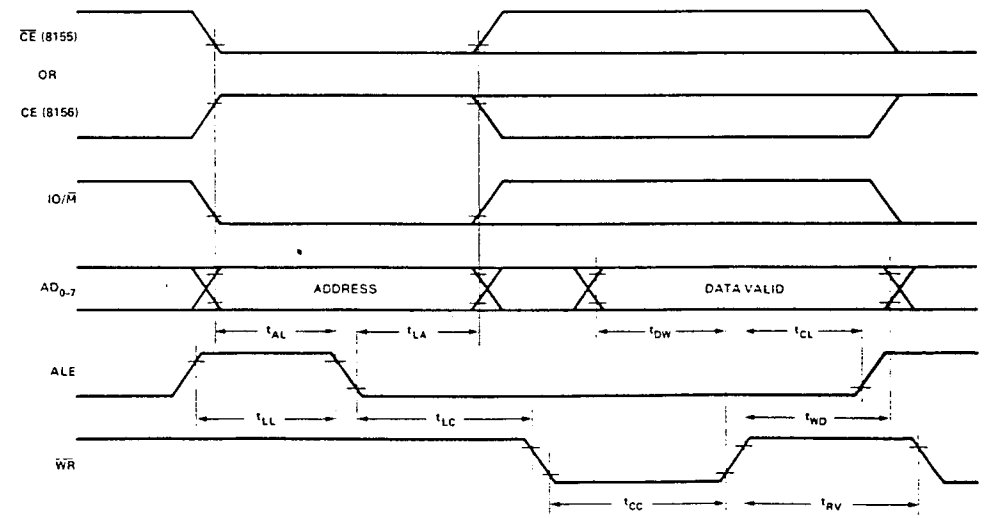
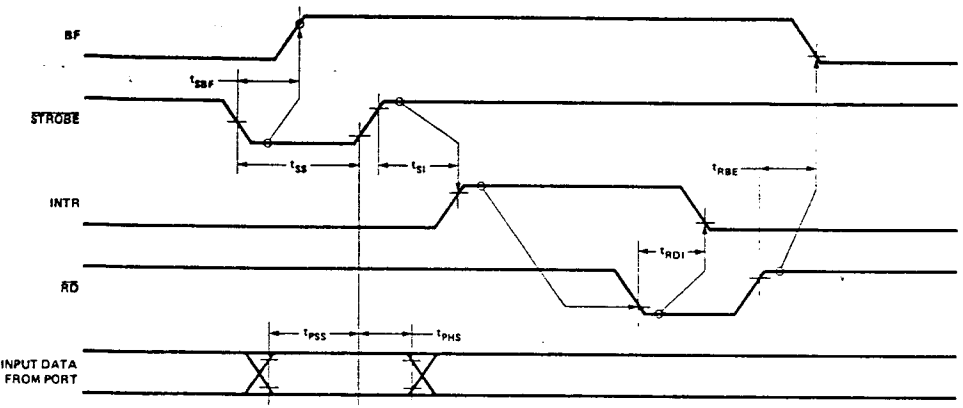


Figure 12. 8155/8156 Read/Write Timing Diagrams

a. Strobed Input Mode



b. Strobed Output Mode

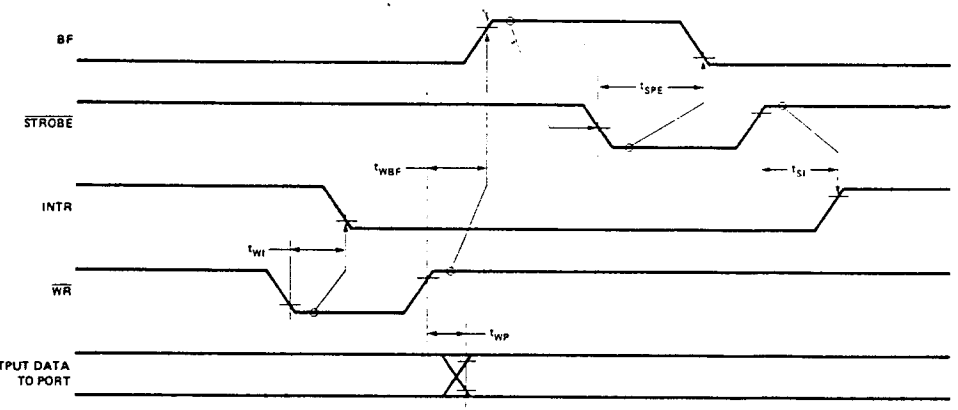
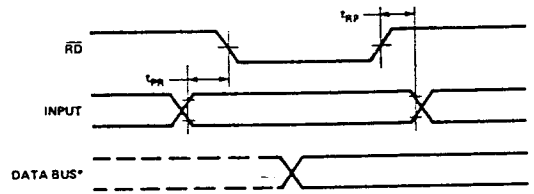
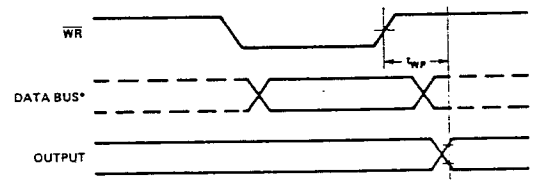


Figure 13. Strobed I/O Timing

a. Basic Input Mode

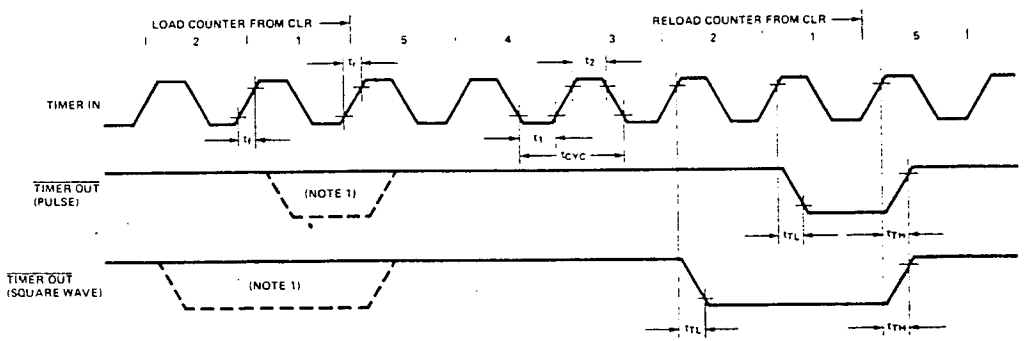


b. Basic Output Mode



*DATA BUS TIMING IS SHOWN IN FIGURE 7.

Figure 14. Basic I/O Timing Waveform



NOTE 1: THE TIMER OUTPUT IS PERIODIC IF IN AN AUTOMATIC RELOAD MODE (M_1 MODE BIT = 1)

Figure 15. Timer Output Waveform Countdown from 5 to 1

8251A
PROGRAMMABLE COMMUNICATION INTERFACE

- Synchronous and Asynchronous Operation
- Synchronous 5–8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion
- Asynchronous 5–8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling
- Synchronous Baud Rate—DC to 64K Baud
- Asynchronous Baud Rate—DC to 19.2K Baud
- Full-Duplex, Double-Buffered Transmitter and Receiver
- Error Detection—Parity, Overrun and Framing
- Compatible with an Extended Range of Intel Microprocessors
- 28-Pin DIP Package
- All Inputs and Outputs are TTL Compatible
- Available in EXPRESS
  —Standard Temperature Range
  —Extended Temperature Range

The Intel® 8251A is the enhanced version of the industry standard, Intel 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's microprocessor families such as MCS-68, 80, 85, and iAPX-86, 88. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is fabricated using N-channel silicon gate technology.

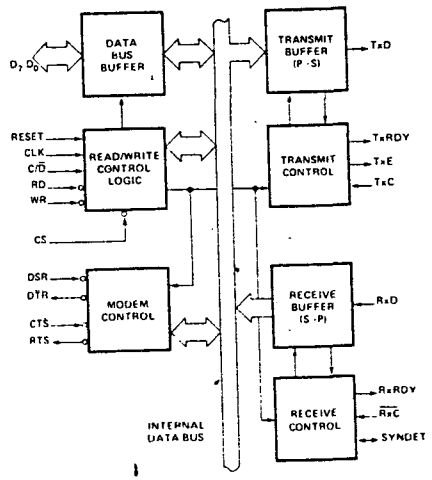


Figure 1. Block Diagram

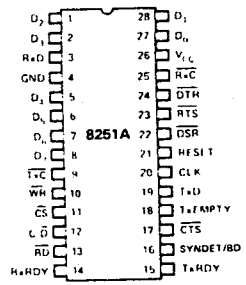


Figure 2. Pin Configuration

AFN-01573D

FEATURES AND ENHANCEMENTS

The 8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.
- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the $\overline{RD}$ and $\overline{WR}$ do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Synchronous Baud rate from DC to 64K.

FUNCTIONAL DESCRIPTION

General

The 8251A is a Universal Synchronous/Asynchronous Receiver/Transmitter designed for a wide range of Intel microcomputers such as 8048, 8080, 8085, 8086 and 8088. Like other I/O devices in a microcomputer system, its functional configuration is programmed by the system's software for maximum flexibility. The 8251A can support most serial data techniques in use, including IBM "bi-sync."

In a communication environment an interface device must convert parallel format system data into *serial format for transmission and convert incoming* serial format data into parallel system data for reception. The interface device must also delete or insert bits or characters that are functionally unique to the communication technique. In essence, the interface should appear "transparent" to the CPU, a simple input or output of byte-oriented system data.

Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8251A to the system Data Bus. Data is transmitted or received by the buffer upon execution of INput or OUTput instructions of the CPU. Control words, Command words and Status information are also transferred through the Data Bus Buffer. The Command Status, Data-In and Data-Out registers are separate, 8-bit registers communicating with the system bus through the Data Bus Buffer.

This functional block accepts inputs from the system Control bus and generates control signals for overall device operation. It contains the Control Word Register and Command Word Register that store the various control formats for the device functional definition.

RESET (Reset)

A "high" on this input forces the 8251A into an "Idle" mode. The device will remain at "Idle" until a new set of control words is written into the 8251A to program its functional definition. Minimum RESET pulse width is 6 t_{CY} (clock must be running).

A command reset operation also puts the device into the "Idle" state.

AFN-01573D

CLK (Clock)

The CLK input is used to generate internal device timing and is normally connected to the Phase 2 (TTL) output of the Clock Generator. No external inputs or outputs are referenced to CLK but the frequency of CLK must be greater than 30 times the Receiver or Transmitter data bit rates.

$\overline{WR}$ (Write)

A "low" on this input informs the 8251A that the CPU is writing data or control words to the 8251A.

$\overline{RD}$ (Read)

A "low" on this input informs the 8251A that the CPU is reading data or status information from the 8251A.

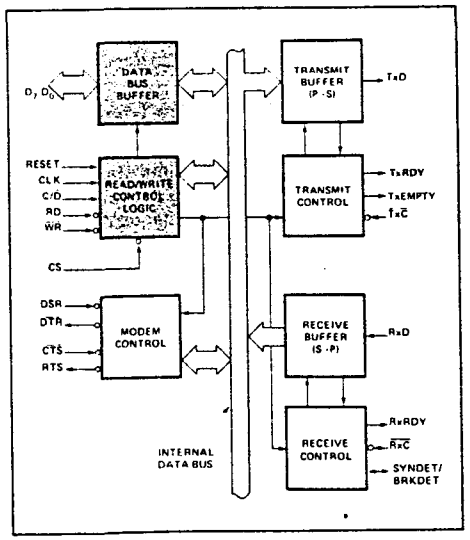


Figure 3. 8251A Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	8251A DATA → DATA BUS
0	1	0	0	DATA BUS → 8251A DATA
1	0	1	0	STATUS → DATA BUS
1	1	0	0	DATA BUS → CONTROL
X	1	1	0	DATA BUS = 3-STATE
X	X	X	1	DATA BUS = 3-STATE

C/$\overline{D}$ (Control/Data)

This input, in conjunction with the $\overline{WR}$ and $\overline{RD}$ inputs, informs the 8251A that the word on the Data Bus is either a data character, control word or status information.

1 = CONTROL/STATUS; 0 = DATA.

$\overline{CS}$ (Chip Select)

A "low" on this input selects the 8251A. No reading or writing will occur unless the device is selected. When $\overline{CS}$ is high, the Data Bus is in the float state and $\overline{RD}$ and $\overline{WR}$ have no effect on the chip.

Modem Control

The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any modem. The modem control signals are general purpose in nature and can be used for functions other than modem control, if necessary.

$\overline{DSR}$ (Data Set Ready)

The $\overline{DSR}$ input signal is a general-purpose, 1-bit inverting input port. Its condition can be tested by the CPU using a Status Read operation. The $\overline{DSR}$ input is normally used to test modem conditions such as Data Set Ready.

$\overline{DTR}$ (Data Terminal Ready)

The $\overline{DTR}$ output signal is a general-purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{DTR}$ output signal is normally used for modem control such as Data Terminal Ready.

$\overline{RTS}$ (Request to Send)

The $\overline{RTS}$ output signal is a general-purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{RTS}$ output signal is normally used for modem control such as Request to Send.

$\overline{CTS}$ (Clear to Send)

A "low" on this input enables the 8251A to transmit serial data if the Tx Enable bit in the Command byte is set to a "one." If either a Tx Enable off or $\overline{CTS}$ off condition occurs while the Tx is in operation, the Tx will transmit all the data in the USART, written prior to Tx Disable command before shutting down.

Transmitter Buffer

The Transmitter Buffer accepts parallel data from the Data Bus Buffer, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin on the falling edge of $\overline{TxC}$. The transmitter will begin transmission upon being enabled if $\overline{CTS}$ = 0. The TxD line will be held in the marking state immediately upon a master Reset or when Tx Enable or $\overline{CTS}$ is off or the transmitter is empty.

Transmitter Control

The Transmitter Control manages all activities associated with the transmission of serial data. It accepts and issues signals both externally and internally to accomplish this function.

TxRDY (Transmitter Ready)

This output signals the CPU that the transmitter is ready to accept a data character. The TxRDY output pin can be used as an interrupt to the system, since it is masked by TxEnable; or, for Polled operation, the CPU can check TxRDY using a Status Read operation. TxRDY is automatically reset by the leading edge of $\overline{WR}$ when a data character is loaded from the CPU.

Note that when using the Polled operation, the TxRDY status bit is *not* masked by TxEnable, but will only indicate the Empty/Full Status of the Tx Data Input Register.

TxE (Transmitter Empty)

When the 8251A has no characters to send, the TxEMPTY output will go "high." It resets upon receiving a character from CPU if the transmitter is enabled. TxEMPTY remains high when the transmitter is disabled. TxEMPTY can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplex operational mode.

In the Synchronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be or are being transmitted automatically as "fillers." TxEMPTY does not go low when the SYNC characters are being shifted out.

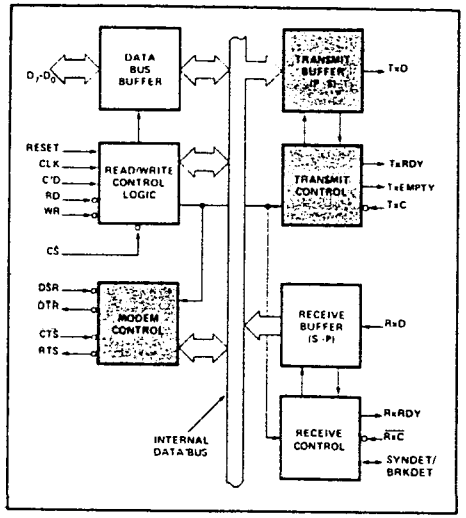


Figure 4. 8251A Block Diagram Showing Modem and Transmitter Buffer and Control Functions

$\overline{TxC}$ (Transmitter Clock)

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the Baud Rate (1x) is equal to the $\overline{TxC}$ frequency. In Asynchronous transmission mode, the baud rate is a fraction of the actual $\overline{TxC}$ frequency. A portion of the mode instruction selects this factor; it can be 1, 1/16 or 1/64 the $\overline{TxC}$.

For Example:

If Baud Rate equals 110 Baud,
$\overline{TxC}$ equals 110 Hz in the 1x mode.
$\overline{TxC}$ equals 1.72 kHz in the 16x mode.
$\overline{TxC}$ equals 7.04 kHz in the 64x mode.

The falling edge of $\overline{TxC}$ shifts the serial data out of the 8251A.

Receiver Buffer

The Receiver accepts serial data, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU. Serial data is input to RxD pin, and is clocked in on the rising edge of $\overline{RxC}$.

Receiver Control

This functional block manages all receiver-related activities which consists of the following features.

The RxD initialization circuit prevents the 8251A from mistaking an unused input line for an active low data line in the "break condition." Before starting to receive serial characters on the RxD line, a valid "1" must first be detected after a chip master Reset. Once this has been determined, a search for a valid low (Start bit) is enabled. This feature is only active in the asynchronous mode, and is only done once for each master Reset.

The False Start bit detection circuit prevents false starts due to a transient noise spike by first detecting the falling edge and then strobing the nominal center of the Start bit (RxD = low).

Parity error detection sets the corresponding status bit.

The Framing Error status bit is set if the Stop bit is absent at the end of the data byte (asynchronous mode).

RxRDY (Receiver Ready)

This output indicates that the 8251A contains a character that is ready to be input to the CPU. RxRDY can be connected to the interrupt structure of the CPU or, for polled operation, the CPU can check the condition of RxRDY using a Status Read operation.

RxEnable, when off, holds RxRDY in the Reset Condition. For Asynchronous mode, to set RxRDY, the Receiver must be enabled to sense a Start Bit and a complete character must be assembled and transferred to the Data Output Register. For Synchronous mode, to set RxRDY, the Receiver must be enabled and a character must finish assembly and be transferred to the Data Output Register.

Failure to read the received character from the Rx Data Output Register prior to the assembly of the next Rx Data character will set overrun condition error and the previous character will be written over and lost. If the Rx Data is being read by the CPU when the internal transfer is occurring, overrun error will be set and the old character will be lost.

$\overline{RxC}$ (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the Baud Rate (1x) is equal to the actual frequency of $\overline{RxC}$. In Asynchronous Mode, the Baud Rate is a fraction of the actual $\overline{RxC}$ frequency. A portion of the mode instruction selects this factor: 1, 1/16 or 1/64 the $\overline{RxC}$.

For example:

Baud Rate equals 300 Baud, if
$\overline{RxC}$ equals 300 Hz in the 1x mode;
$\overline{RxC}$ equals 4800 Hz in the 16x mode;
$\overline{RxC}$ equals 19.2 kHz in the 64x mode.

Baud Rate equals 2400 Baud, if
$\overline{RxC}$ equals 2400 Hz in the 1x mode;
$\overline{RxC}$ equals 38.4 kHz in the 16x mode;
$\overline{RxC}$ equals 153.6 kHz in the 64x mode.

Data is sampled into the 8251A on the rising edge of $\overline{RxC}$.

NOTE: In most communications systems, the 8251A will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the same. Both $\overline{TxC}$ and $\overline{RxC}$ will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

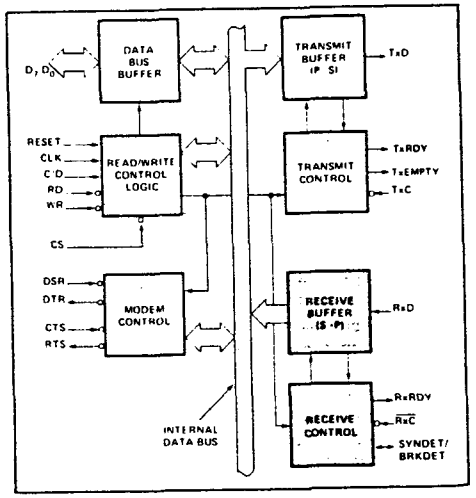


Figure 5. 8251A Block Diagram Showing Receiver Buffer and Control Functions

SYNDET (SYNC Detect/ BRKDET Break Detect)

This pin is used in Synchronous Mode for SYNDET and may be used as either input or output, programmable through the Control Word. It is reset to output mode low upon RESET. When used as an output (internal Sync mode), the SYNDET pin will go "high" to indicate that the 8251A has located the SYNC character in the Receive mode. If the 8251A is programmed to use double Sync characters (bisync), then SYNDET will go "high" in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

When used as an input (external SYNC detect mode), a positive going signal will cause the 8251A to start assembling data characters on the rising edge of the next $\overline{RxC}$. Once in SYNC, the "high" input signal can be removed. When External SYNC Detect is programmed, Internal SYNC Detect is disabled.

BREAK (Async Mode Only)

This output will go high whenever the receiver remains low through two consecutive stop bit sequences (including the start bits, data bits, and parity bits). Break Detect may also be read as a Status bit. It is reset only upon a master chip Reset or Rx Data returning to a "one" state.

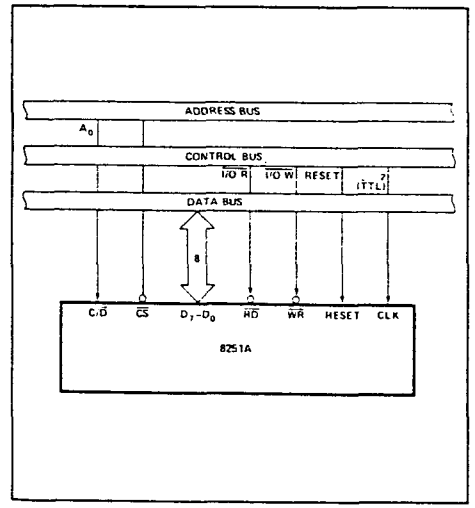


Figure 6. 8251A Interface to 8080 Standard System Bus

DETAILED OPERATION DESCRIPTION

General

The complete functional definition of the 8251A is programmed by the system's software. A set of control words must be sent out by the CPU to initialize the 8251A to support the desired communications format. These control words will program the: BAUD RATE, CHARACTER LENGTH, NUMBER OF STOP BITS, SYNCHRONOUS or ASYNCHRONOUS OPERATION, EVEN/ODD/OFF PARITY, etc. In the Synchronous Mode, options are also provided to select either internal or external character synchronization.

Once programmed, the 8251A is ready to perform its communication functions. The TxRDY output is raised "high" to signal the CPU that the 8251A is ready to receive a data character from the CPU. This output (TxRDY) is reset automatically when the CPU writes a character into the 8251A. On the other hand, the 8251A receives serial data from the MODEM or I/O device. Upon receiving an entire character, the RxRDY output is raised "high" to signal the CPU that the 8251A has a complete character ready for the CPU to fetch. RxRDY is reset automatically upon the CPU data read operation.

The 8251A cannot begin transmission until the Tx Enable (Transmitter Enable) bit is set in the Command Instruction and it has received a Clear To Send ($\overline{CTS}$) input. The TxD output will be held in the marking state upon Reset.

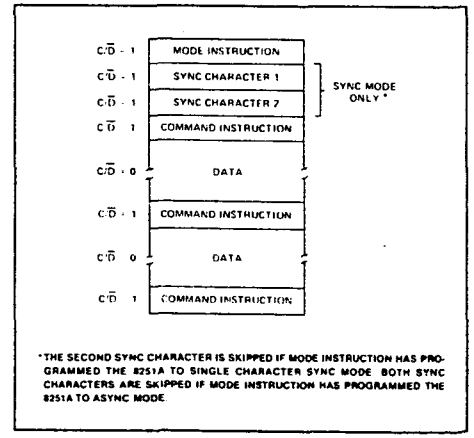


Figure 7. Typical Data Block

Programming the 8251A

Prior to starting data transmission or reception, the 8251A must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251A and must immediately follow a Reset operation (internal or external).

The control words are split into two formats:

1. Mode Instruction
2. Command Instruction

Mode Instruction

This instruction defines the general operational characteristics of the 8251A. It must follow a Reset operation (internal or external). Once the Mode Instruction has been written into the 8251A by the CPU, SYNC characters or Command Instructions may be written.

Command Instruction

This instruction defines a word that is used to control the actual operation of the 8251A.

Both the Mode and Command Instructions must conform to a specified sequence for proper device operation (see Figure 7). The Mode Instruction must be written immediately following a Reset operation, prior to using the 8251A for data communication.

All control words written into the 8251A after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251A at any time in the data block during the operation of the 8251A. To return to the Mode Instruction format, the master Reset bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251A back into the Mode Instruction format. Command Instructions must follow the Mode Instructions or Sync characters.

Mode Instruction Definition

The 8251A can be used for either Asynchronous or Synchronous data communication. To understand how the Mode Instruction defines the functional operation of the 8251A, the designer can best view the device as two separate components, one Asynchronous and the other Synchronous, sharing the same package. The format definition can be changed only after a master chip Reset. For explanation purposes the two formats will be isolated.

NOTE: When parity is enabled it is not considered as one of the data bits for the purpose of programming the word length. The actual parity bit received on the Rx Data line cannot be read on the Data Bus. In the case of a programmed character length of less than 8 bits, the least significant Data Bus bits will hold the data; unused bits are "don't care" when writing data to the 8251A, and will be "zeros" when reading the data from the 8251A.

Asynchronous Mode (Transmission)

Whenever a data character is sent by the CPU the 8251A automatically adds a Start bit (low level) followed by the data bits (least significant bit first), and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TxD output. The serial data is shifted out on the falling edge of $\overline{TxC}$ at a rate equal to 1, 1/16, or 1/64 that of the $\overline{TxC}$, as defined by the Mode Instruction. BREAK characters can be continuously sent to the TxD if commanded to do so.

When no data characters have been loaded into the 8251A the TxD output reamins "high" (marking) unless a Break (continuously low) has been programmed.

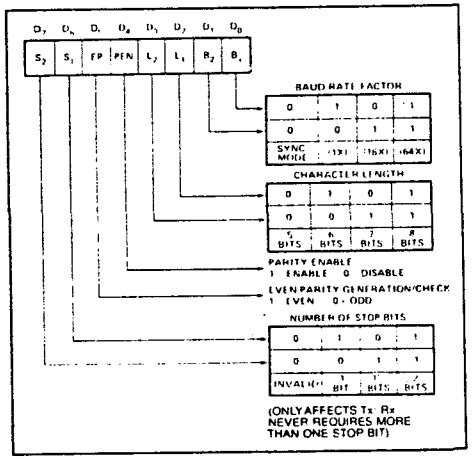


Figure 8. Mode Instruction Format, Asynchronous Mode

Asynchronous Mode (Receive)

The RxD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center (16X or 64X mode only). If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter thus locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RxD pin with the rising edge of $\overline{RxC}$. If a low level is detected as the STOP bit, the Framing Error flag will be set. The STOP bit signals the end of a character. Note that the *receiver* requires only *one* stop bit, regardless of the number of stop bits programmed. This character is then loaded into the parallel I/O buffer of the 8251A. The RxRDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the OVERRUN Error flag is raised (thus the previous character is lost). All of the error flags can be reset by an Error Reset Instruction. The occurrence of any of these errors will not affect the operation of the 8251A.

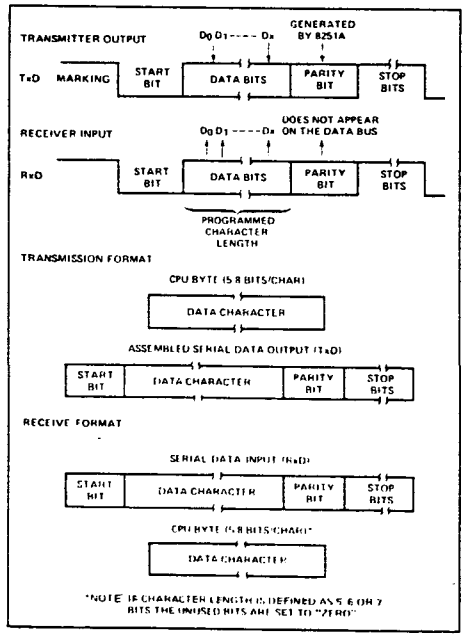
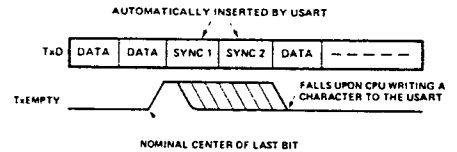


Figure 9. Asynchronous Mode

Synchronous Mode (Transmission)

The TxD output is continuously high until the CPU sends its first character to the 8251A which usually is a SYNC character. When the $\overline{CTS}$ line goes low, the first character is serially transmitted out. All characters are shifted out on the falling edge of $\overline{TxC}$. Data is shifted out at the same rate as the $\overline{TxC}$.

Once transmission has started, the data stream at the TxD output must continue at the $\overline{TxC}$ rate. If the CPU does not provide the 8251A with a data character before the 8251A Transmitter Buffers become empty, the SYNC characters (or character if in single SYNC character mode) will be automatically inserted in the TxD data stream. In this case, the TxEMPTY pin is raised high to signal that the 8251A is empty and SYNC characters are being sent out. TxEMPTY does not go low when the SYNC is being shifted out (see figure below). The TxEMPTY pin is internally reset by a data character being written into the 8251A.



Synchronous Mode (Receive)

In this mode, character synchronization can be internally or externally achieved. If the SYNC mode has been programmed, ENTER HUNT command should be included in the first command instruction word written. Data on the RxD pin is then sampled on the rising edge of $\overline{RxC}$. The content of the Rx buffer is compared at every bit boundary with the first SYNC character until a match occurs. If the 8251A has been programmed for two SYNC characters, the subsequent received character is also compared; when both SYNC characters have been detected, the USART ends the HUNT mode and is in character synchronization. The SYNDET pin is then set high, and is reset automatically by a STATUS READ. If parity is programmed, SYNDET will not be set until the middle of the parity bit instead of the middle of the last data bit.

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDET pin, thus forcing the 8251A out of the HUNT mode. The high level can be removed after one $\overline{RxC}$ cycle. An ENTER HUNT command has no effect in the asynchronous mode of operation.

Parity error and overrun error are both checked in the same way as in the Asynchronous Rx mode. Parity is checked when not in Hunt, regardless of whether the Receiver is enabled or not.

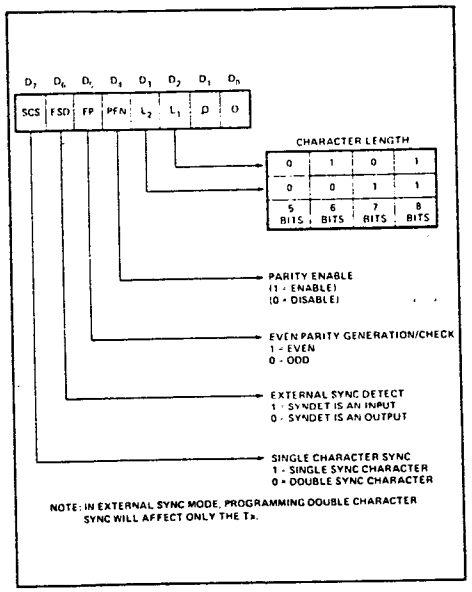


Figure 10. Mode Instruction Format, Synchronous Mode

The CPU can command the receiver to enter the HUNT mode if synchronization is lost. This will also set all the used character bits in the buffer to a "one," thus preventing a possible false SYNDET caused by data that happens to be in the Rx Buffer at ENTER HUNT time. Note that the SYNDET F/F is reset at each Status Read, regardless of whether internal or external SYNC has been programmed. This does not cause the 8251A to return to the HUNT mode. When in SYNC mode, but not in HUNT, Sync Detection is still functional, but only occurs at the "known" word boundaries. Thus, if one Status Read indicates SYNDET and a second Status Read also indicates SYNDET, then the programmed SYNDET characters have been received since the previous Status Read. (If double character sync has been programmed, then both sync characters have been contiguously received to gate a SYNDET indication.) When external SYNDET mode is selected, internal Sync Detect is disabled, and the SYNDET F/F may be set at any bit boundary.

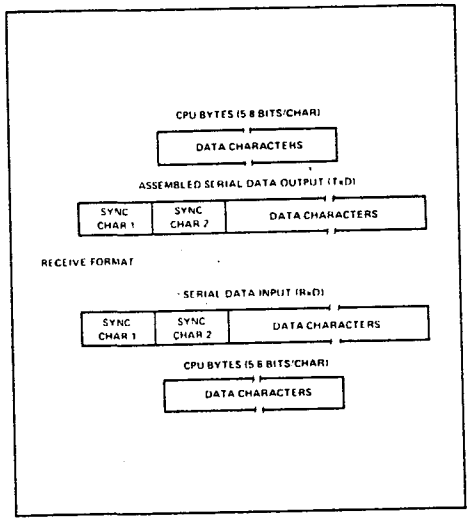


Figure 11. Data Format, Synchronous Mode

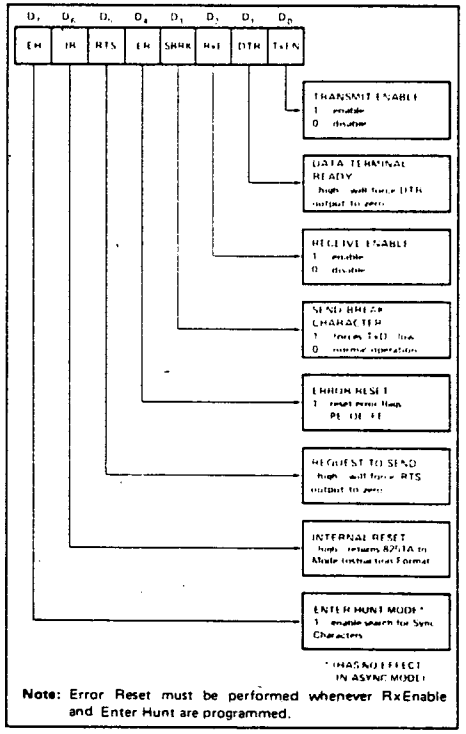
COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the sync characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command Instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, if necessary, then all further "control writes" (C/D = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

Note: Internal Reset on Power-up

When power is first applied, the 8251A may come up in the Mode, Sync character or Command format. To guarantee that the device is in the Command Instruction format before the Reset command is issued, it is safest to execute the worst-case initialization sequence (sync mode with two sync characters). Loading three 00Hs consecutively into the device with C/$\overline{D}$ = 1 configures sync operation and writes two dummy 00H sync characters. An Internal Reset command (40H) may then be issued to return the device to the "Idle" state.



Note: Error Reset must be performed whenever RxEnable and Enter Hunt are programmed.

Figure 12. Command Instruction Format

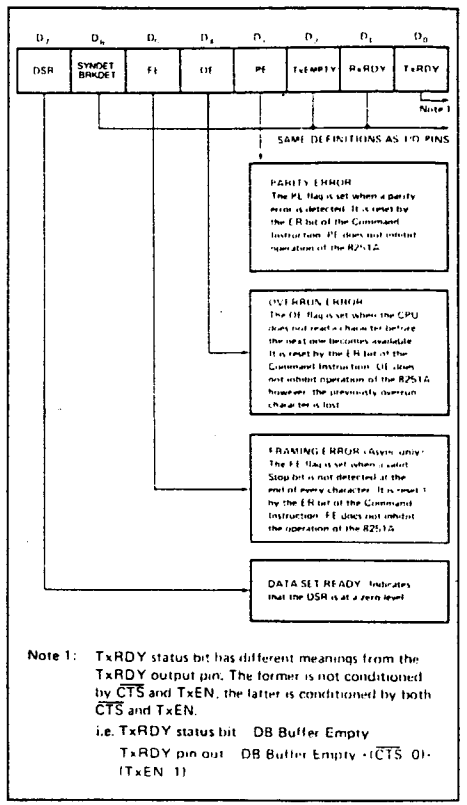
STATUS READ DEFINITION

In data communication systems it is often necessary to examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251A has facilities that allow the programmer to "read" the status of the device at any time during the functional operation. (Status update is inhibited during status read.)

A normal "read" command is issued by the CPU with C/$\overline{D}$ = 1 to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251A can be used in a completely polled or interrupt-driven environment. TxRDY is an exception.

Note that status update can have a maximum delay of 28 clock periods from the actual event affecting the status.



Note 1: TxRDY status bit has different meanings from the TxRDY output pin. The former is not conditioned by $\overline{CTS}$ and TxEN, the latter is conditioned by both $\overline{CTS}$ and TxEN.

i.e. TxRDY status bit DB Buffer Empty

TxRDY pin out DB Buffer Empty · ($\overline{CTS}$ 0) · (TxEN 1)

Figure 13. Status Read Format

APPLICATIONS OF THE 8251A

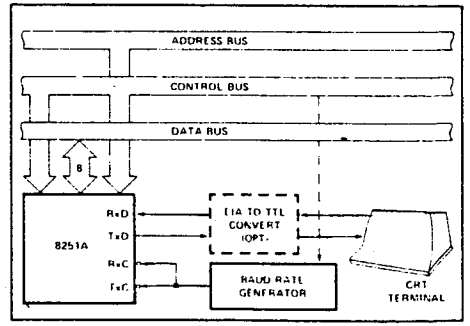


Figure 14. Asynchronous Serial Interface to CRT Terminal, DC—9600 Baud

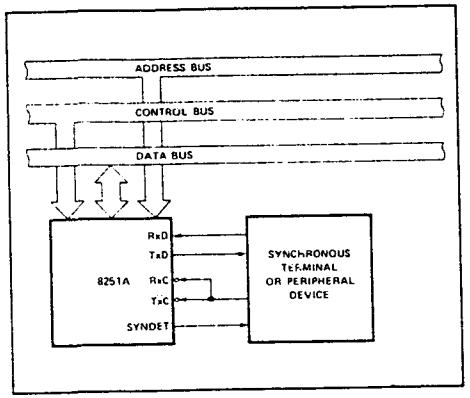


Figure 15. Synchronous Interface to Terminal or Peripheral Device

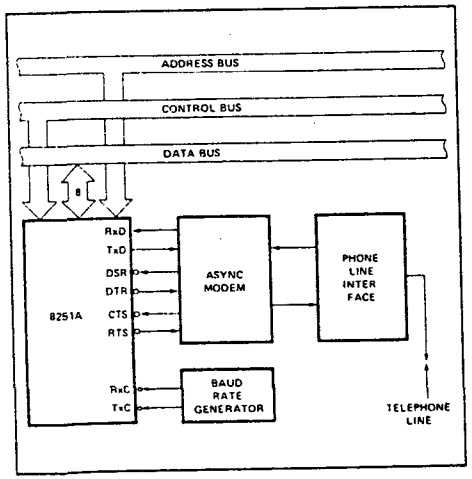


Figure 16. Asynchronous Interface to Telephone Lines

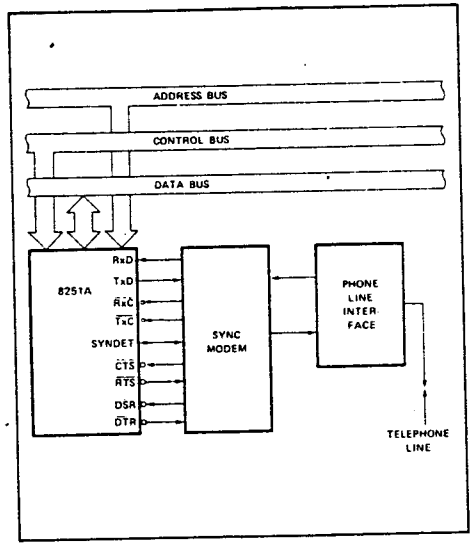


Figure 17. Synchronous Interface to Telephone Lines

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... −65°C to +150°C
Voltage On Any Pin
With Respect To Ground ..... −0.5V to +7V
Power Dissipation ..... 1 Watt

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5.0V ±10%, GND = 0V)*

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OL} = −400 μA
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} TO 0.45V
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} TO 0.45V
I_{CC}	Power Supply Current		100	mA	All Outputs = High

CAPACITANCE (T_A = 25°C, V_{CC} = GND = 0V)

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance		10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND

A.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5.0V ±10%, GND = 0V)*

Bus Parameters (Note 1)

READ CYCLE

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	Address Stable Before $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	0		ns	Note 2
t_{RA}	Address Hold Time for $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	0		ns	Note 2
t_{RR}	$\overline{READ}$ Pulse Width	250		ns	
t_{RD}	Data Delay from $\overline{READ}$		200	ns	3. C_L = 150 pF
t_{DF}	$\overline{READ}$ to Data Floating	10	100	ns	

WRITE CYCLE

Symbol	Parameter	Min.	Max.	Unit	Test Condtions
t_{AW}	Address Stable Before $\overline{WRITE}$	0		ns	
t_{WA}	Address Hold Time for $\overline{WRITE}$	0		ns	
t_{WW}	$\overline{WRITE}$ Pulse Width	250		ns	
t_{DW}	Data Set-Up Time for $\overline{WRITE}$	150		ns	
t_{WD}	Data Hold Time for $\overline{WRITE}$	20		ns	
t_{RV}	Recovery Time Between WRITES	6		t_{CY}	Note 4

A.C. CHARACTERISTICS (Continued)

OTHER TIMINGS

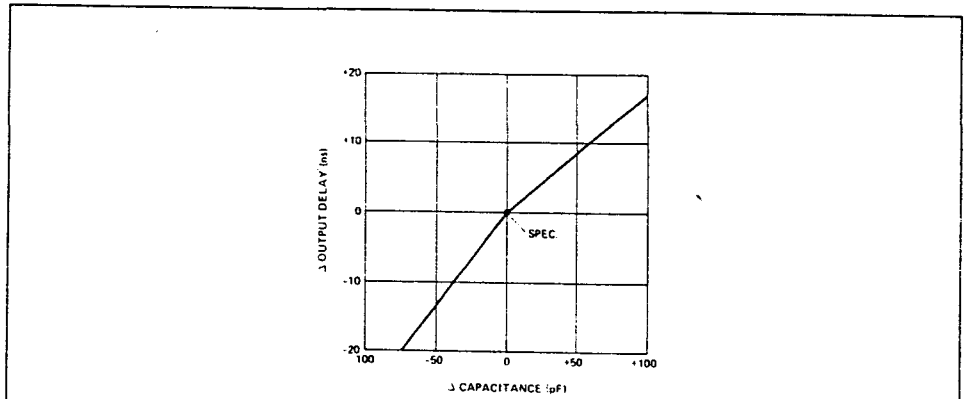
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CY}	Clock Period	320	1350	ns	Notes 5, 6
t_{ϕ}	Clock High Pulse Width	120	$t_{CY}-90$	ns	
$t_{\overline{\phi}}$	Clock Low Pulse Width	90		ns	
t_R, t_F	Clock Rise and Fall Time		20	ns	
t_{DTx}	TxD Delay from Falling Edge of $\overline{TxC}$		1	μs	
f_{Tx}	Transmitter Input Clock Frequency 1x Baud Rate 16x Baud Rate 64x Baud Rate	 DC DC DG	 64 310 615	 kHz kHz kHz	
t_{TPW}	Transmitter Input Clock Pulse Width 1x Baud Rate 16x and 64x Baud Rate	 12 1		 t_{CY} t_{CY}	
t_{TPD}	Transmitter Input Clock Pulse Delay 1x Baud Rate 16x and 64x Baud Rate	 15 3		 t_{CY} t_{CY}	
f_{Rx}	Receiver Input Clock Frequency 1x Baud Rate 16x Baud Rate 64x Baud Rate	 DC DC DC	 64 310 615	 kHz kHz kHz	
t_{RPW}	Receiver Input Clock Pulse Width 1x Baud Rate 16x and 64x Baud Rate	 12 1		 t_{CY} t_{CY}	
t_{RPD}	Receiver Input Clock Pulse Delay 1x Baud Rate 16x and 64x Baud Rate	 15 3		 t_{CY} t_{CY}	
t_{TxRDY}	TxRDY Pin Delay from Center of Last Bit		8	t_{CY}	Note 7
$t_{TxRDY\ CLEAR}$	TxRDY ↓ from Leading Edge of $\overline{WR}$		400	ns	Note 7
t_{RxRDY}	RxRDY Pin Delay from Center of Last Bit		26	t_{CY}	Note 7
$t_{RxRDY\ CLEAR}$	RxRDY ↓ from Leading Edge of $\overline{RD}$		400	ns	Note 7
t_{IS}	Internal SYNDET Delay from Rising Edge of $\overline{RxC}$		26	t_{CY}	Note 7
t_{ES}	External SYNDET Set-Up Time After Rising Edge of $\overline{RxC}$	18		t_{CY}	Note 7
$t_{TxEMPTY}$	TxEMPTY Delay from Center of Last Bit	20		t_{CY}	Note 7
t_{WC}	Control Delay from Rising Edge of WRITE (TxEn, $\overline{DTR}$, $\overline{RTS}$)	8		t_{CY}	Note 7
t_{CR}	Control to READ Set-Up Time ($\overline{DSR}$, $\overline{CTS}$)	20		t_{CY}	Note 7

*NOTE:
1. For Extended Temperature EXPRESS, use M8251A electrical parameters.

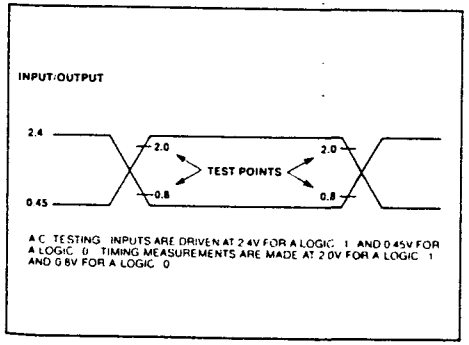
A.C. CHARACTERISTICS (Continued)

NOTES:
1. AC timings measured V_{OH} = 2.0 V_{OL} = 2.0, V_{OL} = 0.8, and with load circuit of Figure 1.
2. Chip Select (CS) and Command/Data (C/D) are considered as Addresses.
3. Assumes that Address is valid before $R_D\downarrow$.
4. This recovery time is for Mode Initialization only. Write Data is allowed only when TxRDY = 1. Recovery Time between Writes for Asynchronous Mode is 8 t_{CY} and for Synchronous Mode is 16 t_{CY}.
5. The TxC and RxC frequencies have the following limitations with respect to CLK: For 1x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(30\ t_{CY})$:
   For 16x and 64x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(4.5\ t_{CY})$.
6. Reset Pulse Width = 6 t_{CY} minimum; System Clock must be running during Reset.
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.

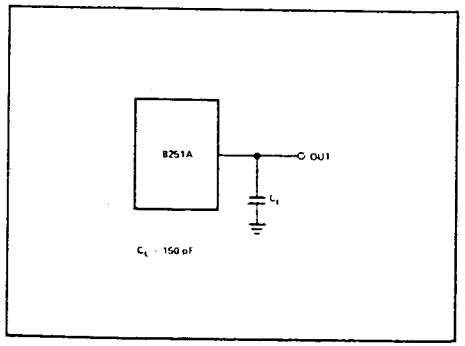
TYPICAL Δ OUTPUT DELAY VS. Δ CAPACITANCE (pF)



A.C. TESTING INPUT, OUTPUT WAVEFORM

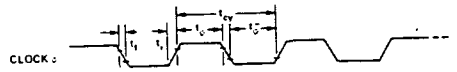


A.C. TESTING LOAD CIRCUIT

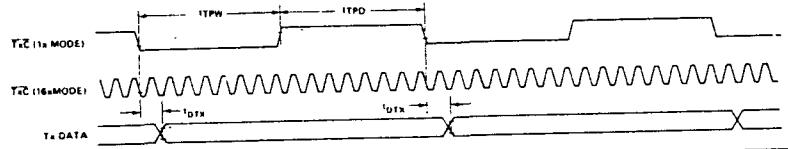


WAVEFORMS

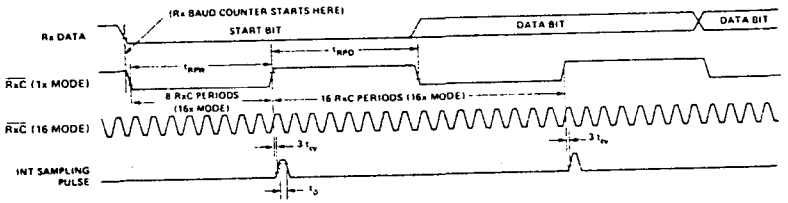
SYSTEM CLOCK INPUT



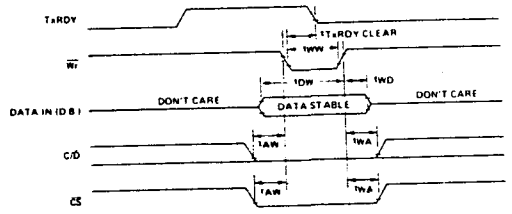
TRANSMITTER CLOCK AND DATA



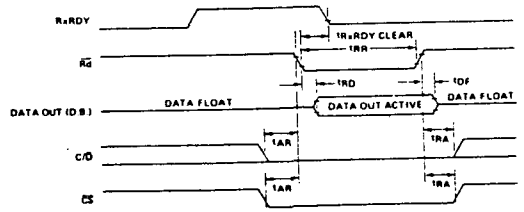
RECEIVER CLOCK AND DATA



WRITE DATA CYCLE (CPU → USART)

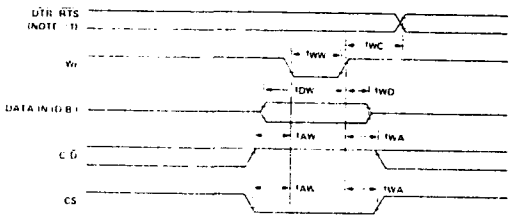


READ DATA CYCLE (CPU ← USART)

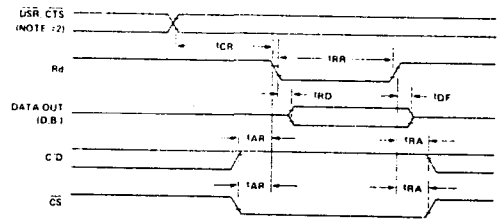


WAVEFORMS (Continued)

WRITE CONTROL OR OUTPUT PORT CYCLE (CPU → USART)

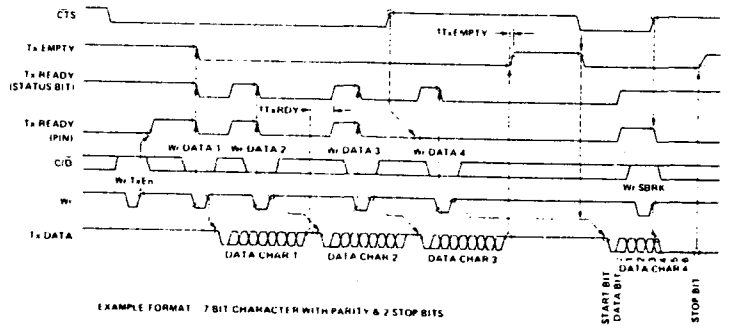


READ CONTROL OR INPUT PORT (CPU ← USART)



NOTE #1 T_{RC} INCLUDES THE RESPONSE TIMING OF A CONTROL BYTE
NOTE #2 T_{CR} INCLUDES THE EFFECT OF CTS ON THE TxENBL CIRCUITRY.

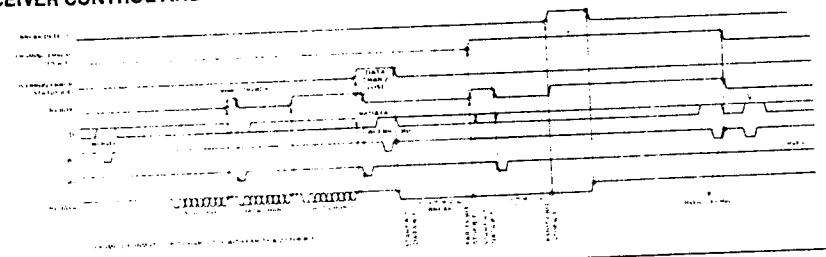
TRANSMITTER CONTROL AND FLAG TIMING (ASYNC MODE)



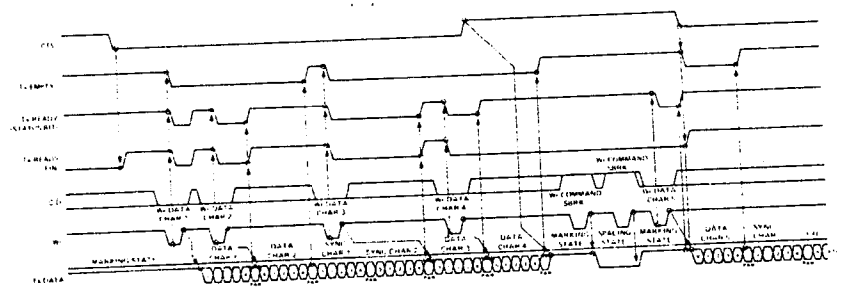
EXAMPLE FORMAT 7 BIT CHARACTER WITH PARITY & 2 STOP BITS

WAVEFORMS (Continued)

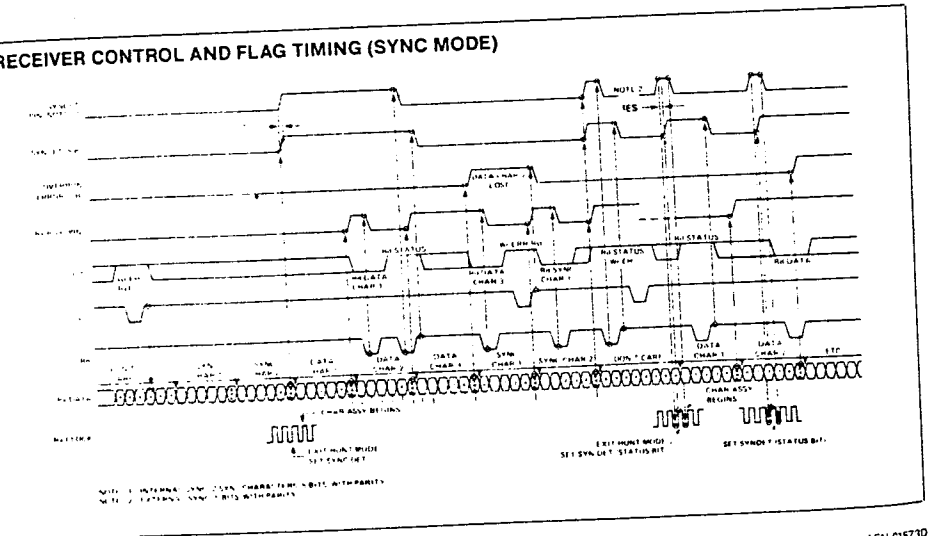
RECEIVER CONTROL AND FLAG TIMING (ASYNC MODE)



TRANSMITTER CONTROL AND FLAG TIMING (SYNC MODE)



RECEIVER CONTROL AND FLAG TIMING (SYNC MODE)



intel

8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- MCS-85™ Compatible 8253-5
- 3 Independent 16-Bit Counters
- DC to 2 MHz
- Programmable Counter Modes
- Count Binary or BCD
- Single +5V Supply
- Available in EXPRESS
  - —Standard Temperature Range
  - —Extended Temperature Range

The Intel® 8253 is a programmable counter/timer chip designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable.

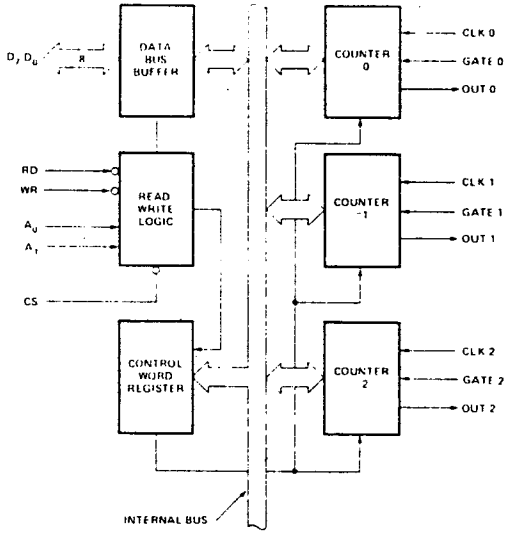


Figure 1. Block Diagram

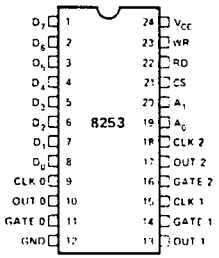
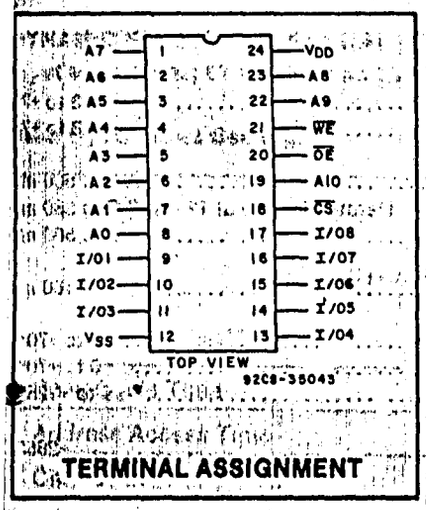


Figure 2. Pin Configuration

Preliminary Data

CDM6116-1, CDM6116-2



TERMINAL ASSIGNMENT

CMOS 2048-Word by 8-Bit LSI Static RAM

Features:

- *Fully static operation*
- *Single power supply - 4.5 V to 5.5 V*
- *All inputs and outputs directly TTL compatible*
- *3-state outputs*
- *Industry standard 24 pin configuration*
- *Input address buffers gated off with chip deselect*
- *Fast access time*
- *Low standby and operating power - I_{DDSI} = 1 µA typical, I_{OPER} = 35 mA maximum*
- *Data retention voltage - 2 V min.*
- *Operating temperature range (Max. Rating) - 0° to 70°C*

The RCA-CDM6116 is a 2048-word by 8-bit static random-access memory. It is designed for use in memory systems where high-speed, low power and simplicity in use are desirable. This type has common data input and data output and utilizes a single power supply of 4.5 V to 5.5 V.

The input address buffers are gated off with chip deselect for minimum standby power with inputs toggling.

The CDM6116 is supplied in 24-lead, hermetic, dual-in-line side-brazed ceramic (D suffix) and in 24-lead dual-in-line plastic packages (E suffix).

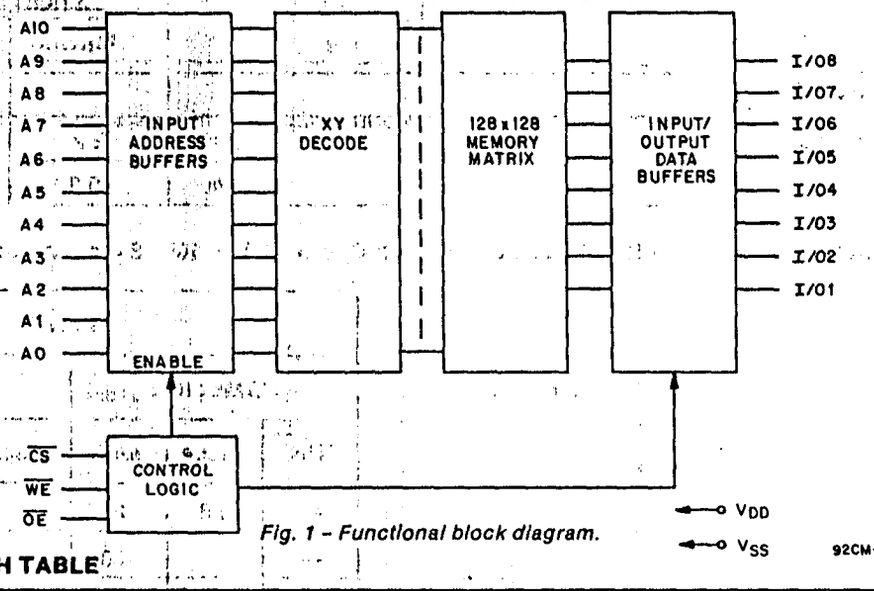


Fig. 1 - Functional block diagram.

TRUTH TABLE

$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	A0 TO A10	MODE	DATA I/O
H	X	X	X	STANDBY	HIGH Z
L	L	H	STABLE	READ	DATA OUT
L	H	L	STABLE	WRITE	DATA IN
L	L	L	STABLE	WRITE	DATA IN

L = LOW H = HIGH X = DON'T CARE

CDM6116-1, CDM6116-2

MAXIMUM RATING, *Absolute-Maximum Values:*

DC SUPPLY-VOLTAGE RANGE, (V_{DD}):
(All voltage values referenced to Vss terminal) ..... -0.3 to +7 V
INPUT VOLTAGE RANGE, ALL INPUTS ..... -0.3 to +7 V
POWER DISSIPATION PER PACKAGE (P_D):
For T_A = 0° to +60° C (PACKAGE TYPE E) ..... 500 mW
For T_A = +60 to +70° C (PACKAGE TYPE E) ..... Derate Linearly at 12 mW/° C to 380 mW
For T_A = 0° to +70° C (PACKAGE TYPE D) ..... 500 mW
DEVICE DISSIPATION PER OUTPUT TRANSISTOR
For T_A = FULL PACKAGE-TEMPERATURE RANGE ..... 100 mW
OPERATING-TEMPERATURE RANGE (T_A):
PACKAGE TYPE D ..... 0 to +70° C
PACKAGE TYPE E ..... 0 to +70° C
STORAGE TEMPERATURE RANGE (T_{stg}) ..... -55 to +125° C
LEAD TEMPERATURE (DURING SOLDERING):
At distance 1/16 ± 1/32 in. (1.59 ± 0.79 mm) from case for 10 s max. ..... +265° C

OPERATING CONDITIONS at T_A = 0° to +70° C

For maximum reliability, operating conditions should be selected so that operation is always within the following ranges:

CHARACTERISTIC		LIMITS		UNITS
		ALL TYPES		
		MIN.	MAX.	
DC Operating Voltage Range		4.5	5.5	V
Input Voltage Range	V_{IH}	2.2	V_{DD} + 0.3	V
	V_{IL}	-0.3	0.8	V

STATIC ELECTRICAL CHARACTERISTICS at T_A = 0 to +70° C, V_{DD} = 5 V ± 10%, Except as noted

CHARACTERISTIC		CONDITIONS	LIMITS						UNITS
			CDM6116-1			CDM6116-2			
			MIN.	TYP.•	MAX.	MIN.	TYP.•	MAX.	
Standby Device Current	I_{DDS}	$\overline{CS}$ = V_{IH}	—	0.6	2	—	0.6	2	mA
	I_{DDS1}	$\overline{CS}$ = V_{DD} -0.2 V	—	1	100	—	1	30	µA
Output Voltage Low-Level		I_{OL} = 2.1 mA	—	—	0.4	—	—	0.4	V
	V_{OL} Max.	I_{OL} = 1 µA	—	0.1	—	—	0.1	—	V
Output Voltage High Level		I_{OH} = -1 mA	2.4	—	—	2.4	—	—	V
	V_{OH} Min.	I_{OH} = -1 µA	—	V_{DD}-0.1	—	—	V_{DD}-0.1	—	V
Input Leakage Current	I_{IN} Max.	V_{DD} = 5.5 V V_{IN} = 0 V to V_{DD}	—	±0.1	±2	—	±0.1	±2	µA
3-State Output Leakage Current	I_{OUT}	$\overline{CS}$ or $\overline{OE}$ = V_{IH} $V_{I/O}$ = 0 V to V_{DD}	—	±0.5	±2	—	±0.5	±2	µA
Operating Device Current	I_{OPER}#	V_{IN} = V_{IL}, V_{IH}	—	20	35	—	20	35	mA
Input Capacitance	C_{IN}	V_{IN} = 0 V, f = 1 MHz, T_A = 25° C	—	4	6	—	4	6	pF
Output Capacitance	$C_{I/O}$	$V_{I/O}$ = 0 V, f = 1 MHz, T_A = 25° C	—	6	8	—	6	8	pF

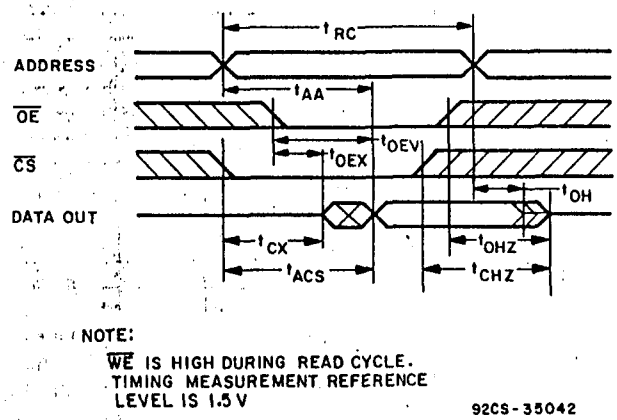
•Typical values are for T_A = 25° C and nominal V_{DD}.
#Outputs open circuited; cycle time = Min. t_{cycle}, duty = 100%

DYNAMIC ELECTRICAL CHARACTERISTICS at T_A = 0 to +70° C, V_{DD} = 5 V ± 10%, Input t_r, t_f = 10 ns; C_L = 100 pF and 1 TTL Load, Input Pulse Levels: 0.8 V to 2.4 V

Read Cycle Times See Fig. 2

CHARACTERISTIC		LIMITS				UNITS
		CDM6116-1		CDM6116-2		
		MIN.†	MAX.	MIN.†	MAX.	
Read Cycle Time	t_{RC}	250	—	200	—	ns
Address Access Time	t_{AA}	—	250	—	200	
Chip Select Access Time	t_{ACS}	—	250	—	200	
Chip Select to Output Active	t_{CX}	15	—	15	—	
Output Enable to Output Valid	t_{OEV}	—	150	—	120	
Output Enable to Output Active	t_{OEX}	15	—	15	—	
Chip Deselect to Output High Z	t_{CHZ}	0	80	0	60	
Output Disable to Output High Z	t_{OHZ}	0	80	0	60	
Output Hold from Address Change	t_{OH}	15	—	15	—	

†Time required by a limit device to allow for the indicated function.



NOTE:
WE IS HIGH DURING READ CYCLE.
TIMING MEASUREMENT REFERENCE LEVEL IS 1.5 V

92CS-35042

Fig. 2 - Read-cycle timing waveforms.

CDM6116-1, CDM6116-2

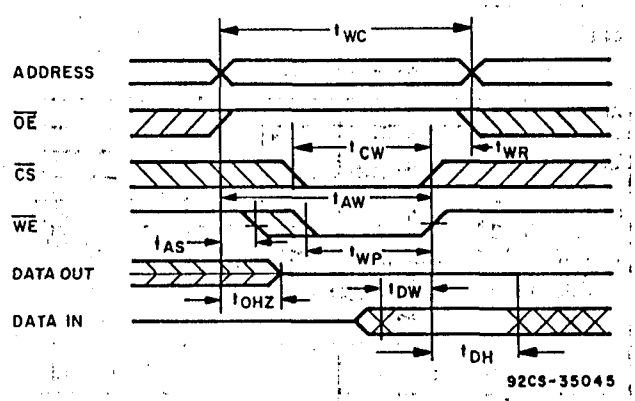
DYNAMIC ELECTRICAL CHARACTERISTICS at T_A = 0 to +70° C, V_{DD} = 5 V ± 10%, Input t_r, t_f = 10 ns; C_L = 100 pF and 1 TTL Load, Input Pulse Levels: 0.8 V to 2.4 V

Write Cycle Times See Fig. 3

CHARACTERISTIC		LIMITS				UNITS
		CDM6116-1		CDM6116-2		
		MIN.†	MAX.	MIN.†	MAX.	
Write Cycle Time	t_{WC}	250	—	200	—	ns
Chip Select to End of Write	t_{CW}	200	—	160	—	
Address Valid to End of Write	t_{AW}	200	—	160	—	
Address Setup Time	t_{AS}	0	—	0	—	
Write Pulse Width	t_{WP}	200	—	160	—	
Write Recovery Time	t_{WR}	10	—	10	—	
Output Disable to Output High Z	t_{OHZ}	0	80	0	60	
Write to Output High Z	t_{WHZ}	0	80	0	60	
Input Data Setup Time	t_{DW}	100	—	80	—	
Input Data Hold Time	t_{DH}	10	—	10	—	
Output Active from End of Write	t_{OW}	10	—	10	—	

†Time required by a limit device to allow for the indicated function.

WRITE CYCLE (1):



WRITE CYCLE (2): $\overline{OE}$ = LOW

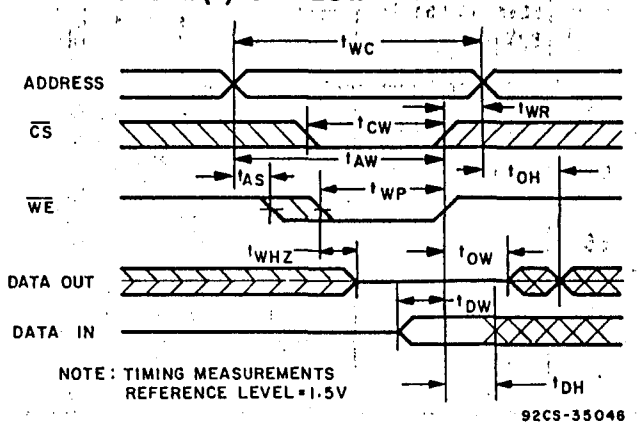


Fig. 3.- Write cycle timing waveforms.

DATA RETENTION CHARACTERISTICS at T_A = 0 to 70° C; See Fig. 4.

CHARACTERISTIC		TEST CONDITIONS	LIMITS ALL TYPES MIN.	LIMITS ALL TYPES MAX.	UNITS
Minimum Data Retention Voltage	V_{DR}	$\overline{CS} \geq V_{DD}$ -0.2 V	2	—	V
Data Retention Quiescent Current	I_{DDDR}				
	CDM6116-1	V_{DD} = 3 V, $\overline{CS} \geq$ 2.8 V	—	50	μA
	CDM6116-2	V_{DD} = 3 V, $\overline{CS} \geq$ 2.8 V	—	15	
Chip Deselect to Data Retention Time	t_{CDR}	See Fig. 4	0	—	ns
Recovery to Normal Operation Time	t_R	See Fig. 4	*t_{RC}	—	

*t_{RC} = Read Cycle Time

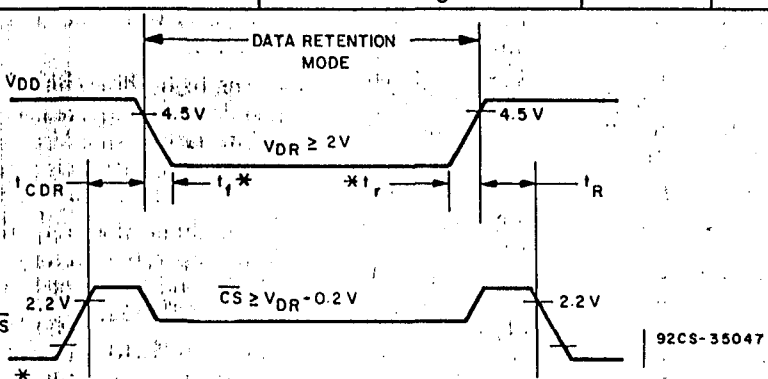


Fig. 4 – Low V_{DD} data retention timing waveforms.

OPERATING AND HANDLING CONSIDERATIONS

1. **Handling**
All inputs and outputs of RCA CMOS devices have a network for electrostatic protection during handling. Recommended handling practices for CMOS devices are described in ICAN-6525, "Guide to Better Handling and Operation of CMOS Integrated Circuits."
2. **Operating**

Operating Voltage

During operation near the maximum supply voltage limit, care should be taken to avoid or suppress power supply turn-on and turn-off transients, power supply ripple, or ground noise; any of these conditions must not cause V_{DD} — Vss to exceed the absolute maximum rating.

Input Signals

To prevent damage to the input protection circuit, input signals should never be greater than V_{DD} nor less than Vss.

Unused Inputs

A connection must be provided at every input terminal. All unused input terminals must be connected to either V_{DD} or Vss, whichever is appropriate.

Output Short Circuits

Shorting of outputs to V_{DD}, or Vss may damage CMOS devices by exceeding the maximum device dissipation.

ORDERING INFORMATION

The RCA-CDM6116 family packages, and electrical options are identified by suffix letters indicated in the following chart. When ordering a Memory/Microprocessor device, it is important that the appropriate suffix letter be affixed to the type number of the device.

Package/Option	Suffix Letter *
Dual-in-Line Side Brazed Ceramic	D
Dual-in-Line Plastic	E
Chip (When applicable)	H

Package/Option	Suffix Letter *
EVP Screening (Extra Value Program) i.e. Burn-In — optional for D, E package types	X
Electrical Option (0° to 70° C Temperature Range)	1 or 2

For example, a CDM6116 with electrical option 1, and in a dual-in-line plastic package will be identified as the CDM6116E1. A CDM6116E1 with EVP screening option will be identified as the CDM6116E1X.

*** Nomenclature Guide**

CDM	6116	E	1	X
Product Designator	Family Code	Package	Electrical Option	EVP (Optional)

National Semiconductor

MOS EPROMs
PRELIMINARY

MM2716E 16,384-Bit (2048 × 8) UV Erasable PROM
Extended Temperature Range

General Description

The MM2716E is a high speed 16k UV erasable and electrically reprogrammable EPROM ideally suited for applications where fast turn-around and pattern experimentation are important requirements.

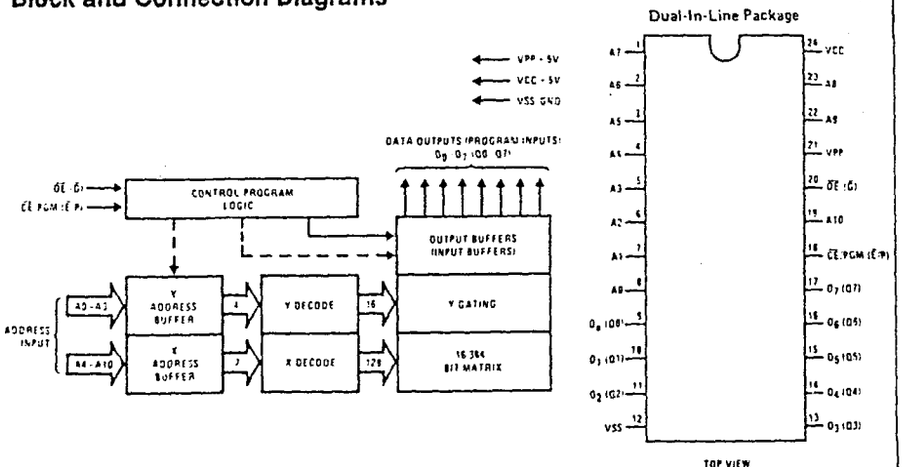
The MM2716E is packaged in a 24-pin dual-in-line package with transparent lid. The transparent lid allows the user to expose the chip to ultraviolet light to erase the bit pattern. A new pattern can then be written into the device by following the programming procedure.

This EPROM is fabricated with the reliable, high volume, time proven, N-channel silicon gate technology.

Features

- −40°C to +85°C
- 2048 x 8 organization
- 550 mW max active power, 137.5 mW max standby power
- Low power during programming
- Access time — 450 ns
- Single 5V ±10% power supply
- Static—no clocks required
- Inputs and outputs TTL compatible during both read and program modes
- TRI-STATE® output

Block and Connection Diagrams*



Dual-In-Line Package

Order Number MM2716QE
See NS Package J24CQ

Pin Connection During Read or Program

MODE	PIN NAME/NUMBER				
	$\overline{CE}$/PGM ($\overline{E}$/P) 18	$\overline{OE}$ ($\overline{G}$) 20	VPP 21	VCC 24	OUTPUTS 9–11, 13–17
Read	VIL	VIL	5	5	DOUT
Program	Pulsed VIL to VIH	VIH	25	5	DIN

Pin Names

A0–A10	Address Inputs
O_0–O_7 (O0–Q7)	Data Outputs
$\overline{CE}$/PGM ($\overline{E}$/P)	Chip Enable/Program
$\overline{OE}$ ($\overline{G}$)	Output Enable
VPP	Read 5V, Program 25V
VCC	Power (5V)
VSS	Ground

*Symbols in parentheses are proposed industry standard

Absolute Maximum Ratings (Note 1)

Temperature Under Bias	−50°C to +100°C	All Input or Output Voltages with Respect to VSS (except VPP)	6V to −0.3V
Storage Temperature	−65°C to +125°C	Power Dissipation	1.5 W
VPP Supply Voltage with Respect to VSS	26.5V to −0.3V	Lead Temperature (Soldering, 10 seconds)	300°C

READ OPERATION (Note 2)

DC Operating Characteristics

T_A = −40°C to +85°C, VCC = 5V ±10%, VPP = VCC ±0.6V (Note 3), VSS = 0V, unless otherwise noted.

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
ILI	Input Current	VIN = 5.5V or VIN = VIL			10	μA
ILO	Output Leakage Current	−VOUT = 5.5V, $\overline{CE}$/PGM = 5V			10	μA
IPP1	VPP Supply Current	VPP = 6.1V			5	mA
ICC1	VCC Supply Current (Standby)	$\overline{CE}$/PGM = VIH, $\overline{OE}$ = VIL		10	25	mA
ICC2	VCC Supply Current (Active)	$\overline{CE}$/PGM = $\overline{OE}$ = VIL		57	100	mA
VIL	Input Low Voltage		−0.1		0.8	V
VIH	Input High Voltage		2.0		VCC + 1	V
VOH	Output High Voltage	IOH = −400 μA	2.4			V
VOL	Output Low Voltage	IOL = 2.1 mA			0.45	V

AC Characteristics (Note 4)

T_A = −40°C to +85°C, VCC = 5V ±10%, VPP = VCC ±0.6V (Note 3), VSS = 0V, unless otherwise noted.

SYMBOL		PARAMETER	CONDITIONS	MIN	MAX	UNITS
ALTERNATE	STANDARD					
t_{ACC}	TAVQV	Address to Output Delay	$\overline{CE}$/PGM = $\overline{OE}$ = VIL		450	ns
t_{CE}	TELQV	$\overline{E}$ to Output Delay	$\overline{OE}$ = VIL		450	ns
t_{OE}	TGLQV	Output Enable to Output Delay	$\overline{CE}$/PGM = VIL		120	ns
t_{DF}	TGHQZ	Output Enable High to Output Hi-Z	$\overline{CE}$/PGM = VIL	0	100	ns
t_{OH}	TAXQX	Address to Output Hold	$\overline{CE}$/PGM = $\overline{OE}$ = VIL	0		ns
t_{OD}	TEHQZ	$\overline{E}$ to Output Hi-Z	$\overline{OE}$ = VIL	0	100	ns

Capacitance (Note 5)

T_A = 25°C, f = 1 MHz

SYMBOL	PARAMETER	CONDITIONS	TYP	MAX	UNITS
CI	Input Capacitance	VIN = 0V	4	6	pF
CO	Output Capacitance	VOUT = 0V	8	12	pF

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Typical conditions are for operation at: T_A = 25°C, VCC = 5V, VPP = VCC, and VSS = 0V.

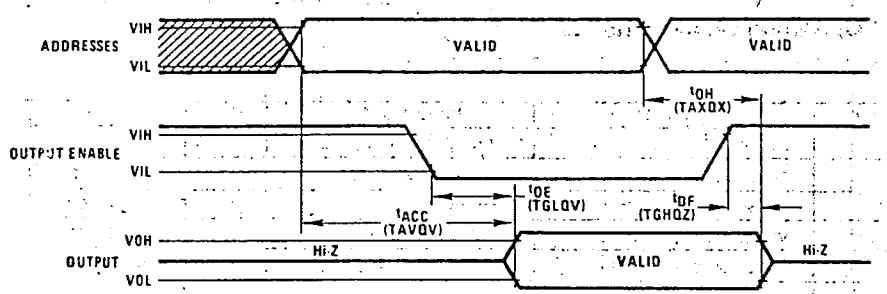
Note 3: VPP may be connected to VCC except during program. The ±0.6V tolerance allows a circuit to switch VPP between the read voltage and the program voltage.

Note 4: Output load: 1 TTL gate and CL = 100 pF. Input rise and fall times ≤ 20 ns.

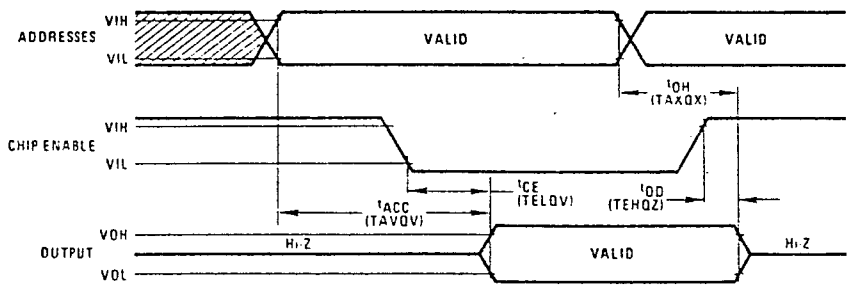
Note 5: Capacitance is guaranteed by periodic testing.

Switching Time Waveforms*

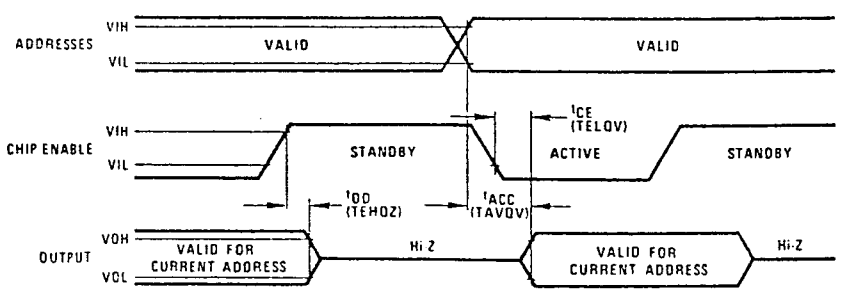
Read Cycle ($\overline{CE}$/PGM = VIL)



Read Cycle ($\overline{OE}$ = VIL)



Standby Power Down Mode ($\overline{OE}$ = VIL)



*Symbols in parentheses are proposed industry standard

PROGRAM OPERATION

DC Electrical Characteristics and Operating Conditions (Notes 1 and 2)

(T_A = 25°C ±5°C) (VCC = 5V ±5%, VPP = 25V ±1V)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS
ILI	Input Leakage Current (Note 3)			10	μA
VIL	Input Low Level	−0.1		0.8	V
VIH	Input High Level	2.0		VCC + 1	V
ICC	VCC Power Supply Current			100	mA
IPP1	VPP Supply Current (Note 4)			5	mA
IPP2	VPP Supply Current During Programming Pulse (Note 5)			30	mA

AC Characteristics and Operating Conditions (Notes 1, 2, and 6)

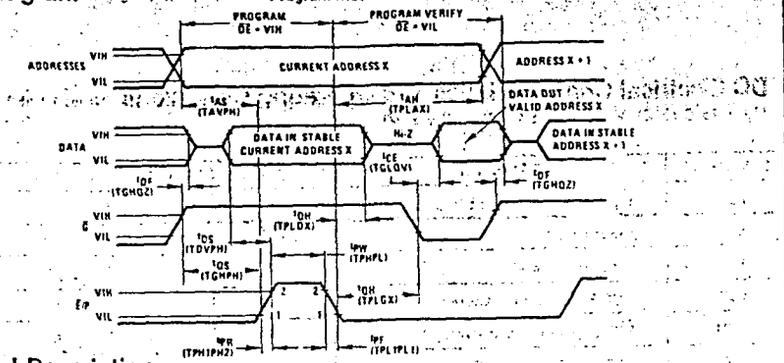
(T_A = 25°C ±5°C) (VCC = 5V ±5%, VPP = 25V ±1V)

SYMBOL		PARAMETER	MIN	TYP	MAX	UNITS
ALTERNATE	STANDARD					
tAS	TAVPH	Address Setup Time	2			μs
tOS	TGHPH	$\overline{OE}$ Setup Time	2			μs
tDS	TDVPH	Data Setup Time	2			μs
tAH	TPLAX	Address Hold Time	2			μs
tOH	TPLGX	$\overline{OE}$ Hold Time	2			μs
tDH	TPLDX	Data Hold Time	2			μs
tDF	TGHQZ	Chip Disable to Output Float Delay (Note 4)	0		100	ns
tCE	TGLQV	Chip Enable to Output Delay (Note 4)			120	ns
tPW	TPHPL	Program Pulse Width	45	50	55	ms
tPR	TPH1PH2	Program Pulse Rise Time	5			ns
tPF	TPL2PL1	Program Pulse Fall Time	5			ns

Note 1: VCC must be applied at the same time or before VPP and removed after or at the same time as VPP. To prevent damage to the device it must not be inserted into a board with power applied.
Note 2: Care must be taken to prevent overshoot of the VPP supply when switching to +25V.
Note 3: 0.45V ≤ VIN ≤ 5.25V.
Note 4: $\overline{CE}$/PGM = VIL, VPP = VCC + 0.6V.
Note 5: VPP = 26V.
Note 6: Transition times ≤ 20 ns unless noted otherwise.

Timing Diagram*

Program Mode



Note: VPP = 25V

Functional Description

DEVICE OPERATION

The MM2716E has 3 modes of operation in the normal system environment. These are shown in Table I.

Read Mode

The MM2716E read operation requires that $\overline{OE}$ = VIL, $\overline{CE}$/PGM = VIL and that addresses A0–A10 have been stabilized. Valid data will appear on the output pins after t_{ACC}, t_{OE}, t_{CE} times (see Switching Time Waveforms) depending on which is limiting.

Deselect Mode

The MM2716E is deselected by making $\overline{OE}$ = VIH. This mode is independent of $\overline{CE}$ PGM and the condition of the addresses. The outputs are Hi-Z when $\overline{OE}$ = VIH. This allows OR-tying 2 or more MM2716Es for memory expansion.

Standby Mode (Power Down)

The MM2716E may be powered down to the standby mode by making $\overline{CE}$/PGM = VIH. This is independent of $\overline{OE}$ and automatically puts the outputs in their Hi-Z state. The power is reduced to 25% (150 mW max) of the normal operating power. VCC and VPP must be maintained at 5V. Access time at power up remains either t_{ACC} or t_{CE} (see Switching Time Waveforms).

PROGRAMMING

The MM2716E is shipped from National completely erased. All bits will be at a "1" level (output high) in this initial state and after any full erasure. Table II shows the 3 programming modes.

TABLE I. OPERATING MODES (VCC = VPP = 5V)

MODE	PIN NAME/NUMBER		
	$\overline{CE}$/PGM ($\overline{E}$/P) 18	$\overline{OE}$ ($\overline{G}$) 20	OUTPUTS 9–11, 13–17
Read	VIL	VIL	DOUT
Deselect	Don't Care	VIH	Hi-Z
Standby	VIH	Don't Care	Hi-Z

TABLE II. PROGRAMMING MODES (VCC = 5V)

MODE	PIN NAME/NUMBER			
	$\overline{CE}$/PGM ($\overline{E}$/P) 18	$\overline{OE}$ ($\overline{G}$) 20	VPP 21	OUTPUTS O 9–11, 13–17
Program	Pulsed VIL to VIH	VIH	25	DIN
Program Verify	VIL	VIL	25(5)	DOUT
Program Inhibit	VIL	VIH	25	Hi-Z

*Symbols in parentheses are proposed industry standard

Functional Description (Continued)

Program Mode

The MM2716E is programmed by introducing "0"s into the desired locations. This is done 8 bits (a byte) at a time. Any individual address, a sequence of addresses, or addresses chosen at random may be programmed. Any or all of the 8 bits associated with an address location may be programmed with a single program pulse applied to the chip enable pin. All input voltage levels, including the program pulse on chip-enable are TTL compatible. The programming sequence is:

With VPP = 25V, VCC = 5V, $\overline{OE}$ = VIH and $\overline{CE}$/PGM = VIL, an address is selected and the desired data word is applied to the output pins. (VIL = "0" and VIH = "1" for both address and data.) After the address and data signals are stable the program pin is pulsed from VIL to VIH with a pulse width between 45 ms and 55 ms.

Multiple pulses are not needed but will not cause device damage. No pins should be left open. A high level (VIH or higher) *must not* be maintained longer than $t_{PW(MAX)}$ on the program pin during programming. MM2716Es may be programmed in parallel with the same data in this mode.

Program Verify Mode

The programming of the MM2716E may be verified either 1 word at a time during the programming (as shown in the timing diagram) or by reading all of the words out at the end of the programming sequence. This can be done with VPP = 25V (or 5V) in either case.

Program Inhibit Mode

The program inhibit mode allows programming several MM2716Es simultaneously with different data for each one by controlling which ones receive the program pulse. All similar inputs of the MM2716E may be paralleled. Pulsing the program pin (from VIL to VIH) will program a unit while inhibiting the program pulse to a unit will keep it from being programmed and keeping $\overline{OE}$ = VIH will put its outputs in the Hi-Z state.

ERASING

The MM2716E is erased by exposure to high intensity ultraviolet light through the transparent window. This exposure discharges the floating gate to its initial state through induced photo current. It is recommended that the MM2716E be kept out of direct sunlight. The UV content of sunlight may cause a partial erasure of some bits in a relatively short period of time. Direct sunlight (any intense light) can cause temporary functional failure due to generation of photo current. Extended exposure to room level fluorescent lighting will also cause erasure. An opaque coating (paint, tape, label, etc.) should be placed over the package window if this product is to be operated under these lighting conditions.

An ultraviolet source of 2537 Å yielding a total integrated dosage of 15 watt-seconds/cm^2 is required. This will erase the part in approximately 15 to 20 minutes if a UV lamp with a 12,000 μW/cm^2 power rating is used. The MM2716E to be erased should be placed 1 inch away from the lamp and no filters should be used.

An erasure system should be calibrated periodically. The distance from lamp to unit should be maintained at 1 inch. The erasure time is increased by the square of the distance (if the distance is doubled the erasure time goes up by a factor of 4). Lamps lose intensity as they age. When a lamp is changed, the distance is changed, or the lamp is aged, the system should be checked to make certain full erasure is occurring. Incomplete erasure will cause symptoms that can be misleading. Programmers, components, and system designs have been erroneously suspected when incomplete erasure was the basic problem.

LINE CIRCUITS

µA1488

QUAD LINE DRIVER

FAIRCHILD LINEAR INTEGRATED CIRCUITS

TTL IN => RS OUT

GENERAL DESCRIPTION – The µA1488 is an EIA RS-232C specified Quad Line Driver. This device is used to interface data terminals with data communications equipment. The µA1488 is a pin-for-pin replacement of the MC1488.

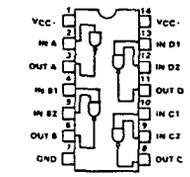
- **CURRENT LIMITED OUTPUT – ± 10 mA TYP**
- **POWER-OFF SOURCE IMPEDANCE – 300 Ω MIN**
- **SIMPLE SLEW RATE CONTROL WITH EXTERNAL CAPACITOR**
- **FLEXIBLE OPERATING SUPPLY RANGE**

ABSOLUTE MAXIMUM RATINGS (at 25°C unless otherwise noted)

Power Supply Voltages	
$V_{CC}+$	+15 V
$V_{CC}-$	−15 V
Input Voltage Range (V_{IR})	−15 V DC to +7.0 V DC
Output Signal Voltage	±15 V DC
Continuous Total Power Dissipation (Note 1)	800 mW
Operating Temperature Range	0°C to 70°C
Pin Temperature	−65°C to +150°C
Hermetic DIP (Soldering, 60 s)	300°C
Molded DIP (Soldering, 10 s)	260°C

Note 1: Above 60°C ambient temperatures, derate linearly at 8.3 mW/°C.

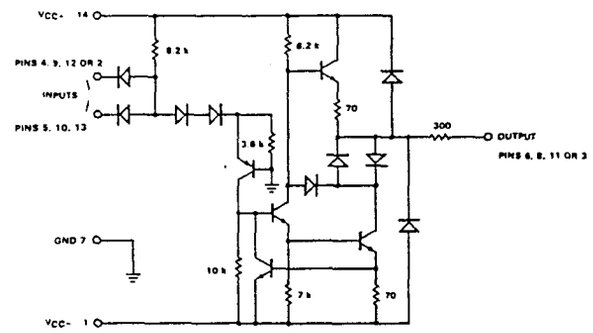
CONNECTION DIAGRAM
14-PIN DIP
(TOP VIEW)
PACKAGE OUTLINE 6A 9A
PACKAGE CODE D P



ORDER INFORMATION

TYPE	PART NO.
µA1488	µA1488DC
µA1488	µA1488PC

CIRCUIT SCHEMATIC (1/4 OF CIRCUIT SHOWN)



ELECTRICAL CHARACTERISTICS: V_{CC+} = +9.0 V ± 1%, V_{CC-} = −9.0 V ± 1%, T_A = 0 to +70°C, unless otherwise noted.

SYMBOL	CHARACTERISTICS	CONDITIONS	FIG.	MIN	TYP	MAX	UNITS
I_{IL}	Input LOW Current	V_{IL} = 0	1		1.0	1.6	mA
I_{IH}	Input HIGH Current	V_{IH} = 5.0 V	1			10	μA
V_{OH}	Output HIGH Voltage	V_{IL} = 0.8 V, R_L = 3.0 kΩ, V_{CC+} = +9.0 V, V_{CC-} = −9.0 V	2	+6.0	+7.0		V
		V_{IL} = 0.8 V, R_L = 3.0 kΩ, V_{CC+} = +13.2 V, V_{CC-} = −13.2 V	2	+9.0	+10.5		
V_{OL}	Output LOW Voltage	V_{IH} = 1.9 V, R_L = 3.0 kΩ, V_{CC+} = +9.0 V, V_{CC-} = −9.0 V	2	−6.0	−7.0		V
		V_{IH} = 1.9 V, R_L = 3.0 kΩ, V_{CC+} = +13.2 V, V_{CC-} = −13.2 V	2	−9.0	−10.5		
I_{OS+}	Positive Output Short-Circuit Current	V_{IL} = 0.8 V (Note 1)	3	+6.0	+10	+12	mA
I_{OS-}	Negative Output Short-Circuit Current	V_{IH} = 1.9 V (Note 1)	3	−6.0	−10	−12	mA
R_{OUT}	Output Resistance	V_{CC+} = V_{CC-} = 0 V, V_O = ±2.0 V	4	300			Ω
I_{CC+}	Positive Supply Current	R_L = ∞ V_{IH} = 1.9 V, V_{CC+} = +9.0 V V_{IL} = 0.8 V, V_{CC-} = +9.0 V V_{IH} = 1.9 V, V_{CC+} = +12 V V_{IL} = 0.8 V, V_{CC+} = +12 V V_{IH} = 1.9 V, V_{CC+} = +15 V V_{IL} = 0.8 V, V_{CC+} = +15 V	5		+15 +4.5 +19 +5.5	+20 +6.0 +25 +7.0 +34 +12	mA
I_{CC-}	Negative Supply Current	R_L = ∞ V_{IH} = 1.9 V, V_{CC-} = −9.0 V V_{IL} = 0.8 V, V_{CC-} = −9.0 V V_{IH} = 1.9 V, V_{CC-} = −12 V V_{IL} = 0.8 V, V_{CC-} = −12 V V_{IH} = 1.9 V, V_{CC-} = −15 V V_{IL} = 0.8 V, V_{CC-} = −15 V	5		−13 −18	−17 −15 −23 −15 −34 −2.5	mA μA mA μA mA mA
P_C	Power Consumption	V_{CC+} = 9.0 V, V_{CC-} = −9.0 V V_{CC+} = 12 V, V_{CC-} = −12 V				333 578	mW

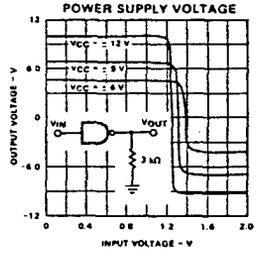
AC CHARACTERISTICS: V_{CC+} = +9.0 V ± 1%, V_{CC-} = −9.0 V ± 1%, T_A = 25°C

SYMBOL	CHARACTERISTICS	CONDITION	FIG.	MIN	TYP	MAX	UNITS
t_{PLH} t_{PHL}	Propagation Delay Time	R_L = 3.0 kΩ, C_L = 15 pF	6		220 70	350 175	ns
t_f t_r	Fall Time Rise Time	R_L = 3.0 kΩ, C_L = 15 pF	6		70 55	75 100	ns

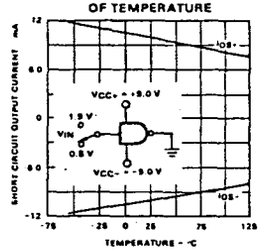
NOTE 1: Maximum Package Power Dissipation may be exceeded if all outputs are shorted simultaneously.

TYPICAL CHARACTERISTICS
T_A = +25°C unless otherwise noted

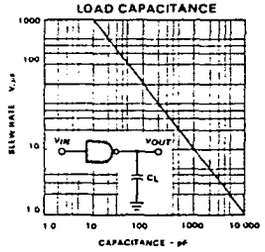
TRANSFER CHARACTERISTICS AS A FUNCTION OF POWER SUPPLY VOLTAGE



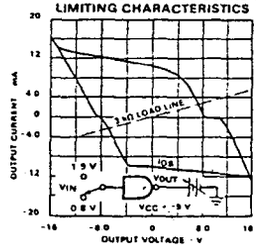
SHORT CIRCUIT OUTPUT CURRENT AS A FUNCTION OF TEMPERATURE



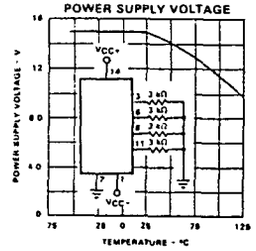
OUTPUT SLEW RATE AS A FUNCTION OF LOAD CAPACITANCE



OUTPUT VOLTAGE AND CURRENT LIMITING CHARACTERISTICS



MAXIMUM OPERATING TEMPERATURE AS A FUNCTION OF POWER SUPPLY VOLTAGE

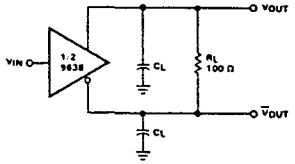


AC CHARACTERISTICS

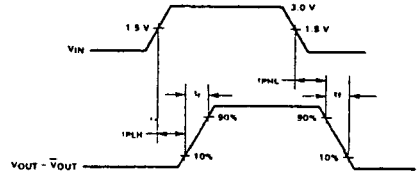
SYMBOL	CHARACTERISTICS	CONDITIONS	MIN	TYP	MAX	UNITS
t_{PHL} t_{PLH}	Propagation Delay	T_A = 25°C, C_L = 15 pF (Note 2), R_L = 100 Ω, See Fig. 2		10 10	15 15	ns ns
t_f	Fall Time, 90% – 10%			10	15	ns
t_r	Rise Time, 10% – 90%			10	15	ns
$t_{PA} - t_{PB}$	Skew Between Outputs A and B			1		ns

AC TEST CIRCUIT AND VOLTAGE WAVEFORM

AC TEST CIRCUIT



VOLTAGE WAVEFORM



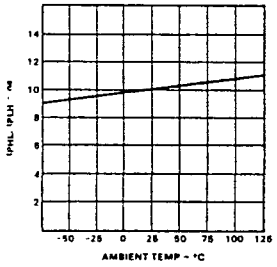
NOTES

1 The pulse generator has the following characteristics
Z_{OUT} = 50 Ω, PRR = 500 kHz
t_w = 100 ns, t_r = t_f = 5 ns

2 C_L includes probe and jig capacitance.

Fig. 2

TYPICAL DELAY CHARACTERISTICS



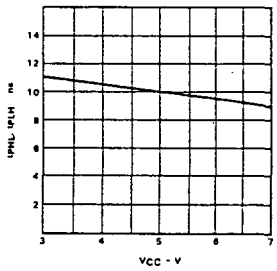


FIG. 3

μA1489 • μA1489A

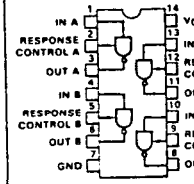
QUAD LINE RECEIVERS

FAIRCHILD LINEAR INTEGRATED CIRCUITS

GENERAL DESCRIPTION – The μA1489 and the μA1489A are EIA RS-232C specified Quad Line Receivers. These devices are used to interface data terminals with data communications equipment. The μA1489 and μA1489A are pin-for-pin replacements of the MC1489 and MC1489A respectively.

- INPUT RESISTANCE – 3.0 kΩ TO 7.0 kΩ
- INPUT SIGNAL RANGE – ±30 V
- INPUT THRESHOLD HYSTERESIS BUILT IN
- RESPONSE CONTROL
  - a) LOGIC THRESHOLD SHIFTING
  - b) INPUT NOISE FILTERING

CONNECTION DIAGRA
14-PIN DIP
(TOP VIEW)
PACKAGE OUTLINES 6A
PACKAGE CODES D



ABSOLUTE MAXIMUM RATINGS

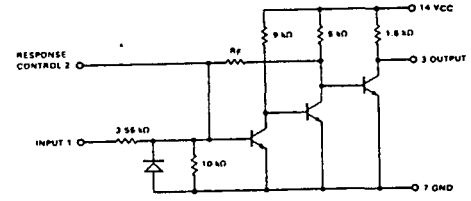
Power Supply Voltage	+10 Vdc
Input Voltage Range	±30 Vdc
Output Load Current	20 mA
Continuous Total Power Dissipation (Note 1)	800 mW
Operating Temperature	0°C to 70°C
Storage Temperature	−65°C to +175°C
Pin Temperature	
Hermetic DIP (Soldering, 60 s)	300°C
Molded DIP (Soldering, 10 s)	260°C

Note 1 Above 60°C ambient temperature, derate linearly at 8.3 mW/°C

ORDER INFORMATI

TYP	PART NO
μA1489	μA1489D
μA1489	μA1489P
μA1489A	μA1489A
μA1489A	μA1489A

CIRCUIT SCHEMATIC (1/4 OF CIRCUIT SHOWN)



	μA1489	μA1489A
Rf	10 kΩ	2 kΩ

ELECTRICAL CHARACTERISTICS: V_{CC} = 5.0 V ±1%, Response control pin is open, T_A = 0°C to 70°C unless otherwise noted.

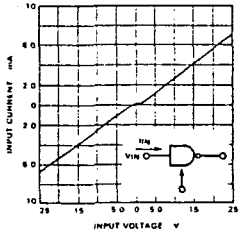
SYMBOL	CHARACTERISTICS	CONDITIONS		FIG	MIN	TYP	MAX	UNITS
I_{IH}	Positive Input Current	V_{IH} = 25 V V_{IH} = 3.0 V		1	3.6 0.43		8.3	mA
I_{IL}	Negative Input Current	V_{IL} = −25 V V_{IL} = −3.0 V		1	−3.6 −0.43		−8.3	mA
V_{IHL}	Input Turn-on Threshold Voltage	T_A = 25°C, $V_{OL} \leq$ 0.45 V	μA1489 μA1489A	2	1.0 1.75	 1.95	1.5 2.25	V
V_{ILH}	Input Turn off Threshold Voltage	T_A = 25°C, $V_{OH} \geq$ 2.5 V, I_L = −0.5 mA	μA1489 μA1489A	2	0.75 0.75	 0.8	1.25 1.25	V
V_{OH}	Output HIGH Voltage	V_{IH} = 0.75 V, I_L = −0.5 mA Input open circuit, I_L = −0.5 mA		2	2.6	4.0	5.0	V
V_{OL}	Output LOW Voltage	V_{IL} = 3.0 V, I_L = 10 mA		2		0.2	0.45	V
I_{OS}	Output Short-circuit Current			3		3.0		mA
I_{CC}	Power Supply Current	V_{IH} = 5.0 V		4		20	26	mA
P_C	Power Consumption	V_{IH} = 5.0 V		4		100	130	mW

AC CHARACTERISTICS: V_{CC} = 5.0 V ±1%, T_A = 25°C

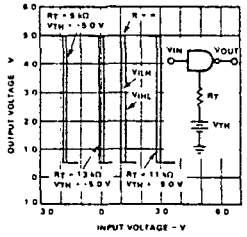
SYMBOL	CHARACTERISTICS	CONDITIONS	FIG.	MIN	TYP	MAX	UNITS
t_{PLH}	Propagation Delay Time	R_L = 3.9 kΩ	5		25	85	ns
t_{PHL}		R = 390 Ω			25	50	
t_r	Rise Time	R_L = 3.9 kΩ	5		120	175	ns
t_f	Fall Time	R_L = 390 Ω			10	20	

TYPICAL PERFORMANCE CURVES

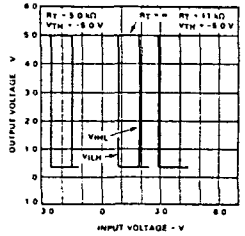
INPUT CURRENT AS A FUNCTION OF INPUT VOLTAGE



μA1489 INPUT THRESHOLD VOLTAGE ADJUSTMENT



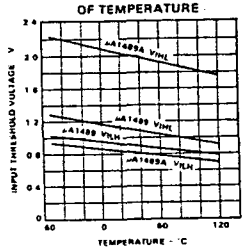
μA1489A INPUT THRESHOLD VOLTAGE ADJUSTMENT



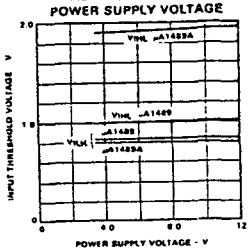
TEST CIRCUIT SAME AS μA1489

TYPICAL PERFORMANCE CURVES (Cont'd)

INPUT THRESHOLD VOLTAGE AS A FUNCTION OF TEMPERATURE



INPUT THRESHOLD VOLTAGE AS A FUNCTION OF POWER SUPPLY VOLTAGE



DC TEST CIRCUITS

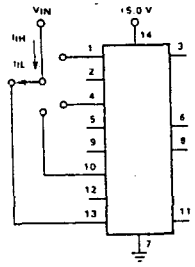


Fig. 1. Input Current

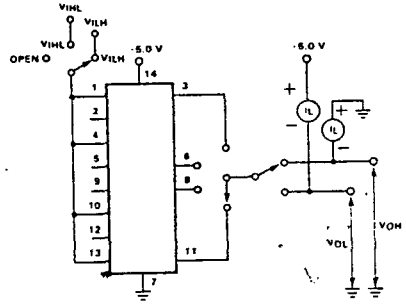


Fig. 2. Output Voltage and Input Threshold Voltage

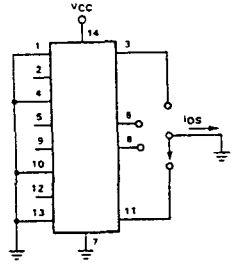


Fig. 3. Output Short-Circuit Current

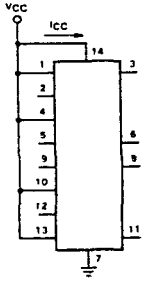


Fig. 4. Power Supply Current

BIBLIOGRAFIA.-

INTEL Corporation:

- ISIS II System User's Guide.
- 8Ø8Ø/8Ø85 Macroassembler Operator's Manual
- 8Ø8Ø/8Ø85 Assembly Language.
- PLM/8Ø Compiler Operator's Manual.
- PLM/8Ø Language.
- ISIS II - Credit - CRT Based Text Editor User's Guide.
- ICE85. In Circuit Emulator Operating Instructions for ISIS II Users.
- Microsystems Components Handbook.

- El libro del RS-232. Ed. Anaya.
- Smartwork. Wintek Corporation.
- Sistema Operativo MS-DOS. Guía del usuario. Ed. Osborne - Mc Graw Hill.
- DXY Plotter Operation Manual. Roland Corporation.
- Radio Shack TRS-8Ø. Microcomputer System. Line Printer V.
- TTL Data Book. Fairchaild.
- GWBASIC. Guía del usuario. Microsoft Corporation.