MEMORIA

UNIVERSIDAD: ESCUELA UNIVERSITARIA POLITECNICA DE LAS PALMAS.

TITULO: "ANALISIS DEL PROGRAMA MONITOR DEL SDK'85".

AUTOR: GINES MEDINA TORRES.

TUTOR: FELIX HERNANDEZ CABRERA.

LAS PALMAS A 4 DE JUNIO DE 1983.

INDICE

PAGINA

PAGINA

OBJETO DEL PROYECTO.

Este Proyecto ha sido realizado a petición de la ESCUELA UNIVERSITARIA POLITECNICA, para la obtención del Título de Ingeniero Técnico de Telecomunicaciones.

Con la realización de éste Proyecto se pretende hacer un estudio sobre el funcionamiento de un microprocesador, asi como de la forma de programar un sistema con microprocesador.

Para dicho estudio se ha elegido la placa SDK'85, (debido a su existencia como instrumento de prácticas en la E.U.P.) y se ha hecho un analisis completo de su Programa Monitor.

Para una mayor comprensión del analisis hecho se aconseja ir viendo cada COMANDO ó RUTINA en su organigrama (paso a paso), y en el lenguaje ensamblador (anexo 2).

.....

2-DESCRIPCION DEL SDK"85.

El SDK"85 es un completo sistema microcomputador formado con la CPU 8085. Está montado en forma de placa y contiene todos los elementos necesarios para construir un útil sistema funcional. Está pensado y montado de tal forma que el usuario puede ampliar las funciones de la placa con sólo añadir más circuitos a su bus, que está preparado para ello.

El SDK'85 se comunica con el mundo exterior por medio de un teclado y de un display (está preparado también para ser conectado a un terminal TTY).

El teclado está formado por las siguientes teclas:

.- 0 dígito 0.

.- 1 dígito 1.

.- 2 dígito 2.

.- 3 dígito 3.

.- 4 dígito 4 y 8 bits de mayor peso del SP.

.- 5 dígito 5 y 8 bits de menor peso del SP.

.- 6 dígito 6 y 8 bits de mayor peso del CP.

.- 7 dígito 7 y 8 bits de menor peso del CP.

.- 8 dígito 8 y registro H.

.- 9 dígito 9 y registro L.

.- A dígito A y registro A (Acumulador).

.- B dígito B y registro B.

.- C dígito C y registro C.

.- D dígito D y registro D.

.- E dígito E y registro E.

.- F dígito F y máscara de interrupciones.

.- RESET — Al pulsarse, comienza el programa monitor.

.- GO — Permite al usuario ejecutar su programa, junto con el comando EXEC.

.- S.STEP — Ejecuta el programa paso a paso.

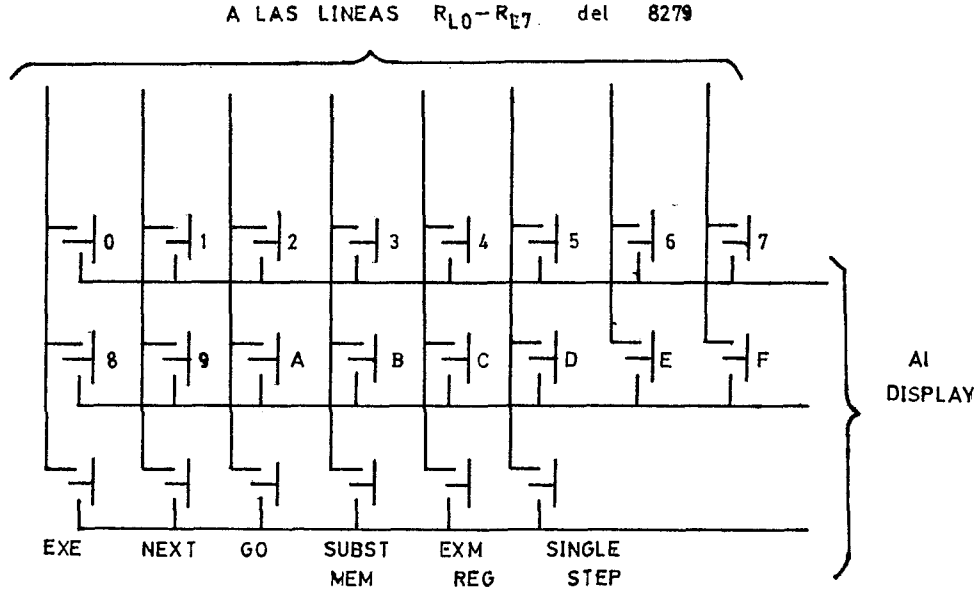
.- S.MEMORY — Permite al usuario examinar y modificar el contenido de las direcciones de memoria.

.- EXAMINE REG. — Permite al usuario examinar y modificar el contenido de los registros de la CPU.

.- V.INTERRUPT — Sirve al usuario como botón de interrupción.

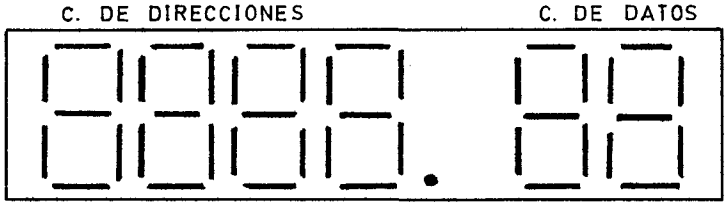
.- NEXT — Al pulsarse, aparece en el display la próxima dirección.

La conección del teclado se realiza de la forma siguiente:

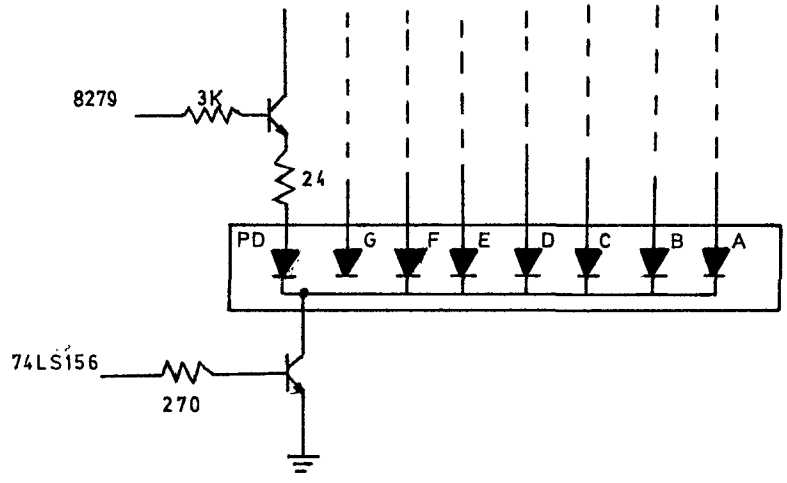


La tecla RESET va unida a la entrada $\overline{\text{RESET IN}}$ del 8085 y la V.INTERRUPT a la patilla 7 (RST 7.5) del 8085.

El display está formado por 6 unidades, cuatro para el campo de direcciones y dos para el campo de datos, de la forma indicada a continuación:

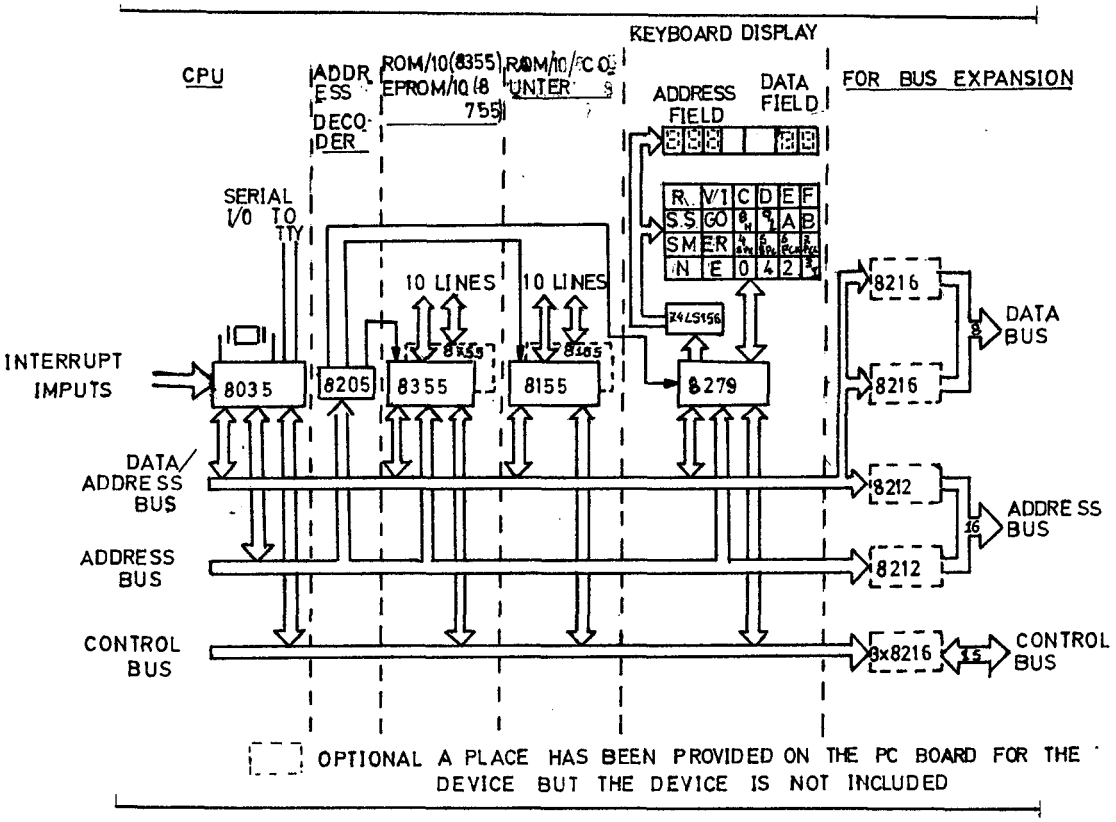


Debido a que los niveles de salida tanto del 8279 como los del 74LS156 (decodificador) no son suficientes para excitar al desplay, se hará uso del siguiente sistema básico:



Esto constituye un sólo dígito, el esquema total serán seis como éste.

En la figura siguiente podemos ver el diagrama de bloques del SDK'85:



2.1.-DISPOSITIVOS USADOS EN EL SDK'85.

En este apartado se tratan las características generales de los distintos dispositivos usados en el montaje básico de la placa (SDK'85).Para poder profundizar en cada uno de ellos es necesario consultar los manuales del fabricante (ver Anexo 1).

Entre los dispositivos usados tenemos:

- El 8279.
- El 8085.
- El 8205.
- El 8155.
- El 74LS156.
- El 8755.
- El 8212.
- El 8216.
- El 8355.

2.1.1.-CARACTERISTICAS DEL 8279.

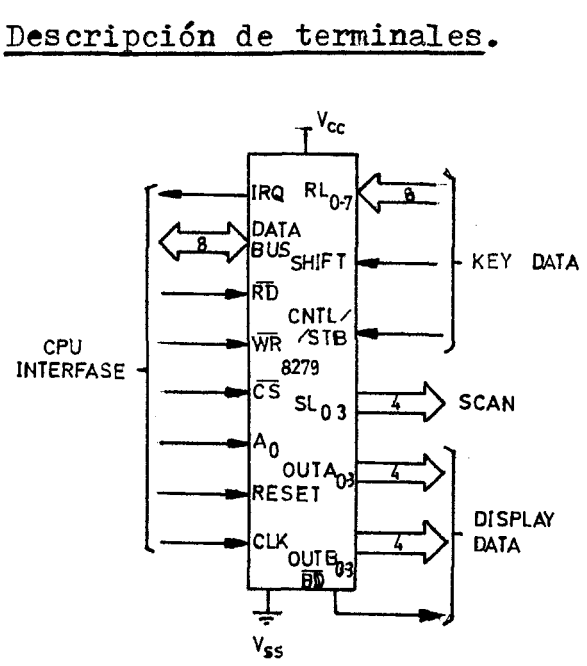
El 8279 de Intel es un interfase programable para teclado y display de aplicación general.

+La parte del teclado puede proporcionar un barrido para una matriz de 64 teclas de contacto. Las entradas del teclado son almacenadas en una FIFO (memoria, primera entrada-primera salida) de 8 caracteres .Si entran más de 8, se activa el estado de desbordamiento. Cuando se produce una entrada desde el teclado, se activa una salida de interrupción (RST 5.5) hacia la CPU, para indicarle que se ha pulsado una tecla.(terminal IRQ del chip).

+La parte del display proporciona un barrido de display para LED incandescentes ó para otras tecnologías de display populares.

El 8279 posee una memoria RAM para el display de 16 x8, la cuál puede ser organizada en dos de 16x4. Esta RAM puede ser escrita o leída por la CPU. La entrada de dígitos al display se puede hacer por la derecha o por la izquierda, según el formato elegido. La lectura ó la escritura en la RAM se pueden hacer de tal forma que se auto incrementen las direcciones de la RAM.

Descripción de terminales.



.- DB_0-DB_7: Bus de datos bi-direccional. Todos los datos y comandos son transmitidos por estas líneas.

.- CIK : Terminal por el que entra el reloj usado por el sistema para que el chip genere su propio reloj interno.

.- RESET : Una señal alta en ésta línea resetea al 8279.Esta señal procede del microprocesador.

.- $\overline{CS}$: Una señal baja en esta línea, habilita el

chip para su utilización.La señal procede del decodificador de direcciones.

.- A_0: Un nivel alto en ésta línea le indica al chip que las señales de entrada o salida son interpretadas como un comando ó un estado. Un nivel bajo indica que las señales son datos.

.- $\overline{RD}$, $\overline{WR}$: Entrada/Salida. Lectura/Escritura.Estas señales habilitan el buffer de datos para enviar datos al bus externo ó para recibir datos de él.

.- IRQ: Terminal usado para indicarle a la CPU que se ha producido una entrada desde el teclado. La línea está a nivel alto siempre que exista información en la RAM del display.

.- SL_0-SL_3: Líneas de Selección, usadas para indicardequélínea recibe o toma los datos el sistema.

.- RL_0-RL_7: Líneas de entrada por medio de las cuales tienen acceso a la unidad las órdenes dadas desde el teclado.

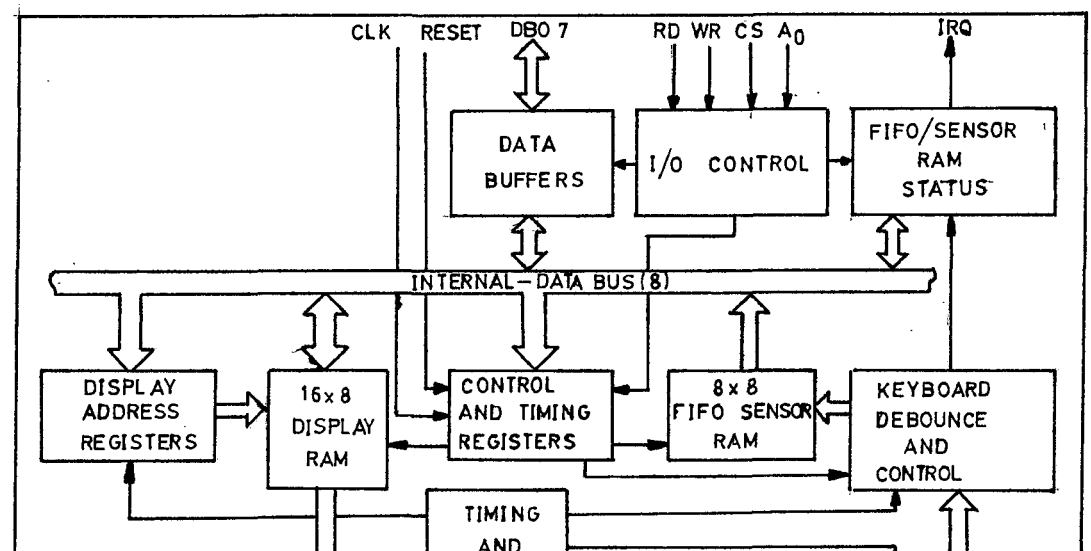
.- SHIFT: Señal de control para el desplazamiento.

.- CNTL/STB: Señal de control para la entrada.

.- OUT A_0-OUT A_3 y OUT B_0-OUT B_3: Estos dos puertos de salida son usados para restaurar el contenido del display. El dato de salida de éstas líneas se sincroniza con la salida de las líneas de selección (SL_0-SL_3) para multiplexar los dígitos del display.Los dos puertos,de cuatro bits cada uno,pueden ser limpiados independientemente.

.-$\overline{BD}$:Esta salida es usada para limpiar el display mediante una tecla ó mediante un comando.

El diagrama de bloques del 8279 lo podemos ver en la figura siguiente:



Para controlar las I/O del chip se usan las patillas $\overline{CS}$, A_0, $\overline{RD}$ y $\overline{WR}$. Con $\overline{CS}$ se habilita todo dato que fluye hacia ó desde el 8279. Con A_0 se identifica si la información (que llega ó sale) es un comando o un estado ó un dato(A_0=0). $\overline{RD}$ y $\overline{WR}$ determinan la dirección en que va el dato dentro del buffer de datos. Si $\overline{RD}$ es activa, la información sale del 8279. Si es $\overline{WR}$ activa, la información entra en el 8279.

En el bloque "Control and Timing Registers" se almacenan los modos en que trabaja el teclado/display y otras condiciones de operación programadas por la CPU. Los modos de operación se programan poniendo el comando deseado en las líneas de datos, (con A_0=1) y $\overline{WR}$ activa.

El reloj de control contiene la cadena básica de contadores. Es un divisor por "N" que puede ser programado para producir una frecuencia interna de 100KHz, dando unos 5'1 ms. de tiempo de lectura del teclado.

El "Scan Counter" (contador de barridos) posee dos modos de operación:

El modo "codificado" donde el contador proporciona una cuenta binaria que debe ser decodificada externamente para proporcionar las líneas de barrido para el teclado/display.

En el modo "codificado" el contador decodifica los dos bits más significativos y proporciona de 1 a 4 barridos , según la decodificación.Es decir,que en este modo solamente pueden aparecer en el display los 4 primeros caráceteres de la RAM.

El bloque "FIFO/Sensor RAM Status" tiene una doble función como RAM (8x8). En los modos de"teclado"o "entrada strobed" es una FIFO (cada nueva entrada es escrita en sucesivas posiciones de la RAM y cada una es leída en el orden de entrada). El estado de la FIFO almacena el número de carácteres que ésta posee y cuando está llena ó vacia.Demasiadas lecturas ó escrituras serán reconocidas como un error. El estado puede ser leído con un $\overline{RD}$, con $\overline{CS}$ bajo y A_0 alto. El estado lógico también proporciona una señal IRQ cuando la FIFO no está vacia.

En el modo "Scanned Sensor Matrix" la memoria es una"Sensor RAM" (cada fila de la "Sensor RAM" se carga con el estado de la corespondiente fila del sensor en "Sensor matrix"). En este modo IRQ es alta si se detecta un cambio

en el sensor.

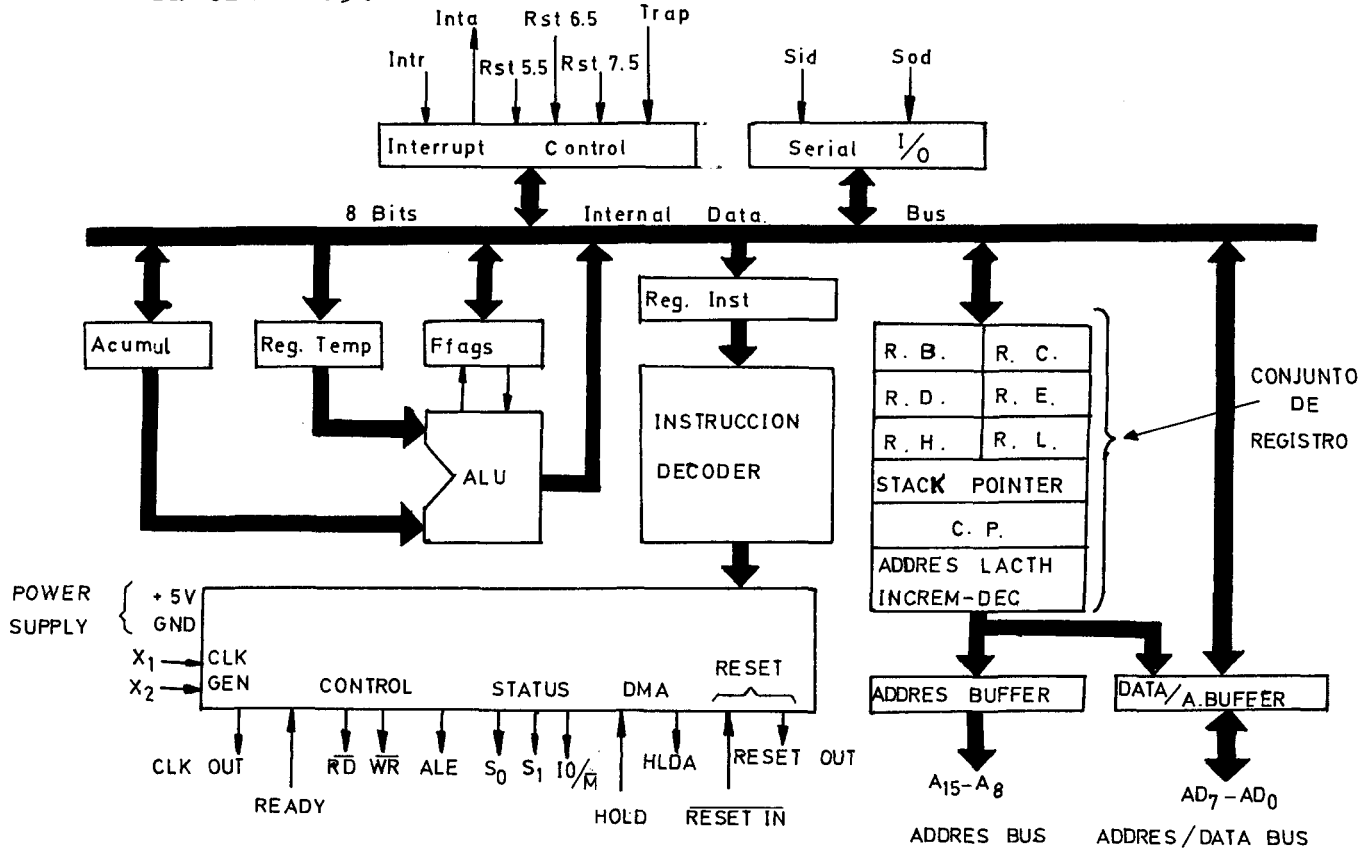
En los bloques "Display Address Registers" y en "Display RAM" se mantienen las direcciones de las palabras que en ese momento están siendo escritas o leídas por la CPU. La "Display RAM" puede ser directamente leída desde la CPU después de que sean programados el modo y la dirección correctas.

Para profundizar más en este dispositivo ver las características del fabricante, en el anexo 1.

2.12 MICROPROCESADOR 8085.

Dispositivo basado en el microprocesador 8080, pero con mejores características. La CPU 8085 genera las señales de control apropiadas para seleccionar dispositivos externos, así como las funciones que ejercen. Puede acceder a 256 direcciones de entrada/salida las cuales van desde la 00 hasta la FF. Transfiere los datos por medio de un bus tri-state bidireccional (AD_{0-7}), multiplexado en el tiempo, para poder transmitir por él los ocho bits menos significativos de una dirección de 16 bits, siendo A_{8-15} las líneas que completan el bus de direcciones (con ésto la capacidad de memoria llega hasta 64 Kbytes).

La figura siguiente nos muestra donde se localizan las distintas líneas y elementos específicos de que costa la CPU 8085:



Cuenta con 8 registros de 8 bits, de los cuales 6 pueden emplearse de tal forma que constituyan 3 de 16 bits.

Posee además 2 registros de 16 bits.

Entre estos registros tenemos:

.- Acumulador: (8 bits) Es el elemento central en las operaciones aritméticas y de transferencia de datos con la memoria o con los puertos Entrada/Salida (I/O).

.- El Contador de Programa: (16 bits) Apunta siempre la dirección de memoria que contiene el código de la próxima instrucción a ejecutar.

.- El Puntero de Stack (SP): (16 bits) Indica la posición de memoria donde automaticamente empezarían a cargarse los puntos de retorno (al producirse una interrupción ó llamada a subrutina.)

.- Existen además tres parejas de registros BC, DE, HL que pueden usarse como registro de 8 bits o de 16 bits, según se desee. La pareja HL puede funcionar como puntero de datos para especificar el origen o destino de algunos datos en ciertas instrucciones.

Dentro de la CPU tenemos otro bloque constituido por 5 flip/flops, para indicaciones especiales.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

registro de estados.

.- Flag de arrastre (CY-Carry): Se fija o repone por efecto del tipo de resultado que ofrezca la ejecución de ciertas instrucciones.

El bit de arrastre auxiliar (AC-Auxiliary Carry): Se pone a 1 cuando se produce arrastre en el cuarto bit. El conocimiento de este bit es muy útil en operaciones aritmé-

ticas en modo BCD.

El flag de signo (S-Sign): Indica el signo del resultado de una operación.

El bit de indicación de 0 (Z-Zero): Se fija siempre que el resultado de una operación sea cero.

El bit de paridad (P-Parity): Se pone 1 siempre que el número de unos del acumulador sea par si no se pone cero.

La ALU (Unidad Aritmética-Lógica) realiza las instrucciones aritméticas, lógicas y de desplazamiento y rotación de bits. Los resultados de estas operaciones se depositan en el acumulador o en el bus de datos internos para una manipulación posterior por parte de la Unidad de Control. Está formada por:

.- Un acumulador de 8 bits y el bit de arrastre.

.- Un acumulador temporal y el bit de arrastre temporal.

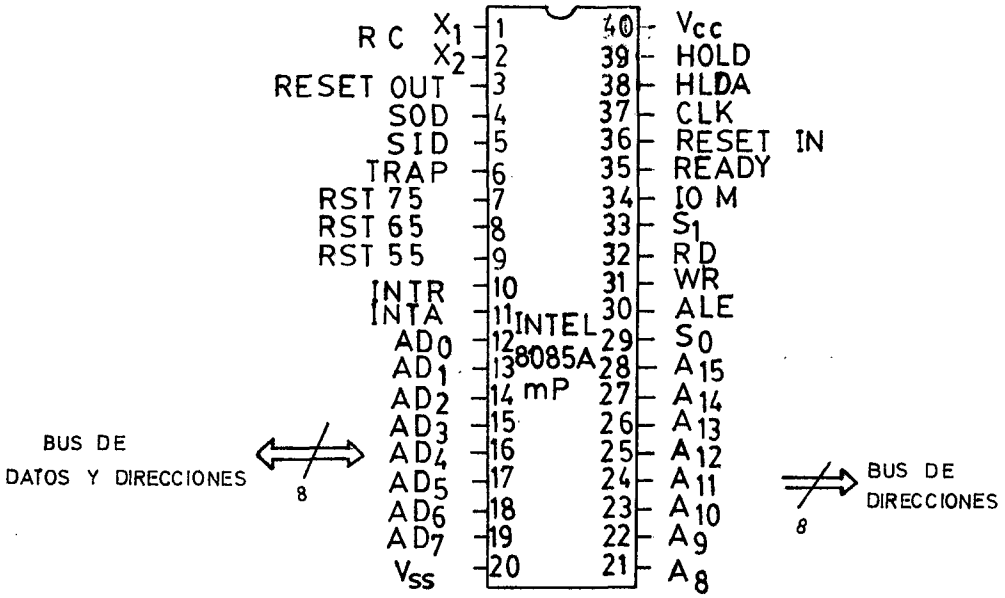
.- El registro de estados.

Registro de Instrucción y Decodificador.

Durante un ciclo de búsqueda el primer byte de una instrucción (el código de operación) se transfiere desde el bus interno al registro de instrucción (de 8 bits). A su vez el contenido de registro de instrucciones está disponible a la entrada del decodificador de instrucciones.

Las señales de salida del decodificador controlan los registros de la CPU, la ALU, y los buffers de dirección y de datos. Es decir, la salidas del decodificador de instrucción y del generador interno de fases del reloj generan los ciclos de estados y de máquina que comprenden los ciclos de instrucción.

DESCRIPCION DE TERMINALES.



.- $A_8 - A_{15}$: Bus de direcciones. Constituyen la mitad más significativa de las líneas que constituirán el bus de direcciones.

.- $AD_0 - AD_7$: Bus de direcciones y de datos. Estas 8 líneas cumplen una doble función, a veces constituyen la mitad menos significativa del bus de direcciones y otras veces forman el bus de datos. Están multiplexadas - en el tiempo para hacerlas funcionar de una u otra forma.

.- RD - WR (Read - Write). Informan al sistema del sentido de las transferencias de datos así como del instante en que deben producirse o sea que temporizan las operaciones de transferencia. Si $\overline{RD}$ está a 0 la operación es de lectura ó deentrada dedatos en un puerto I/O. Si $\overline{WR}$ está a 0 la operación es de escritura en memoria o de salida de datos.

.- IO/M (I/O/MEMORY). Indica qué es lo que está en comunicación con la memoria o el conjunto de puertos - I/O. Si I/O/M está a 1 la comunicación es con el subsistema I/O, en caso contrario es con la memoria.

.- ALE (ADDRESS LATCH ENABLE). Habilita la re-

tención dela información que llevan las líneas AD_0-AD_7 cuando se refiere a una dirección.

.- READY. Indica al micro que existen datos válidos en el bus de datos procedentes de la memoria o de los puertos I/O.

.- HOLD. Cuando está a nivel alto la CPU entra en estado de retención. Los buses de datos y dirección se ponen en estado de alta impedancia. Es útil en operaciones de acceso directo a memoria.

.- HLDA (HOLD ACKNOWLEDGE). Señal que avisa al sistema que el micro está en estado HOLD.

.- $\overline{\text{RESET IN}}$. Señal deentrada al micro necesaria para la iniciación dela CPU.

.- RESET OUT. Indica al sistema que seha recibido una señal de $\overline{\text{RESET IN}}$. Se emplea para la puesta a cero de los dispositivos de interfase.

.- CLK OUT (CLOCK OUT). Señal de salida réplica de la señal de reloj interna.

.- S_0,S_1. Son dos líneas de identificación de estados (al igual que IO/$\overline{\text{M}}$.)

Entradas de interrupciones: El 8085 proporciona 5 entradas deinterrupción: INTR, RST 5.5, RST 6.5, RST 7.5, y TRAP.

.- INTR (INTERRUPT REQUEST). Es usada como una interrupción de propósitos generales. El estado de ésta entrada se muestrea sólamente durante el ciclo de reloj siguiente al último deun ciclo de instrucción y durante los estados HOLD y HALT. Si cuando se muestrea es activa, el CP no se incrementa y se produce una señal de reconocimiento por la línea $\overline{\text{INTA}}$ (INTERRUPT ACKNOWLEDGE).

En este ciclo puede insertarse una instrucción RESTART o CALL para transferir el control del programa a la rutina de servicio de la interrupción. La posibilidad de interrumpir por ésta línea se controla por programa mediante el empleo de las instrucciones EI y DI.

.- RST 5.5, RST 6.5, RST 7.5, (RESTART INTERRUPTS). Estas tres señales de entrada tienen un tratamiento semejante a la anterior excepto que no precisan la introducción en el bus de datos externo del código que permite la vectorización o búsqueda de la dirección de comienzo de la subrutina de servicio. Los vectores que tienen asociados son:

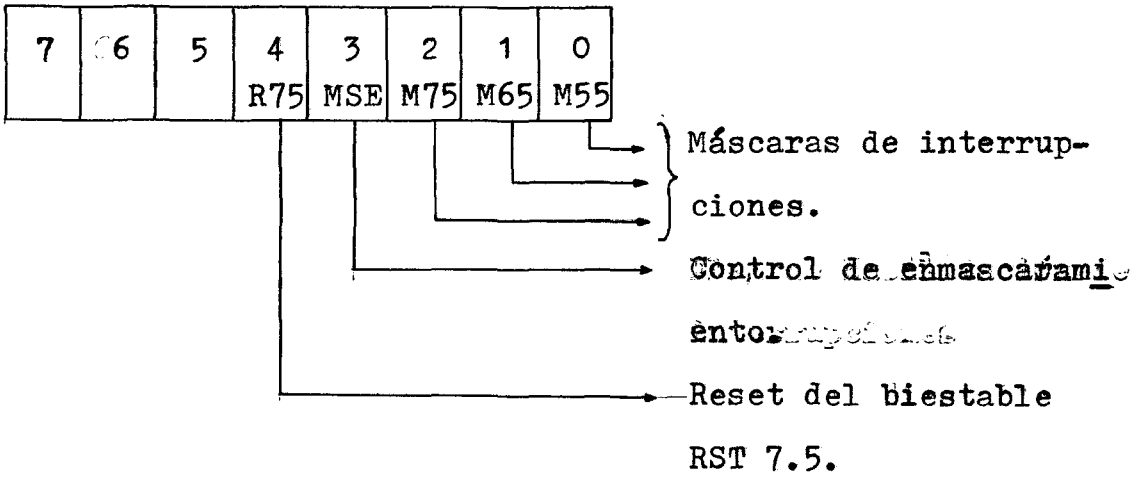
RST 5.5 2C.

RST 6.5 34.

RST 7.5 36.

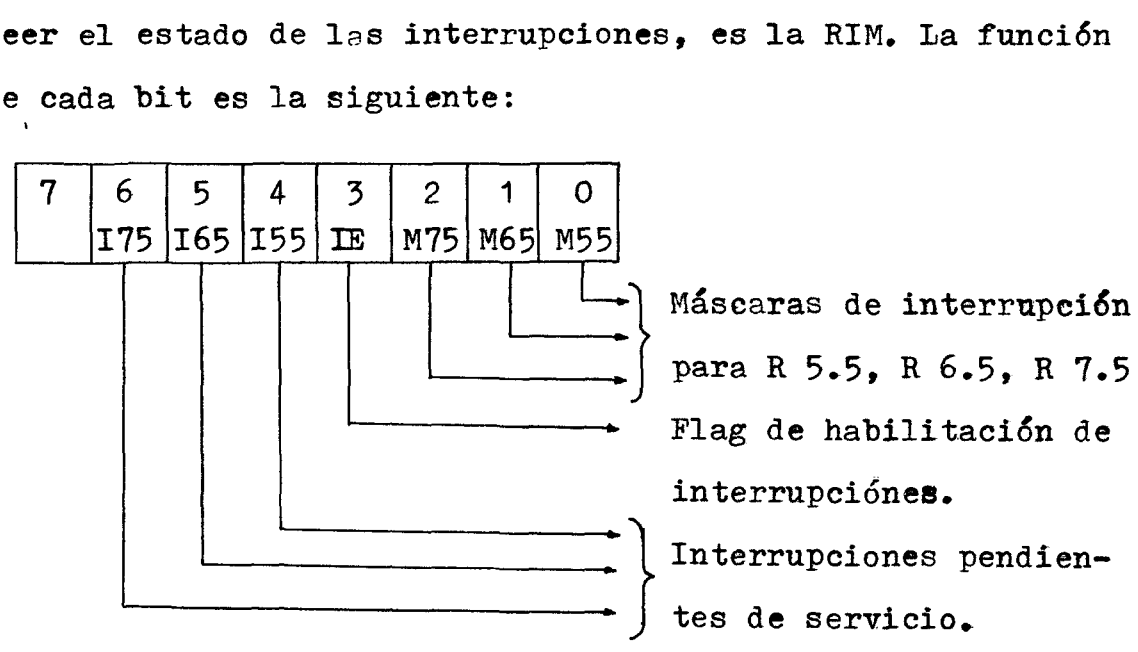
La diferencia entre ellas está en la distinta prioridad y en el modo de activarlas. R 7.5 se activa con un flanco de subida (la petición se retiene en un biestable). R 6.5 tiene que mantenerse la entrada a nivel alto hasta que el micro ha muestrado el estado de esta línea. R 5.5 es igual que la anterior.

Estas tres interrupciones pueden enmascararse empleando la instrucción SIM. Esta habilita o deshabilita las interrupciones en base al contenido de ciertos indicadores. Para ejecutar una instrucción SIM tenemos primero que cargar el acumulador con una información determinada (dependiendo de lo que se quiera hacer). Cada bit del acumulador tiene una misión:



Para programar las máscaras de instrucción tenemos primero que fijar el bit 3 y luego ponemos a 1 los bits correspondientes a las interrupciones que se quierán deshabilitar, luego se ejecuta la instrucción SIM y quedan preparadas las interrupciones.Si fijamos el bit 4, al ejecutar SIM se limpia el biestable R 7.5.

En el 8085 tenemos otra instrucción que nos permite leer el estado de las interrupciones, es la RIM. La función de cada bit es la siguiente:



Las instrucciones EI y DI habilitan y deshabilitan respectivamente las interrupciones INTR, R 5.5 y R 6.5. La R 7.5 no está sujeta a los efectos de estas instrucciones.

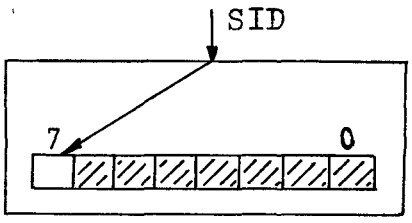
Si tenemos el bit 3 (IE) a 1 las interrupciones INTR, R 5.5 y R 6.5 podrán atenderse siempre que sus correspondientes máscaras estén a 0. Los bits 4, 5 y 6 cuando están activos indican que aún no han sido servidas las interrupciones solicitadas.

En el 8085 tenemos otro tipo de interrupción:
La TRAP, es la interrupción de mayor prioridad. Tiene una entrada específica y es activa a nivel alto.

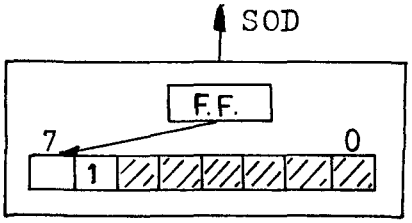
.-Entrada y salida de datos en serie. Por medio de los terminales SID y SOD del 8085 se pueden transmitir o recibir datos en serie.

Cada vez que ejecutamos la instrucción RIM se lee el estado lógico del terminal SID (Serial Input Data), el bit 7 del acumulador. De la misma forma con la ejecución de SIM guardamos en un biestable el bit 7 del acumulador. La salida del biestable coincide con el terminal SOD (Serial Output Data) siempre que el bit 6 del acumulador esté a 1 (si el bit 6 está a 0, el biestable no es afectado con la ejecución de SIM).

EFECTO DE RIM.



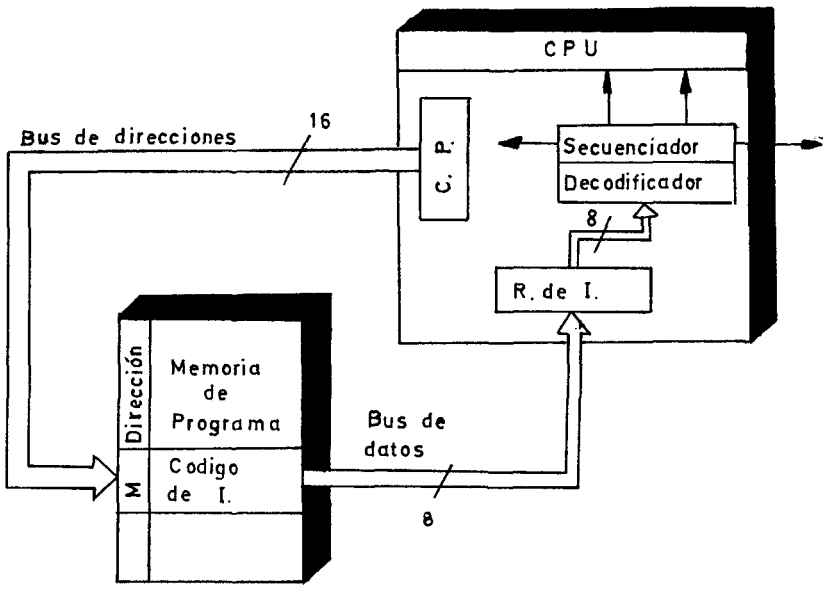
EFECTO DE SIM.



2.12.1. Tratamiento de una instrucción en el 8085.

Las instrucciones circulan desde la memoria hasta la CPU por el bus bidireccional de datos. Si éste consta de 8 líneas las combinaciones binarias que se pueden transmitir son $2^8=256$, es decir, este valor nos indica el número máximo de códigos de operación (C.O.) que puede interpretar el Decodificador de instrucciones.

Las 8 líneas del bus de datos depositan un byte, correspondiente al código de la instrucción en curso, en el Registro de datos e instrucciones, para poder decodificarla y ejecutarla. El proceso lo podemos ver en la figura siguiente:



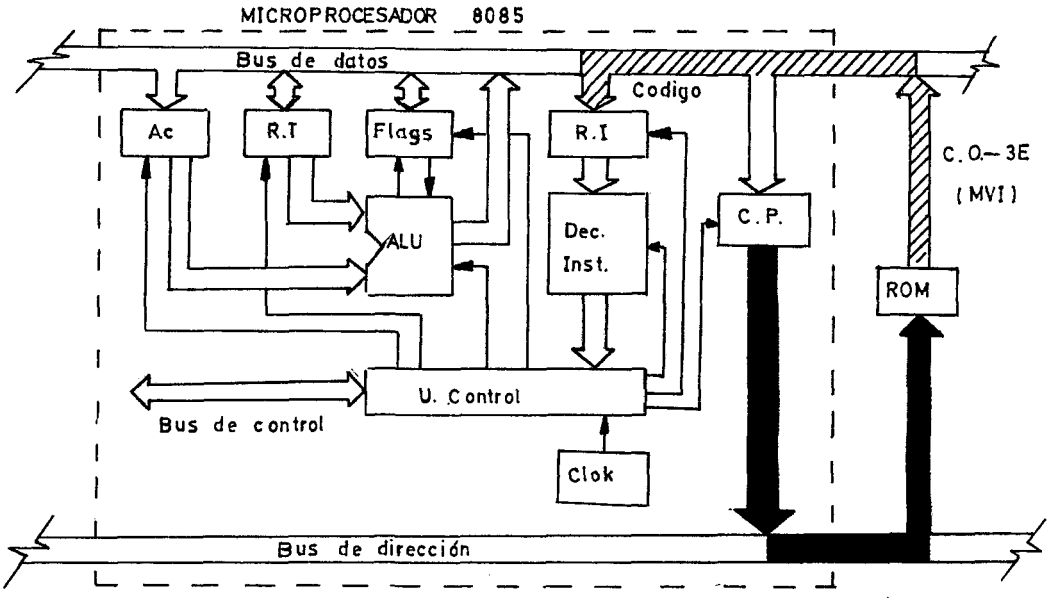
El código de una instrucción se interpreta en el decodificador de instrucciones de la unidad de control de la CPU. Este dispone de una memoria ROM interna, en donde residen los códigos capaces de ejecutar. Cada código gobierna una serie de microinstrucciones que se llevan a cabo a través del secuenciador.

Cada instrucción consta de una fase de búsqueda y otra de ejecución.

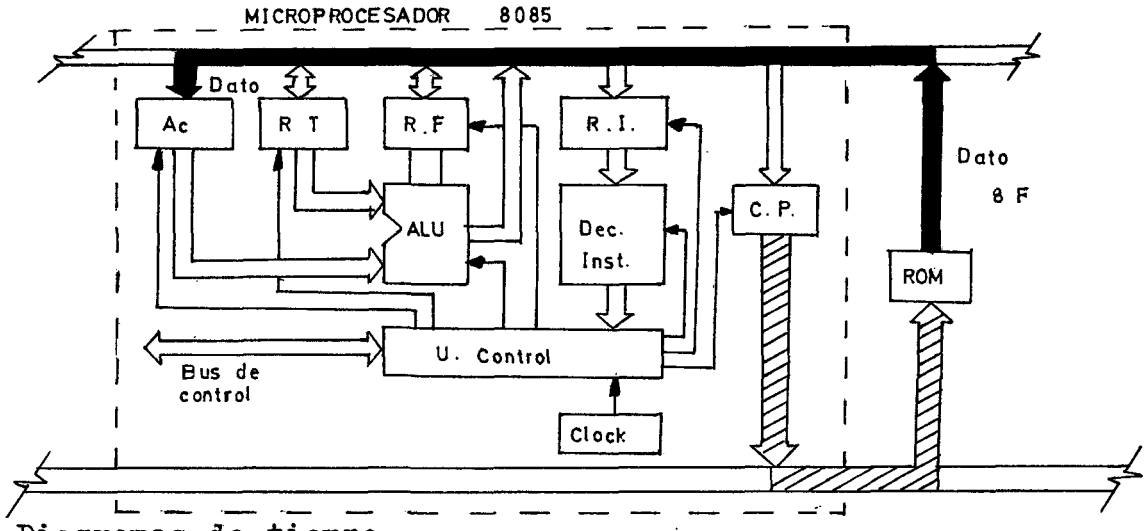
Si cogemos, por ejemplo, la instrucción MVI 8F (queremos cargar el dato 8F en el acumulador). Vamos a suponer que en la dirección 0100 de la memoria tenemos el código de ésta instrucción (3F). El dato (8F) lo tenemos en la dirección 0101. La dirección 0100 parte del CP y se dirige a la memoria por el bus de direcciones. El contenido de la posición 0100 (el código de la operación MVI) se traslada por el bus de datos hasta el registro de instrucciones, donde se deposita, con esto se termina la fase de búsqueda.

Al iniciarse la fase de ejecución, el C.O. depositado en el registro de instrucciones se transfiere al decidificador de instrucciones y después de interpretarse, se producen en el circuito de control las señales necesarias (microinstrucciones) para depositar en el acumulador el contenido de la posición 0101 (el dato). En las figuras siguientes se representan éstas dos fases de una instrucción:

+Flujo de información en la fase de búsqueda:



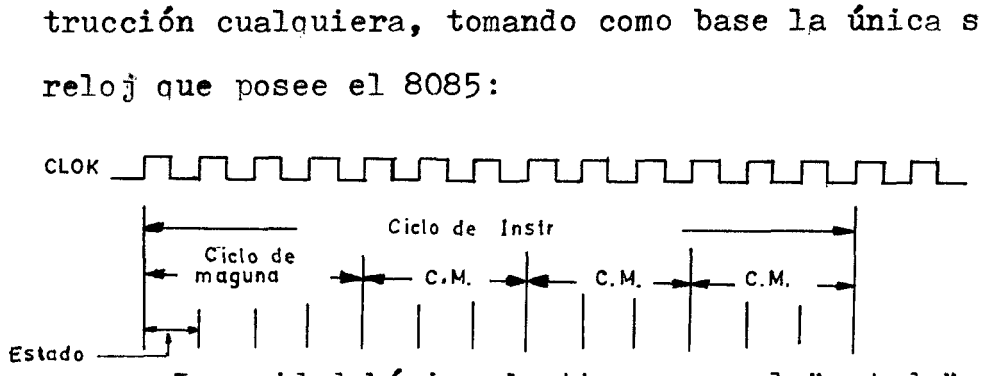
+Fase de ejecución de la instrucción MVI:



Diagramas de tiempo.

En este apartado se exponen algunos conceptos sobre la transmición de estados y transferencia de la información a lo largo del tiempo.

En la siguiente se muestran los tiempos de una instrucción cualquiera, tomando como base la única señal de reloj que posee el 8085:



La unidad básica de tiempo es el "estado" (un ciclo de reloj). Un ciclo de máquina consta de 3 a 6 estados. Las operaciones más sencillas requieren sólo un ciclo de máquina. El "cilo de instrucción" es el tiempo requerido para ejecutar una instrucción completa y puede constar de 1 a 5 ciclos de máquina.

Si los 6 estados que admite el 8085 en un ciclo de máquina, se marcan de T_1 a T_6 . Durante T_1 la CPU envía desde el C.P. una dirección por el bus correspondiente (aquí participa la señal ALE, informando sobre el demultiplexado

del bus de datos). Durante éste ciclo la CPU informa sobre el ciclo que se va a realizar, esto lo hace mediante las líneas IO/M, SO y S1. En la tabla siguiente tenemos los distintos ciclos que se pueden realizar:

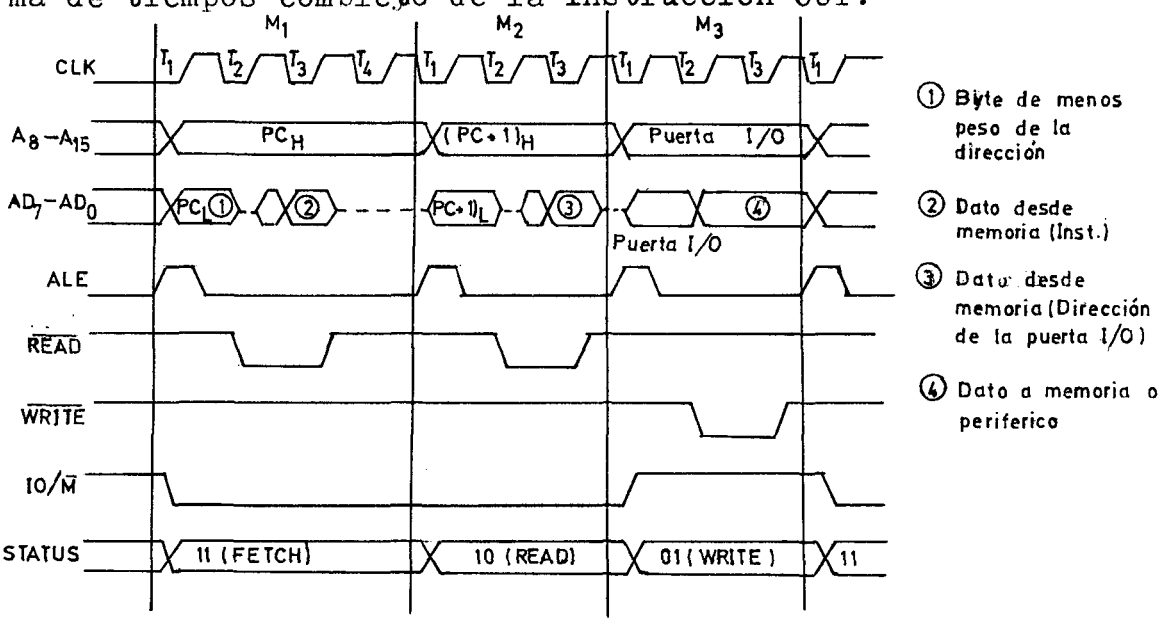
Tipo de ciclo	IO/M	SO	S1
Búsqueda	0	1	1
Lectura memoria	0	1	0
Escritura memoria	0	0	1
Interrupción	0	1	1
Entrada	1	1	0
Salida	1	0	1
Bus vacio	x	x	x

En T_2 el bus de direcciones pasa a contener el dato de la información (lectura ó de escritura). En éste estado se comprueba el nivel de la línea READY, según la cual la CPU entra o no en un estado de espera (WAIT).

En T_3 la CPU comprueba el estado de la línea HOLD.

Durante T_4, T_5, T_6, la CPU se dedica a desarrollar las operaciones internas, según sea la instrucción a ejecutar.

En las gráficas siguientes se representa el diagrama de tiempos completo de la instrucción OUT:



En M_1 se busca el C.O. en la memoria.

En M_2 el segundo byte de la instrucción (la dirección del puerto de salida) es leído desde la memoria.

En M_3 se ejecuta la instrucción: el dato se escribe en el puerto de salida.

La fase de búsqueda se realiza durante M_1 y es común a todas las instrucciones del 8085. Consta de los siguientes estados internos:

T_1: el contenido del C.P. pasa al bus de direcciones.

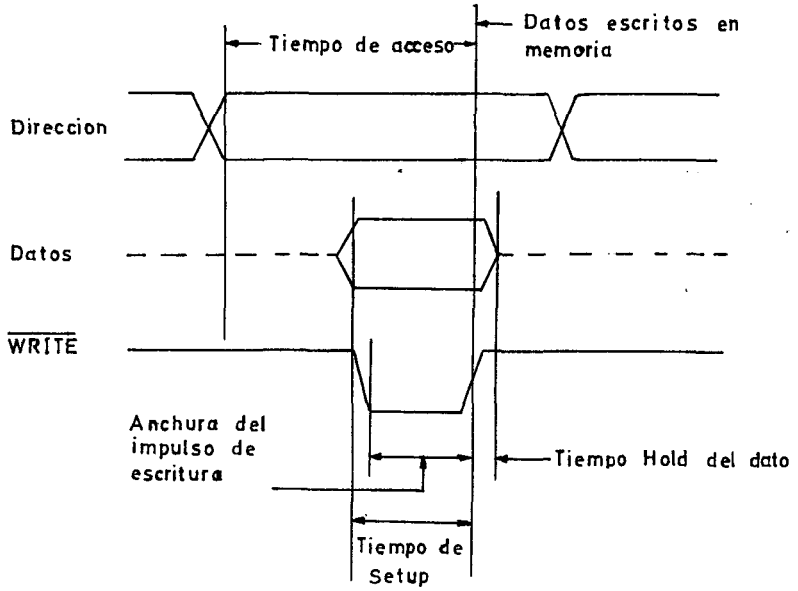
T_2: el C.P. se incrementa, mientras en la memoria se realiza la decodificación de la dirección suministrada en T_1.

T_3: el contenido de la posición de memoria direccionada se coloca en el bus de datos y se dirige al registro de instrucciones.

T_4(y a veces T_5): se decodifica la instrucción para poder ejecutarla.

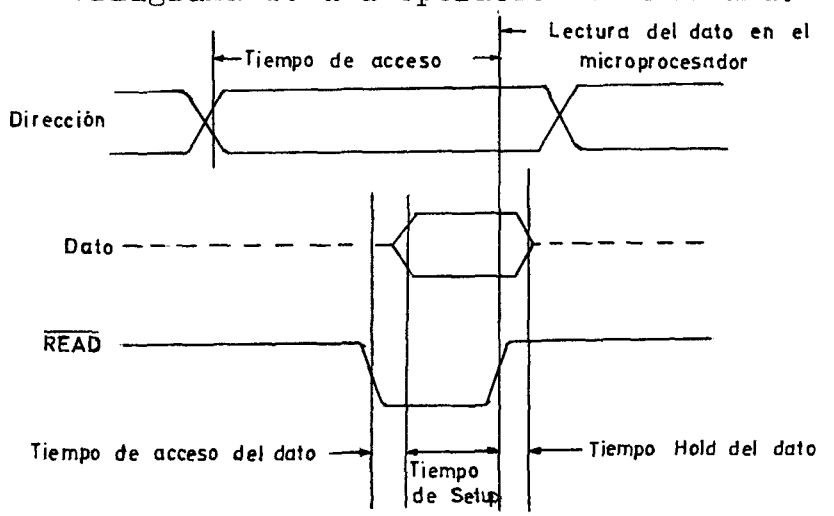
En las figuras siguientes se representan los diagramas de tiempos de una operación de lectura y de escritura en memoria.

+Diagrama de una operación de Escritura.



La dirección debe estar presente en el bus correspondiente, un tiempo previo (tiempo de acceso) que permite al decodificador interno de la memoria seleccionar la posición direccionada. El dato debe ser estable durante el periódo de tiempo adicional (tiempo set-up) antes de efectuar la escritura. El dato debe permanecer estable cierto tiempo después de producir el impulso de escritura (tiempo Hold).

+Diagrama de una operación de Lectura.



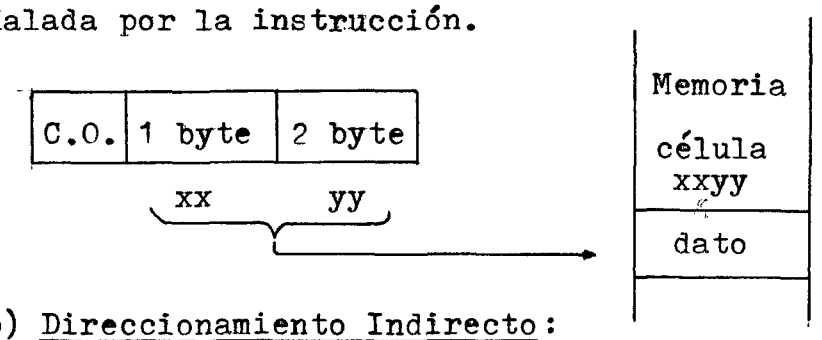
Al igual que antes, la dirección debe ser estable un tiempo previo para permitir la decodificación. Entonces se genera el impulso de lectura $\overline{\text{READ}}$ y trás un tiempo de acceso, la memoria coloca el dato seleccionado en el bus de datos. Este dato ha de ser estable un tiempo de set-up (preparación) antes de producirse el flanco positivo de $\overline{\text{READ}}$ que es cuando el dato es leído por la CPU. Después el dato ha de permanecer un tiempo Hold antes de que termine la operación de lectura.

2.12.2 MODOS DE DIRECCIONAMIENTO.

A continuación se exponen las distintas formas de acceder a la memoria que posee el 8085:

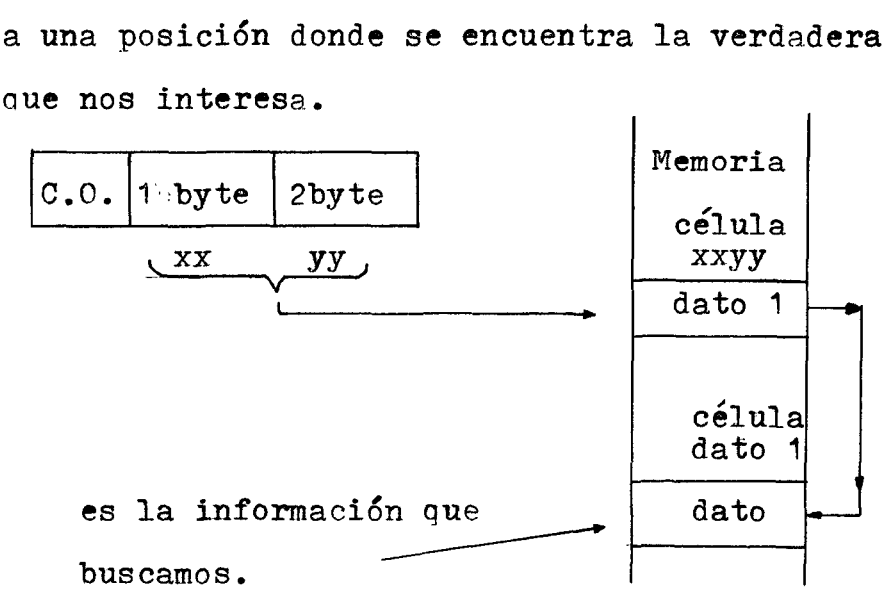
a) Direccionamiento Directo y Absoluto:

Las dos palabras que siguen al código de operación contienen la dirección de memoria donde está la información señalada por la instrucción.



b) Direccionamiento Indirecto:

Las dos palabras que siguen al código de operación apuntan a una posición donde se encuentra la verdadera dirección que nos interesa.

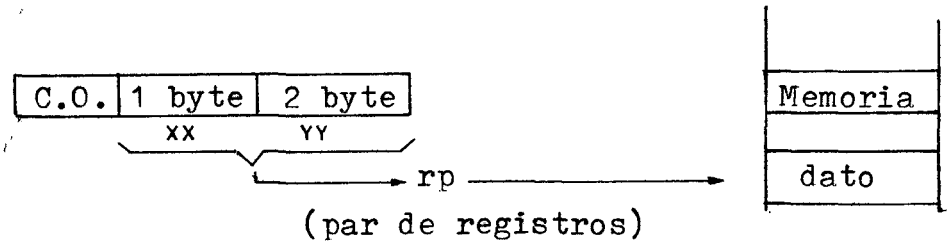


c) Direccionamiento Inmediato:

En este caso la instrucción no contiene ninguna dirección de memoria, sino el operando propiamente dicho.

d) Direccionamiento Indirecto por registro:

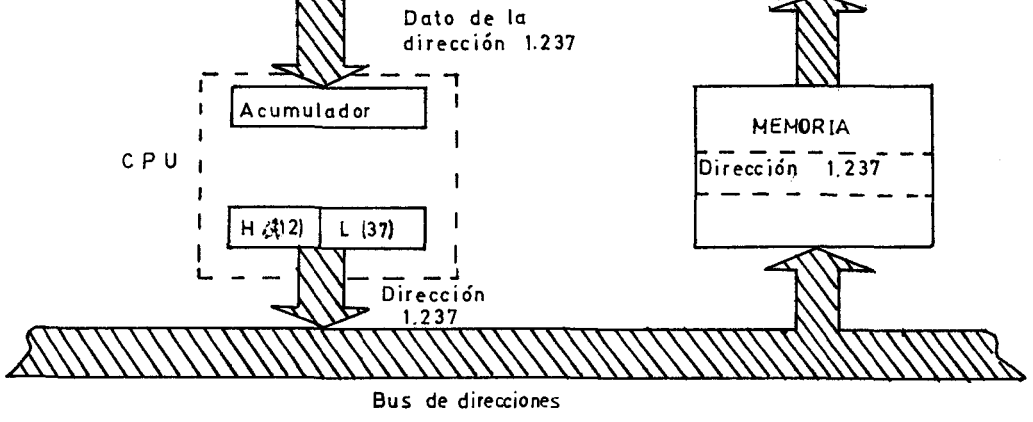
Las palabras que siguen al código de operación especifican el registro ó par de registros donde se encuentra la información a tratar.



e) Direccionamiento por Registro.

Las palabras que siguen al C.O. apuntan al par de registros donde se encuentra la información a tratar.

En la figura siguiente se indica el movimiento de la información en un direccionamiento indirecto por registro:



2123 CONJUNTO DE INSTRUCCIONES DEL 8085.

Vamos a dividirlas en diferentes grupos según su funcionalidad:

a) Que afectan en un solo registro:

.- INR r: (r)←(r) + 1

El contenido del registro r se incrementa en una unidad.

.- INR M: ((HL))←((HL)) + 1

El contenido de la posición de memoria, que está en HL es incrementado en una unidad.

.- DCR r: (r)←(r) - 1

El contenido del registro r es decrementado en uno.

.- DCR M ((HL))←((HL)) - 1

El contenido de la posición de memoria que está en HL es decrementado en uno.

.- CMA: $(A) \leftarrow (\overline{A})$

El contenido del acumulador es complementado (es decir se cambian los unos por ceros y viceversa.

.- DAA:

Convierte el contenido del acumulador, en su correspondiente código en la forma decimal (BCD). Lo hace de la siguiente forma:

+ Si el valor de los 4 bits menos significativos del Ac es mayor que 9 (o el carry está puesto), se le suma 6 al Ac.

+ Si el valor de los 4 bits más significativos del Ac es mayor que 9, (o el carry está puesto) le suma 6 a los 4 bits más significativos del Ac.

b) De transferencia de datos:

.- MOV r_1, r_2: $(r_1) \leftarrow (r_2)$

El contenido del registro r_2 es movido al r_1.

.- MOV r, M: $(r) \leftarrow ((HL))$

El contenido de la dirección de memoria que está en ML es movido al registro r.

.- MOV M, r: $((HL)) \leftarrow (r)$

El contenido del registro r es puesto en la dirección de memoria contenida en HL.

.- LDAX rp: $(A) \leftarrow ((rp))$

El contenido de la dirección de memoria que está en rp es puesto en el Ac.

NOTA: Sólo puede usarse el par de registro (rp) B,C a el D,E.

.- STAX rp: $((rp)) \leftarrow (A)$

El contenido del registro A es movido a la dirección contenida en el par de registros rp (sólo pueden usarse BC o DE).

c) De relación entre memoria o registros y Ac:

.- ADD r: $(A) \leftarrow (A) + (r)$

El contenido del registro r es sumado al acumulador. El resultado lo ponemos en el Ac.

.- ADD M: $(A) \leftarrow (A) + ((HL))$

El contenido de la dirección de memoria que está en HL, se le suma al acumulador. El resultado lo ponemos en el acumulador.

.- ADC r: $(A) \leftarrow (A) + (r) + (CY)$

.- ADC M: $(A) \leftarrow (A) + ((HL)) + (CY)$

Estas dos (ADC r y ADC M), son iguales que las dos

anteriores pero se tiene en cuenta el valor del carry.

.- SUB r: $(A) \leftarrow (A) - (r)$

Al contenido del acumulador le restamos el contenido de r. El resultado lo colocamos en el acumulador.

.- SUB M: $(A) \leftarrow (A) - ((HL))$

Al contenido del acumulador le restamos el contenido de la dirección que está en HL.

.- SBB r: $(A) \leftarrow (A) - (r) - (CY)$

.- SBB M: $(A) \leftarrow (A) - ((HL)) - (CY)$

Estas dos (SBB r y SBB M), son iguales que las dos anteriores pero se tiene en cuenta el valor del carry.

.- ANA r: $(A) \leftarrow (A) \wedge (r)$ $\wedge$ = AND

Se realiza una operación AND entre el contenido del acumulador y el contenido del registro (r). El resultado lo ponemos en el acumulador. La tabla de verdad es:

AB	S
00	0
01	0
10	0
11	1

.- ANA M: $(A) \leftarrow (A) \wedge ((HL))$

Se realiza una operación AND entre el contenido de la dirección que está en HL y el del acumulador.

.- XRA r: $(A) \leftarrow (A) \veebar (r)$

.- XRA M: $(A) \leftarrow (A) \veebar ((HL))$

Estas dos (XRA r y XRA M), son iguales que las dos anteriores pero en vez de realizarse una operación AND se realiza una OR exclusiva ($\veebar$)

La tabla de verdad es:

AB	S
00	0
01	1
10	1
11	0

.- ORA r: (A)←(A) V (r)

.- ORA M: (A)←(A) V ((HL))

En estos dos casos (ORA r y ORA M), se realiza una operación OR inclusiva (V). La tabla de verdad es:

AB	S
00	1
01	0
10	0
11	1

.- CMP r: (A) - (r)

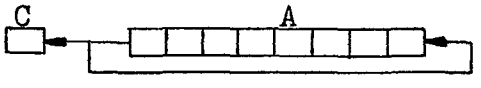
El contenido del registro r. es restado al contenido del acumulador. El contenido del acumulador no varía. Si (A) es igual a (r) el flag Z se pone a cero. Si (A) es menor que (r) el flag CY se pone a 1.

.- CMP M: (A) - ((HL))

El contenido de la dirección que está en HL es restado al contenido del acumulador. El contenido del acumulador no varía. Si (A) es igual que ((HL)) el flag Z se pone a cero. Si (A) es menor que ((HL)) el flag CY se pone a uno.

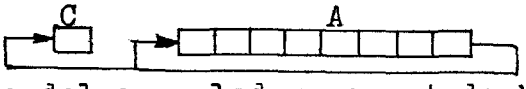
d) De rotación del acumulador:

.- RLC:



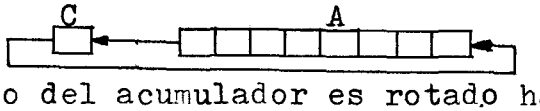
El contenido del acumulador es rotado hacia la izquierda una posición.

.- RRC:



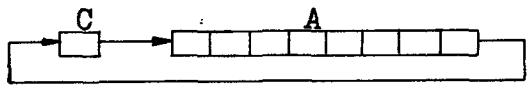
El contenido del acumulador es rotado hacia la derecha una posición.

.- RAL:



El contenido del acumulador es rotado hacia la

izquierda una posición a través del carry.

.- RAR: 

El contenido del acumulador es rotado hacia la derecha una posición a través del carry.

e) De pares de registros:

.- PUSH rp: ((SP) - 1)←(rh)
((SP) - 2)←(rl)
(SP)←(SP) - 2

El contenido del registro más significativo del par lo ponemos en una dirección de memoria menos que la indicada en el SP. El contenido del registro menos significativo del par es colocado en dos direcciones menos que la indicada en el SP. El contenido del SP es decrementado en dos unidades.

.- PUSH PSW: ((SP) - 1)←(A)
((SP) - 2)←PSW
(SP)←(SP) - 2

El contenido del (A) lo salvamos en una dirección menos que la indicada en el SP. y el contenido de la palabra de estados (PSW) en dos direcciones menos que la del SP. El contenido del SP es decrementado en dos unidades.

.- POP rp: (rl)←((SP))
(rh)←((SP) + 1)
(SP)←(SP) + 2

El contenido de la dirección del SP se coloca en el registro menos significativo del par. El contenido de la dirección siguiente a la indicada en el SP se coloca en el registro más significativo del par. El contenido de SP es

incrementado en dos.

.- POP PSW: PSW←((SP))
A←((SP) + 1)
(SP)←(SP) + 2

Hacemos lo mismo que antes pero con el contenido del acumulador y de la palabra de estados en lugar del rp.

.- DAD rp: (HL)←(HL) + (rp)

El contenido del par de registros rp se le suma al contenido de HL. El resultado es colocado en HL.

.- INX rp: (rp)←(rp) + 1

El contenido del par de registros rp es incrementado en uno.

.- DCX rp: (rp)←(rp) - 1

El contenido del par de registros rp es decrementado en uno.

.- XCHG: (H) ←→ D
(L) ←→ E

El contenido de los registros H y L es cambiado con el contenido de los registros D y E.

.- XTHL: (L) ←→ ((SP))
(H) ←→ ((SP) + 1)

El contenido de HL se intecambia con el contenido de las primeras posiciones del SP.

.- SPHL: (SP)←HL

El contenido de los registros HL (16 bits) es movido al stack.

f) Inmediatas:

.- LXI rp, dato (16 bits): (rh)←(byte 3)
(rl)←(byte 2)

El byte 3 de la instrucción es movido al registro más significativo del par. El byte 2 al menos significativo.

.-MVI r, dato: (r)←(byte 2)

El contenido del 2 byte de la instrucción es movido al registro r.

.- MVI M, dato: ((HL))←(byte 2)

El contenido del 2 byte de la instrucción es movido a la dirección que está en HL.

.- ADI dato: (A)←(A) + (byte 2)

Al contenido del acumulador le sumamos el contenido del 2byte de la instrucción. El resultado lo colocamos en el acumulador.

.- ACI dato: (A)←(A) + (byte 2) + (CY)

Igual que antes pero teniendo en cuenta el carry.

.- SUI dato: (A)←(A) - (byte 2)

Al contenido del acumulador le restamos el contenido del 2 byte de la instrucción. El resultado lo colocamos en el acumulador.

.- SBI dato: (A)←(A) - (byte 2) - (CY)

Igual que antes pero teniendo en cuenta el carry.

.- ANI dato: (A)← (A) $\wedge$ (byte 2)

Se realiza una operación AND entre el contenido del acumulador y el contenido del 2 byte de la instrucción. El resultado lo colocamos en el acumulador.

.- XRI dato: (A)← (A) $\oplus$ (byte 2)

Igual que la anterior pero en este caso realizamos una OR-exclusiva.

.- ORI dato: (A) ← (A) $\oplus$ (byte 2)

Realizamos una OR-inclusiva, entre el contenido del acumulador y el segundo byte de la instrucción.

.-CPI dato : (A) - (byte 2)

Al contenido del acumulador le restamos el contenido del 2 byte de la instrucción. Elcontenido del acumulador no varía. Si (A)=(byte 2) se pone a 1 el flag Z. Si (A)<(byte 2) se pone a 1 el flag CY.

g) De direccionamiento directo:

.- STA dirección : ((byte 3)(byte 2))←(A)

El contenido del acumulador se coloca en la dirección indicada por los byte 2 y 3 de la instrucción.

.- LDA dirección : (A)←((byte 3)(byte 2))

El contenido de la dirección indicada por los bytes 3 y 2 de la instrucción se coloca en el acumulador.

.- SHLD dirección : ((byte 3)(byte 2))←(L)

((byte 3)(byte 2)+1)←(H)

El contenido del registro L se coloca en la dirección de memoria indicada por los bytes 2 y 3 de la instrucción. El contenido del registro H se coloca en la dirección siguiente.

.- LHLD dirección : (L)←((byte 3)(byte 2)

(H)←((byte 3)(byte 2)+1)

El contenido de la dirección indicada por los bytes 2 y 3 de la instrucción se coloca en el registro L. El contenido de la dirección siguiente se coloca en H.

h) Que afectan al carry:

.- CMC : (CY)←($\overline{CY}$)

Complementamos el flag carry.

.- STC : (CY)←1

Ponemos el carry a 1.

i) De salto:

.-PCHL : (PCH)←(H)
(PCL)←(L)

El contenido del registro H se coloca en los 8 bits más significativos del CP. El contenido del registro L se coloca en los 8 bits menos significativos.

.- JMP dirección : (CP)←(byte 3)(byte 2)

El contenido del 2 y 3 byte de la instrucción se coloca en el registro CP.

.- J condición dirección : (CP)←(byte 3)(byte 2)

Si la condición es cierta, el contenido del 2 y 3 byte de la instrucción se coloca en el CP. Si la condición no es cierta se realiza la instrucción siguiente.

Las condiciones que se pueden poner son:

.- JC dirección : si el carry=1 se efectúa el salto a la dirección indicada.

.- JNC dirección : se efectúa el salto cuando C=0.

.- JZ dirección : cuando Z=1.

.- JNZ dirección : cuando Z=0.

.- JM dirección : cuando el flag de signo (S)=1.

.- JP dirección : cuando S=0.

.- JPE dirección : cuando el flag de paridad(P)=1.

.- JPO dirección : cuando P=0.

j) De llamada a subrutina:

.- CALL dirección : ((SP)-1)←(PCH)
((SP)-2)←(PCL)
(SP)←(SP)-2
(CP)←(byte 3)(byte 2)

Los 8 bits más significativos de la instrucción son

movidos a una dirección menos que la indicada en el SP. Los 8 bits menos significativos los ponemos en dos direcciónes menos que la indicada en el SP (Se guarda la instrucción siguente para saber el punto de retorno). El contenido del SP se decrementa en dos unidades. En el CP ponemos la dirección indicada en los bytes 2 y 3 de la instrucción.

.- C condición dirección :

Si la condición es cierta, se salta a la dirección indicada, si no se ,continua en la instrucción siguiente.

Las condiciones que se pueden poner son:

.- CC dirección: se salta si el carry=1.

.- CNC dirección: si el C=0.

.- CZ dirección: si Z=1.

.- CNZ dirección: si Z=0.

.- CM dirección: si S=1.

.- CP dirección: si S=0.

.- CPE dirección: si P=1.

.- CPO dirección: si P=0.

k) De retorno de subrutinas:

.- RET : (PCL)←((SP))

(PCH)←((SP)+1)

(SP)←(SP)+2

El contenido de la dirección de memoria indicada en el SP lo ponemos en los 8 bits menos significativos del CP. El contenido de la dirección siguiente en los 8 bits más significativos del CP. El contenido del SP se incrementa en dos.

.- RET condición :

Si la condición es cierta se retorna a la dirección indicada en los contenidos de las dos primeras posiciones del SP.

Las condiciones pueden ser:

.- RC : se retorna si C=1.

.- RNC : si C=0.

.- RZ : si Z=1.

.- RNZ : si Z=0.

.- RM : si S=1.

.- RP : si S=0.

.- RPE : si P=1.

.- RPO : si P=0.

1) De restart:

.- RST n : ((SP)- 1)←(PCH)

((SP) - 2)←(PCL)

(SP)←(SP) - 2

(CP)←Vector.

Desde el programa pueden simularse hasta 8 interrupciones, es decir, con la ejecución de RST n se produce un salto incondicional a una subrutina de servicio. Si n=NNN la dirección que ponemos en el CP es:

Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	N	N	N	Ø	Ø	Ø

Por ejemplo si n=5 NNN=101. El vector (la dirección) que ponemos en el CP es:

Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø	1	Ø	Ø	Ø

0 0 2 8

A partir de ésta dirección de memoria tendremos la dirección de la subrutina de servicio.

Las interrupciones de éste tipo que posee el SDK85 (con sus rutinas de servicio) son:

+ Timer Interrupts (TRAP) = RST 4.5

-Dirección de origen: 24H.

-JMP STP 25 : Salta al comando S.STEP.

Esta interrupción la usa el comando S.STEP parair ejecutando el programa del usuario paso a paso.

+ RST 5.

-Dirección de origen: 28H.

- JMP RSET 5 : Salta a la dirección 20C8 de la RAM.

En las posiciones 20C8, 20C9, 20CA de la RAM, el usuario puede colocar instrucciones de salto a subrutinas.

+ Input Interrupt = RST 5.5

-Dirección de origen: 2CH.

-JMP ININT : Salta a la función ININT cuando se produce una entrada por el teclado.

Esta interrupción la usa el 8279 para indicarle a la CPU cuando se produce alguna entrada desde el teclado.

+ RST 6.

-Dirección de origen: 30H.

-JMP RSET 6 : Salta a la dirección 20CB de la RAM.

El usuario puede cargar en la posiciones 20CB, 20CC, 20CD, las instrucciones que desee.

+Hard wired user interrupts = RST 6.5

-Dirección de origen: 34H.

-JMP RST 6.5 : Salta a 20CE de la RAM.

Esta interrupción es destinada para que el usuario la emplee en el diseño de algún montaje.Al producirse se salta a 20CE.El usuario puede cargar en 20CE, 20CF, 20DO, las instrucciones que desee.

+ RST 7.

-Dirección de origen: 38H.

-JMP RSET 7 : Salta a dirección 20D1 de la RAM.

El usuario puede cargar las posiciones 20D1, 20D2, 20D3, con las instrucciones que desee.

+ Vectored Interrupt = RST 7.5

-Dirección de origen: 3CH.

-JMP USINT : Salta a la dirección 20D4 de la RAM.

Esta interrupción se produce cada vez que el usuario pulse la tecla "Vector Interrupt" desde el teclado. En las posiciones 20D4, 20D5, y 20D6 el usuario puede cargar las instrucciones que desee.

m) De interrupción:

.- EI (Enable Interrupts):

Después de ejecutarse quedará habilitado el sistema de interrupciones. Permite a la CPU reconocer y responder a una interrupción.

.- DI (Disable Interrupts):

Deshabilita la atención de interrupciones. La CPU ignora las interrupciones que se produzcan a partir de ésta instrucción.

n) De entrada/salida:

.- IN puerto : (A)←bus

El dato, colocado en los 8 bits del bus de datos bi-direccional por el puerto indicado, se pone en el acumulador.

.- OUT puerto : bus←(A)

El contenido del registro A es colocado en los 8 bits del bus de datos para transmitirlo por el puerto indicado.

o) De alto y de no operación:

.- HLT :

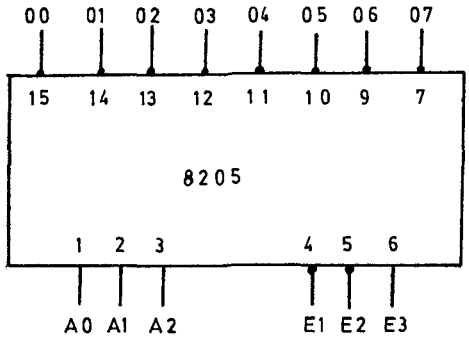
La CPU entra en estado de reposo del que sólo puede salir a través de una interrupción. Los registros no son afectados.

.- NOP :

La instrucción no modifica nada en absoluto y se pasa a la siguente. (Se puede usar para producir tiempos de retardo.)

2.1.3.-CARACTERISTICAS DEL 8205. DECODIFICADOR DE DIRECCIONES.

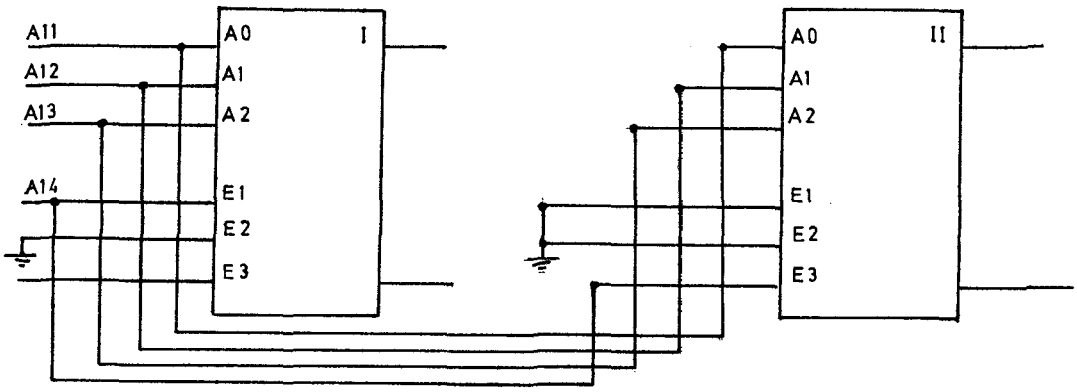
La configuración del chip es la siguiente:



La tabla de verdad que posee éste decodificador es:

A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

Dependiendo del número de bloques que se vayan a direccionar, se pueden usar más de un decodificador, uniéndolos de la forma siguiente:



Para efectuar esta conección emplearemos 4 bits del bus de direcciones, en este caso serán A_{11}, A_{12}, A_{13}, A_{14}, (según vemos ,en la figura anterior). El funcionamiento es el siguiente: Mientras A_{14}, procedente del bus de direcciones se encuentra a nivel bajo, la entrada del primer decodificador estará pues a nivel bajo, al igual que E_2, y como E_3 está a nivel alto se seleccionarán las salidas CS_1 a CS_8 (ver la tabla de verdad). Las salidas de CS_9 a CS_{16} no se pueden seleccionar debido a que E_3 del segundo decodificador está a nivel bajo (por estar A_{14} a nivel bajo).

Cuando A_{14} pasa a nivel alto, se anula la decodificación del primer decodificador y se hace posible la decodificación de las líneas de CS_9 a CS_{16}.

En el montaje del SDK'85 sólo se usa un decodificador, debido al número de bloques que hay que direccionar. (Cuanto mayor sea éste número, más decodificadores se usarán en el montaje).

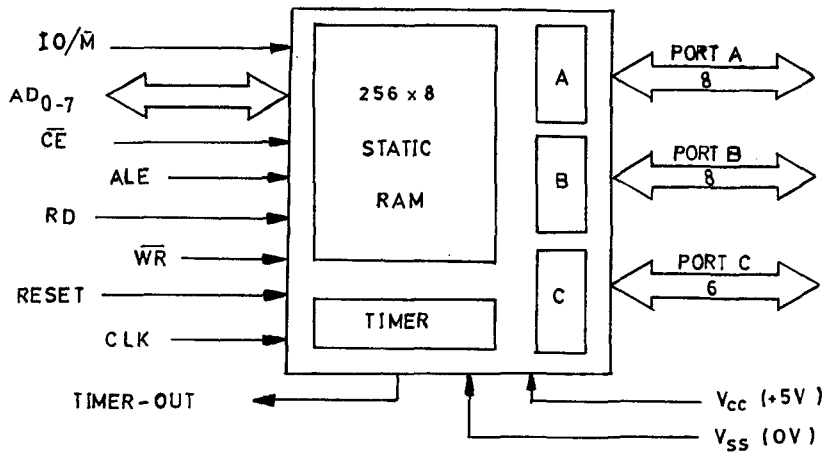
2.1.4 CARACTERISTICAS DEL 8155.

El 8155 es un chip con memoria RAM y puertos de entrada/salida, para ser usados en sistemas con microprocesadores 8080 y 8085. La parte que constituye la RAM está diseñada con 2048 celdas estáticas, organizadas como 256x8. El tiempo máximo deacceso es de 400 nanosegundos, lo que permite usarlo en la CPU 8085 sin tiempo de espera.

Los puertos de entrada/salida consisten en tres puertos de propósito general de I/O. Uno de los tres puede ser programados.

El chip contiene además un contador de tiempo programable, el cual genera una onda cuadrada o un punto dependiendo de la forma en que se ha programado.

Descripción de los terminales:



RESET. Es la entrada del pulso provocado por el 8085 para inicializar el sistema. La entrada a nivel alto en esta línea, recetea el chip e inicializa los tres puertos I/O. (va conectada a la patilla RESET OUT del 8085)

AD_{0-7}. Líneas de dirección/datos (3 estados). Se conectan con los 8 bits de orden bajo del bus de direcciones/datos de la CPU.

$\overline{CE}$. Entrada activa a nivel bajo, se usa para habilitar el chip.

$\overline{RD}$. Entrada usada para controlar la lectura. Una entrada anivel bajo en esta línea (con $\overline{CE}$ activa) se llena las entradas AD_{0-7}. Si la entrada IO/$\overline{M}$ está a nivel bajo, el contenido de la RAM será leído hacia el bus de direcciones. Por otro lado el contenido del puerto entrada/salida seleccionado o el registro de comandos/estados pueden también ser leídos en el bus de direcciones.

$\overline{WR}$. Controla la escritura. Una señal de nivel bajo en esta línea (con $\overline{CE}$ activa) hace que el dato en el bus direcciones/datos sea escrito en la RAM o en el puerto IO/ seleccionado y en el registro comando/estados dependiendo de la señal IO/$\overline{M}$.

ALE. Habilita el latch de direcciones.

IO/$\overline{M}$. Si está a nivel bajo selecciona la memoria. Si está a nivel alto selecciona el puerto I/O y el registro comando/estados.

PA_{0-7}. Son usadas como patas de entrada/salida de propósito general. La dirección de I/O es seleccionada según se programa el registro de comandos.

PB_{0-7}. Son iguales que las anteriores.

PC_{0-5}. Estas 6 patas pueden funcionar como puerto de entrada, como puerto desalida o como señal de control para el puerto A y el puerto B, programándolo a través del registro de comandos. Cuando PC_{0-5} son usadas como señales de control, realizan las siguientes funciones:

PC_0 A INTR (interrumpe al puerto A)

PC_1 ABF (el contenido del puerto A se pone en el Buffer)

PC_2 $\overline{A\ STB}$ (puerto A inhabilitado)

PC_3 B INTR (interrumpe el puerto B)

PC_4 $\overline{BBF}$ (el contenido del puerto B se pone en el BUFFER)

PC_5 B STB (puerto B inhabilitado).

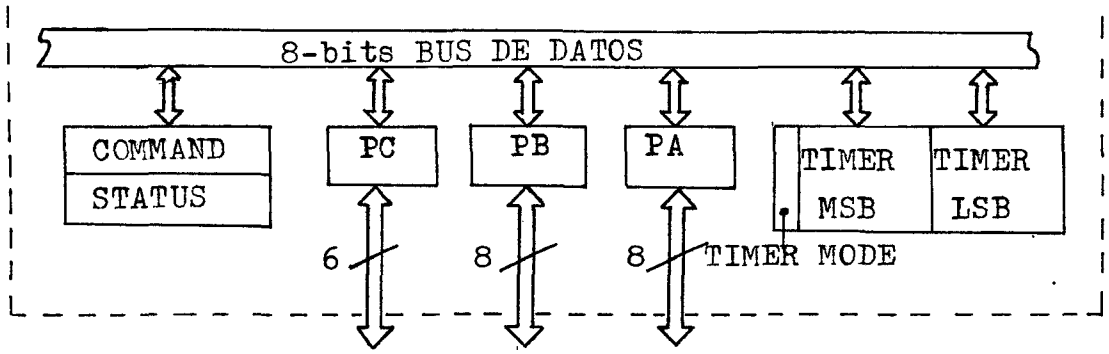
TIMER IN. Entrada para el contador.

$\overline{TIMER\ OUT}$. Salida de reloj. Esta señal desalida puede ser una onda cuadrada o un pulso dependiendo de cómo se programe el TIMER.

VCC. 5 voltios.

VSS. Tierra.

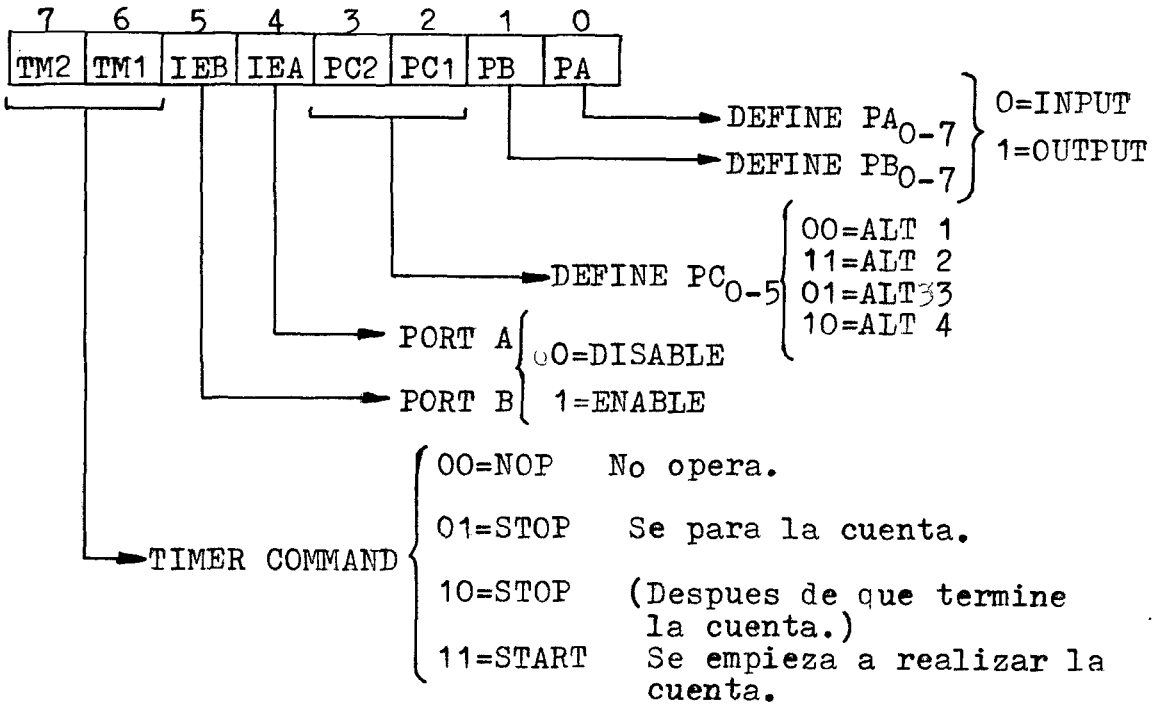
El chip está formado por los siguientes registros; según se ve en la figura:



Programación del registro de comandos.

Este registro consiste en 8 cerrojos. Los 4 bits, del 0 al 3 definen la forma de los puertos, los bits 4 y 5 posibilitan la interrupción desde el puerto C, cuando actúa como puerto de control, y los bits 6 y 7 son para el reloj.

El contenido de este registro puede ser cambiado usando la dirección I/O xxxxx000 durante una operación de escritura (con $\overline{CE}$ activa y IO/$\overline{M}$=1). El significado de cada bit del registro de comandos lo podemos ver en la figura siguiente: (el contenido deeste registro no puede ser leído)

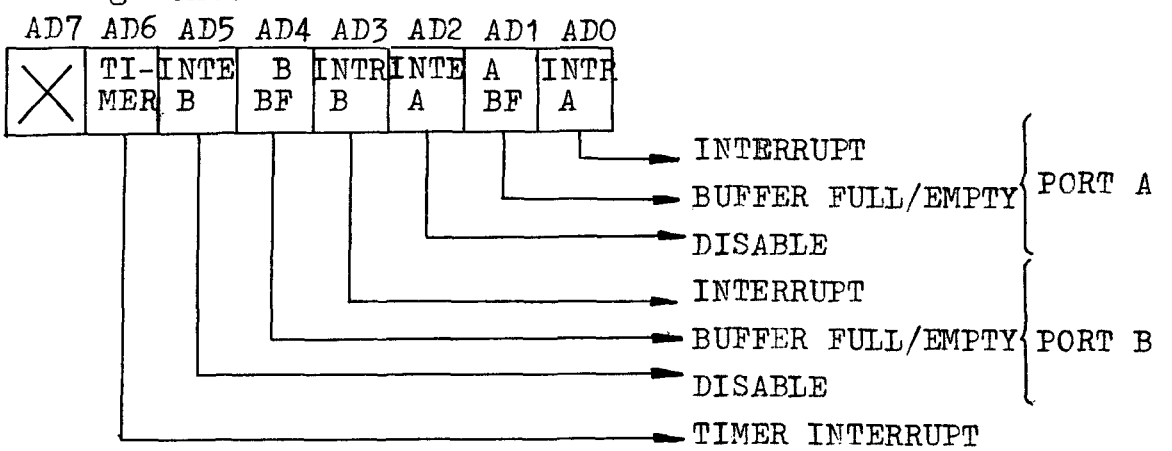


.- Lectura del registro de estados.

Este registro consiste en 7 bits que actuan como llaves, seis (del 0-5) para los estados de las puertas y el sexto para los estados del reloj.

. Los estados del reloj y la sección I/O se pueden obtener por lectura del registro de estados.

El formato del registro lo podemos ver en la figura siguiente:



.- Sección de entrada/salida.

+Registro de Comando/Estados (C/S): a ambos se les

asigna la dirección xxxxx000.Cuando estos registros son seleccionados durante una operación de escritura, un comando es escrito en el registro de comandos. (los contenidos de éste registro no son accesibles a traves de las clavijas).Si se selecciona en una operación de lectura, la informaciónm del estado de las puertas I/O y del reloj llega a ser asequible en las lineas AD_{0-7}.

+Registro PA: este registro puede ser programado para actuar como puerto de Entrada/Salida, dependiendo del estado del registro C/S. También puede actuar en cualquiera de las formas básicas ó en el modo elegido (ver tabla 1) . Las clavijas asignadas a éste registro son PA_{0-7}.

+Registro PB: posee las clavijas PB_{0-7}, funciona igual oue el registro PA.

+Registro PC: posee sólo seis bits, que pueden ser programados para actuar,como puerto de I/O, ó como senales de control para los PA y PB. Esto lo conseguimos programando los bits AD_2 y AD_3 del registro C/S. Cuando actua como puerto de control tres bits se le asignan al PA y otros tres al PB. Cada uno de los bits tiene una misión:

-el primero, es un interruptor que el 8155 envia fuera.

-el segundo, es una señal de salida que nos indica si el buffer está lleno ó vacio.

-el tercero, es una patilla de entrada para aceptar una interrupción (para inhabilitar el puerto que se esta usando).

En la tabla siguiente podemos ver los distintos modos en que se puede programar el registro PC:

Pin	ALT 1	ALT22	ALT 3	ALT44
PC0	Input Port	Output Port	A INTR	A INTR
PC1	Input Port	Output Port	A BF	A BF
PC2	Input Port	Output Port	A $\overline{STB}$	A $\overline{STB}$
PC3	Input Port	Output Port	O. P.	B INTR
PC4	Input Port	Output Port	O. P.	B BF
PC5	Input Port	Output Port	O. P.	B $\overline{STB}$

TABLA 1.

A INTR= Puerto A interrunpido.

A BF= El contenido del puerto A se pone en el buffer.

A $\overline{STB}$= Puerto A inhabilitado.

O. P.= Output Port.

Cada vez que un puerto es cambiado de entrada a salida, todas las patillas se ponen a nivel bajo. Cuando el 8155 se resetea, los latches de salida se limpian y los tres puertos entran en el modo de entrada.

.-Sección del reloj.

El reloj es un contador-decontador de 14 bits que cuenta los pulsos del reloj de entrada y da una salida en forma de onda cuadrada ó un pulso cuando la cuenta se ha realizado. Para programarlo, primero hemos de cargar el registro de longitud de la cuenta (RLC).Los bits de 0-13 especifican la longitud de la próxima cuenta y los bits 14-15 especifican la forma de salida del reloj:

7	6	5	4	3	2	1	0
M_2	M_1	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8

Timer Mode — Bits más significativos de la longitud de la cuenta.

7	6	5	4	3	2	1	0
T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0

Bits menos significativos de la longitud de la cuenta.

Existen cuatro formas de elegir la salida del reloj, segun sea el estado de los bits M_2 y M_1:

M_2	M_1	Forma de la onda de salida.
0	0	Onda cuadrada simple:
0	1	Onda cuadrada continua:
1	0	Pulso simple (al terminar la cuenta):
1	1	Serie continua de pulsos:

Los bits 6-7 (TM_2 y TM_1) del registro de comandos son usados para ,empezar y parar la cuenta:

TM_2	TM_1	
0	0	NOP.No afecta la operación del contador.
0	1	STOP.No opera si el reloj no ha empezado. Si esta corriendo, para la cuenta.
1	0	STOP (despues de que se realice la cuenta)
1	1	START. Carga el Modo y la longitud de la cuenta(si el reloj no estaba corriendo) .Si está corriendo empieza la nueva cuenta, inmediatamente despues de que se acabe la anterior.

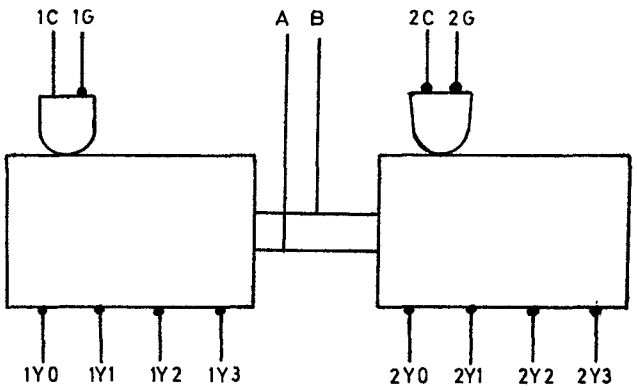
Mientras el reloj está contando se puede cargar una nueva longitud de cuenta en los registros. Antes de que la nueva cuenta sea usada por el reloj hay que mandarle un comando de comienzo (START) al contador. Esto se hace

aunque sólo se desee cambiar la longitud de la cuenta, manteniendo el mismo modo.

El contador del 8155 no se inicializa de ningín modo cuando ocurre un Reset del hardware, es decir la cuenta no puede empezar despues de un Reset hasta que no aparezca un comando START, mandado por el registro C/S.

2.1.5.-CARACTERISTICAS DEL DECODIFICADOR 74LS156.

Debido a que la salida del 8279 sólo sirve para excitar a un sólo display es necesario la utilización de un decodificador, el 74LS156, que posee el siguiente diagrama lógico:



La tabla de verdad es la siguiente:

C	B	A	G	$2Y_0$	$2Y_1$	$2Y_2$	$2Y_3$	$1Y_0$	$1Y_1$	$1Y_2$	$1Y_3$
X	X	X	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H
L	H	H	L	H	H	H	L	H	H	H	H
H	L	L	I	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H
H	H	L	L	H	H	H	H	H	H	L	H
H	H	H	L	H	H	H	H	H	H	H	I

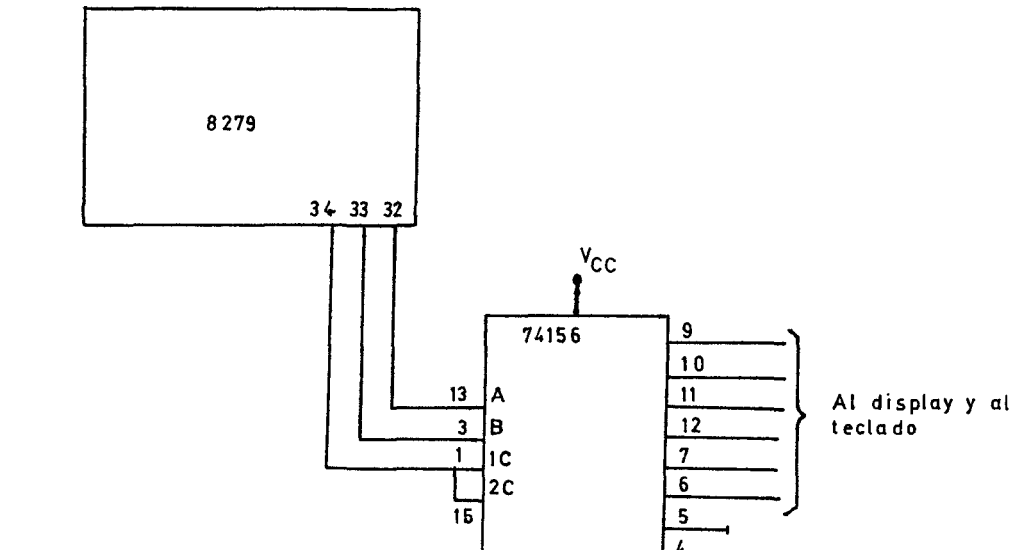
La conección al sistema la realizamos de la forma siguiente:

.- Las líneas de Selección (SL_0-SL_3) del 8279 van unidas a las entradas A, B, C.

.- Las entradas G, unidas, irán a la salida RESET del microprocesador.

.- Las salidas Y, atacarán por un lado a los display y por otro al teclado.

Esto lo podemos ver en la figura siguiente:



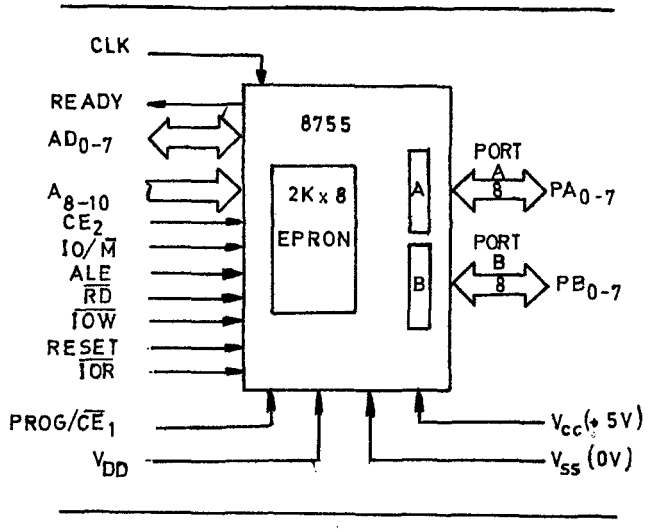
2.1.6.- CARACTERISTICAS DEL 8755.

Este chip es una memoria EPROM con puertos de entrada/salida, para ser usada en sistemas con el 8085.

La parte de la memoria está organizada en 2048 palabras de 8 bits. Tiene un tiempo máximo de acceso de 450 nanosegundos para que pueda ser usado con la CPU 8085 sin estados de espera.

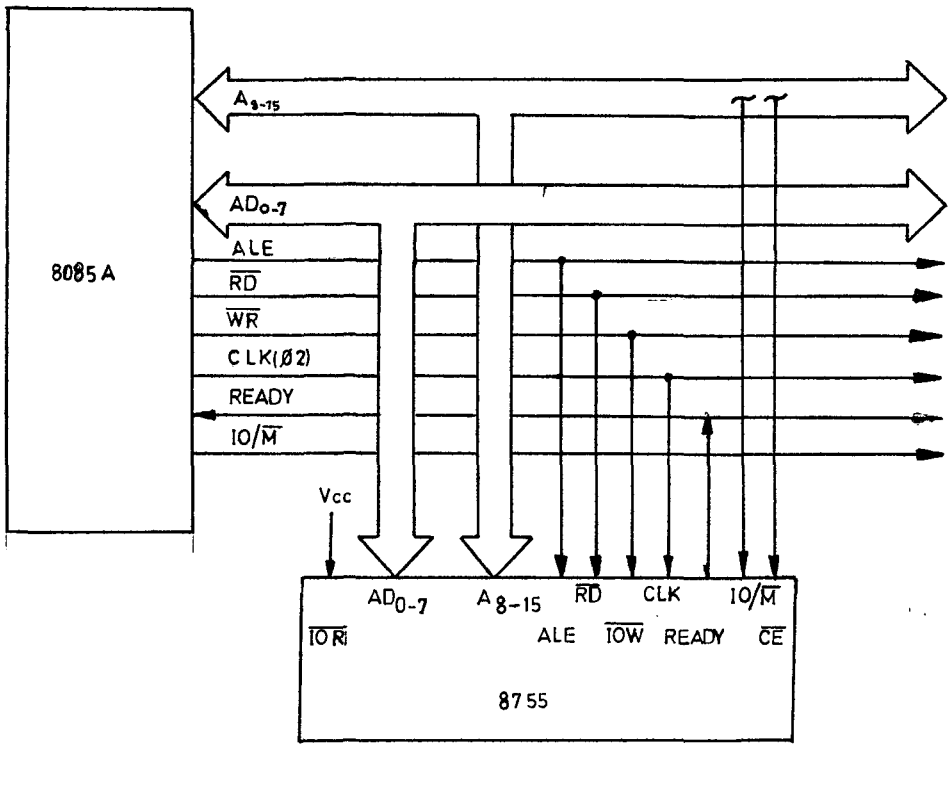
La parte de entrada/salida consiste en dos puertos de propósito general, cada uno posee 8 líneas siendo cada línea progamable como entrada ó como salida.

En la figura siguiente podemos ver la forma del chip con sus terminales:



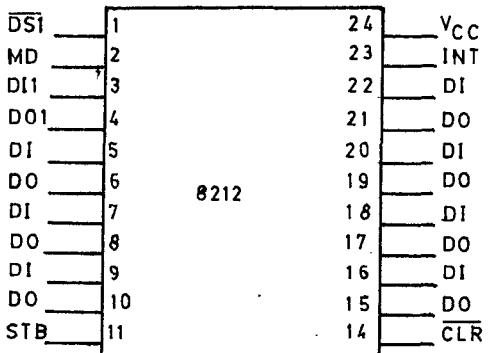
Para profundizar más en las características internas del chip, consultar las hojas que proporciona el fabricante (anexo 1).

El esquema siguiente nos muestra la forma de conectar el 8755 al 8085.



2.1.7-CARACTERISTICAS DEL 8212. LATCH DE DIRECCIONES.

Al encontrarse multiplexado el bus de direcciones y de datos, como sólo tenemos disponibles los 8 bits del bus de direcciones durante el primer ciclo de reloj, cuando sería necesario que estuvierán durante la totalidad del ciclo, tenemos que emplear un circuite de latch (de retención) para mantener el estado de esos 8 bits durante la totalidad del ciclo. El circuito que nos hace ésta misión es el 8212. Es un chip de 24 patillas con las siguientes funciones cada una de ellas:



.- $D.I._1$-$D.I._8$= Datos de entrada.

Entrada de datos, van conectadas al bus de datos bidireccional del microprocesador.

.- $D.O._1$-$D.O._8$= Datos de salida.

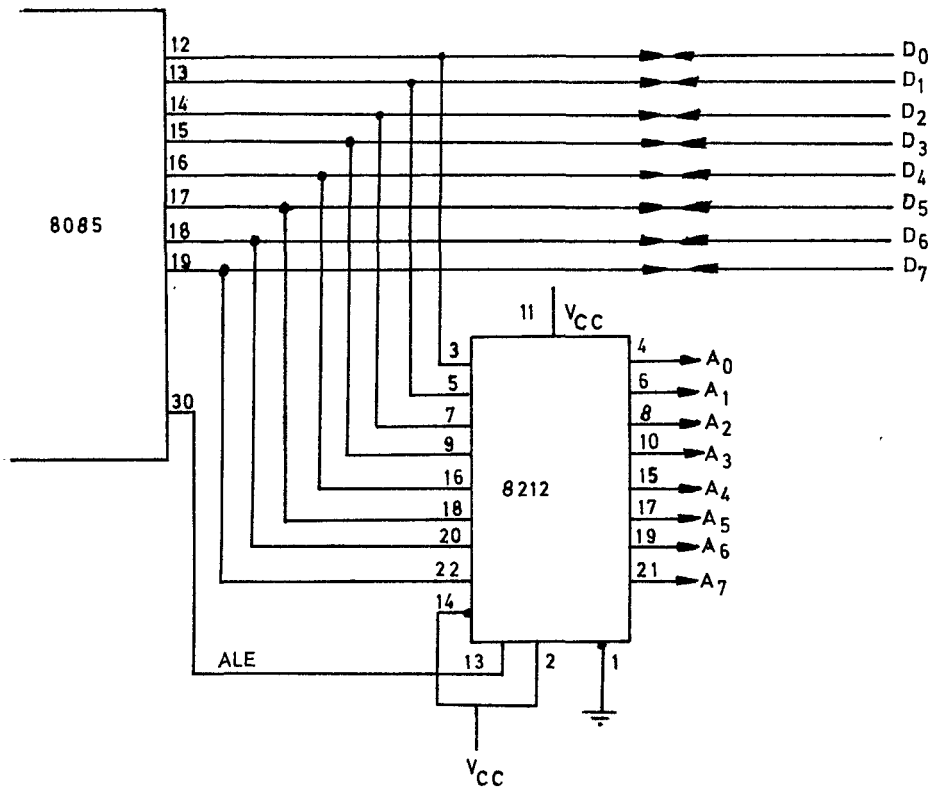
Constituyen los 8 bits de menor peso del bus de direcciones. Estas salidas estarán en estado de alta impedancia hasta que el chip sea habilitado.

.- MD - STB = Control.

Señales de entrada que actúan de control.

Teniendo en cuenta que los 8 bits de menor peso del bus de direcciones estarán disponibles para ser usados,

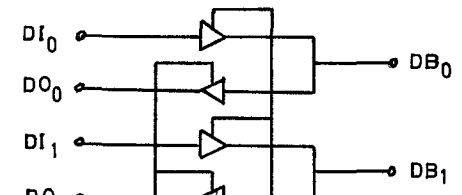
junto con los 8 bits de mayor peso del bus, cuando la señal de salida del mp.:ALE, se encuentre a nivel alto; el 8212 debe ser habilitado por medio de esta señal. Es decir,que la conexión entre el 8085 y el 8212 nos queda de la forma siguiente: (las conecciones se realizan directamente debido a que existe compatibilidad entre los circuitos)



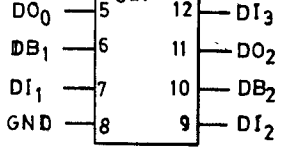
2.1.8.-CARACTERISTICAS DEL 8216.

El 8216 es un bus bi-direccional de cuatro bits conductor/receptor.

En la figura siguiente podemos ver el diagrama lógico del chip:



Posee los siguientes terminales:



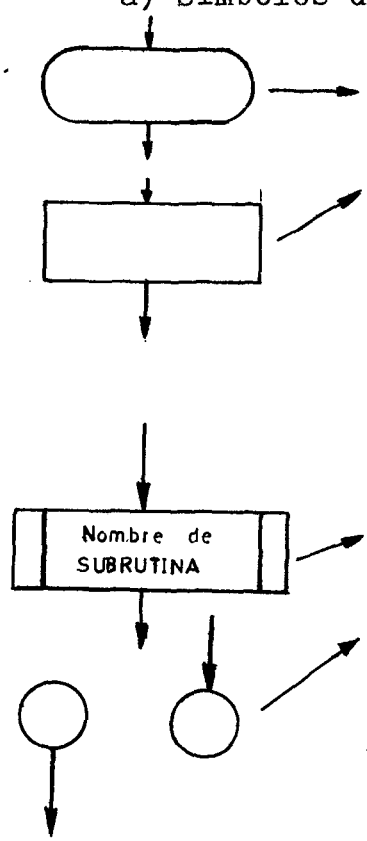
DB_0 - DB_3	DATA BUS BIDIRECCIONAL.
DI_0 - DI_3	DATA INPUT.
DO_0 - DO_3	DATA OUTPUT.
$\overline{DIEN}$	DATA ENABLE. DIRECCION CONTROL.
$\overline{CS}$	CHIP SELECT.

3-ANALISIS DEL PROGRAMA MONITOR DEL SDK'85.

Para el análisis del programa monitor del SDK'85 se realizan organigramas (a nivel general) de los distintos comandos que posee el SDK'85 y de aquellas funciones (rutinas) más complejas. El paso siguiente es analizar cada comando y rutina del programa, instrucción por instrucción, es decir, en organigramas a nivel de registros. Para poder comprender mejor este análisis es aconsejable que al mismo tiempo que se analiza un comando ó rutina en el organigrama a nivel de registros, se vaya viendo dicho comando ó rutina escritos en lenguaje ensamblador; En el anexo 2, podemos ver todos los comandos y funciones del programa editados en lenguaje ensamblador.

Para poder comprender mejor los organigramas a nivel de registros realizados hemos de tener presente las siguientes consideraciones:

a) Símbolos de un organigrama:



Símbolo de principio y de final.

Bloque de asignación. Sirven para definir en cada momento del programa el contenido de los registros ó el valor de una constante, de otra variable y el resultado de una operación aritmética.

De llamada a alguna subrutina.

De conexión. Se usan para unir una parte del organigrama con otra, cuando por motivos físicos, no se puede realizar

directamente.

b) Expresiones que nos vamos a encontrar en los diagramas de flujo realizados:

A←((X)) ; "X"= Posición de memoria, puede ser indicada por un nombre simbólico.

"A"= Acumulador.

Esta expresión nos indica que el contenido de la posición de memoria "X" lo ponemos en el acumulador.

R ó X←R ; "R"= Cualquier registro del 8085.

El contenido del registro lo ponemos en la posición de memoria indicada por "X" ó en otro registro.

CP←Función ; Función= Puede ser cualquier rutina del programa monitor.

La dirección de la rutina indicada la ponemos en el registro CP (con lo que se pasará a ejecutar dicha rutina).

CP←Stack ;

La dirección guardada en las dos primeras posiciones del Stack la ponemos en el CP.

rp←X ;

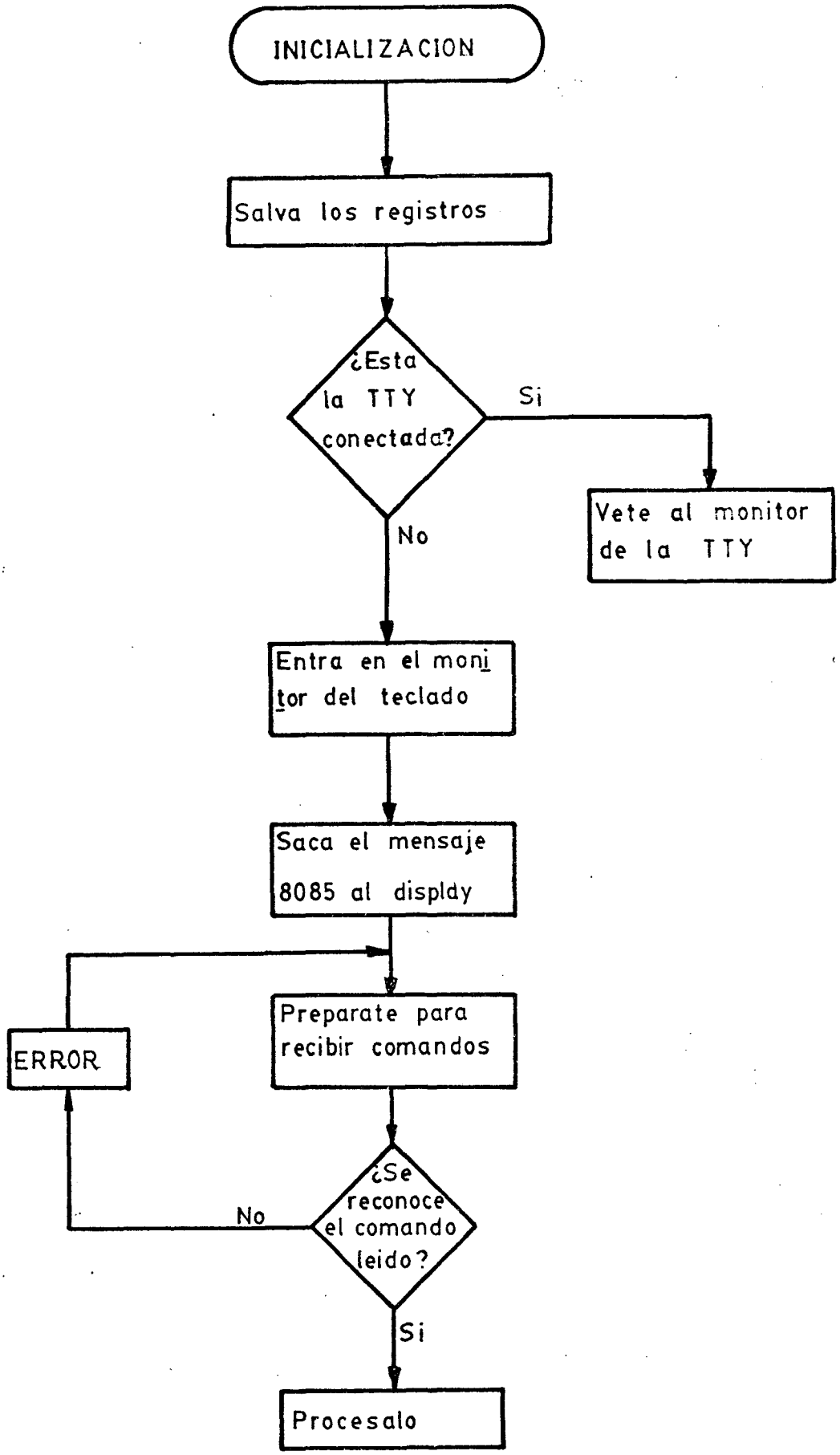
La dirección indicada por el nombre simbolico "X" la ponemos en el par de registros señalados.

A←A $\overset{\text{operación lógica}}{}$ Dato ;

Realizamos la operación lógica indicada, entre el contenido del acumulador y el dato. El resultado de la operación lo ponemos en el acumulador.

HL←((X+1,X)) ;

El contenido de la posición indicada por el nombre simbolico "X" lo ponemos en el registro L y el contenido



de la posición siguiente en el registro H.

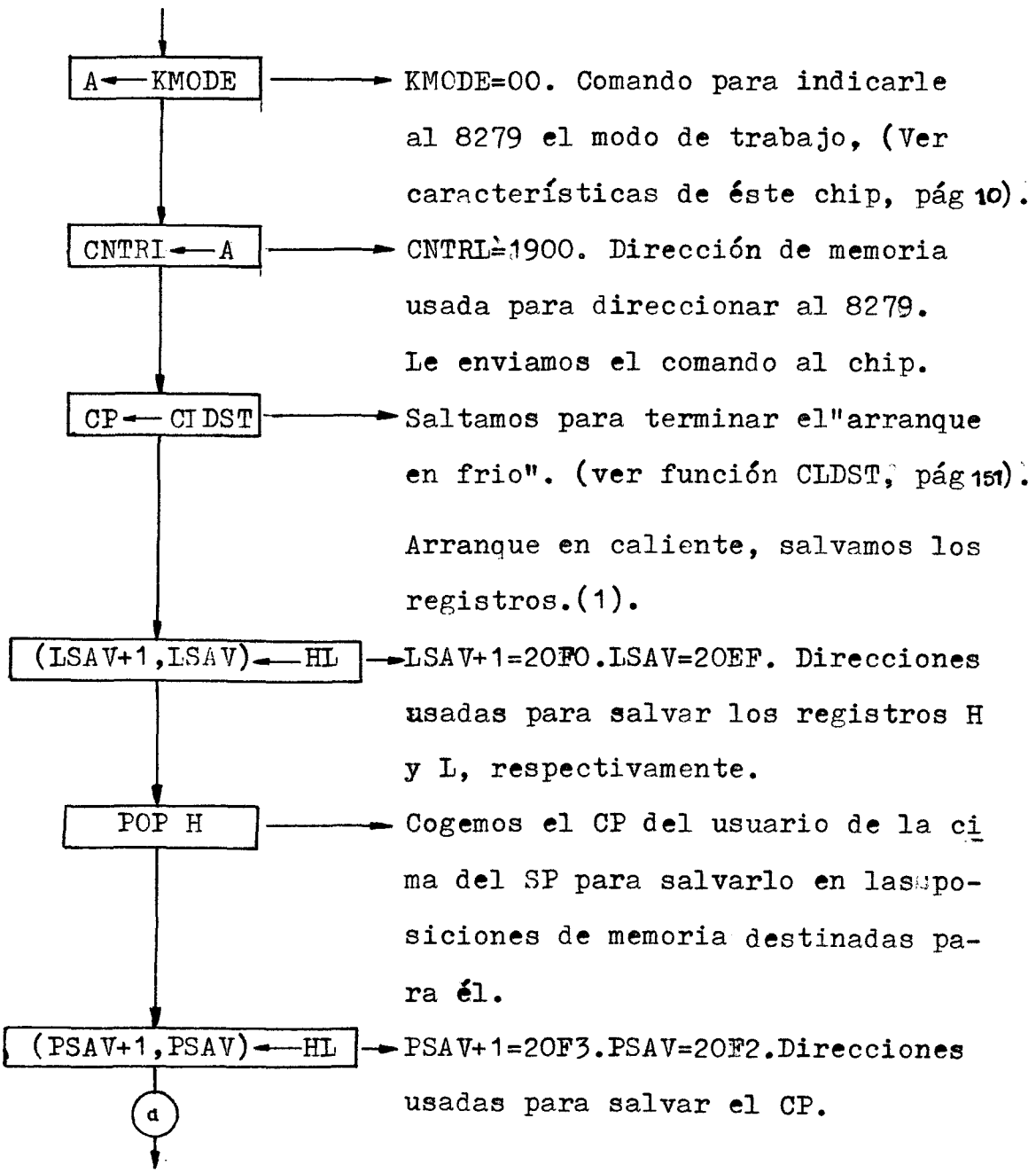
Stack ←—CP;CP ←—rutina ;

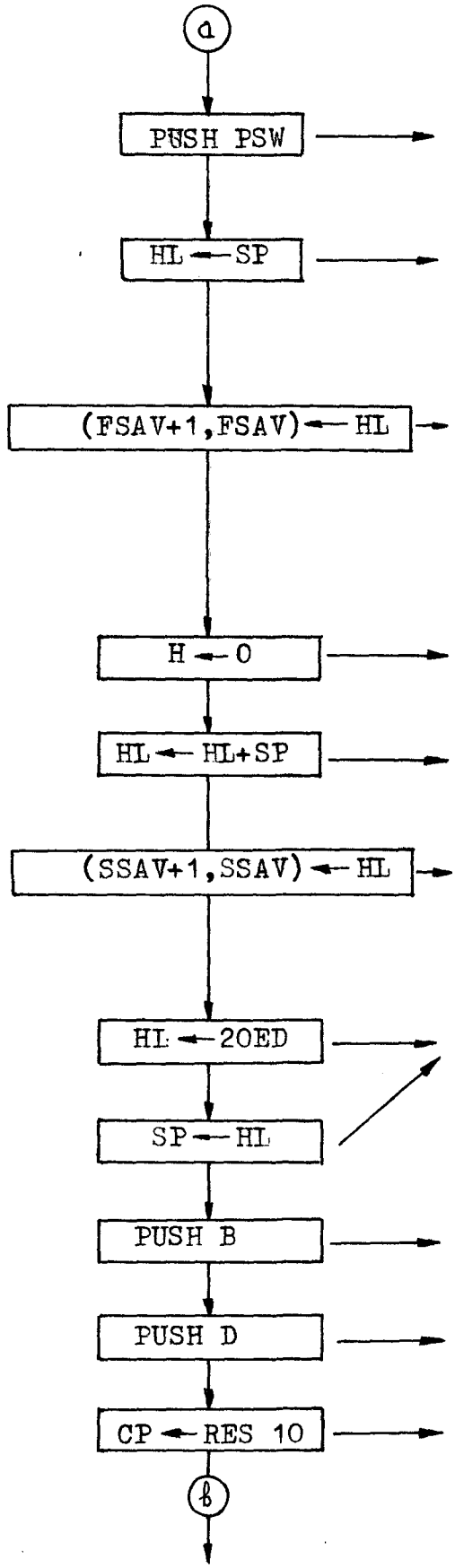
Ponemos en el CP la dirección de la rutina indicada y guardamos el punto de ruptura en el Stack.

.....

3.1.-PRINCIPIO DEL PROGRAMA.

"COLD START" (Arranque en frio).





Ponemos el valor del acumulador y de la palabra de estados en el SP.

El contenido de las dos primeras posiciones del Stack los ponemos en los registros HL.

FSAV+1=20EF.FSAV=20ED. Direcciones del SP del monitor usadas para salvar el valor de la palabra de estados y del acumulador.

Limpiamos HL.

Ponemos en HL la dirección del SP del usuario para salvarla.

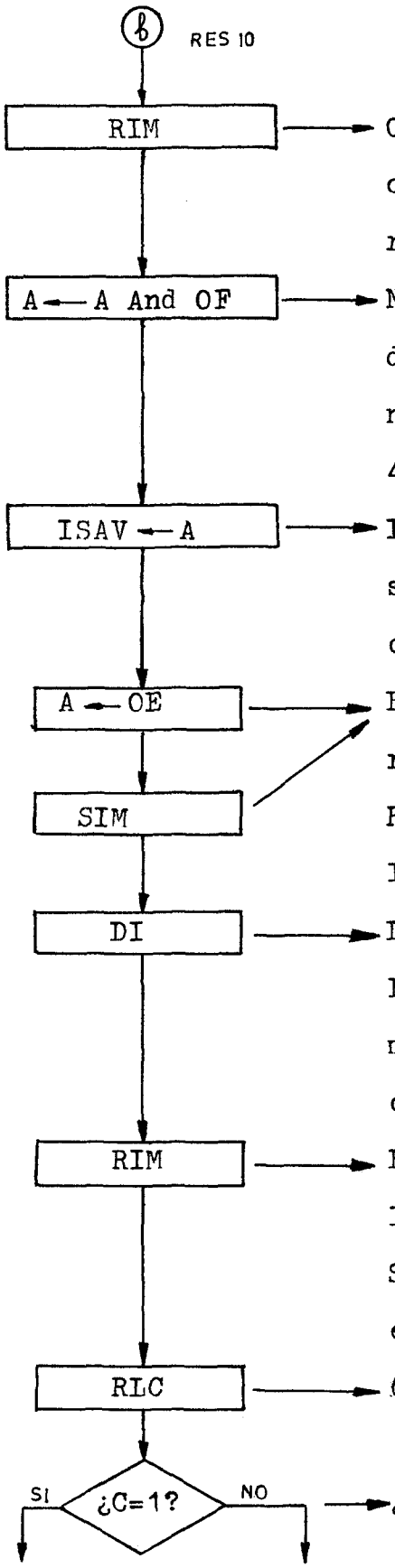
SSAV+1=20F5.SSAV=20F4. Direcciones usadas para salvar el contenido del SP del usuario.

Ponemos en el SP la dirección a partir de la cual se salvarán los contenidos de los registros.

Salvamos B y C.

Salvamos D y E.

RES 10=003F. Saltamos para dejarle sitio a los vectores de interrupción (Ver el programa en lenguaje ensamblador, anexo 2).



Cogemos el estado de las interrupciones del usuario y de las máscaras de interrupción.

Nos quedamos con el bit que nos indica el estado de las interrupciones y con los de las máscaras (los 4 bits de menor orden del Ac.)

ISAV=20F1. Dirección usada para salvar el estado de las interrupciones y de las máscaras.

Enmascaramos las interrupciones para usar el monitor. La interrupción RST 5.5 no se enmascara porque es la que usa el teclado.

DI= Disable Interrupts. Deshabilitamos las interrupciones mientras el programa monitor está corriendo.

Ejecutemos ésta instrucción para leer el estado lógico del terminal SID (del 8085).Para averiguar si está la TTY conectada. (2).

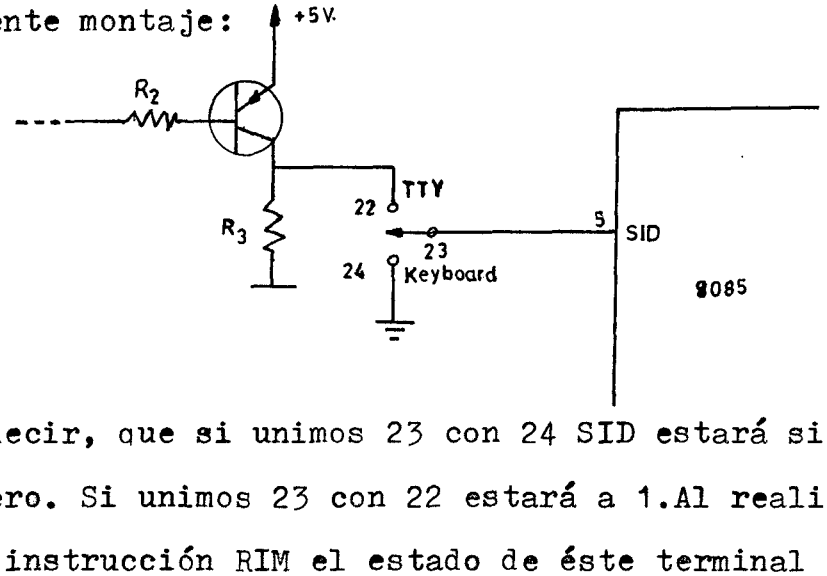
(2).

¿ Está la TTY conectada?

.....

(1) El "arranque en caliente" se produce cuando, estando utilizando el SDK'85, queremos que se resetee el monitor pero que nos guarde lo que tenemos escrito. El "arranque en frío" se produce al encenderse el SDK'85.

(2) Si nos fijamos en el esquema del SDK'85 (anexo 3) vemos que en el terminal SID del 8085 existe el siguiente montaje:

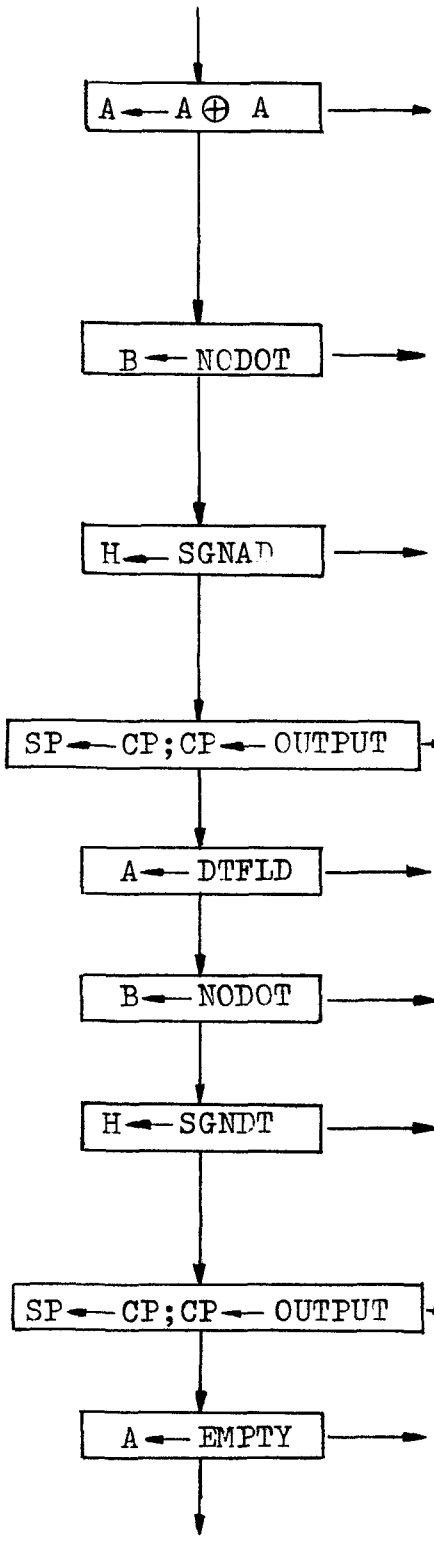


Es decir, que si unimos 23 con 24 SID estará siempre a cero. Si unimos 23 con 22 estará a 1.Al realizar una instrucción RIM el estado de éste terminal se pone en el bit número 7 del acumulador. Esto nos indica que si el carry es 1 después de haber realizado las instrucciones RIM y RLC, la TTY está conectada. Si C=0, la TTY no está conectada (trabajamos con el teclado).

.....

3.2 COMIENZO DEL PROGRAMA MONITOR DEL TECLADO.

9 Sacar el mensaje "8085" al display.



Para indicarle a OUTPUT que se va a usar el campo de direcciones. Al realizar la OR Exclusiva anulamos el contenido del Ac. (1).

NODOT=00. Le indica a OUTPUT que saque el punto en el campo de direcciones.

SGNAD=03A6. A partir de ésta dirección tenemos el mensaje "__80". (Ver pág 22 del anexo 2).(2).

Sacamos el mensaje al campo de direcciones. (1).

DTFLD=01. Le indica a OUTPUT que se va a usar el campo de datos.

Para que OUTPUT no saque el punto en el campo de datos.

SGNDT=03AA. A partir de ésta dirección tenemos el mensaje "85". (Ver pág 22 del anexo 2).(2).

Sacamos el mensaje "85" al campo de datos. (1).

EMPTY=80. Ponemos el flag que indica cuando está el buffer vacio en el buffer de entrada.(Para que lo use la función RDKBD).

(1) Ver función OUTPUT,(pág 141).

(2) Tenemos que poner ésta dirección en los registros HL, para que pueda ser usada por la función CUTPUT. Esta función saca al display (campo de direcciones) los carácteres que se le indiquen en los contenidos de las posiciones señaladas (la posición que se pone en HL, más las tres siguientes).

.....

3.3-COMANDOS DEL PROGRAMA MONITOR.

En éste punto vamos a analizar los distintos comandos que posee el programa monitor del SDK'85. Estos son:

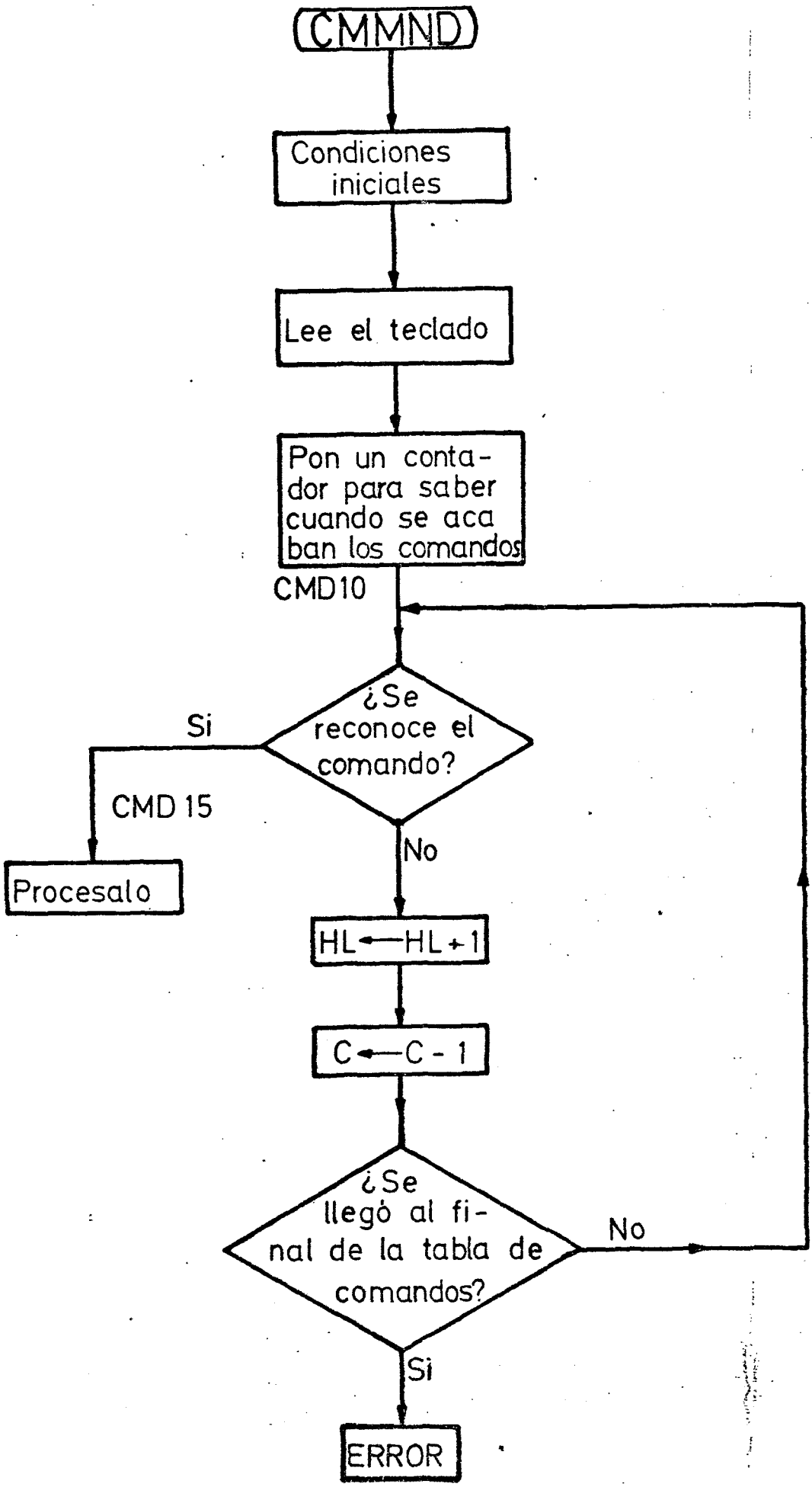
.- EXAM."Examina y modifica el contenido de los registros."

.- GO. "Ejecuta el programa del usuario."

.- S.STEP. "Ejecuta el programa instrucción a instrucción."

.- SUBST. "Sustituye y examina el contenido de las direcciones de memoria".

Antes de entrar en los distintos comandos, se analiza la función "CMMND", por medio de la cual se reconocen los distintos comandos.
(La tecla Vector Interrup,la podemos considerar como un comando, sirve al usuario como botón de interrupción.)



Función: CMMND.

"Reconocer comandos."

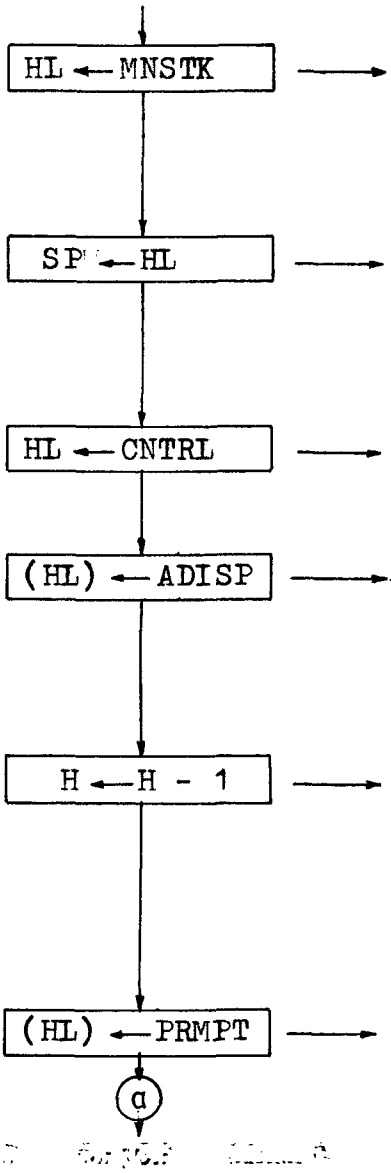
La función coge el comando pulsado por el usuario, y lleva al programa monitor a la posición de memoria donde está dicho comando para procesarlo.

Si la tecla pulsada no es ningún comando saca el mensaje "ERR".

Entradas: ninguna.

Salidas: ninguna.

LLamadas: RDKBD, ERR, SUBST, EXAM, GOCMD, S.STEP.



MNSTK=20E9. Principio del stack del monitor. (Zona de memoria destinada a salvar los registros.)

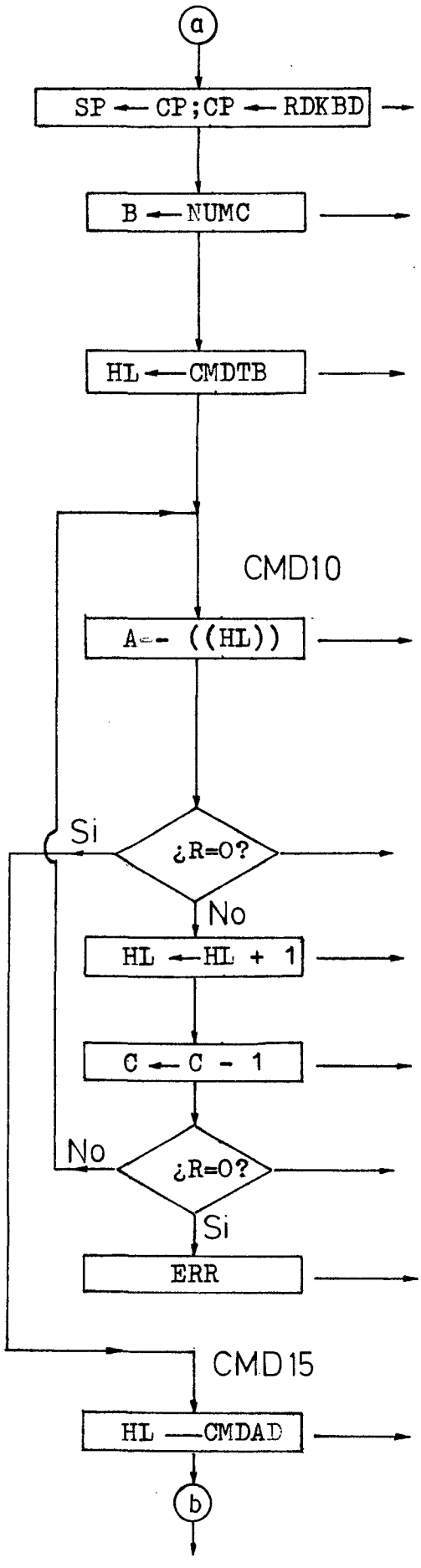
Ponemos la direcdión anterior en el Stack Pointer(SP) para inicializarlo.

CNTRL=1900. Dirección de memoria usada para direccionar al 8279.

ADISP=90. Comando de control que se le envía al 8279 para usar el campo de direcciones.

Al decrementar H nos queda en HL la dirección 1800. Dirección usada para enviar caracteres al display.

PRMPT=FB. Ponemos este dato en la posición indicada en HL. (1).



Lee el teclado. (2).

NUMC=04 (existen cuatro comandos) Ponemos un contador para saber cuando se acaban los comandos.

CMDTB=0378. Dirección de memoria donde empieza la tabla con los comandos (pág 21 (anexo 2)). Ponemos en HL la dirección de la tabla donde están los comandos.

Esta comparación se realiza para ver si el comando que tenemos en el acumulador es el mismo que el primero de la tabla.(3).

¿ Es el primer comando de la tabla?

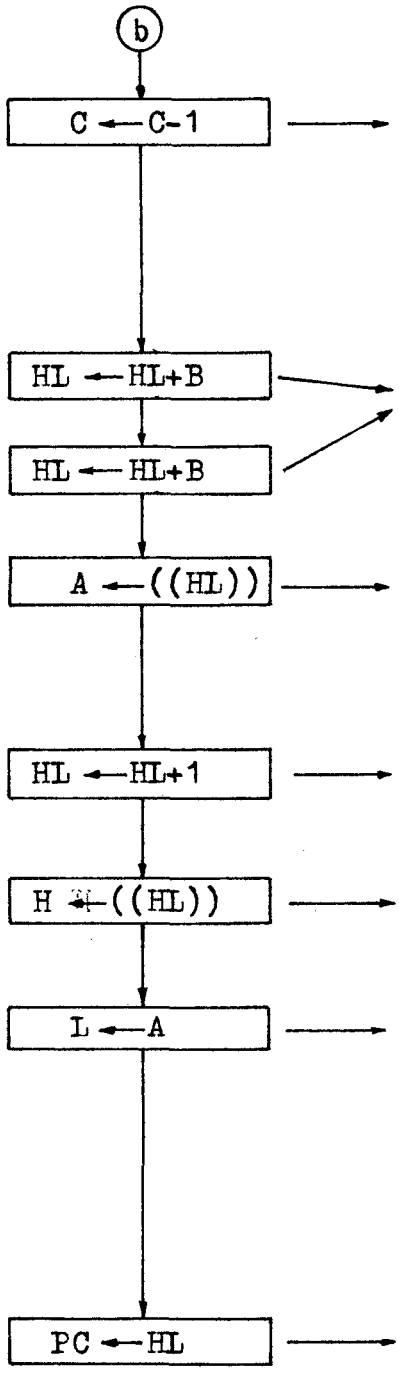
Cogemos la siguiente posición de la tabla.

Para ver si se llegó al final de la tabla.

¿Se terminó la tabla?

Saca el mensaje " ERR " porque el comando es desconocido.

CMDAD=037C. Dirección donde empieza la tabla con las direcciones de los comandos (ver pág 22). (anexo 2)



Ajustamos el contador de comandos (este contador actúa como punto de entrada para la tabla de direcciones) (4).

Hacemos la suma dos veces porque cada dirección de los comandos posee dos bytes. (5)

Ponemos en el acumulador la palabra de menor orden que forma la dirección del comando.

Cogemos la otra posición de la tabla. (6).

Ponemos en H la palabra de mayor orden de la dirección del comando (6).

La palabra de menor orden (que estaba en el acumulador) la pasamos al registro L. Con esto tenemos la dirección del comando señalado por el usuario en HL.

Salta al comando señalado para procesarlo.

.....

(1) Con esto enviamos al display el carácter PRMPT (una raya). FB = 1 1 1 1/1 Ø 1 1 Si ponemos esto en un dígito del display (los led del display son activos a nivel bajo) nos aparecerá: ⊟ (en el primer dígito del campo de direcciones).

(2) El código que se genera al pulsar los comandos es el siguiente:

GO	12H.
SUBST	13H.
EXAM	14H.
S.STEP	15H.

(3) Si al hacer esta comparación el resultado es 0, nos indica que el comando pulsado es el primero de la tabla (comando GO). Si el resultado no es 0, nos indica que se pulsó otro comando distinto de GO. Para averiguar cuál es vamos cogiendo las siguientes posiciones de la tabla y comparando el contenido de cada posición con el del acumulador, hasta que el resultado de la comparación sea 0. Si se llega al final de la tabla y el resultado no ha sido nunca 0, significa que la tecla pulsada por el usuario no es ningún comando (el contenido del Ac. es distinto de 12H, de 13H, de 14H, y de 15H.

(4) Vamos a suponer que el comando pulsado es EXAM. Al llegar a esta instrucción el contenido del registro C (del contador) es 02. El hecho de decrementarlo otra vez es debido a la posición que ocupa cada registro en la tabla CMDAD. (al llegar a este punto la comparación A - ((H)) se ha realizado tres veces, en el caso del registro EXAM, y la operación C C - 1 sólo 2).

(5) La operación: HL HL + BC se realiza para coger de la tabla CMDAD la dirección que nos interesa.

Siguiendo con el caso anterior, en DC tenemos: 01 y en HL: 037C, es decir, en HL nos quedará:

037C + 01 = 037D

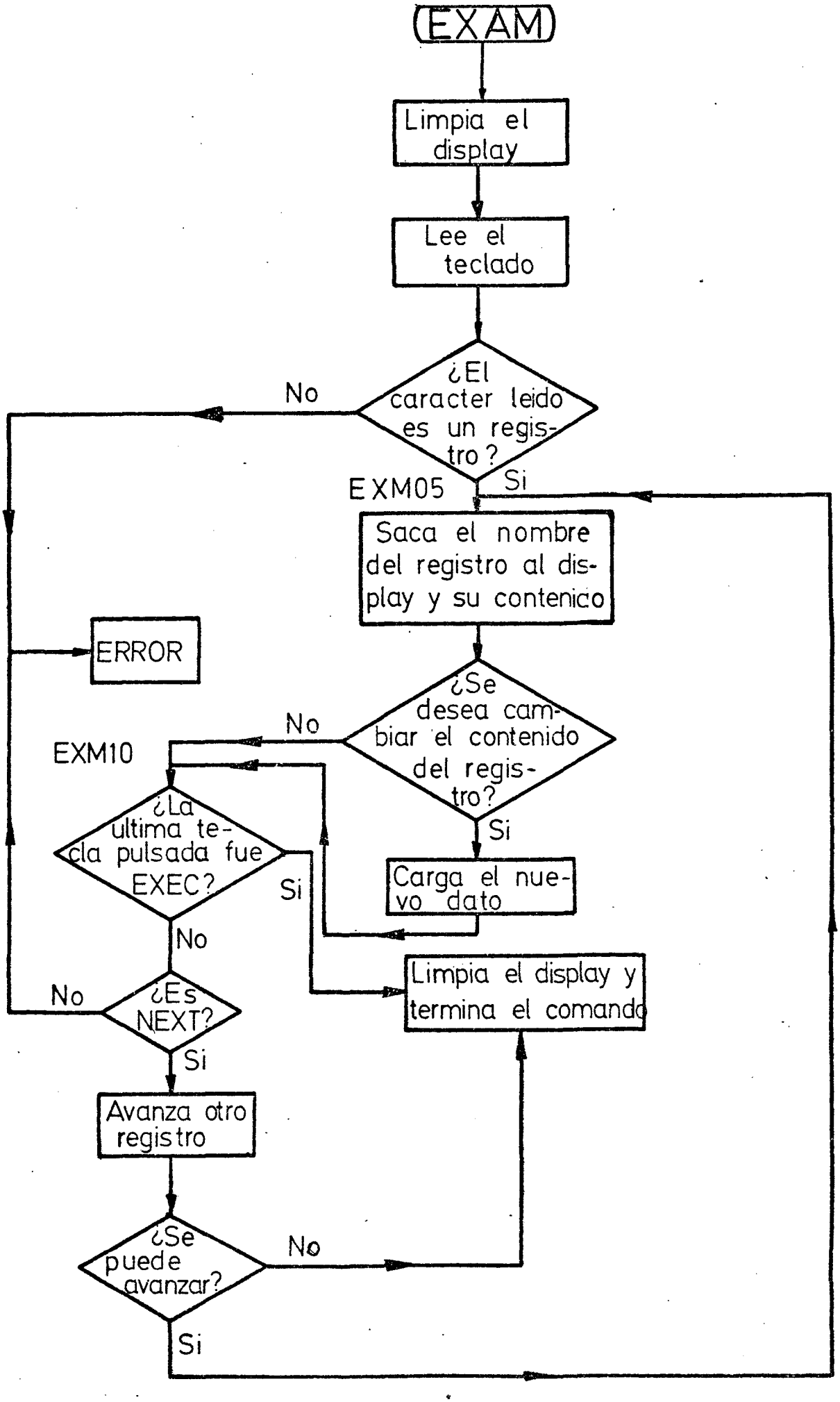
Si volvemos a realizar la operación HL HL+BC, nos quedará en HL la direección siguiente:

037D + 01 = 037E

Fijándonos en la tabla de direcciones, vemos que el contenido de esta dirección es 92 (byte de orden bajo de la dirección buscada).

(6) En la dirección siguiente, la 037F, está el byte de mayor orden de la dirección (00). Es decir, en la dirección 0092 del programa monitor tenemos el comando EXAM.

.....



Comando: EXAM.

" Examina y modifica los registros !

Con ésta comando podemos exponer y modificar los contenidos de los registros de la CPU. Al pulsar este comando, el display se borra y aparece un punto en el margen derecho del campo de direcciones. Luego se ha de pulsar una tecla que designe algún registro, sino aparecerá el mensaje "-ERR" en el display. Al pulsar el nombre de algún registro, nos aparecerá en el display el nombre de ese registro (en el campo de direcciones) y su contenido en el campo de datos.

El contenido de los registros se puede cambiar pulsando los dígitos deseados, el nuevo valor sólo se carga después de pulsar una de las teclas "NEXT" ó "EXEC". Con "NEXT" cargamos el nuevo dato, y se muestra en el display el nombre y el contenido del próximo registro de la tabla (si no se puede avanzar de registro aparecerá en el display el mensaje "-ERR".)

Con "EXEC" se carga el dato, y se termina el comando.

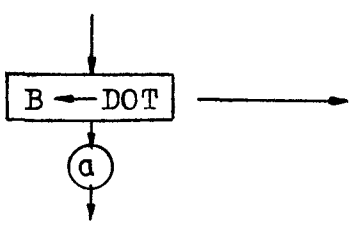
Entradas: ninguna.

Salidas: ninguna.

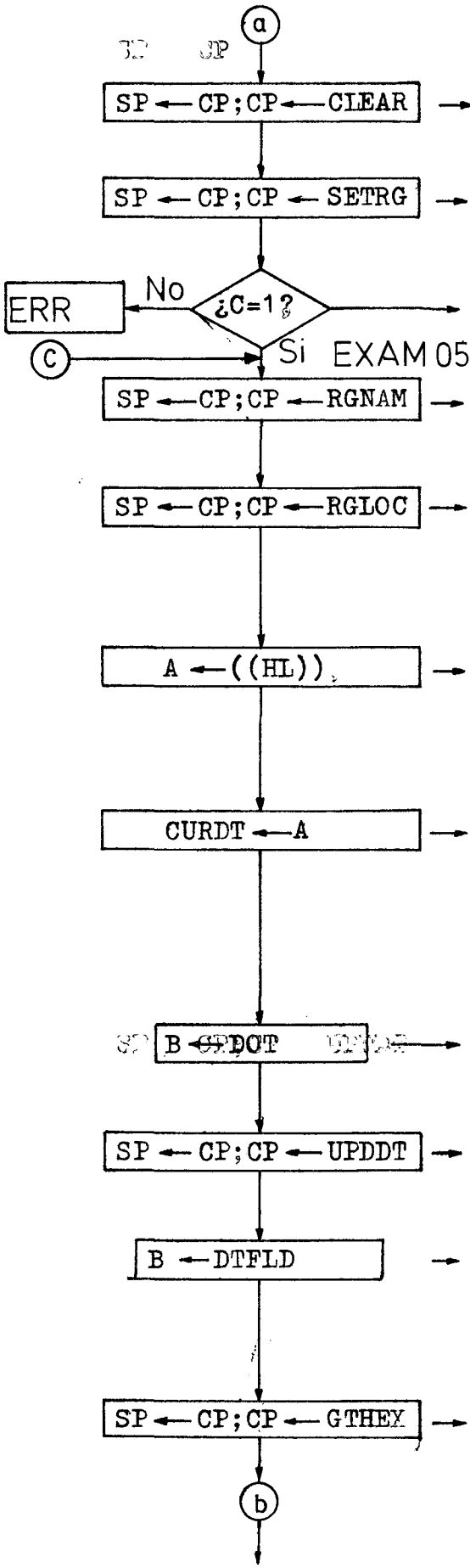
Llamadas: RDKBD, ERR, SETRG, RGNAM, RGLOC, UPDDT, GTHEX, NXTRG, CLDIS.

(Ver el organigrama del comando)

(Ver el comando en lenguaje ensamblador, pág 6).(Anexo 2)



DOT=01.Para indicarle a CLEAR que saque el punto en el campo de direcciones del display.



Limpiamos el display. (1).

Convertimos el registro pulsado en registro puntero. (2).

¿ Se pulsó algún registro ? (3).

Sacamos el nombre del registro al campo de direcciones. (4).

Ponemos en HL la posición de memoria donde se guarda el contenido del registro señalado.(5).

Ponemos el contenido del registro en el Ac. (para poder sacarlo al campo de datos del display)

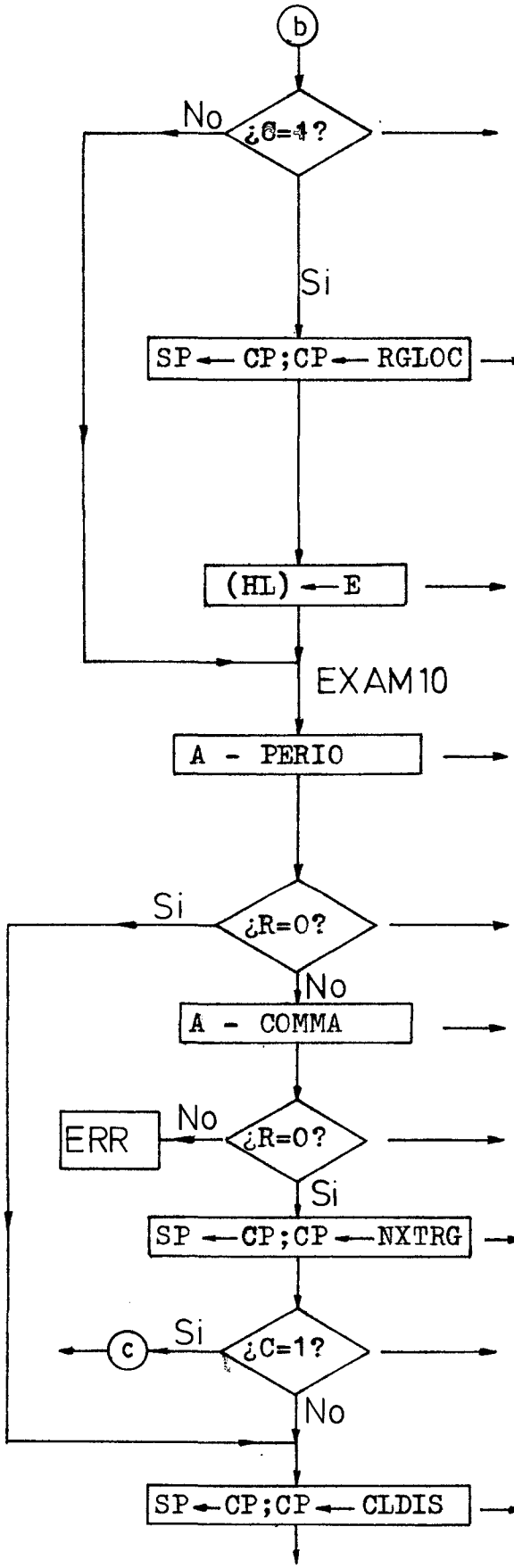
CURDT=20F8. Dirección usada para poner el data que se desea sacar al campo de datos, por medio de la función UPDDT.

Para indicarle a UPDDT que saque un punto en el display. (6).

Sacamos el contenido del registro al campo de datos. (6).

DTFLD=01. Para indicarle a la función GTHEX que se va a usar el campo de datos.

Cómprobamos si se reciben nuevos dígitos. (Para ver si el usuario desea cambiar el contenido del registro). (7).



¿ Se va a cambiar el contenido del registro? La función GTHEX pone el carry a 1 cuando recibe algún digito, sino lo pone a 0.

Ponemos en HL la posición de memoria donde vamos a valvar el nuevo contenido del registro señalado. (5)

Lo que está en E (el nuevo contenido del registro) lo ponemos en la posición señalada por HL.

PERIO=10H. Hacemos ésta comparación para ver si la última tecla pulsada fue EXEC.

¿Se pulsó EXEC?

COMMA=11H. Para ver si se pulsó NEXT.

¿Se pulsó NEXT?

Cogemos el siguiente registro de la tabla. (8).

¿Se puede avanzar?

Limpiamos el display y se termina el comando. (9).

(1) Ver CLEAR, (pág 109). Si cuando llamamos a ésta rutina tenemos en el registro B = 01, nos saca un punto en el display.

(2) Ver SETRG, (pág 135).

(3) Usamos el carry para averiguar si se ha pulsado algún registro porque la función SETRG pone el carry a 1 cuando recibe algún registro (ó cuando se puede avazar de registro) y lo pone a 0 cuando no se recibe algún registro (ó cuando no se puede avanzar a otro).

(4) Ver RGNAM, (pág 128).

(5) Ver RGLOC, (pág 126).

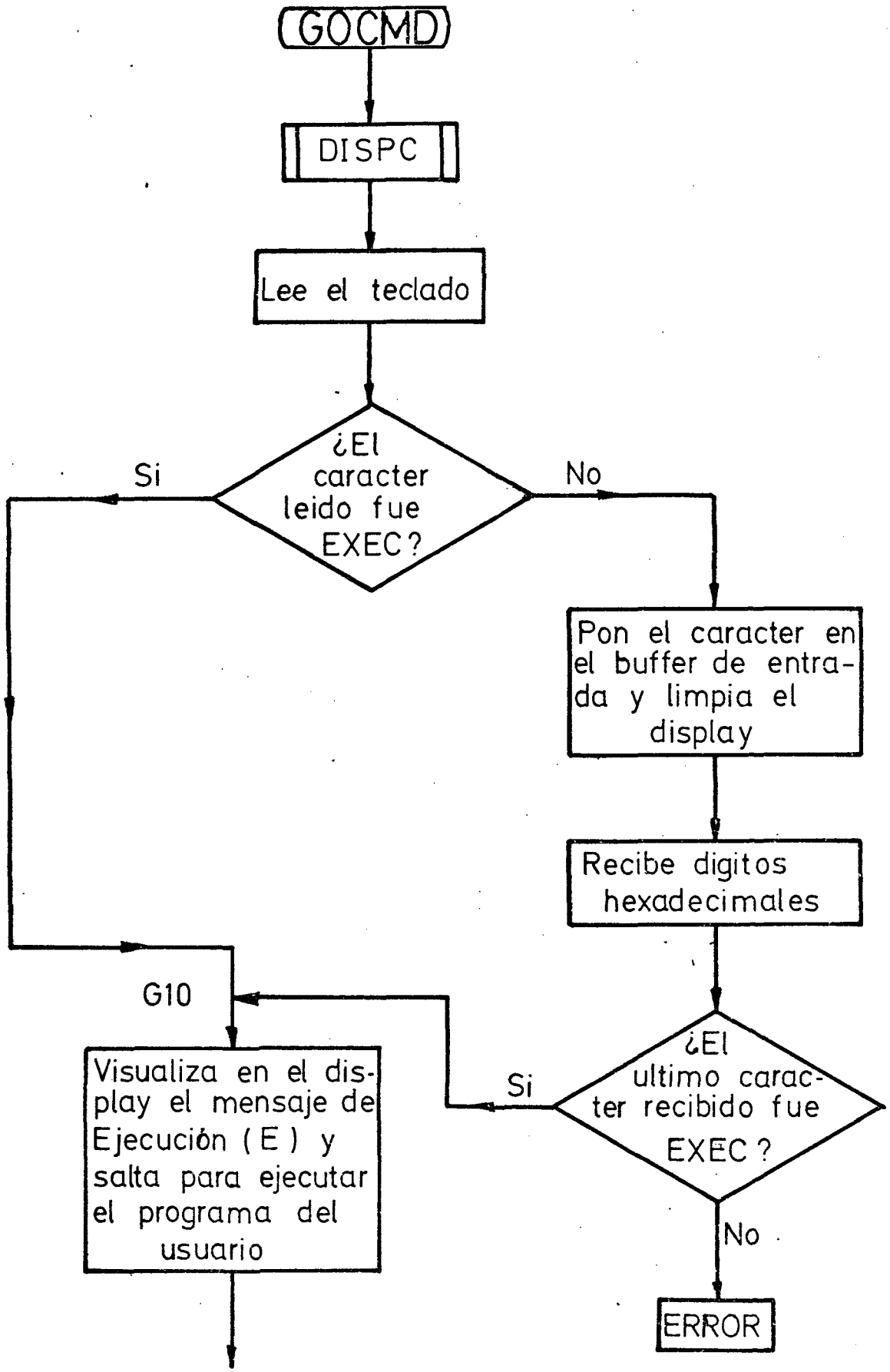
(6) Ver UPDDT, (pág 150).

(7) Ver GTHEX, (pág 119).

(8) Ver NXTRG, (pág 139).

(9) Ver CLDIS, (pág 111).

.....



Comando: GOCMD.

"Ejecuta el programa del usuario".

Por medio de este comando le indicamos al monitor que tiene que ejecutar algún programa que previamente hemos cargado en memoria. Los pasos a seguir para la ejecución de un programa son:

1- Pulsar "GO".

2- Introducir en el campo de direcciones la primera dirección del programa a ejecutar.

3- Pulsar "EXEC".

Al pulsar "EXEC" se limpia el display, aparece en el campo de direcciones la letra E (que indica que se está ejecutando algún programa) y se transfiere el control de la CPU a la dirección que deseamos.

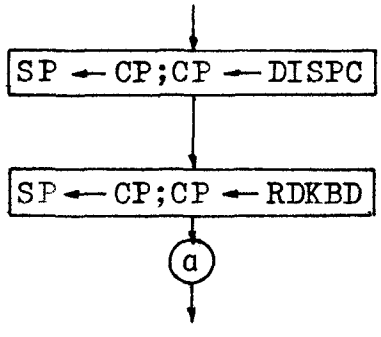
Para que el monitor vuelva a recobrar el control de la CPU se ha de producir un "RESET" ó la ejecución de una instrucción RSTo, RST1 ó JMP ØØ en el programa que se esté ejecutando.

Entradas: ninguna.

Salidas: ninguna.

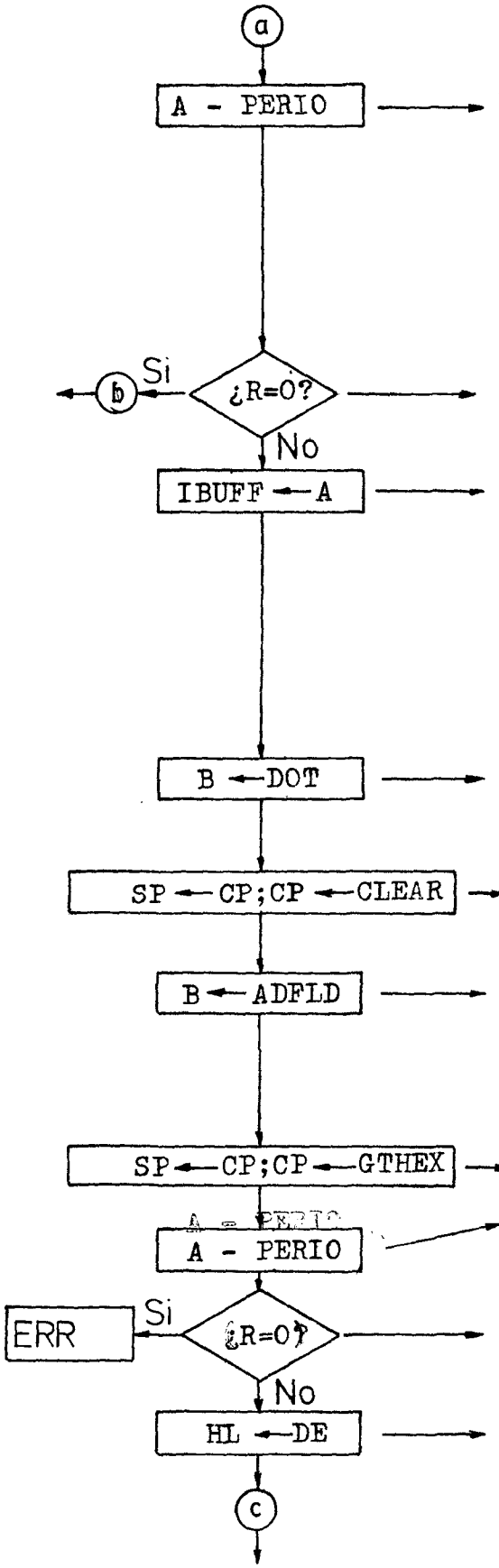
Llamadas: DISPC, RDKBD, CLEAR, GTHEX, ERR, OUTPT.

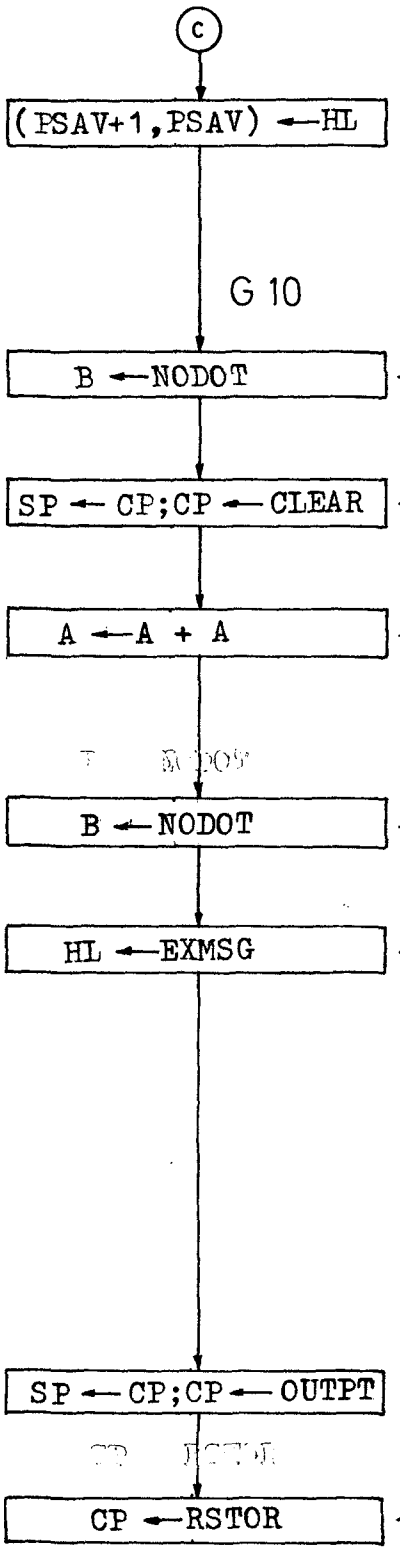
(Ver organigrama del comando, pág. 83)
(Ver el comando en lenguaje ensambladador, pág. 7)



→ Visualizamos el CP en el campo de direcciones. (1).

→ Lee el teclado. (2).





→ PSAV=20F2. PSAV+1=20F3. Dirección de memoria donde ponemos el contenido del CP. Los valores recibidos son el nuevo CP.

→ NODOT=00. Le indica a CLEAR que no saque el punto en el display.

→ Limpiamos el display. (3).

→ Anulamos el contenido del Ac. para indicarle a OUTPT que se va a usar el campo de direcciones.

→ Para indicar que no saque el punto en el display.

→ EXMSG=03A2. Dirección donde tenemos grabada la "E" (que indica que se está ejecutando algún programa). Ponemos esta dirección en HL para que OUTPT saque esta letra en el campo de direcciones del display: E _ _ _.

→ Sacamos la "E" al display. (5).

→ Restauramos los registros del usuario y empezamos a ejecutar su programa. (6).

.....

(1) Ver DISPC, (pág 123).

(2) Ver RDKBD, (pág 145).

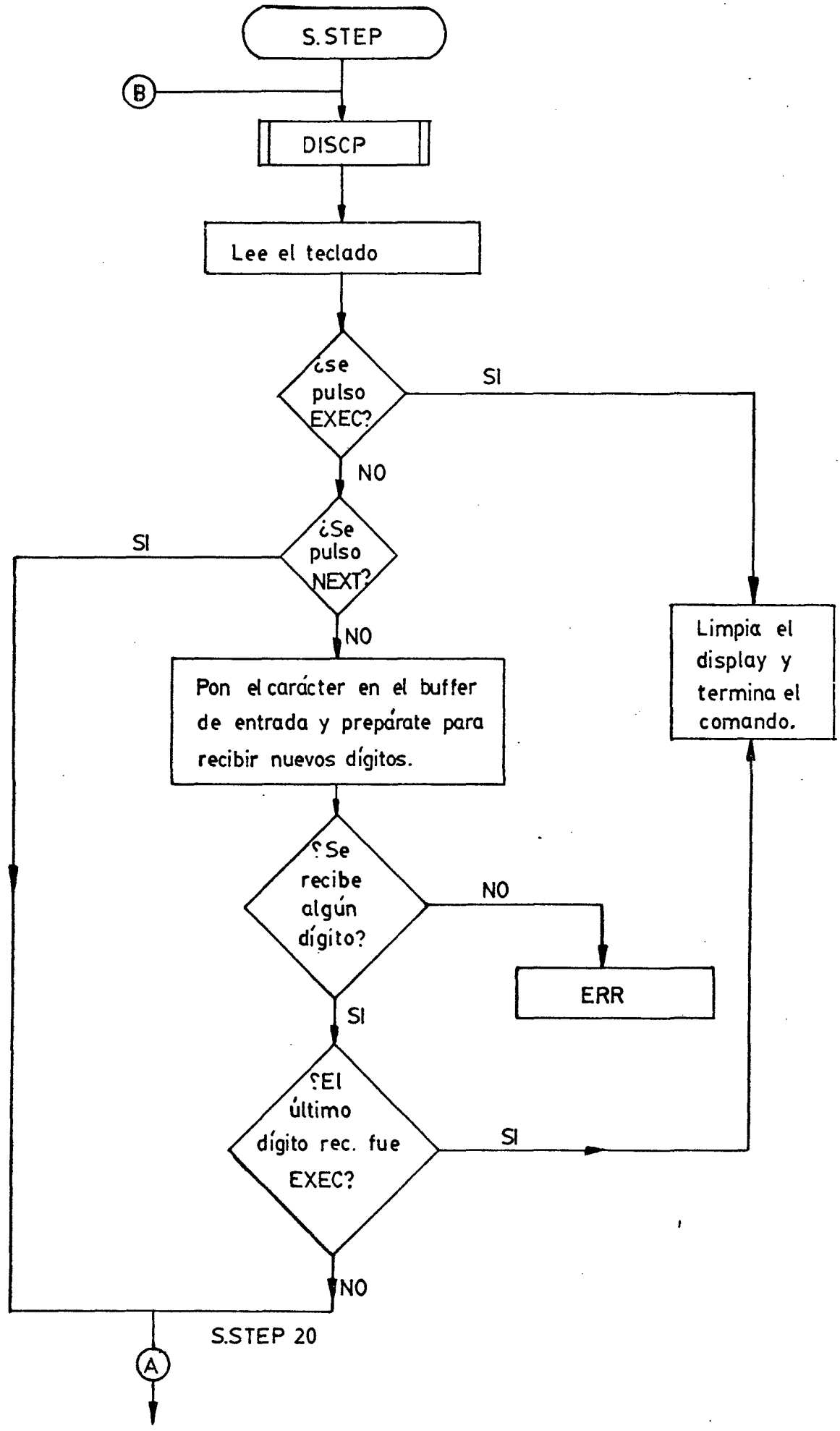
(3) Ver CLEAR, (pág 109).

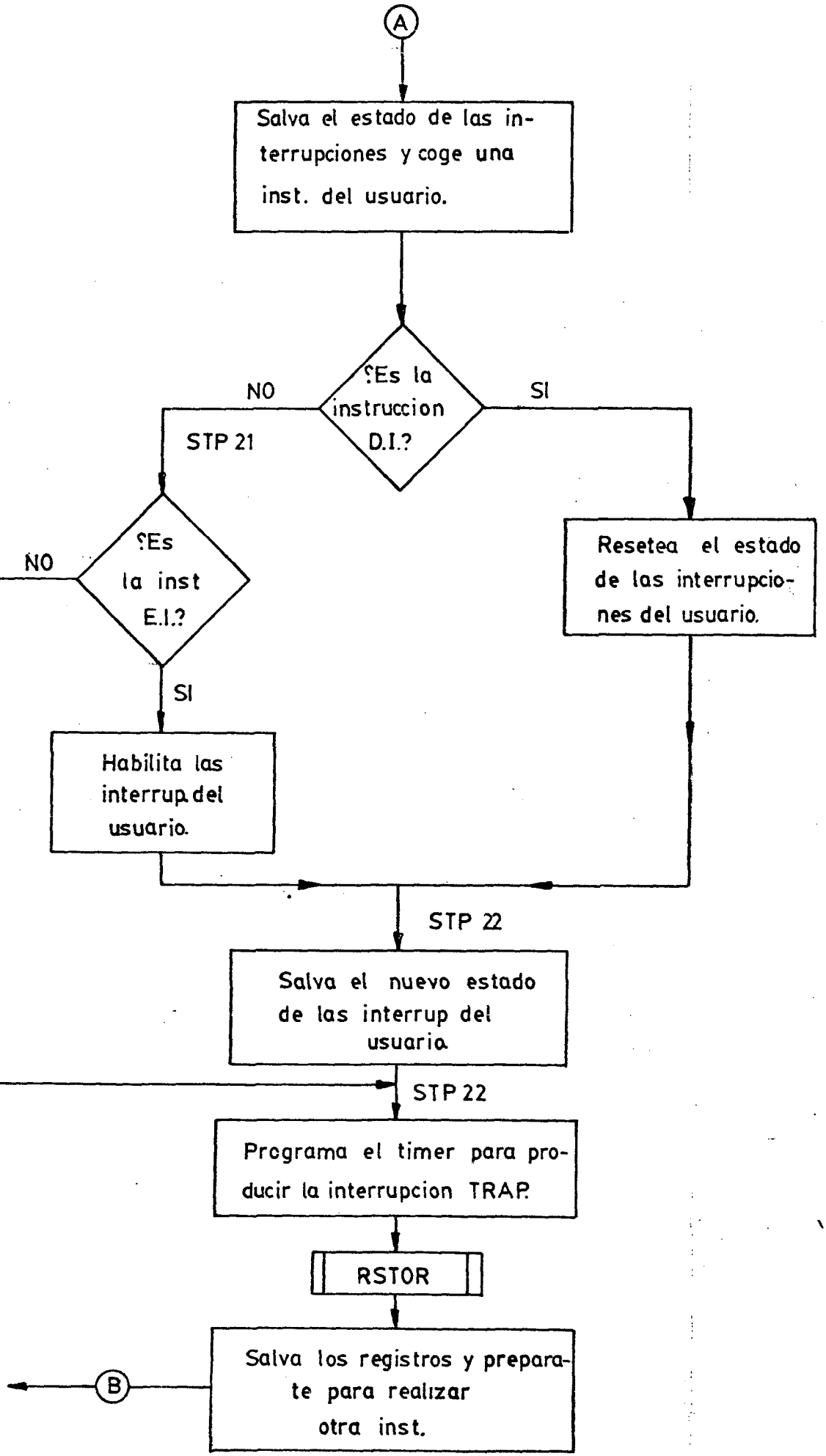
(4) Ver GTHEX, (pág 119).

(5) Ver OUTPT, (pág 141).

(6) Ver RSTOR, (pág 131).

.....





Comando: S. STEP.

"Ejecuta una instrucción del usuario."

Con este comando podemos ir ejecutando el programa del usuario instrucción a instrucción.

Los pasos a seguir para usar este comando son:

1- Pulsar S.STEP.

2- Introducir la dirección deseada.

3- Pulsar "NEXT".

Al pulsar S.STEP aparece en el campo de direcciones del display el contenido del contador de programa (al principio sale una dirección aleatoria) y en el campo de datos el contenido de la dirección señalada en el contador de programa. La dirección inicial del CP (la que aparece en el display) puede ser cambiada según desee el usuario.

Presionando "NEXT" hacemos que la CPU ejecute la instrucción señalada en el CP. Después de la ejecución aparece en el display el nuevo contenido del CP (próxima dirección de memoria) y su contenido (próxima instrucción a ejecutar). En la parte derecha del campo de direcciones nos aparece un punto (junto con la dirección del CP) indicándonos que el monitor está preparado para ejecutar la nueva instrucción señalada en el display.

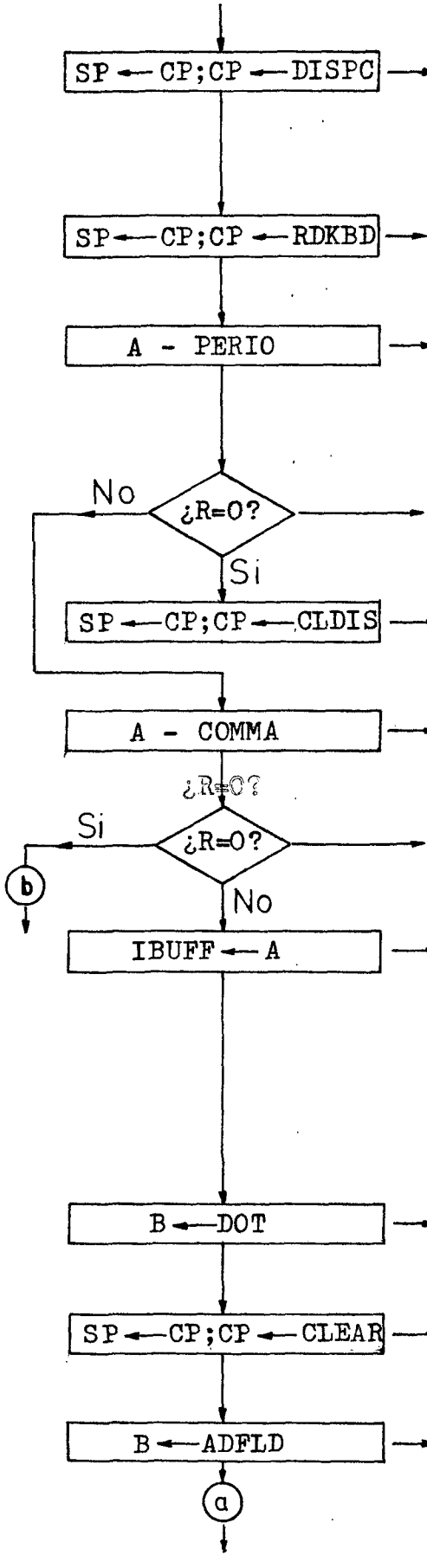
Si se pulsa "EXEC" no se ejecuta ninguna instrucción y finaliza el comando.

Entradas: ninguna.

Salidas: ninguna.

Llamadas: DISPC, RDKBD, CLEAR, GTHEX, ERR, CLDIS. (ver el organigrama del comando).

(ver el comando el lenguaje ensamblador, pág 8)(anexo 2)



→ Visualiza el contador de programa del usuario (al principio nos sale un valor aleatorio). (1).

→ Lee el teclado. (2).

→ PERIO=10H. La comparación la hacemos para averiguar cuál fue la tecla pulsada.

→ ¿ Fue "EXEC"? (el código que se genera al pulsar EXEC es 10H)

→ Limpia el display y termina el comando (3).

→ COMMA=11H. Para ver si fue "NEXT"

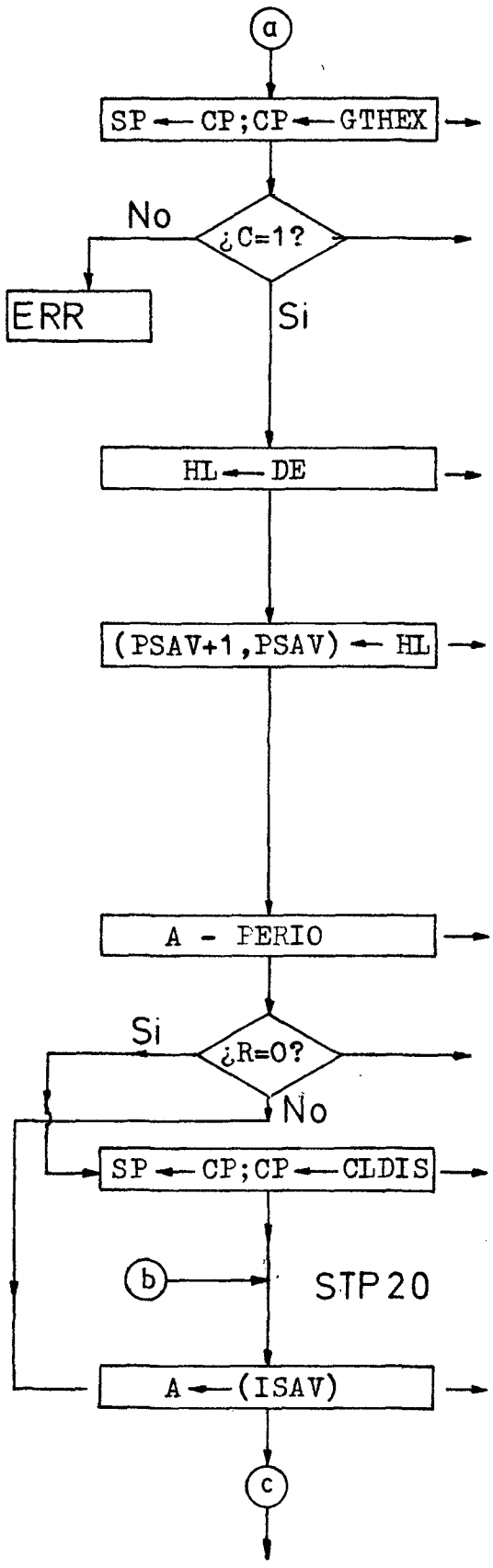
→ ¿Fue NEXT?.(El código que se genera al pulsarla es 11H).

→ IBUFF=20FE. Posición reservada para el buffer de entrada. Ponemos el carácter leído en el teclado (lo tenemos en el Ac.) en el buffer de entrada.

→ DOT=01. Para indicarle a CLEAR que saque el punto en el display.

→ Limpiamos el display. (4).

→ ADFLD=00. Le indicamos a GTHEX que se va a usar el campo de direcciones.



Para ver si se reciben más dígitos hexadecimales. (5).

¿Se reciben dígitos?. Usamos el carry para averiguar la respuesta porque GTHEX lo pone a uno cuando recibe algún dígito.

Ponemos los valores hex. recibidos en HL (la función GTHEX los coloca en DE)

PSAV=20F2. PSAV+1=20F3. Direcciones de memoria donde ponemos el contenido del contador de programa. Los valores recibidos forman la nueva dirección del CP.

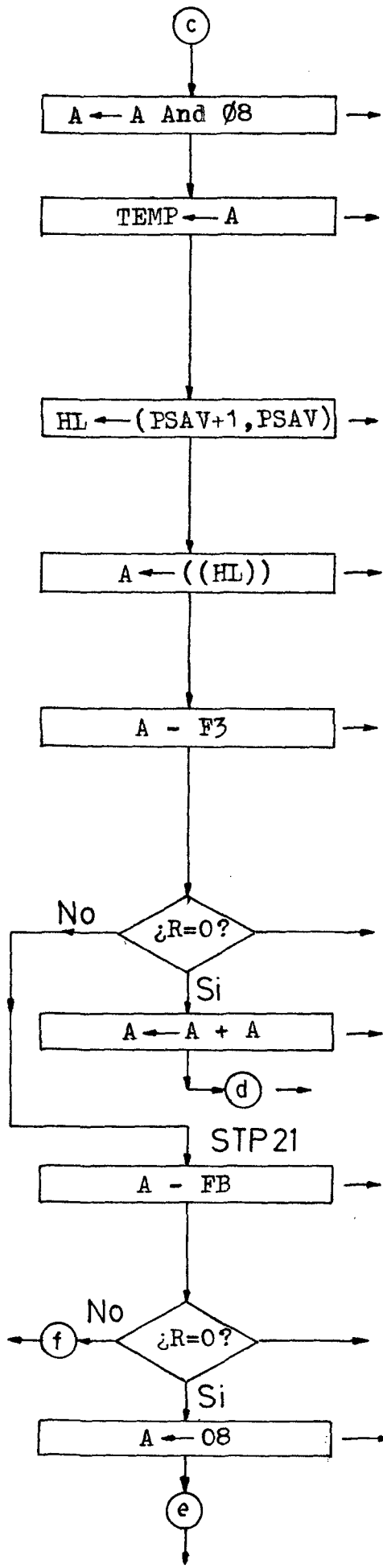
PERIO=10H. Vemos si fue "EXEC" la última tecla pulsada.

¿Se pulsó "EXEC"?

Se pulsó "NEXT".

Limpia el display y termina el comando. (3).

ISAV=20F1. Posición de memoria destinada a salvar las máscaras de interrupciones del usuario. Cogemos el estado de estas máscaras de interrupciones.



Nos quedamos con el estado de las interrupciones.(6).

TEMP=20FD. Es una posición de memoria de uso temporal por la rutina S.STEP. Guardamos el estado de las interrupciones del usuario.

PSAV+1 y PSAV son las posiciones de memoria donde tenemos guardado el CP del usuario.

Ponemos en el Ac. el contenido de la dirección del CP (el código de una instrucción).

F3 es el código de la instrucción DI (Disable Interrupts). Comparamos el contenido del Ac. para ver si se trata de ésta instrucción.

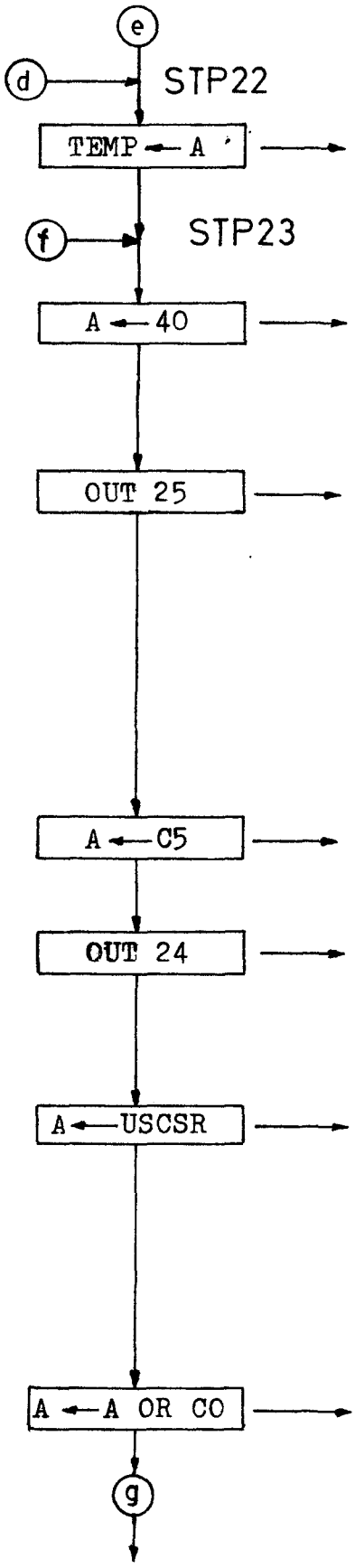
¿ Es DI la instrucción a la que apunta el CP del usuario?.

Reseteamos el estado de las interrupciones del usuario (porque tiene la inst. DI en el programa).

FB es el código de la instrucción EI (Enable Interrupts). Comprobamos si se trata de ésta inst.

¿ El CP del usuario apunta a la instrucción EI ?.

Habilitamos las interrupciones (ponemos a 1 el bit cuatro del Ac.



para indicar que las interrupciones están habilitadas.

Salvamos en 20FD (TEMP) el nuevo estado de las interrupciones del usuario.

Ponemos en el Ac. los bits de mayor orden del valor del Timer (TMODE). (7).

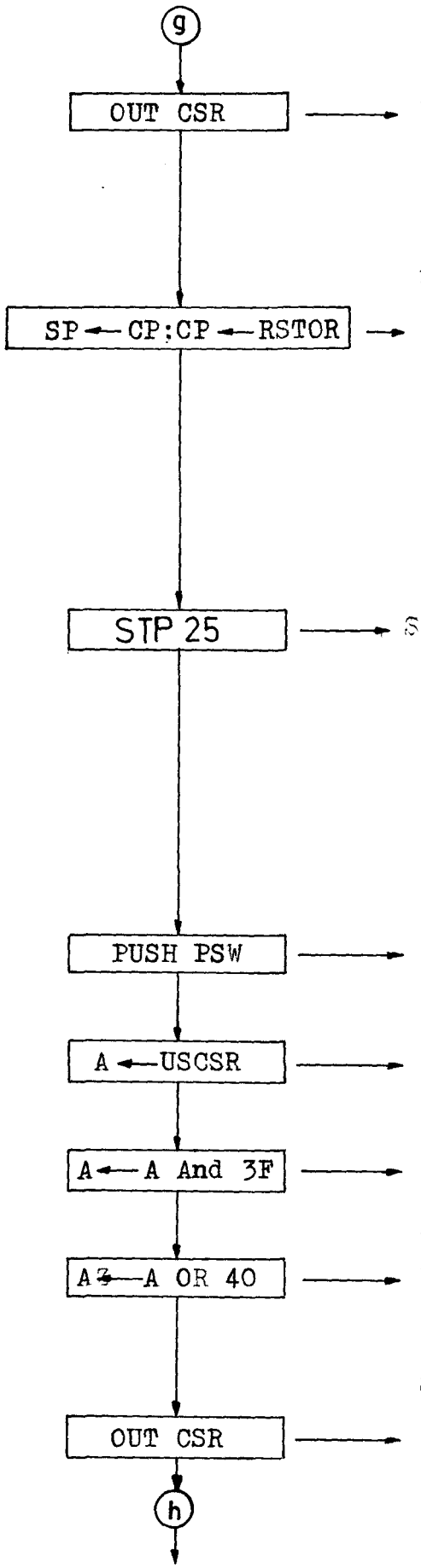
25= Puerto de salida para los bits de mayor orden del valor del timer La instrucción OUT coloca el contenido del Ac. en el bus de datos (bi-direccional) para que se trasmitan por dicho puerto. (8).

Son los bits de menor orden del valor del Timer. (7).

24= Puerto de salida para los bits de menor orden del valor del Timer. (8).

USCSR=20FF. Dirección de memoria donde el usuario puede poner el valor que desee para el registro C/S (comandos/estados). Ponemos dicho valor en el Ac. (9).

Ponemos los dos bits de mayor orden del Ac. a 1 para poner en marcha el Timer. (10).



Ponemos el valor del registro C/S en el bus de datos para indicarle al 8155 que ponga en marcha el Timer.

Restauramos los registros del usuario y realizamos una instrucción.

Se salta aquí cuando (después de que en la onda programada para el Timer se produzca un cambio de ciclo) se halla terminado de realizar una instrucción del usuario.(11).

Salvamos el contenido del Ac. y de la palabra de estados.

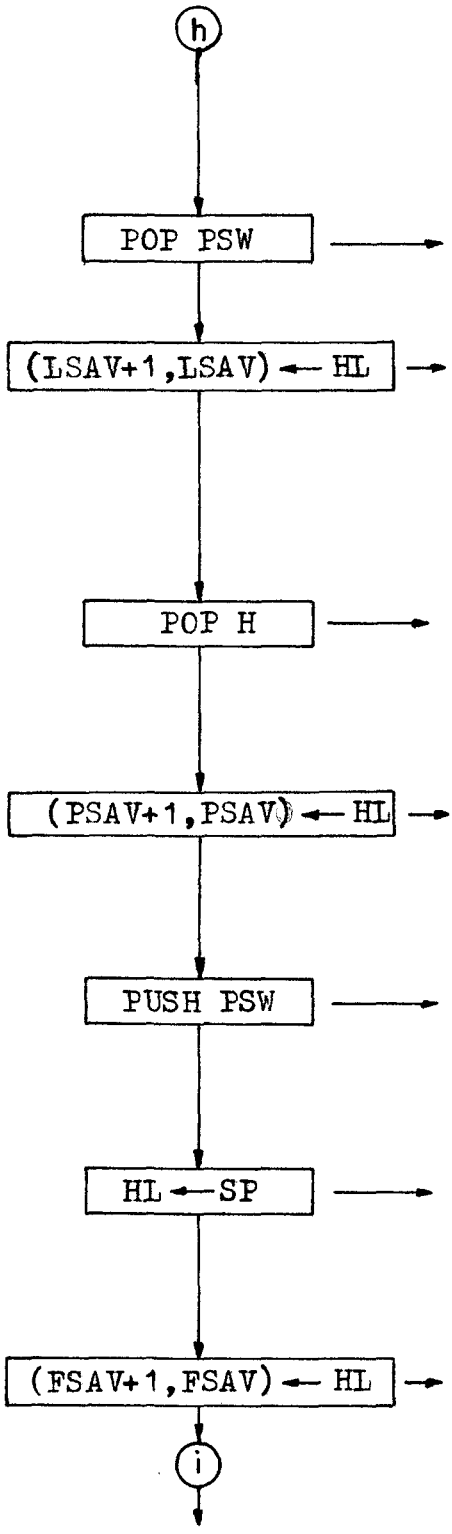
Cogemos el valor que tiene el usuario en el registro C/S.

Ponemos a cero los dos bits de mayor orden del Ac.

Ponemos en el Ac.: 0 1 x x x x x x

Estos dos bits le indican al chip 8155 que para el Timer

Se para el Timer. (tenemos que pararlo porque si no cada vez que se porduzca un cambio de ciclo



en la señal generada por el Timer se produce la interrupción TRAP.)

Con la instrucción señalada le enviamos al 8155 el valor de C/S para que pare el Timer.

Restauramos el valor del acumulador y de la palabra de estados. (11).

LSAV=20EF. LSAV+1=20F0. Posiciones usadas para salvar el contenido de los registros H (20F0) y L (20EF). Salvamos el contenido de HL.

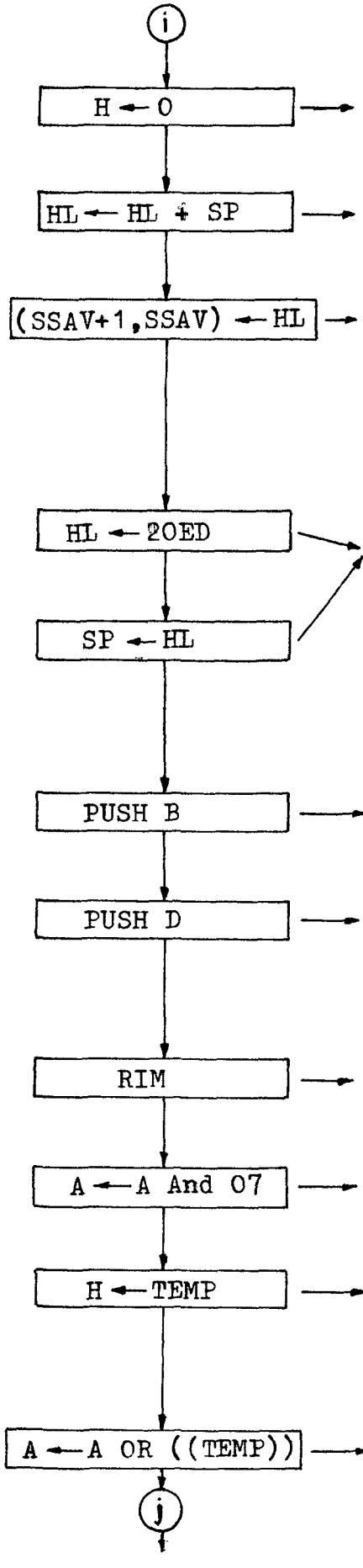
Cogemos el CP del usuario de la cima del Stack. La función RSTOR carga el valor del CP en el Stack.

Salvamos el valor del CP del usuario en las posiciones de memoria destinadas para ello (20F2 y 20F3).

Volvemos a poner en el Stack el valor de la palabra de estados y del acumulador.

Ponemos en L el contenido de la dirección señalada en el SP y en H el contenido de la dirección siguiente.

FSAV+1=20EE. FSAV=20ED.Direcciones de memoria pertenecientes al SP del monitor donde se guardan el contenido del acumulador y de la palabra de estados, respectivamente.



→ Limpiamos el valor de H y de L.

→ Ponemos en HL el contenido del SP del usuario.

→ SSAV+1=20F5. SSAV=20F4.Direcciones de memoria pertenecientes al Stack del monitor, donde se guarda el contenido del SP del usuario.

→ Ponemos en el Stack del monitor la dirección a partir de la cual se empezarán a salvar los contenidos de los registros (ver MNSTK, principio del SP del monitor, pág 48 (anexo 2)).

→ Salvamos los registros B y C.

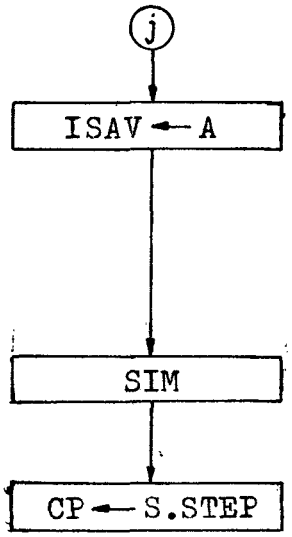
→ Salvamos D y E. El valor de estos registros se restauran despues en la rutina RSTOR.

→ Cogemos las máscaras de interrupciones del usuario.

→ Guardamos sólo los bits de las máscaras. (los otros se anulan).

→ Ponemos en HL la dirección temporal 20FD. (para coger el estado de las interrupciones del usuario).

→ Realizamos una operación OR entre el estado de las máscaras y el estado de las interrupciones.



Salvamos el valor anterior en 20F1 (dirección usada para las máscaras de interrupciones) (éste valor lo cogeremos en STP 20).

Enmascaramos las interrupciones para usar el monitor.(12).

Vamos al principio para realizar otra instrucción del usuario.

.....

(1) Ver DISPC, (pág 123).

(2) Ver RDKBD, (pág 145).

(3) Ver CLDIS, (pág 111).

(4) Ver CLEAR, (pág 109).

(5) Ver GTHEX, (pág).

(6) En la posición ISAV (20F1) tenemos guardado el estado de las interrupciones (indicado por el bit 3 del acumulador) y el estado de las máscaras de interrupciones (indicado por los bits 0, 1, y 2 del acumulador). Al realizar la operación:

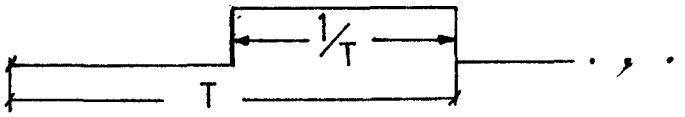
A ← A And 08

Sólo nos queda en el acumulador el valor que tiene el bit número 3 : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

que es el que nos interesa para saber si las interesa para saber si las interrupciones están habilitadas o no.

(7) El 8155 posee un Timer (reloj) programable, en tiempo y modo, es decir se puede programar el periódo

y la forma de onda. En este caso lo que queremos es generar una onda cuadrada:



de tal forma que el tiempo 1/T sea mayor que el menor tiempo que tarda el 8085 en realizar una instrucción.

Si nos fijamos en los esquemas del SDK 85, la salida T/OUT del 8155 esta unida a la entrada TRAP del 8085 (entrada activa a nivel alto). Es decir, que cuando a la entrada de la patilla TRAP existe un nivel alto se produce esta interrupción, la cual lleva el programa monitor a la posición 0175 (STP 25). Lo que hace el comando S.STEP es poner en marcha el Timer del 8155 y llevar el programa monitor a la rutina RSTOR. Luego, la función RSTOR salta a ejecutar el programa del usuario, pero antes de que término de ejecutar una instrucción se produce el cambio de ciclo en la onda del Timer, es decir, que se produce la interrupción TRAP, la CPU termina de realizar la instrucción en curso y luego atiende a la interrupción TRAP.Esta interrupción lleva el monitor a STP25 donde se para el Timer (si no se para se produciría una interrupción TRAP cada vez que la onda del Timer este a nivel alto) y se salta de nuevo a S.STEP para volver a realizar otra instrucción.Por ésto se dice que el comando S.STEP va ejecutando el programa del usuario paso a paso (instrucción a instrucción).

En este caso el valor que se le da al Timer es:

4	0	C	5	= Ø 1 Ø Ø/Ø Ø Ø Ø/1 1 Ø Ø/Ø 1 Ø 1

estos dos bits le indican al 8155 la forma de onda que tiene que generar, en este caso es:

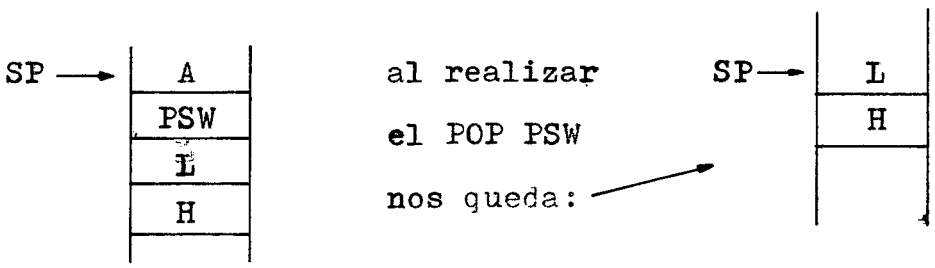
El resto de los bits indican la longitud de la cuenta (el periodo de la onda). Este valor se calcula teniendo en cuenta el menor tiempo que tarda el 8085 en realizar cualquier instrucción de su repertorio. (Ver características del 8155, pág. 48).

(8) El valor del Timer lo ponemos en el bus de datos para que le llegue al 8155 a través de los puertos 24 y 25.

(9) Ver característica del 8155, (pág. 48).

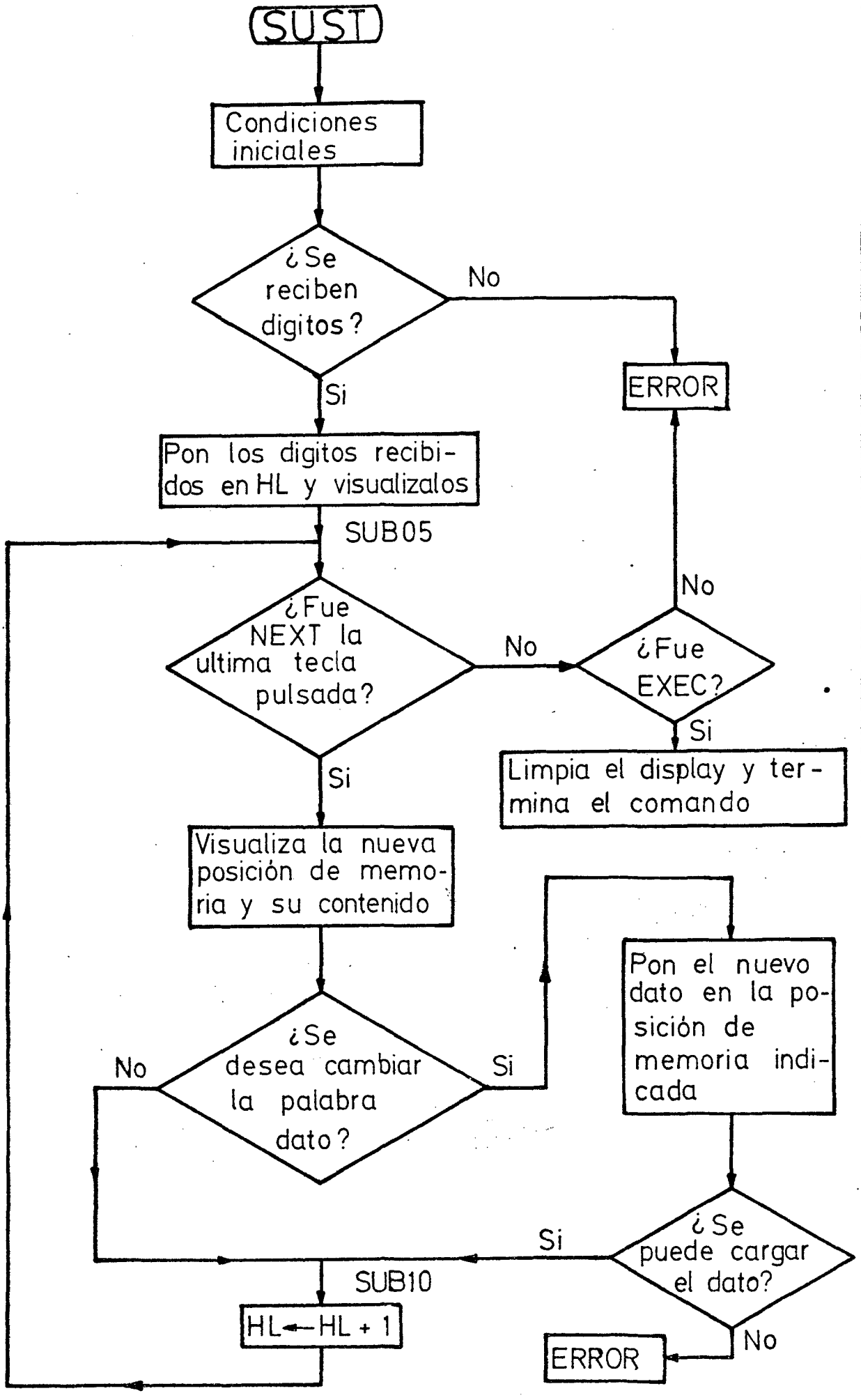
(10) Si los bits de mayor orden del registro de comandos están a uno el 8155 pone en marcha el Timer (ver pág.).

(11) Esta instrucción la hacemos porque queremos sacar del SP el valor de los registros HL, que están después de los del acumulador y de la palabra de estados:



(12) Al poner en el acumulador: Ø Ø Ø Ø/1 1 1 Ø sólo enmascaramos las interrupciones RST 7.5 y RST 6.5. La RST 5.5 no la enmascaramos porque es la que utiliza la rutina RDKBD para leer el teclado.

.....



Comando: SUBST.

" Sustituir datos en memoria !

Con éste comando podemos leer el contenido de la memoria ROM y examinar y modificar el contenido de la RAM.

Las direcciones pueden constar de uno a cuatro dígitos (si se introducen números de más de cuatro cifras sólo son aceptados los cuatro últimos). Al pulsar la tecla "NEXT" después de introducir la dirección de memoria deseada, aparece el contenido de dicha posición de memoria en el campo de datos. Dicho dato se puede cambiar introduciendo nuevos dígitos hexadecimales (el cambio no se produce hasta que no sea pulsada de nuevo la tecla "NEXT" ó la "EXEC".

Cada vez que se pulsa "NEXT" aparece en el display la próxima posición de memoria con su contenido.

Si en lugar del "NEXT" se pulsa "EXEC" se carga el nuevo dato, y se termina el comando.

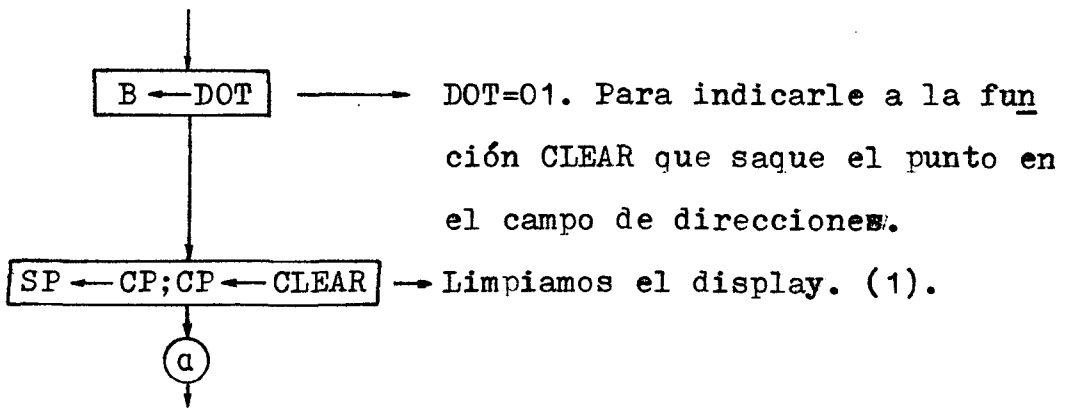
Entradas: ninguna.

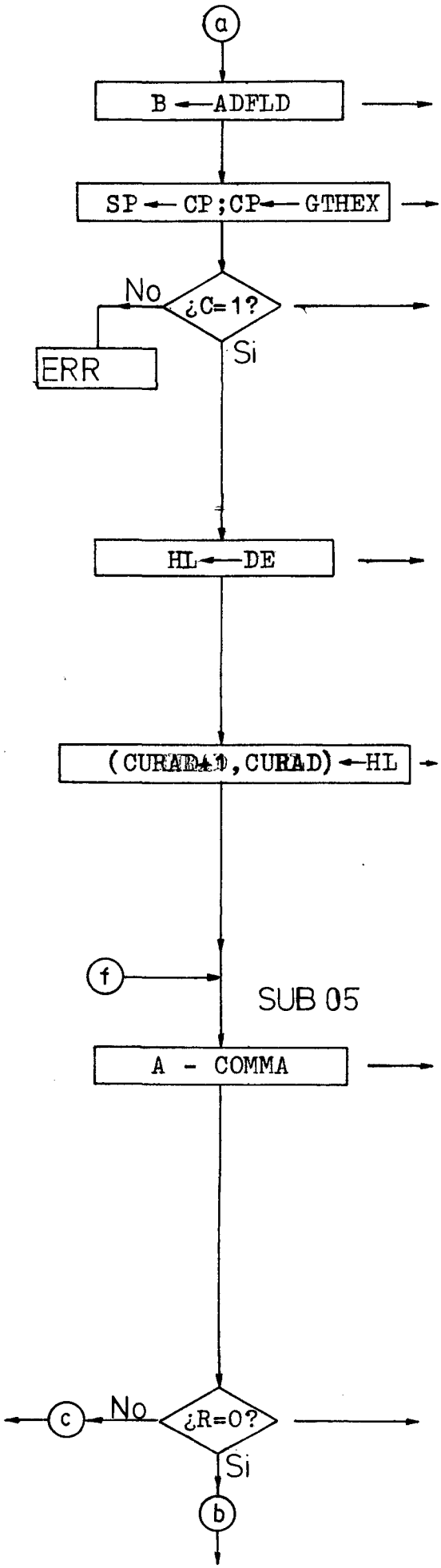
Salidas: ninguna.

Llamadas: CLEAR, GTHEX, UPDDT, ERR, UPDAD.

(Ver el organigrama del comando).

(Ver el comando en lenguaje ensamblador, pág 9).(Anexo 2)





ADFLD=00.Para indicar que se va a usar el campo de direcciones.

Vemos si se reciben dígitos hexadecimales. (2).

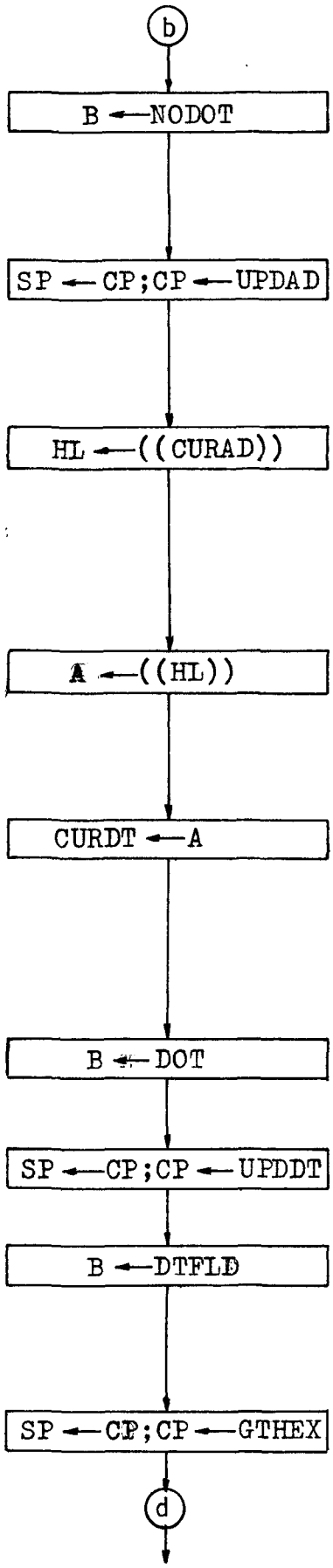
¿Se reciben dígitos? Para averiguarlo, usamos el carry porque la función GTHEX lo pone a 1 cuando recibe algún dígito y a 0 cuando no recibe.

La dirección de memoria recibida (la función GTHEX coloca los dígitos recibidos en DE) la ponemos en HL.

CURAD=20F6. CURAD+1=20F7. Posiciones de memoria usadas para colocar los datos que se van a sacar al display (al campo de direcciones). El contenido de H lo ponemos en 20F7 y el de L en 20F6.

COMMA=11H. En el Ac. tenemos el último carácter recibido desde el teclado.Hacemos esta comparación para ver si fue "NEXT" el último carácter (cuando se pulsa ésta tecla el código que se genera es 11H.)

¿La última tecla pulsada fue NEXT?



→ NODOT=00. Para indicarle a UPDAD que no saque el punto en el campo de direcciones.

→ Sacamos el dato al display (la dirección que teníamos en 20F6 y en 20F7). (3).

→ El contenido de las posiciones 20F6 y 20F7 lo ponemos en HL, es decir, la dirección que anteriormente se sacó al display.

→ Llevamos al Ac. el contenido de la dirección señalada (para poder sacarlo al campo de datos).

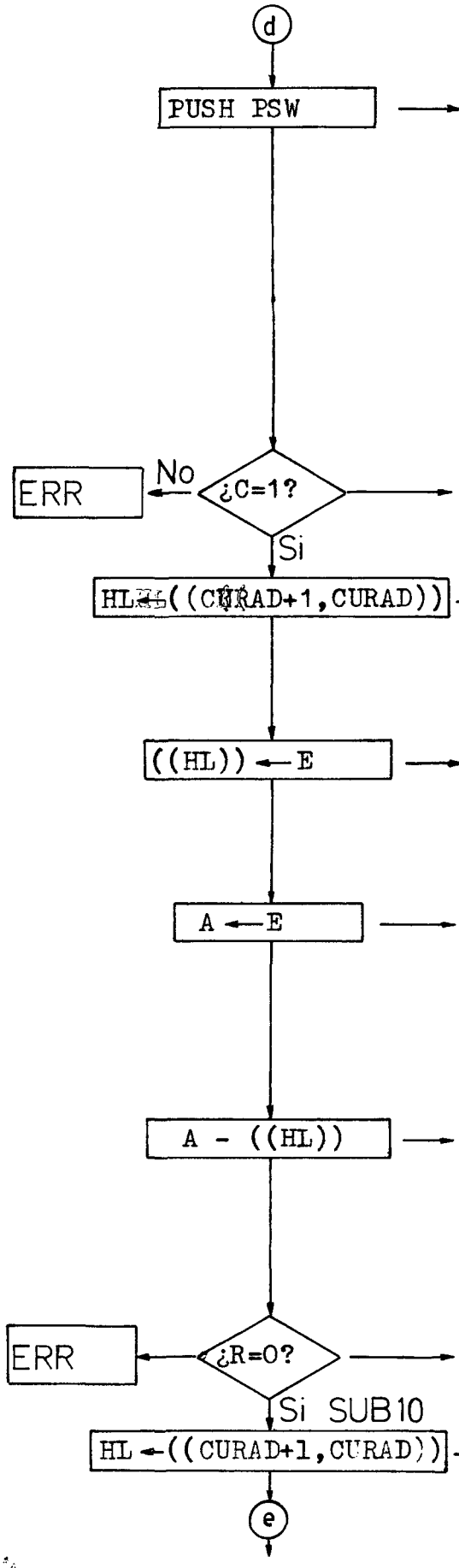
→ CURDT=20F8. Posición usada para colocar el dato que se va a sacar al campo de datos del display por medio de UPDDT.

→ Para indicarle a UPDDT que saque el punto en el display.

→ Sacamos el dato al display. (4).

→ DTFLD=01. Para indicarle a GTHEX que se va a usar el campo de datos.

→ Para ver si se desea cambiar el contenido de la dirección de memoria señalada. (2).



Salvamos en el Stack el contenido del Ac. y de la palabra de estados. (Porque los valores que poseen en este momento, se necesitarán más tarde). Con las operaciones siguientes estos valores se destruyen.

¿ Se desea cambiar el contenido de los registros?

Ponemos en HL la posición de memoria señalada (a la que deseamos cambiarle su contenido).

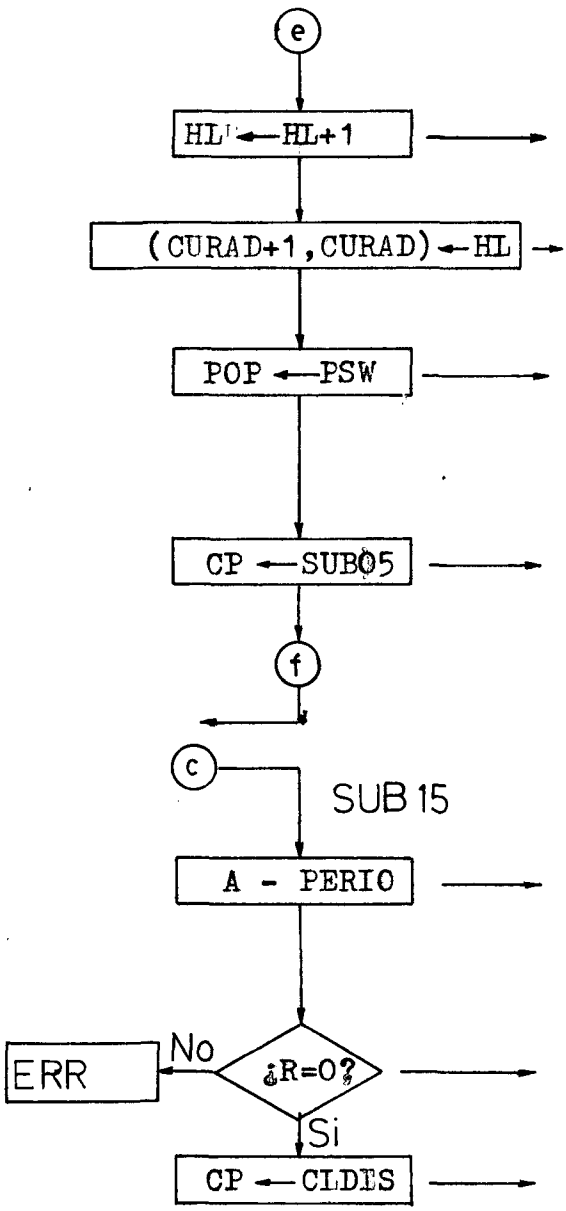
El contenido de E lo llevamos a la dirección indicada (cargamos el nuevo dato).

El nuevo contenido lo llevamos al Ac. para comprobar si el dato se cargó correctamente en la dirección indicada.

Si el resultado de esta comparación es ≠ de cero nos indica que el nuevo dato no se cargó correctamente en la posición indicada.

¿ Se cargó el dato ?

Ponemos en HL el contenido de 20F7 y de 20F6. Para incremen-



tar la dirección que nos señala el display

Incrementamos la dirección.

Cargamos la nueva dirección en CURAD+1 (20F7) y en CURAD (20F6)

Restauramos el último carácter (el contenido del acumulador) y la palabra de estados.

Saltamos para hacer lo mismo con la nueva posición de memoria.

PERIO= 10H. Con esta comparación averiguamos si la última tecla pulsada fue "EXEC".

¿fue "EXEC"?

Limpia el display y termina el comando. (5).

.....

(1) Ver CLEAR, (pág.109).

(2) Ver GTHEX, (pág.119).

(3) Ver UPDAD, (pág.149).

(4) Ver UPDDT, (pág.150).

(5) Ver CLDIS, (pág.111).

.....

Comando: VECTOR INTERRUP.

Esta tecla es similar a la "GO" en el sentido de que ambas llevan el control lejos del monitor.

Al pulsar ésta tecla se reconoce inmediatamente la interrupción RST 7.5, que hace que el control pase a la posición 3C del programa monitor. Esta posición contiene un salto incondicional a la posición 20D4 de la memoria RAM del usuario. En dicha posición (hasta la 20D6) podemos colocar la instrucción que deseemos.

Para que la tecla V.I. funcione se tienen que cumplir dos condiciones:

1.- Las interrupciones han de estar habilitadas, (Ejecución de la instrucción EI).

2.- Poner a cero el flip-flop de RST 7.5 (ésto lo hacemos con la ejecución de una instrucción SIM).

Es decir, para hacer funcionar a V.I. tenemos:

a) Poner en 20D4-20D6 la instrucción que deseamos (por ejemplo, un salto a una posición determinada de memoria).

b) Realizar las siguientes instrucciones (vamos a suponer que empezamos en la posición de memoria 2000 y que en la posición 2020, por ejemplo, tenemos un programa que queremos que se realice sólo cuando se pulse la tecla V.I.) ponemos:

2000	31C820	LXI SP,20C8; Definimos el Stack.
2003	3E18	MVI A,18; Ponemos a 1 los bits del acumulador necesarios para habilitar V.I.

2005	30	SIM; Reseteamos el flip-flop de RST 7.5 .
2006	FB	EI; Habilitamos las interrupciones.
2007	F5	PUSH PSW; Salvamos en el Stack el contenido del A y de los flags.
2008	C30020	JUMP 2000; Con ésto establecemos un lazo, hasta que se pulse la tecla V.I., en éste caso el monitor salta a 20D4 y realiza la instrucción que esté en 20D4,D5, y D6.

Por ejemplo, podemos poner:

20D4	C32020	JUMP 2020; Al pulsar V.I. saltamos a la posición 2020 que es la posición a partir de la cual tenemos (p.e.) un programa a ejecutar.

FUNCIONES (RUTINAS) DEL PROGRAMA.

En éste apartado se analizan las distintas rutinas que posee el programa monitor del SDK'85.

Estas rutinas son las siguientes (se van analizando por éste orden):

.- CLEAR: Limpia el display.

.- CLDIS: Limpia el display y termina el comando.

.- ERR: Visualiza el mensaje error.

.- HXDSP: Expande dígitos hex. por el display.

.- ININT: Tratamiento de las interrupciones de entrada.

.-GTHEX: Coge dígitos Hex.

.- DISPC: Visualiza el Contador de Programa.

.- RETT: Retorna a verdadero.

.- RETF: Retorna a falso.

.- RGLOC: Coge la posición de memoria donde se salvarán los registros.

.- RGNAM: Visualiza el nombre del registro.

.- RSTOR: Restaura los registros del usuario.

.- SETREG: Convierte el registro señalado en registro puntero.

.- NXTRG: Avanza el registro puntero al próximo registro.

.- OUTPUT: Saca carácteres al display.

.- RDKBD: Lee el teclado.

.- INSDG: Inserta dígitos hex.

.- UPDAD: Saca el dato al campo de direcciones del display.

.-UPDDT: Saca el dato al campo de datos del display.

.- CLDST: Arranque en frio.

.....

Función: CLEAR.

" Limpia el display".

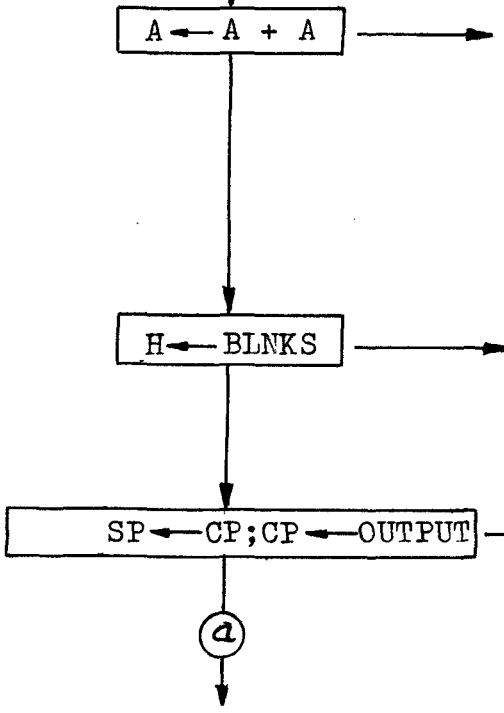
Esta rutina envia carácteres en blanco a ambos campos del display (al de direcciones y al de datos). Si el flag del punto está puesto aparece un punto en la parte derecha del campo de direcciones.

Entradas: en el registro "B" ponemos el flag del punto. Si en "B" tenemos un 1, indica que la función CLEAR saca el punto en el campo de direcciones. Si tenemos un cero, no saca el punto.

Salidas: ninguna.

Llamadas: OUTPUT.

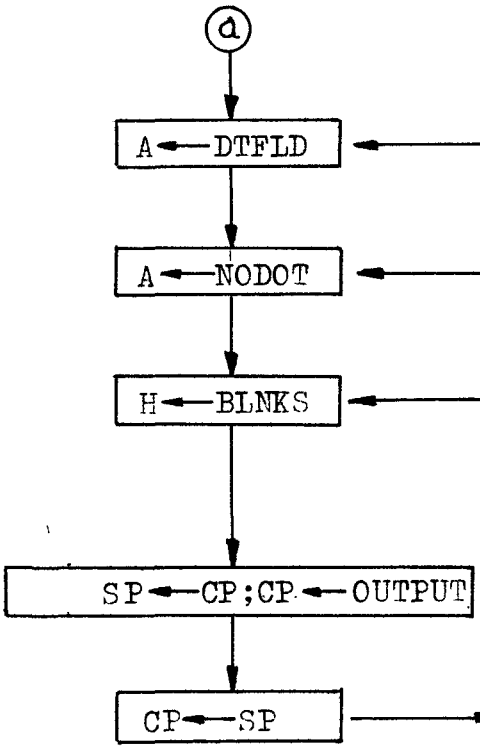
(Ver la función en lenguaje ensamblador, pág 10) (Anexo2)



+ = OR EXCLUSIVA.Despues de realizar ésta operación el contenido del acumulador es 00. Esto lo hacemos para usar el campo de direcciones. (1).

BLNKS=039A. Espacio de memoria donde están los espacios en blanco. (Ver pág.22 del anexo 2).

Limpiamos el campo de direcciones. (1).



DTFLD=01. Indicamos que se va a usar el campo de datos. (1).

NODOT=00. Ponemos en el registro "B" el flag del punto.

Ponemos en los registros HL, la dirección de memoria cuyo contenido son los espacios en blanco.

Limpiamos el campo de datos. (1).

Regresa. Saltamos a la parte del programa donde se produjo la interrupción.

.....

(1) Ver OUTPUT, (pág 141) [illegible].

.....

Función: CLDIS.

" Limpia el display y termina el comando".

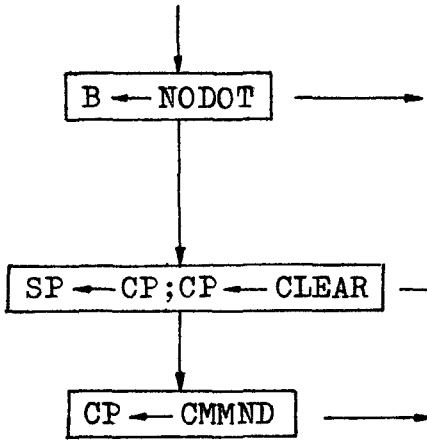
A ésta función salta algun comando (EXAM, SUBST, S.STEP,...) que desee terminar normalmente . CLDIS limpia el display y se ramifica para reconocer otro comando.

Entradas: ninguna.

Salidas: ninguna.

Llamadas: CLEAR.

(Ver la función en lenguaje ensamblador, pág 11). (Anexo 2)



NODOT=00. Para indicarle a CLEAR que no saque el punto en el campo de direcciones.

Limpiamos los dos campos del display. (1).

Salta para recibir otro comando. (2).

.....

(1) Ver CLEAR, (pág109).

(2) Ver CMMND, (pág 73).

.....

Función: ERR.

" Visualiza el mensaje error".

A "ERR" salta algún comando que desee terminar porque se ha producido algún error en el teclado.

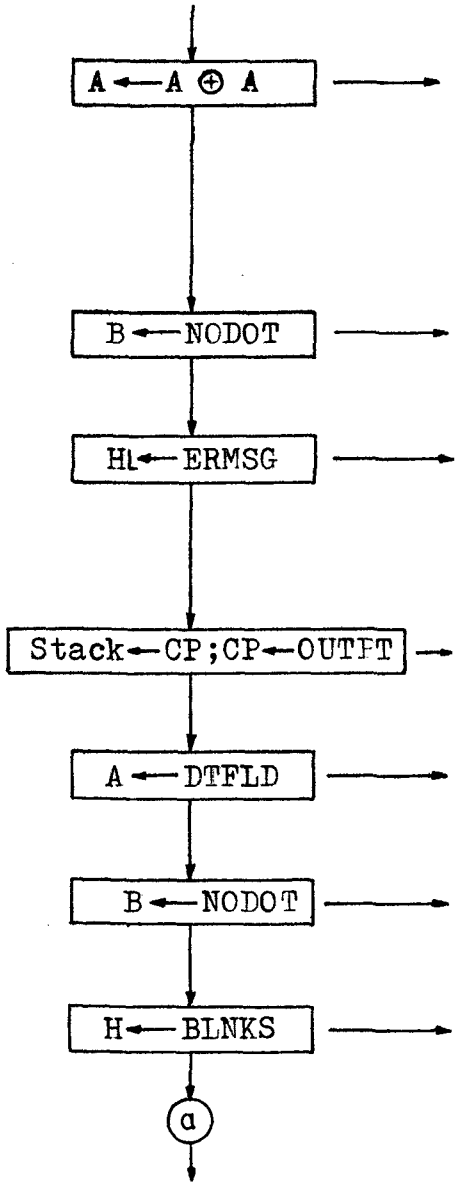
La función saca el mensaje "-ERR" al display se ramifica para reconocer otro comando.

Entradas: ninguna.

Salidas: ninguna.

Llamadas: OUTPT.

(Ver la función en lenguaje ensamblador, pág 12).(Anexo 2)



⊕== OR Exclusiva.Con ésto ponemos el acumulador a cero, para indicar que se va a usar el campo de direcciones.(1).

Para indicarle a OUTPT que no saque el punto al display. (1).

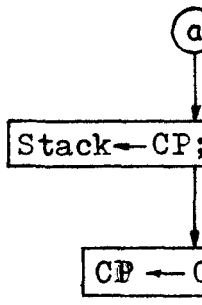
ERMSG=039E.Espacio de memoria donde tenemos escrito el mensaje "ERR" (ver pág 22).(2).(Anexo 2)

Sacamos el mensaje al display. (1).

DTFLD=01.Indicamos que se va a usar el campo de datos. (1).

NODOT=00. Para que OUTPT no saque el punto al display. (1).

BLNKS=039A. Espacio de memoria donde están los espacios en blanco (ver pág 22). (2).(Anexo 2)



Sacamos los espacios en blanco al campo de datos. (1)

Salto para reconocer otro comando. (3).

.....

(1) Ver OUTPT, (pág 141).

(2) Ponemos la primera dirección de los carácteres que se desean sacar al display en HL porque OUTPT sólo saca al display el contenido de ésa dirección y de la siguiente, cuando se trata del campo de datos, ó de las tres siguientes, cuando se trata de campo de direcciones.

(3) Ver CMMND, (pág 73).

.....

Función: HXDSP.

" Expande los dígitos hexadecimales para mostrarlos en el display".

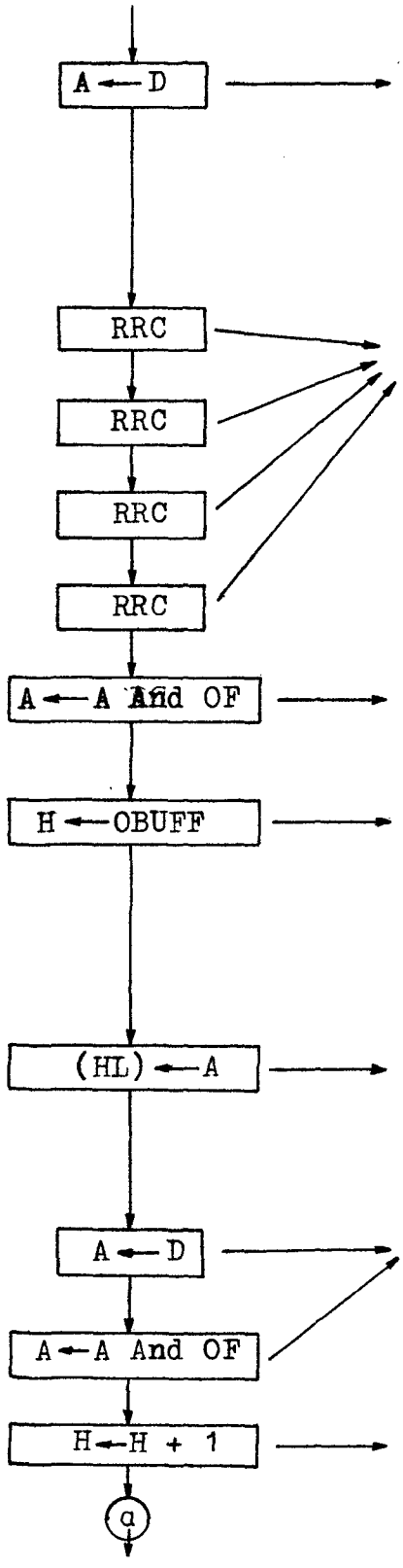
Esta función expande cada dígito hex. recibido en la forma adecuada para poder mostrarlos en el display mediante la rutina OUTPT. Cada dígito hex. recibido es colocado en los cuatro bits de más bajo orden de las posiciones de memoria usadas para éste fin (OBUFF = buffer de salida), los bits de mayor orden son puestos a cero. La función coloca en HL la dirección del buffer de salida.

Entradas: en DE tenemos los cuatro dígitos hex. que vamos a expandir.

Salidas: en HL se colocan las direcciones del buffer de salida(los términos que se desean sacar al display están colocados en posiciones sucesivas de memoria).

Llamadas: ninguna.

(Ver la función en lenguaje ensamblador, pág 13).(Anexo2)



Cogemos la primera palabra dato y la ponemos en el acumulador, (los dos primeros términos que se van a sacar al display.)

Cogemos los cuatro bits de mayor orden y los convertimos en un carácter simple. (1).

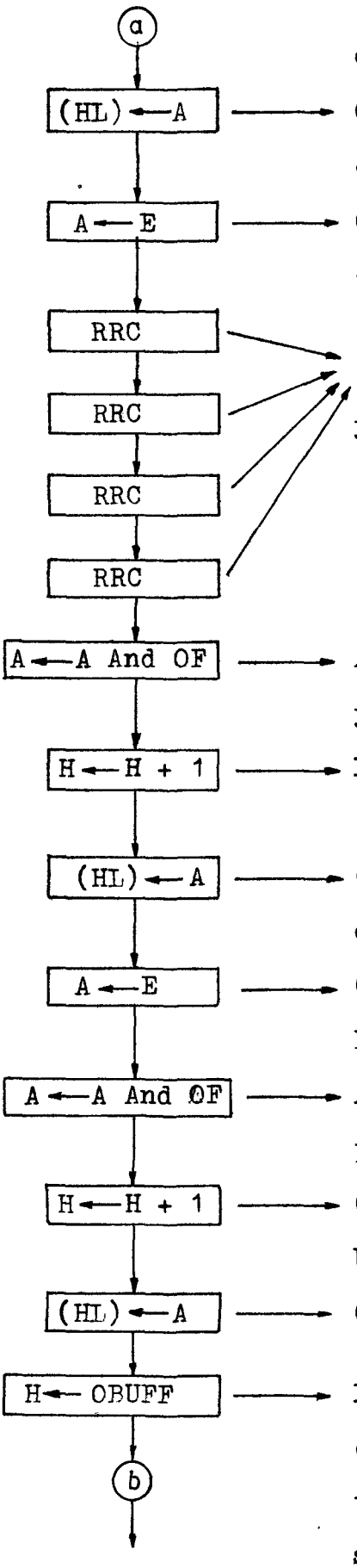
Los bits de mayor orden los anulamos.

OBUFF=20F9.Primera dirección usada para el buffer de salida. Tiene asignadas 4 posiciones de memoria(20F9,20FA,20FB,20FC.)

Lo que está en el Ac. lo ponemos en la primera posición del buffer de salida(20F9).

Cogemos otra vez la primera palabra dato, ahora se anulan los bits de mayor orden. (3).

Ponemos en HL la próxima posi-



ción del buffer (20FA).

Cargamos el contenido del Ac. en dicha posición.

Cogemos la segunda palabra dato. (4)

Cogemos los cuatro bits de mayor orden.

Anulamos los cuatro bits de mayor orden del acumulador.

La próxima posición del buffer. (20FB).

Cargamos el siguiente dato en dicha posición.

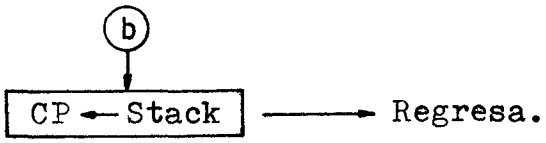
Cogemos de nuevo la segunda palabra dato.

Anulamos el otro término de la palabra dato.

Cogemos la próxima posición del buffer (20FC).

Cargamos el término deseado.

Ponemos en HL la primera posición del buffer.(para que los términos que están en éstas posiciones puedan ser sacados al display por OUTPT.)



.....

(1) Con RRC rotamos el contenido del acumulador un bit hacia la derecha a través del carry. Por ejemplo, si el contenido del acumulador es [4|2] al realizar la instrucción (RRC) cuatro veces nos queda: [2|4].

(2) En éste caso anularíamos el dos.

(3) Ahora anulamos el cuatro.

(4) A partir de aquí volvemos a realizar las mismas instrucciones con la segunda palabra dato. Esta se encuentra en el registro E (la primera estaba en el D).

.....

Función: ININT.

" Tratamiento de las interrupciones de entrada"

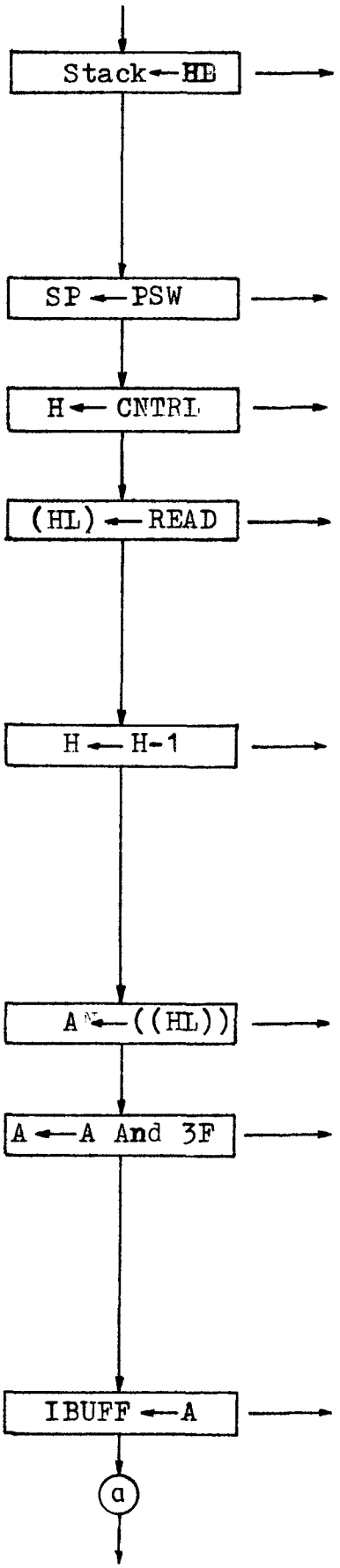
Se salta a ésta función cuando se produce la interrupción RST 5.5 . Cuando la rutina RDKBD (leer tecla do) está esperando un carácter, y el usuario ha pulsado una tecla (excepto "reset" ó "vector interrupt")se produce la interrupción RST 5.5 ; ésta salta el programa a la función ININT y ésta carga el carácter de entrada en el buffer de entrada y devuelve el control a la rutina RDKBD.

Entradas: ninguna.

Salidas: ninguna.

Llamadas: ninguna.

(Ver la función en lenguaje ensamblador, pág 14).(Anexo 2)



Salvamos el contenido de HL, porque vamos a trabajar con éstos reg. y destruiríamos su contenido.

Salvamos el contenido del acumulador y de la palabra de estados.

CNTRL=1900. Dirección usada para direccionar al chip 8279.

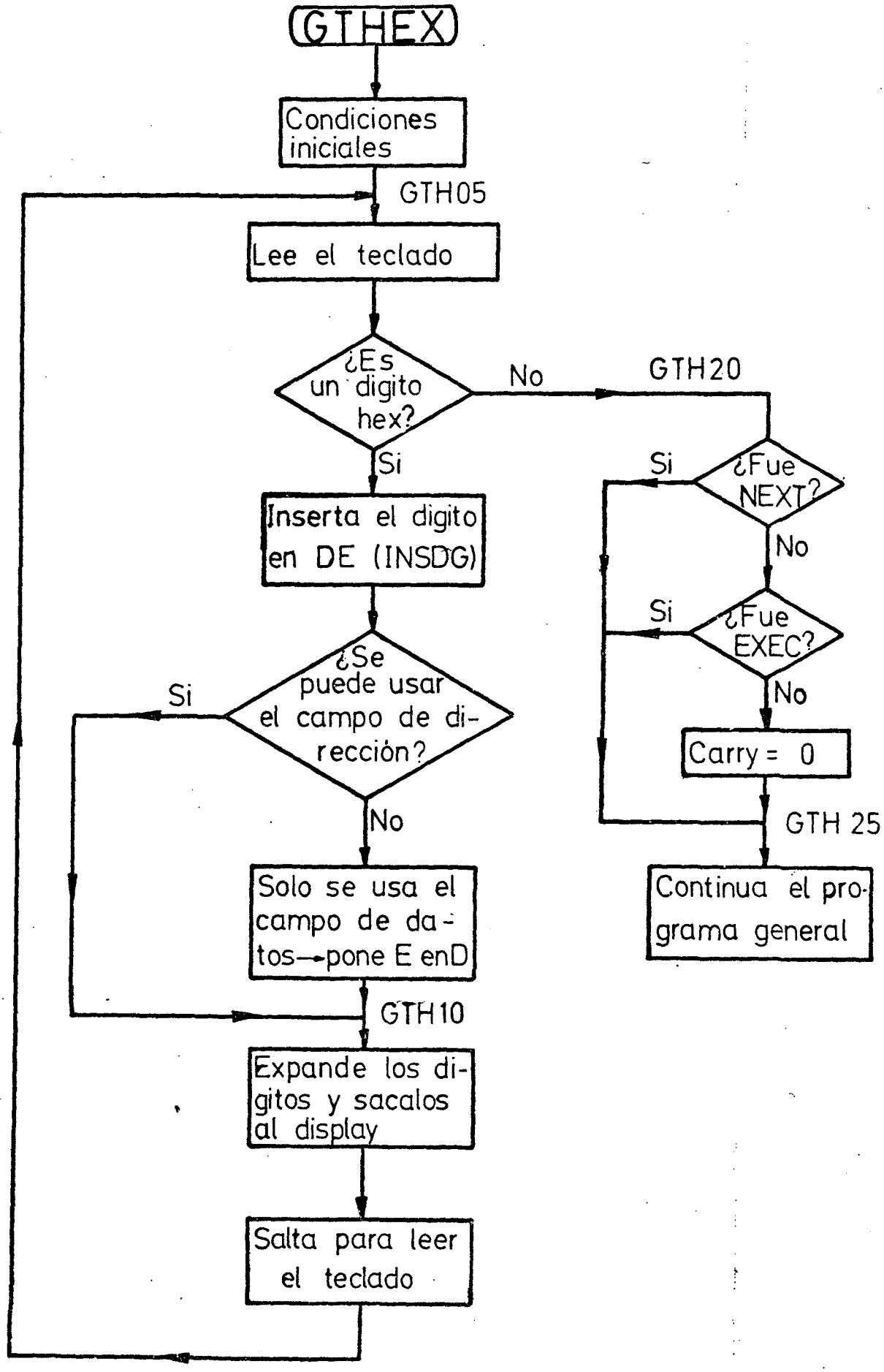
READ=40. Este dato lo ponemos en la dirección 1900 para indicarle al 8279 el modo de trabajo (ver chip 8279, pág 10).

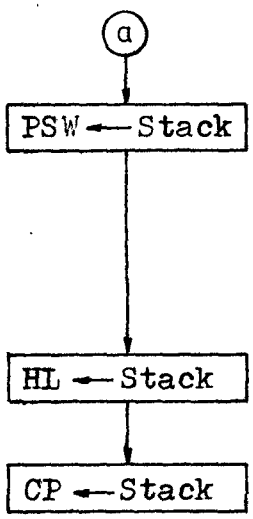
En HL nos queda la dirección 1800, posición de memoria destinada a poner el carácter de entrada. El 8279 se encarga de cargar el carácter leido en ésta posición.

Ponemos el carácter en el acumulador.

Ponemos a cero los dos bits de mayor orden del Ac. para indicar que el buffer de entrada está lleno y comprobar si el carácter leído es válido. (1).

IBUFF=20FE. Dirección de memoria destinada al buffer de entrada (cargamos el carácter en el buffer).





Restauramos el contenido de la palabra de estados y del acumulador (el que tenían antes de entrar en ININT.)

Restauramos el valor de HL.

Regresa.

.....

(1) Ver RDKBD, (pág 145).

.....

Función: GTHEX.

" Entrada de dígitos hexadecimales."

Esta función acepta una fila de dígitos hexadecimales desde el teclado, mostrándolos en el display según son recibidos.

Si se reciben más de cuatro dígitos, solamente son usados los cuatro últimos. Si el "flag display" está puesto,los dos últimos dígitos son mostrados en el campo de datos; si no lo está, los cuatro últimos dígitos son mostrados en el campo de direcciones del display. La función visualiza un punto en la parte derecha del campo de direcciones. Si la última tecla que recibe la función no es "NEXT" ó "EXEC", anula los dígitos recibidos anteriormente. La función indica si ha recibido, por lo menos un dígito hex. poniendo el carry a uno; si no recibe ninguno lo pone a cero.

Entradas: en el registro B está el "flag display"

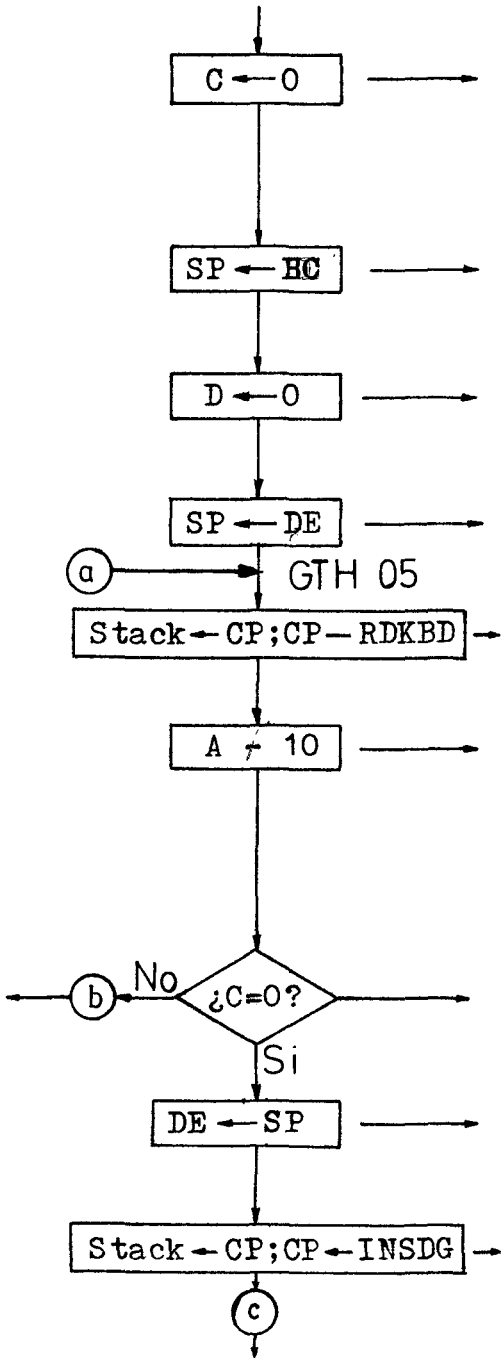
Si es cero: se usa el campo de direcciones.

Si es uno: se usa el campo de datos.

Salidas: A en el acumulador se coloca el último carácter leído desde el teclado.
En los registros DE están los dígitos hex. leídos desde el teclado.
Carry = 1, si por lo menos ha sido leido un dígito. C = 0, si no se recibe ningun dígito.

Llamadas: RDKBD, INSDG, HXSP, OUTPT.

(Ver la función en lenguaje ensamblador, pág 12).(Anexo2)



Reseteamos el flag de dígitos, nos va a indicar si se reciben o no dígitos desde el teclado.

Salvamos el contenido de BC en el Stack.

Anulamos el contenido de los registros DE.

Salvamos los valores hex.

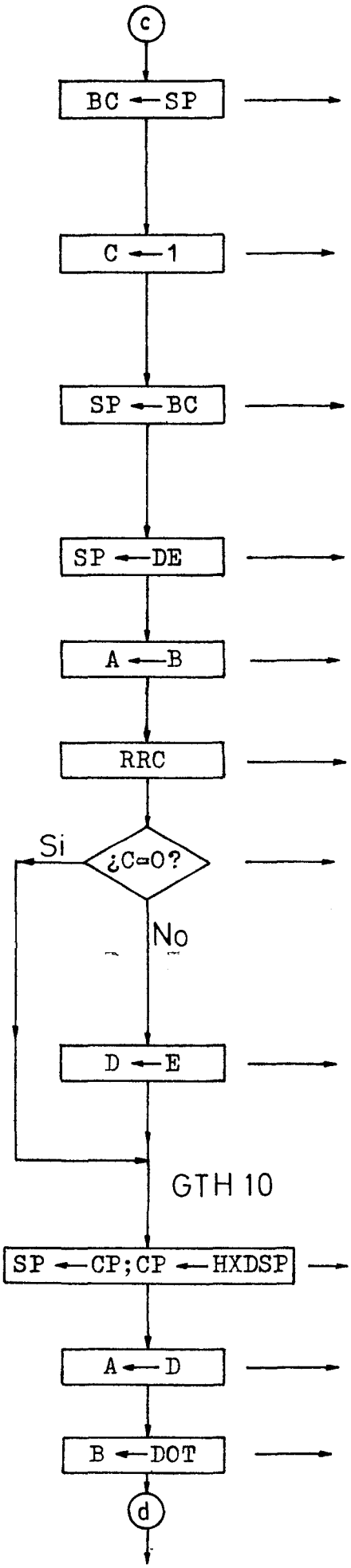
Leemos el teclado. (1).

Comparamos el contenido del acumulador con: Ø Ø Ø 1/Ø Ø Ø Ø . Para distinguir cuando se pulsa un dígito ó un comando.

¿ Es el carácter recibido un dígito hex.?

Restauramos los valores hex. que tenemos en DE.

Insertamos el nuevo dígito en



valor hexadecimal. (2).

Restauramos el valor del "flag display", nos indicará el campo que vamos a usar.

Con ésto indicamos que por lo menos se ha recibido un dígito hex. (es el flag de dígitos hex)

Salvamos el valor de los registros BC (el valor del "flag display" y del "flag de dígitos hex")

Salvamos los valores hex recibidos.

Para ver como está el "flag display".

Rotamos el contenido del Ac.hacia la derecha a través del carry.

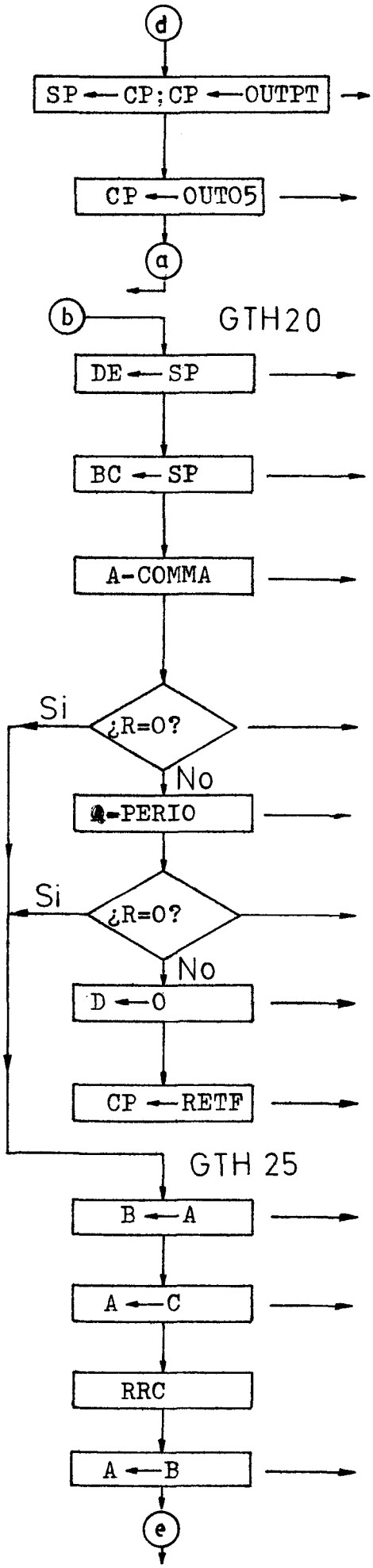
¿ Se va a usar el campo de direcciones? Si el flag estaba puesto, al realizar RRC ponemos C=1; Si no lo estaba ponemos C=0.

Como vamos a usar el campo de datos sólo se usará la palabra menos significativa de los registros DE (la que está en E).

Expandimos los valores hex. para poder visualizarlos. (3).

Cogemos el "flag display". (4).

DOT=01.Para indicarle a OUTPT que saque un punto en el display.



→ Sacamos los valores hex. al display. (4).

→ Cogemos el próximo valor hex.

→ Restauramos los dígitos recibidos

→ Restauramos el "flag display" y el "flag dígitos hex."

→ COMMA=11.Hacemos ésta comparación para saber si la última tecla pulsada fue "NEXT".(5).

→ ¿ Se pulsó "NEXT" ?

→ PERIO=10.Para saber si fue "EXEC" (5).

→ ¿ Se pulsó "EXEC" ?

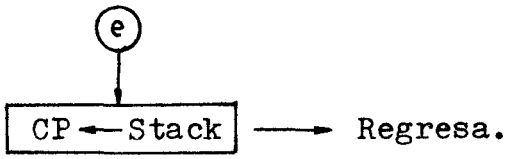
→ Anulamos el último valor recibido.

→ Salta a falso. (6).

→ Salvamos el último carácter.

→ Ponemos el flag de dígitos hex en el carry.

→ Restauramos en el acumulador el último carácter.



.....

(1) Ver RDKBD, (pág145).

(2) Ver INSDG, (pág 147).

(3) Ver HXDSP, (pág 113).

(4) Ver OUTPT, (pág 142).

(5) Cuando pulsamos en el teclado "NEXT", el código que se genera es 11, si se pulsa "EXEC" se genera 10; Por ésto cuando se desea saber si la última tacla pulsada fue alguna de éstas dos, tenemos que comparar el contenido del acumulador con estos códigos, si el resultado es cero , es cierta la comparación.

(6) Ver RETF, (pág 125).

.....

Función: DISPC.

" Visualiza el contador de programa".

Esta función visualiza el contador de programa (CP) del usuario en el campo de direcciones del display, con un punto en la parte derecha del campo.

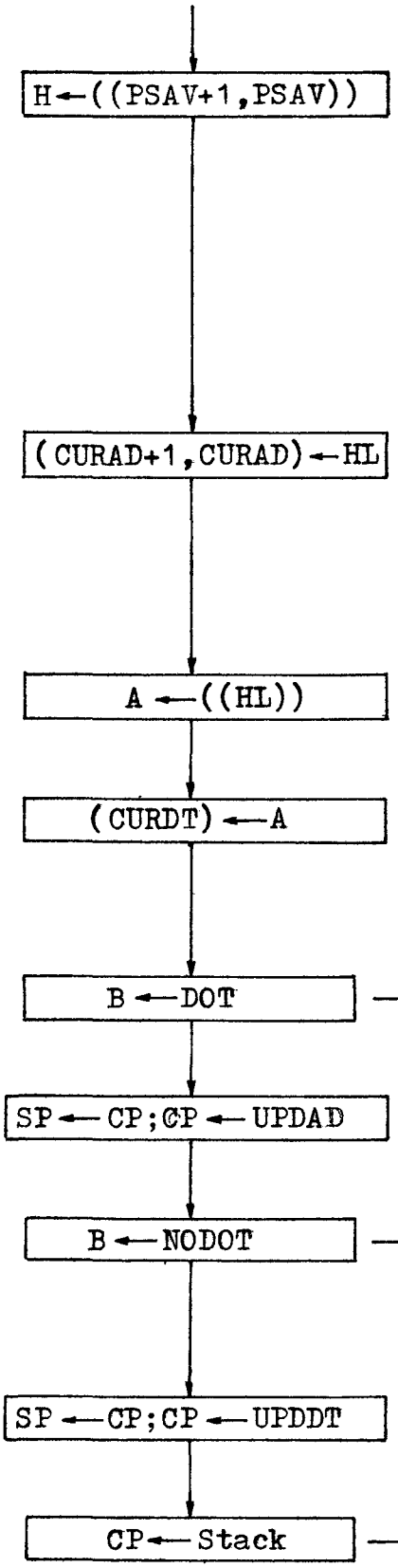
La palabra dato direccionada por el contador de programa es visualizada en el campo de datos del display.

Entradas: ninguna.

Salidas: ninguna.

Llamadas: UPDAD, UPDDT.

(Ver la función en lenguaje ensamblador, pág 11).(Anexo2)



PSAV=20F2;PSAV+1=20F3. Direcciones de memoria destinadas a guardar el contenido del CP. Al realizar ésta instrucción ponemos en H el contenido de 20F3 y en L el de 20F2.

CURAD=20F6 (y 20F7). Son las dos posiciones de memoria donde colocamos el data que se desea sacar al campo de direcciones.(1).

El dato direccionado por el CP lo ponemos en el acumulador.

CURDT=20F8. Posición de memoria usada para poner el dato que se desea sacar al campo de datos.(2)

DOT=01. Para indicarle a UPDAD que saque el punto al display.

Sacamos el contenido del CP deseado al campo de direcciones.(1)

NODOT=00. Para indicarle a UPDDT que no saque el punto al display (en el campo de datos).

Sacamos la instrucción apuntada por el CP al campo de datos. (2).

Regresa.

.....

(1) Ver UPDAD, (pág 149).

(2) Ver UPDDT, (pág 150).

.....

Función: RETF.

"Retorna a falso".

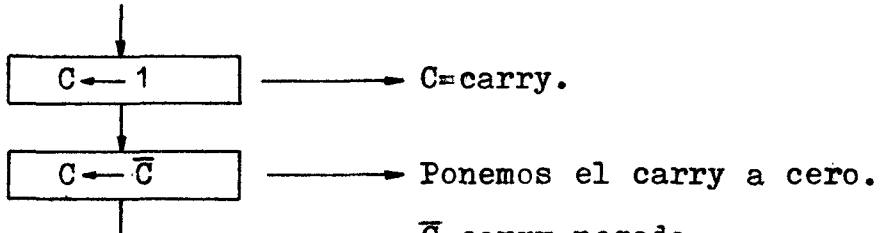
Aésta rutina saltan todas las funciones cuando se comete algún error. RETF resetea el carry (lo pone a cero) y devuelve el control a la rutina que lo invocó.

Entradas: ninguna.

Salidas: carry = 0. (falso)

Llamadas: ninguna.

(Ver la función en lenguaje ensamblador, pág 17)(Anexo 2)



.....

Función: RETT.

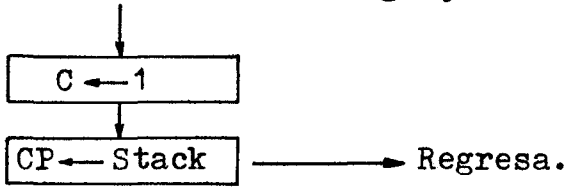
"Retorna a verdadero".

Esta rutina pone el carry a 1 y retorna el control á la función que la invocó.

Entradas: ninguna.

Salidas: carry = 1 (verdadero).

(ver la función en lenguaje ensambldor, pág 17)(Anexo 2)



.....

Función: RGLOC.

"Coge la posición de memoria donde se salvarán los registros."

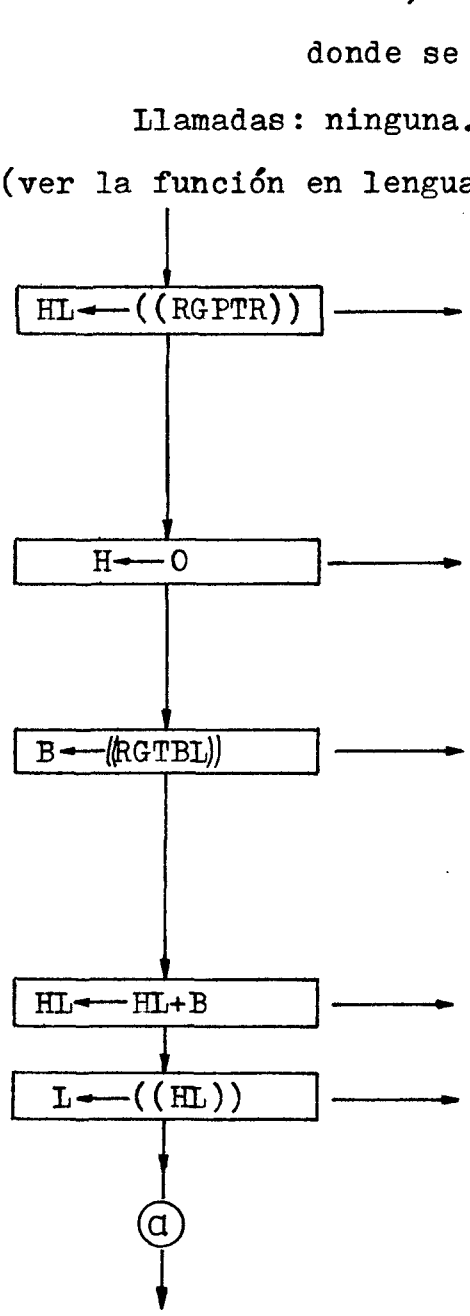
Esta función pone en HL la dirección de memoria donde se va a salvar el contenido del registro que deseamos (el reg. puntero). Usa el valor de dicho registro como punto de entrada en la tabla donde están las distintas posiciones de memoria en las que se salvarán los reg.

Entradas: ninguna.

Salidas: en HL, se pone la dirección de memoria donde se salva el registro.

Llamadas: ninguna.

(ver la función en lenguaje ensamblador, pág 18)(Anexo 2)



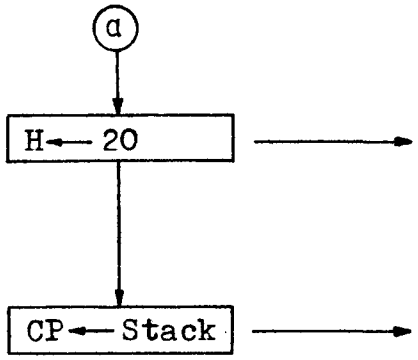
RGPTR=20FF. Posición de memoria donde se guarda el registro puntero. Ponemos el reg. puntero en L.

Anulamos el contenido del registro H (sólo interesa lo que está en L.)

RGTBL=03ED. Dirección de memoria donde tenemos la tabla con las posiciones donde salvaremos a todos los registros.

(1).

El contenido de la posición de memoria que tenemos en HL lo ponemos en el reg. L (es la palabra de menor orden de la dirección buscada.)



.....

(1) Vamos a suponer que el registro puntero es "E" (es decir, el contenido de la posición 20FD es 04, ver tabla RGPTB, pág 48 (Anexo 2)). Al realizar la operación: HL ← HL + B; nos queda en HL la dirección de memoria cuyo contenido es la palabra de menor orden de la posición de memoria usada para salvar el registro "E".

Contenido de BC → Ø Ø Ø Ø/Ø Ø 1 1/1 1 1 1/Ø 1 Ø 1 +

Contenido de HL → Ø Ø Ø Ø/Ø Ø 1 1/1 1 1 1/Ø Ø Ø 1

Si nos fijamos en la tabla RGTBL (pág 24) (Anexo 2) vemos que el contenido de ésta posición de memoria es E9, es decir, que la dirección usada para salvar el contenido del registro "E" es la 20E9 (ver MNSTK, pág 48). (Anexo 2)

.....

Función: RGNAM.

" Visualiza el nombre del registro."

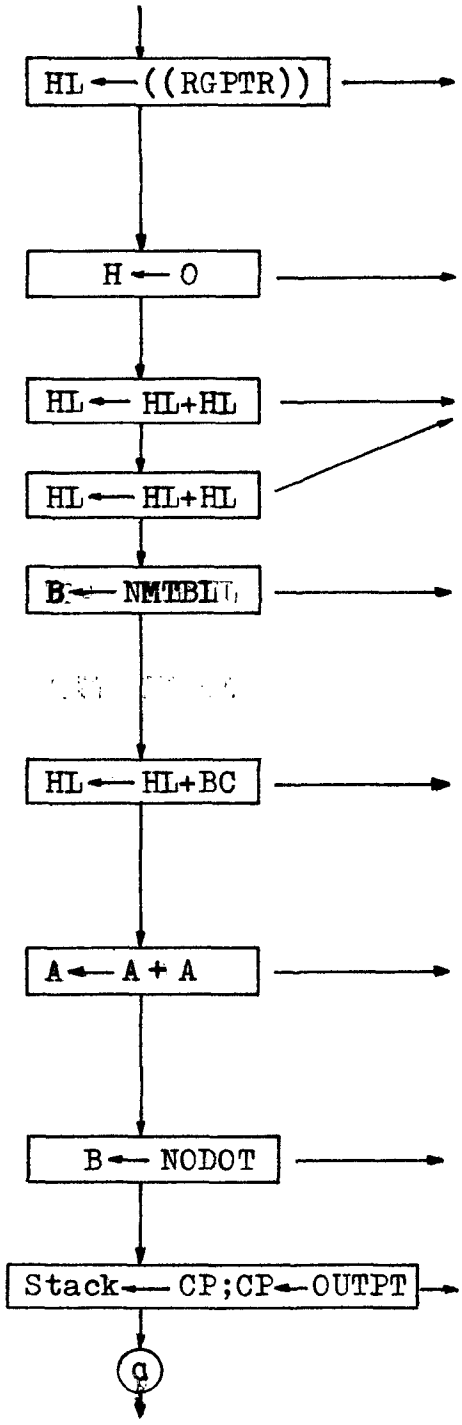
Esta función muestra en el campo de direcciones del display el nombre del registro correspondiente al registro puntero.

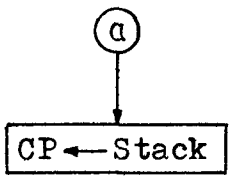
Entradas: ninguna.

Salidas: ninguna.

Llamadas: OUTPT.

(Ver la función en lenguaje ensamblador, pág 13).(Anexo 2)





al display. (3).

Regresa.

.....

(1) ⟶ Vamos a suponer que el registro puntero es el"D" es decir, el contenido de la posición 20FD es 03.

Al realizar la operación: HL←HL + HL ;nos queda:

```
  Ø Ø Ø Ø/Ø Ø 1 1
  Ø Ø Ø Ø/Ø Ø 1 1   +
-----------------
  Ø Ø Ø Ø/Ø 1 1 Ø
      "       "
      0       G
```

Si la realizamos de nuevo, el contenido de HL será:

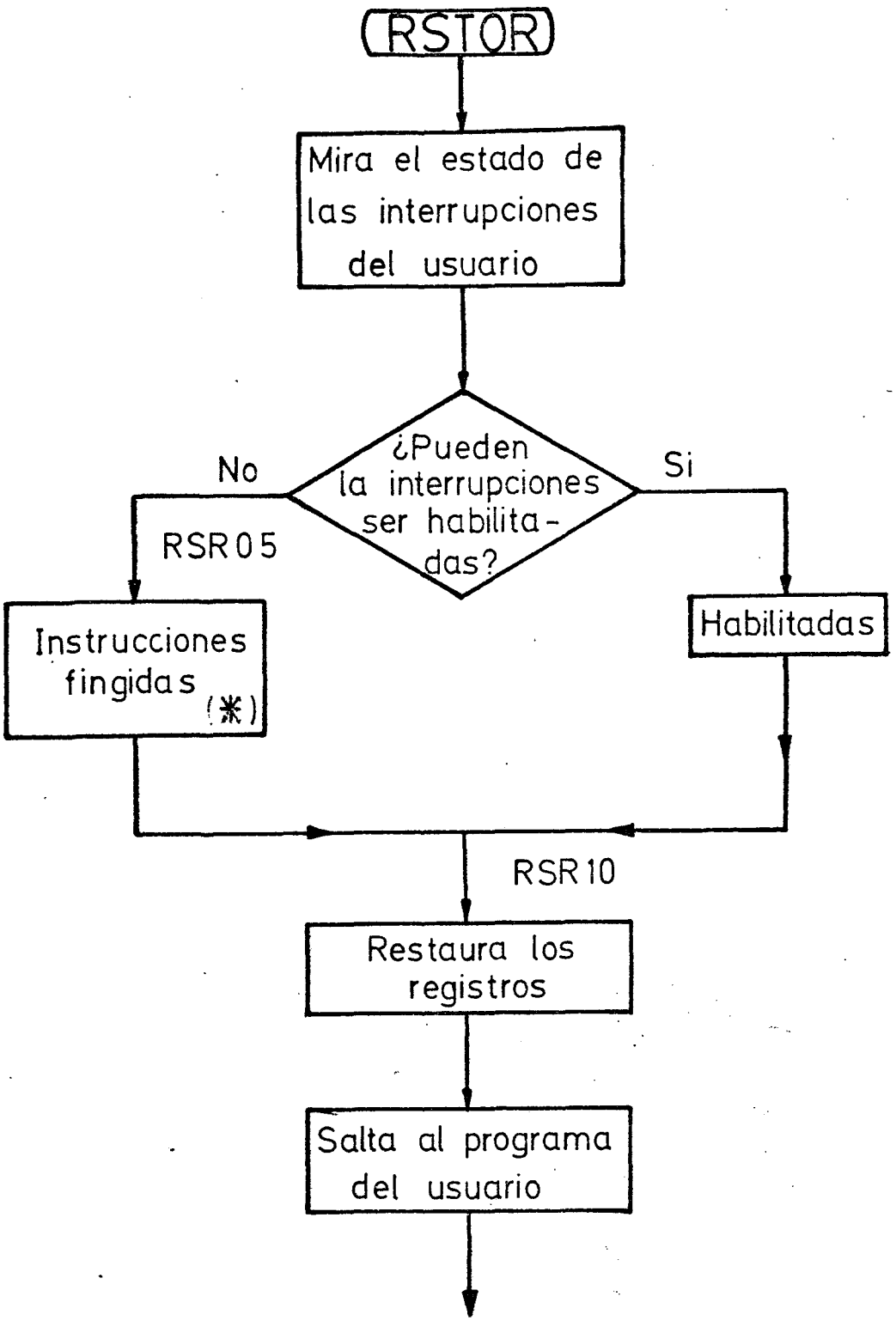
```
  Ø Ø Ø Ø/Ø 1 1 Ø
  Ø Ø Ø Ø/Ø 1 1 Ø   +
-----------------
  Ø Ø Ø Ø 1 1 Ø Ø
      "       "
      0       C
```

Vemos que se ha multiplicado por cuatro el valor inicial del registro puntero. Esto lo hacemos porque la tabla con el nombre de los registros tiene cuatro palabras de entrada, es decir, cada nombre de un registro ocupa cuatro posiciones de memoria.

(2) ⟶ Vamos a realizar la operación: HL←HL + BC , para comprobar que en Hl nos queda la dirección de la tabla NMTBL, donde tenemos grabado el nombre del registro deseado.

```
Cntenido de BC→Ø Ø Ø Ø/Ø Ø 1 1/1 Ø 1 1/1 Ø Ø 1
Contenido de HL ⟶              Ø Ø Ø Ø/1 1 Ø Ø   +
-----------------------------------------------
  Ø Ø Ø Ø/Ø Ø 1 1/1 1 Ø Ø/Ø 1 Ø 1
      "       "       "       "
      0       3       C       5
```

Fijándonos en la tabla vemos que en ésta dirección empiezan las cuatro posiciones destinadas al



* La instrucciones que se realizan por este camino son para conseguir que la función tarde el mismo tiempo por el "No" que por el "Si".

registro "D" (que es el que queriamos sacar al display).

(3) Ver OUTPT (pág 141).

.....

Función: RSTOR.

" Restaura los registros del usuario"

Esta función restaura todos los registros de la CPU, los flip/flops, el estado de las interrupciones, las máscaras de interrupciones, el stack pointer, y el contador de programa, desde sus respectivas posiciones de memoria donde se les salvó.

El tiempo de ésta rutina es crítico para la correcta realización de la rutina S. STEP.

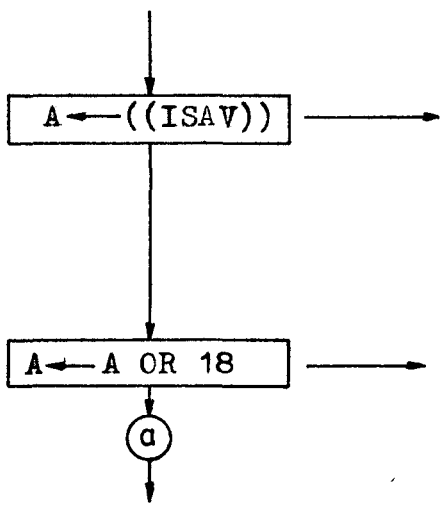
Si por alguna causa se cambia el número de estados que la CPU necesita para ejecutar ésta rutina , entonces el valor del timer (ver función S.STEP, pág 90) debe ser ajustado al nuevo valor.

Entradas: ninguna.

Salidas: ninguna.

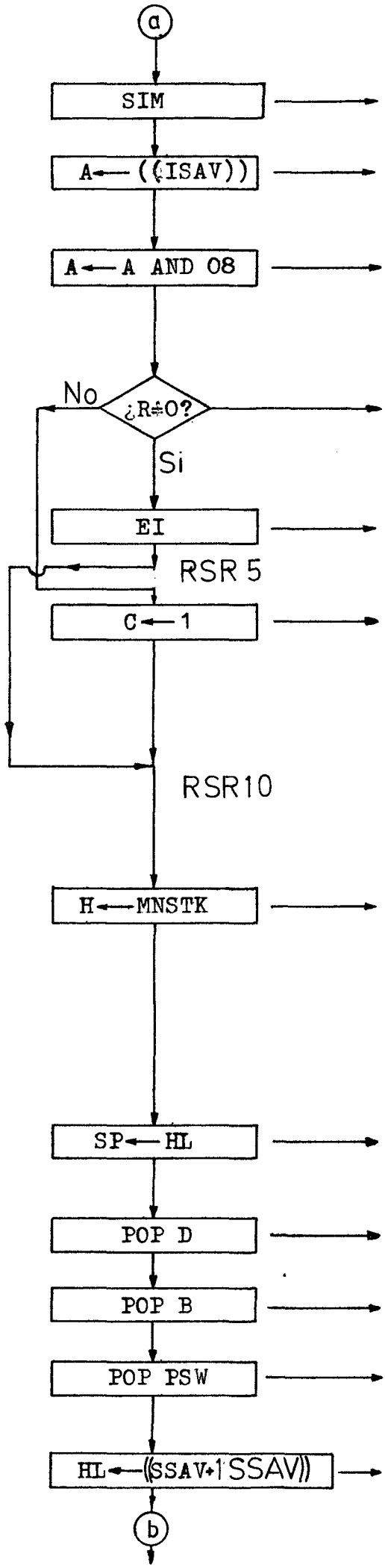
Llamadas: ninguna.

(Ver la función en lenguaje ensamblador, pág 19).(Anexo2)



ISAV=20F1. Posición de memoria usada para guardar el estado de las máscaras de interrupciones.

Habilitamos la puesta de las máscaras de interrupción y reseteamos el flip/flop de RST 7.5.(1).



(1).

Cogemos de nuevo el estada de las interrupciones de usuario.

Con ésta operación comprobamos si las interrupciones se pueden habilitar. (2).

¿Se pueden habilitar?

Las habilitamos (ver instrucción EI, pág 32)

Esta es una instrucción fingida, es decir, se pone sólo para que RSTOR tarde el mismo tiempo por los dos caminos (el del SI y el del NO) (Esto para cuando se está ejecutando el comando S.STEP).

MNSTK= 20E9. Esta dirección nos indica el principio del Stack del monitor (zona de memoria destinada a salvar el contenido de los registros del usuario.)

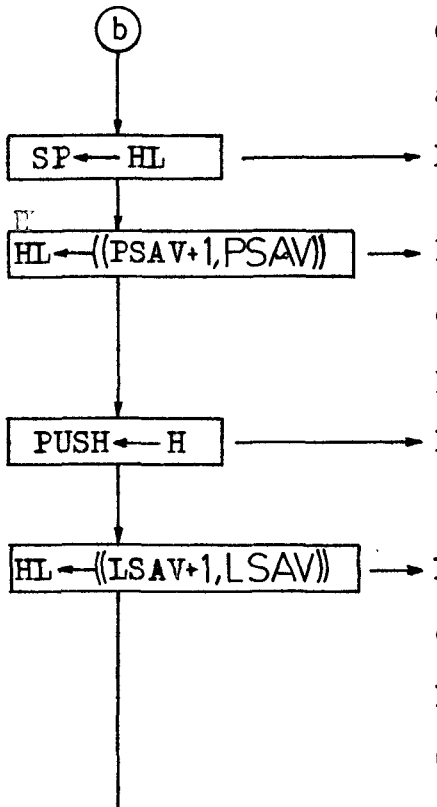
La dirección que tenemos en HL la ponemos en el stack.(SP).

Restauramos el par de reg. DE. (3)

Restauramos BC. (3).

Restauramos la palabra de estados.

SSAV=20F4(y20F5). Posiciones



donde guardamos el SP del usuario.

Restauramos el SP.

PSAV=20F2 (y 20F3) Posiciones donde guardamos el contador de programa (CP) del usuario.

Ponemos el contador de programa en el stack. (4).

LSAV=20EF (y 20F0). Posiciones domde guardamos el contenido de los reg. L y H respectivamente. Con ésta instrucción restauramos los reg. HL.

Regresa.

.....

(1) Ver instrucción SIM (pág 32).

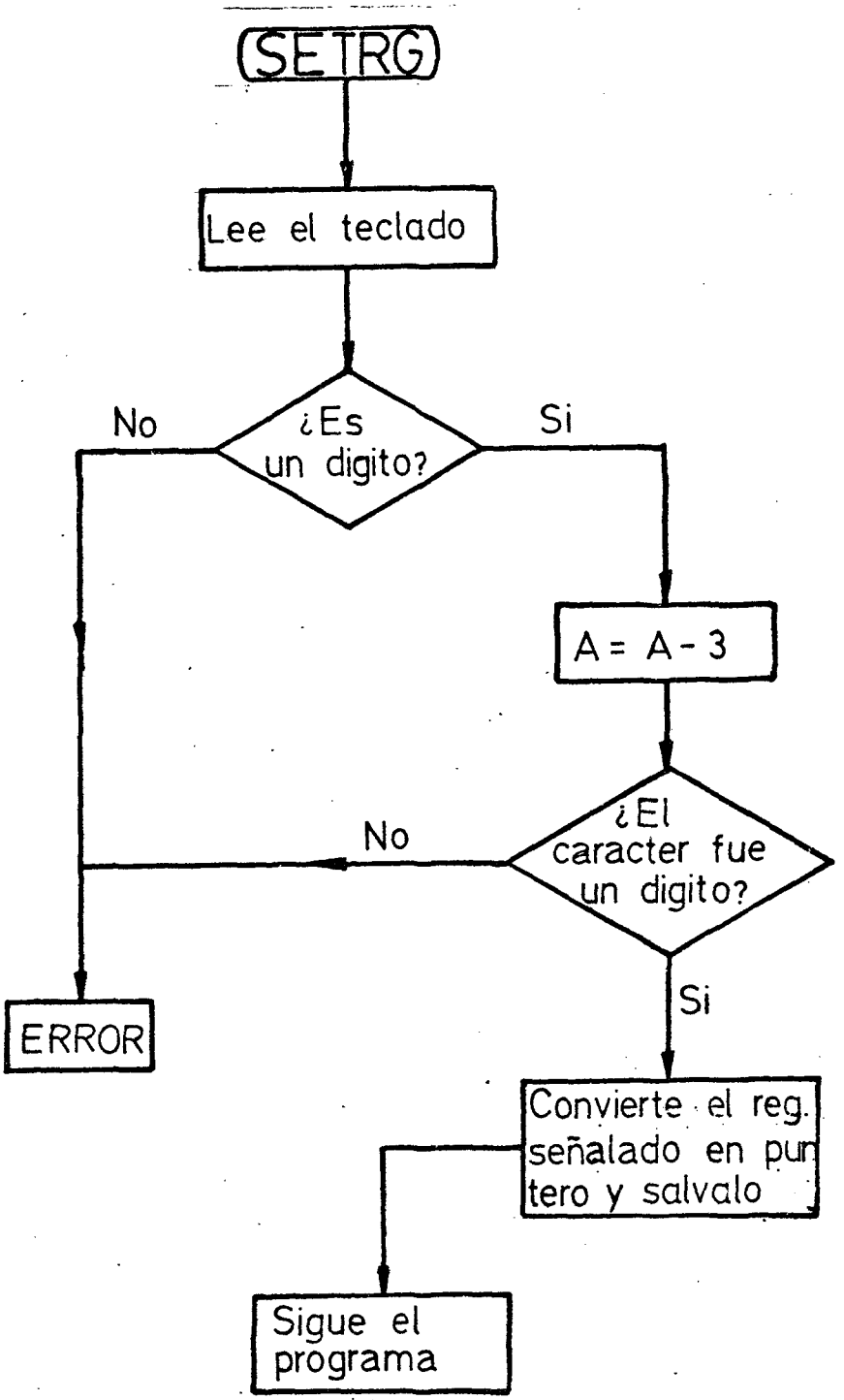
(2) Al realizar la operación AND, cuya tabla de verdad es:

E	E	S
0	0	0
0	1	0
1	0	0
1	1	1

entre el acumulador y 08, nos quedamos sólo con el estado del tercer bit del Ac., que es el que nos interesa para saber si habilitamos las interrupciones o no.

A → | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Si el estado de éste bit es = 1 al realizar la instrucción EI, se habilitan las interrupciones. Si es = 0, no se habilitan.



(3) Ver instrucción POP (pág 32).

(4) Ver instrucción PUSH (pág 32).

.....

Función: SETREG.

"Convierte el registro señalado en reg. puntero"

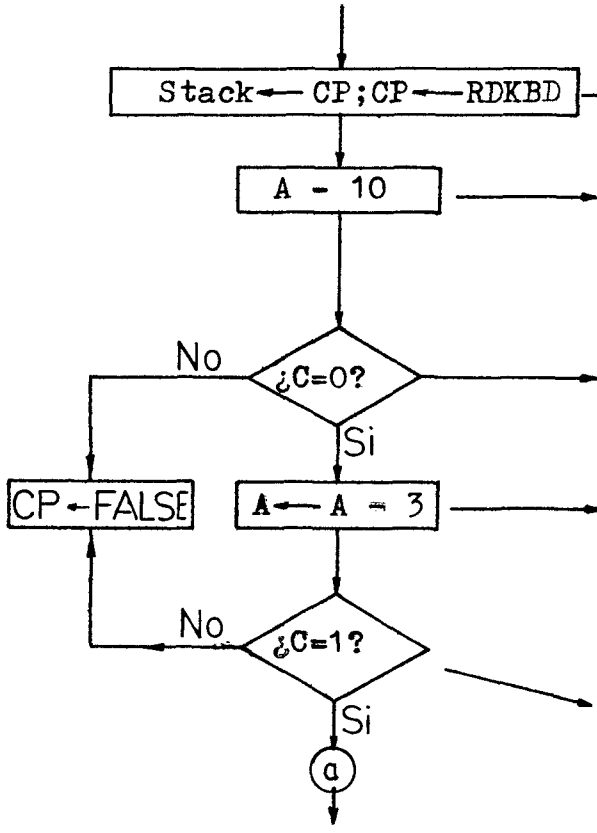
Esta función lee un carácter desde el teclado,si el carácter leído, es un registro, lo convierte en el correspondiente valor de registro puntero. Este valor es salvado (en la posición de memoria destinada para éste fin) y la función salta a TRUE, si no (si el carácter leídono es un registro) la función retorna a FALSE.

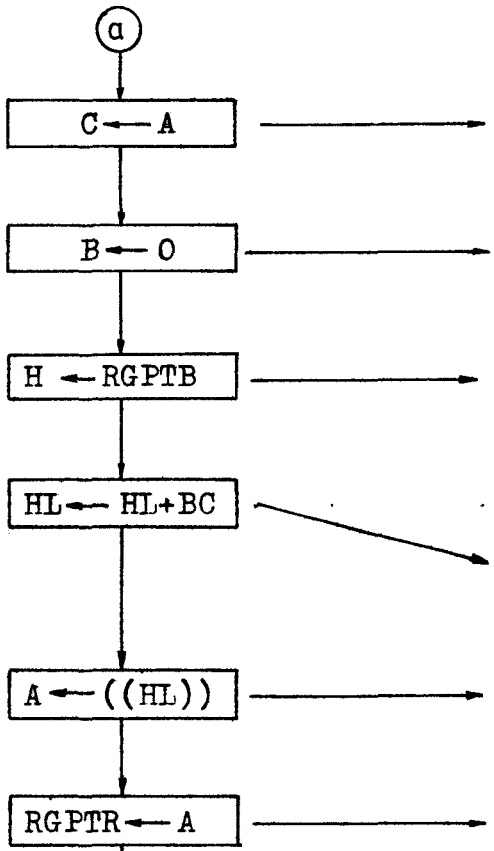
Entradas: ninguna.

Salidas: el carry se pone a 1 si el carácter leído es un registro; si no, se pone a 0.

Llamadas: RDKBD.

(ver la función en lenguaje ensamblador, pág 19).(Anexo2)





.....

(1) Ver RDKBD, (pág 145).

(2) Cuando pulsamos cualquier tecla, de la "0" a la "F", el código que se genera es <10 (en hexadecimal). Si pulsamos un comando generamos un número >10; es decir, si en el acumulador tenemos un número 10, el carry = 0 , nos indica que se pulsó un dígito hex.; Si en el acumulador tenemos un número 10, el carry ≠ 0, nos indica que se pulsó un comando.

(3) Si nos fijamos en el teclado, las teclas que designan a los registros son de la "3" (reg. I) a la "F" (reg.F). Esta resta la hacemos para distinguir cuando se pulsan las teclas "0", "1", "2" que no designan ningún registro.

(4) Si C = 1, indica que se pulsó una tecla $\geqslant 3$.
Si C = 0, indica que se pulsó una tecla < 3.

(5) En ésta tabla los registros están colocados en el mismo orden que en el teclado. Si, por ejemplo, pulsamos la tecla "7" (reg. SPH, palabra de mayor orden del Stack Pointer) al realizar la instrucción: A ← A - 3 ; nos queda en el Ac. 04, es decir, que la dirección que ponemos en los registros HL al realizar: HL ← HL + BC ; será:

03AC + 03 = 03AF

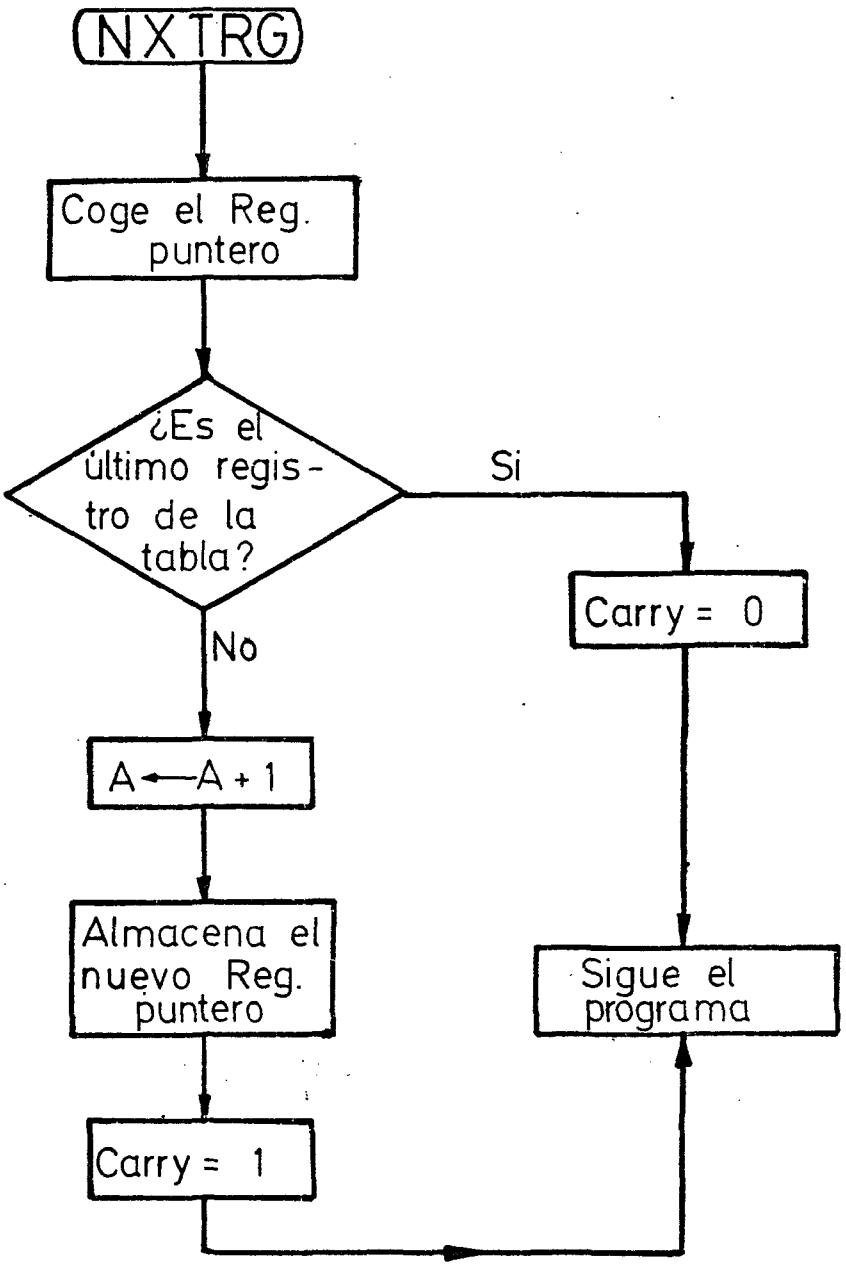
Primera posición de la tabla.

El puntero para entrar en la tabla.

El contenido de la dirección que tenemos en HL (03AF), nos sirve como puntero para la tabla NMTBL (tabla donde están los nombres de los registros.)

(6) Ver función RETT, (pág 125).

.....



Función: NXTRG.

"Avanza el registro señalado al próximo registro".

Si el registro señalado (registro puntero) en el display es el último de la tabla de registros (ver página 22) no se producirá el cambio y la función saca el mensaje "ERR". (Anexo 2)

Si el registro no es el último de la tabla, entonces se avanza al próximo registro y la función salta a "TRUE".

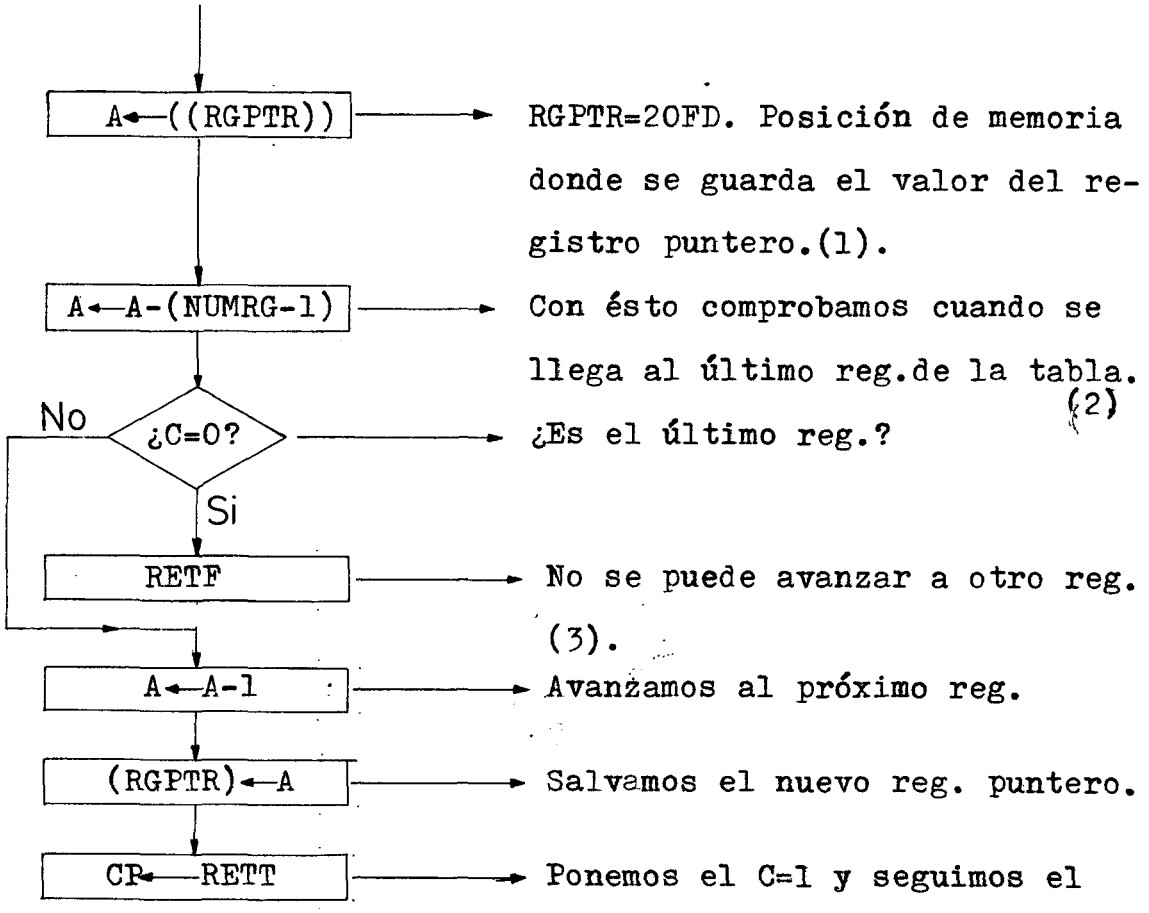
Entradas: ninguna.

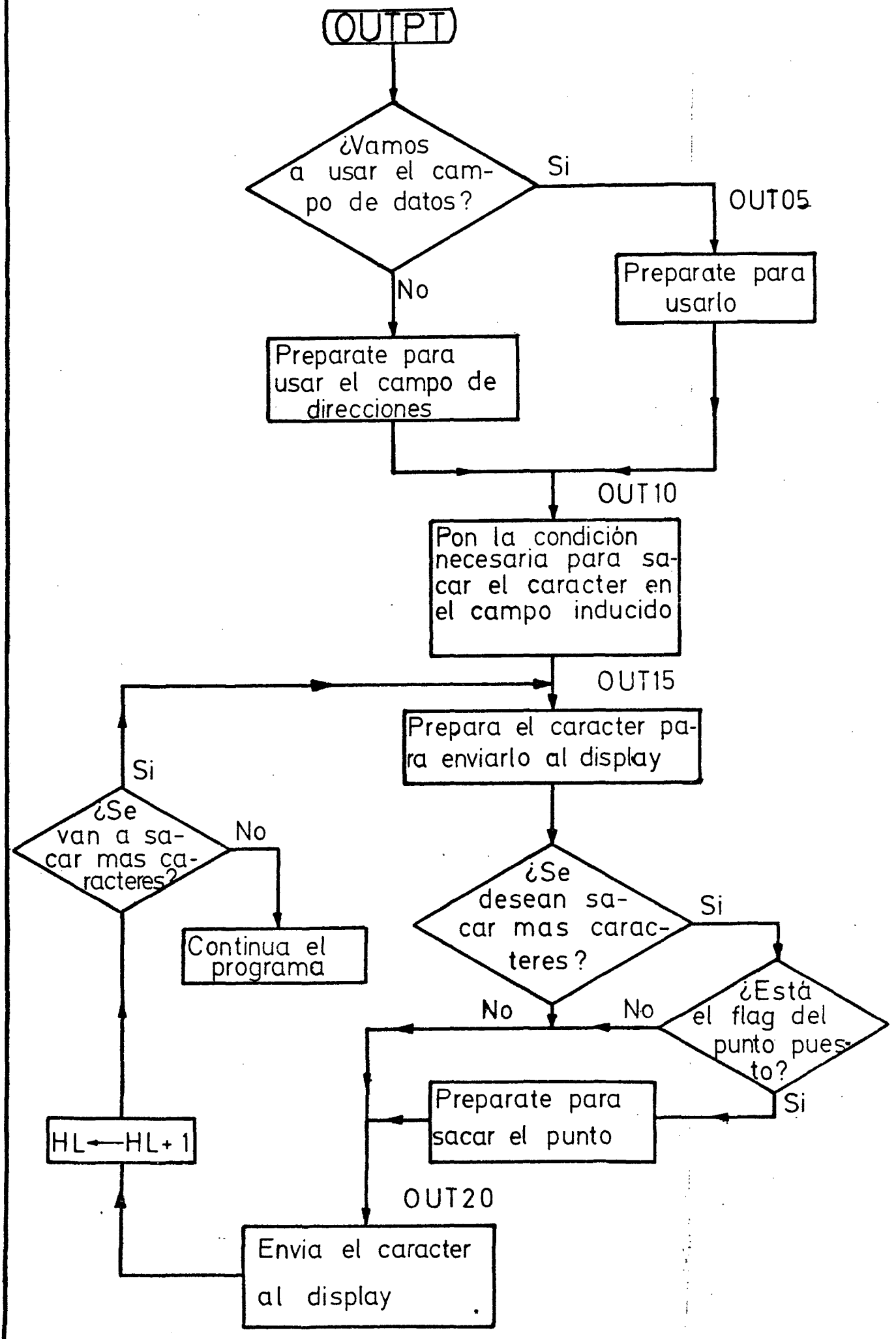
Salidas: C (carry) = 1 ⟶ cuando se puede avanzar a otro registro.

C= 0 ⟶ cuando no se puede avanzar.

Llamadas: ninguna.

(Ver la función en lenguaje ensamblador pág 15)(Anexo 2)





(1) Registro puntero es el que aparece en el display (en el campo de direcciones) cuando se hace uso del comando"EXAM".

(2) NUMRG nos indica el número de posiciones que ocupa la tabla de registros (ver pág 22).(Anexo 2)

(3) Ver la función "RETF" (pág 125).

(4) Ver la función "RETT" (pág 125).

.....

Función: OUTPT.

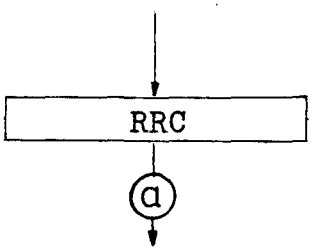
"Sacar caracteres al display".

Esta función envía dos caracteres al campo de datos ó cuatro al campo de direcciones (dependiendo del estado del "display flag"). El estado del flag del punto determina cuando será enviado un punto al display con el último carácter de salida.

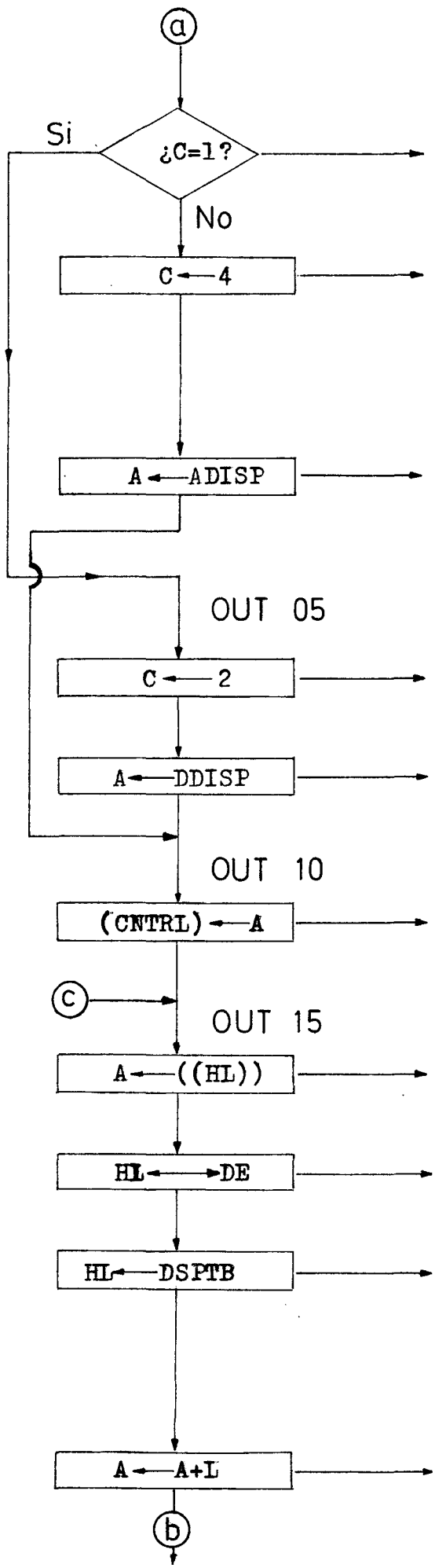
Entradas:-en A (acumulador) está el "display flag"
si es "0" se usa el campo de direcciones
si es "1" se usa el campo de datos.
-en B (reg B) está el "flag del punto"
si es "1" saca el punto en la parte derecha del campo indicado.
si es "0" no lo saca.
-en HL tenemos la dirección del primer carácter que se va a sacar al display.

Llamadas: ninguna.

(Ver la función en lenguaje ensamblador, pág 16).(Anexo 2).



El contenido del Ac. lo rotamos hacia la derecha a través del ca



rry. Se hace para saber el campo del display que se va a usar.(1).

¿Se va a usar el campo de datos?

Es un contador para saber el número de caracteres que se van a sacar(en el campo de direcciones se admiten 4 caracteres).

ADISP=90. Es un comando para que el chip 8279 controle el carácter que se va a enviar el campo de direcciones. (2).)

Es el contador para el campo de datos.

DDISP=94. Comando para que el chip 8279 controle el carácter que se va a enviar al campo de datos.(2).

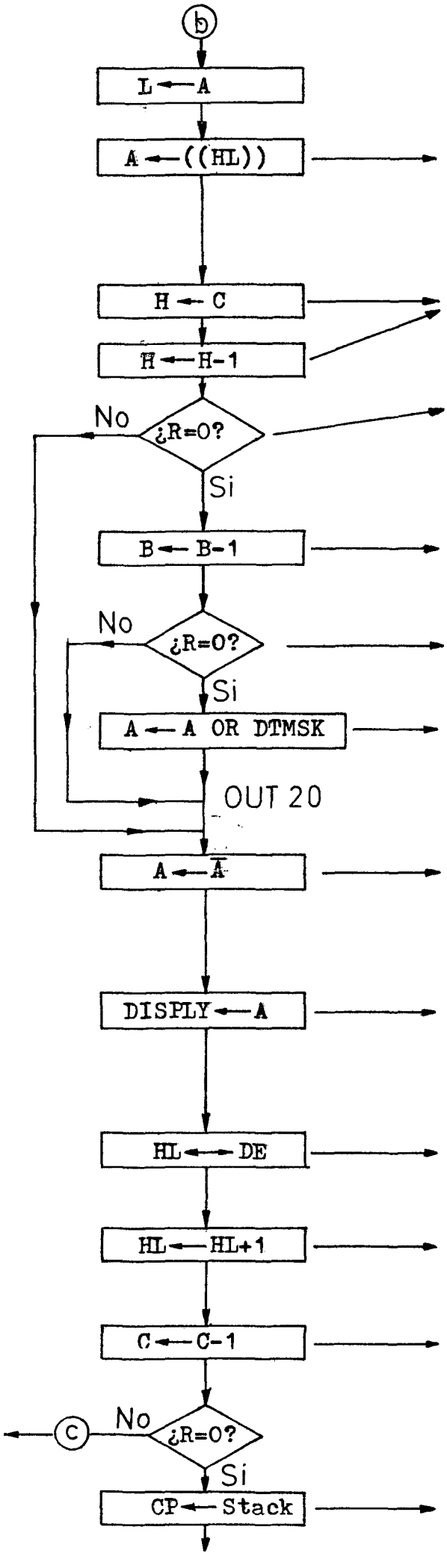
CNTRL=1900. Dirección de memoria usada para direccionar el chip 8279. (2).

Ponemos en el Ac. el primer carácter que se desea sacar.

La dirección que tenemos en HL la salvamos en los registros D y E.

DSPTB=0384. Dirección donde tenemos la tabla con los formatos de los caracteres que se pueden sacar al display (Ver pág 21).(Anexo 2)

Usamos el carácter de salida como



punto de entrada en la tabla.(3).

Cogemos de la tabla el formato perteneciente al carácter que deseamos visualizar.(3).

Para saber el número de caracteres que llevamos enviados al display.

R=Resultado.

¿ Es el último carácter que se va a enviar al display ?

Para averiguar si está el " Flag del punto " puesto.(4).

¿ Está el flag puesto ?

DTMSK=08. Variamos el formato que está en el Ac. para sacar el punto al display.(5).

Complementamos el acumulador debido a que los " led " del display son activos a nivel bajo.

DISPLY=1800. Posición de memoria para enviar los caracteres al display.

Restauramos en HL la dirección donde estaban los caracteres.

Dirección del siguiente carácter que se va a sacar.

Para averiguar cuantos caracteres llevamos sacados.

¿ Es el último carácter ?

Regresa.

(1) La instrucción RRC consiste en :
C=carry.
A=acumulador.
Si, por ejemplo, en el Ac. tenemos: Ø Ø/Ø Ø\Ø Ø Ø 1
después de que se ejecute ésta instrucción nos queda: C = 1 y A = 1 Ø Ø Ø\Ø Ø Ø Ø
Es decir, que el carry se pondrá a 1 ó a 0 dependiendo de lo que halla en el acumulador.

(2) Ver el chip 8279, pág 10 .

(3) Vamos a suponer que el carácter que deseamos sacar es la letra "B". En HL tenemos: 0384.
Al realizar A←A +L y A←L , tendremos:

(HL) = 0384 + B

Si realizamos ésta suma en binario:

```
Ø Ø Ø Ø/Ø Ø 1 1/1 Ø Ø Ø/Ø 1 Ø Ø +
                          1 Ø 1 1Ø
-------------------------------------
Ø Ø Ø Ø/Ø Ø 1 1 1/1 Ø Ø Ø/1 1 1 1
```

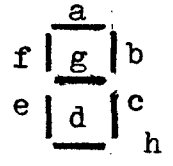
0 3 8 F

es la nueva dirección de HL.

Si nos fijamos en la tabla de formatos, el contenido de ésta posición de memoria es C7, que es el código que ha de tener un dígito del display para escribir "B". (ver características del display, pág 6).

(4) Al realizar la operación B←B - 1 , sabemos(fijándonos en el resultado) si el flag del punto está puesto ó no lo está. Si R ≠ 0 → no está puesto.
Si R = 1 si está.

(5) El formato de los dígitos del display tiene los siguientes segmentos:



Cada uno de los bits del Ac. actúa sobre uno de estos segmentos:

A = | d | c | b | a | h | g | f | e |

Si queremos que aparezca el punto, el bit "h" del acumulador tiene que estar a 1.

.....

Función: RDKBD.

" Leer el teclado"

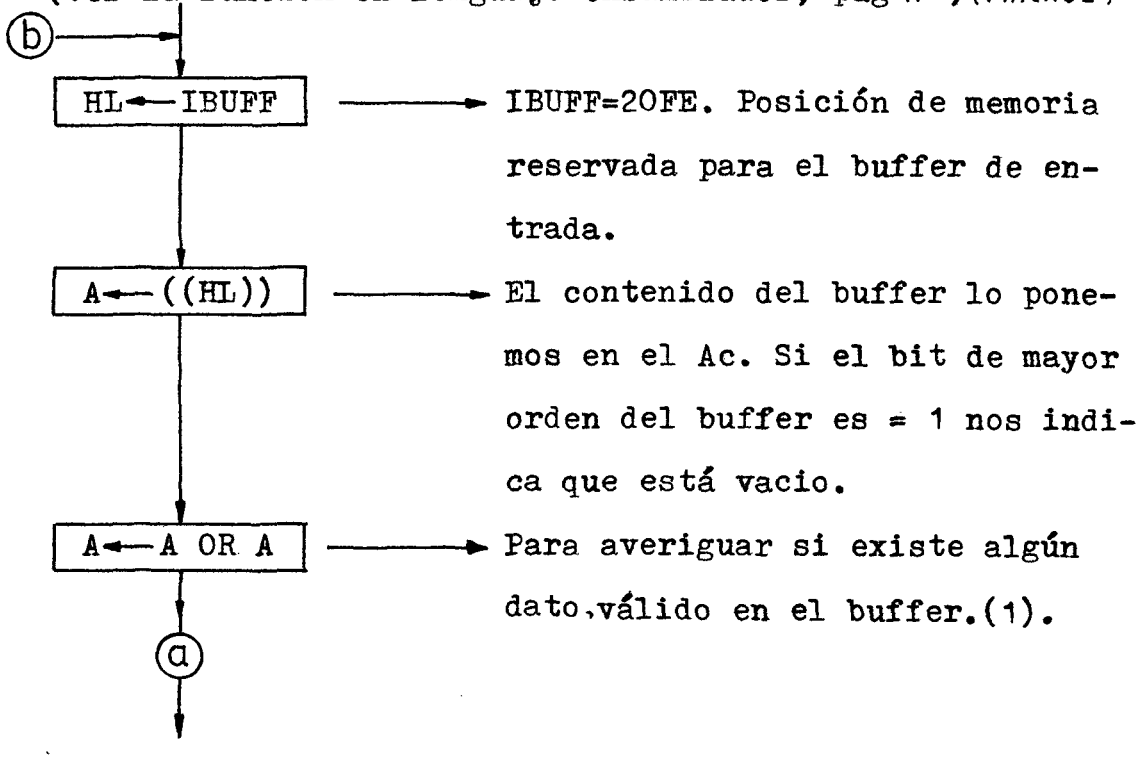
Esta función determina cuando tenemos un carácter en el buffer de entrada. Si no existe ninguna la función habilita las interrupciones y se mete en un lazo hasta que la rutina "ININT" carga algún carácter en el buffer. Cuando éste está lleno, la función coloca el flag de " vacio " en el buffer y pone el carácter recibido como carácter de salida.

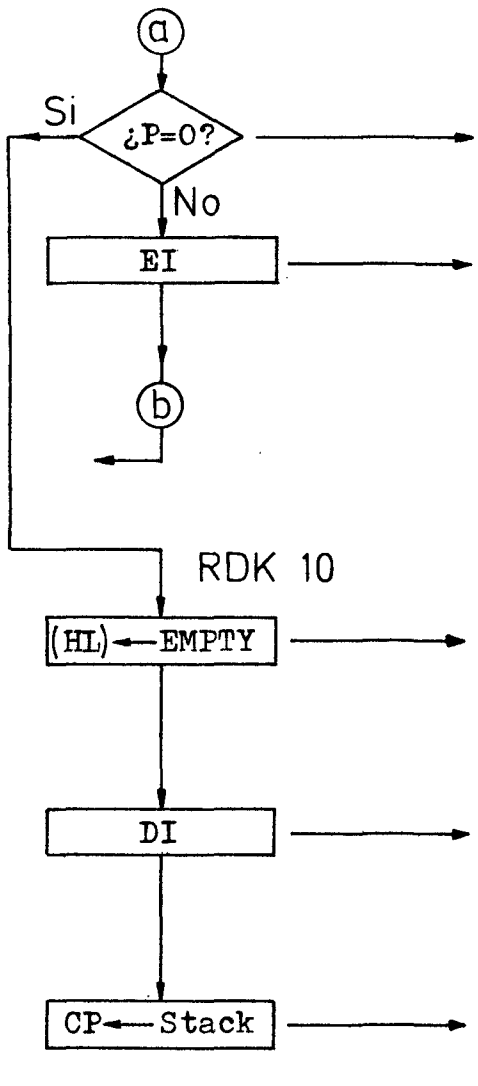
Entradas: ninguna.

Salidas: el carácter leído se coloca en el Ac.

Llamadas: ninguna.

(Ver la función en lenguaje ensamblador, pág 17)(Anexo2)





P= es el bit de signo.
¿hay algún dato válido?
EI= Enable Interrupts. Habilitamos las interrupciones y saltamos al principio. Establecemos un lazo hasta que "ININT" (ver pág 14)Anexo 2) cargue un dato en el buffer de entrada.
EMPTY=80. Ponemos el bit de mayor orden del buffer a 1 , para indicar que el buffer se queda vacio.
DI=Disable Interrupts,para regresar con las interrupciones deshabilitadas.
Regresa.

.....

(1) Si el bit de mayor orden del buffer está a 1, al realizar la operación "OR" sigue estando a 1 ⇒ que el bit de signo se pone a 1 indicándonos que no existe ningún dato en el buffer. Si el bit de mayor orden del buffer está a 0⇒que el bit de signo se pone a 0, indicándonos que sí existe un dato en el buffer.

.....

Función: INSDG.

" Inserta un dígito hexadecimal".

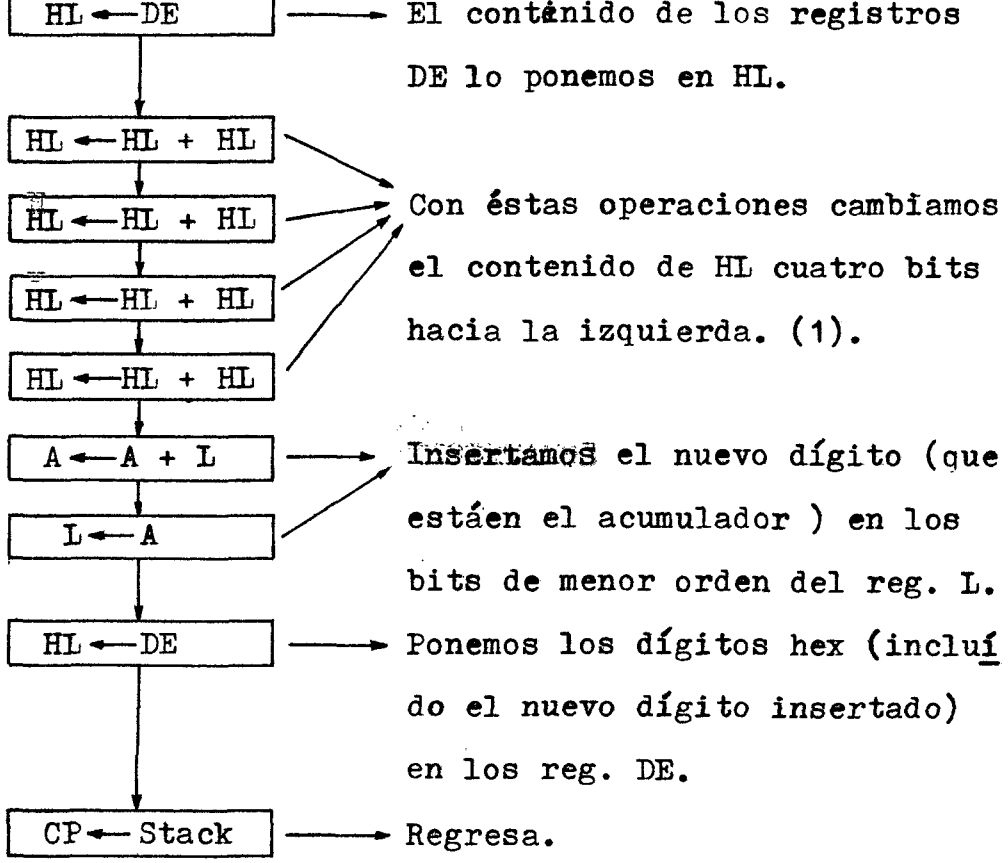
Esta función cambia el contenido de los registros DE cuatro bits hacia la izquierda (un dígito hex.) e inserta el nuevo dígito hex (que está en el acumulador) en los cuatro bits de más bajo orden del reg. E. El acumulador está preparado para contener un dígito hex. en los cuatro bits de orden bajo y ceros en los bits de orden alto.

Entradas: en el acumulador está el dígito hexadecimal que va a ser insertado.

Salidas: en DE tenemos los valores hexadecimales con el nuevo dígito insertado.

Llamadas: ninguna.

(Ver la función en lenguaje ensamblador, pág 15).(Anexo 2)



(1) Por ejemplo, si el contenido de los registros HL es: 0451 ; en binario será:

Ø Ø Ø Ø/Ø 1 Ø Ø/Ø 1 Ø 1/Ø Ø Ø 1

Si realizamos cuatro veces la operación: HL←HL+HL nos queda:

una vez →
Ø Ø Ø Ø/Ø 1 Ø Ø/Ø 1 Ø 1/Ø Ø Ø 1
Ø Ø Ø Ø/Ø 1 Ø Ø/Ø 1 Ø 1/Ø Ø Ø 1 +

dos →
Ø Ø Ø Ø/1 Ø Ø Ø/1 Ø 1 Ø/Ø Ø 1 Ø
Ø Ø Ø Ø/1 Ø Ø Ø/1 Ø 1 Ø/Ø Ø 1 Ø +

tres →
Ø Ø Ø 1/Ø Ø Ø 1/Ø 1 Ø Ø/Ø 1 Ø Ø
Ø Ø Ø 1/Ø Ø Ø 1/Ø 1 Ø Ø/Ø 1 Ø Ø +

cuatro →
Ø Ø 1 Ø/Ø Ø 1 Ø/1 Ø Ø Ø/1 Ø Ø Ø
Ø Ø 1 Ø/Ø Ø 1 Ø/1 Ø Ø Ø/1 Ø Ø Ø +

Ø 1 Ø Ø/Ø 1 Ø 1/Ø Ø Ø 1/Ø Ø Ø Ø
" " " "
4 5 1 0

Vemos que el contenido de HL se ha desplazado cuatro bits hacia la izquierda.

.....

Función: UPDAD.

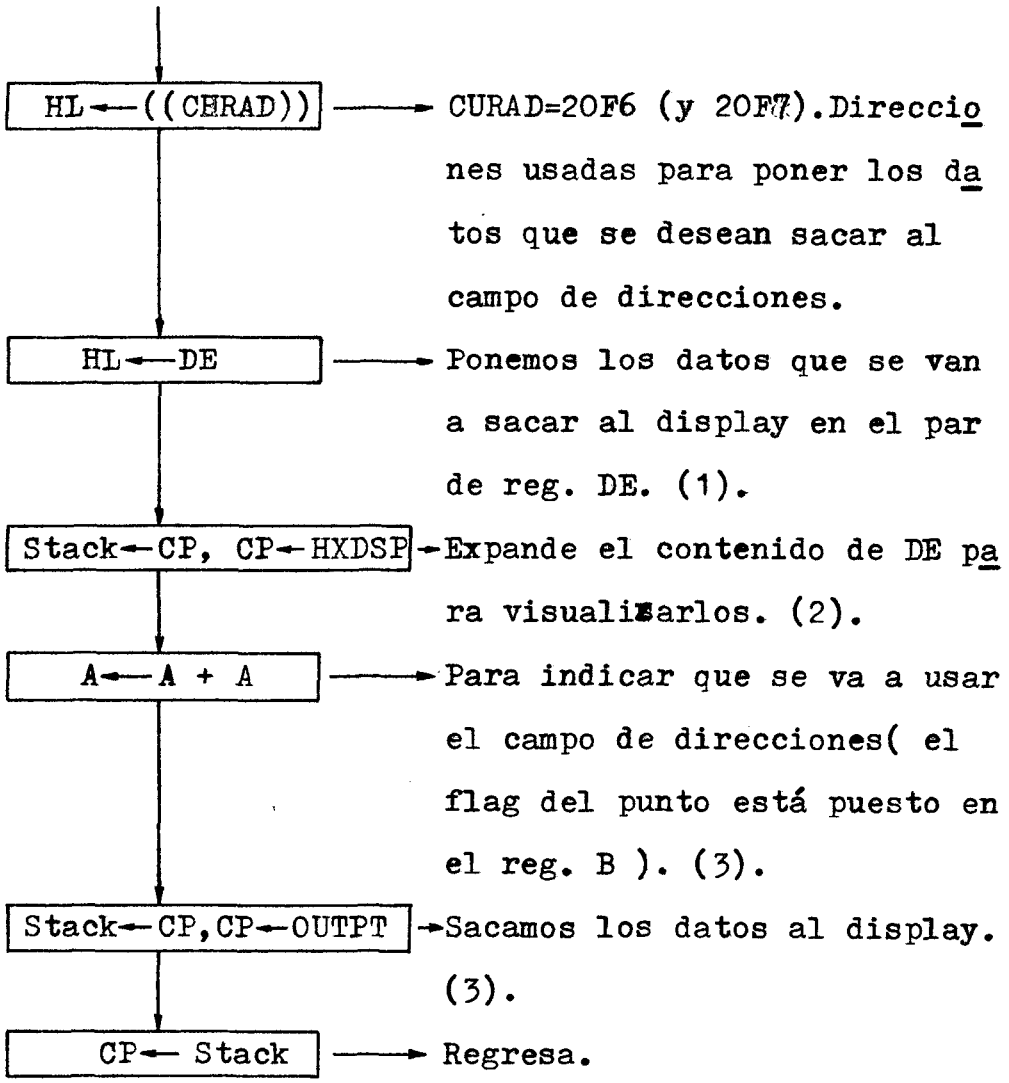
" Saca el dato al campo de direcciones de display."

Entradas: en "B" tenemos el flag del punto, si está a 1, significa que se saca un punto en la parte derecha del campo. Si está a 0, no lo saca.

Salidas: ninguna.

Llamadas: HXDSP, OUTPT.

(Ver la función en lenguaje ensamblador, pág 20).(Anexo 2)



.....

(1) Los datos han de estar en el par de registros

DE para ser expandidos por HXDSP.

(2) Ver HXDSP, (pág 113).

(3) Ver OUTPT, (pág 141).

.....

Función: UPDDT.

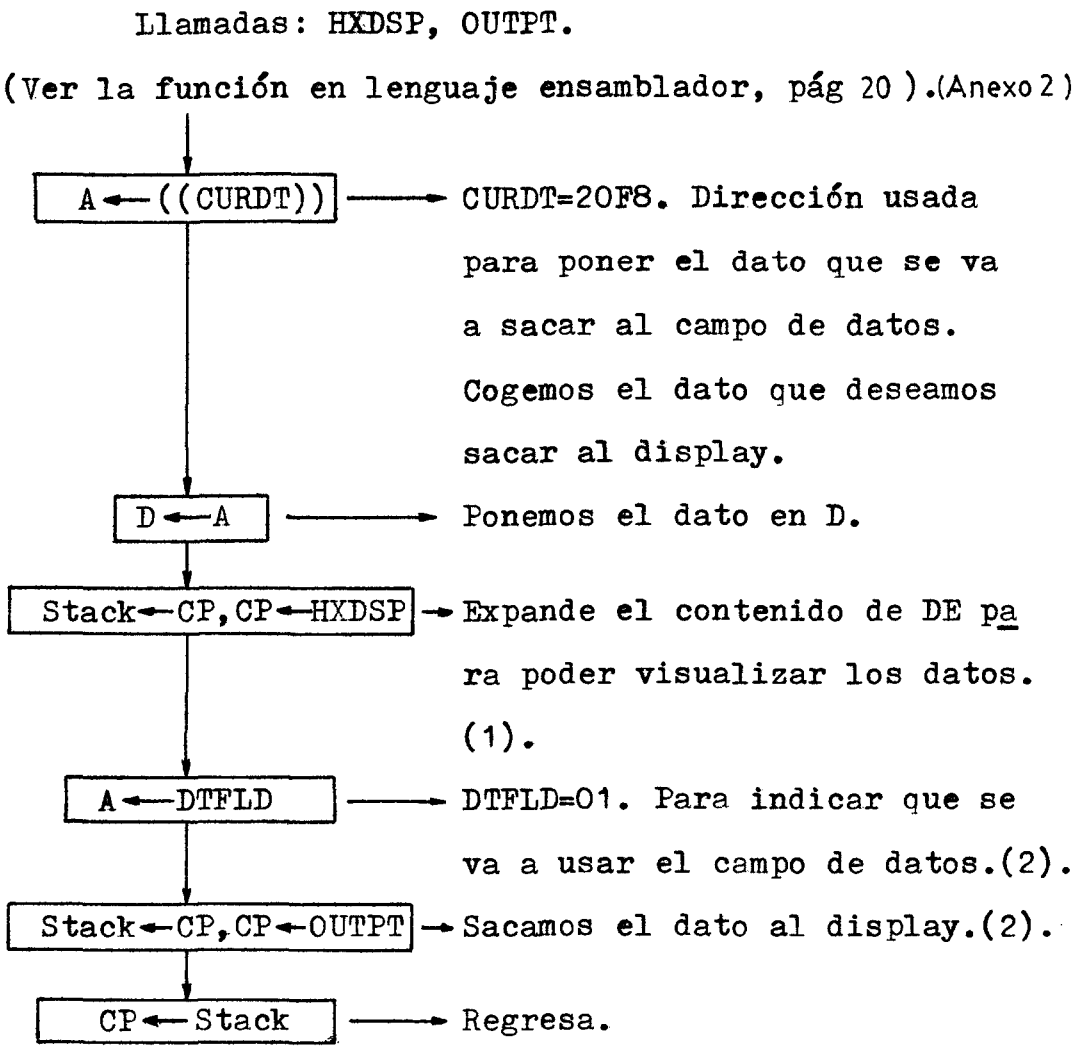
" Saca un dato al campo de datos del display".

Entradas: en "B" tenemos el flag del punto, si es = 1 la función saca un punto en la parte derecha del campo. Si es =0 no lo ,saca.

Salidas: ninguna.

Llamadas: HXDSP, OUTPT.

(Ver la función en lenguaje ensamblador, pág 20).(Anexo 2)



.....

(1) Ver HXDSP, (pág 113).

(2) Ver OUTPT, (pág 148).

.....

Función: CLDST.

"Arranque en frio".

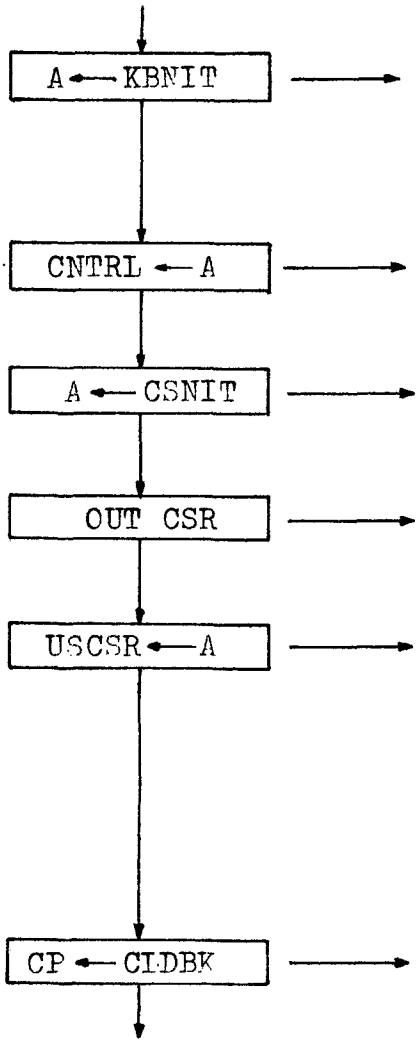
A ésta función saltamos cuando se enciende el SDK'85 para completar la inicialización del programa.

Entradas: ninguna.

Salidas: ninguna.

Llamadas: ninguna.

(Ver la fuhción en lenguaje ensamblador, pág 11).(Anexo2)



KBNIT=CC. Es un comando para indicarle al 8279 lo que tiene que hacer. (1).

CNTRL=1900. Dirección de memoria usada para direccionar el 8279.(1).

CSNIT=0. Valor inicial del registro comando/estado (C/S). (2).

CSR=20. Puerto de salida para el registro C/S .(2).

USCSR=20FF. Dirección de memoria destinada a guardar el contenido del registro C/S.El usuario puede poner aqui el valor que desee para dicho registro.

Saltamos a RST 1. (3).

(1) Ver el 8279,(anexo 1).

(2) Ver el 8155, (anexo 1).

(3) El monitor posee dos puntos de entrada:

RST 0: (arranque en frio), cuando se enciende el SDK"85.

RST 1: (arranque en caliente),si estamos usando el SDK'85 y queremos que se resetee el monitor, pero que nos guarde lo que tenemos escrito, entonces tenemos que saltar a ésta interrupción. Si mediante programa saltamos a RST 0, se borra lo que habiamos escritoanteriormente.

En los dos casos el monitor se resetea y aparece el mensaje "8085" en el display, pero con RST 1 nos guarda lo escrito y con RST 0 no.

.....

ANEXO 1

Características de los distintos dispositivos usados en el SDK'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.

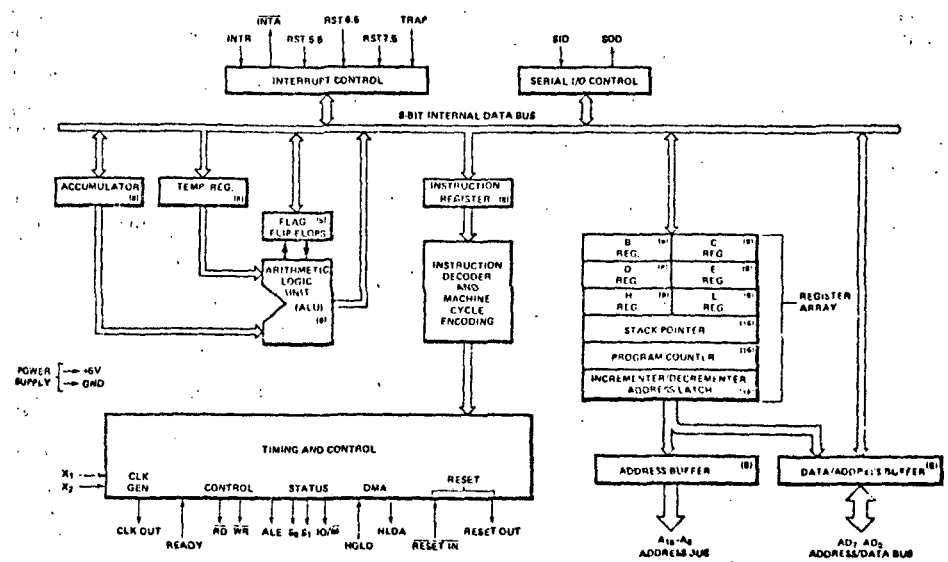


Figure 1. 8085A CPU Functional Block Diagram

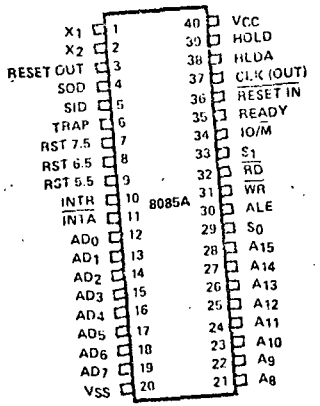


Figure 2. 8085A Pinout Diagram

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
S_1	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.

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.

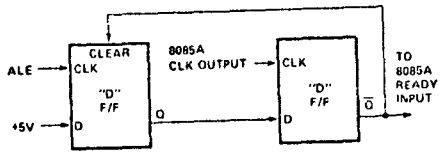


Figure 5. Generation of a Wait State for 8085A CPU

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.

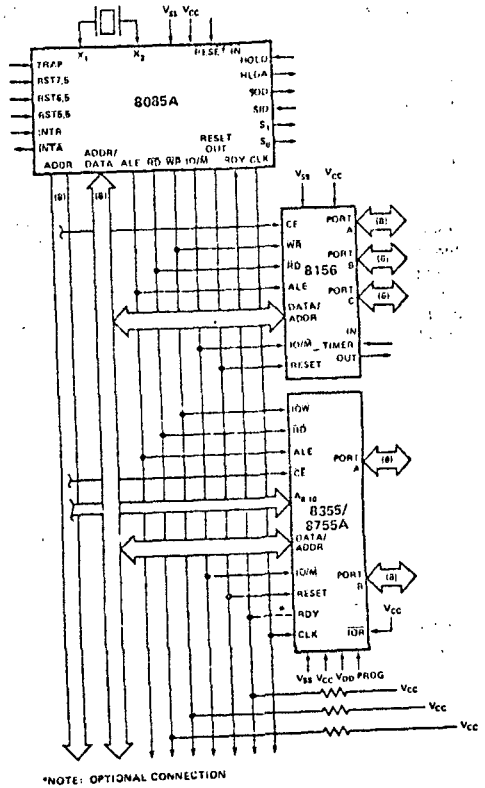


Figure 6. 8085A Minimum System (Standard I/O Technique)

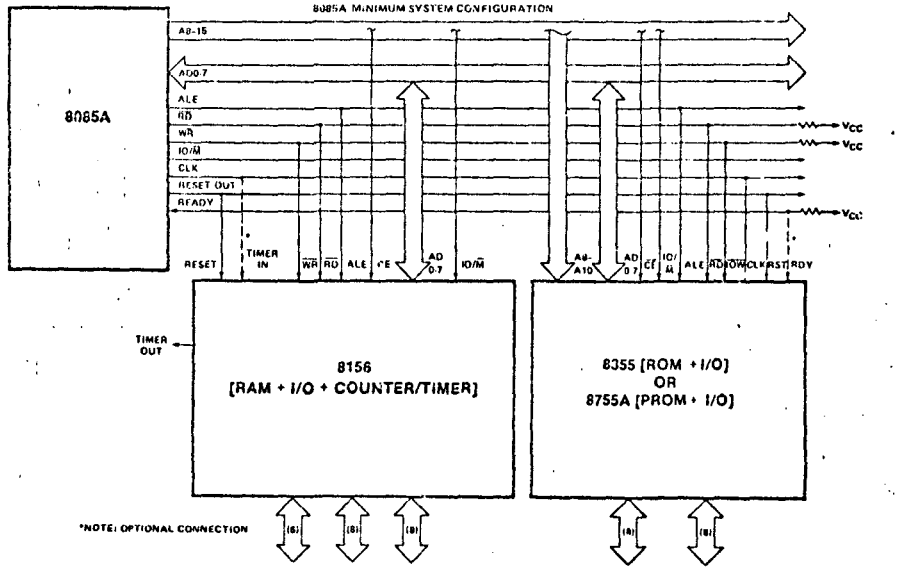


Figure 7. MCS-85™ Minimum System (Memory Mapped I/O)

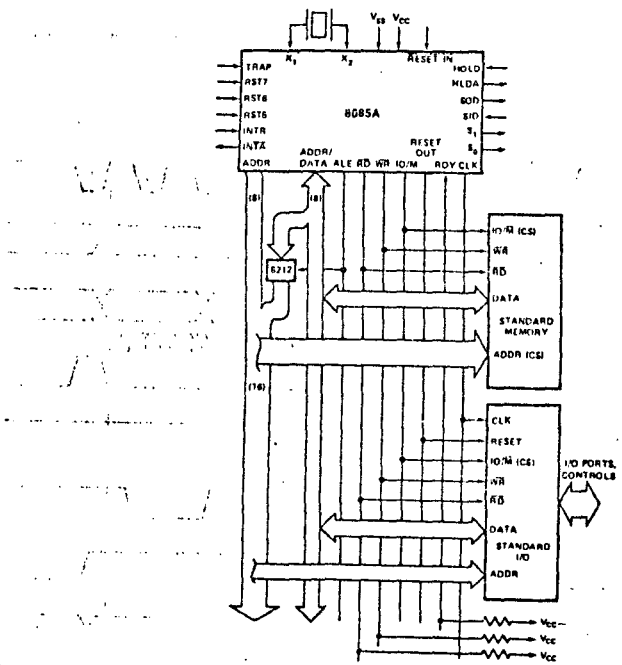


Figure 8. MCS-85™ System (Using Standard Memories)

Hold Operation

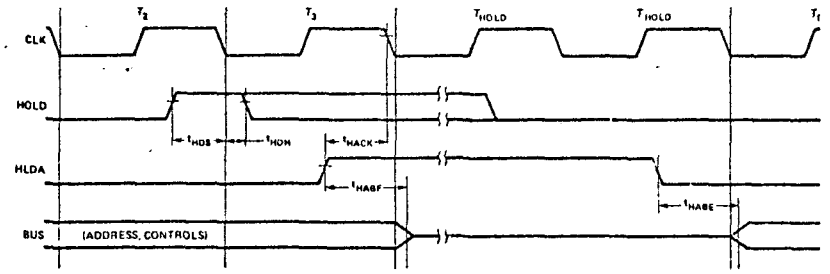


Figure 12. 8085A Hold Timing.

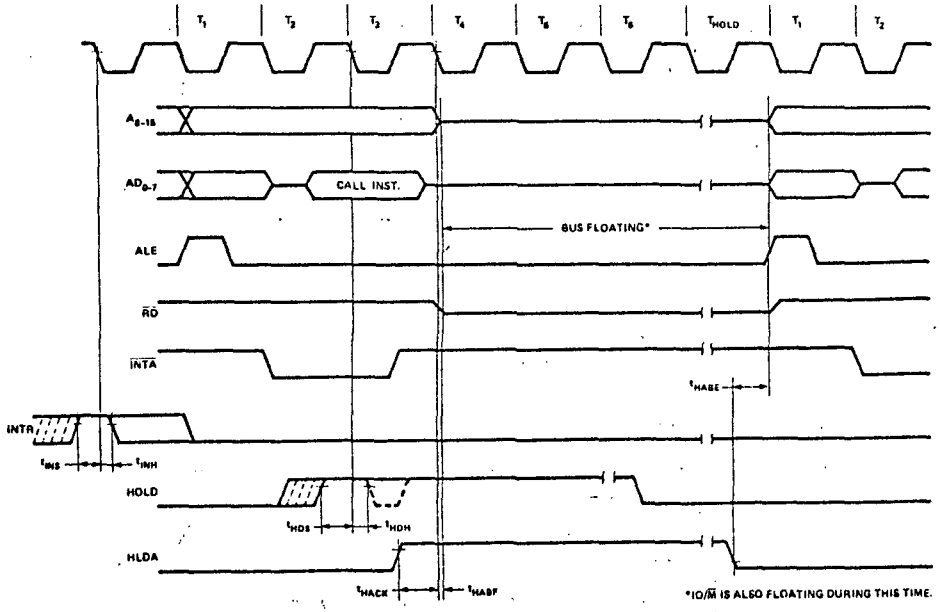


Figure 13. 8085A Interrupt and Hold Timing

8085A INSTRUCTION SET SUMMARY BY FUNCTIONAL GROUPING

Table 6-1

Mnemonic	Description	Instruction Code (1) D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	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	5-6
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	0	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	5-16
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	5-16
OUT	Output	1	1	0	1	0	0	1	1	5-16
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	5-9
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	5-6
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. DDS 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.

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

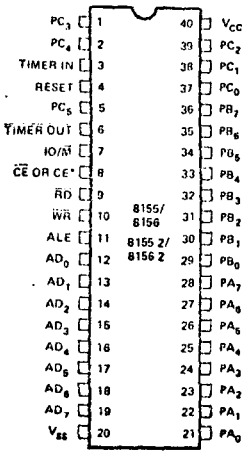
- 256 Word x 8 Bits
- Single +5V Power Supply
- Completely Static Operation
- Internal Address Latch
- 2 Programmable 8 Bit I/O Ports
- 1 Programmable 6-Bit I/O Port
- Programmable 14-Bit Binary Counter/ Timer
- Compatible with 8085A and 8088 CPU
- Multiplexed Address and Data Bus
- 40 Pin DIP

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.

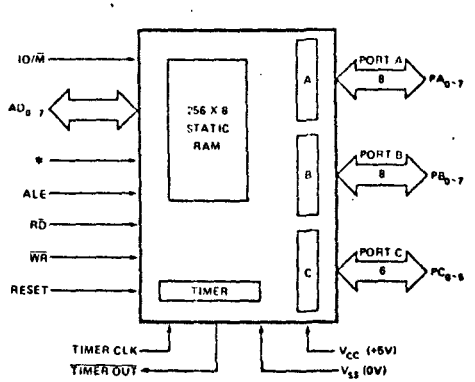
The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is also 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 — 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.)

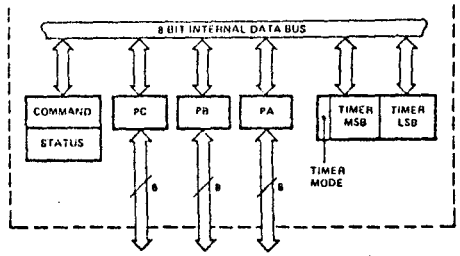
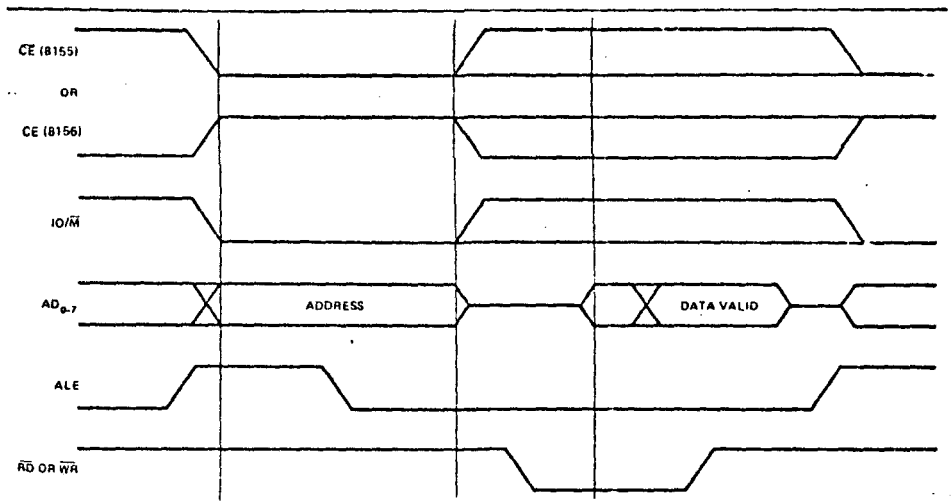


Figure 1. 8155/8156 Internal Registers



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.

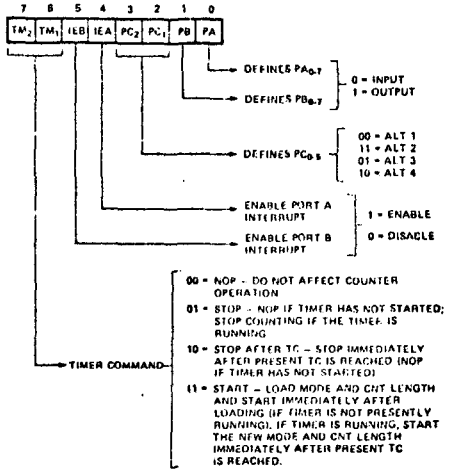


Figure 3. Command Register Bit Assignment

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.

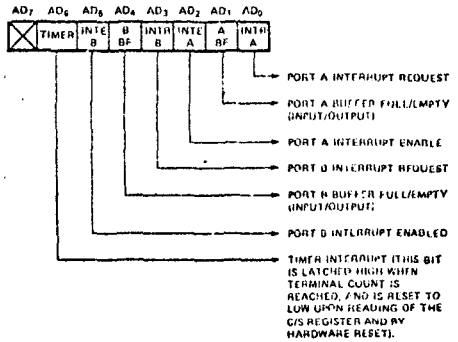


Figure 4. Status Register Bit Assignment

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:

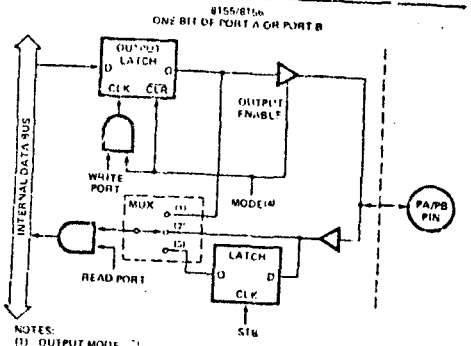


Figure 6. 8155/8156 Port Functions

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.

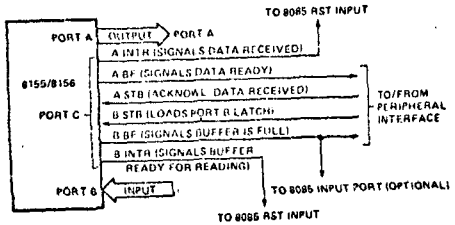


Figure 7. Example: Command Register = 00111001

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.

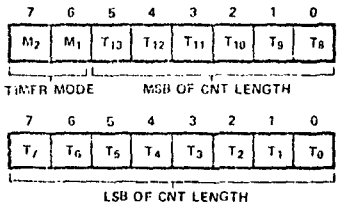


Figure 8. Timer Format

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

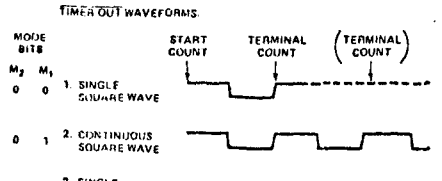


Figure 9. Timer Modes

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.

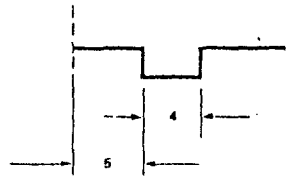


Figure 10. Asymmetrical Square-Wave Output Resulting from Count of 9

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.

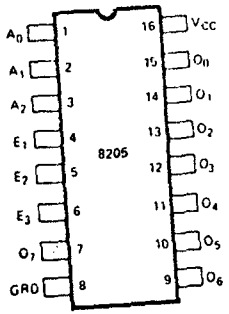
8205
HIGH SPEED 1 OUT OF 8 BINARY DECODER

- I/O Port or Memory Selector
- Simple Expansion — Enable Inputs
- High Speed Schottky Bipolar Technology — 18 ns Max Delay
- Directly Compatible with TTL Logic Circuits
- Low Input Load Current — 0.25 mA Max, 1/6 Standard TTL Input Load
- Minimum Line Reflection — Low Voltage Diode Input Clamp
- Outputs Sink 10 mA Min
- 16-Pin Dual In-Line Ceramic or Plastic Package

The Intel® 8205 decoder can be used for expansion of systems which utilize input ports, output ports, and memory components with active low chip select input. When the 8205 is enabled, one of its 8 outputs goes "low", thus a single row of a memory system is selected. The 3-chip enable inputs on the 8205 allow easy system expansion. For very large systems, 8205 decoders can be cascaded such that each decoder can drive 8 other decoders for arbitrary memory expansions.

The 8205 is packaged in a standard 16-pin dual in-line package, and its performance is specified over the temperature range of 0°C to +75°C, ambient. The use of Schottky barrier diode clamped transistors to obtain fast switching speeds results in higher performance than equivalent devices made with a gold diffussion process.

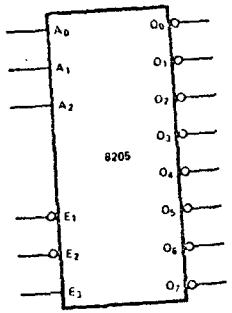
PIN CONFIGURATION



PIN NAMES

A_0-A_2	ADDRESS INPUTS
$\overline{E_1}$, $\overline{E_2}$, E_3	ENABLE INPUTS
$\overline{O_0}$-$\overline{O_7}$	DECODED OUTPUTS

LOGIC SYMBOL



ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

FUNCTIONAL DESCRIPTION

Decoder

The 8205 contains a one out of eight binary decoder. It accepts a three bit binary code and by gating this input, creates an exclusive output that represents the value of the input code.

For example, if a binary code of 101 was present on the A0, A1 and A2 address input lines, and the device was enabled, an active low signal would appear on the $\overline{O5}$ output line. Note that all of the other output pins are sitting at a logic high, thus the decoded output is said to be exclusive. The decoders outputs will follow the truth table shown below in the same manner for all other input variations.

Enable Gate

When using a decoder it is often necessary to gate the outputs with timing or enabling signals so that the exclusive output of the decoded value is synchronous with the overall system.

The 8205 has a built-in function for such gating. The three enable inputs ($\overline{E1}$, $\overline{E2}$, E3) are ANDed together and create a single enable signal for the decoder. The combination of both active "high" and active "low" device enable inputs provides the designer with a powerfully flexible gating function to help reduce package count in his system.

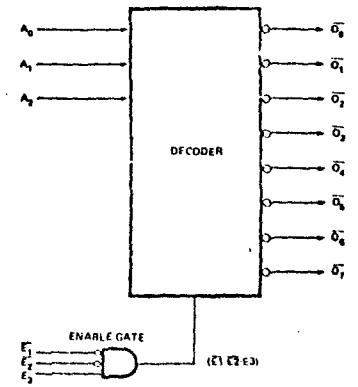


Figure 1. Enable Gate

ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

APPLICATIONS OF THE 8205

The 8205 can be used in a wide variety of applications in microcomputer systems. I/O ports can be decoded from the address bus, chip select signals can be generated to select memory devices and the type of machine state such as in 8008 systems can be derived from a simple decoding of the state lines (S0, S1, S2) of the 8008 CPU.

I/O Port Decoder

Shown in the figure below is a typical application of the 8205. Address input lines are decoded by a group of 8205s (3). Each input has a binary weight. For example, A0 is assigned a value of 1 and is the LSB; A4 is assigned a value of 16 and is the MSB. By connecting them to the decoders as shown, an active low signal that is exclusive in nature and represents the value of the input address lines, is available at the outputs of the 8205s.

This circuit can be used to generate enable signals for I/O ports or any other decoder related application.

Note that no external gating is required to decode up to 24 exclusive devices and that a simple addition of an inverter or two will allow expansion to even larger decoder networks.

Chip Select Decoder

Using a very similar circuit to the I/O port decoder, an array of 8205s can be used to create a simple interface to a 24K memory system.

The memory devices used can be either ROM or RAM and are 1K in storage capacity. 8308s and 8102s are the devices typically used for this application. This type of memory device has ten (10) address inputs and an active "low" chip select ($\overline{CS}$). The lower order address bits A0-A9 which come from the microprocessor are "bussed" to all memory elements and the chip select to enable a specific device or group of devices comes from the array of 8205s. The output of the 8205 is active low so it is directly compatible with the memory components.

Basic operation is that the CPU issues an address to identify a specific memory location in which it wishes to "write" or "read" data. The most significant address bits A10-A14 are decoded by the array of 8205s and an exclusive, active low, chip select is generated that enables a specific memory device. The least significant address bits A0-A9 identify a specific location within the selected device. Thus, all addresses throughout the entire memory array are exclusive in nature and are non-redundant.

This technique can be expanded almost indefinitely to support even larger systems with the addition of a few inverters and an extra decoder (8205).

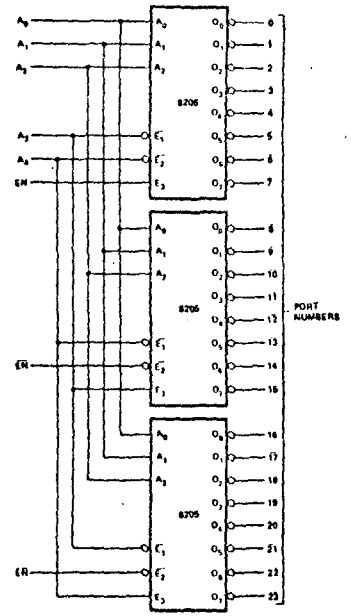


Figure 2. I/O Port Decoder

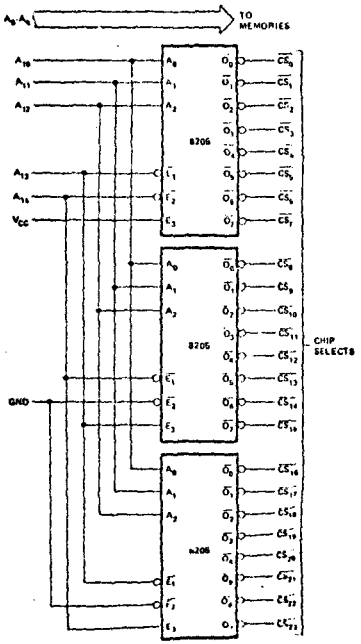


Figure 3. 32K Memory Interface

Logic Element Example

Probably the most overlooked application of the 8205 is that of a general purpose logic element. Using the "on-chip" enabling gate, the 8205 can be configured to gate its decoded outputs with system timing signals and generate strobes that can be directly connected to latches, flip-flops and one-shots that are used throughout the system.

An excellent example of such an application is the "state decoder" in an 8008 CPU based system. The 8008 CPU issues three bits of information (S0, S1, S2) that indicate the nature of the data on the Data Bus during each machine state. Decoding of these signals is vital to generate strobes that can load the address latches, control bus discipline and general machine functions.

In the figure below a circuit is shown using the 8205 as the "state decoder" for an 8008 CPU that not only decodes the S0, S1, S2 outputs but gates these signals with the clock (phase 2) and the SYNC output of the 8008 CPU. The $\overline{T1}$ and $\overline{T2}$ decoded strobes can connect directly to devices like 8212s for latching the address information. The other decoded strobes can be used to generate signals to control the system data bus, memory timing functions and interrupt structure. RESET is connected to the enable gate so that strobes are not generated during system reset, eliminating accidental loading.

The power of such a circuit becomes evident when a single decoded strobe is logically broken down. Consider $\overline{T1}$ output, the boolean equation for it would be:

$$\overline{T1} = (\overline{S0}\cdot S1\cdot\overline{S2})\cdot(\overline{SYNC}\cdot Phase\ 2\cdot\overline{Reset})$$

A six input NAND gate plus a few inverters would be needed to implement this function. The seven remaining outputs would need a similar circuit to duplicate their function, obviously a substantial savings in components can be achieved when using such a technique.

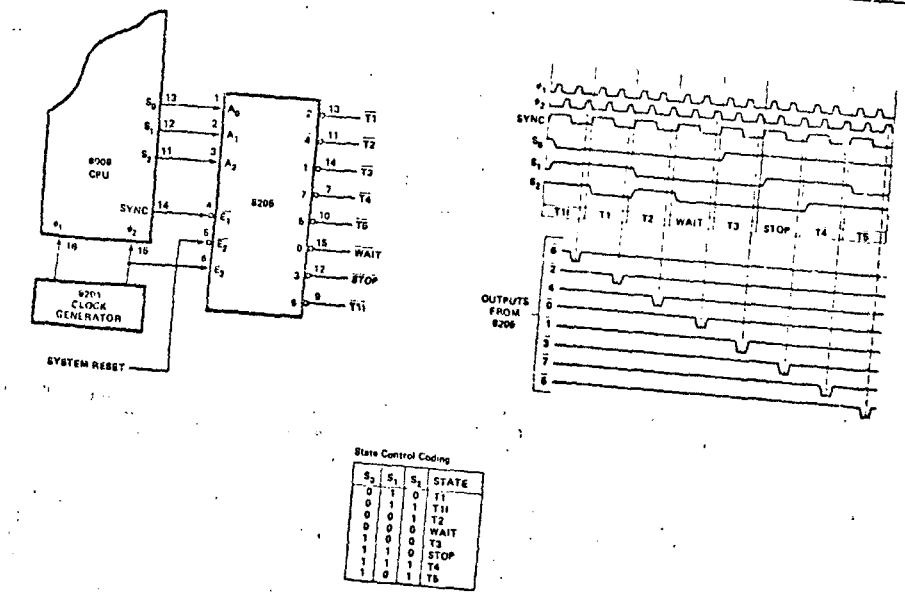


Figure 4. 8205 State Decoder Circuit

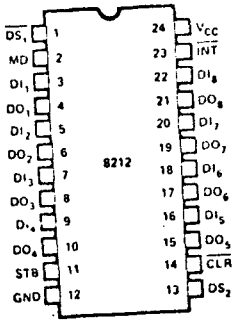
8212
8-BIT INPUT/OUTPUT PORT

- Fully Parallel 8-Bit Data Register and Buffer
- Service Request Flip-Flop for Interrupt Generation
- Low Input Load Current — .25mA Max.
- Three State Outputs
- Outputs Sink 15mA
- 3.65V Output High Voltage for Direct Interface to 8008, 8080A, or 8035A CPU
- Asynchronous Register Clear
- Replaces Buffers, Latches and Multiplexers in Microcomputer Systems
- Reduces System Package Count

The 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.

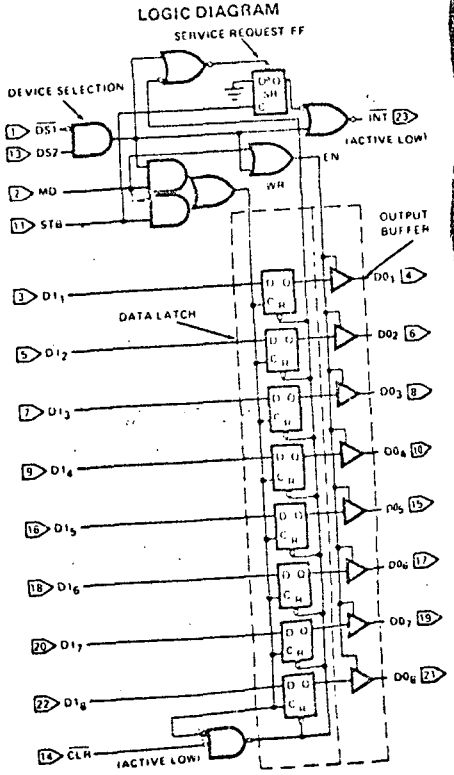
PIN CONFIGURATION



PIN NAMES

DI_1 DI_8	DATA IN
DO_1 DO_8	DATA OUT
$\overline{DS_1}$, DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

LOGIC DIAGRAM



FUNCTIONAL DESCRIPTION

Data Latch

The 8 flip-flops that make up the data latch are of a "D" type design. The output (Q) of the flip-flop will follow the data input (D) while the clock input (C) is high. Latching will occur when the clock (C) returns low.

The latched data is cleared by an asynchronous reset input ($\overline{CLR}$). (Note: Clock (C) Overrides Reset ($\overline{CLR}$).)

Output Buffer

The outputs of the data latch (Q) are connected to 3-state, non-inverting output buffers. These buffers have a common control line (EN); this control line either enables the buffer to transmit the data from the outputs of the data latch (Q) or disables the buffer, forcing the output into a high impedance state. (3-state)

The high-impedance state allows the designer to connect the 8212 directly onto the microprocessor bi-directional data bus.

Control Logic

The 8212 has control inputs $\overline{DS1}$, DS2, MD and STB. These inputs are used to control device selection, data latching, output buffer state and service request flip-flop.

$\overline{DS1}$, DS2 (Device Select)

These 2 inputs are used for device selection. When $\overline{DS1}$ is low and DS2 is high ($\overline{DS1} \cdot DS2$) the device is selected. In the selected state the output buffer is enabled and the service request flip-flop (SR) is asynchronously set.

MD (Mode)

This input is used to control the state of the output buffer and to determine the source of the clock input (C) to the data latch.

When MD is high (output mode) the output buffers are enabled and the source of clock (C) to the data latch is from the device selection logic ($\overline{DS1} \cdot DS2$).

When MD is low (input mode) the output buffer state is determined by the device selection logic ($\overline{DS1} \cdot DS2$) and the source of clock (C) to the data latch is the STB (Strobe) input.

STB (Strobe)

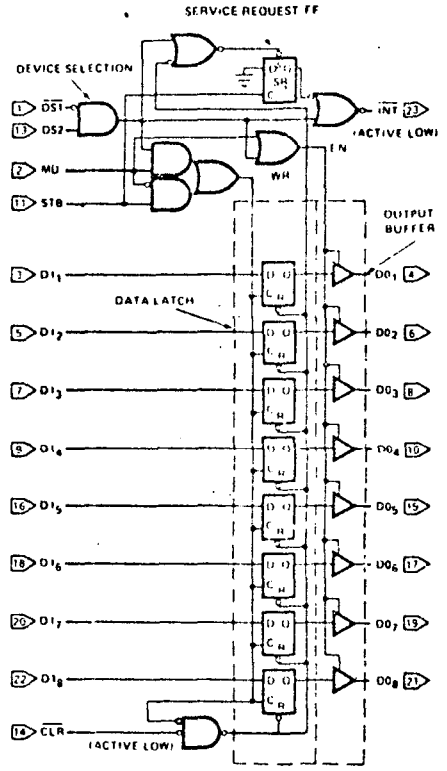
This input is used as the clock (C) to the data latch for the input mode MD = 0) and to synchronously reset the service request flip-flop (SR).

Note that the SR flip-flop is negative edge triggered.

Service Request Flip-Flop

The (SR) flip-flop is used to generate and control interrupts in microcomputer systems. It is asynchronously set by the $\overline{CLR}$ input (active low). When the (SR) flip-flop is set it is in the non-interrupting state.

The output of the (SR) flip-flop (Q) is connected to an inverting input of a "NOR" gate. The other input to the "NOR" gate is non-inverting and is connected to the device selection logic ($\overline{DS1} \cdot DS2$). The output of the "NOR" gate ($\overline{INT}$) is active low (interrupting state) for connection to active low input priority generating circuits.



STB	MD	(DS_1, DS_2)	DATA OUT EQUALS
0	0	0	3 STATE
1	0	0	3 STATE
0	1	0	DATA LATCH
1	1	0	DATA LATCH
0	0	1	DATA LATCH
1	0	1	DATA LATCH
0	1	1	DATA IN
1	1	1	DATA IN

$\overline{CLR}$ – RESETS DATA LATCH
SETS SR FLIP FLOP
(NO EFFECT ON OUTPUT BUFFER)

CLR	(DS_1, DS_2)	STB	*SR	INT
0	–	–	1	1
1	0	0	1	0
1	–	1	0	1
1	1	0	1	0
1	1	↓	1	0

*INTERNAL SR FLIP FLOP

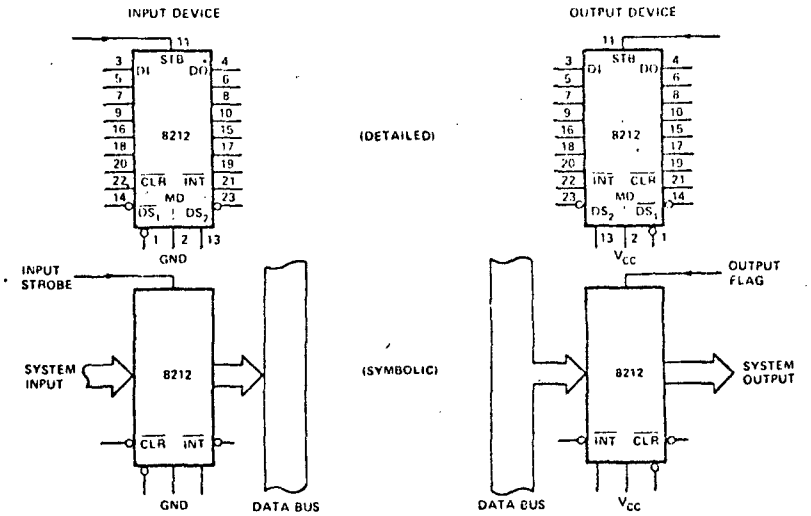
Applications of the 8212 — For Microcomputer Systems

I Basic Schematic Symbol
II Gated Buffer
III Bi-Directional Bus Driver
IV Interrupting Input Port
V Interrupt Instruction Port
VI Output Port
VII 8080A Status Latch
VIII 8085A Address Latch

1. Basic Schematic Symbols

Two examples of ways to draw the 8212 on system schematics — (1) the top being the detailed view showing pin numbers, and (2) the bottom being the symbolic view showing the system input or output as a system bus (bus containing 8 parallel lines). The output to the data bus is symbolic in referencing 8 parallel lines.

BASIC SCHEMATIC SYMBOLS



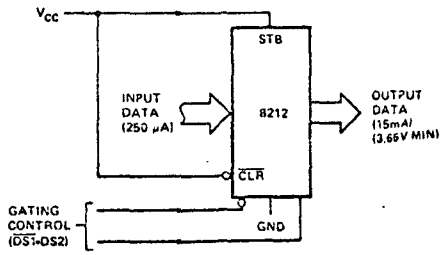
II. Gated Buffer (3-State)

The simplest use of the 8212 is that of a gated buffer. By tying the mode signal low and the strobe input high, the data latch is acting as a straight through gate. The output buffers are then enabled from the device selection logic $\overline{DS1}$ and DS2.

When the device selection logic is false, the outputs are 3-state.

When the device selection logic is true, the input data from the system is directly transferred to the output. The input data load is 250 micro amps. The output data can sink 15 milli amps. The minimum high output is 3.65 volts.

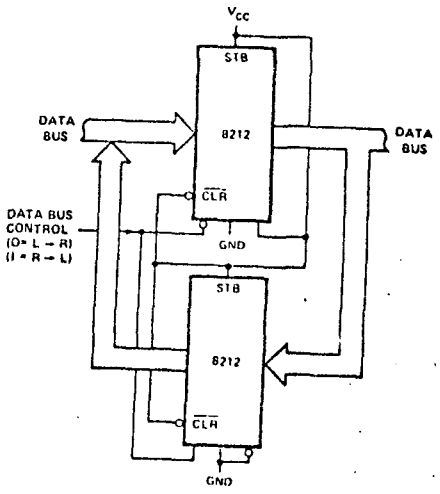
GATED BUFFER



III. Bi-Directional Bus Driver

A pair of 8212's wired (back-to-back) can be used as a symmetrical drive, bi-directional bus driver. The devices are controlled by the data bus input control which is connected to $\overline{DS1}$ on the first 8212 and to DS2 on the second. One device is active, and acting as a straight through buffer the other is in 3-state mode. This is a very useful circuit in small system design.

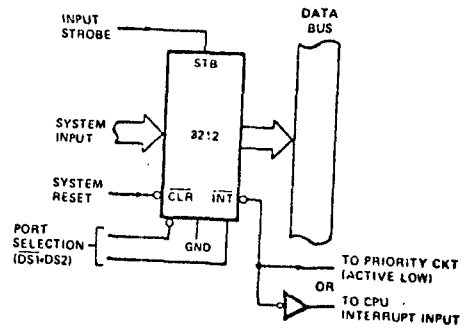
BI-DIRECTIONAL BUS DRIVER



IV. Interrupting Input Port

This use of an 8212 is that of a system input port that accepts a strobe from the system input source, which in turn clears the service request flip-flop and interrupts the processor. The processor then goes through a service routine, identifies the port, and causes the device selection logic to go true — enabling the system input data onto the data bus.

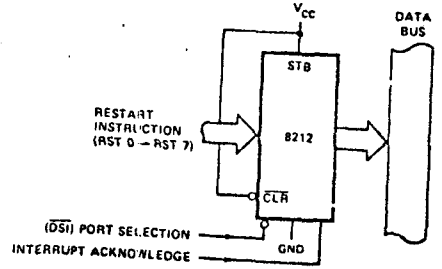
INTERRUPTING INPUT PORT



V. Interrupt Instruction Port

The 8212 can be used to gate the interrupt instruction, normally RESTART instructions, onto the data bus. The device is enabled from the interrupt acknowledge signal from the microprocessor and from a port selection signal. This signal is normally tied to ground. ($\overline{DS1}$ could be used to multiplex a variety of interrupt instruction ports onto a common bus).

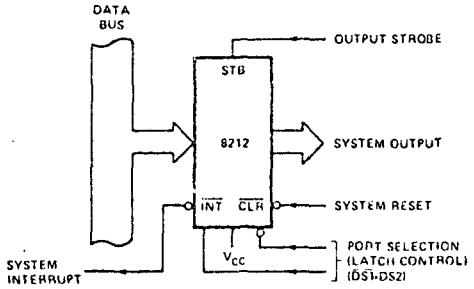
INTERRUPT INSTRUCTION PORT



VI. Output Port (With Hand-Shaking)

The 8212 can be used to transmit data from the data bus to a system output. The output strobe could be a hand-shaking signal such as "reception of data" from the device that the system is outputting to. It in turn, can interrupt the system signifying the reception of data. The selection of the port comes from the device selection logic. ($\overline{DS1} \cdot DS2$)

OUTPUT PORT (WITH HAND-SHAKING)

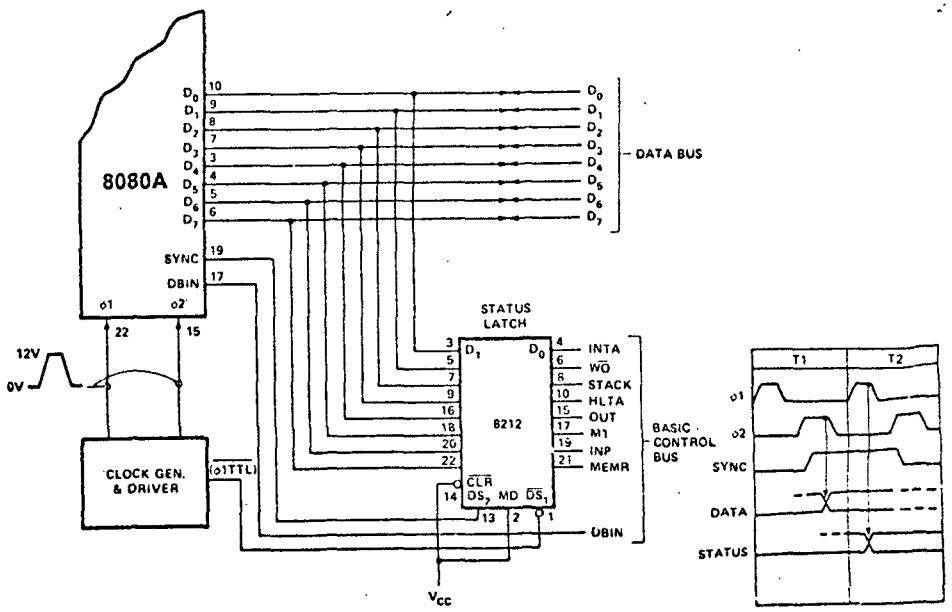


VII. 8080A Status Latch

Here the 8212 is used as the status latch for an 8080A microcomputer system. The input to the 8212 latch is directly from the 8080A data bus. Timing shows that when the SYNC signal is true, which is connected to the DS2 input and the phase 1 signal is true, which is a TTL level coming from the clock generator; then, the status data will be latched into the 8212.

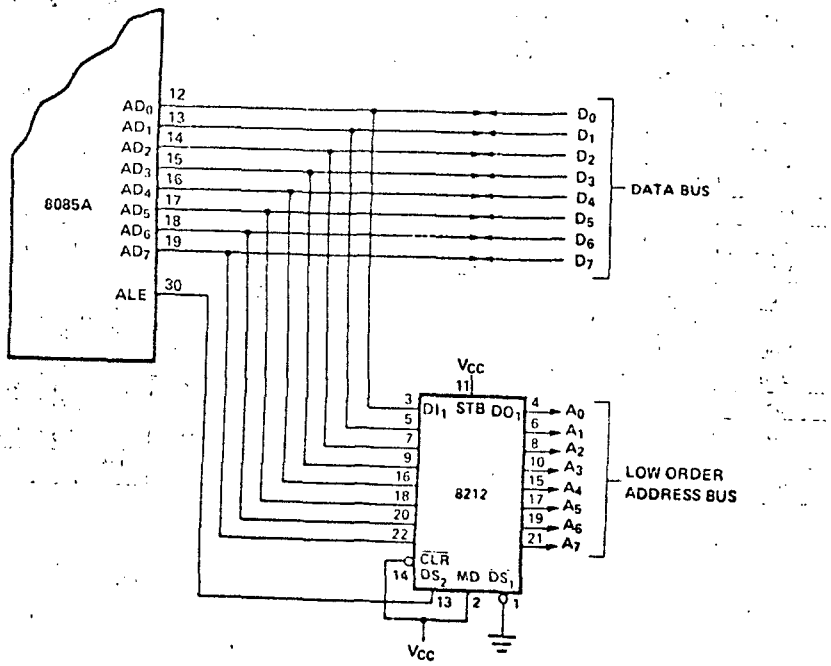
Note: The mode signal is tied high so that the output on the latch is active and enabled all the time.

It is shown that the two areas of concern are the bi-directional data bus of the microprocessor and the control bus.



VIII. 8085A Low-Order Address Latch

The 8085A microprocessor uses a multiplexed address/data bus that contains the low order 8-bits of address information during the first part of a machine cycle. The same bus contains data at a later time in the cycle. An address latch enable (ALE) signal is provided by the 8085A to be used by the 8212 to latch the address so that it may be available through the whole machine cycle. Note: In this configuration, the MODE input is tied high, keeping the 8212's output buffers turned on at all times.



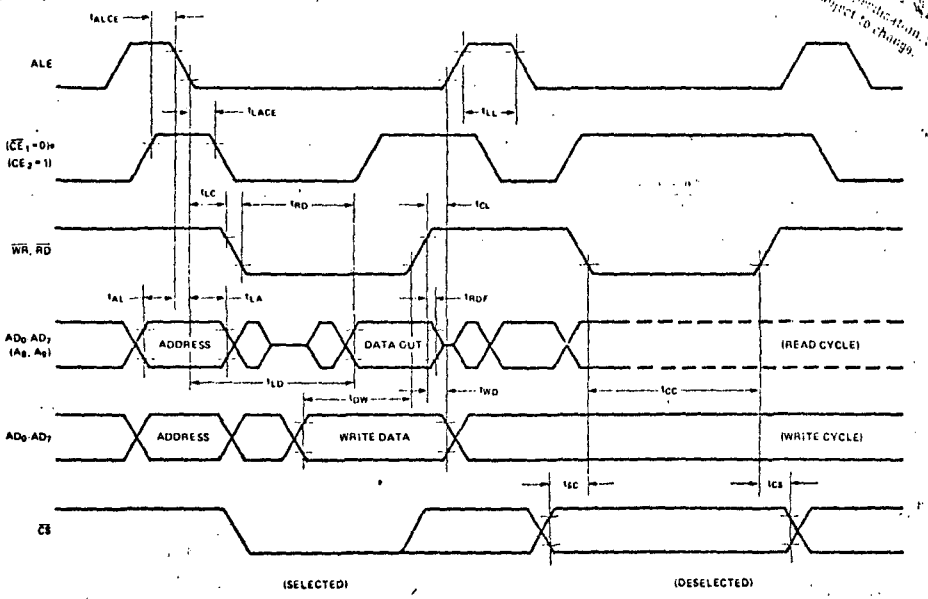


Figure 3. 8185 Timing.

8355/8355-2
16,384-BIT ROM WITH I/O

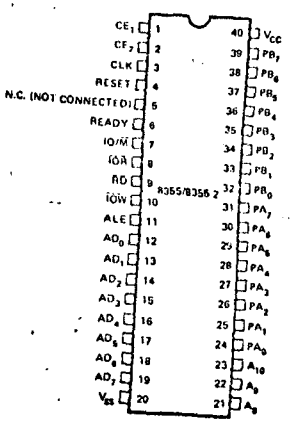
- 2048 Words x 8 Bits
- Single + 5V Power Supply
- Directly compatible with 8085A and 8088 Microprocessors
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- Internal Address Latch
- 40-Pin DIP

The Intel® 8355 is a ROM and I/O chip to be used in the 8085A and 8088 microprocessor systems. The ROM portion is organized as 2048 words by 8 bits. It has a maximum acess time of 400 ns to permit use with no wait states in the 8085A CPU.

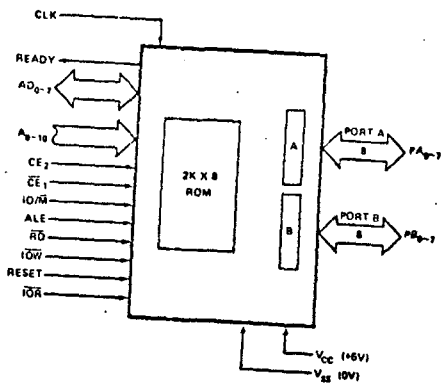
The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines and each I/O port line is individually programmable as input or output.

The 8355-2 has a 300ns access time for compatibility with the 8085A-2 and full speed 5 MHz 8088 microprocessors.

PIN CONFIGURATION



BLOCK DIAGRAM



Symbol	Function
ALE (Input)	When ALE (Address Latch Enable is high, AD_{0-7}, IO/$\overline{M}$, A_{8-10}, CE, and $\overline{CE}$ enter address latched. The signals (AD, IO/$\overline{M}$, A_{8-10}, CE, $\overline{CE}_1$ are latched in at the trailing edge of ALE.
AD_{0-7} (Input)	Bidirectional Address/Data bus. The lower 8-bits of the ROM or I/O address are applied to the bus lines when ALE is high. During an I/O cycle, Port A or B are selected based on the latched value of AD_0. If $\overline{RD}$ or $\overline{IOR}$ is low when the latched chip enables are active, the output buffers present data on the bus.
A_{8-10} (Input)	These are the high order bits of the ROM address. They do not affect I/O operations.
$\overline{CE}_1$ CE_2 (Input)	Chip Enable Inputs: $\overline{CE}_1$ is active low and CE_2 is active high. The 8355 can be accessed only when BOTH Chip Enables are active at the time the ALE signal latches them up. If either Chip Enable input is not active, the AD_{0-7} and READY outputs will be in a high impedance state.
IO/$\overline{M}$ (Input)	If the latched IO/$\overline{M}$ is high when $\overline{RD}$ is low, the output data comes from an I/O port. If it is low the output data comes from the ROM.
$\overline{RD}$ (Input)	If the latched Chip Enables are active when $\overline{RD}$ goes low, the AD_{0-7} output buffers are enabled and output either the selected ROM location or I/O port. When both $\overline{RD}$ and $\overline{IOR}$ are high, the AD_{0-7} output buffers are 3-state.
$\overline{IOW}$ (Input)	If the latched Chip Enables are active, a low on $\overline{IOW}$ causes the output port pointed to by the latched value of AD_0 to be written with the data on AD_{0-7}. The state of IO/$\overline{M}$ is ignored.

Symbol	Function
CLK (Input)	The CLK is used to force the READY into its high impedance state after it has been forced low by $\overline{CE}$ low, CE high and ALE high.
READY (Output)	Ready is a 3-state output controlled by $\overline{CE}_1$, CE_2, ALE and CLK. READY is forced low when the Chip Enables are active during the time ALE is high, and remains low until the rising edge of the next CLK (see Figure 6).
PA_{0-7} (Input/ Output)	These are general purpose I/O pins. Their input/output direction is determined by the contents of Data Direction Register (DDR). Port A is selected for write operations when the Chip Enables are active and $\overline{IOW}$ is low and a 0 was previously latched from AD_0. Read operation is selected by either $\overline{IOR}$ low and active Chip Enables and AD_0 low, or IO/$\overline{M}$ high, $\overline{RD}$ low, active chip enables, and AD_0 low.
PB_{0-7} (Input/ Output)	This general purpose I/O port is identical to Port A except that it is selected by a 1 latched from AD_0.
RESET (Input)	An input high on RESET causes all pins in Port A and B to assume input mode.
$\overline{IOR}$ (Input)	When the Chip Enables are active, a low on $\overline{IOR}$ will output the selected I/O port onto the AD bus. $\overline{IOR}$ low performs the same function as the combination IO/$\overline{M}$ high and $\overline{RD}$ low. When $\overline{IOR}$ is not used in a system, $\overline{IOR}$ should be tied to Vcc ("1").
Vcc	+5 volt supply.
Vss	Ground Reference.

FUNCTIONAL DESCRIPTION

ROM Section

The 8355 contains an 8-bit address latch which allows it to interface directly to MCS-48 and MCS-85 Microcomputers without additional hardware.

The ROM section of the chip is addressed by an 11-bit address and the Chip Enables. The address and levels on the Chip Enable pins are latched into the address latches on the falling edge of ALE. If the latched Chip Enables are active and IO/$\overline{M}$ is low when $\overline{RD}$ goes low, the contents of the ROM location addressed by the latched address are put out through AD_{0-7} output buffers.

I/O Section

The I/O section of the chip is addressed by the latched value of AD_{0-1}. Two 8-bit Data Direction Registers (DDR) in 8355 determine the input/output status of each pin in the corresponding ports. A "0" in a particular bit position of a DDR signifies that the corresponding I/O port bit is in the input mode. A "1" in a particular bit position signifies that the corresponding I/O port bit is in the output mode. In this manner the I/O ports of the 8355 are bit-by-bit programmable as inputs or outputs. The table summarizes port and DDR designation. DDR's cannot be read.

AD_1	AD_0	Selection
0	0	Port A
0	1	Port B
1	0	Port A Data Direction Register (DDR A)
1	1	Port B Data Direction Register (DDR B)

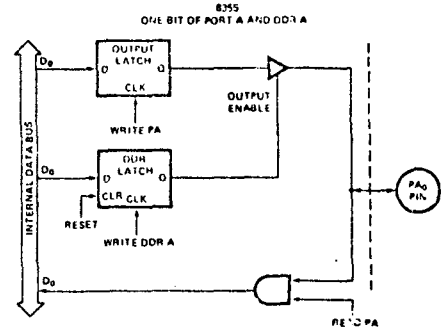
When $\overline{IOW}$ goes low and the Chip Enables are active, the data on the AD_{0-7} is written into I/O port selected by the latched value of AD_{0-1}. During this operation all I/O bits of the selected port are affected, regardless of their I/O mode and the state of IO/$\overline{M}$. The actual output level does not change until $\overline{IOW}$ returns high (glitch free output).

A port can be read out when the latched Chip Enables are active and either $\overline{RD}$ goes low with IO/$\overline{M}$ high, or $\overline{IOR}$ goes low. Both input and output mode bits of a selected port will appear on lines AD_{0-7}.

To clarify the function of the I/O ports and Data Direction Registers, the following diagram shows the configuration of one bit of PORT A and DDR A. The same logic applies to PORT B and DDR B.

Note that hardware RESET or writing a zero to the DDR latch will cause the output latch's output buffer to be disabled, preventing the data in the output latch from being passed through to the pin. This is equivalent to putting the port in the input mode. Note also that the data can be written to the Output Latch even though the Output Buffer has been disabled. This enables a port to be initialized with a value prior to enabling the output.

The diagram also shows that the contents of PORT A and PORT B can be read even when the ports are configured as outputs.



System Interface with 8085A and 8088

A system using the 8355 can use either one of the two I/O Interface techniques:

- Standard I/O
- Memory Mapped I/O

If a standard I/O technique is used, the system can use the feature of both CE and $\overline{CE}$. By using a combination of unused address lines A_{11-15} and the Chip Enable inputs, the system can use up to 5 each 8355's without requiring a CE decoder. See Figure 2a and 2b.

If a memory mapped I/O approach is used the 8355 will be selected by the combination of both the Chip Enables and IO/$\overline{M}$ using the AD_{8-15} address lines. See Figure 1.

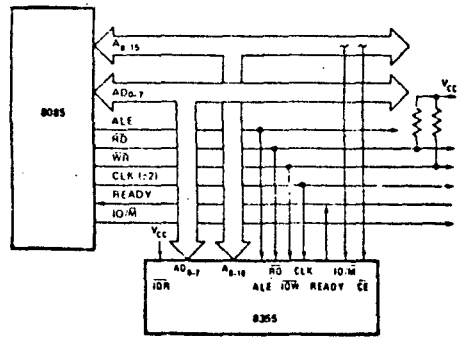


Figure 1. 8355 in 8085A System (Memory-Mapped I/O)

8088 FIVE CHIP SYSTEM

Figure 2a shows a five chip system containing:

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

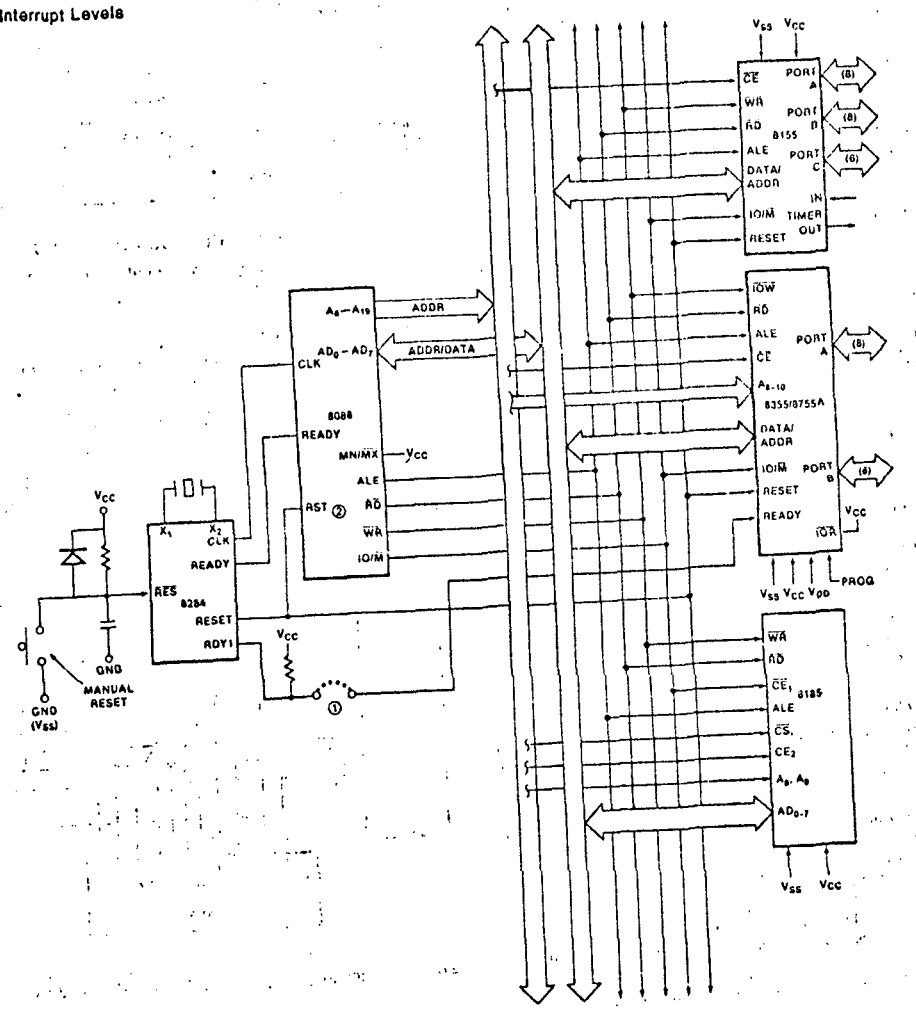
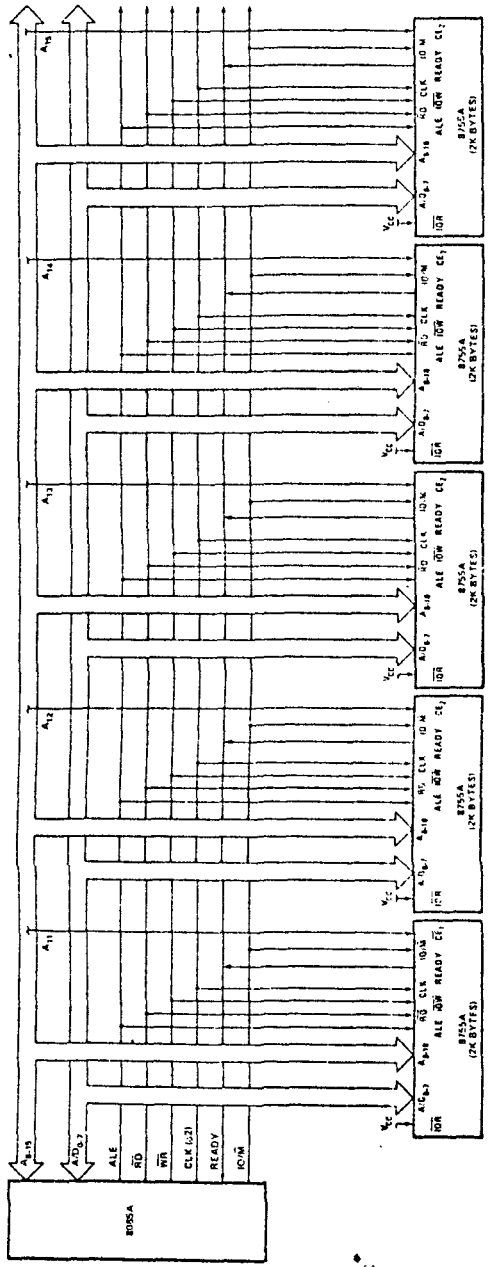


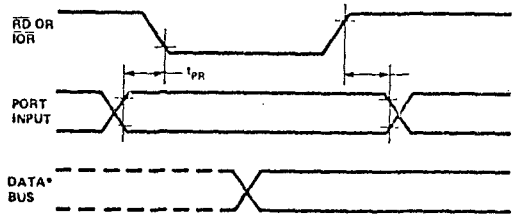
Figure 2a. 8088 Five Chip System Configuration



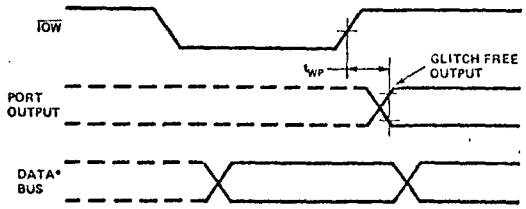
Note: Use $\overline{CE}_1$ for the first 8755A in the system, and CE_2 for the other 8755A's. Permits up to 5-8"55A's in a system without CE decoder.

Figure 2b. 8355 in 8085A System (Standard I/O)

a. Input Mode



b. Output Mode



*DATA BUS TIMING IS SHOWN IN FIGURE 4.

Figure 5. I/O Port Timing

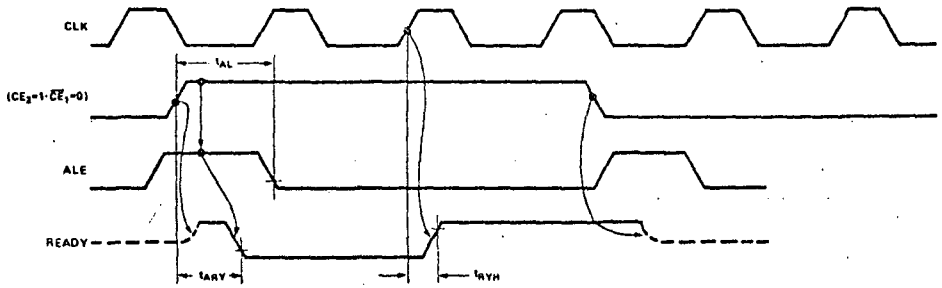


Figure 6. Wait State Timing (Ready = 0)

8755A/8755A-2
16,384-BIT EPROM WITH I/O

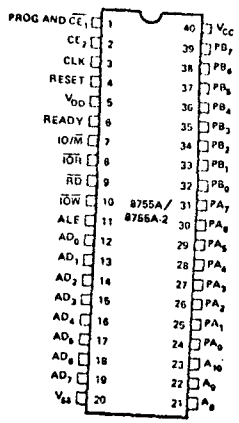
- 2048 Words × 8 Bits
- Single +5V Power Supply (V_{CC})
- Directly Compatible with 8085A and 8088 Microprocessors
- U.V. Erasable and Electrically Reprogrammable
- Internal Address Latch
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- 40-Pin DIP

The Intel® 8755A is an erasable and electrically reprogrammable ROM (EPROM) and I/O chip to be used in the 8085A and 8088 microprocessor systems. The EPROM portion is organized as 2048 words by 8 bits. It has a maximum access time of 450 ns to permit use with no wait states in an 8085A CPU.

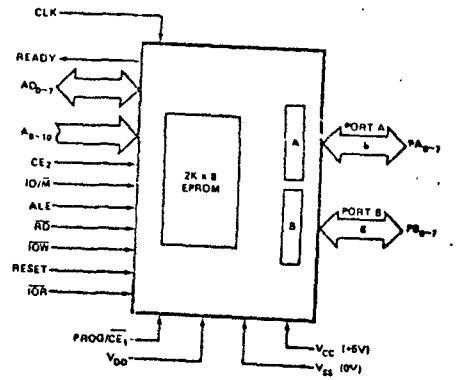
The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines, and each I/O port line is individually programmable as input or output.

The 8755A-2 is a high speed selected version of the 8755A compatible with the 5 MHz 8085A-2 and the full speed 5 MHz 8088.

PIN CONFIGURATION



BLOCK DIAGRAM



8755A FUNCTIONAL PIN DEFINITION

Symbol	Function
ALE (input)	When Address Latch Enable goes high, AD_{0-7}, IO/M, A_{8-10}, CE_2, and $\overline{CE_1}$ enter the address latches. The signals (AD, IO/M, A_{8-10}, CE) are latched in at the trailing edge of ALE.
AD_{0-7} (input/output)	Bidirectional Address/Data bus. The lower 8-bits of the PROM or I/O address are applied to the bus lines when ALE is high. During an I/O cycle, Port A or B are selected based on the latched value of AD_0. If $\overline{RD}$ or $\overline{IOR}$ is low when the latched Chip Enables are active, the output buffers present data on the bus.
A_{8-10} (input)	These are the high order bits of the PROM address. They do not affect I/O operations.
PROG/$\overline{CE_1}$ CE_2 (input)	Chip Enable Inputs: $\overline{CE_1}$ is active low and CE_2 is active high. The 8755A can be accessed only when *BOTH* Chip Enables are active at the time the ALE signal latches them up. If either Chip Enable input is not active, the AD_{0-7} and READY outputs will be in a high impedance state. $\overline{CE_1}$ is also used as a programming pin. (See section on programming.)
IO/$\overline{M}$ (input)	If the latched IO/$\overline{M}$ is high when $\overline{RD}$ is low, the output data comes from an I/O port. If it is low the output data comes from the PROM.
$\overline{RD}$ (input)	If the latched Chip Enables are active when $\overline{RD}$ goes low, the AD_{0-7} output buffers are enabled and output either the selected PROM location or I/O port. When both $\overline{RD}$ and $\overline{IOR}$ are high, the AD_{0-7} output buffers are 3-stated.
$\overline{IOW}$ (input)	If the latched Chip Enables are active, a low on $\overline{IOW}$ causes the output port pointed to by the latched value of AD_0 to be written with the data on AD_{0-7}. The state of IO/$\overline{M}$ is ignored.
CLK (input)	The CLK is used to force the READY into its high impedance state after it has been forced low by $\overline{CE_1}$ low, CE_2 high, and ALE high.

Symbol	Function
READY (output)	READY is a 3-state output controlled by CE_2, $\overline{CE_1}$, ALE and CLK. READY is forced low when the Chip Enables are active during the time ALE is high, and remains low until the rising edge of the next CLK. (See Figure 6.)
PA_{0-7} (input/output)	These are general purpose I/O pins. Their input/output direction is determined by the contents of Data Direction Register (DDR). Port A is selected for write operations when the Chip Enables are active and $\overline{IOW}$ is low and a 0 was previously latched from AD_0, AD_1. Read operation is selected by either $\overline{IOR}$ low and active Chip Enables and AD_0 and AD_1 low, *or* IO/$\overline{M}$ high, $\overline{RD}$ low, active Chip Enables, and AD_0 and AD_1 low.
PB_{0-7} (input/output)	This general purpose I/O port is identical to Port A except that it is selected by a 1 latched from AD_0 and a 0 from AD_1.
RESET (input)	In normal operation, an input high on RESET causes all pins in Ports A and B to assume input mode (clear DDR register).
$\overline{IOR}$ (input)	When the Chip Enables are active, a low on $\overline{IOR}$ will output the selected I/O port onto the AD bus. $\overline{IOR}$ low performs the same function as the combination of IO/$\overline{M}$ high and $\overline{RD}$ low. When $\overline{IOR}$ is not used in a system, $\overline{IOR}$ should be tied to Vcc ("1").
Vcc	+5 volt supply.
Vss	Ground Reference.
V_{DD}	V_{DD} is a programming voltage, and must be tied to +5V when the 8755A is being read. For programming, a high voltage is supplied with V_{DD} = 25V, typical. (See section on programming.)

FUNCTIONAL DESCRIPTION

PROM Section

The 8755A contains an 8-bit address latch which allows it to interface directly to MCS-48 and MCS-85 Microcomputers without additional hardware.

The PROM section of the chip is addressed by the 11-bit address and CE. The address, $\overline{CE_1}$ and CE_2 are latched into the address latches on the falling edge of ALE. If the latched Chip Enables are active and IO/$\overline{M}$ is low when $\overline{RD}$ goes low, the contents of the PROM location addressed by the latched address are put out on the AD_{0-7} lines.

I/O Section

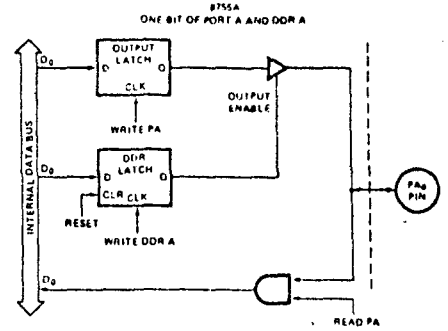
The I/O section of the chip is addressed by the latched value of AD_{0-1}. Two 8-bit Data Direction Registers (DDR) in 8755A determine the input/output status of each pin in the corresponding ports. A "0" in a particular bit position of a DDR signifies that the corresponding I/O port bit is in the input mode. A "1" in a particular bit position signifies that the corresponding I/O port bit is in the output mode. In this manner the I/O ports of the 8755A are bit-by-bit programmable as inputs or outputs. The table summarizes port and DDR designation. DDR's cannot be read.

AD_1	AD_0	Selection
0	0	Port A
0	1	Port B
1	0	Port A Data Direction Register (DDR A)
1	1	Port B Data Direction Register (DDR B)

When $\overline{IOW}$ goes low and the Chip Enables are active, the data on the AD is written into I/O port selected by the latched value of AD_{0-1}. During this operation all I/O bits of the selected port are affected, regardless of their I/O mode and the state of IO/$\overline{M}$. The actual output level does not change until $\overline{IOW}$ returns high. (glitch free output)

A port can be read out when the latched Chip Enables are active and either $\overline{RD}$ goes low with IO/$\overline{M}$ high, or $\overline{IOR}$ goes low. Both input and output mode bits of a selected port will appear on lines AD_{0-7}.

To clarify the function of the I/O Ports and Data Direction Registers, the following diagram shows the configuration of one bit of PORT A and DDR A. The same logic applies to PORT B and DDR B.



WRITE PA = (IOW=0) • (CHIP ENABLES ACTIVE) • (PORT A ADDRESS SELECTED)
WRITE DDR A = (IOW=0) • (CHIP ENABLES ACTIVE) • (DDR A ADDRESS SELECTED)
READ PA = [(IO/M=1) • (RD=0)] + [(IOR=0)] • (CHIP ENABLES ACTIVE) • (PORT A ADDRESS SELECTED)

NOTE: WRITE PA IS NOT QUALIFIED BY IO/M

Note that hardware RESET or writing a zero to the DDR latch will cause the output latch's output buffer to be disabled, preventing the data in the Output Latch from being passed through to the pin. This is equivalent to putting the port in the input mode. Note also that the data can be written to the Output Latch even though the Output Buffer has been disabled. This enables a port to be initialized with a value prior to enabling the output.

The diagram also shows that the contents of PORT A and PORT B can be read even when the ports are configured as outputs.

TABLE 1. 8755A PROGRAMMING MODULE CROSS REFERENCE

MODULE NAME	USE WITH
UPP 955	UPP(4)
UPP UP2(2)	UPP 855
PROMPT 975	PROMPT 80/85(3)
PROMPT 475	PROMPT 48(1)

NOTES:
1. Described on p. 13-34 of 1978 Data Catalog.
2. Special adaptor socket.
3. Described on p. 13-39 of 1978 Data Catalog.
4. Described on p. 13-71 of 1978 Data Catalog.

ERASURE CHARACTERISTICS

The erasure characteristics of the 8755A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8755A in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 8755A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8755 window to prevent unintentional erasure.

The recommended erasure procedure for the 8755A is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15W-sec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000μW/cm² power rating. The 8755A should be placed within one inch from the lamp tubes during erasure. Some lamps have a filter on their tubes and this filter should be removed before erasure.

PROGRAMMING

Initially, and after each erasure, all bits of the EPROM portions of the 8755A are in the "1" state. Information is introduced by selectively programming "0" into the desired bit locations. A programmed "0" can only be changed to a "1" by UV erasure.

The 8755A can be programmed on the Intel® Universal PROM Programmer (UPP), and the PROMPT™ 80/85 and PROMPT-48™ design aids. The appropriate programming modules and adapters for use in programming both 8755A's and 8755's are shown in Table 1.

The program mode itself consists of programming a single address at a time, giving a single 50 msec pulse for every address. Generally, it is desirable to have a verify cycle after a program cycle for the same address as shown in the attached timing diagram. In the verify cycle (i.e., normal memory read cycle) 'VDD' should be at +5V.

Preliminary timing diagrams and parameter values pertaining to the 8755A programming operation are contained in Figure 7.

SYSTEM APPLICATIONS

System Interface with 8085A and 8088

A system using the 8755A can use either one of the two I/O interface techniques:

- Standard I/O
- Memory Mapped I/O

If a standard I/O technique is used, the system can use the feature of both CE_3 and $\overline{CE}_1$. By using a combination of unused address lines A_{11-15} and the Chip Enable inputs, the 8085A system can use up to 5 each 8755A's without requiring a CE decoder. See Figure 2a and 2b.

If a memory mapped I/O approach is used the 8755A will be selected by the combination of both the Chip Enables and IO/M using the AD_{8-15} address lines. See Figure 1.

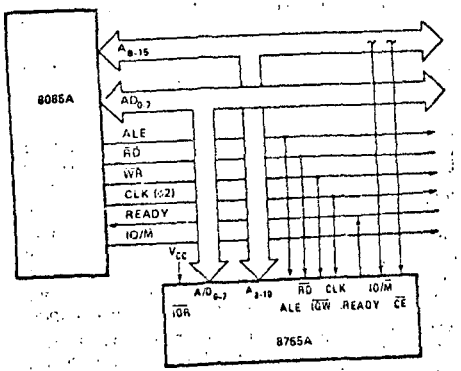


Figure 1. 8755A in 8085A System (Memory-Mapped I/O)

8088 FIVE CHIP SYSTEM

Figure 1b shows a five chip system containing:

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

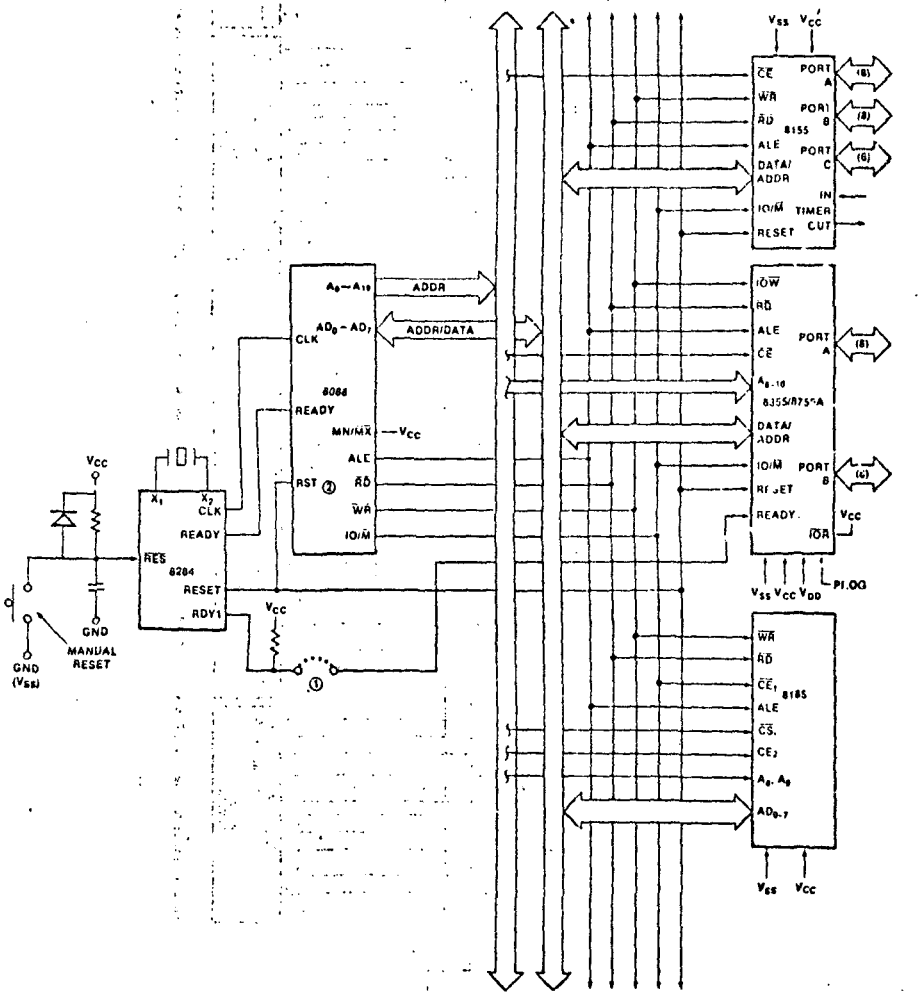


Figure 2a. 8088 Five Chip System Configuration

8216/8226
4-BIT PARALLEL BIDIRECTIONAL BUS DRIVER

- Data Bus Buffer Driver for 8080 CPU
- Low Input Load Current — 0.25 mA Maximum
- High Output Drive Capability for Driving System Bus
- 3.65V Output High Voltage for Direct Interface to 8080 CPU
- 3-State Outputs
- Reduces System Package Count

The 8216/8226 is a 4-bit bidirectional bus driver/receiver. All inputs are low power TTL compatible. For driving MOS, the DO outputs provide a high 3.65V V_{OH}, and for high capacitance terminated bus structures, the DB outputs provide a high 50 mA I_{OL} capability. A non-inverting (8216) and an inverting (8226) are available to meet a wide variety of applications for buffering in microcomputer systems.

*Note: The specifications for the 3216/3226 are identical with those for the 8216/8226.

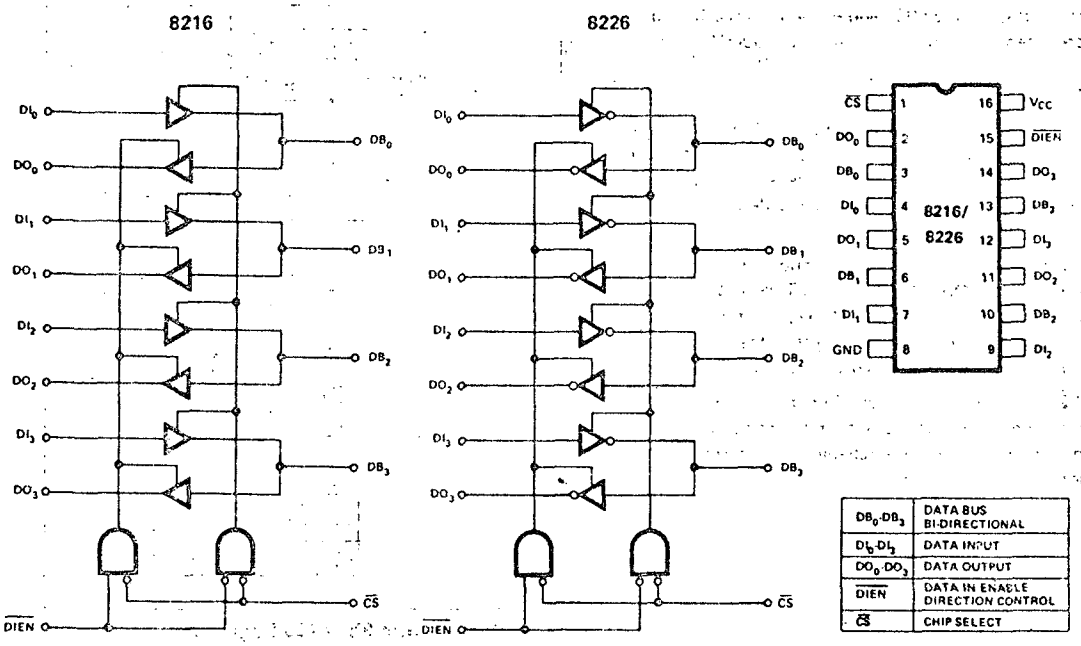


Figure 1. Block Diagrams

Figure 2. Pin Configuration

FUNCTIONAL DESCRIPTION

Microprocessors like the 8080 are MOS devices and are generally capable of driving a single TTL load. The same is true for MOS memory devices. While this type of drive is sufficient in small systems with few components, quite often it is necessary to buffer the microprocessor and memories when adding components or expanding to a multi-board system.

The 8216/8226 is a four bit bi-directional bus driver specifically designed to buffer microcomputer system components.

Bidirectional Driver

Each buffered line of the four bit driver consists of two separate buffers that are tri-state in nature to achieve direct bus interface and bi-directional capability. On one side of the driver the output of one buffer and the input of another are tied together (DB), this side is used to interface to the system side components such as memories, I/O, etc., because its interface is direct TTL compatible and it has high drive (50mA). On the other side of the driver the inputs and outputs are separated to provide maximum flexibility. Of course, they can be tied together so that the driver can be used to buffer a true bi-directional bus such as the 8080 Data Bus. The DO outputs on this side of the driver have a special high voltage output drive capability (3.65V) so that direct interface to the 8080 and 8008 CPUs is achieved with an adequate amount of noise immunity (350mV worst case).

Control Gating $\overline{DIEN}$, $\overline{CS}$

The $\overline{CS}$ input is actually a device select. When it is "high" the output drivers are all forced to their high-impedance state. When it is at "zero" the device is selected (enabled) and the direction of the data flow is determined by the $\overline{DIEN}$ input.

The $\overline{DIEN}$ input controls the direction of data flow (see Figure 3) for complete truth table. This direction control is accomplished by forcing one of the pair of buffers into its high impedance state and allowing the other to transmit its data. A simple two gate circuit is used for this function.

The 8216/8226 is a device that will reduce component count in microcomputer systems and at the same time enhance noise immunity to assure reliable, high performance operation.

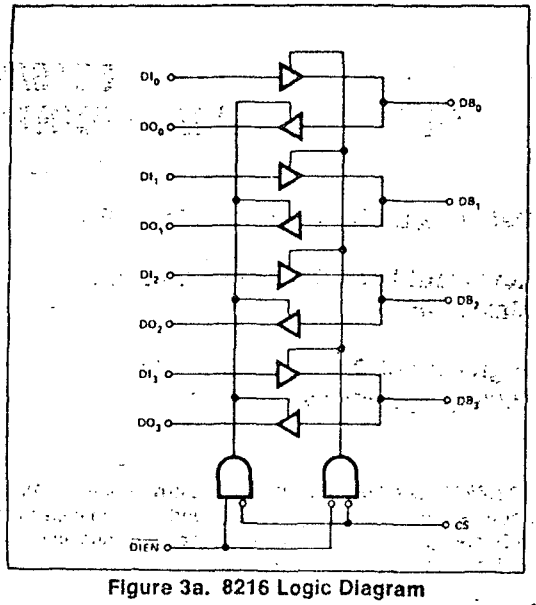


Figure 3a. 8216 Logic Diagram

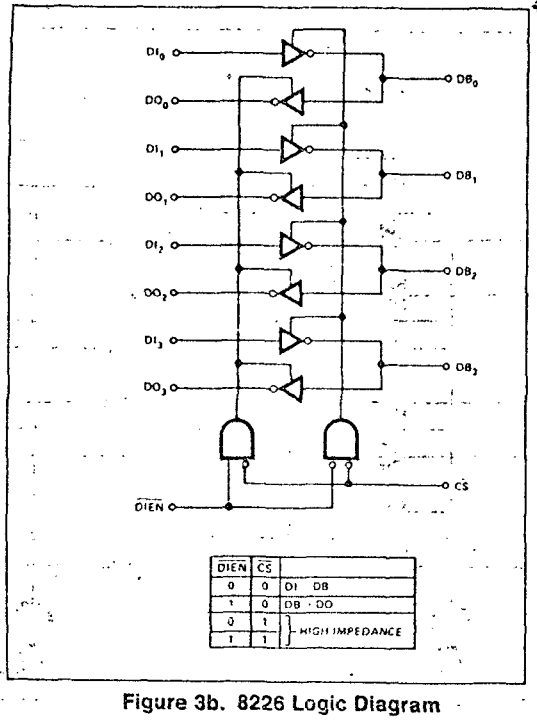


Figure 3b. 8226 Logic Diagram

8279/8279-5
PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- MCS-85™ Compatible 8279-5
- Simultaneous Keyboard Display Operations
- Scanned Keyboard Mode
- Scanned Sensor Mode
- Strobed Input Entry Mode
- 8-Character Keyboard FIFO
- 2-Key Lockout or N-Key Rollover with Contact Debounce
- Dual 8- or 16-Numerical Display
- Single 16-Character Display
- Right or Left Entry 16-Byte Display RAM
- Mode Programmable from CPU
- Programmable Scan Timing
- Interrupt Output on Key Entry

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16X8 display RAM which can be organized into dual 16X4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

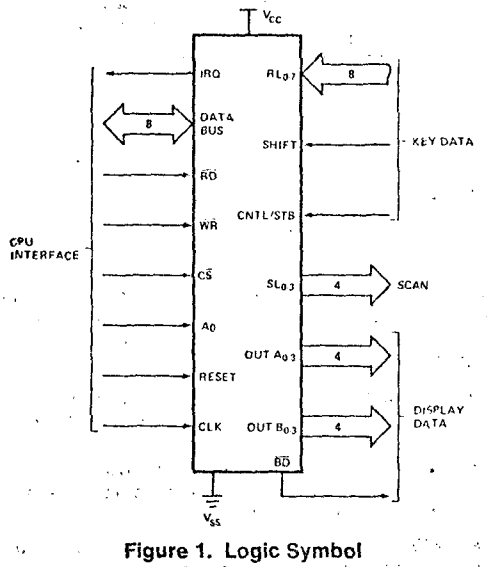


Figure 1. Logic Symbol

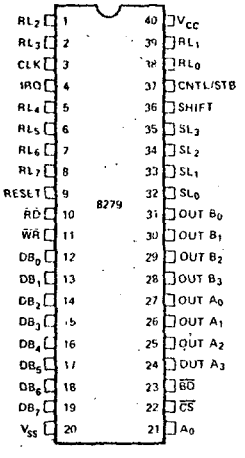


Figure 2. Pin Configuration

HARDWARE DESCRIPTION

The 8279 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Table 1. Pin Descriptions

Symbol	Pin No.	Name and Function
DB_0-DB_7	8	**Bi-directional data bus:** All data and commands between the CPU and the 8279 are transmitted on these lines.
CLK	1	**Clock:** Clock from system used to *generate internal timing.*
RESET	1	**Reset:** A high signal on this pin resets the 8279. After being reset the 8279 is placed in the following mode: 1) 16 8-bit character display —left entry. 2) Encoded scan keyboard—2 key lockout. Along with this the program clock prescaler is set to 31.
CS	1	**Chip Select:** A low on this pin enables the interface functions to receive or transmit.
A_0	1	**Buffer Address:** A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
$\overline{RD}$, $\overline{WR}$	2	**Input/Output Read and Write:** These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
IRQ	1	**Interrupt Request:** In a keyboard mode, the interrupt line is high when there is data in the FIFO/Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM. In a sensor mode, the interrupt line goes high whenever a change in a sensor is detected.
V_{SS}, V_{CC}	2	**Ground and power supply pins.**
SL_0-SL_3	4	**Scan Lines:** Scan lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
RL_0-RL_7	8	**Return Line:** Return line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pullups to keep them high until a switch closure pulls one low. They also serve as an 8-bit input in the Strobed Input mode.
SHIFT	1	**Shift:** The shift input status is stored along with the key position on key closure in the Scanned Keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low.
CNTL/STB	1	**Control/Strobed Input Mode:** For keyboard modes this line is used as a control input and stored like status on a key closure. The line is also the strobe line that enters the data into the FIFO in the Strobed Input mode. (Rising Edge). It has an active internal pullup to keep it high until a switch closure pulls it low.
OUT A_0-OUT A_3 OUT B_0-OUT B_3	4 4	**Outputs:** These two ports are the outputs for the 16 x 4 display refresh registers. The data from these outputs is synchronized to the scan lines (SL_0-SL_3) for multiplexed digit displays. The two 4 bit ports may be blanked independently. These two ports may also be considered as one 8-bit port.
$\overline{BD}$	1	**Blank Display:** This output is used to blank the display during digit switching or by a display blanking command.

FUNCTIONAL DESCRIPTION

Since data input and display are an integral part of many microprocessor designs, the system designer needs an interface that can control these functions without placing a large load on the CPU. The 8279 provides this function for 8-bit microprocessors.

The 8279 has two sections: keyboard and display. The keyboard section can interface to regular typewriter style keyboards or random toggle or thumb switches. The display section drives alphanumeric displays or a bank of indicator lights. Thus the CPU is relieved from scanning the keyboard or refreshing the display.

The 8279 is designed to directly connect to the microprocessor bus. The CPU can program all operating modes for the 8279. These modes include:

Input Modes

- Scanned Keyboard — with encoded (8 x 8 key keyboard) or decoded (4 x 8 key keyboard) scan lines. A key depression generates a 6-bit encoding of key position. Position and shift and control status are stored in the FIFO. Keys are automatically debounced with 2-key lockout or N-key rollover.
- Scanned Sensor Matrix — with encoded (8 x 8 matrix switches) or decoded (4 x 8 matrix switches) scan lines. Key status (open or closed) stored in RAM addressable by CPU.
- Strobed Input — Data on return lines during control line strobe is transferred to FIFO.

Output Modes

- 8 or 16 character multiplexed displays that can be organized as dual 4-bit or single 8-bit ($B_0 = D_0$, $A_3 = D_7$).
- Right entry or left entry display formats.

Other features of the 8279 include:

- Mode programming from the CPU.
- Clock Prescaler
- Interrupt output to signal CPU when there is keyboard or sensor data available.
- An 8 byte FIFO to store keyboard information.
- 16 byte internal Display RAM for display refresh. This RAM can also be read by the CPU.

PRINCIPLES OF OPERATION

The following is a description of the major elements of the 8279 Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 3.

I/O Control and Data Buffers

The I/O control section uses the $\overline{CS}$, A_0, $\overline{RD}$ and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers. All data flow to and from the 8279 is enabled by $\overline{CS}$. The character of the information, given or desired by the CPU, is identified by A_0. A logic one means the information is a command or status. A logic zero means the information is data. $\overline{RD}$ and $\overline{WR}$ determine the direction of data flow through the Data Buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ($\overline{CS}$ = 1), the devices are in a high impedance state. The drivers input during $\overline{WR} \bullet \overline{CS}$ and output during $\overline{RD} \bullet \overline{CS}$.

Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by presenting the proper command on the data lines with $A_0 = 1$ and then sending a $\overline{WR}$. The command is latched on the rising edge of $\overline{WR}$.

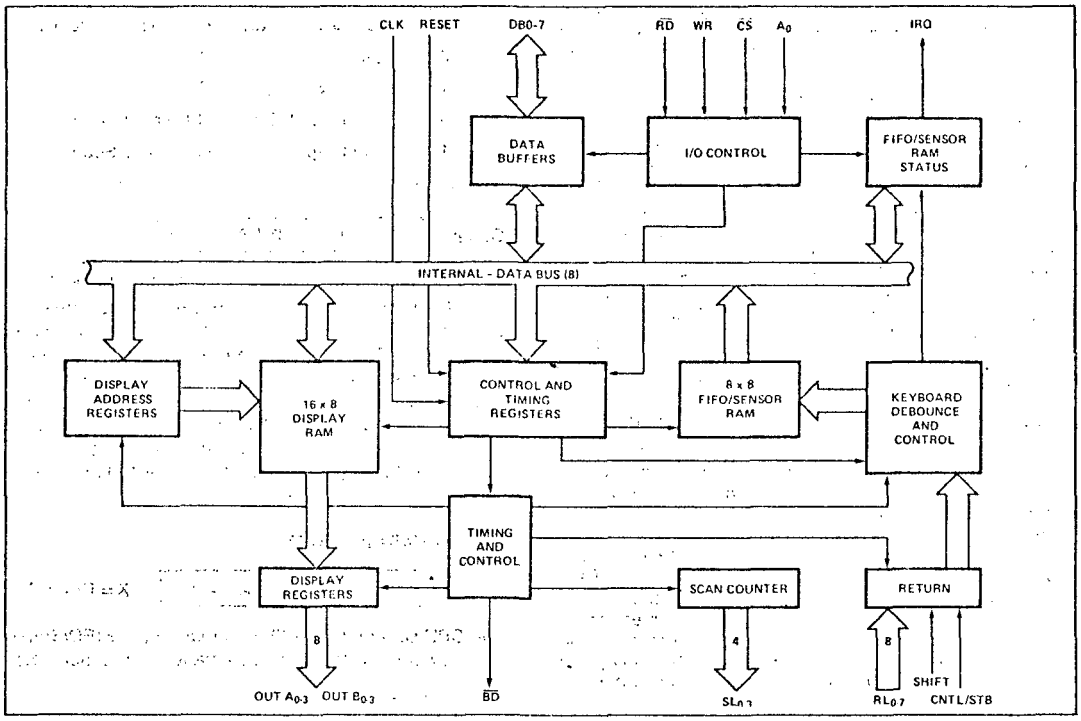


Figure 3. Internal Block Diagram

AFN-00742B

The command is then decoded and the appropriate function is set. The timing control contains the basic timing counter chain. The first counter is a ÷ N prescaler that can be programmed to yield an internal frequency of 100 kHz which gives a 5.1 ms keyboard scan time and a 10.3 ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan times.

Scan Counter

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant 2 bits and provides a decoded 1 of 4 scan. Note than when the keyboard is in decoded scan, so is the display. This means that only the first 4 characters in the Display RAM are displayed.

In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.

Return Buffers and Keyboard Debounce and Control

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a closed switch, it waits about 10 msec to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL are transferred to the FIFO. In the scanned Sensor Matrix modes, the contents of the return lines is directly transferred to the corresponding row of the Sensor RAM (FIFO) each key scan time. In Strobed Input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse.

FIFO/Sensor RAM and Status

This block is a dual function 8 x 8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by an $\overline{RD}$ with $\overline{CS}$ low and A_0 high. The status logic also provides an IRQ signal when the FIFO is not empty. In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected.

Display Address Registers and Display RAM

The Display Address Registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The Display RAM can be directly read by the CPU after the correct mode and address is set. The addresses for the A and B nibbles are automatically updated by the 8279 to match data entry by the CPU. The A and B nibbles can be entered independently or as one word, according to the mode that is set by the CPU. Data entry to the display can be set to either left or right entry. See Interface Considerations for details.

SOFTWARE OPERATION

8279 commands

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with $\overline{CS}$ low and A_0 high and are loaded to the 8279 on the rising edge of $\overline{WR}$.

Keyboard/Display Mode Set

	MSB							LSB
Code:	0	0	0	D	D	K	K	K

Where DD is the Display Mode and KKK is the Keyboard Mode.

DD

0	0	8 8-bit character display — Left entry
0	1	16 8-bit character display — Left entry*
1	0	8 8-bit character display — Right entry
1	1	16 8-bit character display — Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

KKK

0	0	0	Encoded Scan Keyboard — 2 Key Lockout*
0	0	1	Decoded Scan Keyboard — 2-Key Lockout
0	1	0	Encoded Scan Keyboard — N-Key Rollover
0	1	1	Decoded Scan Keyboard — N-Key Rollover
1	0	0	Encoded Scan Sensor Matrix
1	0	1	Decoded Scan Sensor Matrix
1	1	0	Strobed Input, Encoded Display Scan
1	1	1	Strobed Input, Decoded Display Scan

Program Clock

Code:	0	0	1	P	P	P	P	P

All timing and multiplexing signals for the 8279 are generated by an internal prescaler. This prescaler divides the external clock (pin 3) by a programmable integer. Bits PPPPP determine the value of this integer which ranges from 2 to 31. Choosing a divisor that yields 100 kHz will give the specified scan and debounce times. For instance, if Pin 3 of the 8279 is being clocked by a 2 MHz signal, PPPPP should be set to 10100 to divide the clock by 20 to yield the proper 100 kHz operating frequency.

Read FIFO/Sensor RAM

Code:	0	1	0	AI	X	A	A	A	X = Don't Care

The CPU sets up the 8279 for a read of the FIFO/Sensor RAM by first writing this command. In the Scan Key-

*Default after reset.

board Mode, the Auto-Increment flag (AI) and the RAM address bits (AAA) are irrelevant. The 8279 will automatically drive the data bus for each subsequent read ($A_0 = 0$) in the same sequence in which the data first entered the FIFO. All subsequent reads will be from the FIFO until another command is issued.

In the Sensor Matrix Mode, the RAM address bits AAA select one of the 8 rows of the Sensor RAM. If the AI flag is set (AI = 1), each successive read will be from the subsequent row of the sensor RAM.

Read Display RAM

Code:

0	1	1	AI	A	A	A	A

The CPU sets up the 8279 for a read of the Display RAM by first writing this command. The address bits AAAA select one of the 16 rows of the Display RAM. If the AI flag is set (AI = 1), this row address will be incremented after each following read *or write* to the Display RAM. Since the same counter is used for both reading and writing, this command sets the next read *or write* address and the sense of the Auto-Increment mode for both operations.

Write Display RAM

Code:

1	0	0	AI	A	A	A	A

The CPU sets up the 8279 for a write to the Display RAM by first writing this command. After writing the command with $A_0 = 1$, all subsequent writes with $A_0 = 0$ will be to the Display RAM. The addressing and Auto-Increment functions are identical to those for the Read Display RAM. However, this command does not affect the source of subsequent Data Reads; the CPU will read from whichever RAM (Display or FIFO/Sensor) which was last specified. If, indeed, the Display RAM was last specified, the Write Display RAM will, nevertheless, change the next Read location.

Display Write Inhibit/Blanking

Code:

				A	B	A	B
1	0	1	X	IW	IW	BL	BL

The IW Bits can be used to mask nibble A and nibble B in applications requiring separate 4-bit display ports. By setting the IW flag (IW = 1) for one of the ports, the port becomes marked so that entries to the Display RAM from the CPU do not affect that port. Thus, if each nibble is input to a BCD decoder, the CPU may write a digit to the Display RAM without affecting the other digit being displayed. It is important to note that bit B_0 corresponds to bit D_0 on the CPU bus, and that bit A_3 corresponds to bit D_7.

If the user wishes to blank the display, the BL flags are available for each nibble. The last Clear command issued determines the code to be used as a "blank." This code defaults to all zeros after a reset. Note that both BL flags must be set to blank a display formatted with a single 8-bit port.

Clear

Code:

1	1	0	C_D	C_D	C_D	C_F	C_A

The C_D bits are available in this command to clear all rows of the Display RAM to a selectable blanking code as follows:

C_D	C_D	C_D	
0	X		All Zeros (X = Don't Care)
1	0		AB = Hex 20 (0010 0000)
1	1		All Ones

Enable clear display when = 1 (or by C_A = 1)

During the time the Display RAM is being cleared (~160 μs), it may not be written to. The most significant bit of the FIFO status word is set during this time. When the Display RAM becomes available again, it automatically resets.

If the C_F bit is asserted ($C_F = 1$), the FIFO status is cleared and the interrupt output line is reset. Also, the Sensor RAM pointer is set to row 0.

C_A, the Clear All bit, has the combined effect of C_D and C_F; it uses the C_D clearing code on the Display RAM and also clears FIFO status. Furthermore, it resynchronizes the internal timing chain.

End Interrupt/Error Mode Set

Code:

1	1	1	E	X	X	X	X

X = Don't care.

For the sensor matrix modes this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset).

For the N-key rollover mode — if the E bit is programmed to "1" the chip will operate in the special Error mode. (For further details, see Interface Considerations Section.)

Status Word

The status word contains the FIFO status, error, and display unavailable signals. This word is read by the CPU when A_0 is high and $\overline{CS}$ and $\overline{RD}$ are low. See Interface Considerations for more detail on status word.

Data Read

Data is read when A_0, $\overline{CS}$ and $\overline{RD}$ are all low. The source of the data is specified by the Read FIFO or Read Display commands. The trailing edge of $\overline{RD}$ will cause the address of the RAM being read to be incremented if the Auto-Increment flag is set. FIFO reads always increment (if no error occurs) independent of AI.

Data Write

Data that is written with A_0, $\overline{CS}$ and $\overline{WR}$ low is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command. Auto-Incrementing on the rising edge of $\overline{WR}$ occurs if AI set by the latest display command.

INTERFACE CONSIDERATIONS

Scanned Keyboard Mode, 2-Key Lockout

There are three possible combinations of conditions that can occur during debounce scanning. When a key is depressed, the debounce logic is set. Other depressed keys are looked for during the next two scans. If none are encountered, it is a single key depression and the key position is entered into the FIFO along with the status of CNTL and SHIFT lines. If the FIFO was empty, IRQ will be set to signal the CPU that there is an entry in the FIFO. If the FIFO was full, the key will not be entered and the error flag will be set. If another closed switch is encountered, no entry to the FIFO can occur. If all other keys are released before this one, then it will be entered to the FIFO. If this key is released before any other, it will be entirely ignored. A key is entered to the FIFO only once per depression, no matter how many keys were pressed along with it or in what order they were released. If two keys are depressed within the debounce cycle, it is a simultaneous depression. Neither key will be recognized until one key remains depressed alone. The last key will be treated as a single key depression.

Scanned Keyboard Mode, N-Key Rollover

With N-key Rollover each key depression is treated independently from all others. When a key is depressed, the debounce circuit waits 2 keyboard scans and then checks to see if the key is still down. If it is, the key is entered into the FIFO. Any number of keys can be depressed and another can be recognized and entered into the FIFO. If a simultaneous depression occurs, the keys are recognized and entered according to the order the keyboard scan found them.

Scanned Keyboard — Special Error Modes

For N-key rollover mode the user can program a special error mode. This is done by the "End Interrupt/Error Mode Set" command. The debounce cycle and key-validity check are as in normal N-key mode. If during a single debounce cycle, two keys are found depressed, this is considered a simultaneous multiple depression, and sets an error flag. This flag will prevent any further writing into the FIFO and will set interrupt (if not yet set). The error flag could be read in this mode by reading the FIFO STATUS word. (See "FIFO STATUS" for further details.) The error flag is reset by sending the normal CLEAR command with $C_F = 1$.

Sensor Matrix Mode

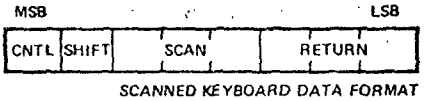
In Sensor Matrix mode, the debounce logic is inhibited. The status of the sensor switch is inputted directly to the Sensor RAM. In this way the Sensor RAM keeps an image of the state of the switches in the sensor matrix. Although debouncing is not provided, this mode has the advantage that the CPU knows how long the sensor was closed and when it was released. A keyboard mode can only indicate a validated closure. To make the software easier, the designer should functionally group the sensors by row since this is the format in which the CPU will read them.

The IRQ line goes high if any sensor value change is detected at the end of a sensor matrix scan. The IRQ line is cleared by the first data read operation if the Auto-Increment flag is set to zero, or by the End Interrupt command if the Auto-Increment flag is set to one.

Note: Multiple changes in the matrix Addressed by ($SL_{0-3} = 0$) may cause multiple interrupts. ($SL_0 = 0$ in the Decoded Mode). Reset may cause the 8279 to see multiple changes.

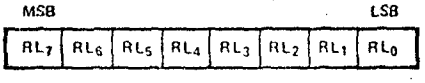
Data Format

In the Scanned Keyboard mode, the character entered into the FIFO corresponds to the position of the switch in the keyboard plus the status of the CNTL and SHIFT lines (non-inverted). CNTL is the MSB of the character and SHIFT is the next most significant bit. The next three bits are from the scan counter and indicate the row the key was found in. The last three bits are from the column counter and indicate to which return line the key was connected.

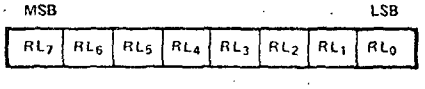


SCANNED KEYBOARD DATA FORMAT

In Sensor Matrix mode, the data on the return lines is entered directly in the row of the Sensor RAM that corresponds to the row in the matrix being scanned. Therefore, each switch postion maps directly to a Sensor RAM position. The SHIFT and CNTL inputs are ignored in this mode. Note that switches are not necessarily the only thing that can be connected to the return lines in this mode. Any logic that can be triggered by the scan lines can enter data to the return line inputs. Eight multiplexed input ports could be tied to the return lines and scanned by the 8279.



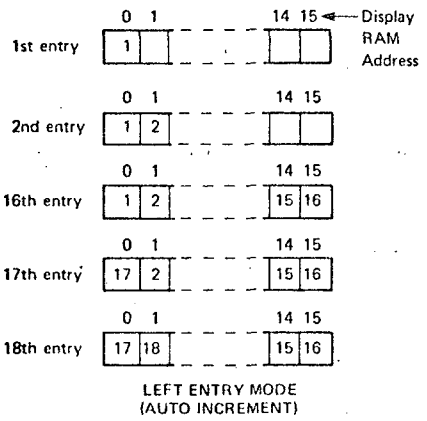
In Strobed Input mode, the data is also entered to the FIFO from the return lines. The data is entered by the rising edge of a CNTL/STB line pulse. Data can come from another encoded keyboard or simple switch matrix. The return lines can also be used as a general purpose strobed input.



Display

Left Entry

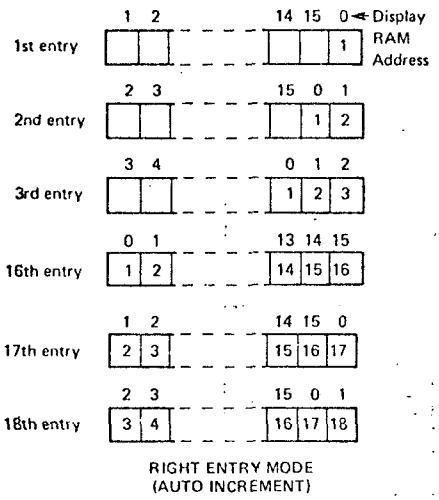
Left Entry mode is the simplest display format in that each display position directly corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 (or address 7 in 8 character display) is the right most display character. Entering characters from position zero causes the display to fill from the left. The 17th (9th) character is entered back in the left most position and filling again proceeds from there.



LEFT ENTRY MODE
(AUTO INCREMENT)

Right Entry

Right entry is the method used by most electronic calculators. The first entry is placed in the right most display character. The next entry is also placed in the right most character after the display is shifted left one character. The left most character is shifted off the end and is lost.

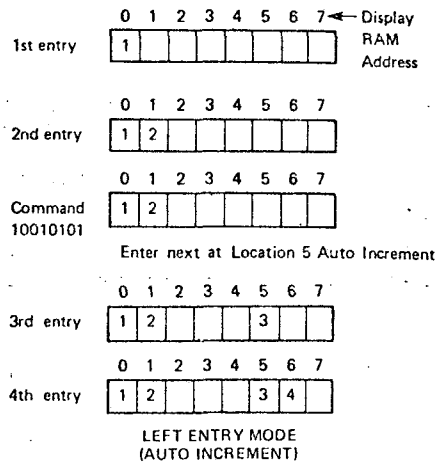


RIGHT ENTRY MODE
(AUTO INCREMENT)

Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended.

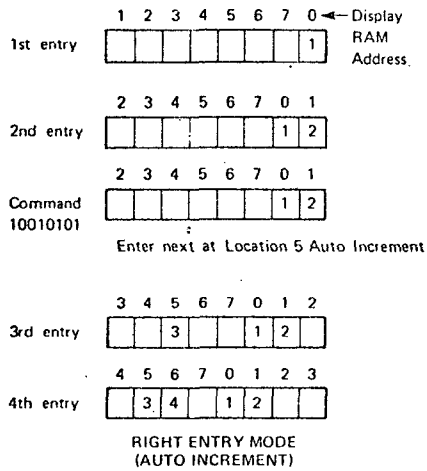
Auto Increment

In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Auto Increment mode has no undesirable side effects and the result is predictable:



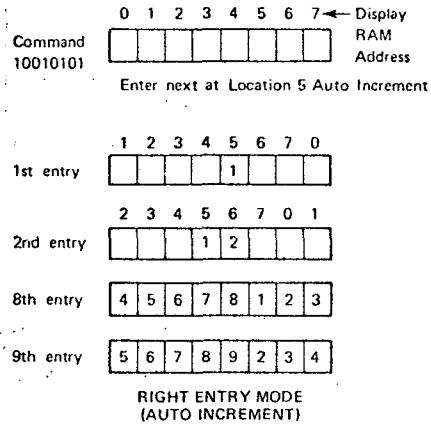
LEFT ENTRY MODE
(AUTO INCREMENT)

In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except if the address sequence is interrupted:



RIGHT ENTRY MODE
(AUTO INCREMENT)

Starting at an arbitrary location operates as shown below:



RIGHT ENTRY MODE
(AUTO INCREMENT)

Entry appears to be from the initial entry point.

8/16 Character Display Formats

If the display mode is set to an 8 character display, the on duty-cycle is double what it would be for a 16 character display (e.g., 5.1 ms scan time for 8 characters vs. 10.3 ms for 16 characters with 100 kHz internal frequency).

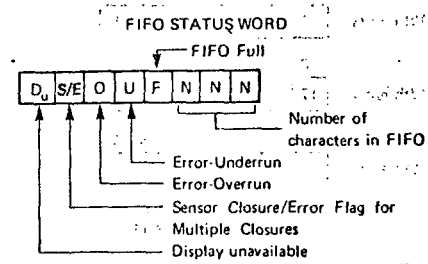
G. FIFO Status

FIFO status is used in the Keyboard and Strobed Input modes to indicate the number of characters in the FIFO and to indicate whether an error has occurred. There are two types of errors possible: overrun and underrun. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO.

The FIFO status word also has a bit to indicate that the Display RAM was unavailable because a Clear Display or Clear All command had not completed its clearing operation.

In a Sensor Matrix mode, a bit is set in the FIFO status word to indicate that at least one sensor closure indication is contained in the Sensor RAM.

In Special Error Mode the S/E bit is showing the error flag and serves as an indication to whether a simultaneous multiple closure error has occurred.



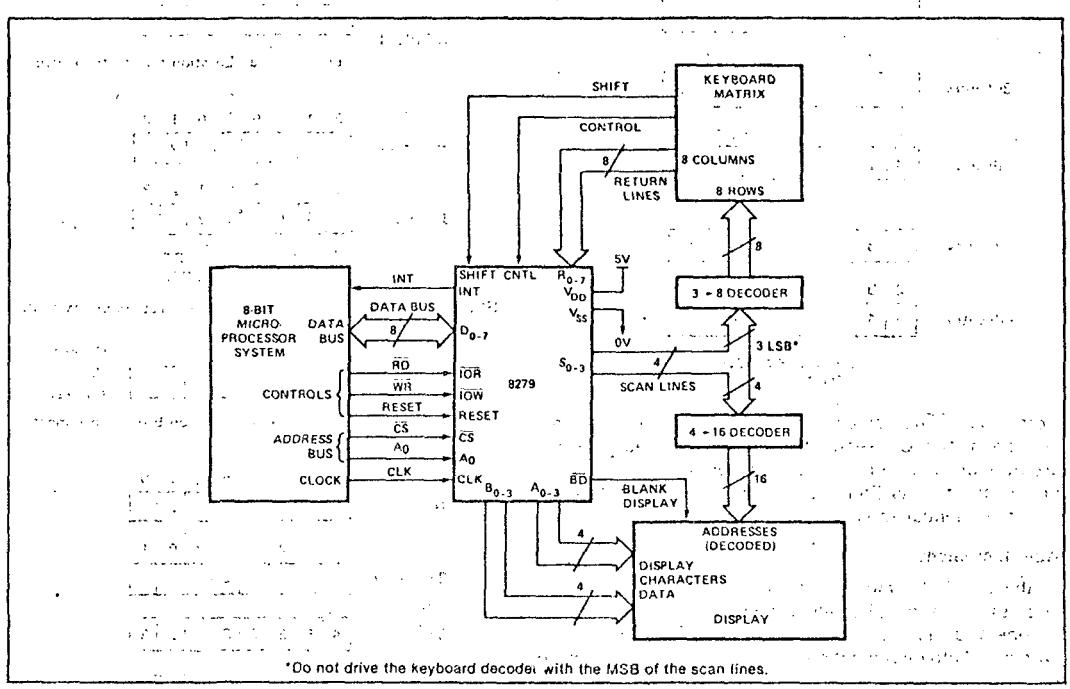


Figure 4. System Block Diagram

ANEXO 2

El programa monitor del SDK'85 en lenguaje ensamblador.

```
ASM80 :F1:SDK85.SRC MOD85 MACROFILE XREF PRINT(:F3:SDK85.LST)

ISIS-II 8080/8085 MACRO ASSEMBLER, X108          SDK85    PAGE    1

  LOC  OBJ          SEQ          SOURCE STATEMENT

                      1 ;*****************************************************************************
                      2 ;
                      3 ;                           PROGRAM: SDK-85 MONITOR    VER 1.2
                      4 ;
                      5 ;                           COPYRIGHT (C) 1977
                      6 ;                           INTEL CORPORATION
                      7 ;                           3065 BOWERS AVENUE
                      8 ;                           SANTA CLARA, CALIFORNIA  95051
                      9 ;
                     10 ;*****************************************************************************
                     11 ;
                     12 ; ABSTRACT
                     13 ; ========
                     14 ;
                     15 ; THIS PROGRAM IS A SMALL MONITOR FOR THE INTEL 8085 KIT AND
                     16 ; PROVIDES A MINIMUM LEVEL OF UTILITY FUNCTIONS FOR THE USER EMPLOYING
                     17 ; EITHER AN INTER-ACTIVE CONSOLE (I.E. TELETYPE) OR THE KIT'S
                     18 ; KEYBOARD/LED DISPLAY. THE KEYBOARD MONITOR ALLOWS THE USER TO PERFORM
                     19 ; SUCH FUNCTIONS AS MEMORY AND REGISTER MANIPULATION, PROGRAM LOADING,
                     20 ; PROGRAM EXECUTION, INTERRUPTION OF AN EXECUTING PROGRAM, AND
                     21 ; SYSTEM RESET.
                     22 ;
                     23 ; PROGRAM ORGANIZATION
                     24 ; ======= ============
                     25 ;
                     26 ; THE PROGRAM IS ORGANIZED AS FOLLOWS :-
                     27 ;       1) COLD START ROUTINE (RESET)
                     28 ;       2) WARM START - REGISTER SAVE ROUTINE
                     29 ;       3) INTERRUPT VECTORS
                     30 ;       4) KEYBOARD MONITOR
                     31 ;       5) TTY MONITOR
                     32 ;       6) LAYOUT OF RAM USAGE
                     33 ;
                     34 ; THE KEYBOARD MONITOR BEGINS WITH THE COMMAND RECOGNIZER, FOLLOWED BY
                     35 ; THE COMMAND ROUTINE SECTION, UTILITY ROUTINE SECTION AND MONITOR
                     36 ; TABLES. THE COMMAND AND UTILITY ROUTINES ARE IN ALPHABETICAL ORDER
                     37 ; WITHIN THEIR RESPECTIVE SECTIONS.
                     38 ; THROUGHOUT THE KEYBOARD MONITOR, A COMMENT FIELD BEGINNING
                     39 ; WITH "ARG - " INDICATES A STATEMENT WHICH LOADS A VALUE INTO
                     40 ; A REGISTER AS AN ARGUMENT FOR A FUNCTION. WHEN THE DESIRED VALUE
                     41 ; LIST OF KEYBOARD MONITOR ROUTINES
                     42 ; ==== == ======== ======= ========
                     43 ;
                     44 ; CMMND
                     45 ; -----
                     46 ; EXAM
                     47 ; GOCMD
                     48 ; SSTEP
                     49 ; SUBST
                     50 ; -----
                     51 ; CLEAR
                     52 ; CLDIS
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                    53 ; CLDST
                    54 ; DISPC
                    55 ; ERR
                    56 ; GTHEX
                    57 ; HXDSP
                    58 ; ININT
                    59 ; INSDG
                    60 ; NXTRG
                    61 ; OUTPT
                    62 ; RDKBD
                    63 ; RETF
                    64 ; RETT
                    65 ; RGLOC
                    66 ; RSTOR
                    67 ; SETRG
                    68 ; UPDAD
                    69 ; UPDDT
                    70 ;
                    71          NAME    SDK85
                    72 ;
                    73 ;*****************************************************************
                    74 ;
                    75 ;                       SET CONDITIONAL ASSEMBLY FLAG
                    76 ;
                    77 ;*****************************************************************
                    78 ;
                    79 ;
0000                80 WAITS    SET     0       ;0=NO WAIT STATES
                    81                          ;1=A WAIT STATE IS GENERATED FOR EVERY M CYCLE
                    82                          ;THE APPROPRIATE DELAY TIME MUST BE USED FOR
                    83                                  ;TTY DELAY OR SET UP SINGLE
                    84                                  ;STEP TIMER FOR EACH CASE
                    85 ;
                    86 ;
                    87 ;*****************************************************************
                    88 ;
                    89 ;                       MONITOR EQUATES
                    90 ;
                    91 ;*****************************************************************
                    92 ;
2000                93 RAMST    EQU     2000H   ; START ADDRESS OF RAM - THIS PROGRAM ASSUMES
                    94 ; THAT 256 BYTES OF RANDOM ACCESS MEMORY BEGIN AT THIS ADDRESS.
                    95 ; THE PROGRAM USES STORAGE AT THE END OF THIS SPACE FOR VARIABLES,
                    96 ; SAVING REGISTERS AND THE PROGRAM STACK
                    97 ;
0017                98 RAMUSE   EQU     23      ; RAM USAGE - CURRENTLY, 23 BYTES ARE USED FOR
                    99                          ; /SAVING REGISTERS AND VARIABLES
                   100 ;
0012               101 SKLN     EQU     18      ; MONITOR STACK USAGE - MAX OF 9 LEVELS
                   102 ;
000F               103 UBRLN    EQU     15      ; 5 USER BRANCHES - 3 BYTES EACH
                   104 ;
0000               105 ADFLD    EQU     0       ; INDICATES USE OF ADDRESS FIELD OF DISPLAY
0090               106 ADISP    EQU     90H     ; CONTROL CHARACTER TO INDICATE OUTPUT TO
                   107                          ; /ADDRESS FIELD OF DISPLAY
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

1900                108 CNTRL   EQU     1900H   ; ADDRESS FOR SENDING CONTROL CHARACTERS TO
                    109                         ; /DISPLAY CHIP
0011                110 COMMA   EQU     11H     ; COMMA FROM KEYBOARD
0000                111 CSNIT   EQU     0       ; INITIAL VALUE FOR COMMAND STATUS REGISTER
0020                112 CSR     EQU     20H     ; OUTPUT PORT FOR COMMAND STATUS REGISTER
0094                113 DDISP   EQU     94H     ; CONTROL CHARACTER TO INDICATE OUTPUT TO
                    114                         ; /DATA FIELD OF DISPLAY
0001                115 DOT     EQU     1       ; INDICATOR FOR DOT IN DISPLAY
1800                116 DSPLY   EQU     1800H   ; ADDRESS FOR SENDING CHARACTERS TO DISPLAY
0001                117 DTFLD   EQU     1       ; INDICATES USE OF DATA FIELD OF DISPLAY
0008                118 DTMSK   EQU     08H     ; MASK FOR TURNING ON DOT IN DISPLAY
0080                119 EMPTY   EQU     80H     ; HIGH ORDER 1 INDICATES EMPTY INPUT BUFFER
00CC                120 KBNIT   EQU     0CCH    ; CONTROL CHARACTER TO SET DISPLAY OUTPUT TO
                    121                         ; /ALL ONES DURING BLANKING PERIOD
0000                122 KMODE   EQU     0       ; CONTROL CHAR. TO SET KEYBOARD/DISPLAY MODE
                    123                         ; (2 KEY ROLLOVER, 8 CHARACTER LEFT ENTRY)
20E9                124 MNSTK   EQU     RAMST + 256 - RMUSE     ;START OF MONITOR STACK
0000                125 NODOT   EQU     0       ; INDICATOR FOR NO DOT IN DISPLAY
                    126 ;NUMC - DEFINED LATER   ; NUMBER OF COMMANDS
                    127 ;NUMRG - DEFINED LATER  ; NUMBER OF REGISTER SAVE LOCATIONS
0010                128 PERIO   EQU     10H     ; PERIOD FROM KEYBOARD
00FB                129 PRMPT   EQU     0FBH    ; PROMPT CHARACTER FOR DISPLAY (DASH)
0040                130 READ    EQU     40H     ; CONTROL CHARACTER TO INDICATE INPUT FROM
                    131                         ; /KEYBOARD
0025                132 TIMHI   EQU     25H     ; OUTPUT PORT FOR HIGH ORDER BYTE OF TIMER VALUE
0024                133 TIMLO   EQU     24H     ; OUTPUT PORT FOR LOW ORDER BYTE OF TIMER VALUE
0040                134 TMODE   EQU     40H     ; TIMER MODE - SQUARE WAVE, AUTO RELOAD
00C0                135 TSTRT   EQU     0C0H    ; START TIMER
000E                136 UNMSK   EQU     0EH     ; UNMASK INPUT INTERRUPT
20C8                137 USRBR   EQU     RAMST + 256 - (RMUSE + SKLN + UBRLN)     ; START OF USER
                    138                                 ; /BRANCH LOCATIONS
                    139         IF      1-WAITS ;TIMER VALUE FOR SINGLE STEP IF NO WAIT STATE
00C5                140 TIMER   EQU     197
                    141         ENDIF
                    142         IF      WAITS   ;TIMER VALUE FOR SINGLE STEP IF ONE WAIT STATE I
                    143 TIMER   EQU     237
                    144         ENDIF
                    145 ;
                    146 ;*******************************************************************
                    147 ;
                    148 ;                       MONITOR MACROS
                    149 ;
                    150 ;*******************************************************************
                    151 ;
                    152 TRUE    MACRO   WHERE   ; BRANCH IF FUNCTION RETURNS TRUE
                    153         JC      WHERE
                    154         ENDM
                    155 ;
                    156 FALSE   MACRO   WHERE   ; BRANCH IF FUNCTION RETURNS FALSE
                    157         JNC     WHERE
                    158         ENDM
                    159 ;
                    160 ;
                    161 ;*******************************************************************
                    162 ;
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                  163 ; ***** "RESET" KEY ENTRY POINT - COLD START
                  164 ; ***** RST 0 ENTRY POINT
                  165 ;
0000 3E00         166          MVI     A,KMODE ; GET CONTROL CHARACTER
0002 320019       167          STA     CNTRL   ; SET KEYBOARD/DISPLAY MODE
0005 C3F101       168          JMP     CLDST   ; GO FINISH COLD START
                  169 CLDBK:                   ; THEN JUMP BACK HERE
                  170 ;
                  171 ; ***** RST 1 ENTRY POINT - WARM START
                  172 ;
0008              173          ORG     8
                  174 ;        SAVE REGISTERS
0008 22EF20       175          SHLD    LSAV    ; SAVE H & L REGISTERS
000B E1           176          POP     H       ; GET USER PROGRAM COUNTER FROM TOP OF STACK
000C 22F220       177          SHLD    PSAV    ; /AND SAVE IT
000F F5           178          PUSH    PSW
0010 E1           179          POP     H
0011 22ED20       180          SHLD    FSAV    ; SAVE FLIP/FLOPS & REGISTER A
0014 210000       181          LXI     H,0     ; CLEAR H & L
0017 39           182          DAD     SP      ; GET USER STACK POINTER
0018 22F420       183          SHLD    SSAV    ; /AND SAVE IT
001B 21ED20       184          LXI     H,BSAV+1 ; SET STACK POINTER FOR SAVING
001E F9           185          SPHL             ; /REMAINING REGISTERS
001F C5           186          PUSH    B       ; SAVE B & C
0020 D5           187          PUSH    D       ; SAVE D & E
0021 C33F00       188          JMP     RES10   ; LEAVE ROOM FOR VECTORED INTERRUPTS
                  189 ;
                  190 ; ***** TIMER INTERRUPT (TRAP) ENTRY POINT (RST 4.5)
0024              191          ORG     24H
0024 C35701       192          JMP     STP25   ; BACK TO SINGLE STEP ROUTINE
                  193 ;
                  194 ; ***** RST 5 ENTRY POINT
                  195 ;
0028              196          ORG     28H
0028 C3C820       197          JMP     RSET5   ; BRANCH TO RST 5 LOCATION IN RAM
                  198 ;
                  199 ; ***** INPUT INTERRUPT ENTRY POINT (RST 5.5)
                  200 ;
002C              201          ORG     2CH
002C C38E02       202          JMP     ININT   ; BRANCH TO INPUT INTERRUPT ROUTINE
                  203 ;
                  204 ; ***** RST 6 ENTRY POINT
                  205 ;
0030              206          ORG     30H
0030 C3CB20       207          JMP     RSET6   ; BRANCH TO RST 6 LOCATION IN RAM
                  208 ;
                  209 ; ***** HARD WIRED USER INTERRUPT ENTRY POINT (RST 6.5)
                  210 ;
0034              211          ORG     34H
0034 C3CE20       212          JMP     RST65   ; BRANCH TO RST 6.5 LOCATION IN RAM
                  213 ;
                  214 ; ***** RST 7 ENTRY POINT
                  215 ;
0038              216          ORG     38H
0038 C3D120       217          JMP     RSET7   ; BRANCH TO RST 7 LOCATION IN RAM
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

                  218 ;
                  219 ; ***** "VECTORED INTERRUPT" KEY ENTRY POINT (RST 7.5)
003C              220         ORG     3CH
003C C3D420       221         JMP     USINT   ; BRANCH TO USER INTERRUPT LOCATION IN RAM
                  222 ;
                  223 RES10:  ; CONTINUE SAVING USER STATUS
003F 20           224         RIM             ; GET USER INTERRUPT STATUS AND INTERRUPT MASK
0040 E60F         225         ANI     0FH     ; KEEP STATUS & MASK BITS
0042 32F120       226         STA     ISAV    ; SAVE INTERRUPT STATUS & MASK
0045 3E0E         227         MVI     A,UNMSK ; UNMASK INTERRUPTS FOR MONITOR USE
0047 30           228         SIM
0048 F3           229         DI              ; INTERRUPTS DISABLED WHILE MONITOR IS RUNNING
                  230                         ; (EXCEPT WHEN WAITING FOR INPUT)
0049 20           231         RIM             ; TTY OR KEYBOARD MONITOR ?
004A 07           232         RLC             ; IS TTY CONNECTED ?
004B DAFA03       233         JC      GO      ; YES - BRANCH TO TTY MONITOR
                  234                         ; NO - ENTER KEYBOARD MONITOR
                  235 ;
                  236 ;*******************************************************************
                  237 ;
                  238 ;                       BEGINNING OF KEYBOARD MONITOR CODE
                  239 ;
                  240 ;*******************************************************************
                  241 ;
                  242 ;       OUTPUT SIGN-ON MESSAGE
004E AF           243         XRA     A       ; ARG - USE ADDRESS FIELD OF DISPLAY
004F 0600         244         MVI     B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
0051 21A603       245         LXI     H,SGNAD ; ARG - GET ADDRESS OF ADDRESS FIELD PORTION OF
                  246                         ; /SIGN-ON MESSAGE
0054 CDB702       247         CALL    OUTPT   ; OUTPUT SIGN-ON MESSAGE TO ADDRESS FIELD
0057 3E01         248         MVI     A,DTFLD ; ARG - USE DATA FIELD OF DISPLAY
0059 0600         249         MVI     B,NODOT ; ARG - NO DOT IN DATA FIELD
005B 21AA03       250         LXI     H,SGNDT ; ARG - GET ADDRESS OF DATA FIELD PORTION OF
                  251                         ; /SIGN-ON MESSAGE
005E CDB702       252         CALL    OUTPT   ; OUTPUT SIGN-ON MESSAGE TO DATA FIELD
0061 3E80         253         MVI     A,EMPTY
0063 32FE20       254         STA     IBUFF   ; SET INPUT BUFFER EMPTY FLAG
                  255 ;
                  256 ;*******************************************************************
                  257 ;
                  258 ; FUNCTION: CMMND - COMMAND RECOGNIZER
                  259 ; INPUTS: NONE
                  260 ; OUTPUTS: NONE
                  261 ; CALLS: RDKBD,ERR,SUBST,EXAM,GOCMD,SSTEP
                  262 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  263 ;
                  264 CMMND:
0066 21E920       265         LXI     H,MNSTK ; INITIALIZE MONITOR STACK POINTER
0069 F9           266         SPHL
                  267                         ; OUTPUT PROMPT CHARACTER TO DISPLAY
006A 210019       268         LXI     H,CNTRL ; GET ADDRESS FOR CONTROL CHARACTER
006D 3690         269         MVI     M,ADISP ; OUTPUT CONTROL CHARACTER TO USE ADDRESS FIELD
006F 25           270         DCR     H       ; ADDRESS FOR CUTPUT CHARACTER
0070 36FB         271         MVI     M,PRMPT ; OUTPUT PROMPT CHARACTER
0072 CDE702       272         CALL    RDKBD   ; READ KEYBOARD
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0075 010400        273          LXI     B,NUMC  ; COUNTER FOR NUMBER OF COMMANDS IN C
0078 217803        274          LXI     H,CMDTB ; GET ADDRESS OF COMMAND TABLE
                   275 CMD10:
007B BE            276          CMP     M       ; RECOGNIZE THE COMMAND ?
007C CA8700        277          JZ      CMD15   ; YES - GO PROCESS IT
007F 23            278          INX     H       ; NO - NEXT COMMAND TABLE ENTRY
0080 0D            279          DCR     C       ; END OF TABLE ?
0081 C27B00        280          JNZ     CMD10   ; NO - GO CHECK NEXT ENTRY
                   281                          ; YES - COMMAND UNKNOWN
0084 C31502        282          JMP     ERR     ; DISPLAY ERROR MESSAGE AND GET ANOTHER COMMAND
                   283 CMD15:
0087 217C03        284          LXI     H,CMDAD ; GET ADDRESS OF COMMAND ADDRESS TABLE
008A 0D            285          DCR     C       ; ADJUST COMMAND COUNTER
                   286                          ; COUNTER ACTS AS POINTER TO COMMAND ADDRESS TABLE
008B 09            287          DAD     B       ; ADD POINTER TO TABLE ADDRESS TWICE BECAUSE
008C 09            288          DAD     B       ; TABLE HAS 2 BYTE ENTRIES
008D 7E            289          MOV     A,M     ; GET LOW ORDER BYTE OF COMMAND ADDRESS
008E 23            290          INX     H
008F 66            291          MOV     H,M     ; GET HIGH ORDER BYTE OF COMMAND ADDRESS IN H
0090 6F            292          MOV     L,A     ; PUT LOW ORDER BYTE IN L
                   293                          ; COMMAND ROUTINE ADDRESS IS NOW IN H & L
0091 E9            294          PCHL             ; BRANCH TO ADDRESS IN H & L
                   295 ;
                   296 ;*****************************************************************
                   297 ;
                   298 ;                          COMMAND ROUTINES
                   299 ;
                   300 ;*****************************************************************
                   301 ;
                   302 ; FUNCTION: EXAM - EXAMINE AND MODIFY REGISTERS
                   303 ; INPUTS: NONE
                   304 ; OUTPUTS: NONE
                   305 ; CALLS: CLEAR,SETRG,ERR,RGNAM,RGLOC,UPDDT,GTHEX,NXTRG
                   306 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   307 ;
                   308 EXAM:
0092 0601          309          MVI     B,DOT   ; ARG - DOT IN ADDRESS FIELD OF DISPLAY
0094 CDD701        310          CALL    CLEAR   ; CLEAR DISPLAY
0097 CD4403        311          CALL    SETRG   ; GET REGISTER DESIGNATOR FROM KEYBOARD AND
                   312                          ;/SET REGISTER POINTER ACCORDINGLY
                   313                          ; WAS CHARACTER A REGISTER DESIGNATOR?
                   314          FALSE   ERR     ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
009A D21502        315+         JNC     ERR
                   316 EXM05:
009D CD0903        317          CALL    RGNAM   ; OUTPUT REGISTER NAME TO ADDRESS FIELD
00A0 CDFC02        318          CALL    RGLOC   ; GET REGISTER SAVE LOCATION IN H & L
00A3 7E            319          MOV     A,M     ; GET REGISTER CONTENTS
00A4 32F820        320          STA     CURDT   ; STORE REGISTER CONTENTS AT CURRENT DATA
00A7 0601          321          MVI     B,DOT   ; ARG - DOT IN DATA FIELD
00A9 CD6B03        322          CALL    UPDDT   ; UPDATE DATA FIELD OF DISPLAY
00AC 0601          323          MVI     B,DTFLD ; ARG - USE DATA FIELD OF DISPLAY
00AE CD2B02        324          CALL    GTHEX   ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED?
                   325          FALSE   EXM10   ; NO - DO NOT UPDATE REGISTER CONTENTS
00B1 D2B800        326+         JNC     EXM10
00B4 CDFC02        327          CALL    RGLOC   ; YES - GET REGISTER SAVE LOCATION IN H & L
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

 00B7 73            328          MOV     M,E     ; UPDATE REGISTER CONTENTS
                    329 EXM10:
 00B8 FE10          330          CPI     PERIO   ; WAS LAST CHARACTER A PERIOD ?
 00BA CAE901        331          JZ      CLDIS   ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
 00BD FE11          332          CPI     COMMA   ; WAS LAST CHARACTER ',' ?
 00BF C21502        333          JNZ     ERR     ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAN
 00C2 CDA802        334          CALL    NXTRG   ; YES - ADVANCE REGISTER POINTER TO
                    335                          ;/NEXT REGISTER
                    336                          ; ANY MORE REGISTERS ?
                    337          TRUE    EXM05   ; YES - CONTINUE PROCESSING WITH NEXT REGISTER
 00C5 DA9D00        338+         JC      EXM05
 00C8 C3E901        339          JMP     CLDIS   ; NO - CLEAR DISPLAY AND TERMINATE COMMAND
                    340 ;
                    341 ;*****************************************************************
                    342 ;
                    343 ; FUNCTION: GOCMD - EXECUTE USER PROGRAM
                    344 ; INPUTS: NONE
                    345 ; OUTPUTS: NONE
                    346 ; CALLS: DISPC,RDKBD,CLEAR,GTHEX,ERR,OUTPT
                    347 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    348 ;
                    349 GOCMD:
 00CB CD0002        350          CALL    DISPC   ; DISPLAY USER PROGRAM COUNTER
 00CE CDE702        351          CALL    RDKBD   ; READ FROM KEYBOARD
 00D1 FE10          352          CPI     PERIO   ; IS CHARACTER A PERIOD ?
 00D3 CAEC00        353          JZ      G10     ; YES - GO EXECUTE THE COMMAND
                    354                          ; NO - ARG - CHARACTER IS STILL IN A
 00D6 32FE20        355          STA     IBUFF   ; REPLACE CHARACTER IN INPUT BUFFER
 00D9 0601          356          MVI     B,DOT   ; ARG - DOT IN ADDRESS FIELD
 00DB CDD701        357          CALL    CLEAR   ; CLEAR DISPLAY
 00DE 0600          358          MVI     B,ADFLD ; ARG - USE ADDRESS FIELD
 00E0 CD2B02        359          CALL    GTHEX   ; GET HEX DIGITS
 00E3 FE10          360          CPI     PERIO   ; WAS LAST CHARACTER A PERIOD ?
 00E5 C21502        361          JNZ     ERR     ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
 00E8 EB            362          XCHG            ; PUT HEX VALUE FROM GTHEX TO H & L
 00E9 22F220        363          SHLD    PSAV    ; HEX VALUE IS NEW USER PC
                    364 G10:
 00EC 0600          365          MVI     B,NODOT ; YES - ARG - NO DOT IN ADDRESS FIELD
 00EE CDD701        366          CALL    CLEAR   ; CLEAR DISPLAY
 00F1 AF            367          XRA     A       ; ARG - USE ADDRESS FIELD OF DISPLAY
 00F2 0600          368          MVI     B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
 00F4 21A203        369          LXI     H,EXMSG ; GET ADDRESS OF EXECUTION MESSAGE IN H & L
 00F7 CDB702        370          CALL    OUTPT   ; DISPLAY EXECUTION MESSAGE
 00FA C31B03        371          JMP     RSTOR   ; RESTORE USER REGISTERS INCL. PROGRAM COUNTER
                    372                          ;/I.E. BEGIN EXECUTION OF USER PROGRAM
                    373 ;
                    374 ;*****************************************************************
                    375 ;
                    376 ; FUNCTION: SSTEP - SINGLE STEP (EXECUTE ONE USER INSTRUCTION)
                    377 ; INPUTS: NONE
                    378 ; OUTPUTS: NONE
                    379 ; CALLS: DISPC,RDKBD,CLEAR,GTHEX,ERR
                    380 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    381 ;
                    382 SSTEP:
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

00FD CD0002        383          CALL    DISPC   ; DISPLAY USER PROGRAM COUNTER
0100 CDE702        384          CALL    RDKBD   ; READ FROM KEYBOARD
0103 FE10          385          CPI     PERIO   ; WAS CHARACTER A PERIOD ?
0105 CAE901        386          JZ      CLDIS   ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
0108 FE11          387          CPI     COMMA   ; WAS LAST CHARACTER ',' ?
010A CA2601        388          JZ      STP20   ; YES - GO SET TIMER
                   389          ; NO - CHARACTER FROM KEYBOARD WAS NEITHER PERIOD NOR COMMA
010D 32FE20        390          STA     IBUFF   ; REPLACE THE CHARACTER IN THE INPUT BUFFER
0110 0601          391          MVI     B,DOT   ; ARG - DOT IN ADDRESS FIELD
0112 CDD701        392          CALL    CLEAR   ; CLEAR DISPLAY
0115 0600          393          MVI     B,ADFLD ; ARG - USE ADDRESS FIELD OF DISPLAY
0117 CD2B02        394          CALL    GTHEX   ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED ?
                   395          FALSE   ERR     ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
011A D21502        396+         JNC     ERR
011D EB            397          XCHG            ; HEX VALUE FROM GTHEX TO H & L
011E 22F220        398          SHLD    PSAV    ; HEX VALUE IS NEW USER PC
0121 FE10          399          CPI     PERIO   ; WAS LAST CHARACTER FROM GTHEX A PERIOD ?
0123 CAE901        400          JZ      CLDIS   ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
                   401                          ; NO - MUST HAVE BEEN A COMMA
                   402 STP20:
0126 3AF120        403          LDA     ISAV    ; GET USER INTERRUPT MASK
0129 E608          404          ANI     08H     ; KEEP INTERRUPT STATUS
012B 32FD20        405          STA     TEMP    ; SAVE USER INTERRUPT STATUS
012E 2AF220        406          LHLD    PSAV    ; GET USER PC
0131 7E            407          MOV     A,M     ; GET USER INSTRUCTION
0132 FEF3          408          CPI     (DI)    ; DI INSTRUCTION ?
0134 C23B01        409          JNZ     STP21   ; NO
0137 AF            410          XRA     A       ; YES - RESET USER INTERRUPT STATUS
0138 C34201        411          JMP     STP22
                   412 STP21:
013B FEFB          413          CPI     (EI)    ; EI INSTRUCTION ?
013D C24501        414          JNZ     STP23   ; NO
0140 3E08          415          MVI     A,08H   ; YES - SET USER INTERRUPT STATUS
                   416 STP22:
0142 32FD20        417          STA     TEMP    ; SAVE NEW USER INTERRUPT STATUS
                   418 STP23:
0145 3E40          419          MVI     A,(TIMER SHR 8) OR TMODE ; HIGH ORDER BITS OF TIMER VALUE
                   420                                           ; /OR'ED WITH TIMER MODE
0147 D325          421          OUT     TIMHI
0149 3EC5          422          MVI     A,TIMER AND OFFH ; LOW ORDER BITS OF TIMER VALUE
014B D324          423          OUT     TIMLO
014D 3AFF20        424          LDA     USCSR   ; GET USER IMAGE OF WHAT'S IN CSR
0150 F6C0          425          ORI     TSTRT   ; SET TIMER COMMAND BITS TO START TIMER
0152 D320          426          OUT     CSR     ; START TIMER
0154 C31B03        427          JMP     RSTOR   ; RESTORE USER REGISTERS
                   428 ;
                   429 STP25:                   ; BRANCH HERE WHEN TIMER INTERRUPTS AFTER
                   430                          ;/ONE USER INSTRUCTION
0157 F5            431          PUSH    PSW     ; SAVE PSW
0158 3AFF20        432          LDA     USCSR   ; GET USER IMAGE OF WHAT'S IN CSR
015B E63F          433          ANI     3FH     ; CLEAR 2 HIGH ORDER BITS
015D F640          434          ORI     40H     ; SET TIMER STOP BIT
015F D320          435          OUT     CSR     ; STOP TIMER
0161 F1            436          POP     PSW     ; RETRIEVE PSW
0162 22EF20        437          SHLD    LSAV    ; SAVE H & L
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0165 E1           438          POP     H        ; GET USER PROGRAM COUNTER FROM TOP OF STACK
0166 22F220       439          SHLD    PSAV     ; SAVE USER PC
0169 F5           440          PUSH    PSW
016A E1           441          POP     H
016B 22ED20       442          SHLD    FSAV     ; SAVE FLIP/FLOPS AND A REGISTER
016E 210000       443          LXI     H,0      ; CLEAR H & L
0171 39           444          DAD     SP       ; GET USER STACK POINTER
0172 22F420       445          SHLD    SSAV     ; SAVE USER STACK POINTER
0175 21ED20       446          LXI     H,BSAV+1         ; SET MONITOR STACK POINTER FOR
0178 F9           447          SPHL             ;/SAVING REMAINING USER REGISTERS
0179 C5           448          PUSH    B        ; SAVE B & C
017A D5           449          PUSH    D        ; SAVE D & E
017B 20           450          RIM              ; GET USER INTERRUPT MASK
017C E607         451          ANI     07H      ; KEEP MASK BITS
017E 21FD20       452          LXI     H,TEMP   ; GET USER INTERRUPT STATUS
0181 B6           453          ORA     M        ; OR IT INTO MASK
0182 32F120       454          STA     ISAV     ; SAVE INTERRUPT STATUS & MASK
0185 3E0E         455          MVI     A,UNMSK  ; UNMASK INTERRUPTS FOR MONITOR USE
0187 30           456          SIM
0188 C3FD00       457          JMP     SSTEP    ; GO GET READY FOR ANOTHER INSTRUCTION
                  458 ;
                  459 ;*****************************************************************
                  460 ;
                  461 ; FUNCTION: SUBST -  SUBSTITUTE MEMORY
                  462 ; INPUTS: NONE
                  463 ; OUTPUTS: NONE
                  464 ; CALLS: CLEAR,GTHEX,UPDAD,UPDDT,ERR
                  465 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  466 ;
                  467 SUBST:
018B 0601         468          MVI     B,DOT    ; ARG - DOT IN ADDRESS FIELD
018D CDD701       469          CALL    CLEAR    ; CLEAR THE DISPLAY
0190 0600         470          MVI     B,ADFLD  ; ARG - USE ADDRESS FIELD OF DISPLAY
0192 CD2B02       471          CALL    GTHEX    ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED?
                  472          FALSE   ERR      ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAN
0195 D21502       473+         JNC     ERR
0198 EB           474          XCHG             ; ASSIGN HEX VALUE RETURNED BY GTHEX TO
0199 22F620       475          SHLD    CURAD    ; / CURRENT ADDRESS
                  476 SUB05:
019C FE11         477          CPI     COMMA    ; WAS ',' THE LAST CHARACTER FROM KEYBOARD?
019E C2CF01       478          JNZ     SUB15    ; NO - GO TERMINATE THE COMMAND
01A1 0600         479          MVI     B,NODOT  ; ARG - NO DOT IN ADDRESS FIELD
01A3 CD5F03       480          CALL    UPDAD    ; UPDATE ADDRESS FIELD OF DISPLAY
01A6 2AF620       481          LHLD    CURAD    ; GET CURRENT ADDRESS IN H & L
01A9 7E           482          MOV     A,M      ; GET DATA BYTE POINTED TO BY CURRENT ADDRESS
01AA 32F820       483          STA     CURDT    ; STORE DATA BYTE AT CURRENT DATA
01AD 0601         484          MVI     B,DOT    ; ARG - DOT IN DATA FIELD
01AF CD6B03       485          CALL    UPDDT    ; UPDATE DATA FIELD OF DISPLAY
01B2 0601         486          MVI     B,DTFLD  ; ARG - USE DATA FIELD
01B4 CD2B02       487          CALL    GTHEX    ; GET HEX DIGITS - WERE ANY HEX DIGITS RECEIVE
01B7 F5           488          PUSH    PSW      ; (SAVE LAST CHARACTER)
                  489          FALSE   SUB10    ; NO - LEAVE DATA UNCHANGED AT CURRENT ADDRESS
01B8 D2C401       490+         JNC     SUB10
01BB 2AF620       491          LHLD    CURAD    ; YES - GET CURRENT ADDRESS IN H & L
01BE 73           492          MOV     M,E      ; STORE NEW DATA AT CURRENT ADDRESS
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   493                  ; MAKE SURE DATA WAS ACTUALLY STORED IN CASE
                   494                  ;/CURRENT ADDRESS IS IN ROM OR IS NON-EXISTANT
01BF 7B            495          MOV     A,E       ; DATA TO A FOR COMPARISON
01C0 BE            496          CMP     M         ; WAS DATA STORED CORRECTLY?
01C1 C21502        497          JNZ     ERR       ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
                   498 SUB10:
01C4 2AF620        499          LHLD    CURAD     ; INCREMENT CURRENT ADDRESS
01C7 23            500          INX     H
01C8 22F620        501          SHLD    CURAD
01CB F1            502          POP     PSW       ; RETRIEVE LAST CHARACTER
01CC C39C01        503          JMP     SUB05     ;
                   504 SUB15:
01CF FE10          505          CPI     PERIO     ; WAS LAST CHARACTER '.' ?
01D1 C21502        506          JNZ     ERR       ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
01D4 C3E901        507          JMP     CLDIS     ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
                   508 ;
                   509 ;
                   510 ;*************************************************************************
                   511 ;
                   512 ;                          UTILITY ROUTINES
                   513 ;
                   514 ;*************************************************************************
                   515 ;
                   516 ; FUNCTION: CLEAR - CLEAR THE DISPLAY
                   517 ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT IN ADDRESS FIELD OF DISPLAY
                   518 ;                       - 0 MEANS NO DOT
                   519 ; OUTPUTS: NONE
                   520 ; CALLS: OUTPT
                   521 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   522 ; DESCRIPTION:  CLEAR SENDS BLANK CHARACTERS TO BOTH THE ADDRESS FIELD
                   523 ;               AND THE DATA FIELD OF THE DISPLAY. IF THE DOT FLAG IS
                   524 ;               SET THEN A DOT WILL APPEAR AT THE RIGHT EDGE OF THE
                   525 ;               ADDRESS FIELD.
                   526 ;
                   527 CLEAR:
01D7 AF            528          XRA     A         ; ARG - USE ADDRESS FIELD OF DISPLAY
                   529                            ; ARG - FLAG FOR DOT IN ADDR. FIELD IS IN B
01D8 219A03        530          LXI     H,BLNKS   ; ARG - ADDRESS OF BLANKS FOR DISPLAY
01DB CDB702        531          CALL    OUTPT     ; OUTPUT BLANKS TO ADDRESS FIELD
01DE 3E01          532          MVI     A,DTFLD   ; ARG - USE DATA FIELD OF DISPLAY
01E0 0600          533          MVI     B,NODOT   ; ARG - NO DOT IN DATA FIELD
01E2 219A03        534          LXI     H,BLNKS   ; ARG - ADDRESS OF BLANKS FOR DISPLAY
01E5 CDB702        535          CALL    OUTPT     ; OUTPUT BLANKS TO DATA FIELD
01E8 C9            536          RET               ; RETURN
                   537 ;
                   538 ;*************************************************************************
                   539 ;
                   540 ; FUNCTION: CLDIS - CLEAR DISPLAY AND TERMINATE COMMAND
                   541 ; INPUTS: NONE
                   542 ; OUTPUTS: NONE
                   543 ; CALLS: CLEAR
                   544 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   545 ; DESCRIPTION: CLDIS IS JUMPED TO BY COMMAND ROUTINES WISHING TO
                   546 ;                TERMINATE NORMALLY. CLDIS CLEARS THE DISPLAY AND
                   547 ;                BRANCHES TO THE COMMAND RECOGNIZER.
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   548 ;
                   549 CLDIS:
01E9 0600          550          MVI     B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
01EB CDD701        551          CALL    CLEAR   ; CLEAR THE DISPLAY
01EE C36600        552          JMP     CMMND   ; GO GET ANOTHER COMMAND
                   553 ;
                   554 ;*****************************************************************
                   555 ;
                   556 ; FUNCTION: CLDST - COLD START
                   557 ; INPUTS: NONE
                   558 ; OUTPUTS: NONE
                   559 ; CALLS: NOTHING
                   560 ; DESTROYS: A
                   561 ; DESCRIPTION:  CLDST IS JUMPED TO BY THE MAIN COLD START PROCEDURE,
                   562 ;               COMPLETES COLD START INITIALIZATION, AND JUMPS BACK
                   563 ;               TO THE MAIN COLD START PROCEDURE.
                   564 ;
                   565 CLDST:
01F1 3ECC          566          MVI     A,KBNIT ; GET CONTROL CHARACTER
01F3 320019        567          STA     CNTRL   ; INITIALIZE KEYBOARD/DISPLAY BLANKING
01F6 3E00          568          MVI     A,CSNIT ; INITIAL VALUE OF COMMAND STATUS REGISTER
01F8 D320          569          OUT     CSR     ; INITIALIZE CSR
01FA 32FF20        570          STA     USCSR   ; INITIALIZE USER CSR VALUE
01FD C30800        571          JMP     CLDBK   ; BACK TO MAIN PROCEDURE
                   572 ;
                   573 ;*****************************************************************
                   574 ;
                   575 ; FUNCTION: DISPC - DISPLAY PROGRAM COUNTER
                   576 ; INPUTS: NONE
                   577 ; OUTPUTS: NONE
                   578 ; CALLS: UPDAD,UPDDT
                   579 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   580 ; DESCRIPTION: DISPC DISPLAYS THE USER PROGRAM COUNTER IN THE ADDRESS
                   581 ;               FIELD OF THE DISPLAY, WITH A DOT AT THE RIGHT EDGE
                   582 ;               OF THE FIELD. THE BYTE OF DATA ADDRESSED BY THE PROGRAM
                   583 ;               COUNTER IS DISPLAYED IN THE DATA FIELD OF THE DISPLAY.
                   584 ;
                   585 DISPC:
0200 2AF220        586          LHLD    PSAV    ; GET USER PROGRAM COUNTER
0203 22F620        587          SHLD    CURAD   ; MAKE IT THE CURRENT ADDRESS
0206 7E            588          MOV     A,M     ; GET THE INSTRUCTION AT THAT ADDRESS
0207 32F820        589          STA     CURDT   ; MAKE IT THE CURRENT DATA
020A 0601          590          MVI     B,DOT   ; ARG - DOT IN ADDRESS FIELD
020C CD5F03        591          CALL    UPDAD   ; UPDATE ADDRESS FIELD OF DISPLAY
020F 0600          592          MVI     B,NODOT ; ARG - NO DOT IN DATA FIELD
0211 CD6B03        593          CALL    UPDDT   ; UPDATE DATA FIELD OF DISPLAY
0214 C9            594          RET
                   595 ;
                   596 ;*****************************************************************
                   597 ;
                   598 ; FUNCTION: ERR - DISPLAY ERROR MESSAGE
                   599 ; INPUTS: NONE
                   600 ; OUTPUTS: NONE
                   601 ; CALLS: OUTPT
                   602 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

                    603 ; DESCRIPTION:  ERR IS JUMPED TO BY COMMAND ROUTINES WISHING TO
                    604 ;               TERMINATE BECAUSE OF AN ERROR.
                    605 ;               ERR OUTPUTS AN ERROR MESSAGE TO THE DISPLAY AND
                    606 ;               BRANCHES TO THE COMMAND RECOGNIZER.
                    607 ;
                    608 ERR:
0215 AF             609         XRA     A       ; ARG - USE ADDRESS FIELD
0216 0600           610         MVI     B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
0218 219E03         611         LXI     H,ERMSG ; ARG - ADDRESS OF ERROR MESSAGE
021B CDB702         612         CALL    OUTPT   ; OUTPUT ERROR MESSAGE TO ADDRESS FIELD
021E 3E01           613         MVI     A,DTFLD ; ARG - USE DATA FIELD
0220 0600           614         MVI     B,NODOT ; ARG - NO DOT IN DATA FIELD
0222 219A03         615         LXI     H,BLNKS ; ARG - ADDRESS OF BLANKS FOR DISPLAY
0225 CDB702         616         CALL    OUTPT   ; OUTPUT BLANKS TO DATA FIELD
0228 C36600         617         JMP     CMMND   ; GO GET A NEW COMMAND
                    618 ;
                    619 ;*****************************************************************************
                    620 ;
                    621 ; FUNCTION: GTHEX - GET HEX DIGITS
                    622 ; INPUTS: B - DISPLAY FLAG - 0 MEANS USE ADDRESS FIELD OF DISPLAY
                    623 ;                           - 1 MEANS USE DATA FIELD OF DISPLAY
                    624 ; OUTPUTS: A - LAST CHARACTER READ FROM KEYBOARD
                    625 ;          DE - HEX DIGITS FROM KEYBOARD EVALUATED MODULO 2**16
                    626 ;          CARRY - SET IF AT LEAST ONE VALID HEX DIGIT WAS READ
                    627 ;                - RESET OTHERWISE
                    628 ; CALLS: RDKBD,INSDG,HXDSP,OUTPT
                    629 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    630 ; DESCRIPTION:  GTHEX ACCEPTS A STRING OF HEX DIGITS FROM THE KEYBOARD,
                    631 ;               DISPLAYS THEM AS THEY ARE RECEIVED, AND RETURNS THEIR
                    632 ;               VALUE AS A 16 BIT INTEGER. IF MORE THAN 4 HEX DIGITS
                    633 ;               ARE RECEIVED, ONLY THE LAST 4 ARE USED. IF THE DISPLAY
                    634 ;               FLAG IS SET, THE LAST 2 HEX DIGITS ARE DISPLAYED IN THE
                    635 ;               DATA FIELD OF THE DISPLAY. OTHERWISE, THE LAST 4 HEX
                    636 ;               DIGITS ARE DISPLAYED IN THE ADDRESS FIELD OF THE
                    637 ;               DISPLAY. IN EITHER CASE, A DOT WILL BE DISPLAYED AT THE
                    638 ;               RIGHTMOST EDGE OF THE FIELD. A CHARACTER WHICH IS NOT
                    639 ;               A HEX DIGIT TERMINATES THE STRING AND IS RETURNED AS
                    640 ;               AN OUTPUT OF THE FUNCTION. IF THE TERMINATOR IS NOT
                    641 ;               A PERIOD OR A COMMA THEN ANY HEX DIGITS WHICH MAY HAVE
                    642 ;               BEEN RECEIVED ARE CONSIDERED TO BE INVALID. THE
                    643 ;               FUNCTION RETURNS A FLAG INDICATING WHETHER OR NOT ANY
                    644 ;               VALID HEX DIGITS WERE RECEIVED.
                    645 ;
                    646 GTHEX:
022B 0E00           647         MVI     C,0     ; RESET HEX DIGIT FLAG
022D C5             648         PUSH    B       ; SAVE DISPLAY AND HEX DIGIT FLAGS
022E 110000         649         LXI     D,0     ; SET HEX VALUE TO ZERO
0231 D5             650         PUSH    D       ; SAVE HEX VALUE
                    651 GTH05:
0232 CDE702         652         CALL    RDKBD   ; READ KEYBOARD
0235 FE10           653         CPI     10H     ; IS CHARACTER A HEX DIGIT?
0237 D25502         654         JNC     GTH20   ; NO - GO CHECK FOR TERMINATOR
                    655                         ; YES - ARG - NEW HEX DIGIT IS IN A
023A D1             656         POP     D       ; ARG - RETRIEVE HEX VALUE
023B CD9F02         657         CALL    INSDG   ; INSERT NEW DIGIT IN HEX VALUE
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

023E C1            658          POP     B        ; RETRIEVE DISPLAY FLAG
023F 0E01          659          MVI     C,1      ; SET HEX DIGIT FLAG
                   660                           ;/(I.E. A HEX DIGIT HAS BEEN READ)
0241 C5            661          PUSH    B        ; SAVE DISPLAY AND HEX DIGIT FLAGS
0242 D5            662          PUSH    D        ; SAVE HEX VALUE
0243 78            663          MOV     A,B      ; TEST DISPLAY FLAG
0244 0F            664          RRC              ; SHOULD ADDRESS FIELD OF DISPLAY BE USED ?
0245 D24902        665          JNC     GTH10    ; YES - USE HEX VALUE AS IS
                   666                           ; NO - ONLY LOW ORDER BYTE OF HEX VALUE SHOULD
                   667                           ; /BE USED FOR DATA FIELD OF DISPLAY
0248 53            668          MOV     D,E      ; PUT LOW ORDER BYTE OF HEX VALUE IN D
                   669 GTH10:
                   670                           ; ARG - HEX VALUE TO BE EXPANDED IS IN D & E
0249 CD6C02        671          CALL    HXDSP    ; EXPAND HEX VALUE FOR DISPLAY
                   672                           ; ARG - ADDRESS OF EXPANDED HEX VALUE IN H & L
024C 78            673          MOV     A,B      ; ARG - PUT DISPLAY FLAG IN A
024D 0601          674          MVI     B,DOT    ; ARG - DOT IN APPROPRIATE FIELD
024F CDB702        675          CALL    OUTPT    ; OUTPUT HEX VALUE TO DISPLAY
0252 C33202        676          JMP     GTH05    ; GO GET NEXT CHARACTER
                   677 GTH20:                    ; LAST CHARACTER WAS NOT A HEX DIGIT
0255 D1            678          POP     D        ; RETRIEVE HEX VALUE
0256 C1            679          POP     B        ; RETRIEVE HEX DIGIT FLAG IN C
0257 FE11          680          CPI     COMMA    ; WAS LAST CHARACTER ',' ?
0259 CA6702        681          JZ      GTH25    ; YES - READY TO RETURN
025C FE10          682          CPI     PERIO    ; NO - WAS LAST CHARACTER '.' ?
025E CA6702        683          JZ      GTH25    ; YES - READY TO RETURN
                   684                           ; NO - INVALID TERMINATOR - IGNORE ANY HEX DIGITS READ
0261 110000        685          LXI     D,0      ; SET HEX VALUE TO ZERO
0264 C3F702        686          JMP     RETF     ; RETURN FALSE
                   687 GTH25:
0267 47            688          MOV     B,A      ; SAVE LAST CHARACTER
0268 79            689          MOV     A,C      ; SHIFT HEX DIGIT FLAG TO
0269 0F            690          RRC              ;/CARRY BIT
026A 78            691          MOV     A,B      ; RESTORE LAST CHARACTER
026B C9            692          RET              ; RETURN
                   693 ;
                   694 ;*****************************************************************
                   695 ;
                   696 ; FUNCTION: HXDSP - EXPAND HEX DIGITS FOR DISPLAY
                   697 ; INPUTS: DE - 4 HEX DIGITS
                   698 ; OUTPUTS: HL - ADDRESS OF OUTPUT BUFFER
                   699 ; CALLS: NOTHING
                   700 ; DESTROYS: A,H,L,F/F'S
                   701 ; DESCRIPTION: HXDSP EXPANDS EACH INPUT BYTE TO 2 BYTES IN A FORM
                   702 ;              SUITABLE FOR DISPLAY BY THE OUTPUT ROUTINES. EACH INPUT
                   703 ;              BYTE IS DIVIDED INTO 2 HEX DIGITS. EACH HEX DIGIT IS
                   704 ;              PLACED IN THE LOW ORDER 4 BITS OF A BYTE WHOSE HIGH
                   705 ;              ORDER 4 BITS ARE SET TO ZERO. THE RESULTING BYTE IS
                   706 ;              STORED IN THE OUTPUT BUFFER. THE FUNCTION RETURNS THE
                   707 ;              ADDRESS OF THE OUTPUT BUFFER.
                   708 ;
                   709 HXDSP:
026C 7A            710          MOV     A,D      ; GET FIRST DATA BYTE
026D 0F            711          RRC              ; CONVERT 4 HIGH ORDER BITS
026E 0F            712          RRC              ; /TO A SINGLE CHARACTER
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

026F 0F            713          RRC
0270 0F            714          RRC
0271 E60F          715          ANI     0FH
0273 21F920        716          LXI     H,OBUFF ; GET ADDRESS OF OUTPUT BUFFER
0276 77            717          MOV     M,A     ; STORE CHARACTER IN OUTPUT BUFFER
0277 7A            718          MOV     A,D     ; GET FIRST DATA BYTE AND CONVERT 4 LOW ORDER
0278 E60F          719          ANI     0FH     ; /BITS TO A SINGLE CHARACTER
027A 23            720          INX     H       ; NEXT BUFFER POSITION
027B 77            721          MOV     M,A     ; STORE CHARACTER IN BUFFER
027C 7B            722          MOV     A,E     ; GET SECOND DATA BYTE
027D 0F            723          RRC             ; CONVERT 4 HIGH ORDER BITS
027E 0F            724          RRC             ; /TO A SINGLE CHARACTER
027F 0F            725          RRC
0280 0F            726          RRC
0281 E60F          727          ANI     0FH
0283 23            728          INX     H       ; NEXT BUFFER POSITION
0284 77            729          MOV     M,A     ; STORE CHARACTER IN BUFFER
0285 7B            730          MOV     A,E     ; GET SECOND DATA BYTE AND CONVERT LOW ORDER
0286 E60F          731          ANI     0FH     ; /4 BITS TO A SINGLE CHARACTER
0288 23            732          INX     H       ; NEXT BUFFER POSITION
0289 77            733          MOV     M,A     ; STORE CHARACTER IN BUFFER
028A 21F920        734          LXI     H,OBUFF ; RETURN ADDRESS OF OUTPUT BUFFER IN H & L
028D C9            735          RET
                   736 ;
                   737 ;*****************************************************************
                   738 ;
                   739 ; FUNCTION: ININT - INPUT INTERRUPT PROCESSING
                   740 ; INPUTS: NONE
                   741 ; OUTPUTS: NONE
                   742 ; CALLS: NOTHING
                   743 ; DESTROYS: NOTHING
                   744 ; DESCRIPTION:  ININT IS ENTERED BY MEANS OF AN INTERRUPT VECTOR (IV2C)
                   745 ;               WHEN THE READ KEYBOARD ROUTINE IS WAITING FOR A
                   746 ;               CHARACTER AND THE USER HAS PRESSED A KEY ON THE
                   747 ;               KEYBOARD (EXCEPT "RESET" OR "VECTORED INTERRUPT").
                   748 ;               ININT STORES THE INPUT CHARACTER IN THE INPUT BUFFER AND
                   749 ;               RETURNS CONTROL TO THE READ KEYBOARD ROUTINE.
                   750 ;
                   751 ININT:
028E E5            752          PUSH    H       ; SAVE H & L
028F F5            753          PUSH    PSW     ; SAVE F/F'S & REGISTER A
0290 210019        754          LXI     H,CNTRL ; ADDRESS FOR CONTROL CHARACTER OUTPUT
0293 3640          755          MVI     M,READ  ; OUTPUT CONTROL CHARACTER FOR READING
                   756                          ; /FROM KEYBOARD
0295 25            757          DCR     H       ; ADDRESS-FOR CHARACTER INPUT
0296 7E            758          MOV     A,M     ; READ A CHARACTER
0297 E63F          759          ANI     3FH     ; ZERO 2 HIGH ORDER BITS
0299 32FE20        760          STA     IBUFF   ; STORE CHARACTER IN INPUT BUFFER
029C F1            761          POP     PSW     ; RESTORE F/F'S & REGISTER A
029D E1            762          POP     H       ; RESTORE H & L
029E C9            763          RET
                   764 ;
                   765 ;*****************************************************************
                   766 ;
                   767 ; FUNCTION: INSDG - INSERT HEX DIGIT
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   768 ; INPUTS: A - HEX DIGIT TO BE INSERTED
                   769 ;         DE - HEX VALUE
                   770 ; OUTPUTS: DE - HEX VALUE WITH DIGIT INSERTED
                   771 ; CALLS: NOTHING
                   772 ; DESTROYS: A,F/F'S
                   773 ; DESCRIPTION: INSDG SHIFTS THE CONTENTS OF D & E LEFT 4 BITS
                   774 ;              (1 HEX DIGIT) AND INSERTS THE HEX DIGIT IN A IN THE LOW
                   775 ;              ORDER DIGIT POSITION OF THE RESULT. A IS ASSUMED TO
                   776 ;              CONTAIN A SINGLE HEX DIGIT IN THE LOW ORDER 4 BITS AND
                   777 ;              ZEROS IN THE HIGH ORDER  4 BITS.
                   778 ;
                   779 INSDG:
029F EB            780          XCHG              ; PUT D & E IN H & L
02A0 29            781          DAD      H        ; SHIFT H & L LEFT 4 BITS
02A1 29            782          DAD      H
02A2 29            783          DAD      H
02A3 29            784          DAD      H
02A4 85            785          ADD      L        ; INSERT LOW ORDER DIGIT
02A5 6F            786          MOV      L,A
02A6 EB            787          XCHG              ; PUT H & L BACK IN D & E
02A7 C9            788          RET
                   789 ;
                   790 ;*****************************************************************
                   791 ;
                   792 ; FUNCTION: NXTRG - ADVANCE REGISTER POINTER TO NEXT REGISTER
                   793 ; INPUTS: NONE
                   794 ; OUTPUTS: CARRY - 1 IF POINTER IS ADVANCED SUCCESSFULLY
                   795 ;                - 0 OTHERWISE
                   796 ; CALLS: NOTHING
                   797 ; DESTROYS: A,F/F'S
                   798 ; DESCRIPTION:  IF THE REGISTER POINTER POINTS TO THE LAST REGISTER IN
                   799 ;               THE EXAMINE REGISTER SEQUENCE, THE POINTER IS NOT
                   800 ;               CHANGED AND THE FUNCTION RETURNS FALSE. IF THE REGISTER
                   801 ;               POINTER DOES NOT POINT TO THE LAST REGISTER THEN THE
                   802 ;               POINTER IS ADVANCED TO THE NEXT REGISTER IN THE SEQUENCE
                   803 ;               AND THE FUNCTION RETURNS TRUE.
                   804 ;
                   805 NXTRG:
02A8 3AFD20        806          LDA      RGPTR    ; GET REGISTER POINTER
02AB FE0C          807          CPI      NUMRG-1  ; DOES POINTER POINT TO LAST REGISTER?
02AD D2F702        808          JNC      RETF     ; YES - UNABLE TO ADVANCE POINTER - RETURN FALSE
02B0 3C            809          INR      A        ; NO - ADVANCE REGISTER POINTER
02B1 32FD20        810          STA      RGPTR    ; SAVE REGISTER POINTER
02B4 C3FA02        811          JMP      RETT     ; RETURN TRUE
                   812 ;
                   813 ;*****************************************************************
                   814 ;
                   815 ; FUNCTION: OUTPT - OUTPUT CHARACTERS TO DISPLAY
                   816 ; INPUTS: A - DISPLAY FLAG - 0 = USE ADDRESS FIELD
                   817 ;                             1 = USE DATA FIELD
                   818 ;         B - DOT FLAG - 1 = OUTPUT DOT AT RIGHT EDGE OF FIELD
                   819 ;                        0 = NO DOT
                   820 ;         HL - ADDRESS OF CHARACTERS TO BE OUTPUT
                   821 ; CALLS: NOTHING
                   822 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   823 ; DESCRIPTION:  OUTPT SENDS CHARACTERS TO THE DISPLAY. THE ADDRESS
                   824 ;               OF THE CHARACTERS IS RECEIVED AS AN ARGUMENT. EITHER
                   825 ;               2 CHARACTERS ARE SENT TO THE DATA FIELD, OR 4 CHARACTERS
                   826 ;               ARE SENT TO THE ADDRESS FIELD, DEPENDING ON THE
                   827 ;               DISPLAY FLAG ARGUMENT. THE DOT FLAG ARGUMENT DETERMINES
                   828 ;               WHETHER OR NOT A DOT (DECIMAL POINT) WILL BE SENT
                   829 ;               ALONG WITH THE LAST OUTPUT CHARACTER.
                   830 ;
                   831 OUTPT:
02B7 0F            832         RRC             ; USE DATA FIELD ?
02B8 DAC202        833         JC      OUT05   ; YES - GO SET UP TO USE DATA FIELD
02BB 0E04          834         MVI     C,4     ; NO - COUNT FOR ADDRESS FIELD
02BD 3E90          835         MVI     A,ADISP ; CONTROL CHARACTER FOR OUTPUT TO ADDRESS
                   836                         ; /FIELD OF DISPLAY
02BF C3C602        837         JMP     OUT10
                   838 OUT05:
02C2 0E02          839         MVI     C,2     ; COUNT FOR DATA FIELD
02C4 3E94          840         MVI     A,DDISP ; CONTROL CHARACTER FOR OUTPUT TO DATA FIELD
                   841                         ; /OF DISPLAY
                   842 OUT10:
02C6 320019        843         STA     CNTRL
                   844 OUT15:
02C9 7E            845         MOV     A,M     ; GET OUTPUT CHARACTER
02CA EB            846         XCHG            ; SAVE OUTPUT CHARACTER ADDRESS IN D & E
02CB 218403        847         LXI     H,DSPTB ; GET DISPLAY FORMAT TABLE ADDRESS
02CE 85            848         ADD     L       ; USE OUTPUT CHARACTER AS A POINTER TO
02CF 6F            849         MOV     L,A     ; /DISPLAY FORMAT TABLE
02D0 7E            850         MOV     A,M     ; GET DISPLAY FORMAT CHARACTER FROM TABLE
02D1 61            851         MOV     H,C     ; TEST COUNTER WITHOUT CHANGING IT
02D2 25            852         DCR     H       ; IS THIS THE LAST CHARACTER ?
02D3 C2DC02        853         JNZ     OUT20   ; NO - GO OUTPUT CHARACTER AS IS
02D6 05            854         DCR     B       ; YES - IS DOT FLAG SET ?
02D7 C2DC02        855         JNZ     OUT20   ; NO - GO OUTPUT CHARACTER AS IS
02DA F608          856         ORI     DTMSK   ; YES - OR IN MASK TO DISPLAY DOT WITH
                   857                         ; /LAST CHARACTER
                   858 OUT20:
02DC 2F            859         CMA             ; COMPLEMENT OUTPUT CHARACTER
02DD 320018        860         STA     DSPLY   ; SEND CHARACTER TO DISPLAY
02E0 EB            861         XCHG            ; RETRIEVE OUTPUT CHARACTER ADDRESS
02E1 23            862         INX     H       ; NEXT OUTPUT CHARACTER
02E2 0D            863         DCR     C       ; ANY MORE OUTPUT CHARACTERS ?
02E3 C2C902        864         JNZ     OUT15   ; YES - GO PROCESS ANOTHER CHARACTER
02E6 C9            865         RET             ; NO - RETURN
                   866 ;
                   867 ;*******************************************************************
                   868 ;
                   869 ; FUNCTION: RDKBD - READ KEYBOARD
                   870 ; INPUTS: NONE
                   871 ; OUTPUTS: A - CHARACTER READ FROM KEYBOARD
                   872 ; CALLS: NOTHING
                   873 ; DESTROYS: A,H,L,F/F'S
                   874 ; DESCRIPTION:  RDKBD DETERMINES WHETHER OR NOT THERE IS A CHARACTER IN
                   875 ;               THE INPUT BUFFER. IF NOT, THE FUNCTION ENABLES
                   876 ;               INTERRUPTS AND LOOPS UNTIL THE INPUT INTERRUPT
                   877 ;               ROUTINE STORES A CHARACTER IN THE BUFFER. WHEN
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   878 ;                THE BUFFER CONTAINS A CHARACTER, THE FUNCTION FLAGS
                   879 ;                THE BUFFER AS EMPTY AND RETURNS THE CHARACTER
                   880 ;                AS OUTPUT.
                   881 ;
                   882 RDKBD:
02E7 21FE20        883          LXI     H,IBUFF ; GET INPUT BUFFER ADDRESS
02EA 7E            884          MOV     A,M     ; GET BUFFER CONTENTS
                   885                          ; HIGH ORDER BIT = 1 MEANS BUFFER IS EMPTY
02EB B7            886          ORA     A       ; IS A CHARACTER AVAILABLE ?
02EC F2F302        887          JP      RDK10   ; YES - EXIT FROM LOOP
02EF FB            888          EI              ; NO - READY FOR CHARACTER FROM KEYBOARD
02F0 C3E702        889          JMP     RDKBD
                   890 RDK10:
02F3 3680          891          MVI     M,EMPTY ; SET BUFFER EMPTY FLAG
02F5 F3            892          DI              ; RETURN WITH INTERRUPTS DISABLED
02F6 C9            893          RET
                   894 ;
                   895 ;*****************************************************************
                   896 ;
                   897 ; FUNCTION: RETF - RETURN FALSE
                   898 ; INPUTS: NONE
                   899 ; OUTPUTS: CARRY = 0 (FALSE)
                   900 ; CALLS: NOTHING
                   901 ; DESTROYS: CARRY
                   902 ; DESCRIPTION: RETF IS JUMPED TO BY FUNCTIONS WISHING TO RETURN FALSE.
                   903 ;                RETF RESETS CARRY TO 0 AND RETURNS TO THE CALLER OF
                   904 ;                THE ROUTINE INVOKING RETF.
                   905 ;
                   906 RETF:
02F7 37            907          STC             ; SET CARRY TRUE
02F8 3F            908          CMC             ; COMPLEMENT CARRY TO MAKE IT FALSE
02F9 C9            909          RET
                   910 ;
                   911 ;*****************************************************************
                   912 ;
                   913 ; FUNCTION: RETT - RETURN TRUE
                   914 ; INPUTS: NONE
                   915 ; OUTPUTS: CARRY = 1 (TRUE)
                   916 ; CALLS: NOTHING
                   917 ; DESTROYS: CARRY
                   918 ; DESCRIPTION: RETT IS JUMPED TO BY ROUTINES WISHING TO RETURN TRUE.
                   919 ;                RETT SETS CARRY TO 1 AND RETURNS TO THE CALLER OF
                   920 ;                THE ROUTINE INVOKING RETT.
                   921 ;
                   922 RETT:
02FA 37            923          STC             ; SET CARRY TRUE
02FB C9            924          RET
                   925 ;
                   926 ;*****************************************************************
                   927 ;
                   928 ; FUNCTION: RGLOC - GET REGISTER SAVE LOCATION
                   929 ; INPUTS: NONE
                   930 ; OUTPUTS: HL - REGISTER SAVE LOCATION
                   931 ; CALLS: NOTHING
                   932 ; DESTROYS: B,C,H,L,F/F'S
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   933 ; DESCRIPTION:  RGLOC RETURNS THE SAVE LOCATION OF THE REGISTER
                   934 ;               INDICATED BY THE CURRENT REGISTER POINTER VALUE.
                   935 ;
                   936 RGLOC:
02FC 2AFD20        937         LHLD    RGPTR   ; GET REGISTER POINTER
02FF 2600          938         MVI     H,0     ; /IN H & L
0301 01ED03        939         LXI     B,RGTBL ; GET REGISTER SAVE LOCATION TABLE ADDRESS
0304 09            940         DAD     B       ; POINTER INDEXES TABLE
0305 6E            941         MOV     L,M     ; GET LOW ORDER BYTE OF REGISTER SAVE LOC.
0306 2620          942         MVI     H,(RAMST SHR 8) ; GET HIGH ORDER BYTE OF
                   943                                 ; /REGISTER SAVE LOCATION
0308 C9            944         RET
                   945 ;
                   946 ;*****************************************************************
                   947 ;
                   948 ; FUNCTION: RGNAM - DISPLAY REGISTER NAME
                   949 ; INPUTS: NONE
                   950 ; OUTPUTS: NONE
                   951 ; CALLS: OUTPT
                   952 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   953 ; DESCRIPTION: RGNAM DISPLAYS, IN THE ADDRESS FIELD OF THE DISPLAY,
                   954 ;              THE REGISTER NAME CORRESPONDING TO THE CURRENT
                   955 ;              REGISTER POINTER VALUE.
                   956 ;
                   957 RGNAM:
0309 2AFD20        958         LHLD    RGPTR   ; GET REGISTER POINTER
030C 2600          959         MVI     H,0
030E 29            960         DAD     H       ; MULTIPLY POINTER VALUE BY 4
030F 29            961         DAD     H       ;/(REGISTER NAME TABLE HAS 4 BYTE ENTRIES)
0310 01B903        962         LXI     B,NMTBL ; GET ADDRESS OF START OF REGISTER NAME TABLE
0313 09            963         DAD     B       ; ARG - ADD TABLE ADDRESS TO POINTER - RESULT IS
                   964                         ;/ADDRESS OF APPROPRIATE REGISTER NAME IN H & L
0314 AF            965         XRA     A       ; ARG - USE ADDRESS FIELD OF DISPLAY
0315 0600          966         MVI     B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
0317 CDB702        967         CALL    OUTPT   ; OUTPUT REGISTER NAME TO ADDRESS FIELD
031A C9            968         RET
                   969 ;
                   970 ;*****************************************************************
                   971 ;
                   972 ; FUNCTION: RSTOR - RESTOR USER REGISTERS
                   973 ; INPUTS: NONE
                   974 ; OUTPUTS: NONE
                   975 ; CALLS: NOTHING
                   976 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   977 ; DESCRIPTION:  RSTOR RESTORES ALL CPU REGISTERS, FLIP/FLOPS,
                   978 ;               INTERRUPT STATUS, INTERRUPT MASK, STACK POINTER
                   979 ;               AND PROGRAM COUNTER FROM THEIR RESPECTIVE
                   980 ;              SAVE LOCATIONS IN MEMORY. BY RESTORING THE PROGRAM
                   981 ;              COUNTER, THE ROUTINE EFFECTIVELY TRANSFERS CONTROL TO
                   982 ;              THE ADDRESS IN THE PROGRAM COUNTER SAVE LOCATION.
                   983 ;
                   984 ;               THE TIMING OF THIS ROUTINE IS CRITICAL TO THE
                   985 ;               CORRECT OPERATION OF THE SINGLE STEP ROUTINE.
                   986 ;               IF ANY MODIFICATION CHANGES THE NUMBER OF CPU
                   987 ;               STATES NEEDED TO EXECUTE THIS ROUTINE THEN THE
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   988 ;                TIMER VALUE MUST BE ADJUSTED BY THE SAME NUMBER.
                   989 ;
                   990 ; ***** THIS IS ALSO THE ENTRY POINT FOR THE TTY MONITOR
                   991 ;       TO RESTORE REGISTERS.
                   992 ;
                   993 RSTOR:
 031B 3AF120       994          LDA     ISAV     ; GET USER INTERRUPT MASK
 031E F618         995          ORI     18H      ; ENABLE SETTING OF INTERRUPT MASK AND
                   996                           ; /RESET IV3C FLIP FLOP
 0320 30           997          SIM              ; RESTORE USER INTERRUPT MASK
                   998 ;        RESTORE USER INTERRUPT STATUS
 0321 3AF120       999          LDA     ISAV     ; GET USER INTERRUPT MASK
 0324 E608        1000          ANI     08H      ; SHOULD USER INTERRUPTS BE ENABLED ?
 0326 CA2D03      1001          JZ      RSR05    ; NO - LEAVE INTERRUPTS DISABLED
 0329 FB          1002          EI               ; YES - ENABLE INTERRUPTS FOR USER PROGRAM
 032A C33103      1003          JMP     RSR10
                  1004 RSR05:
 032D 37          1005          STC              ; DUMMY INSTRUCTIONS - WHEN SINGLE STEP ROUTINE
 032E D23103      1006          JNC     RSR10    ; /IS BEING USED, THE TIMER IS RUNNING AND
                  1007                           ; /EXECUTE TIME FOR THIS ROUTINE MUST NOT
                  1008                           ; /VARY.
                  1009 RSR10:
 0331 21E920      1010          LXI     H,MNSTK  ; SET MONITOR STACK POINTER TO START OF STACK
 0334 F9          1011          SPHL             ; /WHICH IS ALSO END OF REGISTER SAVE AREA
 0335 D1          1012          POP     D        ; RESTORE REGISTERS
 0336 C1          1013          POP     B
 0337 F1          1014          POP     PSW
 0338 2AF420      1015          LHLD    SSAV     ; RESTORE USER STACK POINTER
 033B F9          1016          SPHL
 033C 2AF220      1017          LHLD    PSAV
 033F E5          1018          PUSH    H        ; PUT USER PROGRAM COUNTER ON STACK
 0340 2AEF20      1019          LHLD    LSAV     ; RESTORE H & L REGISTERS
 0343 C9          1020          RET              ; JUMP TO USER PROGRAM COUNTER
                  1021 ;
                  1022 ;*****************************************************************************
                  1023 ;
                  1024 ; FUNCTION: SETRG - SET REGISTER POINTER
                  1025 ; INPUTS: NONE
                  1026 ; OUTPUTS: CARRY - SET IF CHARACTER FROM KEYBOARD IS A REGISTER DESIGNAT
                  1027 ;                 RESET OTHERWISE
                  1028 ; CALLS: RDKBD
                  1029 ; DESTROYS: A,B,C,H,L,F/F'S
                  1030 ; DESCRIPTION: SETRG READS A CHARACTER FROM THE KEYBOARD. IF THE
                  1031 ;              CHARACTER IS A REGISTER DESIGNATOR, IT IS CONVERTED TO
                  1032 ;              THE CORRESPONDING REGISTER POINTER VALUE, THE POINTER IS
                  1033 ;              SAVED, AND THE FUNCTION RETURNS 'TRUE'. OTHERWISE, THE
                  1034 ;              FUNCTION RETURNS 'FALSE'.
                  1035 ;
                  1036 SETRG:
 0344 CDE702      1037          CALL    RDKBD    ; READ FROM KEYBOARD
 0347 FE10        1038          CPI     10H      ; IS CHARACTER A DIGIT?
 0349 D2F702      1039          JNC     RETF     ; NO - RETURN FALSE - CHARACTER IS NOT A
                  1040                           ; /REGISTER DESIGNATOR
 034C D603        1041          SUI     3        ; YES - TRY TO CONVERT REGISTER DESIGNATOR TO
                  1042                           ; / INDEX INTO REGISTER POINTER TABLE
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                  1043                        ; WAS CONVERSION SUCCESSFUL?
034E DAF702       1044          JC      RETF  ; NO - RETURN FALSE
0351 4F           1045          MOV     C,A   ; INDEX TO B & C
0352 0600         1046          MVI     B,0   ;
0354 21AC03       1047          LXI     H,RGPTB ; GET ADDRESS OF REGISTER POINTER TABLE
0357 09           1048          DAD     B     ; INDEX POINTS INTO TABLE
0358 7E           1049          MOV     A,M   ; GET REGISTER POINTER FROM TABLE
0359 32FD20       1050          STA     RGPTR ; SAVE REGISTER POINTER
035C C3FA02       1051          JMP     RETT  ; RETURN TRUE
                  1052 ;
                  1053 ;*****************************************************************
                  1054 ;
                  1055 ; FUNCTION: UPDAD - UPDATE ADDRESS FIELD OF DISPLAY
                  1056 ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT AT RIGHT EDGE OF FIELD
                  1057 ;                       0 MEANS NO DOT
                  1058 ; OUTPUTS: NONE
                  1059 ; CALLS: HXDSP,OUTPT
                  1060 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  1061 ; DESCRIPTION: UPDAD UPDATES THE ADDRESS FIELD OF THE DISPLAY USING
                  1062 ;              THE CURRENT ADDRESS.
                  1063 ;
                  1064 UPDAD:
035F 2AF620       1065          LHLD    CURAD ; GET CURRENT ADDRESS
0362 EB           1066          XCHG          ; ARG - PUT CURRENT ADDRESS IN D & E
0363 CD6C02       1067          CALL    HXDSP ; EXPAND CURRENT ADDRESS FOR DISPLAY
                  1068                        ; ARG - ADDRESS OF EXPANDED ADDRESS IS IN H & L
0366 AF           1069          XRA     A     ; ARG - USE ADDRESS FIELD OF DISPLAY
                  1070                        ; ARG - DOT FLAG IS IN B
0367 CDB702       1071          CALL    OUTPT ; OUTPUT CURRENT ADDRESS TO ADDRESS FIELD
036A C9           1072          RET
                  1073 ;
                  1074 ;*****************************************************************
                  1075 ;
                  1076 ; FUNCTION: UPDDT - UPDATE DATA FIELD OF DISPLAY
                  1077 ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT AT RIGHT EDGE OF FIELD
                  1078 ;                       0 MEANS NO DOT
                  1079 ; OUTPUTS: NONE
                  1080 ; CALLS: HXDSP,OUTDT
                  1081 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  1082 ; DESCRIPTION: UPDDT UPDATES THE DATA FIELD OF THE DISPLAY USING
                  1083 ;              THE CURRENT DATA BYTE.
                  1084 ;
                  1085 UPDDT:
036B 3AF820       1086          LDA     CURDT ; GET CURRENT DATA
036E 57           1087          MOV     D,A   ; ARG - PUT CURRENT DATA IN D
036F CD6C02       1088          CALL    HXDSP ; EXPAND CURRENT DATA FOR DISPLAY
                  1089                        ; ARG - ADDRESS OF EXPANDED DATA IS IN H & L
0372 3E01         1090          MVI     A,DTFLD ; ARG - USE DATA FIELD OF DISPLAY
                  1091                        ; ARG - DOT FLAG IS IN B
0374 CDB702       1092          CALL    OUTPT ; OUTPUT CURRENT DATA TO DATA FIELD
0377 C9           1093          RET
                  1094 ;
                  1095 ;*****************************************************************
                  1096 ;
                  1097 ;                       MONITOR TABLES
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                  1098 ;
                  1099 ;*******************************************************************
                  1100 ;
                  1101 ; COMMAND TABLE
                  1102 ;   COMMAND CHARACTERS AS RECEIVED FROM KEYBOARD
                  1103 CMDTB:
0378 12           1104          DB       12H      ; GO COMMAND
0379 13           1105          DB       13H      ; SUBSTITUTE MEMORY COMMAND
037A 14           1106          DB       14H      ; EXAMINE REGISTERS COMMAND
037B 15           1107          DB       15H      ; SINGLE STEP COMMAND
0004              1108 NUMC     EQU      $-CMDTB ; NUMBER OF COMMANDS
                  1109 ;
                  1110 ;*******************************************************************
                  1111 ;
                  1112 ; COMMAND ROUTINE ADDRESS TABLE
                  1113 ; (MUST BE IN REVERSE ORDER OF COMMAND TABLE)
                  1114 CMDAD:
037C FD00         1115          DW       SSTEP    ; ADDRESS OF SINGLE STEP ROUTINE
037E 9200         1116          DW       EXAM     ; ADDRESS OF EXAMINE REGISTERS ROUTINE
0380 8B01         1117          DW       SUBST    ; ADDRESS OF SUBSTITUTE MEMORY ROUTINE
0382 CB00         1118          DW       GOCMD    ; ADDRESS OF GO ROUTINE
                  1119 ;
                  1120 ;*******************************************************************
                  1121 ;
                  1122 DSPTB:  ; TABLE FOR TRANSLATING CHARACTERS FOR OUTPUT
                  1123 ;
                  1124 ;               DISPLAY
                  1125 ;               FORMAT   CHARACTER
                  1126 ;               =======  =========
                  1127 ;
0000              1128 ZERO     EQU      $ - DSPTB
0384 F3           1129          DB       0F3H     ; 0
0385 60           1130          DB        60H     ; 1
0386 B5           1131          DB       0B5H     ; 2
0387 F4           1132          DB       0F4H     ; 3
0388 66           1133          DB        66H     ; 4
0005              1134 FIVE     EQU      $ - DSPTB
0005              1135 LETRS    EQU      $ - DSPTB
0389 D6           1136          DB       0D6H     ; 5 AND S
038A D7           1137          DB       0D7H     ; 6
038B 70           1138          DB        70H     ; 7
0008              1139 EIGHT    EQU      $ - DSPTB
038C F7           1140          DB       0F7H     ; 8
038D 76           1141          DB        76H     ; 9
000A              1142 LETRA    EQU      $ - DSPTB
038E 77           1143          DB        77H     ; A
000B              1144 LETRB    EQU      $ - DSPTB
038F C7           1145          DB       0C7H     ; B (LOWER CASE)
000C              1146 LETRC    EQU      $ - DSPTB
0390 93           1147          DB        93H     ; C
000D              1148 LETRD    EQU      $ - DSPTB
0391 E5           1149          DB       0E5H     ; D (LOWER CASE)
000E              1150 LETRE    EQU      $ - DSPTB
0392 97           1151          DB        97H     ; E
000F              1152 LETRF    EQU      $ - DSPTB
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0393 17           1153          DB        17H     ; F
0010              1154 LETRH    EQU      $ - DSPTB
0394 67           1155          DB        67H     ; H
0011              1156 LETRL    EQU      $ - DSPTB
0395 83           1157          DB        83H     ; L
0012              1158 LETRP    EQU      $ - DSPTB
0396 37           1159          DB        37H     ; P
0013              1160 LETRI    EQU      $ - DSPTB
0397 60           1161          DB        60H     ; I
0014              1162 LETRR    EQU      $ - DSPTB
0398 05           1163          DB        05H     ; R (LOWER CASE)
0015              1164 BLANK    EQU      $ - DSPTB
0399 00           1165          DB        00H     ; BLANK
                  1166 ;
                  1167 ;*****************************************************************
                  1168 ;
                  1169 ; MESSAGES FOR OUTPUT TO DISPLAY
                  1170 ;
039A 15           1171 BLNKS:   DB       BLANK,BLANK,BLANK,BLANK ; FOR ADDRESS OR DATA FIELD
039B 15
039C 15
039D 15
039E 15           1172 ERMSG:   DB       BLANK,LETRE,LETRR,LETRR ; ERROR MESSAGE FOR ADDR. FIELD
039F 0E
03A0 14
03A1 14
03A2 0E           1173 EXMSG:   DB       LETRE,BLANK,BLANK,BLANK ; EXECUTION MESSAGE
03A3 15
03A4 15
03A5 15
                  1174                                           ; /FOR ADDRESS FIELD
03A6 15           1175 SGNAD:   DB       BLANK,BLANK,EIGHT,ZERO  ; SIGN ON MESSAGE (ADDR. FIELD)
03A7 15
03A8 08
03A9 00
03AA 08           1176 SGNDT:   DB       EIGHT,FIVE              ; SIGN ON MESSAGE (DATA FIELD)
03AB 05
                  1177 ;
                  1178 ;*****************************************************************
                  1179 ;
                  1180 RGPTB:   ; REGISTER POINTER TABLE
                  1181                    ; THE ENTRIES IN THIS TABLE ARE IN THE SAME ORDER
                  1182                    ; AS THE REGISTER DESIGNATOR KEYS ON THE KEYBOARD.
                  1183                    ; EACH ENTRY CONTAINS THE REGISTER POINTER VALUE WHICH
                  1184                    ; CORRESPONDS TO THE REGISTER DESIGNATOR. REGISTER
                  1185                    ; POINTER VALUES ARE USED TO POINT INTO THE REGISTER
                  1186                    ; NAME TABLE (NMTBL) AND REGISTER SAVE LOCATION
                  1187                    ; TABLE (RGTBL).
                  1188 ;
03AC 06           1189          DB       6         ; INTERRUPT MASK
03AD 09           1190          DB       9         ; SPH
03AE 0A           1191          DB       10        ; SPL
03AF 0B           1192          DB       11        ; PCH
03B0 0C           1193          DB       12        ; PCL
03B1 07           1194          DB       7         ; H
```

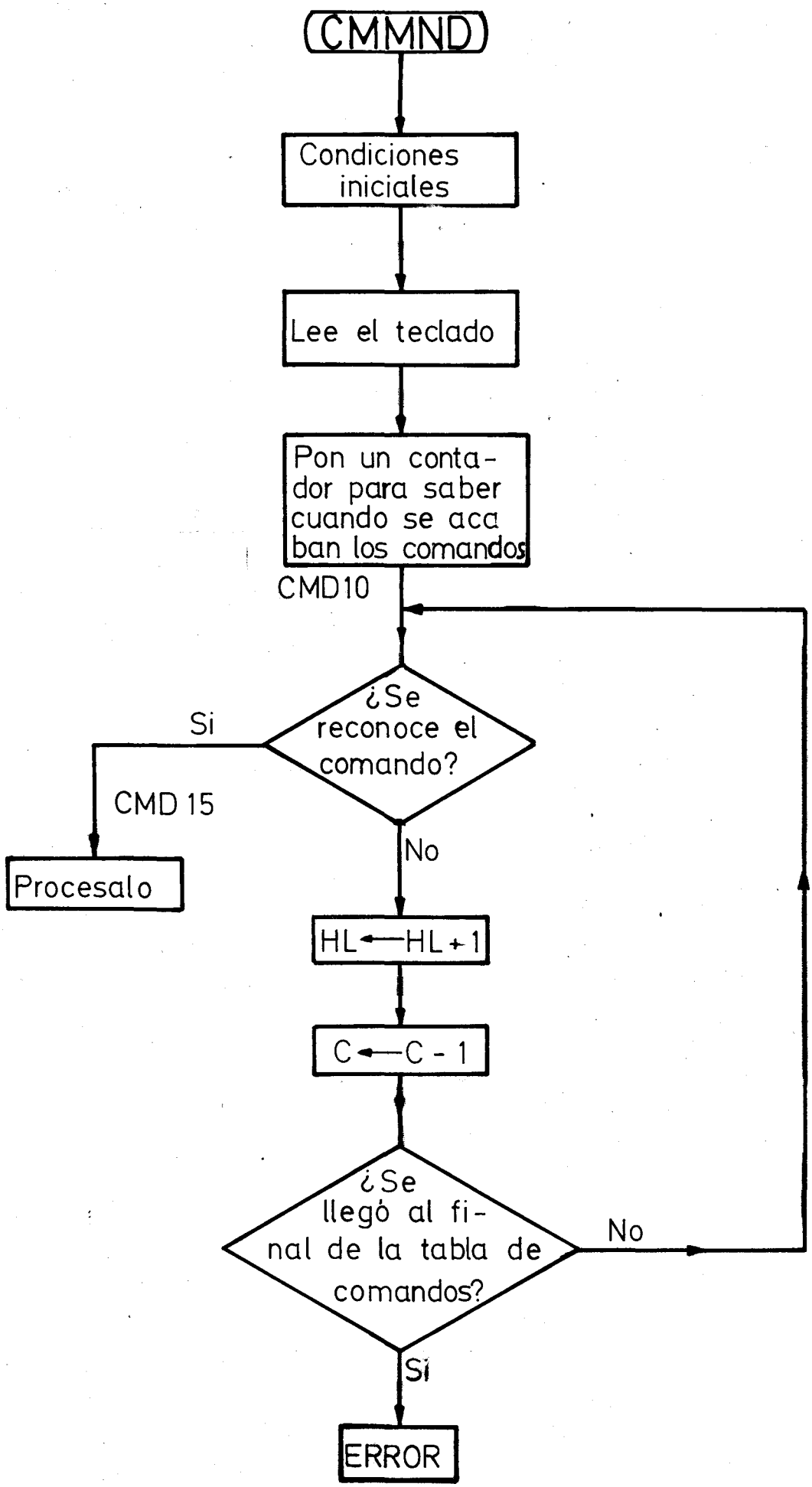
```
LOC  OBJ          SEQ          SOURCE STATEMENT

03B2 08           1195          DB      8       ; L
03B3 00           1196          DB      0       ; A
03B4 01           1197          DB      1       ; B
03B5 02           1198          DB      2       ; C
03B6 03           1199          DB      3       ; D
03B7 04           1200          DB      4       ; E
03B8 05           1201          DB      5       ; FLAGS
                  1202 ;
                  1203 ;*****************************************************************
                  1204 ;
                  1205 NMTBL:                   ; REGISTER NAME TABLE
                  1206                          ; NAMES OF REGISTERS IN DISPLAY FORMAT
03B9 15           1207          DB      BLANK,BLANK,BLANK,LETRA  ; A REGISTER
03BA 15
03BB 15
03BC 0A
03BD 15           1208          DB      BLANK,BLANK,BLANK,LETRB  ; B REGISTER
03BE 15
03BF 15
03C0 0B
03C1 15           1209          DB      BLANK,BLANK,BLANK,LETRC  ; C REGISTER
03C2 15
03C3 15
03C4 0C
03C5 15           1210          DB      BLANK,BLANK,BLANK,LETRD  ; D REGISTER
03C6 15
03C7 15
03C8 0D
03C9 15           1211          DB      BLANK,BLANK,BLANK,LETRE  ; E REGISTER
03CA 15
03CB 15
03CC 0E
03CD 15           1212          DB      BLANK,BLANK,BLANK,LETRF  ; FLAGS
03CE 15
03CF 15
03D0 0F
03D1 15           1213          DB      BLANK,BLANK,BLANK,LETRI  ; INTERRUPT MASK
03D2 15
03D3 15
03D4 13
03D5 15           1214          DB      BLANK,BLANK,BLANK,LETRH  ; H REGISTER
03D6 15
03D7 15
03D8 10
03D9 15           1215          DB      BLANK,BLANK,BLANK,LETRL  ; L REGISTER
03DA 15
03DB 15
03DC 11
03DD 15           1216          DB      BLANK,LETRS,LETRP,LETRH ; STACK POINTER HIGH ORDER BYTE
03DE 05
03DF 12
03E0 10
03E1 15           1217          DB      BLANK,LETRS,LETRP,LETRL  ; STACK POINTER LOW ORDER BYTE
03E2 05
03E3 12
```

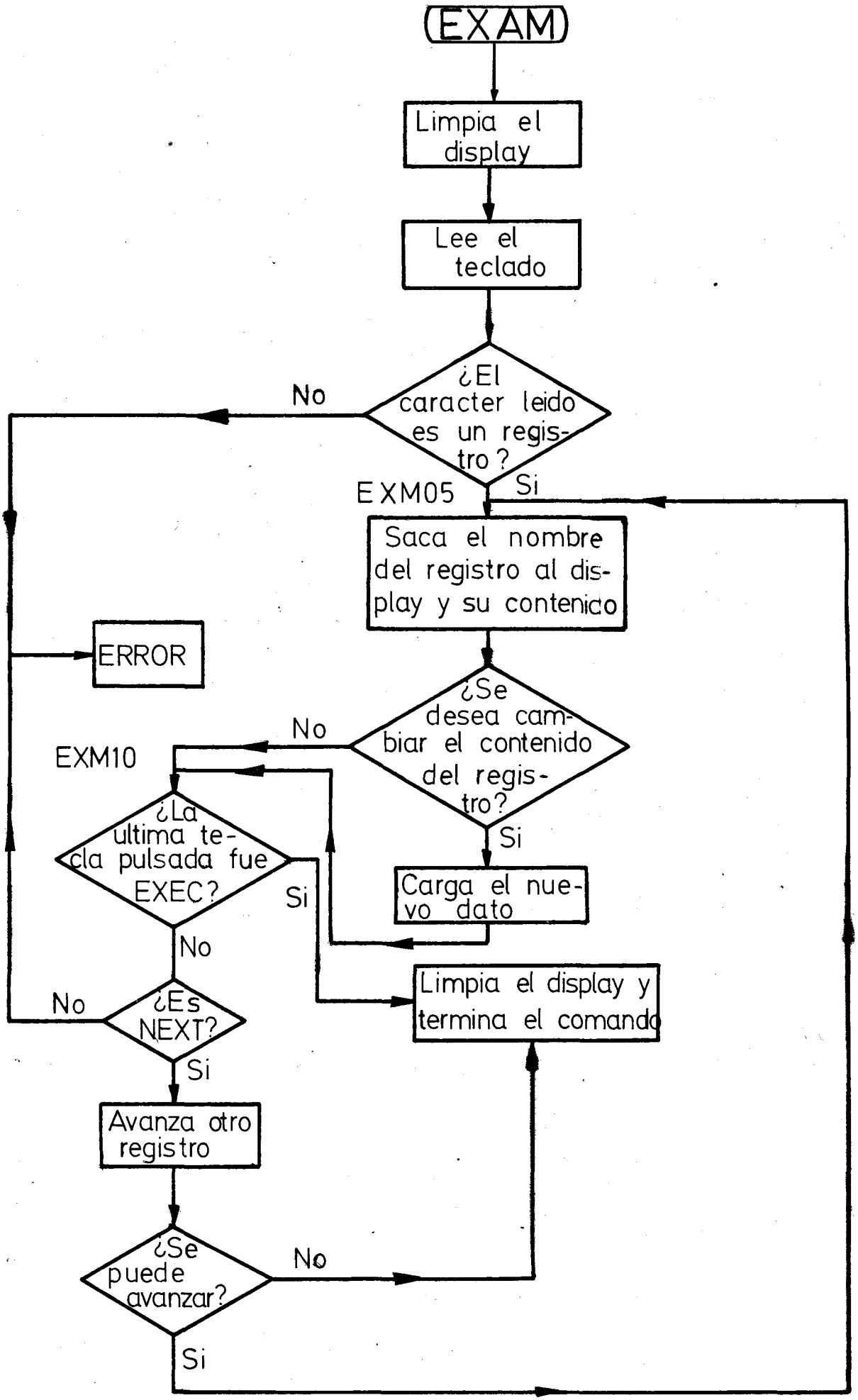
```
LOC  OBJ          SEQ          SOURCE STATEMENT

03E4 11
03E5 15           1218          DB      BLANK,LETRP,LETRC,LETRH  ; PROGRAM COUNTER HIGH BYTE
03E6 12
03E7 0C
03E8 10
03E9 15           1219          DB      BLANK,LETRP,LETRC,LETRL  ; PROGRAM COUNTER LOW BYTE
03EA 12
03EB 0C
03EC 11
                  1220 ;
                  1221 ;*************************************************************************
                  1222 ;
                  1223 ; REGISTER SAVE LOCATION TABLE
                  1224 ; ADDRESSES OF SAVE LOCATIONS OF REGISTERS IN THE ORDER IN WHICH
                  1225 ; THE REGISTERS ARE DISPLAYED BY THE EXAMINE COMMAND
                  1226 ;
                  1227 RGTBL:
03ED EE           1228          DB      ASAV AND OFFH    ; A REGISTER
03EE EC           1229          DB      BSAV AND OFFH    ; B REGISTER
03EF EB           1230          DB      CSAV AND OFFH    ; C REGISTER
03F0 EA           1231          DB      DSAV AND OFFH    ; D REGISTER
03F1 E9           1232          DB      ESAV AND OFFH    ; E REGISTER
03F2 ED           1233          DB      FSAV AND OFFH    ; FLAGS
03F3 F1           1234          DB      ISAV AND OFFH    ; INTERRUPT MASK
03F4 F0           1235          DB      HSAV AND OFFH    ; H REGISTER
03F5 EF           1236          DB      LSAV AND OFFH    ; L REGISTER
03F6 F5           1237          DB      SPHSV AND OFFH   ; STACK POINTER HIGH ORDER BYTE
03F7 F4           1238          DB      SPLSV AND OFFH   ; STACK POINTER LOW ORDER BYTE
03F8 F3           1239          DB      PCHSV AND OFFH   ; PROGRAM COUNTER HIGH ORDER BYTE
03F9 F2           1240          DB      PCLSV AND OFFH   ; PROGRAM COUNTER LOW ORDER BYTE
000D              1241 NUMRG    EQU     ($ - RGTBL)      ; NUMBER OF ENTRIES IN
                  1242                                   ; /REGISTER SAVE LOCATION TABLE
                  1243 ;
                  1244 ;*************************************************************************
                  1245 ;*************************************************************************
                  1246 ;
```

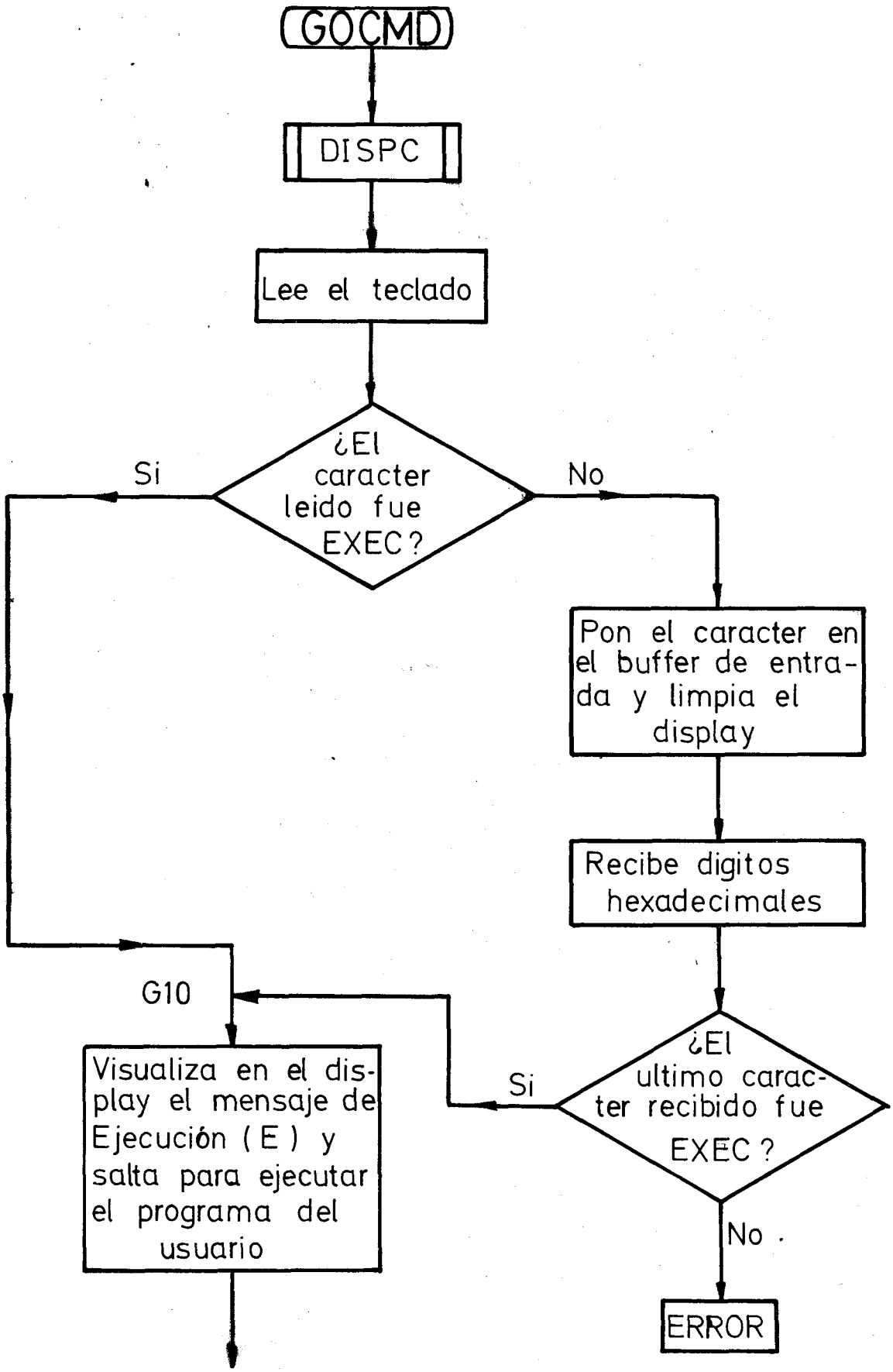
PLANOS

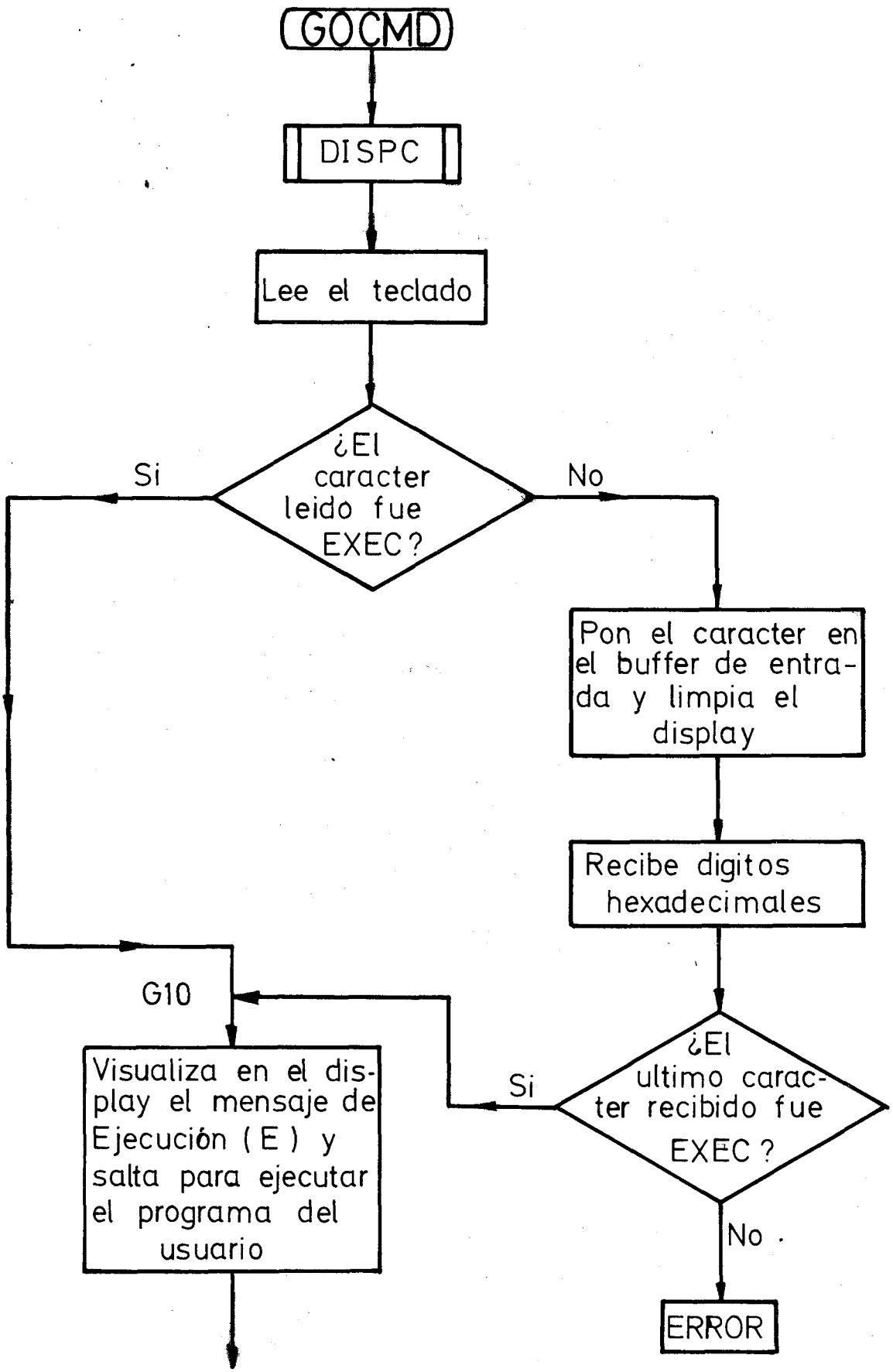
ANEXO 3

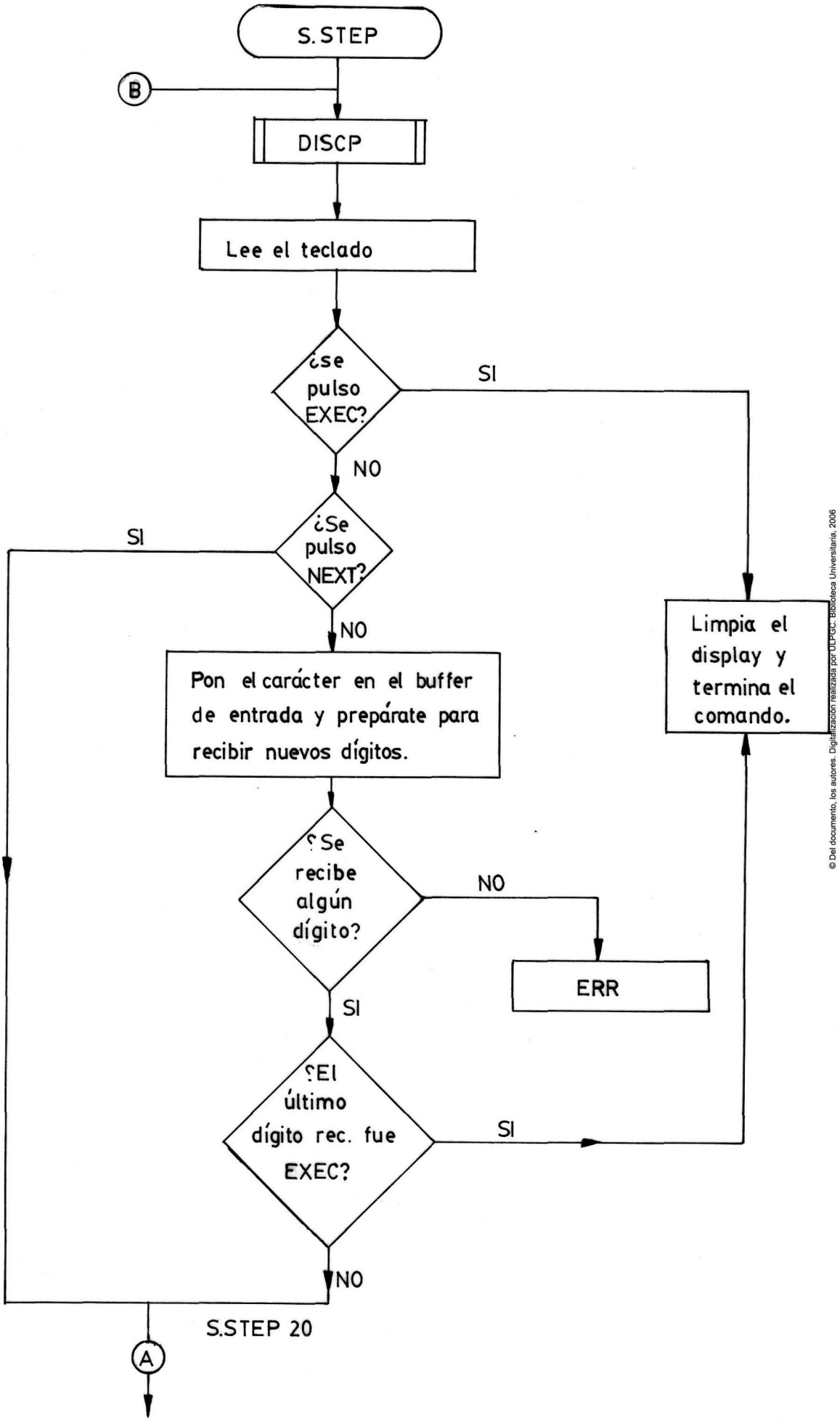
Planos y organigramas.

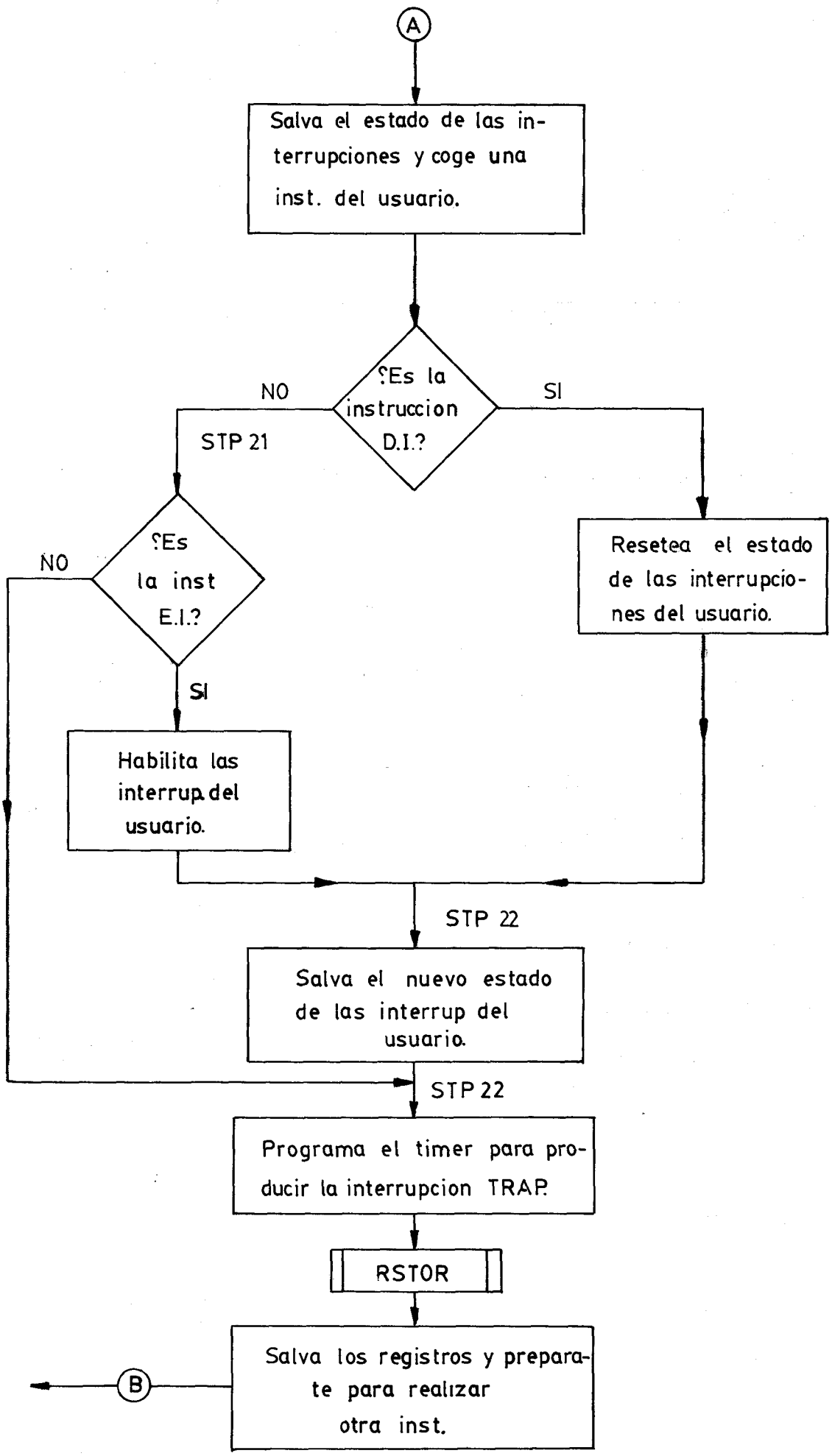


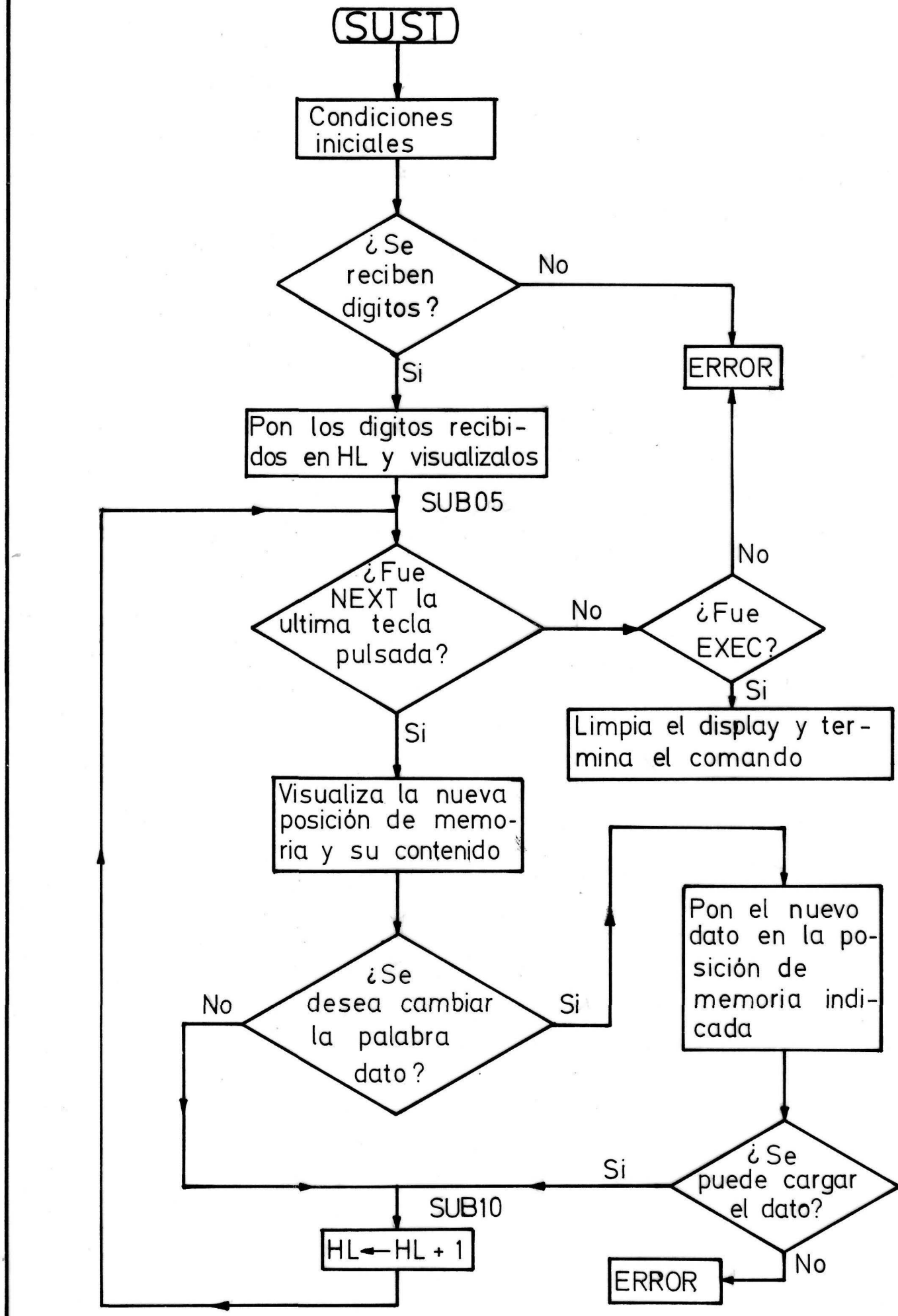


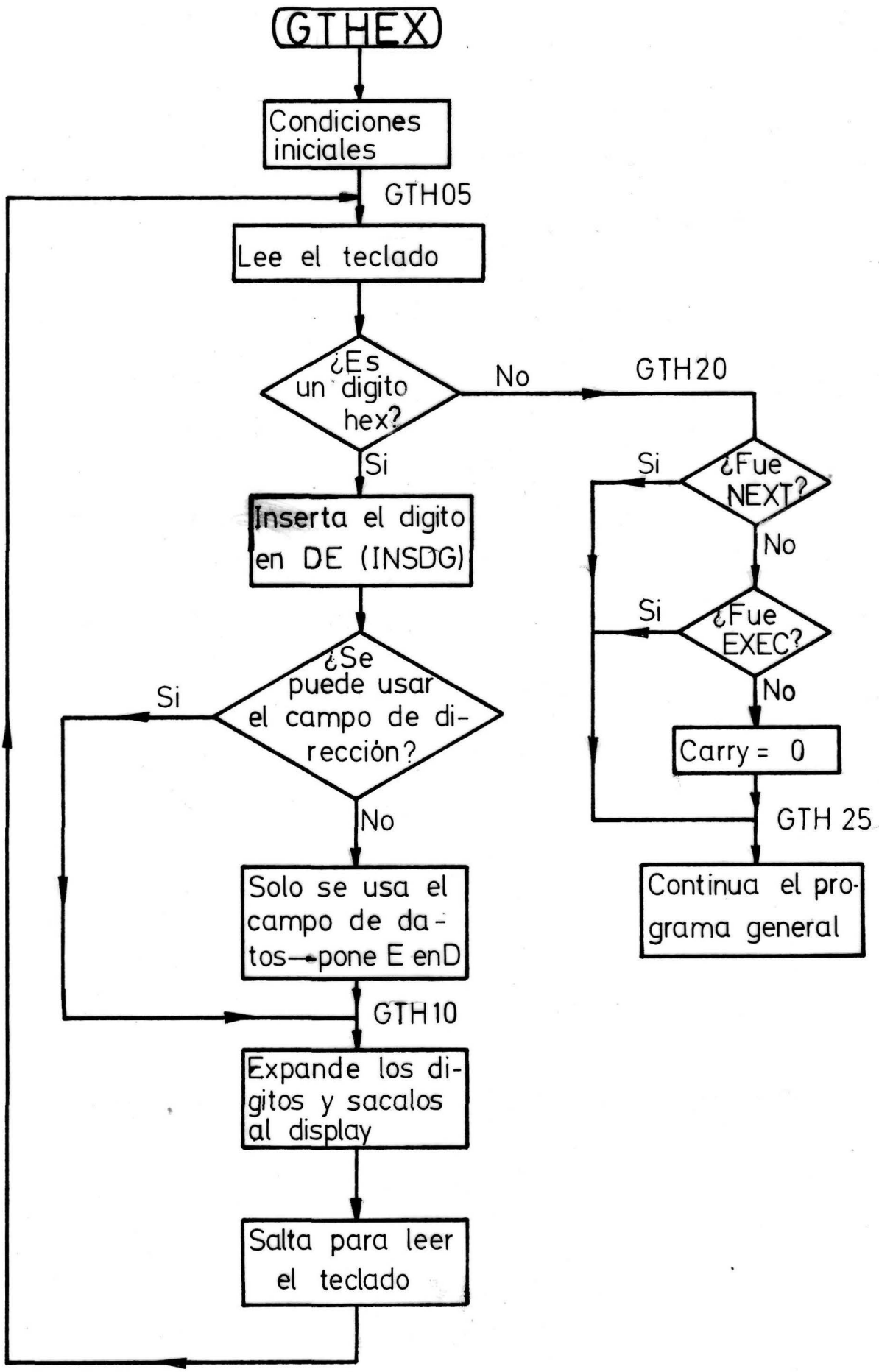


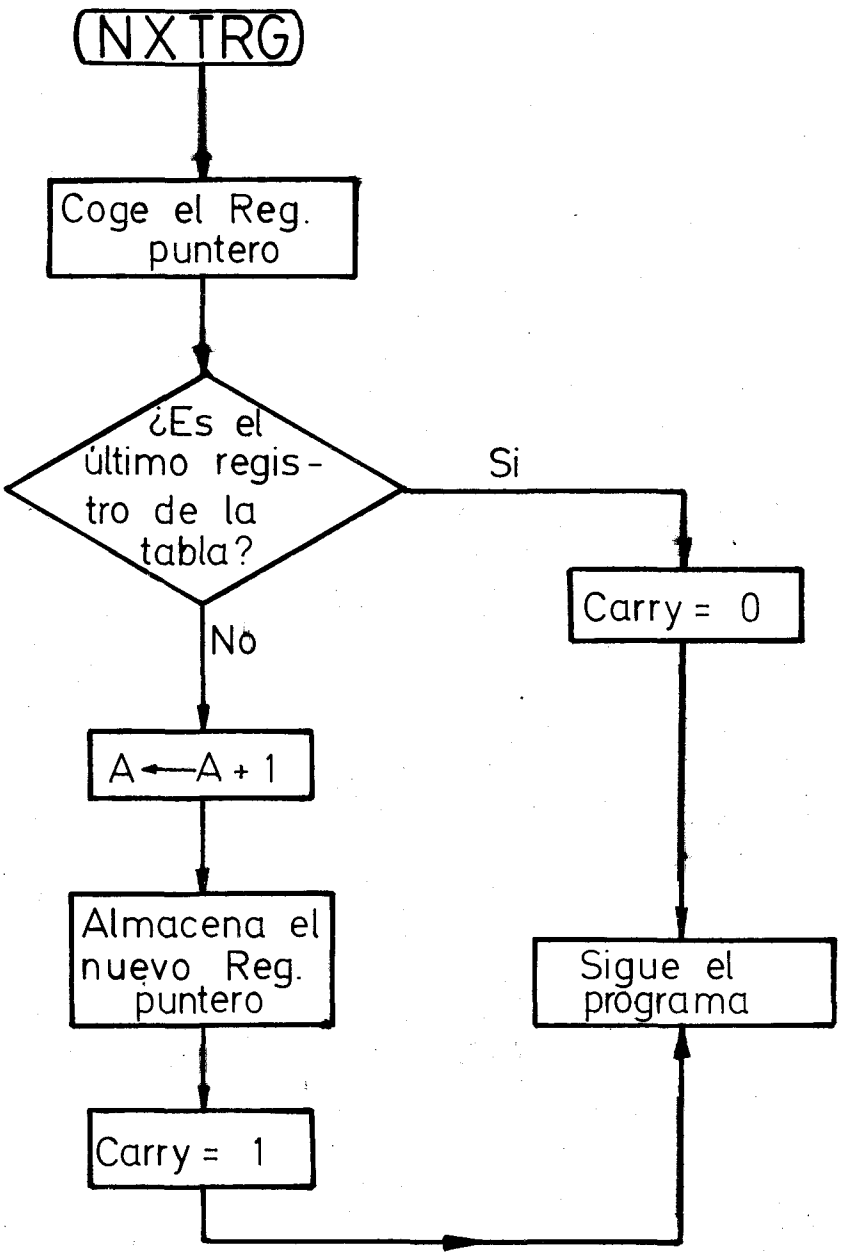


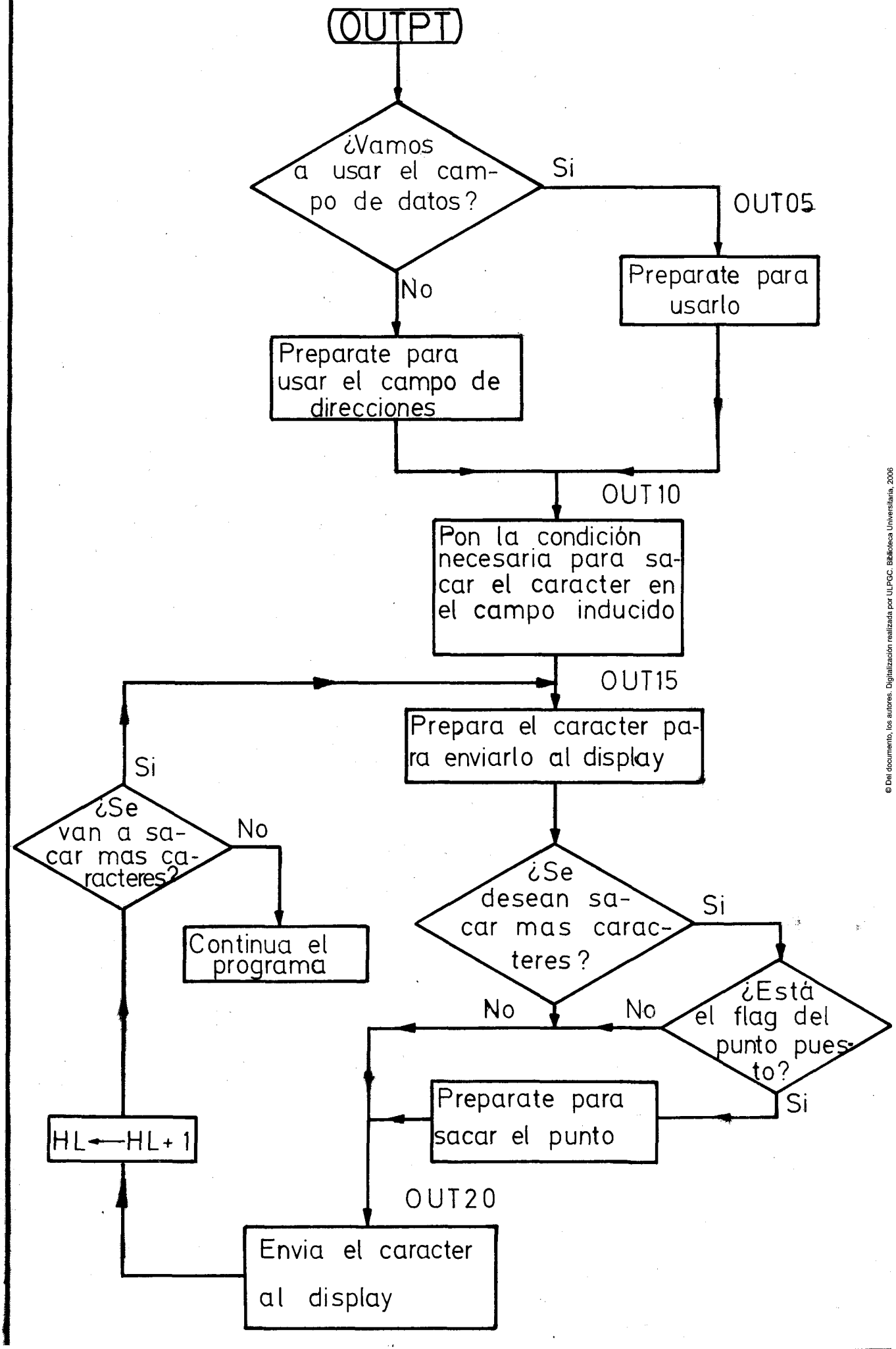


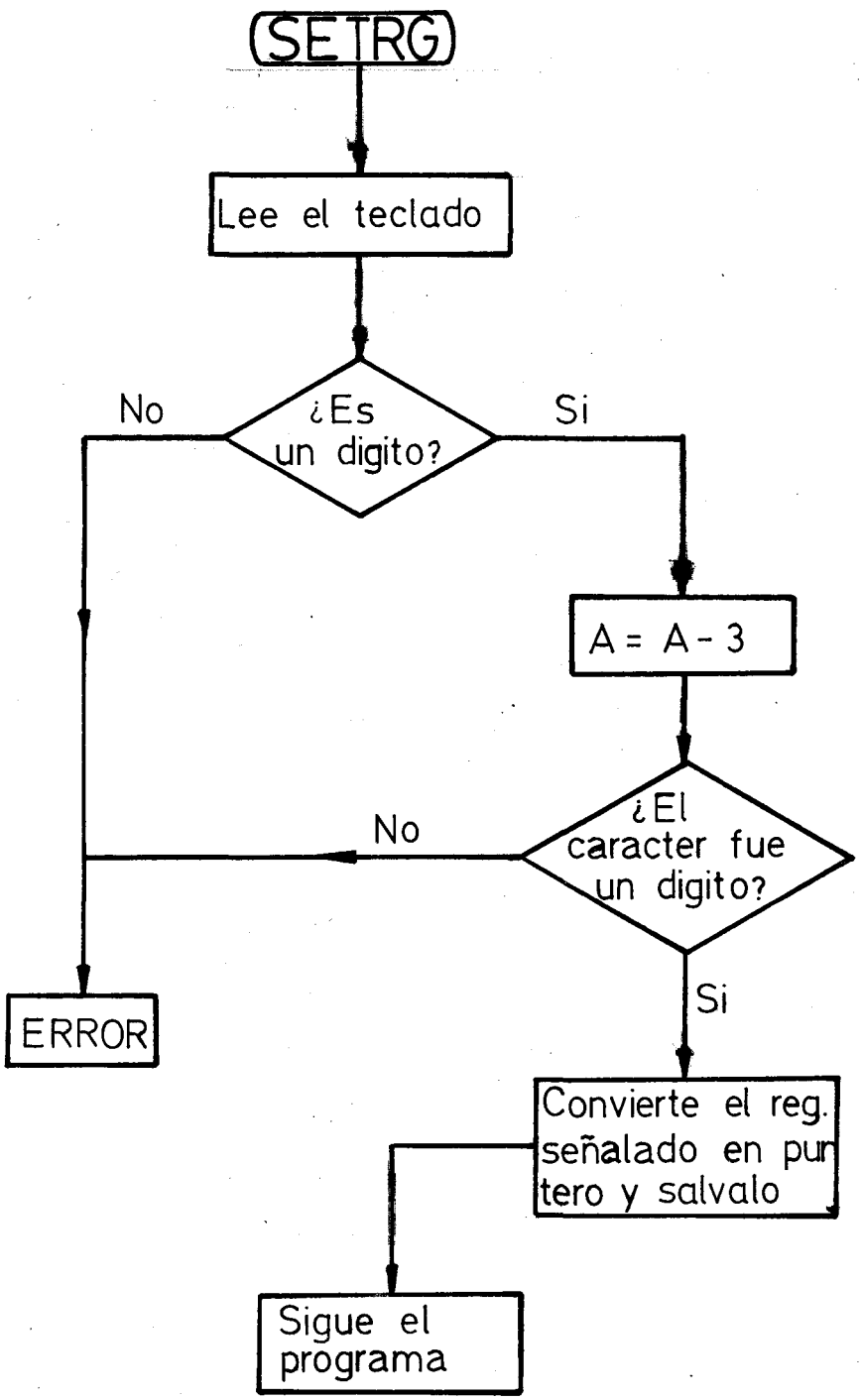


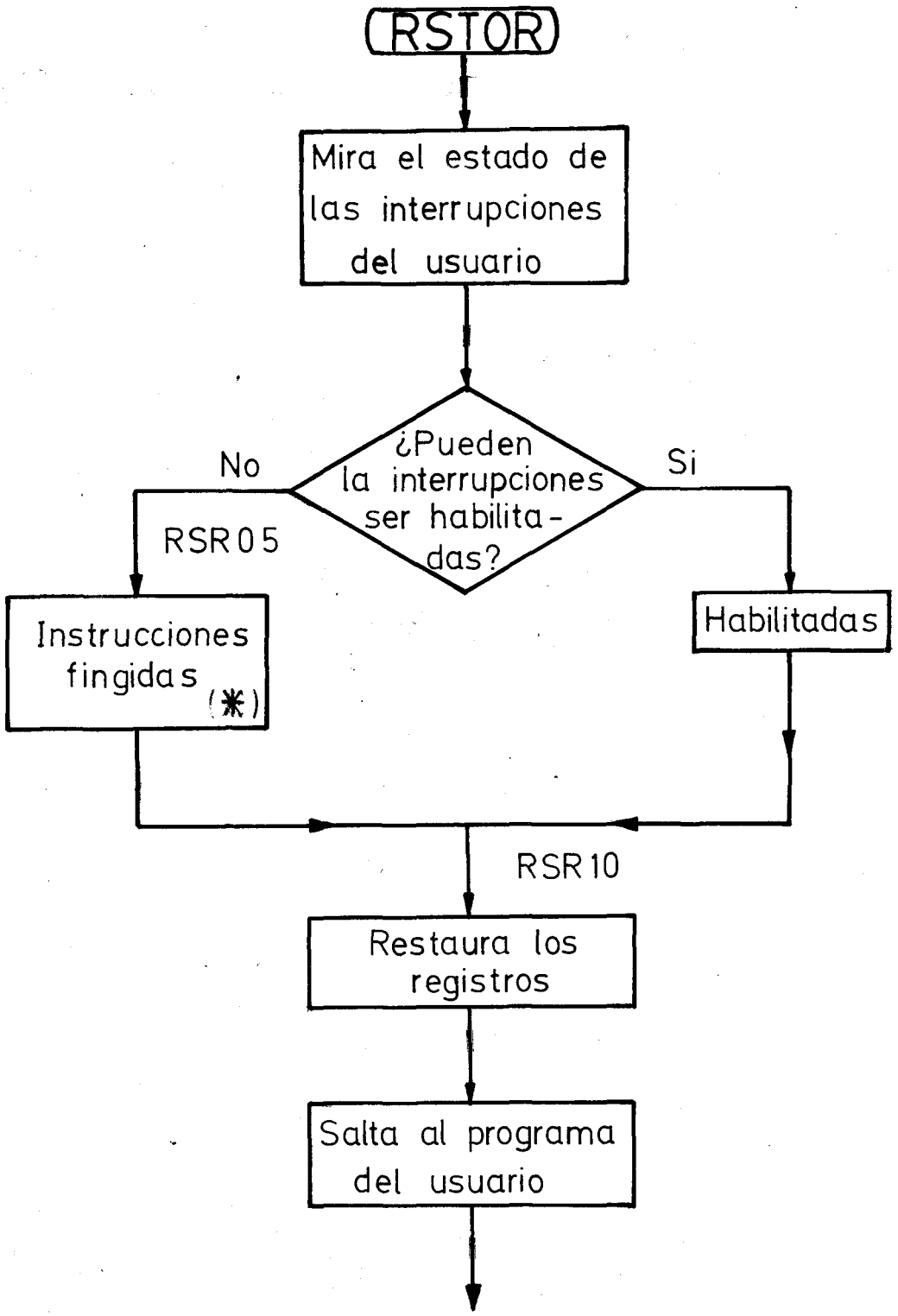




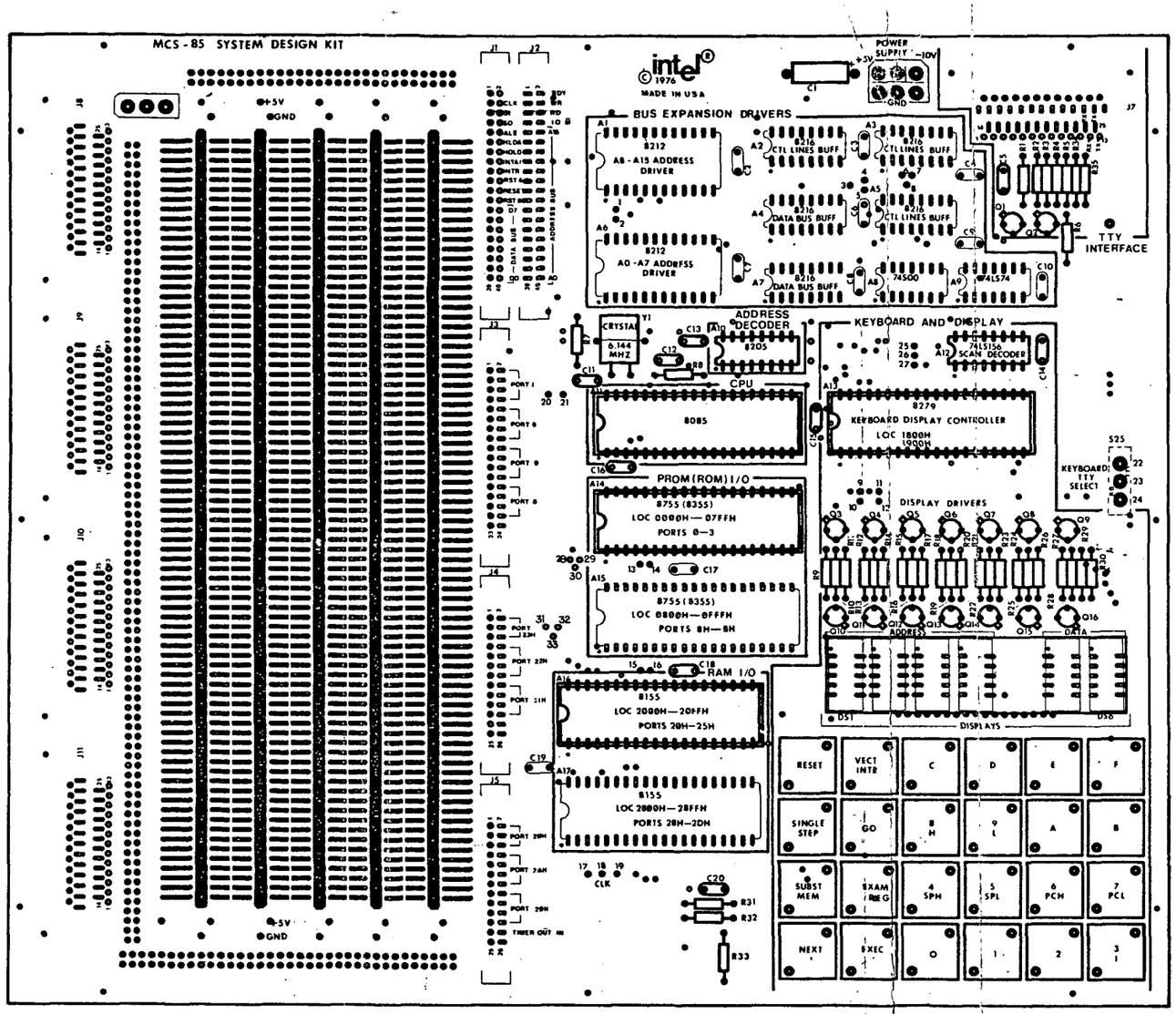








* La instrucciones que se realizan por este camino son para conseguir que la función tarde el mismo tiempo por el "No" que por el "Si".



ITEM NO.	PART NUMBER	DESCRIPTION
QUANTITY PER DASH NO.		PARTS LIST

SCALE: NONE

TOL. ± | 2 PLC | 3 PLC | ANGLE

MATERIAL:

FINISH:

intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

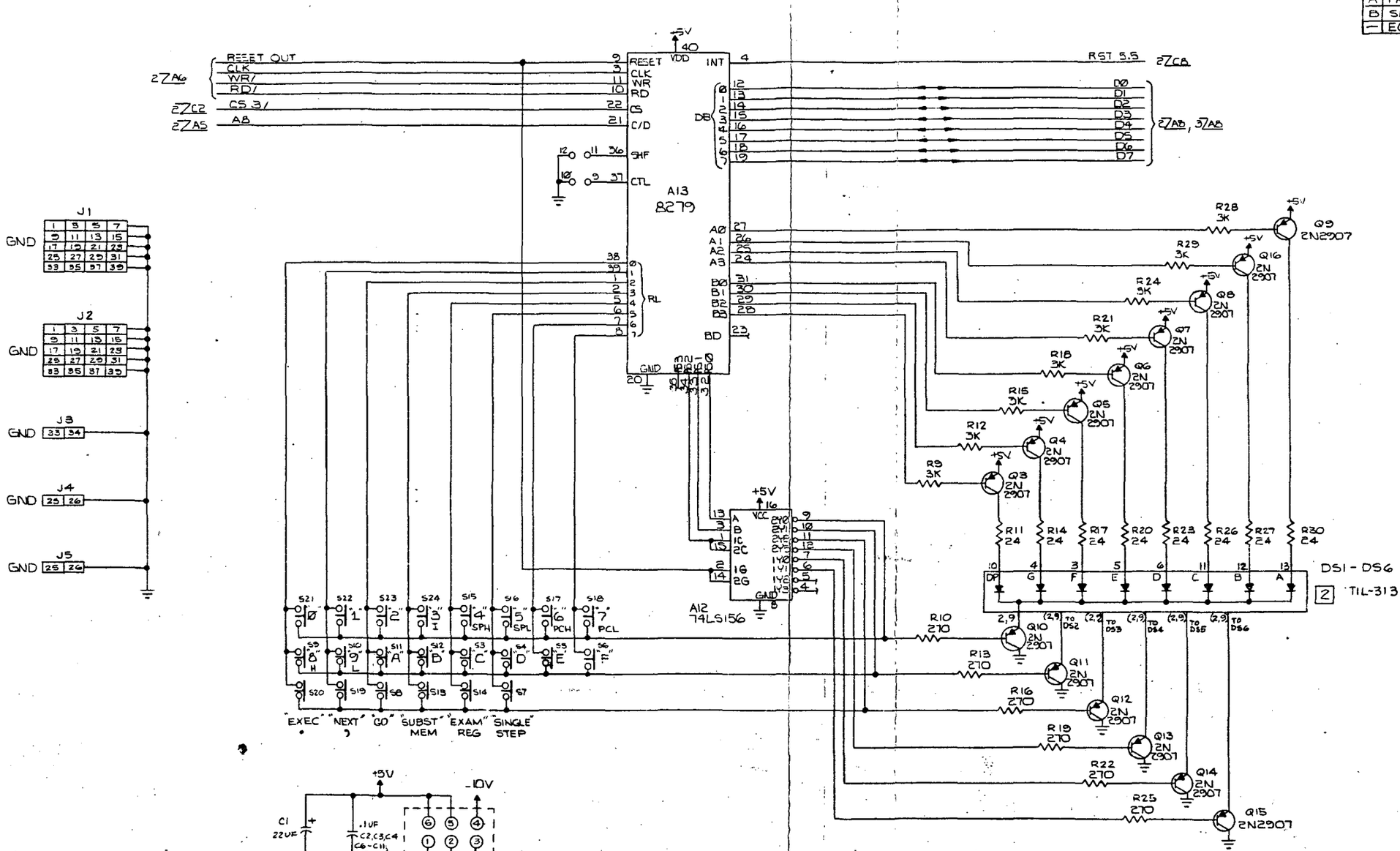
TITLE: PWA SYSTEM DESIGN KIT

SIZE	DEPT	DRAWING NO.	REV
D	416	1001119	A

NEXT ASSY	USED ON
4001117	SDK 85

CODE: | SHEET 1 OF 1

REVISIONS				
LTR	DESCRIPTION	SIGNATURE AND DATE DFT	CHK	ENGR
A	PROD REL	5/77	5/27/77	
B	SEE ECO # 01132-2	9/27	9/27/77	
—	ECO 2-3151			



NOTES:
UNLESS OTHERWISE SPECIFIED,

1. RESISTOR VALUES ARE IN OHMS, ¼W, ±5%.

2 THIS PIN OUT ARRANGEMENT REPRESENTS 6 INDIVIDUAL 7 SEGMENT LED DISPLAYS (TIL 313). ALL ANODE CONNECTIONS OF THE CORRESPONDING SEGMENTS ARE WIRED TOGETHER. CATHODES ARE WIRED SEPARATELY TO EACH TRANSISTOR (Q10-Q15)

3 JUMPER 5 TO 3 AND 6 TO 8 WHEN BUS EXPANSION CIRCUITS ARE NOT INSTALLED.

SPARE GATE COUNT		
TYPE	REF DES	QTY.

REF DESIGNATIONS	
USED	NOT USED
C20	
A17	
R35	
Q16	
DS6	
33	
S25	

4001117 NEXT ASSY | SDK 85 USED ON

QUANTITY PER DASH NO.

ITEM NO.	PART NUMBER	DESCRIPTION

PARTS LIST

SCALE: NONE

TOL ±	2 PLC	3 PLC	ANGLE

MATERIAL:

FINISH:

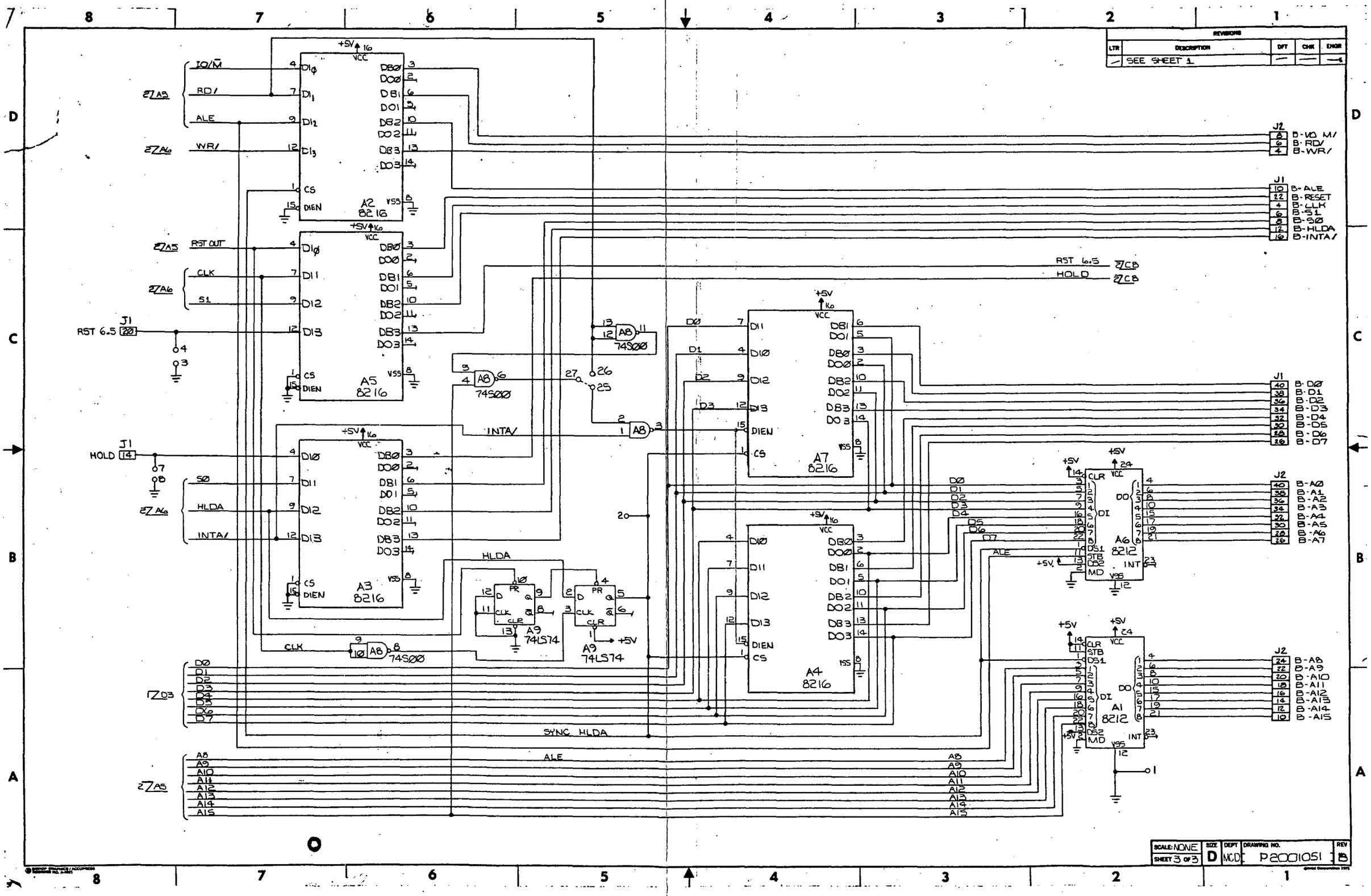
	SIGNATURE	DATE
DRN BY		8/76
CHK BY		12/15/76
DFT APPD		1/5/76
ENGR APPD	A. Chang	1/5/77
AUTH BY		6/1/77
CODE:	SHEET 1 OF 3	

intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

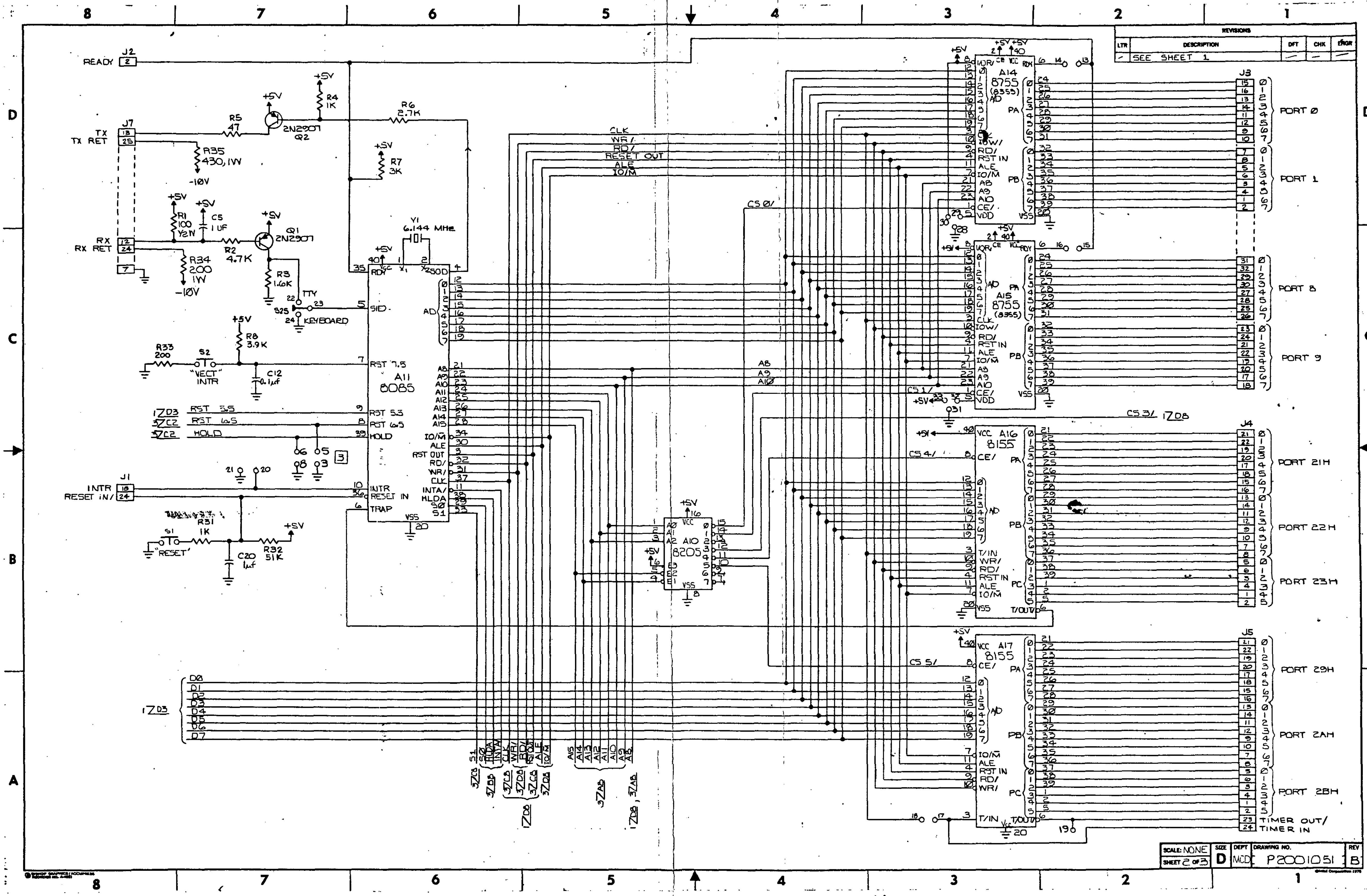
TITLE: SCHEMATIC SYSTEM DESIGN KIT

SIZE	DEPT	DRAWING NO.	REV
D	MCD	P2001051	B

REVISIONS				
LTR	DESCRIPTION	DFT	CHK	ENGR
—	SEE SHEET 1	—	—	—



SCALE: NONE	SIZE	DEPT	DRAWING NO.	REV
SHEET 3 OF 3	D	MCD	P2001051	B



BIBLIOGRAFIA

.- MANUAL DEL SDK '85.

.- PROCESADORES PROGRAMABLES. EL MICROPROCESADOR.

+ Enrique Mandado.

.- DEL MICROPROCESADOR AL MICROORDENADOR.

+ H. Lilen.

.- MICROPROCESADORES Y MICROCOMPUTADORES.

+ Serie: Mundo Electronico.

.- MANUAL DE CARACTERISTICAS DE INTEL.

.- CURSO BASICO DE MICROPROCESADORES. (Tomo 3).

.....