SHARP

COMPUTADORA DE BOLSILLO

MODELO PC-1500

MANUAL DE INSTRUCCIONES

NOTA INTRODUCTORIA

Permítanos agradecerle haber comprado la computadora de bolsillo PC-1500. Nosotros vida cotidiana. La PC-1500 es una de las computadoras de mano más sofisticadas del mundo. Aunque comparte muchas características con la computadora de bolsillo PC-1211, la PC-1500 le provee con capacidades tan avanzadas como:

- 7 por 156 puntos de matriz programable en la representación visual LCD.
- Un generador de tonos para crear efectos especiales programable.
- El conjunto de caracteres ASCII con letras mayúsculas y minúsculas.
- Funciones matemáticas y científicas.
- Teclas de función definida para el usuario.
- Una versión extendida del lenguaje BASIC de la computadora que provee ordenamientos bidimensionales, longitud variable de las series de caracteres, mandatos gráficos, programación en cadena, y muchas otras atracciones especiales.
- Hasta 4K y 8K bytes de RAM opcional.
- Un interfaz/impresor (modelo CE-150) opcional, que permite 4 colores en las coordenades X-Y, impresión de programas y datos en uno de los nueve caracteres de tamaño diferente y grabación de los mismos en cinta magnética.

Esta máquina es capaz de muchas de las funciones que solo hace unos pocos años habría llenado un almacén con tubos, cables, e ingenieros. Para utilizar esta sofisticación no requiere credenciales de ingeniería. Por el contrario, la PC-1500, y este manual son designados para ayudarle a ganar un acceso rápido a esta nueva tecnología.

Hemos dividido este manual en cinco secciones permitiendo que el usuario sin experiencia gane competencia rápidamente. Los usuarios avanzados pueden explorar estas atracciones especiales de la PC-1500 a través de los Capítulos de Programación Avanzada y Cálculos Avanzados, y a través de Apéndices.

El estilo de este manual es conversacional y muchos ejemplos son provistos. Pero no nos crea demasiado, para ver qué fácil es empezar, vaya al Capítulo O. Pero primero, asegúrese que las baterías han sido cargadas. Si no lo han sido el Apéndice B provee las instrucciones.

¡Sobre todo, diviértase y no vacile en experimentar!

NOTAS OPERACIONALES

Gracias por haber comprado la Computadora de Bolsillo SHARP PC-1500.

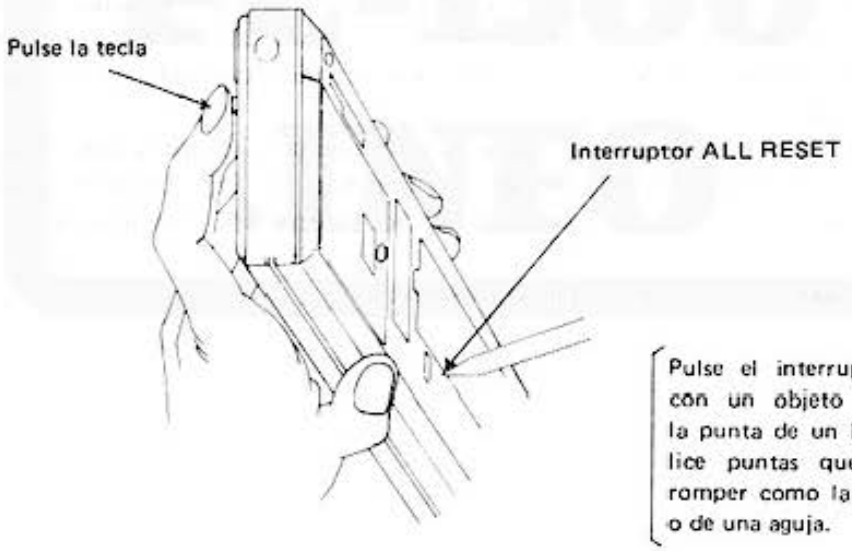
Como el cristal líquido de la representación visual de la computadora es de vidrio, debe de tratarla con cuidado. No ponga la PC-1500 en el bolsillo de atrás pues puede dañarla cuando se siente.

Para asegurar una operación sin problemas de su computadora de bolsillo le recomendamos que:

1. Guarde la computadora en lugares sin cambios bruscos de temperatura, humedad o polvo.
2. Utilice un paño seco para limpiar la computadora. No utilice jabones, agua o paños húmedos.
3. Para evitar gasto de baterías, quite las baterías cuando la computadora no va a ser utilizada por un período largo de tiempo.
4. Si requiere reparaciones, la computadora puede llevarse a un centro de reparaciones SHARP solamente.
5. Este manual debe ser guardado como referencia.

Cómo Evitar Problemas

Esta unidad, si es expuesta a un ruido exterior bastante fuerte o un impacto durante su operación, puede dejar todas sus teclas incluyendo la tecla BREAK ON (break/on) inoperables.
Si esto ocurriese, pulse el interruptor ALL RESET en la parte de strás de la unidad por 15 segundos, pulsando la tecla BREAK ON al mismo tiempo.



Pulse el interruptor ALL RESET con un objeto puntiagudo como la punta de un bolígrafo. No utilice puntas que sean fáciles de romper como la punta de un lápiz o de una aguja.

Después, compruebe que [NEW∅ ? : CHECK], es indicado en la representación gráfica, y pulse las teclas de [CL] [N] [E] [W] [∅] [ENTER]. (∅ = cero, tecla numérica).
Si la representación gráfica no lee NEW∅? : CHECK, ejecute la operación mencionada arriba otra vez. Con esta operación, el programa, datos y todos los contenidos reservados se anulan. Así es que no intente pulsar el interruptor ALL RESET excepto en caso de problemas.

ESPECIFICACIONES DE LA PC-1500

Modelo:	Computadora de bolsillo PC-1500
Número de dígitos para cálculos:	10 digitos (mantisa) + 2 digitos (exponente)
Sistema de cálculo:	Según la fórmula matemática (con la función de juzgamiento prioritaria).
Lenguaje del programa:	BASIC
Capacidad:	CPU: CMOS 8 bit Sistema ROM: 16K bytes Capacidad de memoria (RAM): 3,5K bytes Area del sistema: 0,9K bytes Memoria intermedia de entrada: 80 bytes Memoria de retención temporal: 196 bytes Otras: Area del usuario: 2,6K bytes Area de la memoria fija: 624 bytes (A–Z, A$–Z$) Area de datos del programma BASIC: 1850 bytes Area de reserva: 188 bytes
Cálculacos:	Cuatro operaciones aritméticos, cálculos de potencia, funciones trigonométricas inversas, funciones logarímicas y exponenciales, conversión angular, extracción de la raíz cuadrada, funcion de signos valores absolutos y enteros, y cálculos lógicos.
Función editora:	Cursor cambiante (▶◀) Inserción (INS) Supresión (DEL) Línea hacia arriba y hacia abajo (↓ , ↑)
Protección de la memoria:	Batería CMOS de soporte. (Programa, datos y memorias de reserva protegidas).
Representatión gráfica:	Cristal líquido Anchura de 26 caracteres 7 x 156 puntos gráficos
Teclas:	65 teclas, incluyendo: Alfabéticas Numéricas Función definible por el usuario Pre-programada
Electricidad:	6,0 V, c.c.: 4 pilas secas (tipo: AA o R6)
Consumo de electricidad:	6,0 V, c.c.: 0,13 W
Tiempo de operación:	50 horas aproximadamente con pilas (tipo AA o R6)
Temperatura de operación:	0 a 40°C
Dimensiones:	195 (ancho) x 86 (largo) x 25.5 (altura) mm
Peso:	375 g aproximadamente (con baterías)
Accesorios:	Estuche, 4 baterías secas, dos plantillas, etiqueta de nombre y manual de instrucciones
Opciones:	Impresor/Interfaz de cassette (CE-150) Módulo de expansión de la memoria (tipo de enchufe, 4K byte RAM CE-151, 8K byte RAM CE-155)

TABLA DE CONTENIDOS

Etiqueta de nombre
Escriba su nombre en la etiqueta adjunta y péguela en la parte trasera de la computadora.

PROGRAMACION INSTANTANEA

(Como el café descafeinado)

Esta sección está dedicada exclusivamente a un grupo selecto de gente (incluyendo los autores) cuya curiosidad es mayor que su paciencia (y quizás más que su sentido común). Para aquellas personas que deben hacer algo con el milagro de la electrónica moderna, les presentamos un ejercicio de programación sencilla. (Aviso: Aconsejamos a las personas tímidas o débiles de corazón que pasen al Capitulo 1, familiarizándose, para una introducción más exhaustiva y sosegada de la computadora SHARP PC-1500.)

Antes de continuar, otro aviso más. Es necesario **seguir todos los pasos en el orden dado.** En contra de la opinión popular, las computadoras no son "super-cerebros" y no tiene la habilidad humana de adivinar lo que usted desea. La computadora PC-1500 simplemente espera sus instrucciones y las ejecuta. ¿Está usted preparado? Bien, ¡Comencemos!

Ejemplo 1

Primero encuentre la tecla escrita con la palabra "ON" en la esquina derecha del teclado. Al pulsar esta tecla el genio electrónico que está dormido se despertará. (No espere una bocanada de humo). La representación visual de la computadora debe ser similar a la ilustración inferior:

Pulse la tecla [MODE] (en el lado derecho) hasta que la abreviatura PRO aparezca en la parte superior derecha de la representación (si pulsa la tecla demasiadas veces, púlsela otra vez hasta que el resultado deseado sea obtenido). La computadora SHARP PC-1500 está preparada para aceptar la serie de instrucciones que componen un programa de computación.

Pulsando las teclas correspondientes:

[1] [Ø] [A] [=] [1] [ENTER]

Observe que al pulsar la tecla [ENTER] , la computadora modificará lo que usted ha escrito. Ahora, la representación visual debe aparecer así:

Nota: A lo largo de este manual utilizaremos Ø para designar el número cero, para que usted pueda distinguir entre la letra O y el número Ø.

Continúe pulsando las teclas. No se alarme si cada renglón desaparece al escribir el renglón siguiente.

[2] [Ø] [P] [A] [U] [S] [E] [A] [ENTER]
[3] [Ø] [A] [=] [A] [+] [2] [ENTER]
[4] [Ø] [G] [O] [T] [O] [2] [Ø] [ENTER]

En este instante su primer programa está completo y usted debe decirle a la computadora que

"ejecute", o lleve a cabo, las instrucciones que contiene. Este proceso se llama "correr" el programa, y se realiza en el modo RUN. Pulse el botón MODE una vez más y las letras PRO serán reemplazadas por las letras RUN encima de la representación visual.

Un último paso: escriba palabra [R] [U] [N] y pulse [ENTER].

¡Felicitaciones! Su primer programa en BASIC está siendo ahora ejecutado. Sus instrucciones son seguidas y la computadora está ocupada registrando todos los números impares positivos en orden.

"Muy bien," se dice a sí mismo, "soy un genio. ¿Pero cuando se parará?"

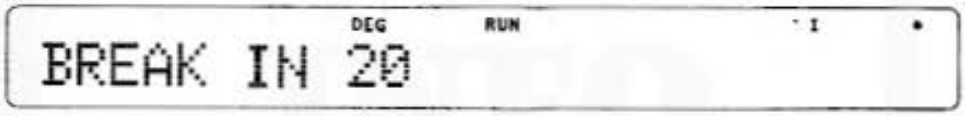
Bueno, . . . desafortunadamente, sin su intervención o falla de pilas, este programa particular nunca acabará. Para ver porqué, vamos a revisar el programa:

```
10  A = 1
20  PAUSE A
30  A = A + 2
40  GOTO 20
```

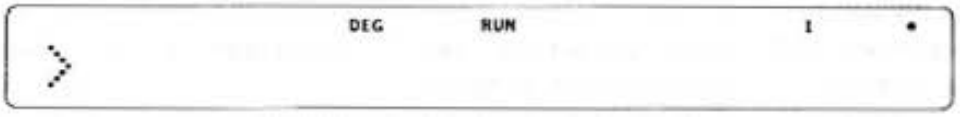
El efecto de la línea número 40 hace que la computadora re-ejecute todas las líneas posteriores a la línea número 20. Esto incluye la línea 40 que, por supuesto, le dice a la computadora que re-re-ejecute líneas 20, 30, 40 . . . . y así indefinidamente. Esta repetición sin fin es conocida en el lenguaje de computadoras como "looping" o un "bucle".

Nuestro programa está trabado en un "bucle infinito" y sólo usted tiene la habilidad de parar este trágico desgaste de las pilas. Para hacer esto, pulse la tecla [ON]. Como la PC-1500 ya está encendida, usted está seleccionando la función BREAK. No se alarme. A pesar de lo que el nombre parece indicar, esta tecla no es una tecla que se destruye a sí misma. Para su propio conocimiento, usted no puede de ninguna forma, dañar o perjudicar la computadora simplemente pulsando las teclas. **¡Entonces siéntase libre para experimentar!**

Si usted ha pulsado la tecla BREAK, un mensaje similar al siguiente es mostrado en el visor:



Este le informa que instrucción estaba siendo ejecutada, o trabajada, cuando usted interrumpió la computadora. Pulse la tecla BREAK una vez más, y la computadora estará lista para su próxima instrucción:



Para aquellas personas que acaban de recorbar que dejaron el grifo abierto en casa, esta es una buena oportunidad para parar. (Antes de dejar la computadora, para conservar las pilas, no se olvide de pulsar la tecla [OFF]). Los demás ya están convirtiéndose en adictos a la programación y querrán continuar su educación con nuestro segundo ejemplo. (Desde ahora nosotros no nos hacemos responsables si llega tarde a cenar con su mujer).

Ejemplo 2

Para empezar nuestro segundo programa, es necesario entrar el modo del programa pulsando la tecla MODE hasta que las letras PRO (abreviatura de la palabra PROgrama) reemplacen las letras RUN encima de la representación visual. La PC-1500 ahora le dejará ingresar un programa nuevo o

modificar el programa viejo. Debido a que nuestro programa nuevo no se construirá con las instrucciones de nuestro programa viejo, debemos quitar estas instrucciones de la memoria de la computadora. Para hacer esto, escriba la palabra NEW y pulse [ENTER]. Después de una pausa el carácter > volverá.

Pulse lo siguiente para entrar el primer renglón del programa:

[1] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [T] [A] [M] [A] [N] [O] [SPACE]

[D] [E] [SPACE] [L] [I] [S] [T] [A] [SHIFT] [/] [SHIFT] ["] [SHIFT] [+] [A] [ENTER]

Observe que al pulsar la tecla [SHIFT] seguida por una tecla que tiene otro carácter escrito encima de ella, entrará el carácter superior. La tecla [SHIFT] permite a dos caracteres compartir el mismo botón, y a veces también se llama "la segunda función" de la tecla. Así, en la primera línea de nuestro programa (línea 10 arriba), pulsando [SHIFT] seguido por el signo [+] causa que un [:] (carácter de punto y coma) sea entrado. La línea entera está almacenada en la computadora como:

```
                DEG        PRO          I    •
10:INPUT "TAMANO DE LISTA?
```

En este manual nosotros ilustraremos la selección del carácter de la segunda función con la tecla [SHIFT] y el carácter deseado. Por ejemplo, la línea numerada 10 arriba se mostrará así:

[1] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [T] [A] [M] [A] [N] [O] [SPACE]

[D] [E] [SPACE] [L] [I] [S] [T] [A] [SHIFT] [?] [SHIFT] [~] [SHIFT] [:] [A] [ENTER]

Complete nuestro segundo programa con las siguientes entradas:

[2] [Ø] [I] [F] [A] [SHIFT] [<] [=] [Ø] [G] [O] [T] [O] [9] [9] [ENTER]

[3] [Ø] [F] [O] [R] [I] [=] [1] [T] [O] [A] [ENTER]

[4] [Ø] [P] [A] [U] [S] [E] [I] [SHIFT] [,] [I] [*] [I] [ENTER]

[5] [Ø] [N] [E] [X] [T] [I] [ENTER]

[9] [9] [B] [E] [E] [P] [2] [SHIFT] [:] [E] [N] [D] [ENTER]

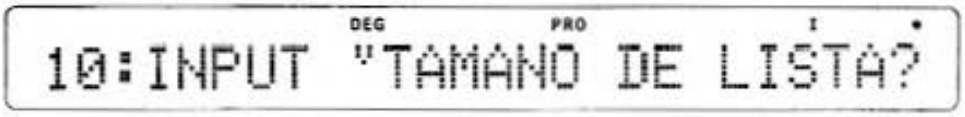
Nuestro segundo programa está ahora almacenado en la memoria de la PC-1500. ¿Se acuerda lo que debe hacer ahora? Si su respuesta es: "Correr el programa" usted está por convertirse en un buen programador.

Vuelva al modo RUN (Aviso: use la tecla [MODE]) y escriba la palabra RUN. Pulse [ENTER] para empezar la "ejecución" de nuestro segundo programa.

¿Está la computadora preguntándole de la manera siguiente? (Si no, vuelva al modo del PROgrama y verifique su escritura).

```
                DEG        RUN          I    •
TAMANO DE LISTA?_
```

¡Buena Actuación! Nuestro programa le está pidiendo a usted la información necesaria para funcionar. ¿Recuerda la primera línea de nuestro programa BASIC que usted escribió?



Las instrucciones en esta línea son seguidas por la computadora. El resultado es que la computadora está esperando que usted "INPUT" (escriba) alguna información.

Este programa imprimirá una lista de números y sus cuadrados (el número multiplicado por sí mismo). Primero, sin embargo, le preguntará a usted cuantos números y cuadrados hay que imprimir (TAMANO DE LISTA?). El usuario (usted otra vez) debe responder escribiendo un número y pulsando, (usted acertó), [ENTER].

Escriba [8] y pulse [ENTER].

Mire despacio como pares de números aparecen brevemente en la representación visual. El segundo número en el par (a la derecha) será el cuadrado del primero. **Ocho pares de números serán desplegados. Esto es lo que usted pidió cuando usted respondió a la pregunta del programa escribiendo el número ocho.**

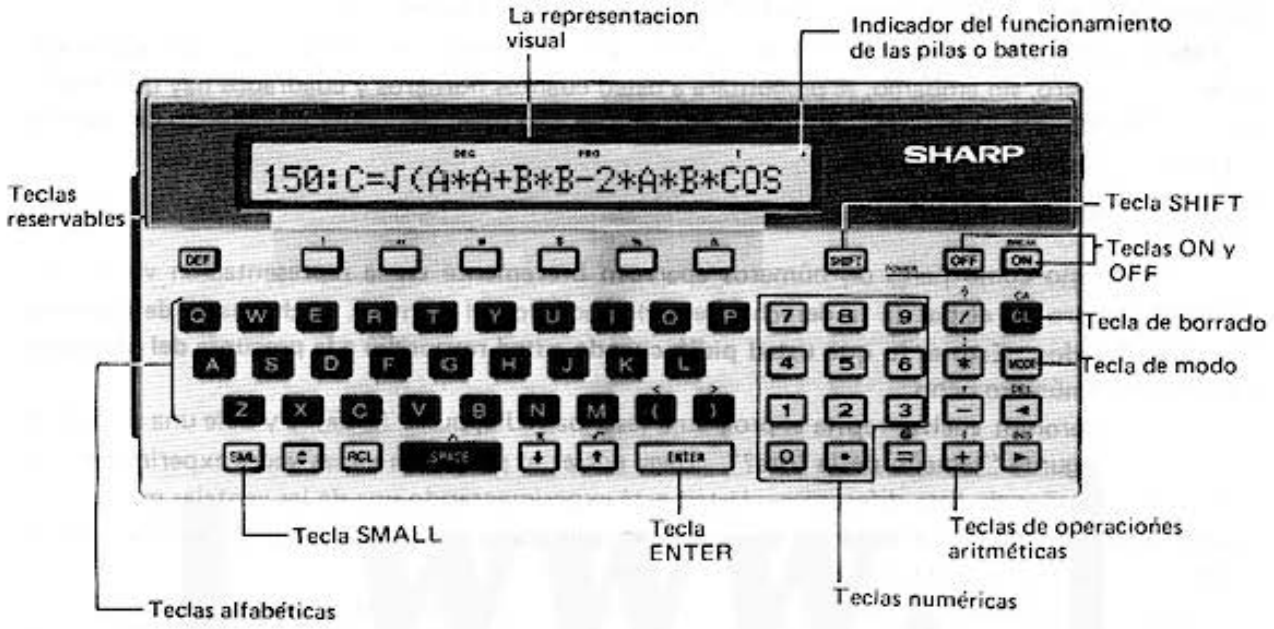
Cuando la prompt vuelva, repita el programa (escriba RUN, pulse [ENTER]) y déle una respuesta nueva a la pregunta "tamaño de la lista?". Haga correr el programa varias veces experimentando con varios tamaños de lista diferentes. Usted está experimentando una de las ventajas más importantes de las computadoras programables; ellas pueden realizar una tarea aburrida repetidamente, variándola ligeramente cada vez en respuesta al INPUT.

Ejecute el programa una vez más pero esta vez ponga un cero en el tamaño de lista. ¿Qué pasa? El resultado es que el programa termina sin haber producido una lista. Aunque esto parezca extraño, la PC-1500 sencillamente está siguiendo nuestras instrucciones.

Esto demuestra por qué la computadora es una herramienta tan poderosa. Puede ser programada para seguir grupos distintos de instrucciones y procesar la variedad de información dada. Debido a las instrucciones en la línea 20, si su entrada es cero (o menos), la computadora ignora la computación de la lista y va al final del programa. En efecto, ha hecho una decisión basada en la entrada que usted solicitó. Como programador, usted controla qué decisiones son posibles y cuándo son hechas. De esta manera, el poder entero de la computadora está a su disposición para resolver sus problemas específicos y en la manera que usted cree es la mejor.

I. FAMILIARIZANDOSE

Después que usted desenvuelva su Computadora de Bolsillo Sharp y admire la belleza de su nueva amiga, usted puede que se pregunte qué es lo que está mirando. Examinemos la PC-1500:



Nosotros describiremos la representación visual en un momento. Pero primero, antes que usted encienda la PC-1500, observe varias atracciones especialmente importantes del teclado:

A. TECLAS ON y OFF (Encendido y Apagado)

Obviamente estas teclas encienden y apagan la computadora; para conservar su pila, automáticamente se apagará si usted no escribe nada durante un período de siete minutos, a menos que un programa se esté ejecutando. Observe escrita encima de la tecla [ON] la palabra BREAK. La tecla [ON] puede ser utilizada para interrumpir la ejecución de un programa. Esta función está descrita detalladamente más adelante en este manual.

B. TECLAS ALFABETICAS

Las teclas alfabéticas permiten al usuario de la computadora (usted) dar instrucciones y registrar la información. Además estas teclas pueden ser utilizadas para designar "áreas de almacenamiento" dentro de la memoria de la computadora desde las cuales usted podrá guardar o retirar datos. Este uso se cubrirá en la sección sobre variables. Letras minúsculas: Se obtienen mediante el uso de las teclas [SHIFT] y [SML] (SMALL), descritas más abajo.

C. TECLAS NUMERICAS y TECLAS DE OPERACIONES ARITMETICAS

Con estas teclas usted registra números para calcular. Las teclas [+], [−], [*], y [/] ordenan a la PC-1500 que sume, reste, multiplique y divida respectivamente. La tecla [E] permite el registro de números en "notación cientifica". El uso de esta notación y otras funciones más sofisticadas son descritas en el capítulo de cálculos avanzados.

D. SHIFT (Cambio)

Esta tecla libera las funciones secundarias escritas encima de muchas teclas no alfabéticas. Por ejemplo, para escribir dos puntos, pulse [SHIFT] y luego la tecla con el asterisco ([*]). Cuando la tecla SHIFT va seguida por una tecla alfabética, la letra minúscula aparece en la representación visual. (NOTA: En el modo SMALL, la tecla producirá una letra mayúscula.)

Cuando la tecla [SHIFT] está activada, la palabra SHIFT aparece en la esquina derecha de la representación visual. El modo SHIFT se activa solamente pulsando las teclas una por una.

Las seis teclas en la parte superior del teclado, directamente debajo de la representación visual, se llaman **TECLAS RESERVABLES**. Utilizando el [SHIFT] de una manera que describiremos después, usted puede asignar comandos escritos frecuentemente u otras operaciones a estas teclas. NOTA: Si usted pulsa la tecla [SHIFT] por equivocación, púlsela otra vez para cancelarla.

E. LETRAS MINUSCULAS Y TECLA SMALL

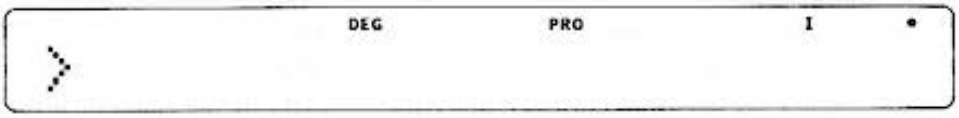
La teclas [SML] le permite especificar minúsculas para todas las teclas alfabéticas. Si usted no especifica el modo SMALL, la PC-1500 seleccionará letras mayúsculas por usted, cada vez que usted pulse una tecla alfabética. (Nosotros denominamos este proceso el modo ejecución por defecto significando que si usted no indica lo contrario prevalecerá la forma preprogramada de operación de la máquina). Usted puede escribir letras minúsculas individualmente pulsando la tecla [SHIFT] antes que la letra.

La tecla [SML] puede ser utilizada para afectar el modo SMALL. En el modo SMALL las letras minúsculas resultan al pulsar una tecla alfabética y las letra mayúsculas individuales son presentadas al pulsar primero la tecla [SHIFT]. Cuando la computadora está en el modo SMALL, la palabra SMALL aparecerá en la parte superior de la representación visual. Una vez que usted haya escogido el modo SMALL, la computadora permanecerá en este modo hasta que usted pulse la tecla [SML] otra vez.

NOTA: Le recomendamos que limite el uso de letras minúsculas por el momento. Esto es debido a que la PC-1500 sólo reconoce instrucciones en letras mayúsculas. Cuando usted aprenda a programar, usted encontrará muy práctico el uso de las letras minúsculas.

F. LA REPRESENTACION VISUAL (VISOR)

Pulse [ON]. La parte de la computadora que tiene una pantalla de cristal se llama la representación visual. Se parece a esto:



En la representación visual usted debe ver un > (que se llama una "prompt"), varias palabras o abreviaturas, y un punto (indicando que las pilas o baterías están funcionando.) No se preocupe si las abreviaturas específicas que aparecen en su representación visual no son las de nuestra ilustración. Estos símbolos cambian cuando usted opera la computadora.

G. EL CURSOR y la Prompt

A la izquierda del visor busque el símbolo prompt >; este símbolo le sugiere que se comunique con la PC-1500. Cuando la prompt aparece significa que computadora no tiene planes inmediatos y espera su mandato. **Escriba una letra de su preferencia.** Esto reemplaza la prompt a la izquierda

de su representación visual, mientras que a la derecha de su letra aparece un _ (un símbolo subrayado). Este es el cursor. Cuando usted pulsa cada tecla, el cursor avanza uno a uno a través de la representación visual, indicando dónde el próximo símbolo que usted escriba aparecerá. **Escriba su nombre y observe el movimiento del cursor.**

Si usted escribe más de 25 caracteres, el límite que puede ser representado al mismo tiempo, el renglón entero se desplazará a la izquierda. (¡Pruébelo!). Los caracteres empujados fuera de la representación visual no se pierden: quedan en la máquina como parte del renglón escrito, hasta un máximo de 80 caracteres por cada renglón. Nosotros veremos cómo hacer volver y cómo cambiar estos caracteres más adelante.

H. CLEAR

(La tecla roja [CL] en la parte superior derecha)

Pulse este botón y usted limpiará (CLear) la representación visual de su contenido. **Utilícelo para borrar los caracteres que usted acaba de escribir.** Observe que la prompt ha vuelto, indicando que la computadora está otra vez esperando sus mandatos.

La tecla [CL] se utiliza también para cancelar un mandato incorrecto. (Vea la sección de mensajes de error más abajo).

I. ENTER

Al escribir en la computadora, las letras o números aparecerán en la representación visual. La PC-1500 **NO TOMARA NINGUNA ACCION**, hasta que señale que usted ha terminado de escribaribir (después de todo la computadora no puede leer su mente). Este se hace pulsando la tecla ENTER. En este instante, la computadora analizará los caracteres que usted ha escrito buscando algún error. Ciertos errores, pero no todos los errores, causarán que su entrada sea rechazada.

RECUERDE: Pulse la tecla [ENTER] cada vez que desee dar una instrucción o dato a la máquina.

J. MENSAJES DE ERROR

Pulse las teclas siguientes:

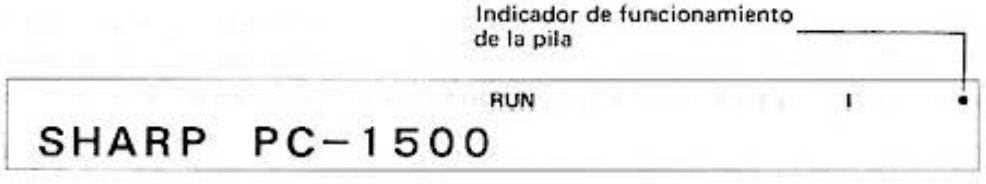
[1] [+] [1] [=]

Ahora pulse [ENTER].

La respuesta debe ser mostrada. ¿Tres, verada? ¿No? ¿Usted obtuvo "ERROR 1" como respuesta? ¿Será que su computadora está defectuosa? ¡Nunca! Hubo una falla en la forma de su mandato. "ERROR 1" es un error en código que le dice que usted ha ejecutado una operación incorrectamente. (Para las personas curiosas, una lista completa de mensajes de error está incluída en un apéndice.) Nosotros seremos los culpables por este primer error y en secciones posteriores le enseñaremos como utilizar otras teclas para corregir un mandato erróneo. Por ahora, usted puede utilizar la tecla CL para borrar el mensaje de error.

K. INDICADOR DEL FUNCIONAMIENTO DE LAS PILAS

Cuando este punto desaparece, es hora de rejuvenecer la PC-1500 reemplazando sus pilas. Vea las instrucciones que aparecen en el apéndice.



L. TECLA RCL (DE LLAMADA)

Esta tecla se usa para "llamar" o recuperar información o sentencias previamente almacenadas en el modo RESERVE. Para efectuar la extracción de un registro (entrada) del modo RESERVE, llevar a cabo los siguientes pasos: Registrar primero el modo RESERVE apretando la tecla de selección reservable (▲▼) para seleccionar el grupo I, II o III (la tecla de selección reservable (RESERVE SELECT) está situada inmediatamente a la izquierda de la tecla RCL de llamada). Apretar después la tecla RCL (de llamada) y la tecla reservable previamente asignada. Si ha olvidado sus designaciones previas, sírvase seguir las instrucciones del párrafo B en la página 141.

EXAMEN – Capítulo 1

Encuentre el significado de cada ítem de la columna A en la lista de ítems de la columna B. (Las respuestas están al final de la página).

NOTA: La columna B contiene algunas posibilidades graciosas.

A	B
a) Tecla SHIFT	1) Un mensajero de la edad del primer Triunvirato.
b) La Representación Visual	2) Una tecla que limpia la representación visual y borra los resultado de computaciones anteriores.
c) El Cursor	3) Una tecla que selecciona uno de los dos caracteres que comparten la misma tecla.
d) La Tecla ENTER	4) Esa tecla no existe.
e) La Tecla BREAK	5) Un carácter que aparece en la representación visual, el cual informa al usuario que la computadora espera su órden.
f) La Tecla CLEAR	6) La tecla que señala a la computadora que el usuario ha acabado de escribir.
g) La Prompt	7) Una tecla que interrumpe un programa en el proceso de computación.
h) La Tecla de do bemol	8) El nombre de un pez tropical exótico.
	9) El lugar en el visor, que indica donde el próximo carácter escrito aparecerá.
	10) La pantalla en la cual la información aparece.

Respuestas: A–3, B–10, C–9, D–6, E–7, F–2, G–5, H–4

II. ZAMBULLENDOSE . . .

En realidad, le hemos dado un nombre equivocado a este capítulo, ya que aprender a utilizar la PC-1500 no es tan aprensivo como zambullirse en una piscina de agua. Sin embargo, esto requiere que usted enfoque la computadora sin miedo. Recuerde, **usted no puede dañar la computadora simplemente pulsando las teclas.**

En este capítulo examinaremos las características principales de la PC-1500 sobre las cuales se basan los programas y los cálculos más avanzados. Tome el tiempo necesario para trabajar utilizando los ejercicios ofrecidos en cada sección. Una buena comprensión de las formas básicas le permitirá utilizar esta máquina en su potencial máximo.

Aunque no lo recomendamos, si usted se siente suficientemente avanzado, puede proseguir directamente al resumen que se encuentra al final de este capítulo.

A. MODE

Empezaremos con una tecla que hemos estado ignorando hasta ahora. A la derecha del teclado encuentre la tecla rotulada MODE. Esta tecla es muy importante. Púlsela repetidamente, y observe, cada vez, los cambios en las abreviaturas a la derecha. Cambiando el modo de la máquina de esta forma, se asemeja al cambio de marchas de un coche. En cada cambio de marcha, la maquinaria que exteriormente parece sin ningún cambio, actúa diferentemente. Como un coche, la PC-1500 debe ser colocada en el modo apropiado para funcionar de acuerdo con el plan. Y, también como un coche, cuando usted intente operar en una marcha diferente, la computadora le notificará rápidamente su equivocación.

Al pulsar repetidamente la tecla MODE usted será introducido en dos de los tres modos más importantes de la SHARP: RUNy PROgrama. El tercer modo, RESERVE, se activa pulsando la tecla SHIFT y después la tecla MODE. Los capítulos posteriores de este manual describen como cada uno de estos modos contribuyen a una operación precisa de la computadora. Por ahora, recuerde que para utilizar la como simple calculadora, usted tiene que estar en el modo RUN. Posteriormente, en el modo PROgram, usted escribirá y modificará programas. Con el modo RESERVE usted puede asignar a teclas únicas, funciones utilizadas frecuentemente. Este proceso se explica con más detalles en el capítulo 7.

Observe que la primera vez que usted encienda la computadora (después de ponerle las pilas) se pondrá en el modo PRO. Otras veces la computadora se encenderá mostrando el último modo en que fue utilizada antes de ser apagada.

B. CALCULOS SIMPLES

Vamos a probar computaciones matemáticas fundamentales colocando la SHARP en el modo RUN. **Es necesario que pulse [CL] antes de cada cálculo.** Esto limpia la representación visual de cualquier información previa que pueda interferir con un nuevo cálculo. Practique los siguientes problemas:

Escriba	Representación visual
5 + 2 ENTER	7
5 − 2 ENTER	3
5 / 2 ENTER	2.5
5 * 2 ENTER	10

NOTA: No escriba el signo igual. Acuérdese (Capítulo 1) que es la tecla ENTER la que informa a la SHARP PC-1500 que usted ha terminado de escribir y desea tener su mandato o cálculo ejecutado.

C. CALCULOS EN SERIE

Used puede utilizar la respuesta de un cálculo en el cálculo siguiente si prosigue directamente al segundo cálculo (no pulse CL entre cálculos). Así que si usted estuviera haciendo el balance de su cuenta bancaria, usted operará la PC-1500 de la manera siguiente:

Escriba	Representación visual
161.16 − 47.50 ENTER	113.66
− 12.33	113.66−12.33_
ENTER	101.33

Como usted observará, el resultado del primer cálculo salta a la izquierda del visor cuando usted comienza el segundo cálculo.

NOTA: NO escriba signos monetarios o comas cuando registre números en los cálculos. Estos símbolos tienen significados especiales en el lenguaje BASIC (y consecuentemente para la SHARP PC-1500).

Otras operaciones pueden ser efectuadas de la misma forma. Pruebe estas:

Escriba	Representación visual
5 + 3 ENTER	8
8 + 3 − 1 ENTER	10
10 * 3 − 1 ENTER	29
29/3 − 1 ENTER	8.666666667

D. CALCULOS CON NUMEROS NEGATIVOS

Imagine que usted tiene siete manzanas y le da dos a su profesor de computación, Sr. Galán. Le quedan cinco manzanas y se pregunta "¿Cuántas manzanas tendría ahora si no hubiera sido tan generoso con el Sr. Galán?" Para encontrar la respuesta, escriba en la PC-1500: 5 — 2 [ENTER] Su inventario hipotético sería siete. Pruebe estos cálculos similares con números registrados negativamente:

5 * − 2

5 + − 2

5 / − 2

− 5 − 2 . 3

− 5 + − 2

− 5 / − 2

RECUERDE: Pulse [CL] entre cálculos para limpiar resultados previos.

E. CALCULOS COMBINADOS

Usted puede unir una secuencia de cálculos antes de preguntar a la PC-1500 para obtener una respuesta. Por ejemplo, usted y dos amigos (Pedro y Pablo) desean compartir cinco manzanas dos veces al día por una semana. ¿Cuántas manzanas necesita comprar para que le duren toda la semana? 5 manzanas [/] 3 amigos [*] 2 cada día [*] 7 días:

Pulsaciones	Representación visual
[5] [/] [3] [*] [2] [*] [7] [ENTER]	23. 33333333

(Compre 24 manzanas y un loro al cual le puede dar la 1/3 parte de la manzana). Ejecute los siguientes cálculos, (pero esta vez invente sus propias historias respecto a lo que ellas representan):

Escriba	Representación visual
5 * 2 − 3.675 [ENTER]	6. 325
5 / 3 * 6.2 + 7 − 47 [ENTER]	−29. 66666667

F. USO DE PARENTESIS

Un problema que emerge al mismo tiempo que investigamos cálculos combinados, es el de la prioridad. Por ejemplo, la expresión 5-3/4 puede ser leída de dos formas: (5 menos 3) dividio por 4, en cuyo caso la respuesta es 0.5, ó 5 menos (3 dividido por 4), en cuyo caso la respuesta es 4 25.

Los paréntesis están situados en la primera línea de teclas y usted los puede utilizar para aclarar estas ambiguedades. Ejecute los cálculos siguientes:

Escriba	Representación visual
5 – 3 / 4 ENTER	4. 25
5 – (3 / 4) ENTER	4. 25
(5 – 3) / 4 ENTER	0. 5

La SHARP PC-1500 está predispuesta (tiene prioridades fijas de antemano) a realizar algunos cálculos antes que otros (Encontrará la lista completa del orden en que su computadora calcula, en el apéndica O.) Divisiones y multiplicaciones serán llevadas a cabo antes que la sustracción y la adición, a menos que los paréntesis sean utilizados para dirigir mandatos diferentes. De esta forma, la PC-1500 está construida para interpretar el primer cálculo (en el ejercicio anterior) como si fuera igual al segundo. Para asegurarse que la respuesta que le da es la que usted necesita, utilice los paréntesis para indicar el orden exacto en que debe ejecutar los cálculos.

La SHARP PC-1500 puede interpretar varios niveles de paréntesis como en este problema:

Escriba	Representación visual
((6 – 4) / 2) * (((3 – 1) / 4) * 6) ENTER	3

Si existen varios niveles de paréntesis en una ecuación, la computadora resuelve primero el nivel más interno.

RECUERDE: Cuando tenga una duda, utilica paréntesis para aclarar el orden de sus operaciones aritméticas.

G. COMPARACIONES LOGICAS y DESIGUALDADES

La computadora SHARP le permite "comparar" dos valores o ecuaciones y le indica el resultado de la comparación. Esta habilidad es esencial para diseñar programas que tomen decisiones. La manera en que esto se hace será reconocible a estudiantes de la "nueva matemática" como desigualdad (no se desespere si usted no estudió la nueva matemática; el autor tampoco lo hizo)

Una desigualdad puede considerarse como una comparación que no es ni verdadera ni falsa. Por ejemplo, la afirmación "seis dividido por tres es igual a dos es una comparación que es verdadera. Por otro lado, la afirmación "seis dividido por tres es mayor que cinco" es una comparación falsa.

Las computadoras y los matemáticos utilizan los mímbolos siguientes para expresar los posibles tipos de comparaciones:

<	menor que
>	mayor que
=	igual a
<=	menor que o igual a
>=	mayor que o igual a
<>	no es igual a

Asi, podemos repetir simbólicamente las igualdades arriba mencionadas, de esta manera: 6/3 = 2 y 6/3 > 5 respectivamente.

Dada una desigualdad, la computadora determinará si la comparación es verdadera o falsa. Siguiendo las prácticas modernas en el diseño de computadoras, la PC-1500 indicará una afirmación verdedera con un 1 y una afirmación falsa con un Ø. Por ejemplo, si usted escribe:

6 / 3 = 2 [ENTER]

Responderá con un 1 (cierto). Escribiendo la secuencia:

6 / 3 > 5 [ENTER]

recibirá una respuesta de Ø (falso).

Los siguientes ejercicios le demostrarán el razonamiento de la PC-1500 (asegúrese que usted está en el modo RUN):

Pulsaciones	Resultado
[4] [SHIFT] [>] [5] [ENTER]	Ø
[5] [SHIFT] [>] [4] [ENTER]	1
[2] [*] [2] [SHIFT] [<] [2] [*] [3] [ENTER]	1
[1] [8] [SHIFT] [<] [SHIFT] [>] [1] [8]	Ø
[2] [=] [2] [ENTER]	1
[5] [SHIFT] [<] [=] [5] [ENTER]	1
[2] [5] [*] [2] [SHIFT] [>] [=] [1] [Ø] [Ø] [/] [3] [+] [4] [ENTER]	1

Como usted habrá observado, las ecuaciones comparadas pueden ser tan complejas como sea necesario (el único límite es un máximo de 80 caracteres por renglón).

Aquí hay un problema sencillo para ilustrar un uso práctico de desigualdades:

El Sr. Picaporte entra en la Ferretería "Clavos y Cosas." El encuentra que el cemento está empaquetado en bolsas de 4, 8, y 12 Kilos. El Sr. Picaporte sólo tiene dinero para comprar dos bolsas de 12 Kilos ò 3 de 4 Kilos y una de 8 Kilos. El se pregunta en cual de las formas obtendrá más cemento, y le pregunta a su computadora:

```
2*12>(3*4)+8
```

La SHARP responde 1; por supuesto, el Sr. Picaporte compra dos bolsas de 12 Kilos.

Experimente con desigualdades aplicándolas a sus propios problemas.

H. TECLAS Y FUNCIONES DE EDICION

La mayor parte de los'humanos (excepto nosotros que somos unos genios) tienen la tendencia a commeter errores. Reconociendo la falibilidad humana, los diseñadores de la PC-1500 han incorporado varias características especiales que facilitan cambios y correcciones.

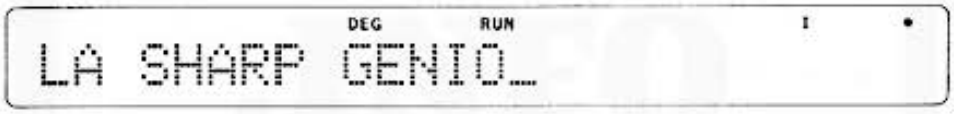
H-1. Flecha Izquierda/Tecla DELete (Supresora) [DEL ◄]

Algunos de ustedes ya deben haber descubierto la tecla con la Flecha Izquierda en el lado derecho del teclado. Esta tecla actúa como la tecla de retroceso en las máquinas de escribir modernas; esta tecla le permite retroceder hacia caracteres que ya han sido escritos.

Utilizando el modo RUN, luego de visualizar el símbolo, es decir la prompt, escriba los siguientes caracteres:

[L] [A] [SPACE] [S] [H] [A] [R] [P] [SPACE] [G] [E] [N] [I] [O]

La representación visual debe aparecer así:



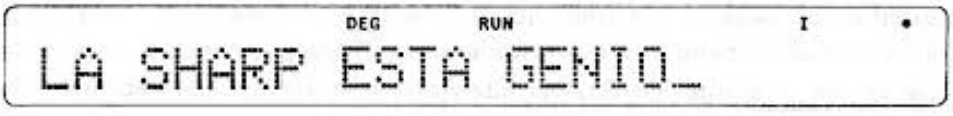
Pulse la tecla con la Flecha Izquierda una vez y observe la manera en que el cursor cambia. En esta forma el cursor parpadeante le permite ver el carácter que se encuentra en la posición actual. Pulse la tecla con la Flecha Izquierda repetidamente (o manténgala oprimida) hasta que el cursor esté colocado sobre la G (si accidentalmente pasa le letra G, mueva el cursor hacia adelante otra vez utilizando la tecla con la Flecha Derecha).

Escriba una E. La E reemplazará a la G y el cursor se moverá hacia adelante. Esto no le sorprenderá si usted recuerda que el cursor indica dónde el próximo carácter será colocado.

Pulse estos caracteres:

[E] [S] [T] [A] [SPACE] [G] [E] [N] [I] [O]

La representación visual ahora muestra:



Como la representación visual ilustra, **los caracteres sobre los que usted ha escrito han sido modificados con la nueva digitación.**

Además de la forma de retroceso, la tecla con la Flecha Izquierda tiene un segundo uso, la función supresora cuya abreviatura está inscrita sobre la tecla. Para suprimir un carácter, coloque el cursor en el carácter que desea eliminar y pulse [SHIFT] [DEL]

Vamos a probarlo; mueva el cursor hacia atrás hasta la U y pulse las teclas [SHIFT] [DEL] dos veces consecutivas. La representación visual ahora muestra:

ya que eliminamos la palabra un y el espacio siguiente.

H-2. Flecha Derecha/Tecla INSertar [INS ▶]

Como ya hemos visto, la tecla con la Flecha Derecha mueve el cursor hacia adelante sin borrar caracteres. Como la Flecha Izquierda, el cursor avanzará repetidamente si la tecla se mantiene oprimida.

Mueva el cursor al final del renglón.

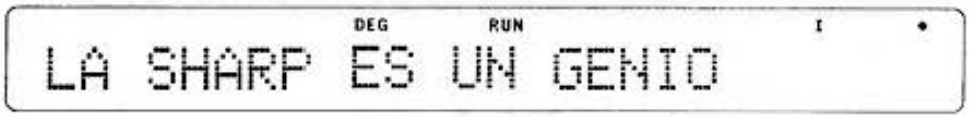
La segunda función de la tecla con la Flecha Derecha nos da la habilidad de insertar caracteres dentro de una línea escrita. Esta función es muy práctica para aquellas personas (como el autor) que tienden a olvidar cosas pequeñas (como letras y palabras).

Mueva el cursor hacia atrás a la letra G en genio. Pulse la secuencia [SHIFT] [INS] tres veces y observe la palabra GENIO moverse tres espacios a la derecha. Los nuevos símbolos como cajas que han aparecido pueden ser razonadas como "retenedores" de espacio. Estos retenedores de espacio pueden ser llenados con nueva información.

Escriba:

[U] [N] [SPACE]

La representación visual ahora muestra:



¡Abracadabra! Usted ha insertado una palabra. ¡Ojalá que mi máquina de escribir pudiera hacer esto!.

H-3. FUNCION RECUPERADORA

Hasta ahora hemos estado discutiendo formas para corregir sentencias que no han sido sometidan todavla a la computadora (es decir, usted no ha pulsado [ENTER] después de sus anotaciones) Una vez que usted pulse [ENTER] , sin embargo, la SHARP inmediatamente intentará ejecutar sus cálculos. Si la computadora tiene éxito, el resultado reemplazará su ecuación en la representación visual. La ecuación, sin embargo, no está perdida y puede ser recuperada (representada de nuevo) al pulsar la Flecha Izquierda o bien la Flecha Derecha.

Limpie la representación visual y escriba una ecuación que usted desee. Pulse la tecla [ENTER] para obtener la respuesta. Recupere la ecuación pulsando la Flecha Izquierda o la Derecha. Observe que para ver el resultado de nuevo es necesario recomputar la ecuación (utilizando la tecla [ENTER])

Afortunadamente, la función recuperadora también funcionará si la PC-1500 encuentra un error mientras está tratando de evaluar (razonar) la información que usted le ha sometido. Esto le permitirá a usted recuperar y corregir ecuaciones erróneas utilizando cualquiera de las formas de edición que usted ya ha aprendido.

Para examinar esto, pulse la siguiente sentencia que contiene un error:

```
45 * 63 / * 2  [ENTER]
           ↑
```

Cuando el mensaje de error aparece (ERROR 1) pulse una de las teclas editoras (◄ ó ►) para recuéror la expresión. Usted verá el cursor parpadear sobre el símbolo de la segunda multiplicación (asterisco). Esta es la forma que utiliza su computadora para indicarle en qué punto se confundió (y por una buena razón en este caso). De aqui en adelante, usted puede efectuar cualquier correción que juzgue apropiada.

I. VARIABLES

La habilidad de trabajar abstractamente a través de variables es una de las funciones más poderosas de la PC-1500. Las variables pueden ser consideradas como un grupo de cajas pequeñas, cada una de las cuales puede ser llenada con un grupo único de datos como un número o un nombre.

Usted tal vez recuerda las variables del álgebra de su escuela superior. Usted aprendió (o por lo menos le enseñaron) que si 5A = 30, entonces A bede ser 6, pero si 5A = 35, entonces A es igual a 7. La A en este caso es una variable que representa un solo número (no siempre el mismo y por lo tanto un número variable) llamada el valor de la variable. La habilidad de utilizar una letra (como la A) en una ecuación en vez de un número específico es muy útil. Vamos a ver porqué en el siguiente ejemplo:

El General Duffer dosea comprar un conjunto de palos cromados de golf de lujo. El conjunto consiste en 5 palos a $12 cada uno, una bolsa a $21,99 y tres pelotas a $1,56 cada una. El "Almacén Militar" le ofrece este conjunto con un 10% de descuento pero tiene un gasto adicional de $8 por lustrar y entregar. La tienda de deportes de enfrente "Bernie-Sport" le ofrece el mismo conjunto con un 5% de descuento y entrega gratis.

Para computar el mejor negocio, el General decide solicitar la ayuda de su SHARP PC-1500. El General calcula el costo básico del conjunto y guarda el resultado dentro de la variable G (G de Golf, no de General):

```
G = (5 * 12) + 21.99 + (3 * 1.56)  [ENTER]
```

El resultado es mostrado en el visor puede ser recuperado de la memoria al pulsar el nombre de la variable: G [ENTER]

Ahora el General calcula la compra al precio del "Almacén Militar":

G – (G * .10) + 8 ENTER

Y el precio de compra en la tienda "Bernie-Sport":

G – (G * .05) ENTER

Claramente, la tienda de Bernie ofrece el mejor precio. (Los lectores más inteligentes pueden observar que el General podria haber resuelto este problema utilizando la función recuperadora (vea Capítulo 2) aunque no tan fácilmente).

Hay varias lecciones que debe aprender del ejemplo anterior . . .
Observe que los primeros cálculos del General tienen la forma siguiente:

variable-nombre = expresión

Una instrucción con esta forma se llama **"Asignación"**. Es importante no confundir una Asignación con una desigualdad. Al contrario de las Asignaciones las desigualdades no son utilizadas separadamente, pero forman parte de otras instrucciones programadas.

La Asignación le indica a la PC-1500 que almacene el resultado obtenido al calcular la expresión, dentro de la localización de la memoria asociada con el nombre variable dado. De aquí en adelante utilizar el nombre de la variable (G en nuestro ejemplo) es como utilizar el resultado mismo. Observe también que una variable puede ser utilizada tantas veces como sea necesaria en el mismo cálculo.

Las variables pueden ser utilizadas para otros trucos bonitos también. El valor de una variable puede ser asignado a otra, como en la sentencia:

H = G

que copiará nuestro resultado previo en H. G no ha sido alterada; el mismo resultado ahora está almacenado en dos variables diferentes.

Las variables que contienen números pueden ser incrementadas o disminuídas en una sentencia como en este ejemplo:

G = G + 5

Esta instrucción causará que la SHARP recuerde el valor de G, añada 5 a ese valor, y almacene el valor nuevo otra vez en G. Esta capacidad es útil para toda clase de cálculos:

El costo de una mecha de lámpara es almacenado en la variable X. Asumiendo un impuesto de 6,5%, ¿cuál es el precio de compra de la mecha?:

X = X * 1.065

Por supuesto, podríamos haber escrito este ejemplo como P = X * 1.065 para no alterar X. Si hubieramos almacenado el tanto por ciento del impuesto en una variable I, podriamos haber dicho X = X * I ó P = X * I.

Hasta ahora, hemos estado utilizando letras solas como nombres de variables. Esto nos provee con 26 variables (A a Z). Pero la, SHARP nos permite el uso de más de 950 nombres variables para números solos. Un grupo adicional de más de 950 variables pueden ser utilizadas para almacenar hasta 16 caracteres cada una. Finalmente (como si no fuese batante), utilizando técnicas más avanzadas, se pueden crear variables que retienen tantos números o caracteres como se desen. El único límite es la cantidad de memoria disponible en la computadora.

El espuema-nomenclatura para variables es sencillo y fácil de aprender. Los nombres de variables numéricas (aquellas que son utilizadas para retener números) pueden ser escogidas utilizando las reglas siguientes:

- El nombre puede ser una letra; de la A a la Z.
- El nombre puede ser una letra seguida por un solo número (del 0 al 9) o por otra letra.

Así, los siguientes son nombres válidos de variables numéricas:

S, Q1, TX, MM, Z9, R0, E.

NOTA: Debido a conflictos con abreviaturas que tienen otro significado en el lenguaje BASIC, la PC-1500 no permite el uso de los siguientes nombres variables: LF, IF, LN, PI, TO.

Los nombres de variables de caracteres (aquellos utilizados para retener las siguen las mismas reglas de arriba excepto que el nombre acaba con el signo de dinero ($). Un signo de dinero alerta a la computadora de que la variable contiene información sobre caracteres. Los siguientes son ejemplos de nombres válidos para variables alfanuméricas:

T$, P2$, T7$, AA$, YR$, X$, ZH$, B5$.

NOTA: Debido a conflictos con palabras que son parte del lenguaje BASIC, no se permite el uso de estos nombres para variables: LF$, IF$, LN$, PI$, TO$, IN$.

Es importante comprender que una variable A y una variable A$ son dos variables diferentes. La primera puede retener solamente un número y la segunda puede retener sólo caracteres. Como varemos después el BASIC de SHARP incluye instrucciones que convierten caracteres a números y números a caracteres.

Para almacenar caracteres en variables alfanuméricas nosotros utilizamos una variación de nuestra amiga la Asignación:

nombre de variable alfanumérica = "caracteres"

Como un ejemplo escriba lo siguiente:

D$ = "PEDRO LOPEZ"

Ahora recupere el contenido de D$ al escribir [D] [SHIFT] [$] [ENTER]

PEDRO LOPEZ

Observe que el espacio entre O y L fue almacenado y que las "(comillas) no lo fueron. Las camillas sirven como delimitadores, como una guía para indicar los otros caracteres que se almacenaran. Cualquier carácter (incluyendo espacios) excepto comillas, puede ser almacenado. Esta secuencia de caracteres circundada por comillas se llama una "ristra" o "serie" de caracteres y las variables alfanuméricas son a veces referidas como "series variables". Cada variable alfanumérica puede retener hasta 16 caracteres. Alguna precaución debe ser utilizada para no exceder este límite o la información puede perderse. Esto se demuestra en el siguiente ejercicio que, por cierto, cambia las costumbres de comer de Manuel. Escriba:

F$ = "MANUEL COME MAYONESA"

Ahora recupere la información: [F] [SHIFT] [$] [ENTER]. ¡Qué cambio!

Una nota final acerca de las variables: estas tienen las memoria de un elefante. La información permanece almacenada en una variable hasta que:

1) Otra Asignación sea ejecutada para la misma variable.
2) Una orden de NEW o CLEAR sea dada.
3) Sea ejecutado un programa que modifique dicha variable.
4) Las pilas de la computadora sean cambiadas.

¡El valor de una variable es incluso retenido cuando la máquina está apagada! Compruebe esto apagándo la y después encendiéndola. En el modo RUN, pregunte por el valor de G (escriba: [G] [ENTER]). ¿Impresionante, no? Cuando usted empiece a programar en el capitulo próximo, usted encontrará lo indispensables que son las variables.

J. HACEMOS UNA PAUSA . . .

¡Felicitaciones! Si usted ha perseverado hasta este punto, usted ahora conoce las funciones básicas de la SHARP PC-1500 bastante bien para ejecutar una variedad amplia de cálculos. Como la PC-1500 es tan versátil, cada lector encontrará muchos usos para ella en sus campos de intereses y estudios. No importa cual sea su aplicación, eventualmente usted querrá aprender a programar a fin de aprovechar totalmente el poder de esta máquina tan extraordinaria.

Llegado este punto, usted tiene una elección. Para aquéllos cuyos corazones están palpitando rápidamente, cuyas respiraciones son más rápidas con el pensamiento de programar (y para aquellos que no se hayan dormido todavia), nosotros queremos pedirles que retrocedan al capítulo 0, completando así su nuevo conocimiento con la información que encontrará allı. Después proceda la Capítulo 3.

Aquellos lectores que tienen una necesidad inmediata de utilizar funciones especiales como Notación Científica y Funciones Trigonométricas, pueden dirigirse directamente al Capítulo de Cálculos Avanzados.

RESUMEN

1) **Modos** – La SHARP PC-1500 opera en uno de los tres modos siguientes: RUN, PROgrama, y RESERVE. Cada cambio en modo produce un pequeño cambio en las funciones internas, análogas al cambio de marchas en un coche. El modo RUN es para cálculos y es el modo en que usted debe ejecutar (RUN) programas. En el modo PROgrama, toda la escritura y revisión (adiciones y correcciones) de programa es ejecutada.

2) **Cálculos** – La SHARP ejecuta cálculos aritmético comunes en el modo RUN. La tecla CL (CLEAR) debe ser pulsada antes de cada cálculo para borrar los resultados de un cálculo previo. Sin pulsar la tecla CL entre cálculos, es posible una serie encadenada de operaciones utilizando los resultados previos.

3) **Caracteres Especiales** – El $ (signo de dinero) y la "," (coma) tienen significados especiales en BASIC y no pueden utilizarse como parte de un número dentro de un cálculo.

4) **Números Negativos** – Se indican al preceder el número con un – (signo de menos). Por ejemplo: –5 + 2 = > –3

5) **Cálculos Combinados** – Siguen leyes algebráicas comunes. Los paréntesis son utilizados para indicar el) significado correcto de una expresión dada. Una información completa sobre el orden de evaluación de expresiones se da en el apéndice 0.

6) **Desigualdades** – Las desigualdades usando los símbolos, <,>, <=, >=, y < > (que significan:

7) **La Tecla con Flecha Izquierda** – [◄] actúa como un cursor no destructivo en retroceso. Reteniendo esta tecla causará repetición automática. El cursor subrayador cambia a centelleante cuando se coloca sobre un carácter previamente escrito. Pulsando SHIFT seguido por la tecla con la Flecha Izquierda se invoca la función supresora (DEL), que borra el carácter en que el cursor está colocado.

8) **La Tecla con Flecha Derecha** – [►] mueve el cursor hacia adelante sin destruir. Reteniendo esta tecla causará repetición automática. La secuencia SHIFT, Flecha Derecha invocará la función INSertar; que inserta un carácter "retenedor" de espacio en la posición del cursor. Este carácter puede entonces sobreinscribirse con la información que va a ser insertada.

9) **Función Recuperadora** – Después de pulsar ENTER y aparecer en el visor el resultado de un cálculo, la ecuación original puede ser evocada al pulsar la tecla con Flecha Izquierda o la Flecha Derecha. En este punto, se pueden hacer modificaciones y la ecuación modificada re-introducirse. La función recuperadora también será efectiva en cualquier expresión no-programada que produce un error. En este caso, el cursor será colocado en el punto de descubrimiento del error.

10) **El Uso de Variables** – En el modo RUN aumenta substandialmente el poder computacional de

11) **Asignaciones utilizando** las formas:
nombre-variable = expresión
nombre – variable alfanumérica = "caracteres"
permiten el almacenamiento de un solo número o una serie de hasta 16 caracteres, respectivamente. Cualquier carácter imprimible puede ser utilizado dentro de la serie excepto las comillas.

12) **Nombres de Variables** para variables numéricas son:
1. Una letra (de la A a la Z)
2. Una letra seguida de un número (0 a 9) u otra letra.

Los nombres de variables alfanuméricas siguen las mismas reglas con la adición de la $ (signo de dinero) como sufijo a todos los nombres.

13) **Excepciones** – Las siguientes son excepciones al esquema de nombres y no pueden ser utilizadas para llamar variables: **IF, LF, LN, PI, TO, IF$, LF$, LN$, PI$, TO$.**

14) **Duración de la Información** – La información dentro de variables es retenida hasta que:
1) Una orden de CLEAR o NEW es dada.
2) Se ejecuta un programa que modifique dicha variable.
3) Otra Asignación es ejecutada utilizando la misma variable.
4) Las pilas de la computadora son cambiadas.

Apagando la computadora no se afecton los valores almacenados en las variables.

III. EL ARTE MISTERIOSO (?) DE LA PROGRAMACION

El arte de programar ha sido innecesariamente cubierto con un velo de misterio por tanto tiempo que la mayoría de la gente lo asocia con mágia o genio matemático. El hecho es que no se requiere talento especial para sacar conejos de los sombreros. Ni es necesario que usted sea adepto a resolver ecuaciones diferenciales parciales. Sus mayores créditos serán su paciencia, sus habilidades para razonar lógicamente, su atención al detalle, y su deseo de aprender. Una buena voluntad para aceptar desafios es también util (no le engañaremos; a veces programar es muy desafiante, por eso es tan divertido).

Programar es un arte, y como tal requiere un poco de habilidad, un poco de instrucción, y mucha práctica. No es nuestra intención en este manual hacer de usted un programador experto. Nosotros le familiarizaremos con las operaciones básicas y conceptos de programación. Pero para ser un programador competente se requiere algo mas, al igual que una buena conducción de automóviles implica mucho más que conocer cómo mover el volante y cambiar las marchas.

Excelentes libros de programación existen ya y nosotros le recomendamos que visite su vendedor local de computadoras o una biblioteca.

A. ¿Qué es un Programa?

Puede que usted esté sorprendido al descubrir que un programa es secillamente un conjunto de instrucciones que la computadora sigue una a una. Estas instrucciones deben ser dadas a la computadora en un lenguaje que ella comprende. La SHARP PC-1500 "habla" un dialecto de BASIC, que es un lenguaje de programación muy popular y extensamente utilizado. Como otros lenguajes, el BASIC tiene un vocabulario especial y reglas de gramática que son combinadas para formar oraciones. Si usted le hable a la SHARP agramaticálmente, o en vocabulario extraño, la computadora le avisará de su error. Pero, no es difícil darle órdenes correctamente. El lenguaje BASIC fue originalmente desarrollado para enseñar principios de programación y muchas de sus oraciones contienen palabras inglesas y otros símbols familiares.

B. ¿Cómo Voy a Programar?

Cuando usted utiliza su computadora PC-1500 para programar, usted seguirá una cierta rutina. Las instrucciones que forman un programa se registran en el modo PROgrama. Estas instrucciones se conocen como "sentencias" en el lenguaje BASIC. Para empezar la ejecución de estas sentencias, es necesario conmutar al modo RUN, y luego instruir a la PC-1500 que prosiga escribiendo el comando RUN. Para los expertos, que ya tienen los dos programas del Capítulo 0 bajo control esto será familiar. Para aquellos que ya han hojeado otros capítulos, vamos a probar entrando y ejecutando un programa:

Cambie al modo PROgrama y dé el comando de BASIC (la diferencia entre comandos y sentencias se discute más tarde):

[N] [E] [W] [ENTER]

Esto borrará cualquier sentencia previa que exista en la memoria. Escriba la línea siguiente:

```
Listado del Programa:

10 PRINT "BIEN HECHO!"
```

Digitaciones:

[1] [0] [P] [R] [I] [N] [T] [SHIFT] ["] [B] [I] [E] [N] [SPACE]
[H] [E] [C] [H] [O] [SHIFT] [!] [SHIFT] ["] [ENTER]

Nuestro programa de un renglón está completo. Cambie al modo RUN y escriba:

[R] [U] [N] [ENTER]

Esta orden instruye a la SHARP PC-1500 a que empiece a procesar las sentancias (o, en este caso, la sentencia) en nuestro programa. Siguiendo las órdenes la SHARP imprime:

```
                DEG     RUN                 1     •
BIEN HECHO!
```

en la representación visual. (Pulse [ENTER] para informar a la computadora cuando usted ha acabado de leer).

Para hacer cualquier adición, cambios, o supresiones a nuestro programa, debemos volver al modo PROgrama. Si su programa contiene un error y no se ha concluido con éxito, será necesario volver al modo PROgrama para corregir la sentencia. Así es como es trabaja en programas en el modo PROgrama, mientras que la ejecución y la prueba de programas se ejecutan en el modo RUN.

C. COMANDOS Y SENTENCIAS

Usted puede haber observiado en el ejemplo previo que nosotros comunicamos nuestros deseos a la SHARP a través de dos métodos diferentes. Las instrucciones NEW y RUN fueron ejecutadas inmediatemente después que pulsamos [ENTER] . Este tipo de instrucción se conoce como un "comando". La instrucción PRINT, por otro lado, fue un poco diferente. Fue pulsada en el modo PROgrama, y precedida por un número (10), y no fue ejecutada inmediatamente. Este tipo de instrucción se llama una "sentencia".

Hasta cierto punto, los comandos le dicen a SHARP lo que debe hacer con las sentencias. Por ejemplo, NEW borrará todas las sentencias acumuladas hasta ahora. Es importante recordar que los comandos no pueden ser utilizados en un programa mientras que las sentencias casi (pero no completamente) siempre están agrupadas para formar un programa.

D. NUMERO DE LINEAS

Un programa BASIC consiste en una serie de líneas individualmente numeradas, conteniendo una o más sentencias. Los "números de línea" son utilizados por la SHARP para mantener la secuencia correcta durante la ejecución pero no forman parte de la producción cuando el programa es ejecutado. Las sentencias pueden ser ejecutadas en cualquier orden pero son procesadas por la computadora en la secuencia de los "números de linea" (sujeto a modificación, como veremos más adelante).

Como demostración, vamos a añadir una sentencia a nuestro programa previo. Cambie al modo PROgrama y escriba:

```
5 PRINT "QUE ";
```

Digitacíones:

[5] [P] [R] [I] [N] [T] [SHIFT] ["] [Q] [U] [E] [SPACE]
[SHIFT] ["] [SHIFT] [:] [ENTER]

Ejecute ahora el programa en el modo RUN. ¿Qué pasa? Pulse [ENTER] depués que "QUE" aparezca.

Observe dos cosas en este ejemplo: lo que la SHARP hizo por usted y lo que usted tuvo que hacer por usted mismo. La SHARP colocó y ejecutó las líneas 5 y 10 en el orden correspondiente. Sin embargo, usted tuvo que empujar a la computadora del renglón 5 al 10. Para vel el resultado de la línea 10, usted tiene que pulsar [ENTER] después de cada salida.

Aunque en "número de línea" puede ser cualquier número del 1 al 65.279 le advertimos enérgicamente que numere sus líneas en incrementos de 10 (i.e. 10, 20, 30, ..., etc.). Esto le permite insertar hasta 9 sentencias entre sus sentencias actuales (11 a 19, por ejemplo). Algunos programadores recomiendan una brecha incluso mayor; numerando por 20. ¡Aunque, con un programa bien diseñado y escritura cuidadosa, usted raramente necesitará insertar sentencias! ¡No cuente con esto! Es mucho más fácil numerar a 10 ahora, que numerar de nuevo todas las líneas.

Recuerde que dos líneas no pueden tener el mismo número. Si esta condición ocurre, la primera línea se perderá. Esta característica puede ser utilizada para suprimir líneas no deseadas simplemente pulsando el número de la línea que debe ser suprimida y pulsando [ENTER] . Asi una línea vacía, pero numerada, suprimirá eficazmente una línea que ya existe con el mismo número.

Sin embargo, si utiliza números de línea duplicados sin querer, va a tener problemas. Para demostara esto, ingresr la siguiente línea (por supuesto, en el modo PROgrama):

```
10 PRINT "HORROR"
```

Digitaciones

[1] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [H] [O] [R] [R] [O] [R]
[SHIFT] ["] [ENTER]

Ejecute el programa de la misma forma que lo hizo anteriormente. Es horroroso perder una línea en el programa, ¿no? Y como la pérdida de una línea puede causar algunos errores más sutiles usted debe ser muy cuidadoso al escribir sus programas.

E. TECLAS PARA REVISAR LISTAS DE PROGRAMAS

"Pero," usted se preguntara a sí mismo, "¿cómo me puedo acordar de las líneas que he escrito?". ¡No se asuste programador intrépido! Este necesidad de revisar ha sido anticipada y provista con la [↑] (Flecha Hacia Arriba) y [↓] (Flecha Hacia Abajo) Usted debe considerar estas teclas como las teclas para revisar las líneas del programa. Al pulsar la tecla apropriada (en el modo PROgrama), uno puede cambiar hacia arriba o hacia abajo a través de las líneas del programa actual (un proceso conocido apropiadamente como "enrollando")

Cambie al modo PROgrama y utilice la tecla de Flecha Abajo para revisar las líneas de nuestro programa en orden ascendente. Después utilice la Flecha Arriba para mover hacia atrás hasta el comienzo del programa (línea 5). Observe que al retener cualquiera de las dos teclas con flechas, líneas sucesivas serán representadas automáticamente. (Esta función no se puede demostrar fácilmente con un programa de dos líneas)

Una vez que usted ha alcanzado una línea dada utilizando las teclas de revision del programa, puede proceder a corregir la línea utilizando las teclas Flecha Derecha o Flecha Izquierda. Austed le agradará descubrir que la operación de estas teclas en el modo PROgrama es idéntica a su operación en el modo RUN (vea el Capítulo 2) Las funciones de las teclas supresora e insertadora también están disponibles en la edición de sentencias, y se usan de la misma forma que antes.

NOTA Observe que después de hacer cualquier cambio en una línea de un programa, usted puede pulsar [ENTER] para efectuar estos cambios. NO UTILICE las teclas con flechas hacia Arriba o hacia Abajo, para proseguir al renglón adyacente siguiente, sin pulsar [ENTER] Si esto ocurre, cualquier corrección que usted haya ejectado se perderá.

F. Mirando de cerca a algunos de nuestros viejos amigos.

Ahora que sabemos como ingresar (ENTER), ejecutar (RUN), y corregir (EDIT) programas, vamos a expandir nuestro vocabulario de sentencias y órdenes útiles. Vamos a empezar examinando algunos de nuestros viejos amigos: el comando NEW, la sentencia LET, y la sentencia PRINT.

F-1. El Comando NEW

Como vimos en nuestros ejemplos previos de programación el comando NEW suprime todas las líneas del programa actualmente en la memoria.

Utilizaremos el comando NEW (en el modo PROgrama) antes de cada ejemplo de programa para estar seguros de que las instrucciones en la memoria de la PC-1500 son las instrucciones de nuestro programa actual. Aunque es posible (y a menudo deseable) tener varios programas en la memoria simultaneamente, nosotros pospondremos el uso de esta función para evitar confusión.

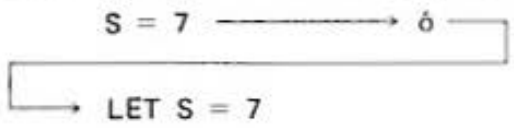
En el modo PROgrama, emita la orden NEW. ¿Qué efecto tienen ahora las teclas con flechas Arriba y Abajo?

F-2. La sentencia LET

No se alarme si no reconoce la sentencia LET. En realidad, la sentencia LET no es más que la sentencia Asignar disfrazada; y Asignar es una vieja amiga, ¿no? (¡Si usted no reconoce tampoco "Asignar", por favor lea de nuevo la sección de variables en el Capítulo 2, inmediatamente!).

Durante los primeros días del BASIC, cada sentencia empezaba con una palabra clave (como PRINT, INPUT, etc.) que indicaba lo que la instruccion iba a hacer. LET es la palabra clave utilizada para indicar que un valor dado será almacenado en una variable. Hoy en día todo el

mundo está de acuerdo en que la palabra LET no es realmente necesaria. Como resultado, la palabra clave LET es opcional en el BASIC PC-1500. Así, una sentencia que almacene el número 7 en una variable S puede ser escrita en cualquiera de las formas siguientes:



La única excepción, el lugar donde LET debe ser utilizado, es una asignación que ocurre como parte de una sentencia IF. Aunque no hemos discutido todavía la sentencia IF, no debe ser difícil de explicar (cruce los dedos).

Utilizando la sentencia IF usted puede escribir una instrucción como la siguiente:

IF expresión THEN sentencia

Si la sentencia que sigue a THEN es una Asignación, la palabra clave LET debe utilizarse. Esto resultará en la forma siguiente:

IF expresión THEN LET nombre-variable = expresión

NOTA: Omitiendo la sentencia LET en este caso, se producirá un ERROR 19 (otro mansaje de error).

F-3. La Sentencia PRINT

Para la mayoría de los programas que usted escriba, las instrucciones seguirán una estructura básica. Habrá instrucciones para leer datos, instrucciones para procesar los datos, e instrucciones para imprimir o representar los resultados. Esta estructura se denomina como ciclo de Input (entrada de la información), Procesamiento, y Output (salida o resultado).

La sentencia PRINT es la sentencia más utilizada para producir OUTPUT o imprimir. No es de sorprender, entonces, que la sentencia PRINT tenga diferentes variaciones. El formato general de la sentencia PRINT es la palabra PRINT seguida por un item o una lista de items para ser impresos. Estos incluyen una serie de caracteres, expresiones, o nombres de variables cuyos valores se imprimirán. Cada item en una lista está separado por una coma o un punto y coma.

Pulse el siguiente programa para demostrar como se imprimen items únicos. (Acuérdese de emitir el comando NEW antes de empezar.)

Listado del Programa:

```
10 Z$ ="VA SEGURO"
20 ZZ = 0
30 PRINT "QUIEN VA DESPACIO,"
40 PRINT ZZ
50 PRINT Z$
```

Digitaciones:

```
1 0 Z SHIFT S = SHIFT " V A SPACE
  S E G U R O SHIFT " ENTER
2 0 Z Z = 0 ENTER
3 0 P R I N T SHIFT " Q U I E N SPACE
  V A SPACE D E S P A C I O
  SHIFT . SHIFT " ENTER
4 0 P R I N T Z Z ENTER
5 0 P R I N T Z SHIFT $ ENTER
```

Después de ejecutado, este programa deberá producir tres renglones de resultado (OUTPUT) (pulse ENTER después de leer cada reglón):

```
DEG  RUN                    I  •
QUIEN VA DESPACIO
DEG  RUN                    I  •
                              0
DEG  RUN                    I  •
VA SEGURO
```

La primera de nuestras sentencias impresas (PRINT) (línea 30) representa una serie de caracteres (llamados "literales"). Observe que las comillas no se imprimen como parte del resultado. Estos símbolos son necesarios para delimitar el comienzo y el final de la secuencia de caracteres que usted desea imprimir. Esta secuencia puede incluir cualquier carácter con e excepción de las comillas mismas.

Los items en la segunda y tercera sentencia (líneas 40 y 50) deben ser reconocibles a todos los lectores como variables. Cuando el nombre de una variable es utilizado en una lista impresora, se imprime el valor de la variable. En nuestro programa, nosotros sabíamos cuales son los valores de ZZ y Z$ porque ellos fueron especificados en las líneas 10 y 20. Imprimiendo una variable "vacía" resultará en un cero o espacio en blanco dependiendo de si la variable es numérica o de carácter.

Los lectores más habiles habrán observado que las literales y las variables de caracteres se imprimen empezando en la parte izquierda de la representación visual. Esto se llama "justificado aizquierda". Por elcontrario, los números y los valores de variables numéricas son "justificados a derecha".

También es posible imprimir el resultado de una expresión que esté contenida en la sentencia PRINT. El siguiente programa de una línea ilustra esto:

Listado del Programa:

```
10 PRINT (1982 – 1956) * 365.25
```

Digitaciones:

[1] [Ø] [P] [R] [I] [N] [T] [(] [1] [9] [8] [2] [−] [1] [9] [5] [6] [)]
[*] [3] [6] [5] [·] [2] [5] [ENTER]

Como usted esperaba, el resultado, que es un número, se imprime "justificado aderecha".

Para mantener la eficiencia del programa, es mejor evitar expresiones computadas dentro de las sentencias PRINT a menos que esa sentencia (y sus expresiones asociadas) sea ejecutada solamente una vez en el programa.

Como la mayoría de los programas computan varios resultados al mismo tiempo, la impresión de varios items simultáneamente es una práctica común.

La más sencilla de las sentencias PRINT con múltiples items divide la pantalla de la representación visual en dos secciones. Cada sección contiene uno de dos artículos especificados en la lista de la sentencia PRINT. Los artículos están separados en la lista por una coma. Esamine este formato utilizando el programa siguiente.

Listado del Programa:

```
10  A = 2 * PI
20  PRINT  "2  X  PI = ", A
```

Digitaciones:

[1] [Ø] [A] [=] [2] [*] [P] [I] [ENTER]
[2] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [2] [SPACE]
[X] [SPACE] [P] [I] [=]
[SHIFT] ["] [SHIFT] [,] [A] [ENTER]

Ejecute el programa. Como en las sentencias PRINT con item único, los números son justificados a la derecha y los caracteres son justificados a la izquierda. En este caso la justificación (alineación) ocurre en cada una de las dos secciones de la representación visual.

Utilizando las técnicas de edición aprendidas, altere la línea 20 para que lea:

```
2Ø PRINT A, "=2  X  PI"
```

Hay varias formas de llevar a cabo esta edición. Un conjunto de pulsaciones podría ser:

[MODE] [↓] [↑] [↓] [▶] [▶] [SHIFT] [INS] [SHIFT] [INS] [A]
[SHIFT] [,] [▶] [SHIFT] [INS] [=]
[▶] [▶] [▶] [▶] [▶] [▶]
[SHIFT] [DEL] [▶] [SHIFT] [DEL] [SHIFT] [DEL] [ENTER]

El resultado de esta versión modificada le debe ayudar a identificar las dos secciones de la representación visual.

Más que dos items pueden representarse en el mismo renglón a través de una variación de la sentencia PRINT la cual utiliza el punto y coma. Construya y verifique el siguiente programa:

Listado del Programa:

```
10 BS = "  BE  "
20 T = 2
30 PRINT T; BS; "OR NOT"; 12/3 – 2; BS
```

Digitaciones:

[1] [Ø] [B] [SHIFT] [$] [=] [SHIFT] ["] [SPACE] [B] [E] [SPACE]
[SHIFT] ["] [ENTER]
[2] [Ø] [T] [=] [2] [ENTER]
[3] [Ø] [P] [R] [I] [N] [T] [T] [SHIFT] [:] [B]
[SHIFT] [$] [SHIFT] [:]
[SHIFT] ["] [O] [R] [SPACE] [N] [O] [T] [SHIFT] ["]
[SHIFT] [:] [1] [2] [/] [3] [–] [2] [SHIFT] [:]
[B] [SHIFT] [$] [ENTER]

Ejecutando esta composición electrónica deberá producir el siguiente resultado:

```
                DEG      RUN                I    •
2  BE  OR  NOT  2  BE
```

Recordemos que en inglés el '2' (two) se pronuncia "tu".

No es Shakespeare, pero ilustra la acción de la sentencia PRINT cuando los artículos imprimibles están separados por punto y coma. En este formato los elementos se representan contiguamente con una separación mínima entre ellos. Esta capacidad es muy utilizada para crear resultados que parezcan "naturales" (es decir, resultados que salen unidos).

NOTA: Si la longitud de la información en la representación visual excede el espacio disponible en la representación visual (26 caracteres), los caracteres al final de la lista impresa no se verán.

Otro uso del punto y coma es al final de la sentencia PRINT. En este caso, el punto y coma indica que el resultado actualmente en la representación visual debe guardarse y que cualquier nuevo resultado (de la próxima sentencia PRINT) deberá unirse al viejo resultado en el mismo renglón. Este proceso es más fácil de programar que de describir, así que vamos a experimentar con el programa siguiente (no olvide el comando NEW):

Listado del Programa:

```
100 PRINT  "COLORIN ";
110 PRINT  "COLORADO"
```

Digitaciones:

1 Ø Ø P R I N T SHIFT " C O L O R I N
SPACE SHIFT " SHIFT ; ENTER
1 1 Ø P R I N T SHIFT " C O L O R A D
O SHIFT " ENTER

Ahora ejecútelo. Como de costumbre, usted tiene que pulsar ENTER después que la primera sentencia PRINT haya representado "COLORIN". Debido al punto y coma "COLORIN" es retenido y "COLORADO" comparte la representación visual con él. Compare esto con la ejecución del programa siguiente, el cual no utiliza el punto y coma, y las diferencias se clarificarán:

Listado del Programa:

```
10 PRINT "COLORIN "
20 PRINT "COLORADO"
```

Digitaciones

1 Ø P R I N T SHIFT " C O L O R I N
SPACE SHIFT " ENTER
2 Ø P R I N T SHIFT " C O L O R A D O
SHIFT " ENTER

El próximo ejemplo de programación nos muestra que el punto y coma puede unir tantos renglones de resultados como caben en la representación visual. Imprimir demasiada información en la representación visual es un error que la SHARP no señalará. Es su reponsabilidad, como programador, asegurarse de que esto no ocurra. Pruebe el siguiente programa:

Listado del Programa:

```
20 PRINT "DO ";
40 PRINT "RE ";
60 PRINT "MI ";
80 PRINT "FA ";
100 PRINT "SOL ";
120 PRINT "LA ";
140 PRINT "TI ";
160 PRINT "DO";
```

Digitaciones:

[2] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [D] [O] [SPACE]
[SHIFT] ["] [SHIFT] [:] [ENTER]

[4] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [R] [E] [SPACE]
[SHIFT] ["] [SHIFT] [:] [ENTER]

[6] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [M] [I] [SPACE]
[SHIFT] ["] [SHIFT] [:] [ENTER]

[8] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [F] [A] [SPACE]
[SHIFT] ["] [SHIFT] [:] [ENTER]

[1] [Ø] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [S] [O] [L] [SPACE]
[SHIFT] ["] [SHIFT] [:] [ENTER]

[1] [2] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [L] [A] [SPACE]
[SHIFT] ["] [SHIFT] [:] [ENTER]

[1] [4] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [T] [I] [SPACE]
[SHIFT] ["] [SHIFT] [:] [ENTER]

[1] [6] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [D] [O] [SHIFT] ["]
[SHIFT] [:] [ENTER]

Ahora ejecute el programa, pulsando [ENTER] repetidamente mientras usted canta alegremente cada nota en la escala. (¡Está bien!, ¡Está bien!, usted no tiene que cantar, pero todavia tiene que pulsar [ENTER]).

G. La Sentencia PAUSE

La sentencia PAUSE es una forma semi-automática de la sentencia PRINT. Muestra en la representación visual los varios items en su lista asociada por un período fijo y breve. El usuario se libera asi de la carga de estar pulsando [ENTER] . Piense en la sentencia PAUSE como una sentencia PRINT seguida de una cuenta a cero. Cuando la cuenta a cero se acaba, el programa continúa.

Los formatos de la sentencia PAUSE son idénticos a los de la sentencia PRINT. Todas las técnicas que hemos discutido en relación con la sentencia PRINT son aplicables a la sentencia PAUSE, aunque el resultado variará un poco por supuesto. Para ilustrar un uso de la sentencia PAUSE, vamos a escribir de nuevo nuestro programa de la escala musical:

Listado del Programa:

```
10 PAUSE "DO ";
20 PAUSE "RE ";
30 PAUSE "MI ";
40 PAUSE "FA ";
50 PAUSE "SOL ";
60 PAUSE "LA ";
70 PAUSE "T I ";
80 PAUSE "DO";
```

Digitaciones:

1 Ø P A U S E SHIFT " D O SPACE
SHIFT " SHIFT : ENTER

2 Ø P A U S E SHIFT " R E SPACE
SHIFT " SHIFT : ENTER

3 Ø P A U S E SHIFT " M I SPACE
SHIFT " SHIFT : ENTER

4 Ø P A U S E SHIFT " F A SPACE
SHIFT " SHIFT : ENTER

5 Ø P A U S E SHIFT " S O L SPACE
SHIFT " SHIFT : ENTER

6 Ø P A U S E SHIFT " L A SPACE
SHIFT " SHIFT : ENTER

7 Ø P A U S E SHIFT " T I SPACE
SHIFT " SHIFT : ENTER

8 Ø P A U S E SHIFT " D O
SHIFT " SHIFT : ENTER

Observe lo que pasa después que la última nota es representada; el programa acaba y la representación visual vuelve a la condición de prompt. Esto ocurre porque no hay otras secuencias siguiendo la última sentencia PAUSE. Para parar la representación visual después de representar la última nota, nosotros podemos cambiar la línea 80 a:

```
80 PRINT "DO"
```

Digitaciones:

8 Ø P R I N T SHIFT " D O
SHIFT " ENTER

Después de todo, no hay razón por la que no podamos mezclar la sentencia PRINT y la sentencia PAUSE a través de nuestro programa. Pruebe esto usted mismo.

H. INPUT

Utilizando las distintas formas de la sentencia PRINT, solas o combinadas, presentarse puede correctamente al usuario de la computadora. La mayor parte de los items que son impresos, sin embargo, son el resultado de procesar alguna información. Esta información inicial es dada al programa por el mismo usuario. La instrucción que controla este proceso es la instrucción INPUT.

En su forma sencilla, INPUT nombre-variable, la instrucción INPUT simplemente muestra un? (signo de interrogación) y luego espera que el usuario escriba la información requerida. Lo que se requiere depende de la variable que aparece como parte de la instrucción INPUT. Por ejemplo,

si la variable es numérica, el usuario debe pulsar un número que será almacenado en la variable.

¿Puede ver problemas en este formato de la secuencia INPUT? Muy bien. Como algunos de ustedes se habrán dado cuenta, a menos que el usuario del programa sea también el programador (y probablemente ni siquiera entonces), él no sabrá qué clase de datos ingresar cuando vea el signo de interrogación. **El programador tiene la obligación de mantener al usuario informado** (y probablemente ni siquiera entonces), él no sabrá qué clase de datos ingresar cuando vea el signo de interrogación. **El programador tiene la obligación de mantener al usauario informado continuamente.**

Como ejemplo de un programa que falla en hacer esto, pulse lo siguiente:

Listado del Programa:

```
10  A = 0
20  INPUT  A
30  PRINT  A * PI
```

Digitaciones

[1] [Ø] [A] [=] [Ø] [ENTER]

[2] [Ø] [I] [N] [P] [U] [T] [A] [ENTER]

[3] [Ø] [P] [R] [I] [N] [T] [A] [*] [P] [I] [ENTER]

Ahora imagine al usuario ejecutando el programa ... La primera cosa que aparece en la representación visual es el signo de interrogación. El usuario con buenos conocimientos se dará cuenta de que se requieren algunos datos. Pero él no sabrá cuales deben ser esos datos. Suponga que él empiece a adivinar, y con suerte, pulse un número. De repente un número largo y complicado aparece. ¿Qué significa? El pulsa ENTER para continuar, pero el programa acaba. Desde su punto de vista, la experiencia ha sido una pérdida de tiempo total. ¿Porqué? A causa de mala programación.

Una solución para resolver este problema es utilizar las sentencias PRINT o PAUSE para ayudar al usuario. Con esta idea podemos escribir de nuevo nuestro programa de esta manera:

Listado del Programa:

```
10  A = 0
20  PAUSE "ENTRE CUALQUIER NUMERO"
30  INPUT  A
40  AP = A * PI
50  PRINT  A; " X PI = "; AP
```

Digitaciones:

[1] [Ø] [A] [=] [Ø] [ENTER]

[2] [Ø] [P] [A] [U] [S] [E] [SHIFT] ["] [E] [N] [T] [R] [E] [SPACE]
[C] [U] [A] [L] [Q] [U] [I] [E] [R] [SPACE]
[N] [U] [M] [E] [R] [O] [SHIFT] ["] [ENTER]

[3] [Ø] [I] [N] [P] [U] [T] [A] [ENTER]

[4] [Ø] [A] [P] [=] [A] [*] [P] [I] [ENTER]

[5] [Ø] [P] [R] [I] [N] [T] [A] [SHIFT] [:] [SHIFT] ["] [SPACE]
[X] [SPACE] [P] [I] [SPACE] [=]
[SPACE] [SHIFT] ["] [SHIFT] [;] [A] [P] [ENTER]

Esta versión es más útil porque "instiga" al usuario a efectuar una entrada y identifica la salida. tifica la salida.

La operación de instigar (mostrando un mensaje por cada información a entrar) es tan común que formas más avanzadas de la sentencia INPUT han sido creadas para incorporarla. La primera de estas se escribe con un punto y coma:

INPUT "literal" ; nombre-variable

De esta manera, con la ayuda de literales facilitamos la comunicación entre la computadora y el usuario. Esta serie de caracteres será presentada en el visor reemplazando al signo de interrogación y quedard la PC-1500 a la espera de la información requerida por el usuario. Esta forma de la sentencia de INPUT nos permite escribir de nuevo nuestro ejemplo previo así:

Listado del Programa:

```
10  A = 0
20  INPUT "ENTRE CUALQUIER NUMERO "; A
30  AP = A*PI
40  PRINT A; " X PI = "; AP
```

Digitaciones:

[1] [Ø] [A] [=] [Ø] [ENTER]

[2] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [E] [N] [T] [R] [E] [SPACE]
[C] [U] [A] [L] [Q] [U] [I] [E] [R] [SPACE]
[N] [U] [M] [E] [R] [O] [SPACE]
[SHIFT] ["] [SHIFT] [;] [A] [ENTER]

[3] [Ø] [A] [P] [=] [A] [*] [P] [I] [ENTER]

[4] [Ø] [P] [R] [I] [N] [T] [A] [SHIFT] [:] [SHIFT] ["] [SPACE]
[X] [SPACE] [P] [I] [SPACE] [=]
[SPACE] [SHIFT] ["] [SHIFT] [;] [A] [P] [ENTER]

Cuando usted ejecute este ejemplo, se dará cuenta de la diferencia entre la sentencia INPUT y la sentencia PAUSE.

La segunda forma de INPUT es casi idéntica a la primera excepto que utiliza una coma en vez del punto y coma:

INPUT "literal", nombre-variable

Cuando esta forma de la sentencia se ejecuta, la serie de caracteres asociados se representa, y la computadora sepera otra vez para una entrada. Esta vez, sin embargo, cuando el usuario empieza a escribir, el literal se borra y los datos del usuario aparecen en su lugar. Esto permite la entrada de datos después de un mensaje largo sin llegar al final del visor.

Cambie el punto y coma por una coma en la línea 20 de nuestro programa y ejecútelo de nuevo.

Por supuesto, la entrada de datos no está limitada a números, como nuestros ejemplos han indicado hasta ahora. Para ingresar caracteres nosotros especificamos una variable de caracteres en la sentencia INPUT, como en este brillante y deductivo programa:

Listado del Programa:

```
10 INPUT "PULSE SU APELLIDO "; L$
20 INPUT "PULSE SU NOMBRE "; F$
30 I$ = LEFT$ (F$, 1)+" . "+ LEFT$ (L$, 1)+" . "
40 PAUSE "BUENO,"
50 PRINT "SUS INICIALES SON "; I$
```

Digitaciones:

[1] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [P] [U] [L] [S] [E] [SPACE]
[S] [U] [SPACE] [A] [P] [E] [L] [L] [I] [D] [O]
[SPACE] [SHIFT] ["] [SHIFT] [;]
[L] [SHIFT] [$] [ENTER]

[2] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [P] [U] [L] [S] [E] [SPACE]
[S] [U] [SPACE] [N] [O] [M] [B] [R] [E]
[SPACE] [SHIFT] ["]
[SHIFT] [;] [F] [SHIFT] [$] [ENTER]

[3] [Ø] [I] [SHIFT] [$] [=] [L] [E] [F] [T] [SHIFT] [$] [(] [F]
[SHIFT] [$] [SHIFT] [,] [1]
[)] [+] [SHIFT] ["] [.] [SHIFT] ["] [+]
[L] [E] [F] [T] [SHIFT] [$] [(] [L] [SHIFT] [$]
[SHIFT] [,] [1] [)] [+] [SHIFT] ["] [.]
[SHIFT] ["] [ENTER]

[4] [Ø] [P] [A] [U] [S] [E] [SHIFT] ["] [B] [U] [E] [N] [O]
[SHIFT] [,] [SHIFT] ["] [ENTER]

[5] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [S] [U] [S] [SPACE]
[I] [N] [I] [C] [I] [A] [L] [E] [S] [SPACE]
[S] [O] [N] [SPACE] [SHIFT] ["] [SHIFT] [;]
[I] [SHIFT] [$] [ENTER]

Nota: No se preocupe por la línea 30; utiliza técnicas avanzadas que usted aprenderá después.

Además de acaptar un único item de datos, la instrucción INPUT puede ser utilicada para recoger y almacenar varios de estos valores. Para programar este proceso, uno simplemente hace una lista de las variables que recibirán los datos dentro de la instrucción INPUT. Cada variable en la lista está separada por una coma.

Como un ejemplo de entradas multiples de artículos constituya este programa estadístico:

Listado del Programa:

```
10 W=0 : X=0 : Y=0 : Z=0
20 INPUT "ENTRE 4 NUMEROS ",W,X,Y,Z
30 S=W+X+Y+Z : A=S/4
40 PAUSE "TOTAL= "; S
50 PRINT "PROMEDIO ES "; A
```

Digitaciones:

[1] [Ø] [W] [=] [Ø] [SHIFT] [:] [X] [=] [Ø] [SHIFT] [:] [Y] [=] [Ø]
[SHIFT] [:] [Z] [=] [Ø] [ENTER]
[2] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [E] [N] [T] [R] [E] [SPACE] [4]
[SPACE] [N] [U] [M] [E] [R] [O] [S] [SPACE] [SHIFT] ["]
[SHIFT] [,] [W] [SHIFT] [,] [X] [SHIFT] [,]
[Y] [SHIFT] [,] [Z] [ENTER]
[3] [Ø] [S] [=] [W] [+] [X] [+] [Y] [+] [Z] [SHIFT] [:] [A] [=] [S] [/] [4]
[ENTER]
[4] [Ø] [P] [A] [U] [S] [E] [SHIFT] ["] [T] [O] [T] [A] [L]
[=] [SPACE] [SHIFT] ["]
[SHIFT] [:] [S] [ENTER]
[5] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["]
[P] [R] [O] [M] [E] [D] [I] [O] [SPACE] [E] [S] [SPACE] [SHIFT] ["]
[SHIFT] [:] [A] [ENTER]

Cuando se ejecute, este programa le invitará a entrar 4 números. El primer número que usted escriba reemplazará la prompt porque utilizamos una coma directamente después de la serie de caracteres en la sentencia INPUT. Cuando el primer número está completamente escrito, usted

debe pulsar ENTER. Un signo de interrogación precederá cada número sucesivo, cada uno de los cuales debe también ir seguido por ENTER.

Si hubiéramos utilizado un punto y coma en la sentencia INPUT, el primer número pulsado no habría reemplazado a la prompt pero habría compartido el renglón con él. Todas las entradas sucesivan serían iguales que con la coma. Compruebe esto usted mismo alternando la línea 20 para que lea:

```
20 INPUT "ENTRE 4 NUMEROS ";W, X, Y, Z
```

Digitaciones:

[2] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [E] [N] [T] [R] [E] [SPACE] [4]
[SPACE] [N] [U] [M] [E] [R] [O] [S] [SPACE]
[SHIFT] ["] [SHIFT] [;] [W] [SHIFT] [,] [X] [SHIFT] [,]
[Y] [SHIFT] [,] [Z] [ENTER]

I. Consideraciones y Sugerencias Provechosas

Como usted ha sido tan diligente y paciente, se añade esta sección como una pequeña compensación. Yo sé que no es tan buena como el dinero pero la información contenida aquí puede ahorrarle algún esfuerzo.

I-1. Abreviaturas

Aquellos de ustedes cuyos dedos de oro no vuelan suavemente sobre las teclas de la computadora, habrán observado que nuestros ejemplos de programas han ido creciendo y son cada vez más largos. En una explosión de compasión, los hábiles diseñadores de SHARP anticiparon sus dificultades. Ellos han puesto en la SHARP la habilidad de reconocer abreviaturas de sentencias y comandos frecuentemente utilizados.

La forma genéral de una abreviatura es una o más letras designadas seguidas por un punto. El punto es muy importante para evitar confusión con los nombres variables. Como ejemplo, pulse el programa siguiente cuyas secuencias están abreviadas:

Listado del Programa:

```
15 PA. "HOLA, HUMANO."
25 I. "COMO SE LLAMA? "; N$
35 P. "ENCANTADO, "; N$
```

Digitaciones:

```
1 5 P A . SHIFT " H O L A SHIFT ,
  H U M A N O . SHIFT " ENTER
2 5 I . SHIFT " C O M O SPACE S E SPACE
  L L A M A SHIFT ? SPACE SHIFT "
  SHIFT : N SHIFT $ ENTER
3 5 P . SHIFT " E N C A N T A D O
  SHIFT , SPACE SHIFT " SHIFT :
  N SHIFT $ ENTER
```

Al completar la entrada de cada línea (pulsando la tecla ENTER), observe que la SHARP expande cualquiera de las abreviaturas en esa línea. La claridad resultante será muy valiosa para usted más tarde, cuando compruebe el programa para encontrar errores. Un beneficio oculto adicional es que el espacio utilizado para almacenar las avreviaturas de las sentencias no es más (ni menos) que el espacio utilizado para almacener la sentencia entera. Así la facilidad de abreviatura es estrictamente una conveniencia para usted, el programador.

Para mayor facilidad de lectura no utilizaremos estas abreviaturas en nuestros listados de programas de ejemplos, (aunque usted puede hacerlo cuando pulse los programas). Las siguientes son abreviaturas permitidas para comandos y sentencias ya introducidas:

PRINT	P. PR. PRI.
PAUSE	PA. PAU.
INPUT	I. IN. INP.
RUN	R.

Una lista completa de abreviaturas se incluye al final de este manual en un Apéndice.

I-2. Sentencias Múltiples y Los Dos Puntos

Como fue mencionado en la sección de líneas numeradas, varias sentencias pueden compartir la misma línea. La SHARP ejectuará las sentencias en secuencia de izquierda a derecha. Para que puede distinguir el final de una sentencia del principio de la siguiente, algún carácter señalador debe ser utilizado. Este carácter es los Dos Puntos (:). Por analogía, la función de los puntos es similar a la función de los períodos que separan las sentencias en esta página; indicando al lector cuando tiene que parar de leer.

El uso de los dos puntos ha sido ya ilustrado en varios de los últimos programas. Su forma general es la siguiente:

número-línea sentencia 1 : sentencia 2 : sentencia 3 (etc.)

La pregunta de cuándo utilizar los dos puntos es una cuestión de estilo de programación. El uso indiscriminado de los dos puntos para escribir programas muy compactos produce programas que son muy difíciles de leer, corregir, o expandir. Como las ventajas más importantes de la PC-1500 son que es personal e interactiva, es altamente deseable que usted pueda modificar y extender los programas para acomodar sus propias necesidades. Por lo tanto, le recomendamos que se refrene en el uso de los dos puntos.

Si usted utiliza los dos puntos, es muy ventajoso agrupar sólo aquellas sentencias que están conceptualmente relacionadas. Es decir, coloque sólo aquelas sentencias que consiguen llevar a cabo una simple tarea, de las muchas que abarcan el programa, en una línea.

Por ejemplo, considere el siguiente programa que lee tres números, encuentra su promedio, e imprime la suma de estas diferencias:

Listado del Programa:

```
10  N1 = 0 : N2 = 0 : N3 = 0
20  INPUT "PULSE 3 NUMEROS", N1, N2, N3
30  A = (N1 + N2 + N3) / 3
40  D1 = N1 – A : D2 = N2 – A : D3 = N3 – A
50  SD = D1 + D2 + D3
60  PRINT "SUMA DE DIFERENCIAS = "; SD
```

Digitaciones:

```
1 Ø N 1 = Ø SHIFT : N 2 = Ø SHIFT : N 3
  = Ø ENTER
2 Ø I N P U T SHIFT " P U L S E SPACE 3
  SPACE N U M E R O S SHIFT " SHIFT ,
  N 1 SHIFT , N 2 SHIFT , N 3 ENTER
3 Ø A = ( N 1 + N 2 + N 3 ) / 3 ENTER
4 Ø D 1 = N 1 – A SHIFT : D 2 = N 2 – A
  SHIFT : D 3 = N 3 – A ENTER
5 Ø S D = D 1 + D 2 + D 3 ENTER
6 Ø P R I N T SHIFT " S U M A SPACE
  D E SPACE D I F E R E N C I A S
  = SPACE
  SHIFT " SHIFT ; S D ENTER
```

Observe como cada línea en el programa lleva a cabo un paso único y necesario. La línea 10 borra cualquier valor previo que pueda estar contenido en las variables N1, N2, y N3 (Esta es una precaución en caso que el usuario falle en entrar los tres números). La línea 20 permite el ingreso de datos. La línea 30 promedia los números. La línea 40 computa las tres diferencias. La línea 50 suma las diferencias y la línea 60 representa el resultado.

No es por accidente que las instrucciones de cada línea corresponden a la descripción del programa en inglés. Por el contrario, este es uno de los principios de buena programación que usted debe tratar de seguir.

A diferencia del caso de las abreviaturas, el uso de los dos punto afecta a la cantidad de memoria utilizada para almacenar el programa. Esta es la mayor justificación para utilizar los dos puntos cuando se coloquen varias sentencias en una línea. Cada línea numerada reserva varios "steps" (pasos) de programación (una unidad de almacenamiento) y, en consecuencia al utilizar un menor número de líneas en un programa, será menor la cantidad de memoria requerida por el programa.

La sugerencia final acerca de la utilización de los dos puntos es que cada programador debe balancear la legibilidad de cada programa y su facilidad de modificación, contra las necesidades de almacenamiento de su aplicación.

J. Corrección de Errores en el modo PROgrama.

Aunque las abreviaturas y los dos puntos nos ayudan a entrar programas facilmente, no pueden evitar que los mejores de nosotros cometamos errores. Incluso programadores profesionales se descuidan y no pueden encontrar errores cuando revisan sus propios programas. Lo que esto quiere decir es que, más pronto o más tarde, usted encontrará una falla mientras ejecuta su programa (si no lo ha hecho ya). La mayoría de estos errores se corregen fácilmente si usted los acepta simplemente como un acertijo o para ser resuelto y cuidadosamente encuentra el problema.

Varias funciones especiales de la PC-1500 le asistirán en esto.

Al descubrir una sentencia incorrecta, la computadora se parará e indicará el problema con un breve mensaje como:

```
                                RUN
ERROR  1  IN  20
```

Vamos a crear un programa con un error deliberado para ilustrar esta función. Entre:

Listado del Programa:

```
25 PAUSE "COLORIN COLORADO"
50 PRIMT "ESTE CUENTO SE HA ACABADO"
```

Digitaciones:

[2] [5] [P] [A] [U] [S] [E] [SHIFT] ["] [C] [O] [L] [O] [R] [I] [N] [SPACE]
[C] [O] [L] [O] [R] [A] [D] [O] [SHIFT] ["] [ENTER]
[5] [0] [P] [R] [I] [M] [T] [SHIFT] ["] [E] [S] [T] [E] [SPACE] [C] [U]
[E] [N] [T] [O] [SPACE] [S] [E] [SPACE] [H] [A] [SPACE] [A] [C] [A]
[B] [A] [D] [O] [SHIFT] ["] [ENTER]

Ahora ejecute el programa. Cuando el mensaje de error aparezca, oprima la tecla [↑] (Flecha Arriba). Mientras que usted retenga esta tecla, la representación visual mostrará el renglón en el

cual la SHARP encontró un error. El cursor parpadeante le indica la naturaleza del problema.

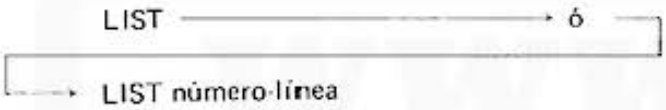
Para corregir la secuencia equivocada, pulse [CL] para parar el programa y cambie al modo PROgrama. Pulse la tecla con Flecha Arriba (pero no la retenga) y la representación visual mostrará otra vez la línea euivoczda:

```
50 PRIMT "ESTE CUENTO SE HA ACABADO"
```

Usted puede proceder a editar la línea utilizando tecla familiares como INSerta, DEL, y las teclas con flechas. Cuando haya acabado de editar, usted debe pulsar [ENTER] para ordenarle a la computadora SHARP que almacene la línea corregida.

K. El comando LIST

Otra forma de visualizar una línea particular de un programa es utilizando el comando LIST cuya forma es:



Si no se da un número-línea, la primera línea del programa será representada.

Si un número-línea es especificado, la línea del programa con ase número será representada. Si no hay ninguna línea en el programa con el número dado, la próxima línea del programa cuyo número es mayor será representada. Por ejemplo, en el programa siguiente:

```
15 PRINT "LA SANTA MARIA"
30 PRINT "FUE UN";
45 PRINT "BUQUE"
```

la orden LIST 40 hará representar la línea 45. La orden LIST representará la línea 15 y la orden LIST 30 representará la línea 30 (¿Algo más?).

NOTA: Si usted especifica un número de línea mayor que el mayor numerado de línea existente, un ERROR 11 ocurrirá.

L. Cuanto Más Mejor

Como hemos mencionado previamente, es posible mantener más de un programa en la memoria al mismo tiempo. El truco para hacer esto es dar a cada programa su propio rango de números de línea. Por ejemplo, un programa podría tener líneas numeradas del 10 al 200 mientras que las de un segundo programa serían numeradas del 300 al 500. Por supuesto, usted debe tener cuidado de no intercambiar accidentalmente las secuencias de varios programas previos (numerando incorrectamente) o podrán aparecer resultados impredecibles).

L-1. La sentencia END

Otro problema que aparece mientras usted almacena varios programas simultáneamente es que cada programa es sencillamente un grupo de líneas numeradas ejecutadas en forma ascendente. Entonces, ¿cómo va a saber la SHARP cuando ha acabado de ejecutar las sentencias de un programa dado? La respuesta es que para indicarle a la computadora que ha alcanzado el final de un programa se utiliza la sentencia END.

Hasta ahora no hemos utilizado, ni necesitado, la sentencia END. La SHARP simplemente ha ejecutado todas nuestras líneas de programas. Cuando se acabaron, la computadora decidió que el programa había acabado y volvió a esperar un nuevo mandato. Ahora, debemos decirle a la SHARP que pare de ejecutar instrucciones antes que invada las líneas del próximo programa.

Para ilustrar el uso de programas múltiples ingrese las siguientes líneas:

```
10 PAUSE "ESTABA LA PAJARA PINTA"
20 PAUSE "SENTADA"
30 PRINT "EN EL VERDE LIMON"
40 END
200 PAUSE "A PALABRAS NECIAS"
210 PRINT "OIDOS SORDOS"
220 END
```

L-2. RUN línea número

Bien, ahora que usted tiene dos programas en la memoria, ¿cómo va usted a ejecutarlos individualmente? Si usted tuvo suficiente valor para ejecutar la orden normal RUN, habrá descubierto que es perfectamente adecuado para iniciar el primer programa. (Si usted no tuvo suficiente valor antes, inténtelo ahora). Para comenzar el segundo programa necesitamos utilizar la variación de la orden RUN que dice a la SHARP en qué línea debe empezar. Esta es la orden: RUN línea-número. Para empezar el segundo programa escriba:

[R] [U] [N] [2] [Ø] [Ø]

¡y voilá, se hizo!

Como la mayoría de los comandos últiles, el comando RUN línea-número puede causar problemas. Porque ordena a la SHARP donde empezar y, como la computadora SHARP es una servidora muy servicial, los programs pueden empezar en el medio. Esto, como podrá adivinar, no debería ser hecho exprofeso. Si se hace, producirá algunos resultados extraños. Inténtelo en nuestro primer programa despachando la orden:

```
RUN 30
```

Este es una falla pequeña comparada con la que podría haber ocurrido en un programa más complejo.

M. Sentencias de Control

Hasta ahora nuestros programas han sido una secuencia consecutiva de instrucciones, cada una realizada una vez por la computadora. Aquellas personas con experiencia en dar instrucciones comprenderán que éste no es siempre el mejor camino para llevar a cabo un trabajo. A veces usted le permite a la persona que le escucha que escoja uno de varios planes. Otras veces usted "condensa" sus instrucciones incluyendo un mandato como, "Ahora repita los últimos tres pasos hasta que usted haya acabado."

Estas capacidades han sido incorporadas en las sentencias del BASIC llamadas sentencias de control. Estas sentencias determinan cuándo, cómo, y cuántas veces otras sentencias serán ejecutadas. Las sentencias de control le permiten construir programas muy versátiles y poderosos. En las próximas secciones discutiremos las sentencias de control más importantes del BASIC; IF . . . THEN, GOTO, y GOSUB.

N. IF y THEN

El potencial para escoger dentro del programa BASIC está provisto por la sentencia IF. Un programa equipado con una sentencia IF puede evaluar entradas y hacer decisiones.

Listado del Programa:

```
10 PAUSE "ESTAS DORMIDO?"
20 INPUT "ESCRIBA SI O NO "; SX$
30 IF SX$ = "SI" THEN PRINT "OH, LO SIENTO MUCHO"
40 END
```

Digitaciones:

[1] [0] [P] [A] [U] [S] [E] [SHIFT] ["] [E] [S] [T] [A] [S]
[SPACE] [D] [O] [R] [M] [I] [D] [O] [SHIFT] [?]
[SHIFT] ["] [ENTER]
[2] [0] [I] [N] [P] [U] [T] [SHIFT] ["] [E] [S] [C] [R] [I] [B] [A]
[SPACE] [S] [I] [SPACE] [O] [SPACE] [N] [O] [SPACE] [SHIFT] ["]
[SHIFT] [;] [S] [X] [SHIFT] [$]
[3] [0] [I] [F] [S] [X] [SHIFT] [$] [=] [SHIFT] ["] [S] [I] [SHIFT] ["]
[T] [H] [E] [N] [P] [R] [I] [N] [T] [SHIFT] ["] [O] [H]
[SHIFT] [,] [SPACE] [L] [O]
[SPACE] [S] [I] [E] [N] [T] [O] [SPACE] [M] [U] [C] [H] [O]
[SHIFT] ["] [ENTER]
[4] [0] [E] [N] [D] [ENTER]

Este programa solamente producirá un resultado si usted respondió afirmativamente. La forma general de la sentencia IF es ilustrada en la línea 30:

IF condición THEN sentencia

Durante la ejecución, se ejecuta la prueba contenida en la cláusula IF. Si la sentencia es ejecutada, o no, depende del resultado de la prueba. La prueba es por lo general una desigualdad y se llama una "condición". Recuerde que desigualdades son comparaciones que no son ni ciertas ni falsas. Si la desigualdad es cierta, entonces la sentecia es ejecutada. Si la desigualdad es falsa, entonces la sentencia es ignorada.

En nuestro programa-ejemplo, la prueba es si la variable SX$ es igual a (contiene) la serie de caracteres "SI". Si lo es, y sólo si lo es, entonces la instrucción PRINT es ejecutada. Si SX$ no es igual a "SI" entonces la instrucción PRINT es ignorada. **En cualquier caso, no importa si la sentencia PRINT es ejecutada o ignorada, la SHARP procederá, como si fuera normal, al próximo renglón.** (En nuestro programa-ejemplo, el próximo renglón es el número 40).

Observe que podríamos haber invertido nuestra prueba alterando varias líneas de la manera siguiente:

```
30 IF SX$ = "NO" THEN END
40 PRINT "OH, LO SIENTO MUCHO"
```

Este programa, sin embargo, no es igual al original. Esto permite que nuestra computadora hable con cualquiera que escriba equivocadamente o no responda con un "NO". Además, este programa realmente tiene dos puntas; la sentencia END en el renglón 30 y el END sobreentendido en el renglón 40. Varios finales no son el resultado de una buena programación. Al contrario, estas sentencias demuestran, sin embargo, que el orden correcto de las sentencias y corrección adecuada son necesarias para que el programa funcione correctamente. En la próxima sección, observaremos una tercera forma de escribir nuestro programa usando la sentencia GOTO y la forma de resolver los problemas de la segunda versión.

Una advertencia más es necesaria. Si la sentencia que le sigue a THEN es una sentencia de Asignación, la palabra clave LET debe ser utilizada. La omisión de hacer esto causará un error de cálculo. Esto fue discutido en la sección acerca de la sentencia LET.

0. GOTO

Usted habrá observado, en la última sección, que nuestras opciones son limitadas después que se hizo la prueba en la sentencia IF. Se nos permitió ejecutar solamente una sentencia si la condición hubiera sido cierta. Por conveniencia, nos gustaría ejecutar varias sentencias. La sentencia GOTO nos permite hacer esto.

La sentencia GOTO modifica el flujo en la ejecución de las sentencias. Esta le dice a la SHARP, que vaya (GOTO) a otra línea diferente de la línea que sigue y comience a ejecutar las sentencias secuencialmente desde allí. El efecto de este salto es que algunas sentencias pueden ignorarse completamente. Por ejemplo, revise el siguiente programa:

Listado del Program

```
10 PAUSE "EVADA ";
20 GOTO 50
30 PRINT X*3/4+2
40 PRINT "UN LIBRO QUE EMPLEE";
50 PRINT "OFUSCACION!"
60 END
```

Digitaciones:

```
1 Ø P A U S E SHIFT " E V A D A
  SPACE SHIFT " SHIFT : ENTER
2 Ø G O T O 5 Ø ENTER
3 Ø P R I N T X * 3 / 4 + 2
4 Ø P R I N T SHIFT " U N SPACE
  L I B R O SPACE Q U E SPACE
  E M P L E E SHIFT " SHIFT : ENTER
5 Ø P R I N T SHIFT " O F U S C
  A C I O N SHIFT ! SHIFT " ENTER
6 Ø E N D ENTER
```

Normalmente, por supuesto, este programa se ejecutaría en secuencias de números de línea ascendentes. El efecto del GOTO en la línea 20, sin embargo, hace que la computadora prosiga inmediatamente a la línea 50 y continúe en secuencia ascendente desde allí. Las líneas 30 y 40 no son ejecutadas nunca.

La forma general de la frase GOTO es:

GOTO expresión

donde:

la expresión evalúa a un número que es una línea del programa válida (por ejemplo del 1 al 65279).

NOTA: Especificando un número de línea que no existe resultará un ERROR 11.

Si deseamos tener varias instrucciones ejecutadas como resultado de una cierta decisión, nosotros utilizaremos sentencias GOTO en combinación con la sentencia IF:

```
10   IF prueba THEN GOTO 100
20   | Aquí están las sentencias |
 :   | ejecutadas sólo si la     |
 :   | prueba es falsa.          |
90   GOTO 200
100  | Aquí están las sentencias |
 :   | ejecutadas sólo si la     |
 :   | prueba es cierta.         |
200  | Aquí están las sentencias |
 :   | siempre ejecutadas.       |
 :
999  END
```

La lógica de este programa es aplicable en muchas situaciones. Cualquier número de sentencias pueden ser insertadas en cada una de las secciones que son formadas por la sentencia GOTO.

Como ejemplo de esta estructura en la práctica, la sección siguiente de una cuenta de ahorros

determina si la cantidad de una entrada determinada es un depósito (número positivo) o un retiro (número negativo). El saldo de cuenta inicial es ingresado por el usuario:

Listado del Programa:

```
10 INPUT "SALDO INICIAL?", B
20 INPUT "SUMA A TRAMITAR?", TA
25 IF TA = 0 THEN GOTO 80
30 B = B + TA
40 IF TA < 0 THEN GOTO 70
50 PRINT "DEPOSITO DE $"; TA
60 GOTO 80
70 PRINT TA; " RETIRO DE $"
80 PRINT "SALDO FINAL =  "; B
90 END
```

Digitaciones:

[1] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [S] [A] [L] [D] [O]
[SPACE] [I] [N] [I] [C] [I] [A] [L]
[SHIFT] [?] [SHIFT] ["] [SHIFT] [,] [B] [ENTER]

[2] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [S] [U] [M] [A] [SPACE]
[A] [SPACE] [T] [R] [A] [M] [I] [T] [A] [R] [SHIFT] [?]
[SHIFT] ["] [SHIFT] [,] [T] [A] [ENTER]

[2] [5] [I] [F] [T] [A] [=] [Ø] [T] [H] [E] [N] [G] [O] [T] [O]
[8] [Ø] [ENTER]

[3] [Ø] [B] [=] [B] [+] [T] [A] [ENTER]

[4] [Ø] [I] [F] [T] [A] [SHIFT] [<] [Ø] [T] [H] [E] [N]
[G] [O] [T] [O] [7] [Ø] [ENTER]

[5] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [D] [E] [P] [O] [S]
[I] [T] [O] [SPACE] [D] [E] [SPACE] [SHIFT] [$]
[SHIFT] ["] [SHIFT] [:] [T] [A]
[ENTER]

[6] [Ø] [G] [O] [T] [O] [8] [Ø] [ENTER]

[7] [Ø] [P] [R] [I] [N] [T] [T] [A] [SHIFT] [:] [SHIFT] ["]
[SPACE] [R] [E] [T] [I] [R] [O] [SPACE] [D] [E] [SPACE] [$]
[SHIFT] ["] [ENTER]

[8] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [S] [A] [L] [D] [O]
[SPACE] [F] [I] [N] [A] [L] [=] [SPACE]
[SHIFT] ["] [SHIFT] [:] [B] [ENTER]

[9] [Ø] [E] [N] [D] [ENTER]

Observe que este programa termina solamente si la cantidad de la transacción es cero. La segunda sentencia IF es la única que representa la estructura que mencionamos antes. Observe que además de las dos acciones separadas, asociadas con la sentencia IF (líneas 80 y 90) que son ejecutadas independientemente del resultado de las pruebas de la sentencia IF.

Ahora podemos escribir la tercera versión del programa de la sección previa:

Listado del Programa:

```
10 PAUSE "ESTAS DORMIDO?"
20 INPUT "ESCRIBA SI O NO "; SXS
30 IF SXS<>"SI" THEN GOTO 99
40 PRINT "OH, LO SIENTO MUCHO"
99 END
```

Digitaciones:

```
[1] [Ø] [P] [A] [U] [S] [E] [SHIFT] ["] [E] [S] [T] [A] [S]
    [SPACE] [D] [O] [R] [M] [I] [D] [O] [SHIFT] [?]
    [SHIFT] ["] [ENTER]
[2] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [E] [S] [C] [R] [I] [B] [A]
    [SPACE] [S] [I] [SPACE] [O] [SPACE] [N] [O] [SPACE]
    [SHIFT] ["] [SHIFT] [;] [S] [X] [SHIFT] [$] [ENTER]
[3] [Ø] [I] [F] [S] [X] [SHIFT] [$] [SHIFT] [<] [SHIFT] [>]
    [SHIFT] ["] [S] [I] [SHIFT] ["] [T] [H] [E] [N]
    [G] [O] [T] [O] [9] [9] [ENTER]
[4] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [O] [H] [SHIFT] [,]
    [SPACE] [L] [O] [SPACE] [S] [I] [E] [N] [T] [O]
    [SPACE] [M] [U] [C] [H] [O] [SHIFT] ["]
    [ENTER]
[9] [9] [E] [N] [D] [ENTER]
```

Esta versión invierte la prueba condicional especificando alguna acción para ser ejecutada si la entrada no es igual a "SI". De esta forma, la computadora elimina el problema de la segunda versión y sentencia IF escrita más arriba. Disponiendo las sentencias de esta manera, es a menudo una maniobra útil cuando se programa. Haciendo algunos experimentos por su cuenta usted conseguirá obtener mejores programas.

Otro uso común de la sentencia GOTO es originar ejecuciones repetidas de una secuencia de sentencias. Este proceso se conoce como "looping" (circuito cerrado o formación de bucles). Un ejemplo sencillo de formación de bucles se representa en este programa:

Listado del Programa:

```
10 WAIT 30
20 PRINT "CORRE CORRE CORRE CORRE"
30 PRINT "    CORRE CORRE CORRE CORRE"
40 GOTO 20
```

Digitaciones:

[1] [Ø] [W] [A] [I] [T] [3] [Ø] [ENTER]

[2] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [C] [O] [R] [R] [E] [SPACE]
[C] [O] [R] [R] [E] [SPACE] [C] [O] [R] [R] [E] [SPACE] [C] [O]
[R] [R] [E] [SHIFT] ["] [ENTER]

[3] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [SPACE] [SPACE]
[SPACE] [C] [O] [R] [R] [E] [SPACE] [C] [O] [R] [R] [E]
[SPACE] [C] [O] [R] [R] [E] [SPACE] [C] [O] [R] [R] [E] [SHIFT] ["]
[ENTER]

[4] [Ø] [G] [O] [T] [O] [2] [Ø] [ENTER]

Desafortunadamente, este programa continuará "corriendo" por siempre. (Usted puede parar el programa utilizando la tecla BREAK). Debemos buscar una manera de que estos programs acaben. Podemos hacer esto utilizando un contador al mismo tiempo que la sentencia IF. Un contador es una variable en la cual guardaremos el total de las veces que hemos hecho algo (la usamos por ej., para contar). Con esta técnica, la sentencia IF probará si hemos ejecutado nuestra sentencia PRINT un número determinado de veces. Por supuesto, es nuestra decisión determinar el número de repeticiones. Vamos a escoger diez veces por cada sentencia PRINT (después de 10 veces uno se aburre . . .).

Utilizando nuestro contador y la sentencia IF podemos escribir:

Listado del Programa:

```
10 WAIT 30
15 C = 1
20 PRINT "CORRE CORRE CORRE CORRE"
30 PRINT "    CORRE CORRE CORRE CORRE"
40 C = C + 1
50 IF C<=10 THEN 20
60 END
```

Digitaciones:

```
10 Ø W A I T 30 ENTER
15 C = 1 ENTER
20 P R I N T SHIFT " C O R R E SPACE
     C O R R E SPACE C O R R E SPACE
     C O R R E SHIFT " ENTER
30 P R I N T SHIFT " SPACE SPACE
     SPACE C O R R E SPACE C O R R E
     SPACE C O R R E SPACE C O R R E SHIFT "
     ENTER
40 C = C + 1 ENTER
50 I F C SHIFT < = 10 T H E N
     20 ENTER
60 E N D ENTER
```

Siga la operación del contador al mismo tiempo que cada bucle es ejecutado. Observe que en la línea 15 debemos asignarle al contador un valor inicial. En la línea 40, incrementamos este valor para reflejar una ejecución más de las sentencias 20 y 30.

Hay muchas otras formas de utilizar contadores y bucles dentro de programas. Desafortunadamente, no tenemos el espacio para describirlos aquí. Le sugerimos que usted continúe estudiándolo en uno de los libros de la lista del Apéndice F.

La intrucción GOTO es un comando al mismo tiempo que una sentencia. Como comando su utilización es diferente de su uso como sentencia de programa. Emitido como comando, en el modo RUN, GOTO empieza la ejecución del programa en una forma similar a la orden RUN. La diferencia descansa en ciertas preparaciones internas que se hacen antes que las instrucciones del programa se ejecuten. (Para una comparación de los métodos utilizados para comenzar un programa, vea la sección titulada Empezando la Ejecución en el Capítulo V). A diferencia del comando RUN, el comando **GOTO no borra los valores de ninguna variable antes que empiece la ejecución del programa.**

Para empezar la ejecución del programa con el comando GOTO escriba:

GOTO número-línea

donde:

número-línea es el número de la primera línea del programa que va a ser ejecutado.

NOTA: Especificando un número-línea que no existe causará un ERROR 11.

P. FOR y NEXT

Como hemos visto en la sección previa, la habilidad de repetir una serie de instrucciones es muy útil. De hecho esta característica especial es utilizada tan a menudo, que el BASIC incorpora varias sentencias para mecanizar el proceso. Se trata de la sentencia FOR y de su compañera, la sentencia NEXT. Unidas, las sentencias FOR y NEXT contienen una serie de instrucciones que son repetidas

un número de veces dado. A la sentencia FOR se asocia un contador-variable y una condición de prueba construida en si misma. Esto también permite la especificación del valor-inicial y el valor de incremento de la variable-contadora.

La forma para toda esta información es:

FOR variable-contadora = valor-inicial TO valor-final STEP valor-incremento

donde:

variable contadora es el nombre de la variable utilizada para retener el valor del circuito.

valor-inicial es el valor almacenado en el contador-variable antes de pasar la primera vez a través del bucle. El margen permitido para este valor es de −32768 a 32767.

Valor-final es el número que se utiliza en la prueba. Si el contador-variable contiene un valor mayor que el valor-final, los bucles se acaban. El margen permitido para esté valor es de −32768 a 32767.

STEP valor-incremento es una cláusula opcional. El valor-incremento indica cuanto debe aumentar o disminuir el contador-variable cada vez que pase a través del bucle. El margen permitido para este valor es de −32768 a 32767. Si la orden de step es omitida, entonces se asume que el valor-incremento es igual a uno.

Esto es demasiado para asimilar, así que entonces vamos a observar el desarrollo de algunos ejemplos de programas sencillos. El primer programa es similar a la versión del programa CORRE CORRE que utilizó un contador. En vez de "correr", representamos el valor del contador-variable C:

```
15 FOR C = 1 TO 10
30 PAUSE C
50 NEXT C
```

(Para hacer este programa "CORRER" como antes, simplemente intercalar las sentencias 10, 20, y 30 de aquel programa). Observe que esta versión es más limpia y más concisa, utilizando menos sentencias para conseguir la misma cuenta y funciones de bucle que en la versión anterior.

En caso de que haya todavía alguna confusión acerca de lo que las sentencias FOR y NEXT hacen, presentamos una comparación de un bucle FOR . . . NEXT con las sentencias equivalentes:

```
10 FOR I = 1 TO 8
20 [ algunas sentencias para
:    ser repetidas ]
90 NEXT I
100 END
```

```
10 I = 1
12 IF I > 8 THEN 100
20 [ algunas sentencias para
     ser repetidas ]
90 I = I + 1
92 GOTO 12
100 END
```

Nuestro segundo, programa nos muestra como uno puede programar un "bucle general" cuyas repeticiones estén controladas por el valor de una variable. Empezamos preguntando al usuario cuantos números él piensa introducir. Nosotros almacenamos ese número en una variable N y continuamos dando vueltas a través de las sentencias que utilizan un número para procesar. El bucle se ejecuta N veces, a menos que N sea menor que, o igual a, cero, en cuyo caso acabamos (END) sin continuar procesando.

Listado del Programa:

```
10 N=0 : V=0 : T=0 : A=0
20 WAIT 0
30 INPUT "CUANTOS VALORES? "; N
40 IF N=0 THEN GOTO 999
50 FOR I=1 TO N
60 CLS : CURSOR 0
70 INPUT V
80 T=T+V
90 NEXT I
100 WAIT : CLS : CURSOR 0
110 A=T/N
120 PAUSE "TOTAL = "; T
130 PRINT "PROMEDIO = "; A
999 END
```

Digitaciones:

[1] [Ø] [N] [=] [Ø] [SHIFT] [:] [V] [=] [Ø] [SHIFT] [:] [T] [=] [Ø]
[SHIFT] [:] [A] [=] [Ø] [ENTER]

[2] [Ø] [W] [A] [I] [T] [Ø] [ENTER]

[3] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [C] [U] [A] [N] [T] [O] [S]
[SPACE] [V] [A] [L] [O] [R] [E] [S] [SHIFT] [?] [SPACE]
[SHIFT] ["] [SHIFT] [;] [N] [ENTER]

[4] [Ø] [I] [F] [N] [=] [Ø] [T] [H] [E] [N] [G] [O] [T] [O]
[9] [9] [9] [ENTER]

[5] [Ø] [F] [O] [R] [I] [=] [1] [T] [O] [N] [ENTER]

[6] [Ø] [C] [L] [S] [SHIFT] [:] [C] [U] [R] [S] [O] [R] [Ø] [ENTER]

[7] [Ø] [I] [N] [P] [U] [T] [V] [ENTER]

[8] [Ø] [T] [=] [T] [+] [V] [ENTER]

[9] [Ø] [N] [E] [X] [T] [I] [ENTER]

[1] [Ø] [Ø] [W] [A] [I] [T] [SHIFT] [:] [C] [L] [S] [SHIFT] [:]

C U R S O R Ø ENTER

1 1 Ø A = T / N ENTER

1 2 Ø P A U S E SHIFT " T O T A L

SPACE = SPACE SHIFT " SHIFT :

T ENTER

1 3 Ø P R I N T SHIFT " P R O M E D I

O SPACE = SPACE SHIFT " SHIFT :

A ENTER

9 9 9 E N D ENTER

El bucle FOR . . . NEXT no necesita aumentar siempre el contador-variable por 1 ni siempre darle un valor inicial de 1. Utilizando la cláusula STEP, el programador puede especificar el tamaño del incremento (o disminución). Nuestro próximo ejemplo demuestra esto y nos trae memorias de la escuela cuando gritabamos de júbilo en los partidos de fútbol:

Listado del Programa:

```
10 WAIT 30
20 FOR HS = 2 TO 8 STEP 2
30 PRINT HS; "! ";
40 NEXT HS
50 WAIT 60 : CLS : CURSOR 0
60 PRINT "ARRIBA CAMPEON"
70 PRINT "SHARP PC-1500!"
80 END
```

Digitaciones:

1 Ø W A I T 3 Ø ENTER

2 Ø F O R H S = 2 T O 8 S T E P

2 ENTER

3 Ø P R I N T H S SHIFT : SHIFT "

SHIFT ! SPACE SHIFT " SHIFT :

ENTER

4 Ø N E X T H S ENTER

5 Ø W A I T 6 Ø SHIFT : C L S SHIFT :

C U R S O R Ø ENTER

6 Ø P R I N T SHIFT " A R R I B A SPACE

C A M P E O N

SHIFT " ENTER

[7] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [S] [H] [A] [R] [P] [SPACE]
[P] [C] [−] [1] [5] [Ø] [Ø] [SHIFT] [!] [SHIFT] ["] [ENTER]
[8] [Ø] [E] [N] [D] [ENTER]

Además de contar hacia arriba, la cláusula STEP también permite contar hacia abajo. Esto se hace al invertir los valores iniciales y finales y al especificar un incremento negativo. El programa siguiente (dedicado a consumidores de todo el mundo), ilustra esto:

El almacén "Alfombras Imperiales" ofrece alfombras al increíble precio de $,99 por metro cuadrado. Las alfombras fluctúan en radio de 40 metros (para la entrada de un hotel) a 1 metro (para los baños). Entre cada alfombra y la próxima más pequeña hay una diferencia radial de 3 metros. El Señor Consumidor Astuto decide calcular el precio de cada una, utilizando su computadora SHARP. El escribe el programa siguiente . . . .

```
10 FOR R = 40 TO 1 STEP −3
20 P = .99 * (PI * R ^ 2)
30 PRINT R; " ␣ PIE ␣ ES ␣ $" ; P
40 NEXT R
50 END
```

y descubre que la alfombra fluctúa en precio de $4976,28 a $3,11.

Q. WAIT

La sentencia WAIT permite al programador cambiar la operación de la sentencia PRINT. La información presentada visualmente por una sentencia PRINT quedará en la representación visual durante el período de tiempo especificado por la sentencia WAIT.

El formato para la sentencia WAIT es:

WAIT argumento

El argumento es opcional. Si no hay ningún argumento especificado, el período de tiempo de ejecución incompleta es "infinito", esto es, la información quedará en la representación visual hasta que el usuario pulse [ENTER] . Este es el modo de operación utilizado en la mayoría de nuestros programs hasta este punto.

Si se da un argumento, todas las sentencias PRINT subsecuentes retendrán su información en la representación visual por un período de tiempo proporcional al número especificado como argumento. Este tipo de impresión es similar a la sentencia PAUSE, excepto que el período de tiempo de la sentencia PAUSE está fijado. Observe que la sentencia WAIT no tiene ningún efecto en la operación de la sentencia PAUSE.

El número (o la expresión que resulta en un número) dado como argumento debe fluctuar del 0 al 65535. WAIT 0 hace que la información sea representada tan rápidamente que es casi imposible leerla. ¡WAIT 65535 hará que cada sentencia PRINT presente visualmente su información alrededor de 17 minutos! En la práctica, WAIT 64 da un período de un segundo, y WAIT 3840 un período de un minuto. Para restaurar de nuevo la operación de la sentencia PRINT de forma que espere hasta que el usuario pulse ENTER, utilice la instrucción WAIT sin ningún argument.

Programa de Demostración

Este programa ilustra el efecto de la sentencia WAIT en las subsecuentes frases PRINT. Aqui variamos el tiempo WAIT (de espera) de 0 a 102 en incrementos de 2 mientras representamos puntos en un bucle. La sentencia BEEP (campanilla) es unicamente para ayudarle a comprender el intervalo de tiempo, dado que parte de la acción pasa demasiado rápida para ser vista.

Listado del Programa:

```
10 FOR W = 0 TO 102 STEP 2
20 WAIT W
30 BEEP 1,5
40 PRINT " . ";
50 NEXT W
60 END
```

Digitaciones:

1 Ø F O R W = Ø T O 1 Ø 2 S T E P
2 ENTER
2 Ø W A I T W ENTER
3 Ø B E E P 1 SHIFT + 5 ENTER
4 Ø P R I N T SHIFT " . SHIFT "
SHIFT ; ENTER
5 Ø N E X T W ENTER
6 Ø E N D ENTER

R READ DATA Y RESTORE

No todos los datos de un programa tienen que ser ingresados por la persona que utiliza el programa. A menudo, los datos útiles son relativamente estáticos como listas de impuestos dentro de una aplicación financiera o una tensión contante en una aplicación de ingeniería. Este tipo de información puede formar parte de un programa, y es utilizada cuando sea necesaria, a través del uso de las sentencias DATA, READ, y RESTORE. Estas sentencias actúan de acuerdo para especificar los datos utilizados por un programa, para transferir los datos dentro de las variables, y para repetir el proceso tantas veces como fuera necesario.

La sentencia DATA consiste de la palabra-clave DATA seguida por una lista de datos. Estos incluyen números, en notación científica o real, y series de caracteres. Los datos en la lista están separados por comas. Sentencias con datos pueden aparecer en cualquier parte dentro de un programa, pero muchos programadores prefieren agruparlos al comienzo del programa. Esto permite que se encuentren más fácilmente cuando el programa se lee.

Una sentencia DATA típica puede parecerse a la siguiente:

```
10 DATA "MOBY DICK", 20000, "BLANCA", "M", 112
```

La sentencia READ consiste de la palabra clave READ seguida por una lista de nombres de variables. Estas pueden ser variables numéricas o de caracteres. Las variables dentro de la lista están separadas por comas. La sentencia READ causa que uno o más items de datos sean "leídos" de una sentencia DATA y almacenados en las variables asociadas. Una sentencia READ que corresponde a nuestra previa sentencia DATA es:

```
120 READ N$, WT, CS, SX$, L
```

La PC-1500 insiste en que cada vez que se ejecuta, una sentencia READ, existe una lista de datos correspondientes dentro de una sentencia DATA. Así, el programa siguiente producirá un error en la línea 30 porque todos los items de datos han sido utilizados por la sentencia READ en la línea 20:

```
10 DATA 1, 2, 3
20 READ A, B, C
30 READ D
```

Para corregir esta situación nosotros podemos añadir un item de datos a la línea 10:

```
10 DATA 1, 2, 3, 65
```

o podemos utilizar otra sentencia separada DATA en cualquier parte del programa:

```
10 DATA 1, 2, 3
20 READ A, B, C
30 READ D
40 DATA 65
```

Esto demuestra que la PC-1500 ve todas las sentencias DATA dentro de un programa como una lista única de datos. A medida que la computadora encuentra cada una de las variables dentro de una sentencia READ, asigna el próximo dato de la lista a esa variable. Si la PC-1500 no puede ejecutar la petición de un dato, la computadora parará el programa y señalará un error. Los datos extras, que no son utilizados cuando el programa acaba de una manera normal, son ignorados.

Si el tipo (carácter o número) del próximo item no corresponde al tipo de la variable que tiene que llenarse, un error ocurrirá. Los buenos programadores agrupan los datos en sentencias DATA separadas, cada una de las cuales corresponde a su sentencia READ dentro del programa. Esto se muestra en el programa siguiente que lee tres datos cuatro veces:

```
10 DATA 1, "A", 1
20 DATA 2, "B", 3
30 DATA 5, "C", 8
40 DATA 13, "D", 21
50 FOR I = 1 TO 3
60 READ A, A$, Z
70 T = T + A * Z
80 NEXT I
```

Las líneas del 10 al 40 podrían haber sido escritos así:

```
10 DATA 1, "A", 1, 2, "B", 3, 5, "C", 8, 13, "D", 21
```

o incluso así:

```
10  DATA 1
20  DATA "A"
30  DATA 1
40  DATA 2
  :
(etc.)
```

Ambas formas alternadas obscurecen el hecho de que tres datos son leídos cada vez por la sentencia READ. Las formas alternas también hacen más difícil verificar los tipos de datos ya que la estructura de número, carácter, número no se puede distinguir inmediatamente.

A veces se desea re-utilizar algunos o todos los valores en las sentencias DATA. La sentencia RESTORE (devolver) nos permite hacer esto. RESTORE causa que la SHARP re-utilice todos los datos comenzando con la primera sentencia DATA del programa. Así, cualquier sentencia READ ejecutada después de la sentencia RESTORE recibirá datos que fueron utilizados previamente.

La sentencia RESTORE también puede especificar un "reciclamiento" más selectivo de los datos. La sentencia:

RESTORE línea-número

causará que los datos sean re-asignados al principio de las sentencias DATA en la línea especificada. Por ejemplo:

```
10  DATA .8, 4
15  DATA 1, 3, 6, 8E2
100  READ X, Y
110  READ M, N, O, P
120  RESTORE 15
130  READ A, B
```

Cuando este programa termine, los valores de las variables A y B serán 1 y 3, respectivamente.

Además de las líneas numeradas, las sentencias DATA pueden ser etiquetadas (rotuladas) con un carácter único. La sentencia RESTORE puede entonces ser utilizada para re-emitir datos al comienzo de la sentencia DATA con la etiqueta dada. Un ejemplo de una sentencia etiquetada es:

```
20 "A" : DATA 1, −12.2, 8
```

El siguiente segmento del programa restaura la sentencia DATA etiquetada "X":

```
10  DATA 4.2, 3, 1
20  "X" : DATA -2, 0, 3, 5
100 READ Q, Y, Z
110 READ MA, MB, MC, MD
120 RESTORE "X"
130 READ N, Z
```

Al final del programa, N contiene −2 y Z contiene 0.

S. REM (REMarque)

La sentencia REM tiene la capacidad de insertar comentarios entre las sentencias de un programa. Aunque estos comentarios son ignorados por la SHARP, ellos son extremamente importantes porque asisten a otros seres humanos a leer su programa. Cuanto más fácil su programa sea leído y comprendido, tanto más lo querrán utilizar otros programadores, y quizás mejorarlo.

Para aliviar la carga de escribir a máquina, hemos omitido los comentarios en este manual. Nosotros le recomendamos que NO siganuestro ejemplo. Los comentarios son muy importantes cuando usted está imprimiendo, almacenando, o compartiendo sus programas. No dependa de su agilidad mental para recordar cual es el significado de cada variable en un programa. ¡Utilice una nota! Esto le ayudará a recordar su significado seis meses más tarde.

Las notas que sigan a la palabra clave REM pueden insertarse en su propia línea o al final de otra sentencia o serie de sentencias. Las notas no pueden aparecer antes o en el medio de sentencias ejecutables. La razón de ésto es que cuando la computadora SHARP ve la palabra clave REM, ignora cualquier carácter que se encuentre en el resto de la línea. Si REM fuera utilizado al principio de una línea, las sentencias válidas del programa serían ignoradas.

T. GOSUB. . . RETURN

Cuando usted empiece a diseñar programas, encontrará que puede utilizar ciertas funciones repetidamente dentro de un programa único. Por ejemplo, algunas de estas funciones pueden incluir: calcular el área de un círculo o aceptar y comprobar un número dado por el usuario. Cada repetición causa duplicación de las sentencias que ejecutan la función. Para evitar esta duplicación, el programador puede utilizar la sentencia GOSUB.

La sentencia GOSUB permite que un grupo de otras sentencias, que son utilizadas en varios lugares dentro del programa, puedan separarse. Este grupo de sentencias se llama "subrutina" (de aquí viene el término GOSUB) (vaya a subrutina). En cada punto del programa donde el grupo puede utilizarse, una instrucción GOSUB es insertada.

La sentencia GOSUB instruye a la SHARP para que empiece a ejecutar el grupo de sentencias que han sido separadas. Este proceso se conoce como "llamado a una subrutina". Debido a que la sentencia GOSUB causa un cambio en el flujo de ejecución de la secuencia normal, es similar a la sentencia GOTO. Sin embargo, la diferencia está en que antes que SHARP empiece a ejecutar las sentencias de la subrutina, la computadora se acuerda donde estaba. Cuando la computadora termina de ejecutar la subrutina, vuelve al punto desde donde fue llamada. Esto se conoce como "volver de una subrutina."

¿Pero, cómo discierne la PC-1500 el final de las sentencias que forman la subrutina? La respuesta es que usted debe informarla con la sentencia RETURN. La forma de la sentencia GOSUB es:

GOSUB número de línea

donde el número de línea es el correspondiente a la primera línea de la subrutina. La forma de la sentencia RETURN es simplemente:

RETURN

Como ejemplo de una subrutina en acción, considere un programa que computa y compara el área de dos rectángulos dado el largo y el ancho de los lados:

Listado del Programa:

```
10  REM LEER LARGO Y ANCHO
20  REM DE DOS RECTANGULOS
30  FOR I = 1 TO 2
40  PAUSE "RECTANGULO "; I; " :"
50  INPUT "ENTRE LARGO, ANCHO", L, W
60  GOSUB 200
70  IF I = 1 LET A1 = A
80  IF I = 2 LET A2 = A
90  NEXT I
100 REM IMPRIMA Y COMPARE LAS
110 REM AREAS DE RECTANGULOS.
120 PRINT "AREA DE RECTANGULO 1 "; A1
130 PRINT "AREA DE RECTANGULO 2 "; A2
140 IF A1 > A2 THEN 170
150 PRINT "AREA DE 2 ES > AREA 1"
160 GOTO 180
170 PRINT "AREA DE 1 ES > AREA 2"
180 END
200 REM SUBRUTINA COMPUTAR EL AREA
210 REM DEL RECTANGULO DANDO LA
205 REM LONGITUD DE LOS LADOS
220 REM EN L Y W
230 A = L * W
240 RETURN
```

Observe donde está colocada la subrutina. Todas las subrutinas deben estar colocadas al final de la sentencia END del principal programa. Esto previene su uso accidental en el proceso normal de ejecución por secuencia. Una sentencia RETURN debe terminar cada una de las subrutinas.

Una subrutina puede incluir cualquier sentencia legal permitida y puede ejecutar cualquier clase de proceso deseado. Los buenos programadores del BASIC diseñan sus programas como un conjunto de piezas o "módulos". Generalmente una subrutina es utilizada para codificar cada módulo. El programa principal es entonces utilizado para controlar el orden en que las subrutinas son ejecutadas. Para más información al respecto, refiérase a uno de los libros de programación estructurada registrados en el apréndice F.

Resumen de características del modo edición de programas

1) Las líneas del programa deben ser numeradas a intervalos de, por lo menos, diez. Esto permitirá que una línea adicional sea insertada entre cualquiera de las ya existentes, simplemente escogiendo un número de línea que esté entre las que se desea insertar. Por ejemplo, para insertar una línea entre las líneas 40 y 50, dé a la nueva línea un número del conjunto de números entre 41 y 49.

2) Para suprimir una línea existente, escriba su número y pulse [ENTER]

   NOTA: Debido a que varias sentencias del programa pueden agruparse juntas en una línea, hay que tener mucho cuidado cuando éstas se suprimen.

3) Las teclas con Flechas Arriba y Abajo pueden ser utilizadas para mover a tráves del programa una línea a la vez. Reteniendo cualquiera de las teclas con flecha, se causará una repetición automática del movimiento.

4) El comando LIST puede ser utilizado para ir directamente a la línea deseada. La orden LIST presentará visualmente la primera línea del primer programa en la memoria. LIST N (donde la N es un número) presentará visualmente la línea N o, si no existe una línea N numerada, la primera línea cuyo número es mayor que N se presentará.

5) Una vez que una línea se representa visualmente, las teclas con Flechas Derecha o Izquierda pueden utilizarse para colocar el cursor en cualquier parte de la misma. Las funciones INSert (insertar) y DELete (suprimir) pueden entonces utilizarse para efectuar cambios.

   NOTA: Si se hacen cambios a una línea, es necesario pulsar ENTER antes de que la próxima línea se represente o todos los cambios se invalidarán.

6) Si se encuentra un error la línea en un programa durante su ejecución, la tecla con Flecha Arriba "recordará" la línea en que ocurrió el error. Para volver a llamar a esa línea, cambie el modo PROgrama y pulse la Flecha Arriba.

Nota: El área de datos del programa se utiliza conjuntamente para programa y datos. Cuando la suma del área del programa y el área de los datos exceda la capacidad del área de datos del programa entero, pulse **65279 END** al programar.

IV. CALCULOS AVANZADOS

A. Notación Científica

Para ingresar un número en notación científica (A x 10 ^ B), entre la mantisa, pulse la letra E y entre el exponente.

Ejemplo 1: Para ingresar 6, 7 x 10 ^ 8:

Digitaciones	Representación Visual
[6] [.] [7]	6. 7_
[E]	6. 7E_
[8]	6. 7E8_

Ejemplo 2: Para ingresar: −9,12 x 10 ^ −34:

Digitaciones	Representación Visual
[−] [9] [.] [1] [2]	−9. 12_
[E]	−9. 12E_
[−] [3] [4]	−9. 12E−34_

Sólo los 10 primeros dígitos de la operación son significativos (vea el ejemplo). Para un número menor que 1 pero mayor que −1 el dato es exacto hasta un máximo de 10 dígitos.

Ejemplo 3: Para ingresar 1234567898765:

Digitaciones	Representación Visual
[1] [2] [3] [4] [5] [6] [7] [8]	
[9] [8] [7] [6] [5]	1234567898765_
[ENTER]	1. 234567898E 12

Ejemplo 4: Para ingresar 9.87654321234:

Digitaciones	Representación Visual
[9] [.] [8] [7] [6] [5] [4]	
[3] [2] [1] [2] [3] [4]	9. 87654321234_
[ENTER]	9. 876543212

Ejemplo 5: Para ingresar 0.0000000002345678:

Digitaciones	Representación Visual
[.] [0] [0] [0] [0] [0]	
[0] [0] [0] [0] [2] [3]	
[4] [5] [6] [7] [8]	.000000002345678_
[ENTER]	2.345678E-10

Ejemplo 6: Para ingresar 0.00001234567 x 10 ^ 24:

Digitaciones	Representación Visual
[.] [0] [0] [0] [0] [1] [2]	
[3] [4] [5] [6] [7] [E] [2] [4]	.00001234567E24_
[ENTER]	1.234567E 19

Observe que para exponentes, sólo los últimos dos dígitos escritos son efectivos.

Ejemplo 7: Para ingresar 3 x 10 ^ 123:

Digitaciones	Representación Visual
[3] [E] [1] [2] [3]	3E123_
[ENTER]	3E 23

Ejemplo 8: Para ingresar 4 x 10 ^ −3210:

Digitaciones	Representación Visual
[4] [E] [−] [3] [2] [1] [0]	4E-3210_
[ENTER]	4E-10

B. El Rango de los Cálculos

La mayoría de las máquinas tienen un rango de números con el cual pueden operar. En la PC-1500, este rango es cualquier número entre 9.999999999 x 10 ^ 99 y −9.999999999 x 10 ^ 99. Cuando un número excede este rango, es demasiado largo para que sea manejado por la computadora y una condición de "desborde" (señalada por el ERROR 37) ocurre. También

existe la condición de "subdesborde"; aquí un número se hace muy pequeño para manejarlo. La condición de subdesborde no se señala; ningún mensaje de error o interrupción ocurrirá. Cualquier número que esté dentro del margen de -1×10^{-99} a 1×10^{-99} será considerado como cero. Esta explicación se ilustra por la siguiente información gráfica:

		-1×10^{-99}	1×10^{-99}	
$-9.999999999 \times 10^{99}$		0	$9.999999999 \times 10^{99}$	
ERROR	RANGO COMPUTACIONAL	CONSIDE-RADO COMO CERO	RANGO COMPUTACIONAL	ERROR

Ejemplo 1: Si usted intenta resolver la exuación $(5.67 \times 10^{55}) * (8.90 \times 10^{65})$, causará un desborde:

Digitaciones	Representación Visual
(5 . 6 7 E 5 5) *	
(8 . 9 0 E 6 5)	(5. 67E55) * (8. 90E65)_
ENTER	ERROR 37

Error 37 indica que hay un desborde en el cálculo.

C. Raíz, Potencia, Pi

Raíz

Ejemplo 1: Para calcular la raíz cuadrada de 73:

Digitaciones	Representación Visual
SHIFT √ 7 3	√73_
ENTER	8. 544003745

Ejemplo 2: Para calcular $\sqrt[4]{256}$:

Digitaciones	Representación Visual
SHIFT √ SHIFT √ 2 5 6	√√256_
ENTER	4

Ejemplo 3: Para calcular $\sqrt{3 \wedge 2 + 4 \wedge 2}$:

Digitaciones	Representación Visual
[SHIFT] [√] [(] [3] [*] [3] [+]	
[4] [*] [4] [)]	√ (3*3+4*4)_
[ENTER]	5

Esta ecuación también puede ser computada de la manera siguiente:

Ejemplo 4:

Digitaciones	Representación Visual
[SHIFT] [√] [(] [3] [SHIFT] [^] [2] [+] [4] [SHIFT] [^] [2] [)]	√ (3^2+4^2)_
[ENTER]	5

Potencia

La potencia, o función exponencial, le permite aumentar un número a una potencia.

Ejemplo 1: Para calcular 4 ^ 3 (= 4 x 4 x 4):

Digitaciones	Representación Visual
[4] [SHIFT] [^] [3]	4^3_
[ENTER]	64

Ejemplo 2: Para calcular 3 ^ 3.2 x 4 ^ – 2.4:

Digitaciones	Representación Visual
[3] [SHIFT] [^] [3] [.] [2] [*]	
[4] [SHIFT] [^] [-] [2] [.] [4]	3^3. 2*4^-2. 4_
[ENTER]	1. 207380162

Ejemplo 3: Para calcular 4 ^ (3 ^ 2):

Digitaciones	Representación Visual
[4] [SHIFT] [^] [3] [SHIFT]	
[^] [2]	4^3^2_
[ENTER]	262144

PI [π]

El valor de PI (3.141592654) es almacenado en 2 lugares: el símbolo [π] y PI como una constante fija. Cualquiera de los símbolos puede ser utilizado en el cálculo cuando se necesite el valor de PI.

Como ejemplo, para encontrar el área de una alfombra que tiene un diámetro de cinco metros, en el modo RUN, ingrese:

Digitaciones	Representación Visual
[P] [I] [*] [(] [5] [/] [2] [)]	
[SHIFT] [^] [2]	PI* (5/2) ^2_
[ENTER]	19. 63495408

D. Modos Angulares

La PC-1500 permite calcular funciones angulares en cualquiera de los tres modos angulares siguientes:

Para fijar la PC-1500 en el modo Grados sexagesimales:

DEG. [ENTER]

(DEG aparecerá en la parte superior de la representación visual)

Para fijar la PC-1500 en el modo Radianes:

RAD. [ENTER]

(RAD aparecerá en la parte superior de la representación visual)

Para fijar la PC-1500 en el modo Grados centesimales:

GRA. [ENTER]

(GRAD aparecerá en la parte superior de la representación visual)

E. Funciones Trigonométricas

Las seis funciones trigonométricas provistas en la PC-1500 son SIN, COS, TAN, ASN, ACS, y ATN. Cada función puede calcularse en cualquiera de los modos GRAD, DEG, o RAD. La ejecución es como sigue:

DEG. [ENTER]		Fija el modo Grados.
SIN 30 [ENTER]	0. 5	SIN 30 en grados.
[CL]		Borra la representación visual.
GRA. [ENTER]		Fija el modo Grados centesimales.
SIN 30 [ENTER]	4. 539904997E-01 →	SIN 30 en Grados centesimales.

[CL]	⟶	Borra la representación visual.
RAD. [ENTER]	⟶	Fija el modo Radianes.
SIN 30 [ENTER]	−9. 880316241E−01 ⟶	SIN 30 en radianes.

En los ejemplos arriba mencionados, el seno de 30 se computa en cada modo produciendo tres respuestas diferentes (pero equivalentes). Las funciones trigonométricas inversas pueden también ejecutarse como sigue:

RAD. [ENTER]	⟶	Fija el modo.
ASN −0.5 [ENTER]	−5. 235987756E−01	Arcseno de −.5
DEG. [ENTER]	⟶	Fija el modo.
ASN −0.5 [ENTER]	−30	Arcseno de −.5
ACS (−.5 + .1) [ENTER]	113. 5781785	Arccoseno
ATN 2.3 [ENTER]	66. 50143432	Arctangente

En el modo de radianes o gradientes:
Si es necesario computar: SIN (X/Y), COS (X/Y), TAN (X/Y)
en donde X = cualquier número que no sea cero
e Y = cualquier número que no sea cero
Igualar la fracción a una variable, por ejemplo, P = X/Y
Quedando: SIN P, COS P, TAN P

F. Funciones Logarítmicas

LN, LOG

La función LN computará el logaritmo natural (base e) mientras que la función LOG computará el logaritmo común (base 10). Estos se ejecutan en el modo RUN como sigue:

LN 7.4 [ENTER]	2. 00148
LOG 7.4 [ENTER]	8. 692317197E−01
LN 25 [ENTER]	3. 218875825
LOG 100 [ENTER]	2

EXP

La función inversa de LOG es un número elevado a la potencia de 10. Por ejemplo:

LOG 100 [ENTER]	2
10 [SHIFT] [^] 2 [ENTER]	100

Debido a quel el logaritmo natural (LN) no está basado en una potencia de 10 sino en una potencia de e, una función inversa es necesaria. Esta función es EXP:

Ejemplo:

LN 7.4 [ENTER]	2. 00148
EXP 2.00148 [ENTER]	7. 399999998

G. Conversión de Angulos

La PC-1500 ejecuta conversiones de ángulos de DMS (grados, minutos, segundos) a la forma DEG (grado decimal). Cuando se convierten grados decimales a grados, minutos, y segundos equivalentes, la respuesta está comprendida en una porción de número entero representando grados, y una porción fraccional de la cual el primero y el segundo números decimales representan los minutos, y el tercero y cuarto números decimales los segundos. Del quinto hasta el final son grados decimales. Para convertir un ángulo dado en grados, minutos, y segundos a la forma de grado decimal, se debe ingresar el orden de un número decimal entero.

Ejemplo 1: Para convertir 16.1932 grados decimales en forma DMS:

DMS 16.1932 [ENTER]	16. 113552

Ejemplo 2: Para convertir 32.2513 DMS en forma de grados decimales:

DEG 32.2513 [ENTER]	32. 42027778

H. Funciones Varias

ABS

La función ABS deduce el valor absoluto de un valor numérico o una variable.

Ejemplo 1:

ABS (25 − 86) [ENTER]	61

Normalmente, 25 − 86 = −61. La función ABS toma la diferencia de los números para conseguir 61.

INT

La función INT redondea un valor numérico al mayor número entero, no mayor que el valor numérico mismo.

Ejemplo 1:

(25/3) + 7 [ENTER]	15. 33333333
INT (25/3) + 7 [ENTER]	15

Ejemplo 2:

(31.62 + 21.18) [ENTER]	52. 8
INT (31.62 + 21.18) [ENTER]	52

En el Ejemplo 2, la respuesta no está redondeada de 52.8 a 53 porque esto causaría que la respuesta fuera mayor que el valor original.

NOTA: No olvide el orden de evaluación. Los paréntesis deben ser utilizados si usted desea el INT de la expresión resultante. Por ejemplo, vamos a alterar el ejemplo previo dejando aparte los paréntesis:

Ejemplo 3:

INT 31.62 + 21.18 [ENTER]	52. 18

La computadora toma el INT del primer número 31.62 (resultado: 31) y añade 21.18 para conseguir 52.18. Esto es un poco diferente de la primera computación, ¿no?.

SGN

Para cualquier número X la función SGN devuelve un valor indicando si el número es negativo, cero, o positivo. Los valores son los siguientes:

+1	si	$X > 0$
0	si	$X = 0$
−0	si	$X < 0$

Ejemplos:

5 − 10 [ENTER]	−5
SGN (5 − 10) [ENTER]	−1
12 − 4 [ENTER]	8
SGN (12 − 4) [ENTER]	1
15 − 15 [ENTER]	0
SGN (15 − 15) [ENTER]	0

V. PROGRAMACION AVANZADA

A. Los ARRAYS y la Sentencia DIM

La mayoría de nuestros ejemplos de programas han utilizado hasta ahora un pequeño número de variables. Cuando usted empiece a utilizar el potencial completo de procesamiento de la PC-1500, descubrirá que las variables que retienen un número único pueden tener desventajas. Quizás, por ejemplo, usted está considerando un programa que lea cincuenta números y los clasifique. Usted puede concluir rápidamente que, aunque la PC-1500 tiene más que suficiente posibles variables, debe haber una forma más fácil. Pues la hay, y se llama una variable "array".

Un array es simplemente un grupo de áreas de almacenaje consecutivas o "localidades" con un nombre único. Cada área de almacenamiento puede tener un nombre único o cada área de almacenamiento puede tener una serie de caracteres. Todas las áreas de almacenamiento dentro de un array dado deben retener el mismo tipo de datos.

El número de localizaciones dentro de un array único puede ser hasta de 256 y es determinado por su especificación. Así, si usted define un array numérico con 50 localizaciones, ésto le permite almacenar hasta 50 números asociados con un nombre único. Si usted define un array de una serie de caracteres (llamado un array alfanumériс), usted también puede especificar el tamaño de las series, hasta un máximo de 80 caracteres por serie. La forma de crear series de caracteres de varios tamaños es descrita en la sección B-1.

Para definir un array, se utiliza la sentencia DIM (abreviatura de DIMensión). Los arrays DEBEN SER SIEMPRE "declarados" (definidos) antes de ser utilizados. (No como las variables de valores únicos que hemos estado utilizando.) La forma de la sentencia DIM para declarar un array con valores numéricos es:

DIM nombre-variable-numérico (tamaño)

donde:

nombre-variable-numérico es un nombre de variable que se conforma a las reglas normales de nombres de variables numéricas discutidos previamente.

tamaño es el número de localizaciones de almacenamiento y debe ser un número que fluctúe entre 0 y 255. Observe que cuando usted especifica un número para el tamaño, recibe una localización más de la que usted especificó.

Ejemplos de sentencias legales de DIMensión numérica son:

```
DIM X (5)
DIM AA (24)
DIM Q5 (0)
```

La primera sentencia crea un array con 6 localizaciones de almacenamiento. La segunda sentencia crea un array AA con 25 localizaciones. La tercera sentencia crea un array con una localización y es bastante simple pues (para los números por lo menos) es lo mismo que declarar una variable numérica de valor-único.

Es importante saber que un array-variable X y una variable X están separadas y son distintas para la SHARP. La primera denota una serie de localizaciones de almacenamiento numérica, y la segunda una localización única y diferente.

Ahora que usted sabe cómo crear arrays, puede que se esté preguntado cómo es que nosotros nos referimos a cada localización del almacenamiento. Como el grupo entero tiene un solo nombre, la forma en que nos referimos a una localización única ((llamada un "elemento") es seguir el nombre del grupo con un número en paréntesis. Este número se llama un "suscripto". Así por ejemplo, para almacenar el número 8 en el quinto elemento de nuestra array X (declarada previamente) nosotros escribiremos:

```
X (4) = 8
```

Si la utilización del 4 es enigmática, recuerde que la numeración de elementos empieza en cero y continúa a través del tamaño de los números declarados en la sentencia DIM.

El verdadero poder de los arrays reside en la habilidad de utilizar una expresión o un nombre variable como un suscripto. Por ejemplo, para crear une tabla conteniendo los cuadrados de los números del 0 al 9, podemos escribir las sentencias siguientes:

```
10 DIM SW (9)
20 FOR I = 0 TO 9
30 SW (I) = I * I
40 NEXT I
```

En este ejemplo, la variable I es utilizada para seleccionar qué localización de almacenamiento retendrá el resultado y es también utilizada para computar el resultado.

Para declarar un array de caracteres, se utiliza una forma ligeramente diferente de la sentencia DIM:

DIM nombre-carácter-variable (tamaño) * longitud

donde:

nombre-carácter-variable es un nombre de variable que se conforma a las reglas para caracteres variables normales como discutimos previamente.

Tamaño es el número de localizaciones de almacenamiento y debe estar en el margen del 0 al 255. Observe que cuando usted especifica un número, recibe una localizacion más de la que usted especificó.

Longitud es opcional. Cuando se utiliza, especifica la longitud de cada una de la series de caracteres que comprende el array. Longitud es un número que fluctúa de 1 al 80. Si esta cláusula no es utilizada, las series tendrán la longitud de 16 caracteres (es la longitud normal de variables únicas).

Ejemplos de declaraciones de arrays de caracteres legales son:

```
DIM X$ (4)
DIM NM$ (10) * 10
DIM IN$ (1) * 80
DIM R$ (0) * 26
```

El primer ejemplo crea un array de cinco series cada uno para almacenar 16 caracteres. La segunda sentencia DIM declara un array NM, con once series de 10 caracteres cada una. La definición explícita de series más pequeñas que la serie "default" (ejecución incompleta) ayuda a conservar espacio en la memoria. El tercer ejemplo declara un array de dos elementos de una serie de 80 caracteres y el último ejemplo declara una serie única de 26 caracteres (vea sección B-1.).

Además de los arrays simples que hemos estudiado, la PC-1500 permite arrays "bidimensionales". Por analogía, un array de una dimensión es una lista de datos arreglados en una columna única. Un array de dos dimensiones es una tabla de datos con filas y columnas. El array bidimensional está declarado por la sentencia:

DIM nombre-variable-numérico (filas, columnas)

o

DIM nombre-caracter-variable (filas, columnas) * longitud

donde:

filas especifica el número de filas en el array. Este debe ser un número que fluctúe del 0 al 255. Observe que cuando usted especifica el número de filas, consigue una fila más que las espedificadas.

columnas especifica el número de columnas en el array. Este debe ser un número que fluctúe del 0 al 255. Observe que cuando usted especifica el número de columnas, consigue una columna más que las especificadas.

El siguiente diagrama ilustra las localizaciones de almacenaje que resultan de la declaración DIM T (2, 3) y los suscriptos (ahora compuestos de dos números) que tienen que ver con cada localización de almacenamiento:

	columna 1	columna 2	columna 3	columna 4
fila 1	T (0, 0)	T (0, 1)	T (0, 2)	T (0, 3)
fila 2	T (1, 0)	T (1, 1)	T (1, 2)	T (1, 3)
fila 3	T (2, 0)	T (2, 1)	T (2, 2)	T (2, 3)

NOTA: Los arrays bidimensionales pueden llenar rápidamente el espacio de almacenamiento. Por ejemplo, lun array de longitud 1 con 25 filas y 35 columnas utiliza 875 localizaciones de almecenamiento!

Los arrays son instrumentos de programación muy poderosos. Para un tratamiento más completo de arrays, le recomendamos unas lecturas suplementarias.

B. Más Información Variables de Caracteres

B-1. Series DIMensionales

Las series alfanuméricas son limitadas, por omision del mandato, a dieciseis caracteres de longitud. Al dimensionar una serie de caracteres, es posible crear una serie cuya longitud sea de hasta 80 caracteres. Las reducciones en la longitud de la serie, para conservar espacio de la

memoria, son también posibles.

La longitud de una serie se especifica en la sentencia DIMensión como sigue:

DIM nombre-variable (limitado) * longitud

donde:

nombre-variable es el nombre del ordenamiento de caracteres.

limitado es el suscripto máximo del array.

longitud es la longitud de cada serie dentro del array (ordenamiento).

Si solamente es necesaria una serie, un array con un elemento (suscripto como elemento cero) puede ser especificado para conservar espacio en la memoria. Esto se ilustra por la declaración siguiente de una serie de 26 caracteres:

```
DIM A$ (0) * 26
```

B-2. Concatenación

Varias series de caracteres (o caracteres dentro de variables alfanuméricas) pueden unirse para formar una serie única. Esta aumento de series de caracteres se llama "concatenación". La forma de concatenación es:

$$\text{variable} = \frac{\text{serie-carácter}}{\text{carácter-variable}} + \frac{\text{serie-carácter}}{\text{carácter-variable}}$$

Ejemplo 1:

```
10 S$ = "SUPER"
20 T$ = S$ + "MAN"
30 PRINT T$
```

Salida:

SUPERMAN

En la línea 20, el contenido de la variable S$ (la serie "SUPER") se añade a la serie "MAN". Observe que ningún espacio se ha insertado durante la concatenación. Varias series pueden ser concatenadas en la misma expresión como en el siguiente ejemplo:

Ejemplo 2:

```
10 A$ = "LA"
20 B$ = " A"
30 C$ = "BUE"
40 D$ = " SU"
50 PRINT "ESCRIBA A" + D$ + B$ + C$ + A$
```

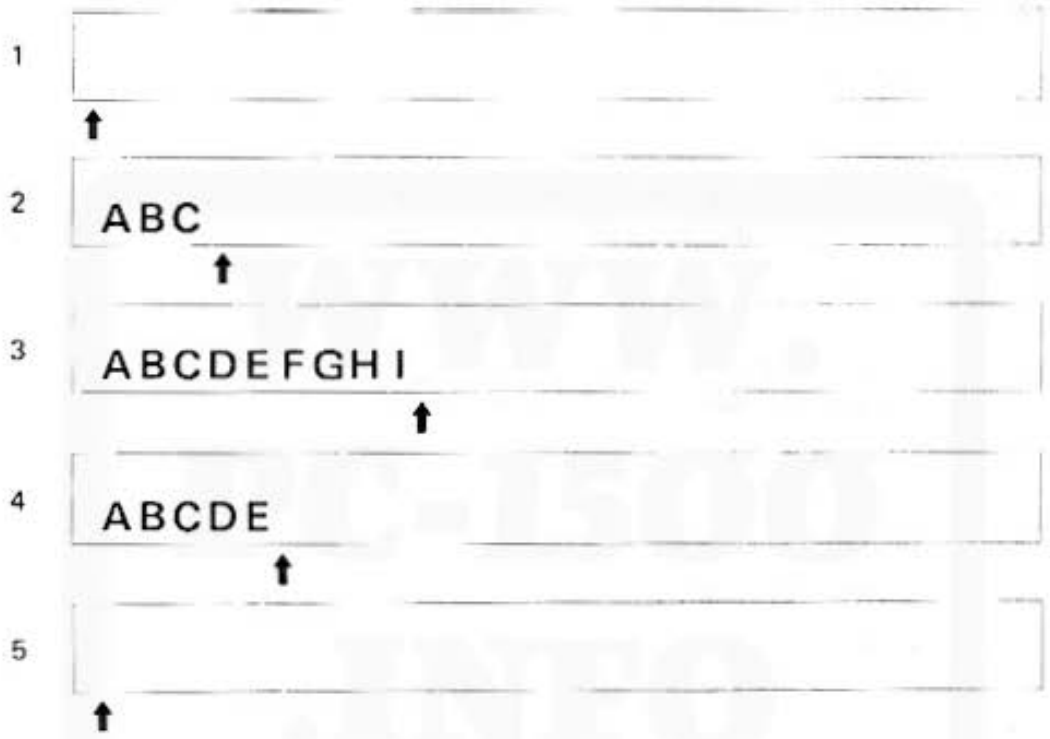
Salida:

```
ESCRIBA A SU ABUELA
```

Cuando las operaciones de concatenación son ejecutadas, un carácter temporal interno del área de almacenamiento es utilizado para construir la nueva serie. Esta área de almacenamiento tiene una capacidad de 80 caracteres. Si la nueva serie excede esta longitud, un ERROR 15 ocurrirá. Una ilustración de esta área durante una operación de concatenación se muestra en el ejemplo siguiente:

Ejemplo 3:

```
X = LEN ("ABC" + LEFT$ ("DEFGHI", 2)) ENTER
```



NOTA: El símbolo ⬆ representa un carácter interno apuntador que vigila la cantidad de almacenamiento utilizado.

1) Al comienzo de la ejecución, el área de almacenamiento será despejado y el carácter apuntador será reajustado a la posición de partida.
2) "ABC" es entrado en el área, tomando las tres primeras posiciones.
3) "DEFGHI" es añadido al área de almacenamiento de caracteres siguiendo "ABC".
4) La función LEFT$ actúa sobre la serie "DEFGHI" para extraer "DE", la cual reemplaza "DEFGHI" en el área.
5) La asignación es ejecutada y el área de almacenamiento es despejada otra vez.

B-3. Comparación de Series

Series de caracteres pueden ser comparadas para determinar qué serie es "mayor" o "menor"

que la otra. Estas determinaciones están basadas en la secuencia comparada (dadas en el Apéndice C) que es el orden de todos los caracteres reconocidos por la computadora.

Si la serie contiene una cantidad de caracteres desigual, la serie más corta es rellenada con caracteres nulos (ASCII 0). Los operadores legítimos para comparación de series son:

- = Cierto si las dos series son iguales en longitud y contienen los mismos caracteres en el mismo orden.
- <> Cierto si las dos series difieren en longitud, caracteres, uorden de caracteres.
- > Cierto si los caracteres de la primera serie son "mayores" que los caracteres de la segunda serie.
- < Cierto si los caracteres de la primera serie son "menores" que los caracteres de la segunda serie.

El formato para la comparación de series es:

serie de caracteres / variable de caracteres	OP	serie de caracteres / variable de caracteres

donde OP es uno de los operadores de la comparación nombrados arriba.

Ejemplos:

"MARY" > "MARI"	es Cierto
"MARY" = "MARY "	es Falso
"abc" <> "ABC"	es Cierto
"DATA 1" < "DATA 2"	es Cierto
"?" < "#"	es Falso

Nota: La forma A$ <= B$, A$ >= B$ no puede ser utilizada para comparar series de caracteres. Comparaciones son posibles en las formas (A$ < B$) O (A$ = B$) y (A$ > B$) O (A$ = B$).

C. Funciones

C-1. ASC

Hay dos funciones utilizadas en la conversión de caracteres del código ASCII. La función ASC convierte un carácter en su código decimal ASCII. La función inversa CHR$ convierte el código decimal ASCII en una serie de caracteres.

ASC { "carácter" / nombre-variable de caracteres }

El argumento en esta función es cualquier serie de caracteres o una serie de caracteres de nombre variable. El valor devuelto por esta función es el código correspondiente ASCII para el primer carácter de la serie especificada.

Ejemplo 1:

```
10 LET X$ = "PATTI"
20 LET A = ASC X$
30 PRINT A
```

Salida:

```
RUN
 80
```

En el ejemplo arriba X$ es asignada con los caracteres "PATTI". La función ASC toma el primer carácter (P) y lo convierte a su código ASCII (80).

Ejemplo 2:

```
10 PRINT ASC "K"
```

Salida:

```
RUN
 75
```

ASC "K" coloca el código ASCII de la "K" el cual es 75.

C-2. CHR$

CHR$ es el complemento de la función ASC. La función CHR$ toma un código decimal, del 0 al 127, y devuelve la serie de caracteres equivalentes. (Nota: algunos códigos representan caracteres especiales que no se imprimen).

CHR$ { ASCII código decimal
       nombre-variable-numérico

Ejemplo 1:

```
10 PRINT (CHR$ 67) + "OP"
```

Salida:

```
RUN
COP
```

Ejemplo 2:

```
10 Z = 65
20 PRINT CHR$ Z
```

Salida:

```
                    RUN
A
```

El primer ejemplo muestra la utilización de un código decimal ASCII como argumento. El resultado es concatenado a la serie "OP" produciendo "COP". El segundo ejemplo asigna un valor numérico a la variable Z. El nombre de la variable es entonces utilizado como el argumento, resultando en el carácter "A".

Nuestro próximo ejemplo de programa convierte las letras mayúsculas de un texto a letras minúsculas, utilizando las funciones ASC y STR$:

Ejemplo 3:

```
 5 WAIT 0
10 INPUT "ESCRIBA MENSAJE", M$
20 FOR I = 1 TO LEN (M$)
30 T$ = MID$ (M$, I, 1)
40 L = ASC (T$)
45 IF (L<65) OR (L>90) THEN 60
50 T$ = CHR$ (L + 32)
60 PRINT T$;
70 NEXT I
80 WAIT : PRINT
```

C-3. INKEY$

Esta función toma cualquier carácter del teclado y lo almacena en la variable especificada. No hay necesidad de pulsar [ENTER] porque el carácter será automáticamente aceptado.

variable = INKEY$

Durante la ejecución de esta sentencia, una prompt no está representada a menos que una sentencia PRINT previa sea utilizada. La instrucción no es devuelta a la representación visual y, por lo tanto, ésta queda inalterada.

Ejemplo:

```
10 A$ = "  "
20 A$ = INKEY$
30 IF A$ = "S" THEN 50
40 GOTO 20
50 PRINT A$
```

Este programa no responderá hasta que una letra S se ingrese.

Ejempo:

```
10 WAIT 0
20 A$ = INKEY$
30 IF A$ = " " THEN PRINT "NINGUNA TECLA": GOTO 20
40 PRINT A$
50 GOTO 20
```

Esta función aceptará solamente un carácter. Si más de uno es pulsado, sólo el primer carácter será representado visualmente. Todos los demás serán ignorados.

C-4. LEN

Durante la manipulación de caracteres es deseable conocer el número de caracteres en una serie. Esto se puede hacer utilizando la función LEN. Esto devuelve el número de caracteres en una expresión especificada o carácter variable.

LEN { "serie de caracteres" / nombre-variable de caracteres }

Ejemplo 1:

```
10 A$ = "CARLA"
20 C = LEN A$
30 PRINT C
```

Salida:

```
RUN
 5
```

Ejemplo 2:

```
10 C = LEN "GATO"
20 PRINT C
```

Salida:

```
RUN
 4
```

Si LNE es utilizada en una serie vacía (i.e. no se incluye nada entre comillas), un cero será devuelto.

Ejemplo 3:

```
10 A = LEN " "
20 PRINT A
```

Salida:

```
RUN
 0
```

Salida:

C-5. LEFT$

Hay tres funciones utilizadas para seleccionar o extraer secciones especificas de una serie de caracteres. LEFT$ extrae caracteres de la izquierda, RIGHT$ de la derecha, y MID$ del medio.

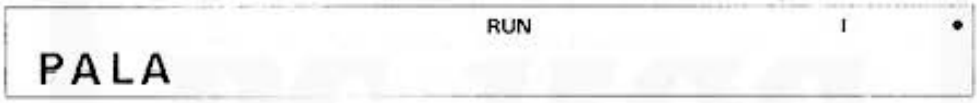
LEFT$ { ("serie de caracteres", número)
(nombre-variable de caracteres, número)

El argumento "número" especifica cuantos caracteres tienen que ser extraidos empezando en el lado izquierdo.

Ejemplo 1:

```
10 A$ = LEFT$ ("PALABRA", 4)
20 PRINT A$
```

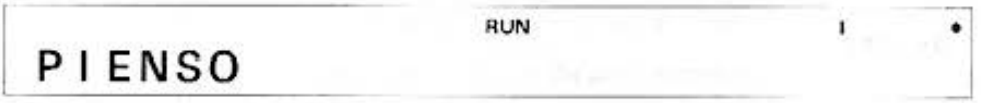
Salida:



Ejemplo 2:

```
10 B$ = "PIENSO MUCHO"
20 A$ = LEFT$ (B$, 6)
30 PRINT A$
```

Salida:



En ambos ejemplos, empezando en el lado izquierdo de la serie de caracteres, los caracteres son extraidos y almacenados en A$. Imprimiendo A$ resulta en "PALA" y "PIENSO" respectivamente.

C-6. MID$

Para extraer la porción intermedia de una serie de caracteres, utiliza la función MID$.

MID$ { ("serie de caracteres", exp 1, exp 2)
(nombre-variable de caracteres, exp 1, exp 2)

Ejemplo 1:

```
10 AS = "RECONCENTRAR"
20 BS = MIDS (A$, 3, 3)
30 PRINT BS
```

Salida:

```
RUN
CON
```

Ejemplo 2:

```
10 TS = MID$ (" (415) 743 – 1602", 6, 3)
20 PRINT TS
```

Salida:

```
RUN
743
```

El primer argumento de esta función es una serie de caracteres o una variable de caracteres. El segundo argumento es un número representando el primer carácter que es extraído. El tercer argumento es el número total de caracteres, incluyendo el primero, para ser extraído. En el primer ejemplo, la palabra "RECONCENTRAR" es almacenada en A$. MID$ extrae tres caracteres empezando con el tercero, y los coloca en B$. Cuando se imprime B$, se encuentra que contiene "CON". En el segundo ejemplo, una serie conteniendo un número de teléfono es el primer argumento. El sexto carácter ("7") es localizado, y junto con los dos caracteres siguientes es almacenado en la variable T$. Cuando se imprime, el resultado es "743".

C-7. RIGHT$

La función RIGHT$ trabaja como LEFT$. La única diferencia es que empieza en el lado opuesto (derecha) al final de la serie. Los argumentos son los mismos que LEFT$:

RIGHT$ { ("serie de caracteres", número)
(nombre-variable de caracteres, número)

El argumento "número" de esta función especifica cuántos caracteres tienen que ser extraídos de la serie de caracteres empezando en el lado derecho.

Ejemplo 1:

```
10 X$ = "PUNTO Y COMA"
20 Y$ = RIGHT$ (X$, 4)
30 PRINT Y$
```

En este programa la función RIGHT$ toma cuatro caracteres de la esquina derecha de la serie y los almacena en la variable Y$. El contenido de la variable Y$ ahora es "COMA".

C-8. RND

Habrá veces en que usted querrá proveer a su programa con números aleatorios. La función RND permite a la computadora generar números aleatorios en un margen que va uno a un número especificado (Nota: El margen siempre empieza con uno).

Ejemplo 1:

```
10  A = RND  5
```

En este ejemplo, A podría tener cualquier valor entre uno y cinco, inclusive. Si usted quiere números aleatorios en un margen que empiece con un número que no sea 1 (por ejemplo de 40 a 50) usted tendrá que simular esto generando números aleatorios de 1 a 10 y añadiendo una constante (39 en nuestro ejemplo):

Ejemplo 2:

```
10  FOR I = 1  TO  5
20  B = 40 + RND  10
40  PRINT B;
50  NEXT I
```

Salida:

```
                                RUN
42  44  47  48  42
```

C-9 RANDOM

Los números al azar son generados a través de una fórmula matenática y son accesibles utilizando la función RND. Cuando se enciende la computadora, una serie de números es generada por la misma. Esta lista queda inalterada a menos que la función RANDOM sea utilizada. Esto significa que un programa utilizará la misma serie de números casuales cada vez que se encienda la computadora. Para prevenir esto, la función RANDOM repone de nuevo la "semilla" utilizada por la fórmula para generar sus números al azar.

Ejemplo 1:

```
10  FOR I = 1  TO  5
15  A = RND  3
20  PRINT A;
30  NEXT I
```

Salida: **1 2 2 3 1**

Ejemplo 2:

```
10  FOR I = 1  TO  5
15  A = RND  3
20  PRINT A;
30  NEXT I
```

Salida: **1 2 2 3 1**

Para obtener números auténticamente al azar en este caso, la función RANDOM debe aprecer antes que la sentencia RND. Esta función siembra una nueva semilla en la generación de números al azar y causa así que los números difieran. En conformidad, un programa ejecutado bajo condiciones idénticas producirá diversos resultados.

Ejemplo 1:

```
10 RANDOM
15 FOR I=1 TO 5
20 A=RND 3
30 PRINT A;
40 NEXT I
```

Salida: 3 1 2 2 3

Ejemplo 2:

```
10 RANDOM
15 FOR I=1 TO 5
20 A=RND 3
30 PRINT A;
40 NEXT I
```

Salida: 2 3 1 3 2

C-10. STR$

STR$ funciona de una manera opuesta a VAL. Esta función convertirá una variable numérica interna a su serie de caracteres o expresión equivalente.

Ejemplo:

```
10 INPUT "PULSE UN NUMERO "; I
20 S$=STR$ (I)
30 PAUSE "ESE NUMERO ES"
40 PRINT "LA SERIE "; S$
```

El programa anterior acepta un número, forma la representación de ese número, y lo almacena en el carácter variable S$. Debido a que la representación numérica interna no puede ser representada visualmente, ésta es automáticamente convertida en una serie de caracteres por la sentencia PRINT.

C-11. STATUS

Para representar la memoria del programa que le queda disponible o la memoria que un programa utiliza, la función STATUS es utilizada. STATUS 0 presentará los "steps" (pasos) del almacenamiento que hay todavía disponibles. STATUS 1 presenta el número de "steps" ya utilizados. El número máximo de "steps" disponibles es 1850.

Ejemplo:

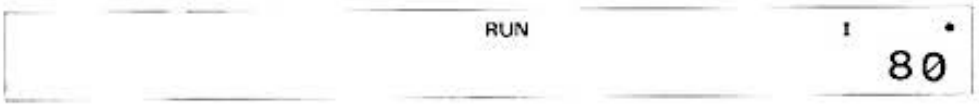
```
STATUS 0
```

Salida:

```
RUN
 1771
```

STATUS 1

Salida:



La instrucción STATUS 0 es equivalente a la instrucción MEM de la PC-1211. MEM también es una orden válida en la PC-1500.

C-12. TIME

Para representar o fijar el mes, día, y hora, la función TIME es utilizada de la forma siguiente:

Fijación: TIME = MMDDHH [•] MMSS [ENTER]

Representación: TIME [ENTER]

donde: MM representa los dos dígitos del mes, DD los dos dígitos del día, y HH los dos dígitos para la hora. La porción fraccional define los minutos (MM) y segundos (SS).

Cuando se representa TIME, el resultado estará en el mismo formato que le hemos dado. El resultado de la función puede manejarse en la misma forma que un número y puede también ser utilizado libremente en expresiones.

El programa siguiente representa un reloj. No se olvide de fijar la hora antes de ejecutar el programa.

Listado del Programa:

```
10  WAIT 0
20  A$ = STR$ TIME
30  IF TIME > 99999 THEN 50
40  A$ = "0" + A$
50  M$ = LEFT$ (A$, 2)
60  D$ = MID$ (A$, 3, 2)
70  H$ = MID$ (A$, 5, 2)
80  DS$ = M$ + " / " + D$ + " / 82 "
90  DS = VAL (M$ + D$ + "00")
100 PRINT DS$;
110 T = TIME - DS
120 IF T >= 1 THEN 140
130 T = T + 12
140 IF T > 23.5959 GOTO 20
150 CURSOR 18 : PRINT USING "###.####"; T
160 GOTO 110
```

C-13. VAL

VAL y STR$ son funciones complementarias que convierten series de caracteres a y desde una variable numérica. La función VAL convierte una serie conteniendo la representación de un número dentro de un número, que es posteriormente almacenado en una variable.

NOTA: Cuando se utilice cualquier otra cosa que no sea un dígito 0 a 9, · (punto decimal), + (signo positivo), − (signo negativo), o E (notación científica) en la expresión, la conversión ignorará estos caracteres inutilizables.

Ejemplo 1:

```
10 ZS = VAL "−37"
```

Resultado:

ZS contiene el número −37

Ejemplo 2:

```
10 A = VAL "237.6"
```

Resultado:

A contiene el número 237.6

D. PRINT USING

La sentencia USING permite al programador controlar rígidamente el formato de información en la representación visual. Esto permite representaciones visuales de modelos y evita la pérdida de información.

Cuando la cláusula USING aparece, sola o dentro una sentencia PRINT o PAUSE, ésta define el formato para todas las sentencias PRINT o PAUSE subsiguientes hasta que la próxima cláusula USING aparezca en el programa.

Varias cláusulas USING pueden aparecer dentro de una sola sentencia PRINT o PAUSE. En este caso, cada una define el formato a utilizar para presentar las variables registradas hasta que la próxima cláusula USING aparece.

Un formato es especificado via una serie de caracteres especiales llamada "editing string" (una "serie editora"). Los caracteres dentro de la "serie editora" definen las áreas de la representación visual disponible para información y limitan el tipo de información que puede ser presentado en estas áreas. Este esquema es el mismo esquema general empleado por otros lenguajes como COBOL y PL/I.

Una "serie editora" puede ser almacenada en una variable de caracteres. El nombre de la variable reemplazará entonces la "serie editora" utilizando la cláusula USING. Esto permite formatos múltiples que son seleccionados bajo el control de programa.

Los caracteres que pueden ser empleados dentro de una "serie editora" son resumidos más abajo:

TABLA DE CARACTERES-EDITORES

Carácter	USO
#	Especifica un campo numérico. Los números son "justificados a la derecha" dentro de este campo. Si el ancho del campo no es suficiente para retener el número, un ERROR 36 ocurrirá. Los ceros delanteros son convertidos a espacios en blancos.
*	Especifica que se llene de asteriscos las posiciones especificadas de un campo numérico que no contiene datos.
.	Causa que un punto decimal sea presentado dentro de un campo numérico.
,	Es utilizado al principio de un campo numérico para especificar la inserción de comas después de cada tres digitos.
^	Es utilizado dentro de un campo numérico para hacer que el número sea presentado en notación cientifica.
+	Es utilizado en un campo numérico para forzar presentación del signo de los datos.
&	Especifica un campo de caracteres. Los caracteres son "justificados a la izquierda" dentro del campo. Si el largo del campo no es suficiente para retener la serie de datos, la serie es trucada.

NOTA: El largo de un campo numérico debe ser siempre uno más que el largo de los datos para permitir la entrada del signo. Esto es cierto siempre e independientemente de si usted utiliza el o no carácter-editor +.

NOTA: El uso de la coma requiere que usted inserte un signo # extra por cada coma en la "serie editora".

Ejemplos

```
X = PI   Y = 1234   AS = "ABCDEF"
PRINT USING "# # #"; X
```

```
                                3
```

```
PRINT USING "+# # #. # # #"; X
```

```
                           +3. 141
```

```
PRINT USING "###. ##^"; X
```

```
 3. 14E 00
```

```
PRINT USING "###.^"; X
```

```
 3.E 00
```

```
PRINT USING "*######"; Y
```

```
**1234
```

```
PRINT USING "***##"; Y
```

```
 1234
```

```
PRINT USING "&&&&&&&########"; A$; Y
```

```
ABCDEF      1234
```

```
PRINT USING "&&&"; A$
```

```
ABC
```

```
10 U$ = "*######. ##"
20 USING U$
30 PRINT Y; "$"
```

```
**1234. 00$
```

```
PRINT X; "$"
```

```
*****3. 14$
```

```
PRINT USING; AS ; X
```

```
ABCDEF 3. 141592654
```

```
PRINT USING "¤¤¤, ¤¤¤, ¤¤¤"; 246813
```

```
246, 813
```

Nota: Usar el número de marcas # (incluyendo ∗) para la designación de variables (números enteros) en el rango siguiente:
Sin usar una puntuación de 3 dígitos (,): Dentro de 11 (incluyendo signo)
Usando una puntuación de 3 dígitos (,): Dentro de 14 (incluyendo signo)
Esta computadora tiene 10 dígitos significativos para los números.
Cuando se designa un formato que excede 10 números enteros por medio de la sentencia USING y aparece representada la cifra que excede de 10 números enteros por la sentencia PRINT (LPRINT), el número que aparece representado puede ser incorrecto.

Ejemplo: Modo RUN

```
USING "##############"  ENTER  → >
PRINT 888888888888  ENTER  → 888888888800
(LPRINT 888888888888  ENTER  →  88888888880)
        12 dígitos
```

E. Transferencia de Control Computado

Además de las sentencias de control básico del Capítulo III, la PC-1500 provee otras dos sentencias de control de gran utilidad. Estas son la ON GUSUB y la ON GOTO. Como usted ya habrá adivinado por sus nombres, estas sentencias actúan como las sentencias GOSUB y GOTO discutidas previamente. La diferencia, sin embargo, reside en su habilidad para transferir control (es decir, para ejecutar sentencias en una localización diferente) automáticamente. Esto es, las funciones GOTO o GOSUB irán a una de las varias sentencias (o subrutinas) dependiendo del valor de una variable numérica. Esta dependencia en una variable para guiar es lo que da a la sentencia ON el sobrenombre de "sentencias de control computado".

La sentencia ON tiene el formato:

```
                 { GOTO    línea # 1,   línea # 2,   línea # 3,. . . (etc.)
ON expresión     { GOSUB     "            "            "
```

La expresión que sigue a la palabra clave ON debe contener a un ENTERO positivo mayor que cero y menor que la cantidad de números de línea registrados después de las palabras claves GOTO o GOSUB. Durante la ejecución, cuando la computadora encuentra una sentencia ON, ella tranfiere el flujo de ejecución a la línea que corresponde al valor de la expresión.

Una sentencia tipica ON puede ser:

```
ON  TX  GOSUB  100, 200, 250, 300
```

En este caso, la variable TX DEBE contener un número en el margen de 1 hasta 4 porque sólo hay cuatro números de línea registrados. Cualquier otro número en TX resultará en un error pues no hay número de línea correspondiente. Por esta razón, es importante que usted incluya pruebas suficientes (sentencias IF) para asegurarse de que sus expresiones resulten en un número válido.

Las sentencias ON son muy útiles para automatizar una serie de opciones. Por ejemplo, considere el siguiente fragmento de programa que permite al usuario seleccionar una de las varias tablas de impuestos. Sin la sentencia ON, esto podría escribirse:

```
10 PAUSE "TABLAS DE IMPUESTOS PARA"
15 PAUSE "SER UTILIZADA: "
20 PAUSE "(1) SOLTERO,"
30 PAUSE "(2) CASADO,"
40 INPUT "(3) NEGOCIOS ?";TT
50 IF (TT<1) OR (TT>3) GOTO 10
60 REM UTILICE TABLA APROPRIADA
70 IF TT = 1 THEN GOTO 220
80 IF TT = 2 THEN GOTO 300
90 IF TT = 3 THEN GOTO 450
(etc)
```

Utilizando la sentencia ON, podemos consolidar las líneas 70, 80, y 90 en una sentencia única:

```
70 ON TT GOTO 220, 300, 450
```

ON ERROR GOTO

Utilizando otra forma de transferencia controlada, permite a un programa detectar cuando ocurre un error. Después de detectarlo, el program apuede ejecutar sentencias que traten de salir del error. Estas sentencias pueden instruir e informar al usuario, o pueden ahorrar datos valiosos.

La sentencia ON ERROR GOTO indica a la SHARP donde ir al detectar la presencia de un error. La forma de esta sentencia es:

ON ERROR GOTO número de línea

donde número de línea es el correspondiente a una línea del programa conteniendo instrucciones a seguir en caso de un error.

F. PROGRAMACION DE LA REPRESENTACION VISUAL

El visor de la representación visual incorporada en la PC-1500 es un mecanismo extraordinario de resultado flexibles. Para permitir que los programadores exploten la potencia entera de la representación visual, varias sentencias nuevas han sido añadidas al dialecto BASIC utilizado por la PC-1500. Estas extensiones son descritas en esta sección.

La representación visual utiliza la tecnología de cristal líquido para presentar gráficamente hasta 16 caracteres al mismo tiempo. Cada carácter, en el conjunto de caracteres de la computadora, ocupa una matriz de puntos de 5 x 7. Utilizando la instrucción GPRINT, los programadores pueden desarrollar y presentar gráficamente sus propios caracteres.

Para propósitos gráficos, el campo entero de la representación visual puede ser utilizado como una matriz 7 x 156. Puntos individuales dentro de cada una de las 156 columnas pueden ser activados para crear gráficas, figuras o símbolos. La instrucción POINT permite "sensibilizar" cualquier columna con el objecto de descubrir qué puntos están activados en ese momento.

Un parlante y un generador de tonos le permiten al programador añadir la dimensión del sonido a la interacción hombre-máquina. Los tonos pueden ser creados en cualquiera de las 256 frecuencias (margen desde 230 Hz hasta cerca de 7 KHz). La repetición automática de un tono y control de la duración de un tono son también posibles bajo control de programa.

F.1. BEEP

La sentencia BEEP permite al programador crear tonos para jugar, señalar errores, y otras aplicaciones interactivas. El formato de la sentencia BEEP que crea el sonido es:

BEEP expresión 1 , expresión 2 , expresión 3

donde:

expresión 1 es el único parámetro (factor determinante) requerido y especifica cuantas veces el tono es repetido. El margen permisible es de 0 a 65535.

expresión 2 es opcional y especifica la frecuencia de tono (s). Este es un número entre 0 y 255

expresión 3 es también opcional y especifica la duración de cada tono. Esta duración se especifica como un tono en el margen de 0 a 65279.

La sentencia BEEP puede también ser utilizada para apagar y encender el altavoz interno de la PC-1500. Asi, el ruido originado por ua operación de cassette SAVE o LOAD puede ser eliminado.

El formato de la sentencia BEEP que controla el altavoz interno es:

NOTA: Cuando la PC-1500 es apagada y después encendida, el altavoz es restaurado a el modo activo.

Programa de Demostración

```
10  D = 60
20  DATA  14
30  DATA  245, 1, 245, 1, 160, 1, 160, 1
40  DATA  143, 1, 143, 1, 160, 2
50  DATA  180, 1, 180, 1, 195, 1, 195, 1
60  DATA  220, 1, 220, 1, 245, 2
100  READ  X
110  FOR  I = 1 TO X
120  READ  N, S
130  BEEP  1, N, (D * S)
140  NEXT  I
150  END
```

F.2. CURSOR

La sentencia CURSOR coloca el cursor en una de las posiciones de los 26 caracteres en la representación visual. La forma de esta sentencia es:

CURSOR expresión-de-posición

donde:

expresión-de-posición evalúa a un número en el margen 0 a 25 el cual señala a donde el cursor se moverá.

El uso normal de la instrucción CURSOR es colocar el cursor preparado en posición antes de presentar alguna información. Utilizado de esta forma, lé permite al programador definir su propia separación entre datos en las siguientes sentencias:

Listado del Programa:

```
10: WAIT 20
20: X = 8
30: PRINT "A"
40: CURSOR 4
50: PRINT "B"
60: CURSOR X
70: PRINT "C"
80: CURSOR ((X ^ 2) / 4) - 4
90: PRINT "D"
100: END
```

Digitaciones:

[1] [Ø] [W] [A] [I] [T] [2] [Ø] [ENTER]

[2] [Ø] [X] [=] [8] [ENTER]

[3] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [A] [SHIFT] ["] [ENTER]

[4] [Ø] [C] [U] [R] [S] [O] [R] [4] [ENTER]

[5] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [B] [SHIFT] ["] [ENTER]

[6] [Ø] [C] [U] [R] [S] [O] [R] [X] [ENTER]

[7] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [C] [SHIFT] ["] [ENTER]

[8] [Ø] [C] [U] [R] [S] [O] [R] [(] [(] [X] [SHIFT] [^] [2] [)]
[/] [4] [)] [-] [4] [ENTER]

[9] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [D] [SHIFT] ["] [ENTER]

[1] [Ø] [Ø] [E] [N] [D] [ENTER]

Este programa causará que las letras A, B, C, y D aparezcan en las posiciones 0, 4, 8, y 12, respectivamente, en la representación visual.

```
                    RUN
A   B   C   D
```

NOTA: Al especificar una posición de cursor mayor que 25 o menor que 0, resultará en un ERROR 19.

Programa de Demostración

```
10 WAIT 0
20 DIM A$(0)*13
30 INPUT "ENTRE SU NOMBRE", A$(0)
40 C=0 : CLS
50 FOR I=1 TO (LEN A$(0)-1)
60 CURSOR C
70 PRINT MID$(A$(0), I, 1)
80 C=C+2
90 NEXT I
100 WAIT
110 CURSOR C
120 PRINT RIGHT$(A$(0), 1)
130 END
```

Digitaciones:

[1] [Ø] [W] [A] [I] [T] [Ø] [ENTER]

[2] [Ø] [D] [I] [M] [A] [SHIFT] [$] [(] [Ø] [)] [*] [1] [3] [ENTER]

[3] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [E] [N] [T] [R] [E] [SPACE]
[S] [U] [SPACE] [N] [O] [M] [B] [R] [E] [SHIFT] ["]
[SHIFT] [,] [A] [SHIFT] [$] [(] [Ø] [)] [ENTER]

[4] [Ø] [C] [=] [Ø] [SHIFT] [:] [C] [L] [S] [ENTER]

[5] [Ø] [F] [O] [R] [I] [=] [1] [T] [O] [(] [L] [E] [N] [A] [SHIFT] [$] [(] [Ø] [)]
[-] [1] [)] [ENTER]

[6] [Ø] [C] [U] [R] [S] [O] [R] [C] [ENTER]

[7] [Ø] [P] [R] [I] [N] [T] [M] [I] [D] [SHIFT] [$] [(] [A] [SHIFT] [$] [(] [Ø] [)]
[SHIFT] [,] [I] [SHIFT] [,] [1] [)] [ENTER]

[8] [Ø] [C] [=] [C] [+] [2] [ENTER]

[9] [Ø] [N] [E] [X] [T] [I] [ENTER]

[1] [Ø] [Ø] [W] [A] [I] [T] [ENTER]

[1][1][Ø][C][U][R][S][O][R][C][ENTER]

[1][2][Ø][P][R][I][N][T][R][I][G][H][T][SHIFT][$]
[(][A][SHIFT][$][)][Ø][)][SHIFT][;][1][)]
[ENTER]

[1][3][Ø][E][N][D][ENTER]

F.3. CLS

La sentencia CLS borra la representación visual apagando todos los puntos de la pantalla. Es utilizada antes de otras sentencias para borrar cualquier dato previo de la representación visual. El cursor es colocado a la izquierda de la pantalla mediante la sentencia CLS. El formato es simplemente CLS.

Programa de Demostración

```
10 GPRINT "7F7F7F7F7F"
20 CLS
30 PRINT "NUEVA INFORMACION"
```

Digitaciones:

[1][Ø][G][P][R][I][N][T][SHIFT]["][7][F][7][F][7][F]
[7][F][7][F][SHIFT]["][ENTER]

[2][Ø][C][L][S][ENTER]

[3][Ø][P][R][I][N][T][SHIFT]["][N][U][E][V][A][SPACE]
[I][N][F][O][R][M][A][C][I][O][N][SHIFT]["][ENTER]

F.4. GCURSOR

La sentencia GCURSOR señala una de las 156 columnas de puntos (disponibles en la representación visual) como la primera columna para cualquier representación de información subsecuente en la representación visual. El formato de la instrucción GCURSOR es:

GCURSOR posición-expresión

donde:

Posición-expresión evalúa a un número en el margen de 0 a 155. Este número especifica una de las 156 columnas de 7-puntos en la representación visual.

NOTA: Si una posición-expresión resulta en un número que es menor que 0 o mayor que 155, un ERROR 19 ocurrirá.

La sentencia GCURSOR es utilizada generalmente, junto con la sentencia GPRINT, con el propósito de crear representaciones gráticas (esto será ilustrado más detalladamente en la próxima sección). Otros tipos de instrucciones pueden seguir la sentencia GCURSOR puesto que esta

sentencia no escribe ninguna información. GCURSOR simplemente indica en qué columna la información subsecuente será escrita. Esto es ilustrado en los renglones siguientes:

Listado del Programa:

```
10 GCURSOR 50
20 PRINT "A"
30 GCURSOR 80
40 PRINT 26/3
```

Digitaciones:

[1] [Ø] [G] [C] [U] [R] [S] [O] [R] [5] [Ø] [ENTER]

[2] [Ø] [P] [R] [I] [N] [T] [SHIFT] ["] [A] [SHIFT] ["] [ENTER]

[3] [Ø] [G] [C] [U] [R] [S] [O] [R] [8] [Ø] [ENTER]

[4] [Ø] [P] [R] [I] [N] [T] [2] [6] [/] [3] [ENTER]

lo cual produce un resultado que aparece normal en las posiciones 50 y 80:

```
RUN                                    •
                    A          8. 666666667
```

¿Qué pasa si alteramos la línea 30 para que empiece a representar en la posición 93? Pruebe esto al substituir la línea siguiente:

```
30 GCURSOR 93
```

¡Sorpresa! El segundo resultado ha sido "truncado" (cortado) porque se salió de la representación visual.

Como un ejemplo final de la sentencia GCURSOR, le presentamos un programa avanzado que crea un efecto extraño cuando se sobreponen caracteres:

Listado del Programa:

```
10 WAIT 0
20 DIM A$ (0) * 10
30 A$ (0) = " " : C = 0
40 INPUT "ENTRE MENSAJE (< 10 CARS)", A$ (0)
50 CLS
60 FOR I = 1 TO (LEN A$ (0))
70 GCURSOR C
80 FOR J = 1 TO 3
90 GCURSOR C
```

```
100 PRINT MID$ (A$ (0), I, 1)
105 C = C + 3
110 NEXT J
120 C = C + 5
130 NEXT I
140 WAIT
150 GCURSOR 155
160 GPRINT "00"
170 END
```

Digitaciones:

[1] [Ø] [W] [A] [I] [T] [Ø] [ENTER]

[2] [Ø] [D] [I] [M] [A] [SHIFT] [$] [(] [Ø] [)] [*] [1] [Ø] [ENTER]

[3] [Ø] [A] [SHIFT] [$] [(] [Ø] [)] [=] [SHIFT] ["] [SHIFT] ["]
[SHIFT] [:] [C] [=] [Ø] [ENTER]

[4] [Ø] [I] [N] [P] [U] [T] [SHIFT] ["] [E] [N] [T] [R] [E] [SPACE]
[M] [E] [N] [S] [A] [J] [E] [SPACE] [(] [SHIFT] [<] [1] [Ø]
[SPACE] [C] [A] [R] [S] [)] [SHIFT] ["] [SHIFT] [,]
[A] [SHIFT] [$] [(] [Ø] [)] [ENTER]

[5] [Ø] [C] [L] [S] [ENTER]

[6] [Ø] [F] [O] [R] [I] [=] [1] [T] [O] [(] [L] [E] [N] [A] [SHIFT] [$]
[(] [Ø] [)] [)] [ENTER]

[7] [Ø] [G] [C] [U] [R] [S] [O] [R] [C] [ENTER]

[8] [Ø] [F] [O] [R] [J] [=] [1] [T] [O] [3] [ENTER]

[9] [Ø] [G] [C] [U] [R] [S] [O] [R] [C] [ENTER]

[1] [Ø] [Ø] [P] [R] [I] [N] [T] [M] [I] [D] [SHIFT] [$] [(] [A]
[SHIFT] [$] [(] [Ø] [)] [SHIFT] [,] [I] [SHIFT] [,]
[1] [)] [ENTER]

[1] [Ø] [5] [C] [=] [C] [+] [3] [ENTER]

[1] [1] [Ø] [N] [E] [X] [T] [J] [ENTER]

[1] [2] [Ø] [C] [=] [C] [+] [5] [ENTER]

[1] [3] [Ø] [N] [E] [X] [T] [I] [ENTER]

[1] [4] [Ø] [W] [A] [I] [T] [ENTER]

[1] [5] [Ø] [G] [C] [U] [R] [S] [O] [R] [1] [5] [5] [ENTER]

[1] [6] [Ø] [G] [P] [R] [I] [N] [T] [SHIFT] ["] [Ø] [Ø]
[SHIFT] ["] [ENTER]

[1] [7] [Ø] [E] [N] [D] [ENTER]

F.5. GPRINT

La sentencia GPRINT provee control programable directamente sobre los puntos de la pantalla de la representación visual. Como la sentencia GPRINT fija y prepara los puntos dentro de cualquiera de los 7-puntos de una columna, esta es normalmente utilizada en conjunción con la instrucción GCURSOR. La sentencia GCURSOR selecciona la columna apropiada para la modificación y la sentencia GPRINT manipula los puntos dentro de esa columna. La sentencia GPRINT también es capaz de presentar varias columnas de información contiguas en una sentencia única.

Para poder comprender el formato de la sentencia GPRINT, es necesario comprender como los puntos dentro de una columna son controlados. La estructura de puntos energizados dentro de un columna puede ser especificada bien como un número decimal o como una serie de caracteres hexadecimales. Si se utiliza el sistema decimal, entonces cada fila puede ser visualizada como si fuera numerada, de arriba a abajo, por una potencia de dos. Esto se ilustra aquí:

```
1  - - - - - - - -
2  - - - - - - - -
4  - - - - - - - -
8  - - - - - - - -
16 - - - - - - - -
32 - - - - - - - -
64 - - - - - - - -
```

Con 7 puntos por columna, cada uno de los cuales puede estar encendido o apagado, existen 128 estructuras de puntos posibles. Así, para especificar una estructura particular, se utiliza este formato:

GPRINT estructura-expresión ; estructura-expresión 2 . . (etc) . .

donde:

estructura-expresión evalúa a un número que fluctúa del 0 al 127 y especifica la estructura de puntos energizados. Varias estructuras-expresiones pueden ser especificadas opcionalmente pero deben ser separadas por un punto y coma o una coma. Si la coma es utilizada, una columna en blanco aparecerá entre cada columna presentada.

Vamos a ilustrar la utilidad de la instrucción GPRINT creando un nuevo carácter; una Flecha hacia arriba. Primero diseñamos nuestro carácter en una rejilla representando las filas y columnas:

```
1  - - - - * - - - -
2  - - * - * - * - -
4  * - - - * - - - *
8  - - - - * - - - -
16 - - - - * - - - -
32 - - - - * - - - -
64 - - - - * - - - -
     1   2   3   4   5
```

Advertencia

Al corregir en un programa el comando PRINT al comando GPRINT, tener en cuenta lo siguiente:

Ejemplo:

Volver a escribir totalmente "PRINT" en "GPRINT",

Insertando solamente "G" no se conseguirá que la computadora lo juzgue como "GPRINT". (Esto se considera como "G" y "PRINT".)

Lo mismo ocurre cuando se corrige el comando CURSOR a comando GCURSOR, o a los comandos del impresor.

Debido a que nuestro carácter tiene cinco columnas de ancho, necesitaremos cinco números separados en la lista de estructura-expresión de la sentencia GPRINT. Los números que representan las columnas 1 y 5 deben especificar cada uno un punto único en la fila etiquetada 4. De la misma manera, los números representando las columnas 2 y 4 deben especificar cada uno un punto único en la fila 2. La sentencia final es:

```
GPRINT 4;2;127;2;4
```

La especificación de la tercera columna (127) es el único número cuya derivación puede que no sea inmediatamente obvia. El número 127 es la suma de un punto en la primera fila (1), un punto en la segunda fila (2), un punto en la tercera fila (4), etc. Así, 127 es 1 + 2 + 4 + 8 + 16 + 32 + 64 y especifica los 7 puntos de la columna. Cualquier estructura puede crearse especificando una fila o una suma de varias filas.

Si el esquema hexadecimal es utilizado, las 7 filas de la representación visual son conceptualmente divididas en un grupo inferior de 3 filas y en un grupo superior de 4 filas. Cada grupo está numerado, desde su fila superior, por la potencia de 2 como se ilustra más abajo:

```
1  - - - - - - - -
2  - - - - - - - -
4  - - - - - - - -
8  - - - - - - - -
1  - - - - - - - -
2  - - - - - - - -
4  - - - - - - - -
```

De esta manera, es posible representar todas las estructuras de un grupo por una sola cifra hexadecimal. Debido a que el grupo inferior sólo tiene 3 filas, el margen de dígitos permisibles para este grupo será de 0 a 7. De los dos dígitos hexadecimales requeridos para especificar una columna entera, el primer dígito representa el grupo inferior y el segundo representa el grupo superior.

La forma de la GPRINT hexadecimal es:

GPRINT "serie hexadecimal"

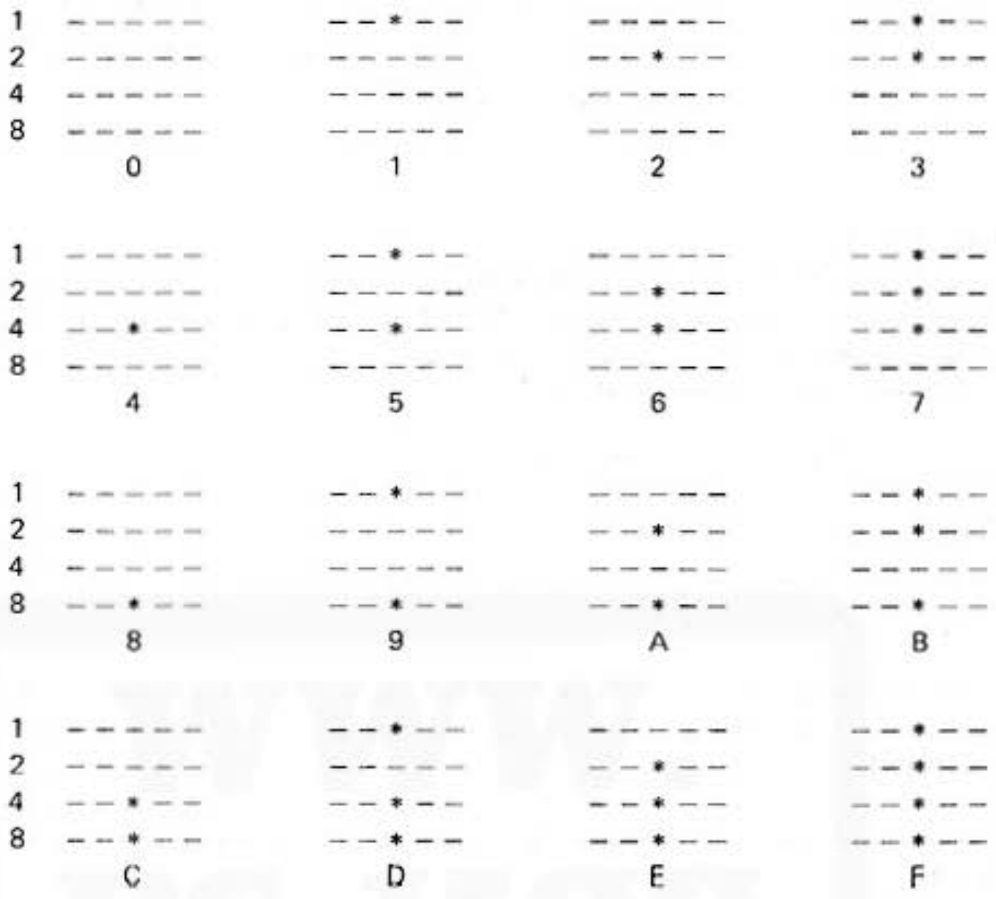
donde:

serie hexadecimal es una serie de pares de dígitos hexadecimales, donde cada par especifica la estructura de puntos de una columna.

Usando este formato para crear el caracter de flecha hacia arriba del ejemplo anterior, nos daría la sentencia:

```
GPRINT "04027F0204"
```

Tabla de Caracteres Hexadecimales



Programa de Demostración

Este programa presenta todas las estructuras de puntos posibles, en orden, del 0 al 127.

Listado del Programa:

```
10 WAIT 0 : CLS
20 FOR I=0 TO 127
30 GCURSOR I
40 GPRINT I
50 NEXT I
60 WAIT
70 GCURSOR 155 : GPRINT "00"
80 END
```

Digitaciones:

1 Ø W A I T Ø SHIFT : C L S ENTER

2 Ø F O R I = Ø T O 1 2 7 ENTER

3 Ø G C U R S O R I ENTER

4 Ø G P R I N T I ENTER

5 Ø N E X T I ENTER

```
6 Ø W A I T ENTER
7 Ø G C U R S O R 1 5 5 SHIFT :
      G P R I N T SHIFT " Ø Ø SHIFT " ENTER
8 Ø E N D ENTER
```

F.6. POINT

La función POINT devuelve un número que representa la estructura de puntos activados dentro de una columna dada. Así, la función POINT permite la "sensibilización" de cualquier columna en la representación visual bajo control de programa.

El formato de la función POINT es:

POINT posición-expresión

donde:

posición-expresión evalúa un número en el margen de 0 a 155 y representa la columna que va a ser investigada.

El valor devuelto por la función POINT es un número en el margen de 0 a 127. La interpretación de este número es una suma de potencias de dos, tal como se explica en la sección de la GPRINT.

Para ilustrar esto, asuma que en la representación visual hay una I mayúscula en las columnas 40 a 44:

```
1   - - * - * - * - -
2   - - - - * - - - -
4   - - - - * - - - -
8   - - - - * - - - -
16  - - - - * - - - -
32  - - - - * - - - -
64  - - * - * - * - -
        4   4   4   4   4
        0   1   2   3   4
```

La expresión:

POINT 40 volvería 0,
POINT 41 volvería 65,
POINT 42 volvería 127.

Programa de Demostración

El programa registrado aquí llena la representación visual con el carácter almacenado en la variable A$ y crea estructuras insólitas al invertir este carácter. Este programa utiliza varias de las sentencias discutidas en este capítulo.

Listado del Programa:

```
10 A$ = "X"
20 WAIT 0
30 Y = 5 : X = 155/Y : C = 0
40 FOR I = 1 TO X
50 GCURSOR C
60 PRINT A$
70 C = C + Y
80 NEXT I
90 FOR I = 0 TO 155
100 GCURSOR I
110 A = 127 - POINT I
120 GPRINT A
130 NEXT I
140 GOTO 90
```

Digitaciones:

[1] [Ø] [A] [SHIFT] [$] [=] [SHIFT] ["] [X] [SHIFT] ["] [ENTER]

[2] [Ø] [W] [A] [I] [T] [Ø] [ENTER]

[3] [Ø] [Y] [=] [5] [SHIFT] [:] [X] [=] [1] [5] [5] [/] [Y]
[SHIFT] [:] [C] [=] [Ø] [ENTER]

[4] [Ø] [F] [O] [R] [I] [=] [1] [T] [O] [X] [ENTER]

[5] [Ø] [G] [C] [U] [R] [S] [O] [R] [C] [ENTER]

[6] [Ø] [P] [R] [I] [N] [T] [A] [SHIFT] [$] [ENTER]

[7] [Ø] [C] [=] [C] [+] [Y] [ENTER]

[8] [Ø] [N] [E] [X] [T] [I] [ENTER]

[9] [Ø] [F] [O] [R] [I] [=] [Ø] [T] [O] [1] [5] [5] [ENTER]

[1] [Ø] [Ø] [G] [C] [U] [R] [S] [O] [R] [I] [ENTER]

[1] [1] [Ø] [A] [=] [1] [2] [7] [-] [P] [O] [I] [N] [T] [I] [ENTER]

[1] [2] [Ø] [G] [P] [R] [I] [N] [T] [A] [ENTER]

[1] [3] [Ø] [N] [E] [X] [T] [I] [ENTER]

[1] [4] [Ø] [G] [O] [T] [O] [9] [Ø] [ENTER]

NOTA: Si la última línea es incluída, este programa entrará en un bucle infinito cuando se ejecute. (Esto puede pararse con la tecla BREAK). El carácter representado puede cambiarse insertando un nuevo carácter en la asignación de la línea 10.

G. Depurando

Aunque usted sea muy cuidadoso, eventualmente creará un programa que no hace exactamente lo que usted esperaba. Para aislar el problema, los diseñadores de SHARP han provisto un método especial de ejecución de programas conocido como modo "Trace". En el modo Trace, la PC-1500 representará visualmente el número de línea de cada línea de programa y se parará después de ejecutar esa línea. Esto le permite seguir (o trazar) la secuencia de instrucciones de la forma en que exactamente son ejecutadas. Cuando el programa pausa después de la ejecución de ua línea, usted puede inspeccionar o alterar los valores de las variables.

La forma de la instrucción para iniciar el modo Trace es simplemente: TRON. La instrucción TRON puede ser emitida como un comando (en el modo RUN) o puede fijarse, como una sentencia, dentro de un programa. Utilizada como un comando, TRON informa a SHARP que se requiere trazar durante la ejecución de todos los programas subsecuentes. Los programas a ser trazados son entonces empezados en una manera normal, con la orden GOTO o RUN.

Si TRON es utilizado como una sentencia, iniciará el modo Trace sólo cuando la línea conteniéndolo se ejecute. Si, por alguna razón, esa línea nunca es alcanzada, el modo Trace permanerá inactivo.

Una vez iniciado, el modo de operación Trace queda en efecto hasta que es cancelado por una instrucción TROFF. La Instrucción TROFF puede ser emitida como un comando o como una sentencia. El modo Trace también puede ser cancelado por la secuencia de teclas:

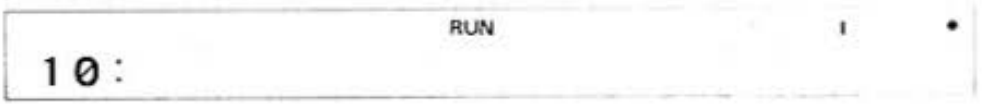
[SHIFT] [CL]

Como un ejemplo de como utilizar el modo Trace, pulse el siguiente programa que computa la longitud de la hipotenusa de un triángulo, dada la longitud de los lados:

Listado del Programa:

```
10 INPUT A, B
20 A = A * A : B = B * B
30 H = √ (A + B)
40 PRINT "HIPOTENUSA = "; H
```

En el modo RUN, emita el comando TRON, seguida por el comando RUN. Observe que el comando INPUT opera de manera usual al presentar un signo de interrogación por cada valor. Tan pronto como haya entrado dos valores, aparece el número de línea de la sentencia INPUT:



Pulsando la tecla [↑] (flecha hacia arriba) y reteniéndola, usted puede revisar la línea entera:

Para continuar el programa, pulse la tecla [↓] (flecha hacia abajo) una vez. Esto causa que la próxima línea sea ejecutada y su número representado. De nuevo, usted puede revisar la línea con la tecla [↑] (flecha hacia arriba). Usted puede también comprobar el contenido de cualquier variable escribiendo su nombre y pulsando [ENTER] :

[A] [ENTER] donde A es una variable de programa

```
                    RUN                    I    •
                                             4
```

Es necesario pulsar la tecla [↓] (flecha hacia abajo) una vez por cada línea que va a ser ejecutada hasta que el programa finalice. Si usted no desea continuar la ejecución normal línea por línea, pulse la tecla [ENTER] para suspender la ejecución del programa. Si usted cambia de idea nuevamente, los programas suspendidos pueden ser continuados con la orden CONT.

Un ejemplo, utilizando nuestro programa de la hipotenusa, sería el siguiente:

Digitaciones	Representación visual
	>
T R O N	TRON_
ENTER	>
R U N	RUN_
ENTER	?
3	3_
ENTER	?
4	4_
ENTER	10:
↑	10: INPUT A, B_
↓	20:
↑	20: A=A*A: B=B*B_
A	A_
ENTER	9
B	B_
ENTER	16
↓	30:
H	H_
ENTER	5
↓	HIPOTENUSA= 5
↑	40: PRINT "HIPOTENUSA= ";H_
↓	40:
↓	>

H. NUMEROS HEXADECIMALES Y FUNCIONES DE BOOLE

H.1. Números Hexadecimales

La PC-1500 tiene la capacidad de utilizar un número hexadecimal (base 16) dentro de cualquier expresión en la cual un número decimal puede ser utilizado. Los números hexadecimales se distinguen de los números decimales en que los precede un signo &. Los siguientes son números hexadecimales válidos:

&16 &F &7ECA &08 &99A −&5B

Los números hexadecimales pueden ser utilizados en cálculos:

10 + &A [ENTER]

```
RUN
                                    20
```

O dentro de programas:

```
Programas:

35 GPRINT &F, 54, &3E
40 DATA 67, &7F, &2B, 12, 305
```

H.2. Función AND

La función AND provee la operación AND de Boole de la representación interna de dos valores. Los valores deben estar entre − 32768 y 32767. Los números que excedan este margen causarán un ERROR 19.

Ejemplo:	Resultado:
10 AND &F	10
1 AND 0	0
−1 AND 1	1
55 AND 64	0
16 AND 63	16

H.3. Función OR

La función OR efectúa un OR de Boole en la representación interna de dos valores. Los valores deben estar entre −32768 y 32767. Los números que excedan este margen causarán un ERROR 19.

Ejemplo:	Resultado:
10 OR &F	15
1 OR 0	1
−1 OR 1	−1
55 OR 64	119
16 OR 63	63

H.4. Función NOT

La función NOT devuelve el NOT de Boole, o complemento, de la representación interna de un valor único. El valor debe estar entre −32768 y 32767. Si el valor excede este margen, un ERROR 19 ocurrirá.

Ejemplo:	Resultado:
NOT Ø	−1
NOT &F	−16
NOT 55	−56
NOT 1	−2
NOT −2	1

I. DETENER LA EJECUCION DEL PROGRAMA

STOP, CONT

La sentencia STOP (alto) causa que la computadora suspenda la ejecución de un programa. Cuando el programa se interrumpe, los valores de todas las variables son retenidos y el programador puede inspeccionarlas y cambiarlas. El programa puede entonces continuarse, en el punto donde fue detenido, con la orden CONT (para CONTinuar).

Cuando la computadora encuentra la sentencia STOP, un mensaje similar al siguiente aparece en la representación visual:

```
                                RUN                    1          •
BREAK IN 60
```

donde 60 es el número de línea que contiene la sentencia STOP.

Si usted desea revisar esta línea, pulse y retenga la tecla [↑] (flecha hacia arriba).

Cuando el mensaje BREAK aparece, usted puede también revisar y cambiar los valores de las variables. Por ejemplo:

HQ [ENTER]

```
                    56.23
```

AS [ENTER]

```
"DEDUCCIONES"
```

Cuando usted esté listo para continuar la ejecución, simplemente vuelva al prompt (>) y escriba CONT [ENTER] .

J. CONTROL DE MODOS

LOCK, UNLOCK

La instrucción LOCK puede ser utilizada para controlar el modo (RUN, PROgrama, y RESERVE) en el cual la computadora opera. Incluído dentro de un programa, previene que el usuario cambie accidentalmente el modo y dañe el programa. La instrucción LOCK incapacita la tecla MODE, "cerrando" la computadora en el modo en que está siendo utilizada corrientemente.

Para habilitar nuevamente la tecla MODE, se utiliza la instrucción UNLOCK. UNLOCK repone el funcionamiento normal permitiendo cambios de modo.

Ambas instrucciónes pueden ser utilizadas como comando manual o como sentencia. Las formas son simplemente:

LOCK

UNLOCK

VI. EXPANDIENDO LA PC-1500

A. El IMPRESOR/INTERFAZ DE CASSETTE (CE-150)

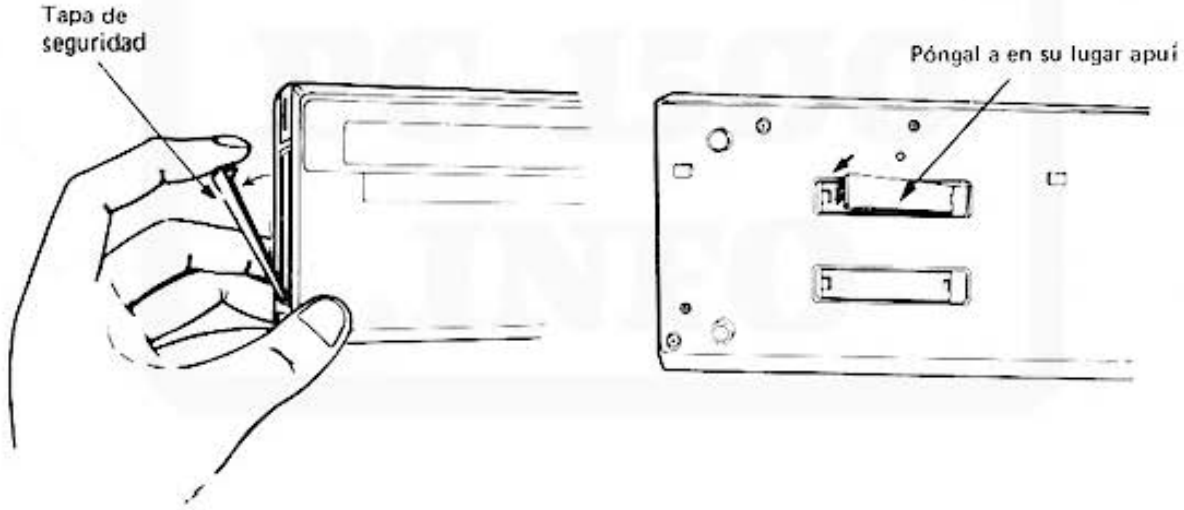
El impresor/interfaz de cassette es una opción de la computadora de bolsillo SHARP PC-1500. Esta unidad puede ser conectada a una o dos grabadores de cassettes. Los grabadores pueden ser utilizados para almacenar programas y datos en cassettes estándar de audio. Los programas pueden ser cargados en la PC-1500 para ser utilizados posteriormente, ahorrándole el trabajo de escribirlos otra vez.
La utilización de estas formas especiales será discutida en las secciones siguientes.

1. Conectando la computadora al impresor/interfaz

Conecte el impresor/interfaz (CE-150) y la computadora (PC-1500) de la siguiente forma:

(1) Apague la computadora (OFF).
Nota importante: Es esencial que la computadora esté apagada. Si está encendida, la computadora puede que se vuelva inoperable. Si esto ocurre, pulse el interruptor ALL REST en la parte de abajo de la computadora mientras pulsa la tecla [ON]

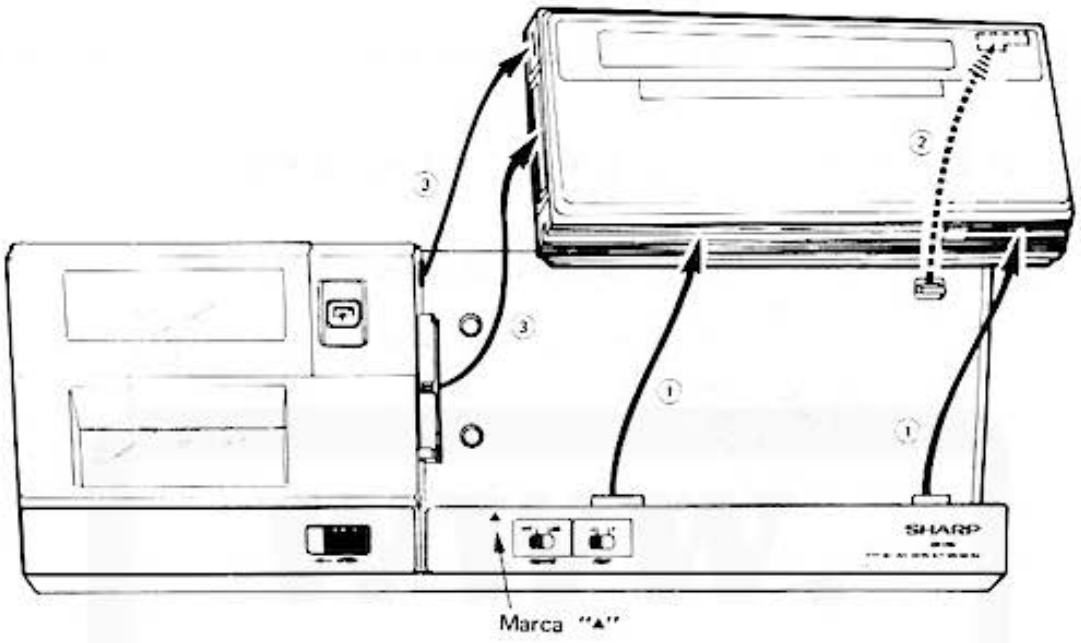
(2) Quite la tapa de seguridad de la cubierta de la parte izquierda de la computadora y póngala en su lugar en la parte inferior del impresor (vea la figura).

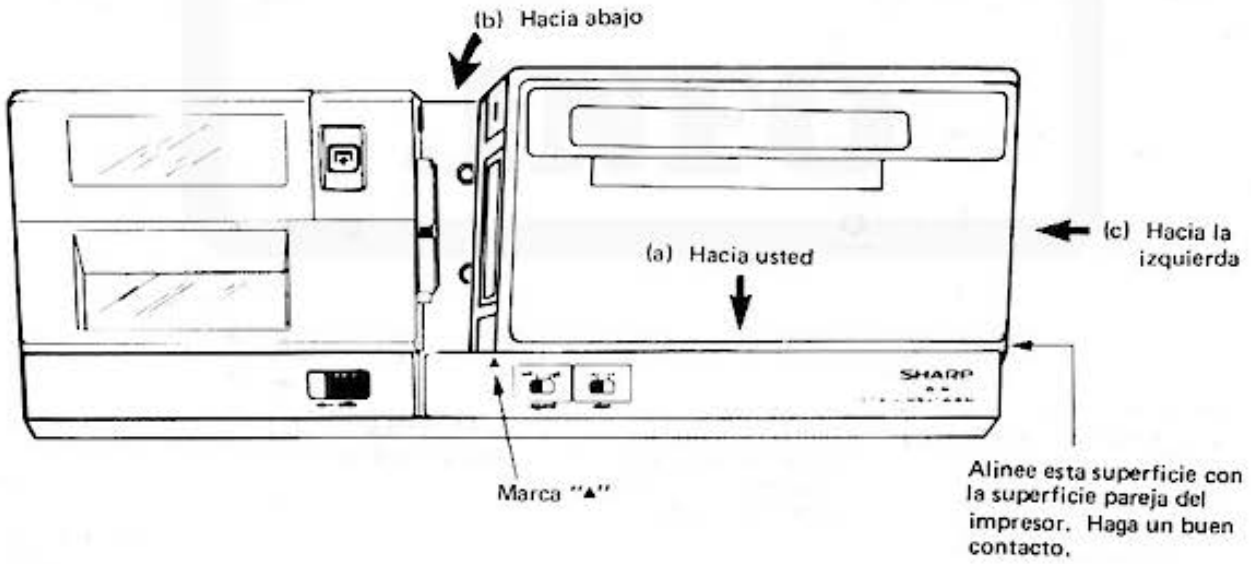


(3) Coloque el lado inferior de la computadora en su lugar para que la guía del impresor se alinee con la guía de la hendidura.

(4) Coloque la computadora en forma plana.

(5) Suavemente deslice la computadora a la izquierda para que los enchufes del impresor se inserten en la computadora (vea la figura)





No fuerce el impresor ni la computadora. Si el alineamiento no se produce fácilmente, deslice la computadora cuidadosamente de izquierda a derecha hasta conseguir que la posición de las superficies se iguale.

2. Recargando las baterías

La CE-150 utiliza una batería recargable Ni-CAD. Es necesario pues recargar la batería después de desenvolverla, y cuando el mensaje siguiente sea representado gráficamente. (En este caso, el impresor está bloqueado. Para desbloquear el impresor, pulse las teclas [OFF] y [ON] de la computadora en este orden después de cargar la batería.)

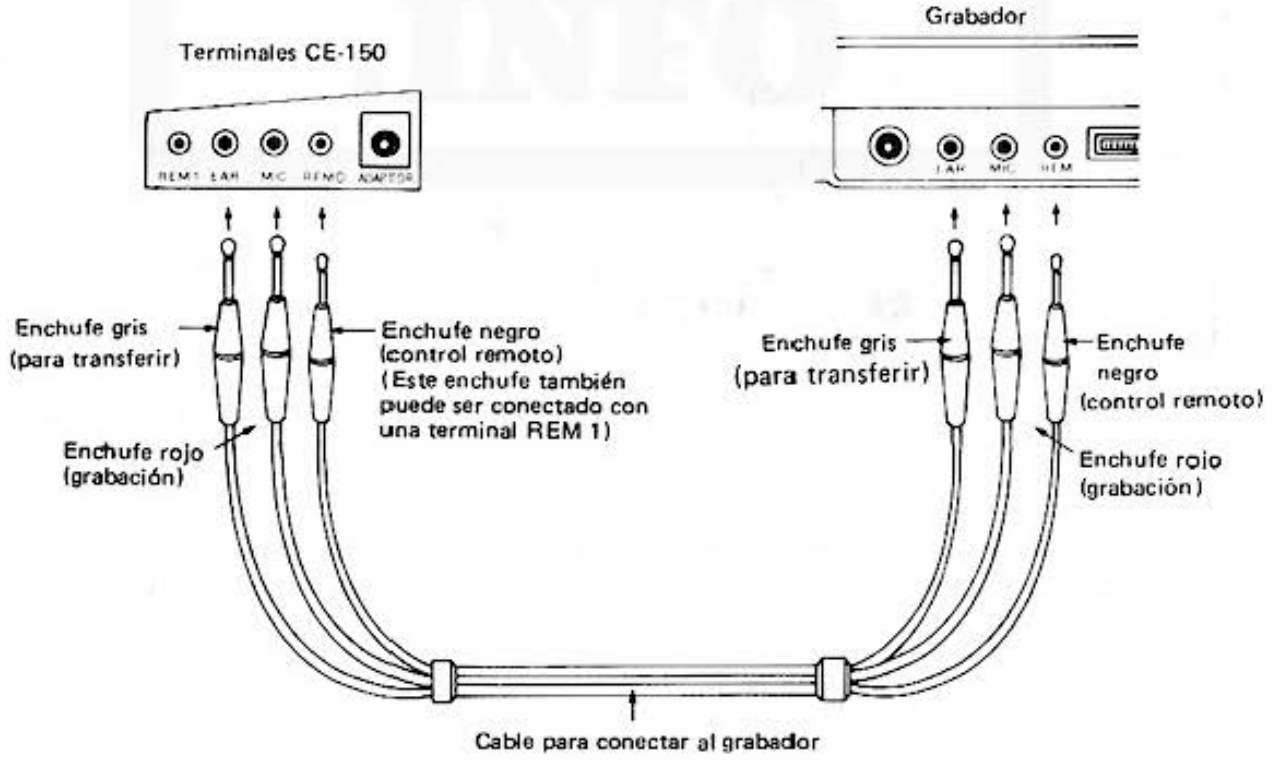
(1) E R R O R 8 Ø o E R R O R 7 8

Nota: Cuando el impresor está en estado de reemplazamiento de la pluma, el mensaje ERROR 78 puede aparecer.

(2) : C H E C K 6 o N E W Ø ? : C H E C K 6

3. Conectando un grabador al interfaz

Primero conecte la CE-150 y la computadora, y conecte después un grabador con la CE-150 como muestra el diagrama siguiente:



Lo siguiente es una descripción de las especificaciones mínimas necesarias para acoplar un grabador con la CE-150.

Item	Requisitos
1. Tipo de grabador	Cualquier grabadora a cassette, micro-cassette, o grabador de cinta magnética puede ser utilizado de acuerdo con los requisitos descritos arriba.
2. Enchufe de entrada	El grabador tendrá un mini enchufe etiquetado "MIC". Nunca utilice el enchufe "AUX".
3. Impedancia de entrada	El enchufe de entrada debe ser de impedancia baja (200–1,000 OHM.)
4. Nivel de entrada mínima	Inferior a 3mV o –50dB.
5. Enchufe de salida	Debe ser un minienchufe etiquetado "EXT", "MONITOR", "EAR" o equivalente.
6. Impedancia de salida	Debe ser inferior a 10 OHM.
7. Nivel de salida	Debe estar dentro del 15% en un rango de 2kHz a 4kHz. 100mW)
8. Distorsión	Debe estar dentro del 15% en un rango de 2kHz a 4kHz.
9. Wow y flutter	Máximo 0,3 %
10. Otros	El motor del grabador no debe variar la velocidad.

* En caso de que el mini-enchufe adjunto con el CE-150 no sea compatible con los enchufes de entrada/salida de su grabador, puede encontrar enchufes especiales de conversión en tiendas de equipos electrónicos.

NOTA:

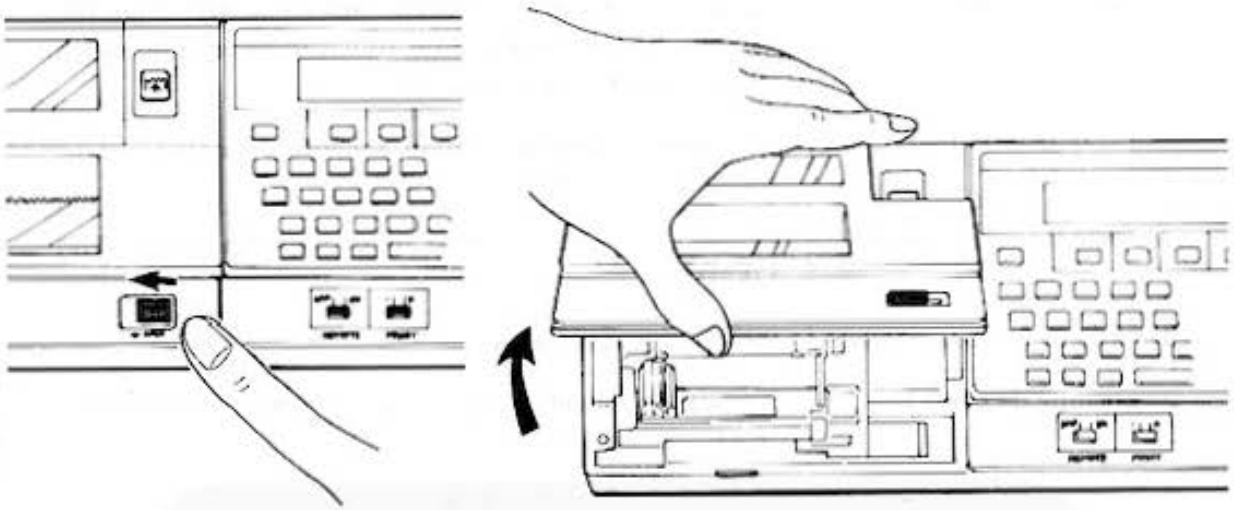
- Algunos grabadores pueden rechazar la conexion debido a diferentes especificaciones. Los grabadores que presenten distorsión, excesivo ruido o potencia disminuida después de varios años de uso, quizás no brinden resultados satisfactorious debido a cambios en sus características eléctricas.

- Instrucciones preventivas para el uso del grabador.
  (1) Para cualquier transferencia o cotejo utilize el grabador empleado para grabación original. Si se cambia de grabador, puede resultar imposible efectuar la transferencia.
  (2) La cabeza del grabador, en caso de estar sucia, incrementa el ruido y disminuye el nivel. Por lo tanto mantengala limpia.
  (3) Utilize cintas magnéticas que tengan buena respuesta en frecuencia y sin ralladuras y plieques.

4. Colocando el papel

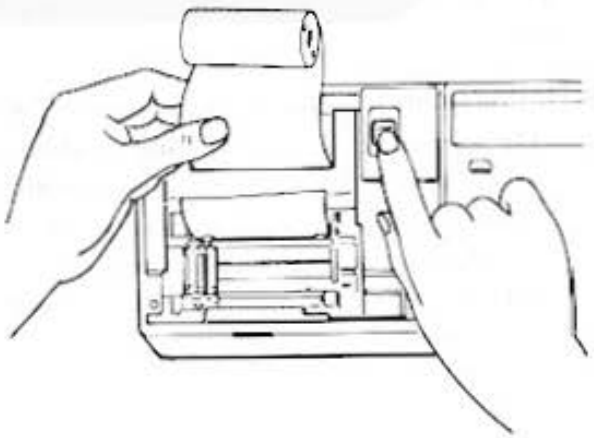
Para más detalles, refiérase al manual de instrucciones de la CE-150.

(1) Para quitar la cubierta del impresor desplace la traba en la dirección de la flecha.

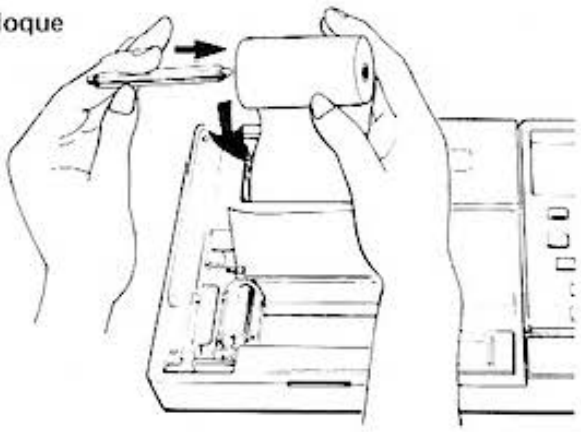


(2) Corte la parte superior del rollo de papel en forma recta, e inserte el papel correctamente en la entrada del papel.

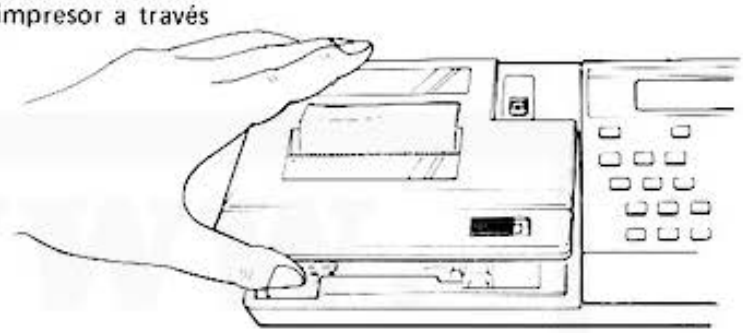
(3) Pulse la tecla [ON] de la computadora para encenderla. Pulse la tecla [↑] para avanzar papel. Ahora, avance más papel hasta que la parte superior del mismo sobresalga de 3 a 5 cm del impresor.



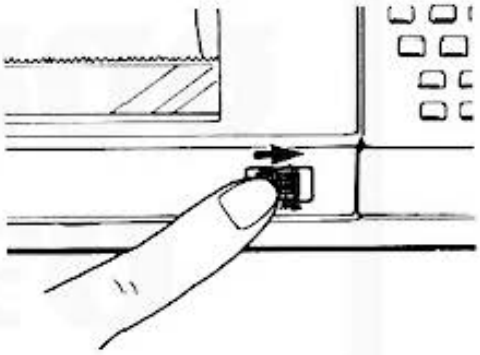
(4) Inserte el eje en el rollo de papel y coloque el papel en el estuche para el papel.



(5) Coloque la cubierta del impresor otra vez en su posición. Ahora, guíe la parte superior del rollo de papel fuera del impresor a través del cortador de papel.

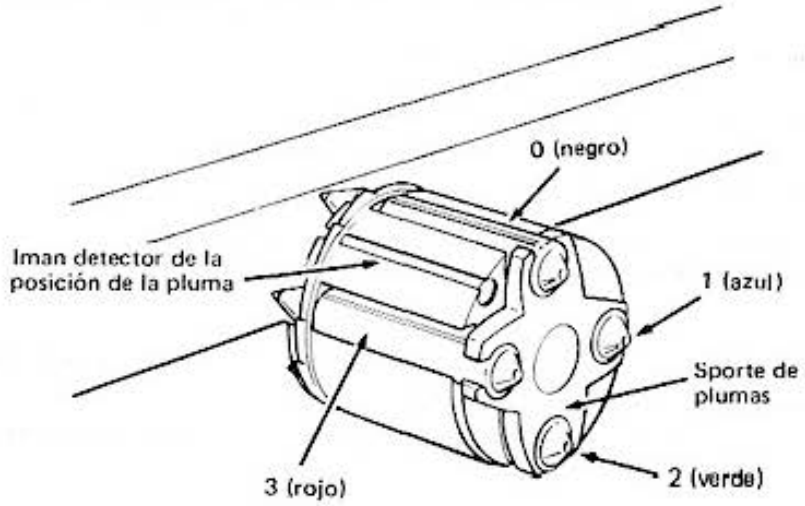


(6) Cierre la cubierta del impresor.



5. Reemplazando las plumas.

Cuatro clases de plumas pueden ser instaladas en esta unidad. Las posiciones para la instalación de las plumas están ilustradas más abajo. Para más detalles, refiérase al manual de instrucciones.



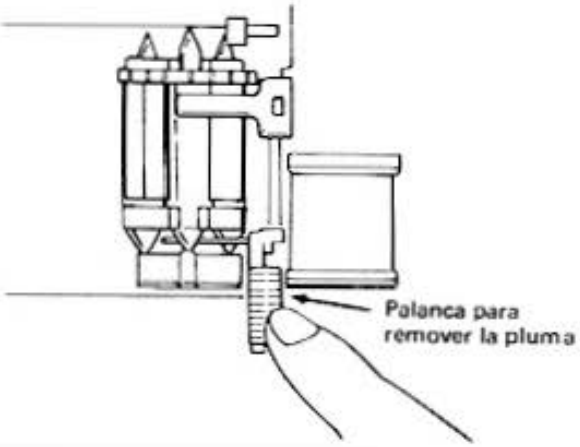
Las posiciones de las plumas están numeradas 0, 1, 2 y 3 en el sentido contrario de un reloj, desde la posición de la pluma del imán detector. Estos números corresponden a las posiciónes seleccionadas por el comando COLOR.

Para instalar, o reemplazar una pluma, siga el procedimiento siguiente:

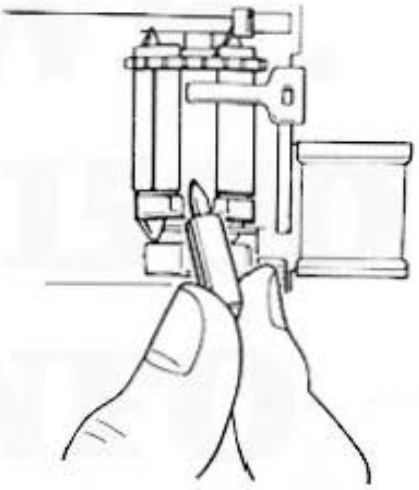
(1) Con la tecla [0] de la computadora pulsada, pulse la [↑] del impresor. Esto permite al impresor que se coloque en el estado de cambio de las plumas cuando el soporte de plumas se desplaza a la izquierda y gira. Con la pluma de arriba cambiada, el soporte de plumas se mueve hacia la derecha. (Suelte la tecla cuando el soporte de plumas empieza a moverse.)

(2) Para quitar la pluma, apriete la palanca de cambio de plumas. Esto causa que le pluma superior sea soltada.

(Nota): Cuando quite la pluma, sosténgala suavemente para prevenir que caiga en el impresor.



(3) Instale una nueva pluma.



(4) Para instalar o quitar la próxima pluma, pulse la tecla [↑] . El soporte de plumas vuelve a la izquierda y gira para que la próxima pluma quede arriba y se desplace a la derecha otra vez. Quite la pluma y reemplácela con una nueva como en los puntos (2) y (3).

(5) Después de reemplazar o instalar la pluma, pulse el impresor [↑] con la tecla de la computadora [CL] pulsada. Esto causa que el impresor sea liberado de su estado de cambio de plumas, y el soporte de plumas vuelve a la izquierda.

(Nota): Para utilizar esta unidad, instale las cuatro plumas en el soporte de plumas. La operación con falta de alguna pluma puede causar fallas en el cambio de colores.

Manejando las plumas:

Las plumas son instaladas en el impresor cuando es utilizado, y quitadas del impresor después de su uso. Tape las plumas y colóquelas en su tubito para almacenamiento.

Al dejar las plumas instaladas en el impresor por bastante tiempo o destapadas, **puede que la tinta se seque.**

B. UTILIZANDO UN GRABADOR

LOS COMANDOS DEL IMPRESOR Y DEL GRABADOR DE CINTA QUE SE DESCRIBEN AQUI ESTAN SOLAMENTE DISPONIBLES EN EL IMPRESOR CE-150 OPCIONAL (CON UN INTERFAZ DE CASSETTE INCORPORADO). DADO QUE LA COMPUTADORA NO ESTA EQUIPADA CON ESTOS COMANDOS, LA PROGRAMACION CON ELLOS ES POSIBLE SOLAMENTE CUANDO ESTEN CONECTADOS AL CE-150. POR LO TANTO, ASEGURARSE DE CONECTAR AL CE-150 PARA PROGRAMAR USANDO ESTOS COMANDOS.

1. Operación del grabador

Recomendamos que siga este proceso utilizando un pequeño programa. Será más fácil re-ejecutar la operación en caso de tener problemas. Escriba el programa ahora.

Usted debe preparar el grabador para una transferencia de programa y datos. Los necesarios para hacer esto son los siguientes:

1. Apague el interruptor remoto en el impresor/interfaz.
2. Ponga una cinta de grabación en el grabador o un cassette (un paso importante).
3. Encuentre una porción libre de la cinta. Si la cinta es nueva, avance más allá de la porción de cinta de arrastre. Si su grabador tiene un contador, marque el número. Esto es muy útil para localizar el programa que usted está guardando. O.K. luego. . .
4. Si su grabador tiene un control de volúmen automático, fíjelo en automático. Si tiene un control de volúmen manual, coloque el nivel del volúmen entre la mitad y el máximo (i.e. 3/4 nivel). Si su grabador tiene un control para el tono, ponga el control del tono en una posición entre la parte intermedia y la alta (posición 3/4).
5. Encienda al interruptor del impresor/interfaz. Si su grabador no tiene un interruptor remoto (esto quiere decir que no hay lugar para conectar uno de los cables), utilice la tecla PAUSE temporalmente para parar la grabación. Si su grabador no tiene un interruptor "pausa", la operación se está poniendo difícil. Le recomendamos que compre un grabador que tenga uno. Esto hará la programación más fácil. Mejor todavía, consiga uno que tenga ambas, un interruptor remoto, y un interruptor para pausas.
6. Pulse las teclas RECORD y PLAY simultáneamente. Si usted está utilizando una máquina sin "pausa", usted querrá hacer esto inmediatamente antes de guardar el programa.

¿Preparado? OK, siga leyendo para encontrar lo mejor. . . . . .

2. El comando CSAVE

Todo preparado, oista a verificar. . . .

1. ¿Grabador preparado y esperando?	sí ____	no ____
2. ¿Cinta en el grabador?	sí ____	no ____
3. ¿Computadura encendida?	sí ____	no ____
4. ¿Programa en la computadora?	sí ____	no ____
5. ¿Cables colocados?	sí ____	no ____
6. ¿Puso el contador a cero?	sí ____	no ____
7. ¿Está usted vestido?	irrelevante	____

Si usted respondió no a cualquiera de estas preguntas, vuelva a leer la sección previa para resolver el problema.

OK, una cosa más y usted estará preparado:

Con el mismo comando que guarda su programa, usted debe dar un nombre de archivo al programa. Esto se hace para utilizarlo como referencia. Su nombre no debe ser mayor que 16 caracteres. Para guardar el programa con un nombre (rótulo) escriba:

CSAVE [SHIFT] ["] PROG1 [SHIFT] ["]

Su programa se guardará con el rótulo "PROG1". Usted puede asignar cualquier nombre que desse; cualquiera que sea fácil de encontrar. También, observe que hay una longitud límite de 16 caracteres para su rótulo. Si el nombre es más largo que 16 caracteres, el exceso es ignorado. Una buena práctica es mantener un cuaderno de programas que incluya el nombre del programa, el punto de partida y de parada de la cinta (utilice los números del contador y una descripción breve de lo que el programa hace.

Pulse la tecla [ENTER]. En este instante usted debe escuchar un sonido muy agudo, y la cinta debe estar girando. También el indicador "busy" (ocupado) debe de estar encendido. Este le indica que la computadora esta ocupada transfiriendo su programa de la memoria a la cinta. Si esto no ocurre empiece otra vez desde el comienzo de la sección. Me tomó sólo 10 intentos para hacerlo bien. Así, es que no se desanime. Si yo lo puedo hacer, usted también lo puede hacer.

Una vez que la computadora llega al final del programa, el indicador "busy" (ocupado) se apagará y el grabador se detendrá. Y el prompt re-aparecerá en la representación visual. Felicitaciones, su primer programa ha sido guardado para uso en el futuro. Para poder asegurar que esto ha sido conseguido, podemos leer otra vez en la memoria de la cinta, tal como se explica en la próxima sección.

3. El comando CLOAD

Ahora que su primer programa está guardado en cinta, usted sin duda querra saber si está allí realmente. Hacer esto es relativamente sencillo.

"¿Qué es lo que hace la computadora?" usted se pregunta. Bien, después de guardar (CSAVE) su programa, simplemente escriba CLOAD?. La computadora compara el programa con el que

tiene en su memoria. Si todo fue tan bien, calmadamente representará gráficamente su nombre en el visor y acabará comprobación. Si todo no fue tan bien, un mensaje ERROR será representado gráficamente, normalmente ERROR 43. Este le dice que el programa en la cinta es diferente del programa en la memoria de la SHARP. Borre esa porción de la cinta y empiece otra vez. Compruebe todas sus conexiones e intente poner más alto el volumen y el tono.

Al usar el comando CLOAD, es necesario que la cinta esté localizada (sincronizada) con el tono portador que precede a los datos grabados. El tono portador consiste en una señal constante y sin modular que tiene aproximadamente 2,5kHz. Si la cinta no está adecuadamente localizada, no se indicará ningún error ni ningún dato aparecerá representado en el visor.

Las siguientes son las instrucciones para cargar un programa:

1. Apague el interruptor "REMOTE" del impresor/interfaz.
2. Coloque la cinta en el lugar que usted empezó utilizando el número del contador otra vez (¿comprende ahora porqué el número del contador es tan útil?).
3. Pare la cinta.
4. Encienda el interruptor "remoto".
5. Pulse el botón play.
6. Escriba:

   CLOAD [SHIFT] ["] PROG1 [SHIFT] ["]

   y pulse la tecla [ENTER] .

   (Recuerde que "PROG1" es el nombre (rótulo) que hemos dado a su programa. Si usted ha guardado el programa bajo otro nombre, usted debe utilizar ese nombre en vez de "PROG1".)
7. El indicador "busy" (ocupado) se encenderá y el programa será retrotraido a la memoria de la computadora para su utilización.
8. Escriba RUN y el programa que usted "CSAVEd" (guardó) previamente re-aparecerá. El cassette contiene una copia del programa de forma que usted puede CLOAD (cargar) el mismo programa varias veces. Mientras carga, cuando aparezca en la representación visual un mensaje de error ERROR 43 o ERROR 44, empiece de nuevo desde el paso anterior (1).

4. Comandos PRINT # y INPUT

PRINT # :

Ahora que que hemos ilustrado el uso de CSAVE y CLOAD, nos gustaria introducir dos comandos similares. El comando PRINT # guarda el valor de una variable o conjunto de variables en cinta. Esto es diferente de CSAVE que graba un programa. Esto le permite utilizar los mismos datos en otrop programa. Por ejemplo, en el programa siguiente, la variable T$ se utiliza así:

```
10: PRINT "COMO SE LLAMA?"
20: INPUT T$
30: PRINT T$
```

Si usted quiere guardar el valor de T$ para ser utilizado en otro programa, puede emitir un comando PRINT # para guardarlo en la cinta. Usted puede hacer esto de dos formas:

1. Manualmente
2. A través de un Programa

Nota: Esta operación requiere el uso del grabador. Así es que prepare el grabador para recibir datos. Si usted no está seguro de cómo hacerlo, vuelva a la sección previa.

1. El Método Manual

El método manual ofrece varias opciones:

OPCION 1:

Después de ejecutar un programa, cambie al modo RUN y escriba:

[P] [R] [I] [N] [T] [SHIFT] [#] A, B, C (y después pulse [ENTER])

El grabador comenzará ahora su acción y guardará el valor de todas las variables en cinta.

OPCION 2:

Si usted quiere identificar sólo ciertas variables para ser guardadas escriba lo siguiente:

[P] [R] [I] [N] [T] [SHIFT] [#] "nombre de archivo" ; A, B, C.

En este momento usted ha especificado las variables A, B y C como las únicas que deben ser guardadas en la cinta bajo ese rótulo.

OPCION 3:

Usted puede especificar todos los valores de variables relacionadas escribiendo:

[P] [R] [I] [N] [T] [SHIFT] [#] "nombre de archivo" ; B (*)

El símbolo * guardará todas las variables de "B", incluyendo B(1) y B misma.

2. A Través de un Programa

Para hacer esto, simplemente asigne una línea a la orden PRINT # en su programa. Usted puede utilizar cualquiera de los formatos descritos arriba. Cuando la SHARP encuentra esta línea automáticamente activará al grabador y empezará a transferir los datos a la cinta. Una vez más el experimento funcionó. Si la cosa no funciona, vuelva a las secciones previas. Quizás algo muy simple no se ha estudiado bien.

INPUT # :

Este comando permite los mismos formatos que PRINT #. La diferencia es que INPUT # transfiere datos de la cinta a la computadora. (PRINT # transfiere datos de la computadora a la cinta). Usted puede utilizar la orden INPUT # manualmente o como parte de un programa. Acuérdese de preparar el grabador antes que usted empiece su comando.

NOTA: Si usted está especificando más variables (como parte de su comando INPUT #) que existen en la cinta, a las variables que quedan será asignado el valor 0. Si usted está especificando menos variables (como parte de su INPUT #) de las que existen en la cinta, las variables que quedan serán ignoradas.

5. El comando MERGE

El comando MERGE le permite almacenar numerosos programas al mismo tiempo en la memoria de la computadora. Por ejemplo, vamos a suponer que la memoria de la computadora contiene el siguiente programa:

```
10: PRINT "DEPRECIACION N PERMISIBLE
20: INPUT "Pulse su método:  ": A
```

En este momento usted se acuerda que puede que tenga una parte similar de un programa en la cinta bajo el nombre de archivo "DEP1". Desde luego que querrá ver si este programa tiene partes útiles para el programa que está construyendo actualmente. El primer paso a realizar el buscar la cinta que tenga "DEP1". Después localice el lugar de la cinta donde comienza "DEP1".

Escribe ahora: MERGE "DEP1" y pulse [ENTER]

La computadora cargará ahora "DEP1" en la memoria JUNTO con el programa anterior. Una vez que se ha cargado "DEP1", puede que encuentre en la memoria algo similar a esto:

```
10: PRINT "DEPRECIACION PERMISIBLE"
20: INPUT "Pulse su método:  ": A
10: "DEP1": REM >>segundo módulo<<
20: PRINT "CARGOS DE INTERES"
30: INPUT "Cantidad prestada: ": B
  :
(etc.)
```

Observe que a diferencia del comando CLOAD, el nuevo programa no reemplaza al que ya existía, y que se han duplicado algunos números de líneas. Note también que se ha usado un "rótulo" en la primera línea del módulo que se ha unido. (Ver más abajo: JUNTANDO MODULOS UNIDOS)

Es importante que revise la siguiente información antes de continuar más adelante editando o programando.

NOTAS IMPORTANTES

Una vez que se ha efectuado un MERGE, no se permiten INSERCIONES, SUPRESIONES o CAMBIOS en las líneas de programas ya existentes.

Ejemplo:

```
10 "A" REM Este es el programa existente
20 FOR T=1 TO 100
30 LPRINT T
40 NEXT T
   (Etc.)
    :
```

Antes de hacer un MERGE del próximo programa, haga todos los cambios que considere necesarios en este programa.

Luego, una le siguiente programa; MERGE "PROG2" (ejemplo)

```
10 "B" REM Este es el programa MERGED (unido)
20 INPUT "Pulse depreciación:   "; D
30 INPUT "Número de años:       "; Y
40    (Etc.)
       :
```

Ahora pueden hacerse cambios al programa anterior dato que fue la ultima parte unida. Si necesita hacer más cambios en el primer programa, sírvase seguir el siguiente procedimiento:

1. Guarde (CSAVE) lo que haya hecho hasta este momento; use un nuevo nombre cuando guarde en una cinta, por ej. "PROG3" (Ejemplo).
2. Cargue (CLOAD) en la memoria el programa que ha guardado en el paso 1.
3. Haga ahora cualesquiera cambios o modificaciones que necesite. Sin embargo, tenga en cuenta que los cambios pueden hacerse sólo en el primer programa o en el primer caso de cualesquiera números de líneas duplicados. Por ejemplo, si la línea de programa 10 aparece en el primer programa así como en el segundo, los cambios afectarán solamente al primer caso de la línea 10. De forma que si intenta editar el segundo caso de la línea 10, este cambio se reflejará erróneamente en la primera parte del programa solamente.

AÑADIENDO LINEAS DE PROGRAMAS ADICIONALES

Se pueden añadir líneas adicionales al final de los programas existentes solamente si tienen números de línea mayores que aquellos previamente utilizados.
Obsérvese que las líneas de programas adicionales no tienen por qué aparecer necesariamente en el "fondo" del listado. Esto no tiene por qué preocuparle dado que los modulos deben juntarse a través de rótulos.

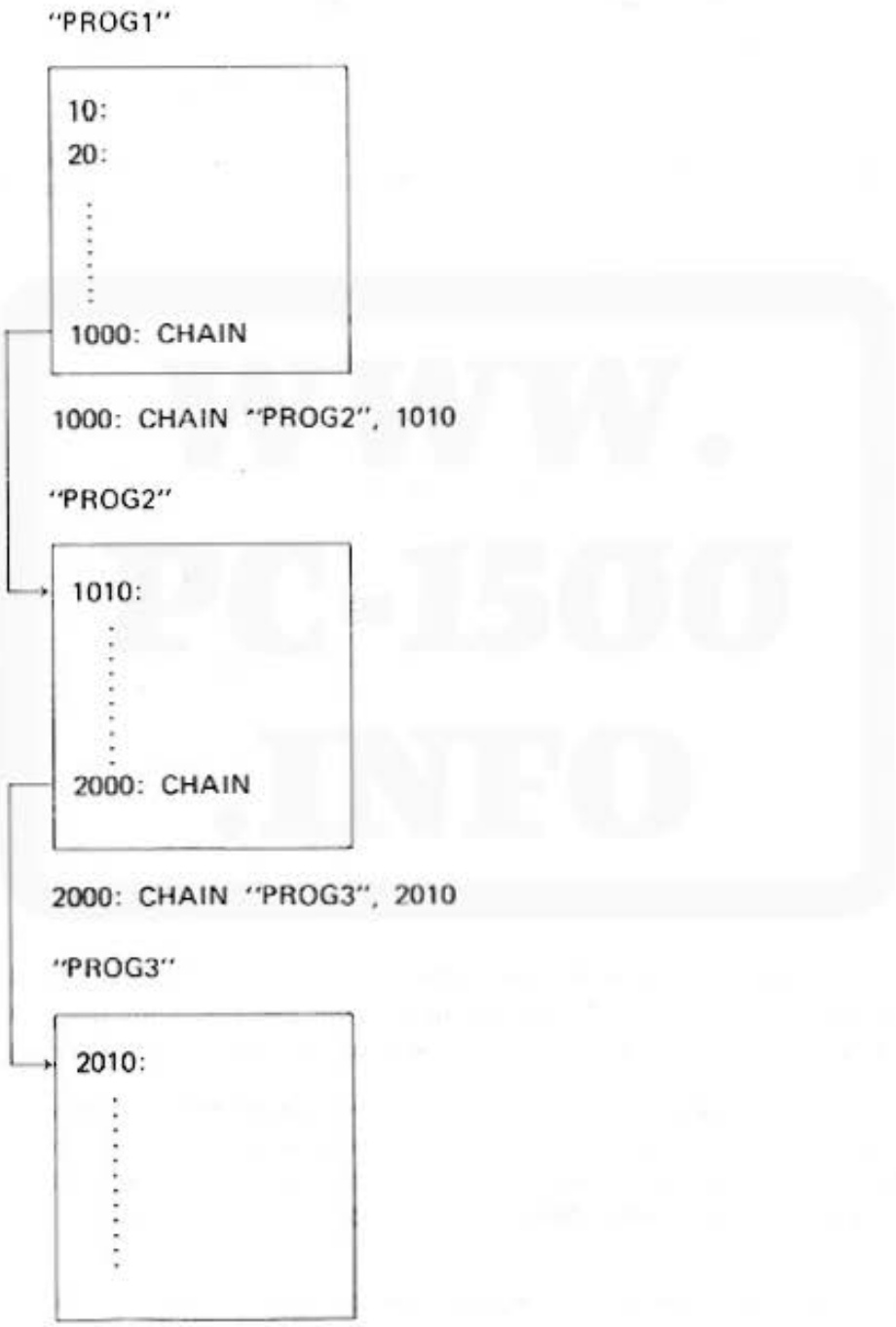
JUNTANDO MODULOS UNIDOS (programas)

Dado que el procesador ejecuta las líneas de su programa en una secuencia lógica, se detendrá cuando encuentre una pausa en la numeración de la secuencia en línea, por ej., si los números de línea 10, 20, 30 van seguidos por números de línea duplicados en un segundo módulo, el procesador detendrá su ejecución después de la línea 30 del primer módulo.
Para juntar varios módulos, son válidas las siguientes técnicas: GOTO "B", GOSUB "B", IF . . . THEN "B" (B se usa por ejemplo sola; puede usar cualquier rótulo excepto palabras o letras de reserva que aparecen en la fila #3 del teclado, por ej., de Q a P).

6. La Sentencia CHAIN (cadena)

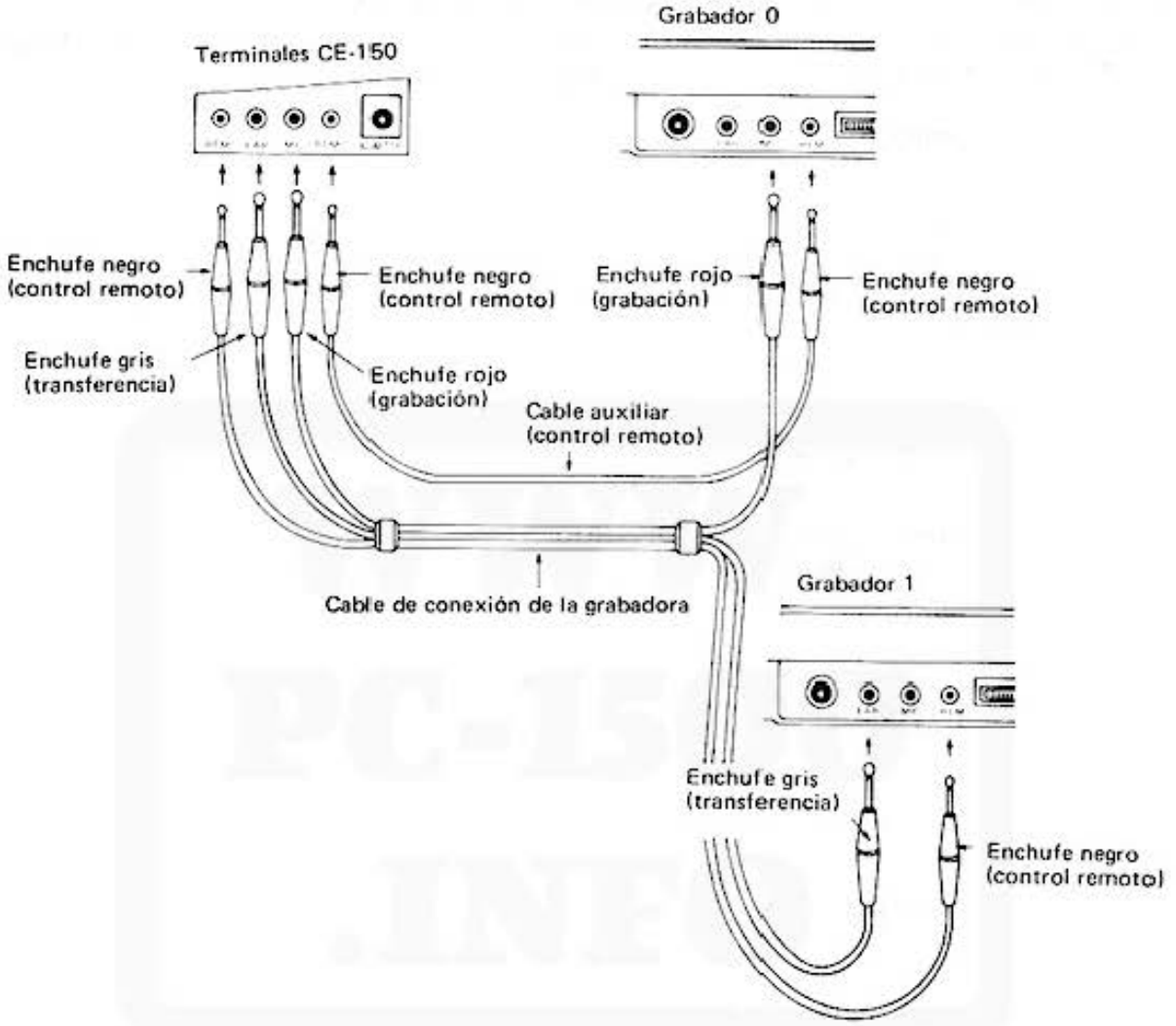
CHAIN es una instrucción de programa; sólo puede ser utilizada dentro de un programma pero no puede ser utilizada manualmente como CSAVE, CLOAD y MERGE. La sentencia CHAIN le permite ejecutar un programa que es demasiado largo pasa caber en la memoria al mismo tiempo. Estos programas largos deben ser divididos en secciones con una sentencia CHAIN al final de cada sección. Estas secciones pueden ser guardadas en cinta con CSAVE.

Por ejemplo, vamos a asumir que usted tiene tres secciones en el programa, llamadas PROG1, PROG2, PROG3. Cada una de estas secciones acaba con una sentencia CHAIN.



Durante la ejecución, cuando la computadora encuentra la sentencia CHAIN, la próxima sección es traída de la cinta a la memoria y ejecutada. De esta forma todas las secciones son eventualmente ejecutadas.

7. Utilizando dos grabadores



Cuando utilice dos grabadores, uno de ellos es utilizado para grabar y el otro para transferir partes grabadas. Como está ilustrado, el impresor/interfaz de cassette y los dos grabadores están conectados utilizando los cables de conexión y el cable auxiliar par telecontrol remoto.

- La CE-150 está equipada con dos terminales de control remoto REM 0 y REM 1; cualquiera de los dos puede ser utilizado manualmente. En la operación por programa (no manual), el programa designa el grabador conectado a la terminal REM 0 y a la terminal REM 1. Entonces, estas terminales de control remoto deben ser conectadas según el programa.

En esta sección se explica cómo operar el segundo grabador conectado al terminal REM 1.

1. Procedimientos para grabar un programa.

(1) Escriba RMT OFF y pulse [ENTER] para desligar la función del segundo control remoto. (Control del grabador 1 en la ilustración de arriba).

(2) Ponga una cinta en el grabador.

(3) Escriba RMT ON y pulse [ENTER] para accionar la función del segundo control remoto.

(4) Fije los controles del volúmen y tono de la misma forma que se ha explicado previamente para un grabador único.

(5) Oprima las teclas RECORD y PLAY simultáneamente.

(6) Ejecute la grabación.

Programa: **CSAVE-1 "nombre de archivo"** [ENTER]

Datos: **PRINT #-1, "nombre de archivo" ; variable, variable, ...**

(Ejemplo) Designe los modo PRO y RUN.
CSAVE-1 "PR-1" [ENTER]

Después de grabar el programa, el "prompt" reaparecerá en la pantalla y la cinta se para. Vuelva la cinta hacia atrás para verificación.

2. Procedimientos de comparación entre los contenidos de la cinta y los de la computadora.

(1) Escriba RMT OFF [ENTER] para quitar las funciones del control remoto.

(2) Vuelva la cinta hacia atrás al lugar que usted empezó, otra vez utilizando el número del contador.

(3) Pulse RMT ON para accionar las funciones de control remoto.

(4) Fije los controles para el volumen y el tono de la misma forma que explicamos previamente con un grabador único.

(5) Pulse la tecla PLAY.

(6) Ejecute la comparación
CLOAD?-1 " nombre de archivo" [ENTER]
(Ejemplo) Designe el modo PRO o RUN. COLAD? "PR-1" [ENTER]
La ejecución acaba cuando ambos contenidos se igualan, resultando en la representación gráfica de un "prompt".

3. Procedimiento de transferencia de cinta grabada.

(1) Pulse RMT OFF y la tecla [ENTER] para quitar las funciones de control remoto.

(2) Ponga una cinta grabada en el grabador.

(3) Escriba RMT ON y pulse [ENTER] para accionar las funciones de control remoto.

(4) Fije los controles de volúmen y tono de la misma forma que ha sido explicado previamente para la grabador único.

(5) Pulse el botón PLAY.

(6) Ejecute la "TRANSFERENCIA".
Programa: **CLOAD-1 "nombre de archivo"** [ENTER]
INPUT #-1, "nombre de archivo" ; variable, variable . . . . . [ENTER]
(Ejemplo) Designe el modo PRO o RUN.
CLOAD-1 "PR-1" [ENTER]
Después de la transferencia, el "prompt" reaparece en la pantalla.

C. UTILIZANDO EL IMPRESOR

NOTA: Todas las explicaciones y ejemplos en esta sección asumen que usted ya ha:

1) Conectado apropiadamente la computadora PC-1500 al opcional CE-150.
2) Cargado las baterías de la CE-150 mediante un adaptador eléctrico EA-150.
3) Colocado las plumas en el impresor.
4) Cargado el papel en el impresor.

Si no ha hecho estas cosas, vuelva a la sección 1 de este Capítulo de instruccions.

C.1. CE-150 Especificaciones del Impresor

Caracteres por línea:	4, 5, 6, 7, 9, 12, 18, o 36 dependiendo del tamaño del carácter escogido.
Tamaño del carácter:	1,2 x 0,8 mm a 10,8 x 7,2 mm dependiendo del tamaño del carácter escogido.
Velocidad de Impresión:	Máximo 11 caracteres por segundo cuando imprime los caracteres más pequeños.
Rotación:	Los caracteres pueden ser impresos en cualquiera de las dos direcciones de los dos ejes.
Colores:	4 – Rojo, azul, verde, negro
Sistema Gráfico:	Tramado en ejes X–Y
Avance de papel.:	Manual o programable.

C.2. El comando TEST

Lo primero que usted quiere hacer es probar el funcionamiento del impresor CE-150. Con la computadora y el interfaz encendido, escriba:

TEST y pulse [ENTER]

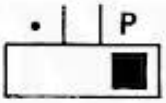
El impresor ahora dibujará 4 cajas, cada una de color diferente. El color de las cajas, de izquierda a derecha, es lo que usted escogión como colores 0 a 3 cuando usted insertó las plumas para escribir.

C.3. Imprimiendo Cálculos

Utilizando el interfaz CE-150 es posible hacer una copia impresa de una serie de cálculos manuales ejecutados en la PC-1500 como en la figura 1:

```
12000*.065
                    780

780+25.56
                  805.56

SIN 30
                      0.5

TX=.065
                   0.065

P=20000*TX
                    1300

P/12
            108.3333333
```

Para hacer esto, simplemente fije el interruptor Print en el impresor/interfaz en la posición P:



Para prevenir la impresión automática de cálculos manuales, el interruptor Print debe estar colocado en la posición " . ". Con el interruptor en esta posición, usted puede imprimir resultados seleccionados prolongando la computación con una instrucción LPRINT. (vea LPRINT). Otras instrucciones que causan impresión o dibujo, como LLIST, LINE, y otras, son también funcionales en este modo.

Cuando está imprimiendo, el color utilizado será el color que fue especificado previamente. Si usted acaba de encender la máquina, ésta será del color de la pluma que usted ha seleccionado para corresponder al color 0. Para cambiar el color, usted debe emitir la instrucción COLOR (vea sección apropiada).

El tamaño del carácter utilizado para imprimir cálculos manuales depende de la especificación previa. Si el tamaño del carácter previo especificado fue tamaño 1 ó tamaño 2, entonces este tamaño queda vigente. Si el tamaño previo fue mayor que 2, entonces se utiliza el tamaño 2.

La impresión automática causa al modo impresor que sea fijado en TEXT. Si usted estaba en el modo GRAPH y desea volver a este modo, usted debe emitir la instrucción GRAPH. (Los modos del impresor son explicados en la sección siguiente).

C.4. Modos del Impresor

Cuando el impresor está operando, puede estar en uno de estos dos modos: TEXT o GRAPH. Estos modos corresponden toscamente a la relación de la escritura humana con el dibujo humano. Como la mayoría de las sentencias funcionan sólo en un modo u otro, es importante seleccionar el modo propio antes de emitir instrucciones.

El modo TEXT es utilizado para imprimir números y caracteres. El ancho del papel del impresor está dividido en dos columnas, el número de las cuales está relacionado al tamaño específico de los caracteres. Dos instrucciones de tabulación vertical y horizontal son provistas para el formato de la información de texto.

En el modo GRAPH, una variedad de figuras diversas, gráficos y tablas pueden ser creadas. Instrucciones para dibujar líneas sólidas y quebradas utilizando un sistema de coordenadas directo o relativo, son disponibles. Todos los dibujos utilizan un esquema normal de coordinadas X–Y.

Para especificar el modo TEXT la sentencia:

```
TEXT
```

es suficiente. Ciertas instrucciones (discutidas más tarde) causan un cambio automático al modo TEXT.

Especificar el modo GRAPH es igualmente sencillo. La sentencia:

```
GRAPH
```

iniciará este modo, fijando la pluma en el lado izquierdo del papel.

C.5. Impresión de Programas

El comando LLIST causa la impresión, total o parcial, de programas. La orden LLIST es extremamente útil durante el proceso de desarrollo de programas.

La forma del comando LLIST es similar a la forma del comando LLIST. Debido a que LLIST es más versátil, hay diferencias sutiles. Las formas LLIST son así:

LLIST
- Imprime todas las líneas de los programas existentes en la memoria de programa.

LLIST expresión (número de línea)
- Imprime sólo la línea del programa cuyo número de línea está dado por la expresión.

LLIST , expresión (número de línea)
- Imprime todas las líneas del programa hasta, e incluyendo, la línea cuyo número está dado por la expresión.

LLIST expresión, (número de línea)
- Imprime líneas de programa empezando con la línea cuyo número está dado por la expresión.

LLIST expresión 1, expresión 2
- Imprime líneas de programa empezando con la línea cuyo número es dado por la expresión 1 y acabando con la línea cuyo número es dado por la expresión 2. Así,

si el comando es:

LLIST 100, 150

entonces todas las líneas entre 100 y 150 (si hay algunas) serán listadas.

LLIST "rótulo"

– Imprime la línea de programa que contiene el rótulo dado.

LLIST "rótulo",

– Imprime las líneas de programa empezando con la línea que contiene un rótulo dado y continúa hasta el final.

NOTA: La especificación de un rótulo que no existe será señalada por un mensaje ERROR 11.

Cuando imprima un programa, el color utilizado será el color que fue especificado previamente. Si usted acaba de encender la máquina, este será el color de la pluma que usted seleccionó correspondiente al color 0. Para cambiar el color, usted debe emitir la instrucción COLOR (vea sección apropiada).

El tamaño del carácter utilizado para listar un programa es dependiente de especificaciones previas. Si el tamaño de carácter previo especificado fue tamaño 1 o tamaño 2, entonces este tamaño queda vigente. Si el tamaño previo fue mayor que 2, entonces el tamaño 2 es utilizado.

La orden LLIST causa que el modo impresor sea fijado en TEXT. Si usted estuviera en el modo GRAPH y desea volver a este modo, usted debe emitir la instrucción GRAPH.

Mientras está listando un programa, la computadora PC-1500 intenta ajustar las líneas del programa para una mejor lectura. Esto se hace dejando espacios en las líneas numeradas. Las líneas numeradas que tienen 1 ó 3 dígitos serán "justificadas a la derecha" dentro de un campo de 3 caracteres. Los números de las líneas que tienen 4 o 5 dígitos serán impresos en un campo de 5 caracteres:

```
   10: REM ANCHURA 3
   20: REM      "
  300: REM      "
 2001: REM ANCHURA 5
 2010: REM      "
```

C.6. Control Programable del Impresor

C.6.1. CSIZE

La instrucción CSIZE especifica el tamaño de los caracteres para toda impresión subsecuente. Hay nueve tamaños disponibles fluctuando de 36 caracteres por renglón impreso a 4 caracteres por renglón impreso. La forma de la instrucción es:

CSIZE expresión
(cualquier modo)

La expresión debe evaluar hasta un número en el rango de 1 a 9. El ancho y la altura de los caracteres para cada tamaño están dados en la siguiente tabla.:

Tabla 1:

CSIZE		1	2	3	4	5	6	7	8	9
Caracteres por cada línea.		36	18	12	9	7	6	5	4	4
Altura de cada carácter	(mm)	1,2	2,4	3,6	4,8	6,0	7,2	8,4	9,6	10,8
Ancho de cada carácter	(mm)	0,8	1,6	2,4	3,2	4,0	4,8	5,6	6,4	7,2

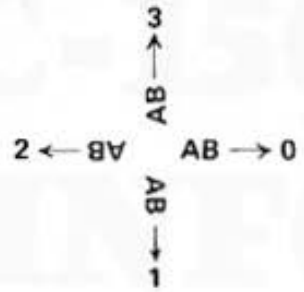
C.6.2. ROTATE

La sentencia ROTATE es utilizada, sólo en el modo GRAPH, para especificar la dirección en que la impresión se realiza. Cuatro direcciones son posibles: arriba, abajo, izquierda a derecha, y derecha a izquierda (con las letras boca abajo). Las direcciones están ilustradas en la figura 1. La forma de la sentencia ROTATE es:

ROTATE expresión
(sólo en el modo GRAPH)

La expresión debe evaluar hasta un número entero en el rango de 0 a 3. ROTATE 0 designa la manera normal de imprimir caracteres de izquierda a derecha.

Figura1:



C.6.3. COLOR

La instrucción COLOR permite la especificación de la pluma que va a ser utilizada para toda impresión y dibujo subsecuente. Si cada posición de la pluma contiene un color de pluma diferente, entonces la instrucción COLOR puede usarse para cambiar colores. La forma de la instrucción COLOR es:

COLOR expresión
(cualquier modo)

La expresión debe evaluar un número entero en el rango de 0 a 3. Cada número entero corresponde a una pluma diferente. El color representado por el número entero variará dependiendo del orden en que las plumas fueron cargadas en el portador. La correspondencia puede averiguarse con la orden TEST (descrita arriba).

Números que no son números enteros pero que están en el margen de 0 a 3 serán cambiados a números enteros. Todos los demás números causarán un ERROR 19.

Cuando la computadora PC-1500 es apagada y después encendida otra vez, la pluma que corresponde a cero es seleccionada.

En el modo TEXT, la ejecución de la instrucción COLOR causará que la posición de la pluma sea fijada a la izquierda del papel. En el modo GRAPH, la pluma volverá a su posición previa.

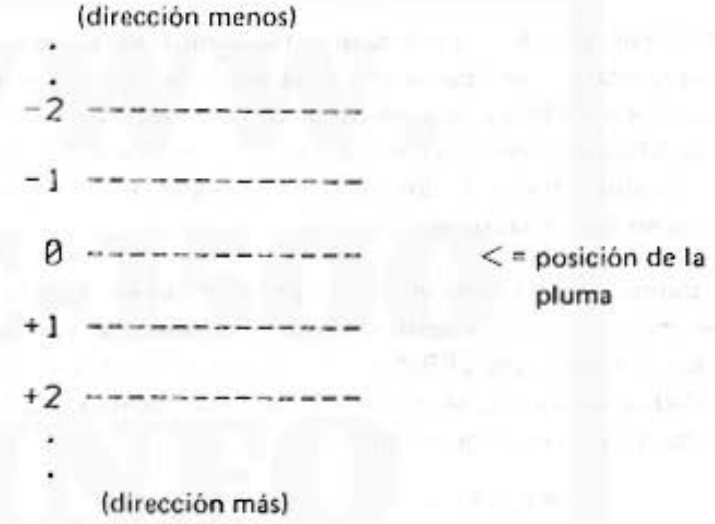
C.6.4. LF (Avanzando el papel)

La instrucción LF causa que el papel en el impresor se mueva hacia adelante o hacia atrás. La forma de la instrucción es:

LF expresión
(sólo en el modo TEXT)

Si la expresión evalúa a un número positivo, el papel es avanzado el número de líneas especificado por la expresión. Una expresión que evalúa a un número negativo causará que el papel retroceda el número de líneas especificado por la expresión. Esto se ilustra en la Figura 2:

Figura 2:



La distancia real que el papel se mueve está relacionada al tamaño del carácter vigente cuando la instrucción LF se especifica.

Cuando el papel se está moviéndose en la dirección reversa (i. e. se tira hacia atrás) un contador interno previene que no sobrepase más de 10,24 centímentros (alrededor de 4 pulgadas).

NOTA: No intente insertar el papel mientras el mecanismo de avance del papel está en opéración. Esto puede dañar el impresor.

C.6.5. LPRINT

La instrucción LPRINT es la instrucción más importante para imprimir información de textos. Es similar a la instrucción PRINT de la PC-1500, y ambas comparten varias de las mismas formas. Sin embargo, debido a las formas adicionales disponibles en el impresor, las acciones de la instrucción LPRINT son más complejas. Nosotros nos concentraremos en las utilidades encontradas al utilizar la sentencia LPRINT.

La siguiente discusión de la sentencia LPRINT asume sólo el modo TEXT. Aunque las formas

dadas pueden funcionar en el modo GRAPH, su operación será diferente.

La impresión de un item único queda generalmente igual.

```
LPRINT item
```

en el cual el item es una expresión, serie de caracteres, número o nombre de la variable cuyos contenidos serán impresos. Como es usual, los caracteres serán colocados a la derecha.

Como el cursor en la representación visual, si la pluma no está colocada a la izquierda del papel, la impresión empezará desde la posición de la pluma. La posición de la pluma puede ser cambiada por la sentencia LCURSOR o por la cláusula TAB (vea más abajo).

Un problema ocurre cuando uno intenta imprimir un item que es demasiado grande para caber en un renglón impreso debido al tamaño corriente del carácter. Si el item largo es un número, un ERROR 76 resultará. Si el item largo es una serie de caracteres, la serie será continuada en la línea próxima.

El tamaño de un item impreso es también importante para la sentencia LPRINT con dos items, cuya forma es:

```
LPRINT it 1, it 2
```

Utilizando CSIZE 1 grantiza que dos items numéricos, serán impresos en la misma línea. En este caso, estarán justificados en la manera usual dentro de dos mitades de la línea impresa. Para una sentencia LPRINT que envuelva varias series esto no es tan simple. Si ambos items caben, serán justificados como usual e impresos en la misma línea. Si no caben, los resultados son generalmente divididos a través de dos líneas. Para caracteres de tamaños más grandes, los dos items son impresos en dos líneas sucesivas.

El punto y coma también puede ser utilizado en la sentencia LPRINT. Esto sirve tanto para indicar mínimo espaciamiento de los items como, al final de una sentencia, para agrupar items impresos por sucesivas LPRINT. En cualquier caso, si la longitud total de los items excede la capacidad de una línea, se imprimirá en líneas sucesivas (tantas como sea necesario). La LPRINT con el punto y coma tiene la forma:

```
LPRINT it 1 ; it 2 ; . . . (etc)
```

o la forma:

```
LPRINT it-lista;
```

Varias de las formas mencionadas son utilizadas en el siguiente programa de demostración:

```
10 A$ = "ABCDEFG"
20 B = 123456
30 FOR I = 1 TO 3
40 CSIZE I
50 LPRINT A$
60 LPRINT A$, B
70 LPRINT A$ ; B
80 LF 5
90 NEXT I
```

Que produce la salida:

```
ABCDEFG
ABCDEFG
ABCDEFG 123456
```

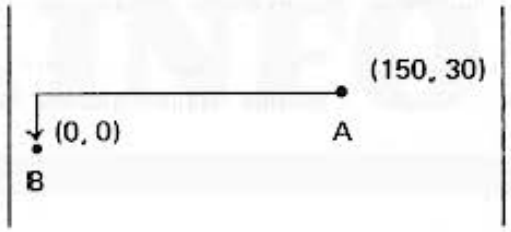
```
ABCDEFG
ABCDEFG
                123456
ABCDEFG 123456
```

```
ABCDEFG
ABCDEFG
          123456
ABCDEFG 1234
56
```

La instrucción LPRINT puede ser utilizado sin una especificación de item, en cualquier modo TEXT o GRAPH:

```
LPRINT
```

Utilizado de esta forma, causará un retorno del portaplumas y el avance de una línea de papel. **Sin embargo no restaura los contadores del modo GRAPH**. Así, en el ejemplo siguiente, aunque la sentencia LPRINT ha sido utilizada para mover la pluma del punto A al punto B, el impresor cree que está en las coordenadas (150, 30) y ejecutará todos los comandos subsecuentes como si lo fuera.



La sentencia LPRINT también incorpora una cláusula USING que opera en la misma manera como lo hace en la sentencia PRINT. La cláusula USING puede ocurrir sólo en una sentencia LPRINT que sea ejecutada en el modo GRAPH.

C.6.6. LCURSOR

La sentencia LCURSOR permite la colocación de la pluma de una manera análoga a la sentencia CURSOR de la representación visual. La forma de la sentencia LCURSOR es:

```
LCURSOR posición
(sólo en el modo TEXT)
```

La posición del carácter al cual la pluma puede ser movida es, por supuesto, dependiente del tamaño del carácter usado. En general, la pluma puede ser colocada un espacio menos que el máximo para el tamaño del carácter. Para una lista de los anchos de líneas de impresión para cada tamaño de carácter, refiérase a la instrucción CSIZE en esta sección.

C.6.7 TAB

La sentencia TAB es idéntica a la sentencia LCURSOR, excepto que puede ser utilizada dentro de una sentencia LPRINT. Este tipo de sentencia LPRINT tiene la forma:

LPRINT TAB posición; lista de items

Los mismos comentarios que se aplican a la expresión de la posición del LCURSOR se aplican a TAB. Si la lista de elementos está vacia, el resultado neto de la instrucción registrada arriba será es paciar una línea el papel.

C.6.8. SORGN (Fijar ORIGEN)

La sentencia SORGN es utilizada para establecer el origen del sistema de coordenadas X–Y para comandos gráficos subsecuentes. La sentencia SORGN hace que la actual posición de la pluma se considere el origen. Así, esta instrucción es dada directamente detrás de una instrucción que mueva la pluma a un punto dado en el papel. La forma de la sentencia SORGN es sencillamente:

SORGN
(sólo en el modo GRAPH)

NOTA: El impresor CE-150 permite que una posición de la pluma sea especificada fuera del margen de las posiciones dibujables. En este caso, la pluma se mueve tan lejos como puede y después se para. Si la pluma se mueve dentro de este dominio imaginario y después la instrucción SORGN se emite, la impresión subsecuente o las sentencias dibujadas no tendrán ningún efecto. Esto aparecerá como si el programa tuviera un error o como si el interfaz estuviese dañado.

El siguiente programa fija el origen en 100 unidades hacia arriba y 100 unidades a la derecha de la posición corriente de la pluma y después dibuja una caja de 10 unidades con una de las esquinas de las cajas en el nuevo origen.

```
10 GRAPH
20 LINE (0, 0) – (100, 100), 9
30 SORGN
40 LINE (0, 0) – (10, 10), 0, 0, B
50 TEXT
60 END
```

C.6.9. GLCURSOR

La sentencia GLCURSOR mueve la pluma a cualquier coordenada X–Y sin dibujar una línea. La forma de la sentencia GLCURSOR es:

GLCURSOR (expresión 1, expresión 2)

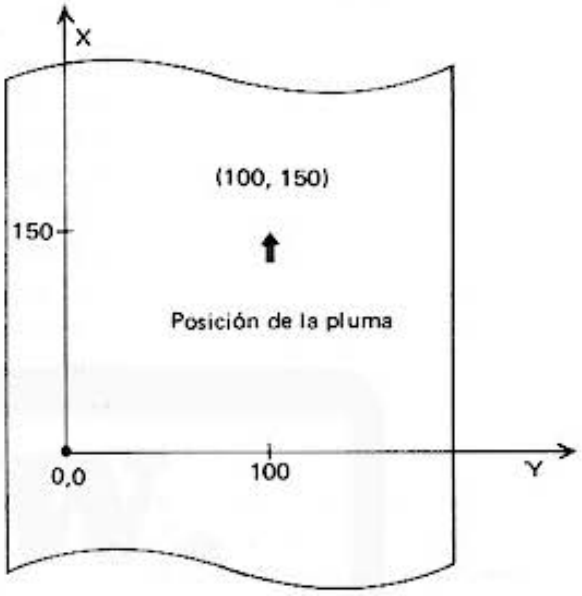
Ambas expresiones deben evaluar a un número en el rango de –2048 a +2047. La expresión 1

representa la distancia "X" al punto de destino y la expresión 2 representa la distancia "Y" al punto de la destino.

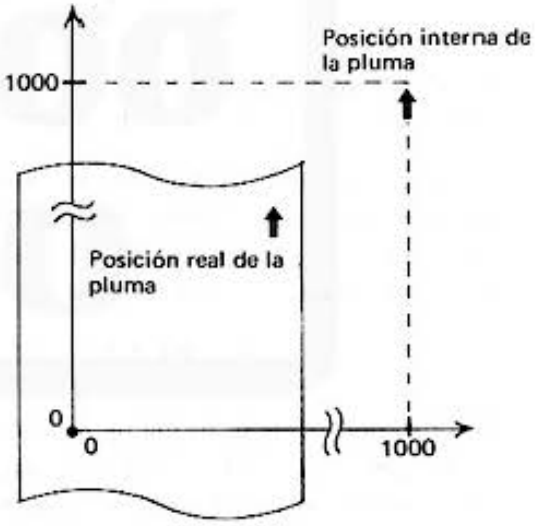
NOTA: Si el punto de destino está fuera del rango de puntos al cual la pluma puede moverse en realidad, la pluma se parará a un lado del papel. Internamente, los contadores que controlan la pluma están contando todavía hacia su meta.

Ejemplo de utilización de la sentencia GLCURSOR:

En el ejemplo a la derecha la pluma está colocada en la posición (100, 150).



En este ejemplo, la pluma está colocada incorrectamente en la región imaginaria de la posición (1000, 1000). En realidad la pluma se mueve hacia la derecha del margen y enrolla el papel hacia atrás. Al alcanzar el lado derecho continúa hacia arriba hasta que alcanza el límite permitido de 10 cm, y en ese momento se detiene.



C.6.10. LINE

La sentencia LINE es la primera sentencia del modo GRAPH. Especifica el movimiento de la pluma de un punto a otro. Si la pluma está colocada hacia abajo, una línea es dibujada. La sentencia LINE también permite líneas quebradas para ser dibujadas con ocho longitudes de rayas diferentes. La primera forma de la sentencia es:

LINE (X1, Y1) – (X2, Y2), línea-tipo, color

El punto inicial de la línea está determinado por los valores de las expresiones X1 e Y1. El punto de destino de la línea está determinado por los valores de la expresión X2 e Y2. Ambos valores de X y de Y deben estar en el margen entre –2048 a +2047. Una especificación de valor que exceda este rango producirá un error.

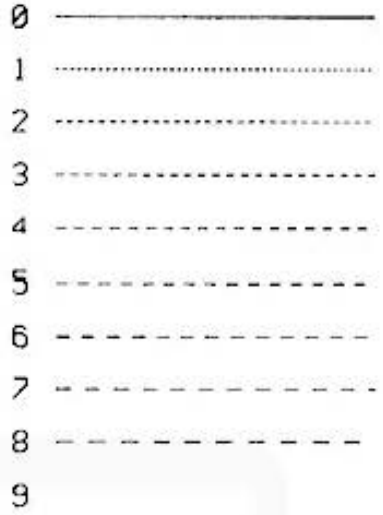
El tipo de línea, y los parámetros del color son opcionales. Si son omitidos, los valores utilizados serán los valores que estabán en efecto antes de la sentencia. El color es, claro está, uno de

los colores en el rango de 0 a 3.

El tipo de línea debe ser una expresión que evalua a un número en el margen de 0 a 9. La tabla 2 identifica el significado de cada opción:

Tabla 2:

Linea-tipo Valor	Resultado Tamaño de línea
0	Linea Sólida
1	0,4 mm raya
2	0,6 mm raya
3	0,8 mm raya
4	1,0 mm raya
5	1,2 mm raya
6	1,4 mm raya
7	1,6 mm raya
8	1,8 mm raya
9	Pluma hacia arriba (sin línea)



Utilizada de otra forma, la sentencia LINE interpreta las especificaciones de puntos como puntos finales de una diagonal. Después proseguirá a dibujar la caja representada por la diagonal. La forma de esta sentencia LINE es:

LINE (X1, Y1) – (X2, Y2), línea-tipo, color, B

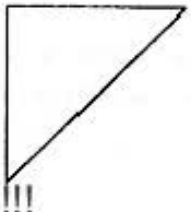
La letra mayúscula B indica que es una sentencia de dibujo de una caja. Los otros parámetros son iguales como la forma previa de la sentencia.

La forma final de la sentencia LINE permite múltiples especificaciones de puntos que tienen que hacerse. Cada punto, después del primero, representa un punto final de destino de cada línea segmento que es dibujado. La posición corriente se asume que es el comienzo del punto final de la línea segmento. Esta forma de la sentencia LINE es:

LINE (X1, Y1) – (X2, Y2) – . . . . (X6, Y6), linea-tipo, color

Los tres puntos de la forma de arriba son utilizados para indicar que una serie de especificaciones de puntos (hasta seis en una fila) pueden darse. Observe que el parámetro B quizás no pueda ser utilizado en esta forma de la sentencia. Damos ahora un programa-ejemplo para dibujar un triángulo utilizando cuatro segmentos. La línea 15 sirve meramente para establecer el origen, mientras que el triángulo es dibujado por la línea 20:

```
10:GRAPH
15:LINE (0,0)-(10
   0,0),9:SORGN
20:LINE (0,0)-(50
   ,50)-(-50,50)-
   (-50,-50)-(0,0
   ),0,0
30:TEXT
```

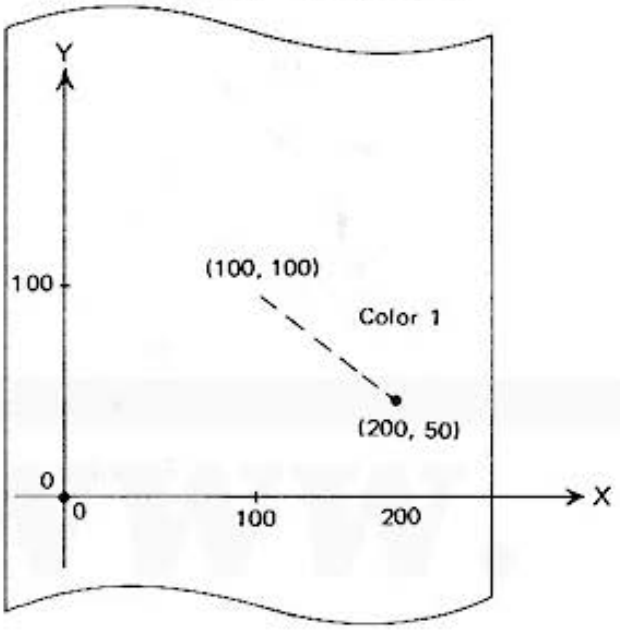


C.6.11. RLINE

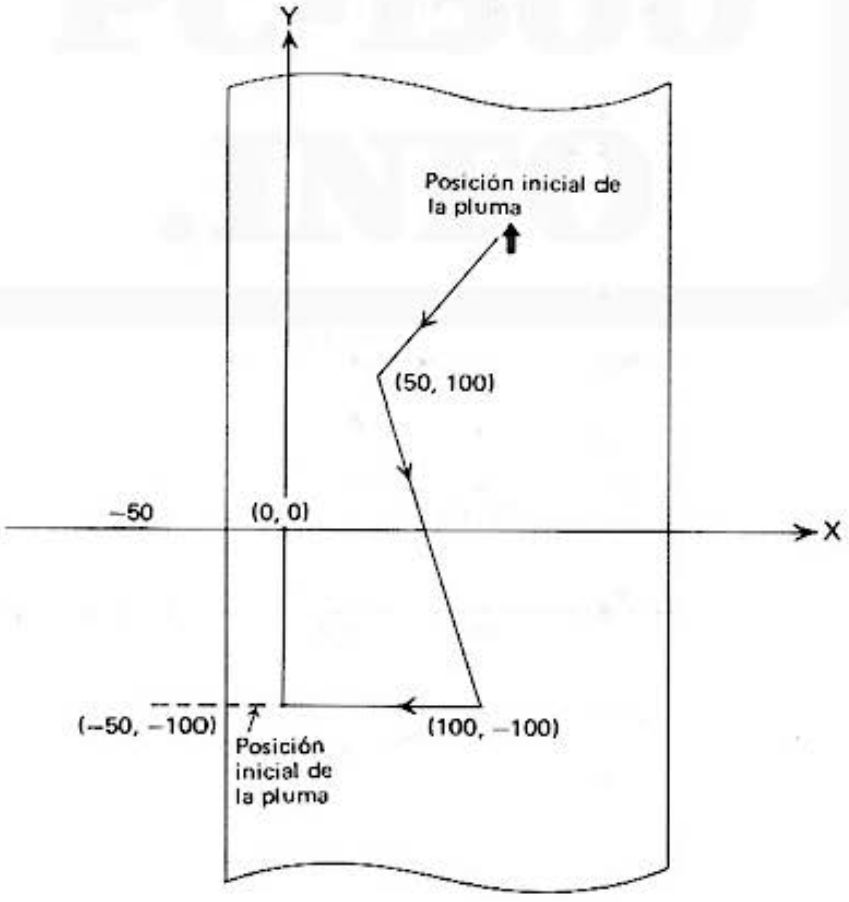
La sentencia RLINE es la misma que la sentencia LINE excepto que todas las especificaciones de puntos representan una posición relativa a la posición actual de la pluma, más que del origen. Las formas de las sentencias RLINE son las mismas que aquellas de las sentencias LINE con las substitución de la palabra clave RLINE por LINE.

Ejemplos:

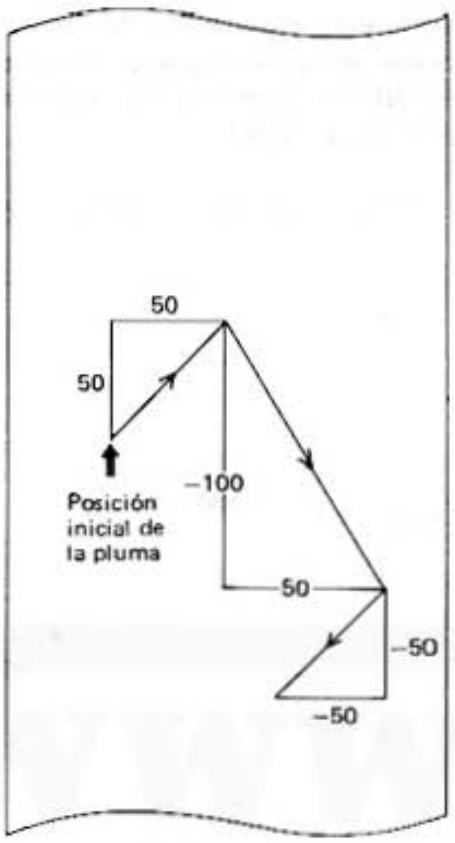
LINE (100, 100) – (200, 50), 2, 1



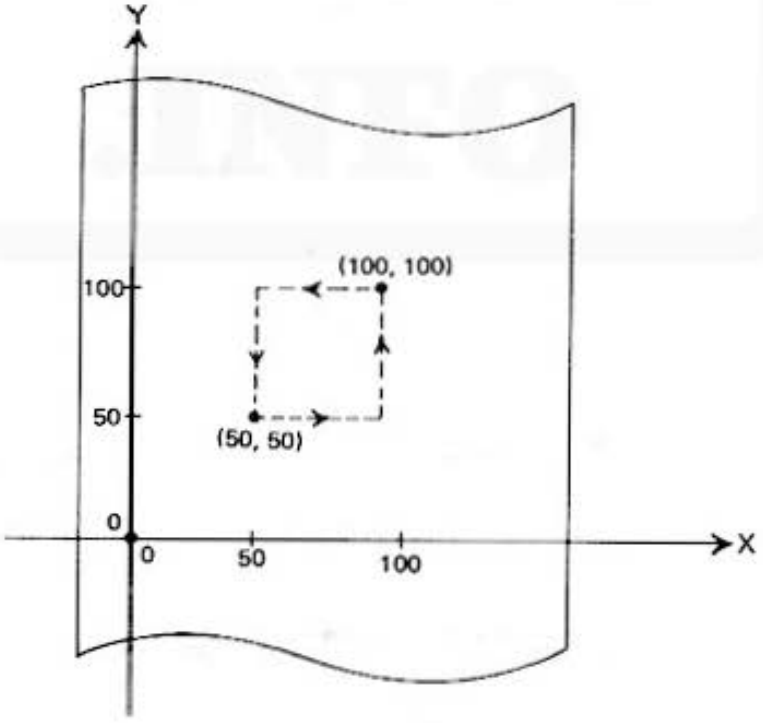
LINE – (50, 100) – (100, –100) – (–50, –100)



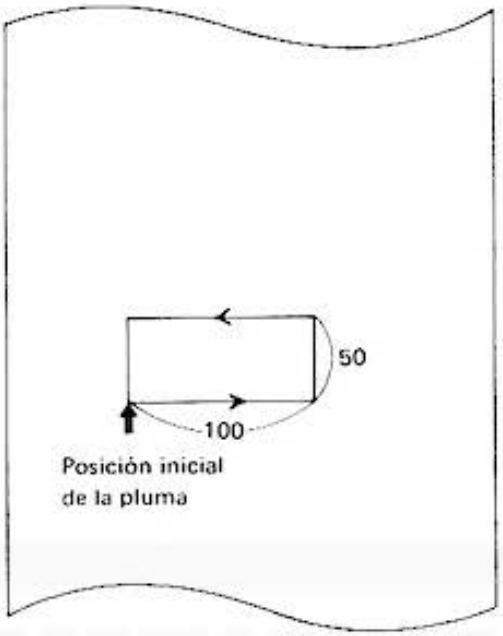
RLINE – (50, 50) – (50, –100) – (–50, –50)



LINE (50, 50) – (100, 100), 2, , B



```
RLINE – (100, 50), , , B
```



VII. MODO RESERVE

A. Definición y Selección de las Teclas de Reserva

Una caracteristica importante de la SHARP PC-1500 que ahorra trabajo es la función de seis teclas la cual puede ser redefinida. Estas teclas de reserva permiten al usuario de la computadora especificar frases escritas frecuentemente o palabras claves con un toque de botón. Las teclas de Reserva son las seis teclas, en la fila superior del teclado, con los rólutos !, ", #, $, %, y &. A cada una de estas teclas pueden ser asignadas hasta tres frases o palabras claves, con un total de 18 frases almacenadas.

La asignación de frases es ejecutada en el tercer modo de la SHARP: el modo RESERVE. Para entrar el modo RESERVE pulse:

SHIFT MODE

El modo indicador en la parte superior de la representación visual debe leer ahora "RESERVE". Para salir del modo RESERVE, sencillamente pulse la tecla MODE una vez.

Debido a que cada tecla reservable puede ser utilizada para usar cualquiera de las tres frases almacenadas, debe haber un método para seleccionar. La frase es utilizada cuando una tecla reservable es pulsada. Este método es una tecla en el lado inferior izquierdo marcada con el símbolo (▲▼). Llamada tecla de selección de reserva, esta tecla selecciona qué frase almacenada corresponde corrientemente a las teclas reservadas. Es importante notar que la tecla de selección de reserva cambia la correspondencia para todas las teclas reservadas al mismo tiempo. Esto quiere decir que la tecla de selección de reserva selecciona un grupo de frases cada una de las cuales corresponde a una tecla reservada única. El grupo que es corrientemente seleccionado se indica por los números Romanos (I, II y III) en la parte superior de la representación visual.

Para asignar una frase a una tecla reservada, coloque el modo RESERVE (pulse [SHIFT] [MODE]) y utilice la tecla de selección de reserva para seleccionar el grupo (I II o III) en el cual la frase será almacenada. Después pulse la tecla Reserve apropriada (!, ", #, $, %, o &). Dependiendo de qué tecla Reserva pulsó, la representación visual aparecerá así:

(El número que sigue la F6 en nuestro ejemplo representa la tecla que fue pulsada). Cuando este prompt aparece, usted puede escribir la frase que va a ser almacenada. Como ejemplo escriba la siguiente:

[R] [U] [N] [1] [0] [0] [ENTER]

Esta frase ahora está asociada con la tecla de reserva que usted seleccionó.

Vamos a probar esto. Vuelva al modo RUN pulsando la tecla MODE. Pulse la tecla de reserva que usted utilizó en nuestro ejemplo. La representación visual ahora mostrará la instrucción que acabamos de almacenar:

Si usted pulsa la tecla ENTER, la computadora intentará ejecutar un programa en la línea 100. La ventaja de la tecla de reserva es que usted no tiene que escribir un comando entero cada vez que desee emitirlo.

Una notación especial permitida en el modo RESERVE podría haber ahorrado problemas en el ejemplo previo. Esta notación es la utilización del signo @ (signo arroba) para representar una pulsación de la tecla ENTER. Si hubiéramos asignado la frase "RUN 100@" a nuestra tecla de reserva, la ejecución del programa habría empezado tan pronto como hubiéramos vuelto al modo RUN y pulsado la tecla de reserva. Para demostrar esto, escriba las sentencias siguientes como línea 222:

```
222 BEEP 5,50: END
```

Ahora vaya al modo RESERVE y defina una de las teclas de reserva utilizando las pulsaciones siguientes:

[G] [O] [T] [O] [2] [2] [2] [@] [ENTER]

(Observe que la pulsación [ENTER] es requerida todavía para definir la tecla de reserva misma).

Vuelva al modo RUN y pulse la tecla de reserva que acabamos de definir. Usted observará que la pulsación ENTER después de la pulsación Reserva es ahora superflua.

En realidad nuestro ejemplo podría haber sido conseguido asignando la frase:

```
BEEP 5,50 @
```

directamente a la tecla de reserva. Inténtelo.

B. Indentificación de Teclas de Reserva

Al mismo tiempo que usted incrementa el uso de las teclas de reserva, usted querrá recorder qué tecla tiene asignada cuál función. La PC-1500 le permite almacenar tres series de caracteres (una por cada grupo de teclas de reserva) que identifican las funciones de las teclas. Estas series son análogas a los comentarios en el modo PROgram.

La identificación de series (llamadas "plantillas") son creadas en el modo RESERVE. Cambie a este modo y seleccione el grupo apropriado de teclas de reserva utilizando la tecla de selección de reserva. Entonces en vez de pulsar una tecla de reserva, como usted normalmente ha hecho, escriba una plantilla y pulse [ENTER]. La plantilla es almacenada en asociación con el grupo.

Como un ejemplo, suponga que nosotros hemo asignado a las teclas de reserva de una a seis (en grupo I) los nombres de las funciones trigonométricas (Seno, Coseno, Tangente, Arcoseno, Arcocoseno y Arcotangente). Para poder recordar qué tecla corresponde a cuál función, nosotros especificaremos un plantila. Para hacer esto cambie al modo RESERVE (pulse [SHIFT] [MODE]), y utilice la tecla de selección de reserva para seleccionar el grupo I (un número romano I aparecerá en la representación visual). Ahora escriba:

"SIN COS TAN ACS ASN ATN"

Pulsaciones:

[SHIFT] ["] [SPACE] [S] [I] [N] [SPACE] [C] [O] [S] [SPACE]
[T] [A] [N] [SPACE] [A] [C] [S] [SPACE] [A] [S] [N]
[SPACE] [A] [T] [N] [SPACE]
[SHIFT] ["] [ENTER]

La plantilla ha sido ahora definida y almacenada.

Vuelva al modo RUN pulsando la tecla [MODE]. Para recordar el significado de las teclas de reserva, simplemente pulse la tecla [RCL] (revocar) y ¡allí están! Pulse la tecla [RCL] una vez más y la prompt vuelve.
Las plantillas pueden ser creadas para cada grupo de teclas de reserva, y pueden ser hasta de 26 caracteres de longitud. Observe que la plantilla es estrictamente un recordatorio para usted: las palabras o letras que usted almacena en una plantilla no tienen significado para la computadora.

C. Supreión de programas de reserva

Como usted sabe, al pulsar [N] [E] [W] [ENTER] borra todas las memorias de reserva. Por favor observe que la operación debe ser hecha en el modo RESERVE.

VIII. COMIENZO DE LA EJECUCION DE LOS PROGRAMAS

A. La Tecla DEF

La tecla DEF (abreviatura de DEFinir) provee varias formas para ahorrar tiempo.

A.1. Ejecución De Programas DEFinibles

La tecla DEF es el tercer método para empezar un programa. Como hemos visto en la sección de la orden RUN, un programa puede ser rotulado con una letra. La tecla DEF puede ser utilizada para iniciar rápidamente un programa rotulado. Esto se hace pulsando la tecla DEF seguida por la tecla alfabética que corresponde al rótulo del programa. Las únicas teclas alfabéticas que pueden ser utilizadas de esta forma son:

A, S, D, F, G, H, J, K, L, Z, X, C, V,
B, N, M, y SPACE

Como ejemplo, escriba las sentencias siguientes para crear tres programas rotulados.

Listado de Programa:

```
10 "Z" : GOSUB 500
20 PRINT "TECLA Z"
30 END
140 "A" : GOSUB 500
150 PRINT "TECLA A"
160 END
270 " " : GOSUB 500
280 PRINT "TECLA ESPACIO"
290 END
500 CLS : PAUSE "USTED PULSA LA ";
510 RETURN
```

En el modo RUN, intente empezar cada programa con la tecla DEF (definir). Observe que al especificar una letra cuyo programa rotulado correspondiente no existe, se producirá un ERROR 11.

A.2. Palabras Claves Pre-Asignadas

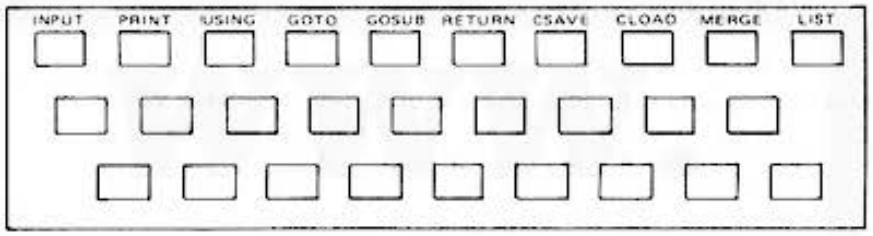
Algunas de las palabras claves más frecuentemente utilizadas han sido asignadas permanentemente a una tecla alfabética única en la segunda fila del teclado. Estas palabras claves pueden ser recuperadas en cualquier modo, pulsando la tecla DEFinir seguida por una de estas teclas alfábeticas. Por ejemplo, para recuperar la palabra clave USING pulse:

DEF E

Las palabras claves disponibles en sus teclas alfábeticas correspondientes son:

Tecla Alfabética	Palabra clave	
Q	INPUT	
W	PRINT	
E	USING	
R	GOTO	
T	GOSUB	
Y	RETURN	
U	CSAVE	Estas palabras claves pueden ser utilizadas cuando la computadora está conectada con el impresor/ interfaz de cassette.
I	CLOAD	
O	MERGE	
P	LIST	

Plantillas



Dos plantillas son provistas con la computadora. Utilícelas para identificar la operación funcional asignada a las teclas de definición.

A.3. La sentencia AREAD

A un programa rotulado que sea iniciado utilizando la tecla DEF, le puede ser ingresado un valor único cada vez que el programa es ejecutado sin utilizar la sentencia INPUT. La lectura del valor es ejecutada por la sentencia AREAD, que debe seguir al rótulo del programa en la misma línea. La sentencia AREAD tiene la forma:

AREAD nombre-variable

donde nombre-variable es un número o nombre de variable de caracteres.

Para pasar un valor a un programa que incorpora una sentencia AREAD, el usuario escribe el valor, pulsa la tecla DEF, y después pulsa la tecla correspondiente al rótulo del programa.

Como ejemplo, escriba los dos programas siguientes:

Listado del Programa:

```
10 "X" : AREAD TM
20 TIME = TM
30 PRINT "PONGA EL RELOJ A " ;TM
40 END
100 "Z" : AREAD D$
110 PRINT "HOY ES " ;D$
120 END
```

Vuelva al modo RUN y empiece el programa rotulado X escribiendo el mes, día, hora y DEF X:

[1] [2] [3] [1] [0] [2] [.] [4] [0] [0] [0] [DEF] [X]

Este programa fijará el sistema del reloj a cualquier hora que usted especifica antes que la pulsación DEF (vea la función TIME).

Para empezar el programa rotulado Z escriba un día de la semana y DEF Z

[V] [I] [E] [R] [N] [E] [S] [DEF] [Z]

B. Iniciación del Programa Automático

Los programas no sólo pueden empezar rápida y fácilmente utilizando la tecla DEF; también pueden empezar automáticamente cuando usted enciende la PC-1500.

Para causar esto, utilice la sentencia ARUN. Este sentencia debe ser la primera sentencia en el programa de la memoria o esta será ignorada. Además, otras condiciones generales son necesarias para que la sentencia ARUN funcione. Estas son que la PC-1500 se apague en el modo RUN, y que no haya ningún error cuando la PC-1500 sea encendida.

El programa siguiente utiliza la sentencia ARUN para saludar al operador de la computadora.

Listado del Programa:

```
10 ARUN
30 CLS
50 BEEP 5,50
70 PRINT "HOLA, COMO ESTAS?"
90 END
```

C. Comparación de los Métodos de Iniciación

Aunque los varios métodos de iniciar un programa consiguen superficialmente el mismo resultado, su operación interna difiere. Para poder utilizar estas diferencias para su ventaja, es necesario discutir el almacenamiento de datos dentro de la computadora. También se incluye una sección comparando varias preparaciones internas que son hechas por la PC-1500 antes de ejecutar un programa.

C.1. Area Fija de la Memoria

Aunque todas las variables del mismo tipo son utilizadas de la misma forma, no son tratadas igual internamente. La PC-1500 incluye un area de memoria fija con bastante espacio de almacenamiento para 26 variables numéricas y 26 variables de series de caracteres (tamaño de la serie de 16 caracteres). Consecuentemente las variables de a A a Z, y las variables de A$ a Z$ están incluídas dentro de esta área.

Todas las demás variables, incluyendo aquellas con nombres de dos caracteres, están alojadas dentro del área mayor de la memoria de la computadora. Esta memoria mayor es también compartida por las instrucciones del programa. Aunque las variables son colocadas empezando en el extremo opuesto de la memoria de las instrucciones, debido a que las instrucciones del programa y los datos comparten la misma área, es posible que ellos utilicen todo el espacio de almacenamiento. En este caso un ERROR en el rango de 177 a 181 ocurrirá.

Es importante darse cuenta que las dos áreas de la memoria no son tratatas igualmente después de iniciar el programa. Esto es explicado en la hoja de información gráfica en la próxima sección. Básicamente las variables en el área fijada de la memoria no son nunca despejadas excepto por una sentencia explicita CLEAR. Aquellas que están en la memoria mayor son despejadas cuando un programa empieza con la orden RUN.

Otra idiosincracia acerca del área fija de la memoria es que los datos en esta área pueden ser redefinidos como un array cuyo nombre es @ (signo arroba), y que dará variables numéricas, y @ $ para variables alfanuméricas. Así la designación @ (1) es la misma localización de almacenamiento que la variable A, y @ (26) es la misma localización del almacenamiento que la variable Z. La designación @ $ (5) se refiere a la misma localización de E$, y la designación @ $ (20) se refiere a la misma localización que T$. Por razones obvias, suscritos que sobrepasan 26 no son pemitidos. Observe que todos los arrays @ y @$ no necesitan ser DIMensionados antes de ser utilizados.

C.2. GRAFICO PARA COMPARACION DE METODOS DE INICIACION DEL PROGRAMA

METODOS DE INICIACION

	RUN	GOTO	DEF
Representación Visual despejada.	S	S	N
Cursor vuelve a la primera columna.	S	N	N
El intervalo WAIT es fijado al infinito.	N	N	N
El modo TRACE es alterado.	N	N	N
El área fijada de la memoria es despejada.	N	N	N
El área mayor de la memoria es despejada.	S	N	N
El "stack" FOR-NEXT, GOSUB es despejado.	S	S	S
ON ERROR GOSUB es cancelado.	S	N	N
Puntero de DATA para a operación READ es restaurado.	S	N	N
El formato USING es cancelado.	S	N	N

APPENDICES

A. TABLA DE ABREVIATURAS

Commandos del Impresor

COLOR	COL. COLO.
CSIZE	CSI. CSIZ.
GLCURSOR	GL. GLC. GLCU. GLCUR. GLCURS. GLCURSO.
GRAPH	GRAP.
LCURSOR	LC. LCU. LCUR. LCURS. LCURSO.
LF	– –
LINE	LIN.
LLIST	LL. LLI. LLIS.
LPRINT	LP. LPR. LPRI. LPRIN.
RLINE	RL. RLI. RLIN.
ROTATE	RO. ROT. ROTA. ROTAT.
SORGN	SO. SOR. SORG.
TAB	– –
TEST	TE. TES.
TEXT	TEX.

Commandos de la Cassette

CHAIN	CHA. CHAI.
CLOAD	CLO. CLOA.
CLOAD?	CLO.? CLOA.?
CSAVE	CS. CSA. CSAV.
INPUT #	I. # IN. # INP. # INPU. #
MERGE	MER. MERG.
PRINT #	P. # PR. # PRI. # PRIN. #
RMT OFF	RM. OF.
RMT ON	RM. O.

Sentencias

AREAD	A. AR. ARE. AREA.
ARUN	ARU.
BEEP	B. BE. BEE.
CLEAR	CL. CLE. CLEA.
CLS	– –
CURSOR	CU. CUR. CURS. CURSO.
DATA	DA. DAT.
DEGREE	DE. DEG. DEGR. DEGRE.
DIM	D. DI.
END	E. EN.
ERROR	ER. ERR. ERRO.
FOR	F. FO.
GOSUB	GOS. GOSU.
GOTO	G. GO. GOT.
GPRINT	GP. GPR. GPRI. GPRIN.
GRAD	GR. GRA.
IF	– –
INPUT	I. IN. INP. INPU.
LET	LE.
LOCK	LOC.
NEXT	N. NE. NEX.
ON	O.

GCURSOR	GC. GCU. GCUR. GCURS. GCURSO.
PAUSE	PA. PAU. PAUS.
PRINT	P. PR. PRI. PRIN.
RADIAN	RAD. RADI. RADIA.
RANDOM	RA. RAN. RAND. RANDO.
READ	REA.
REM	– –
RESTORE	RES. REST. RESTO. RESTOR.
RETURN	RE. RET. RETU. RETUR.
STEP	STE.
STOP	S. ST. STO.
THEN	T. TH. THE.
TRON	TR. TRO.
TROFF	TROF.
UNLOCK	UN. UNL. UNLO. UNLOC.
USING	U. US. USI. USIN.
WAIT	W. WA. WAI.

Commandos

CONT	C. CO. CON.	NEW	– –
LIST	L. LI. LIS.	RUN	R. RU.

Funciones

ABS	AB.	MEM	M. ME.
ACS	AC.	MID$	MI. MID.
AND	AN.	NOT	NO.
ASC	– –	OR	– –
ASN	AS.		
ATN	AT.	PI	– –
CHR$	CH. CHR.	POINT	POI. POIN.
COS	– –	RIGHT$	RI. RIG. RIGH. RIGHT.
DEG	– –	RND	RN.
DMS	DM.	SGN	SG.
EXP	EX.	SIN	SI.
INKEY$	INK. INKE. INKEY.	SQR	SQ.
INT	– –	STATUS	STA. STAT. STATU.
LEFT$	LEF. LEFT.	STR$	STR.

LEN	– –	TAN	TA.
LOG	LO.	TIME	TI. TIM.
LN	– –	VAL	V. VA.

B. REEMPLAZAMIENTO DE BATERIAS DE LA PC-1500

Cuando reemplace las pilas, estas precauciones eliminarán muchos problemas:

- Siempre reemplace las cuatro pilas al mismo tiempo.
- No mezcle baterías nuevas con pilas usadas.
- Utilice solamente pilas (tipo AA, o R6 de 1,5V).

PROCEDIMIENTO PARA EL REEMPLAZO DE LAS PILAS

Antes que la Computadora sea utilizada, o cuando el indicador de la pila desaparece, reemplace las pilas. Por favor siga este procedimiento:

1. Apague la computadora pulsando la tecla [OFF].
2. Quite el tornillo de la cubierta de las pilas con una moneda o un destornillador pequeño (vea fig. 1).
3. Reemplace las cuatro pilas (vea fig. 2).

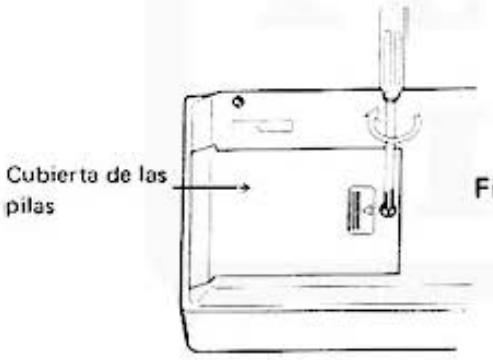


Fig. 1

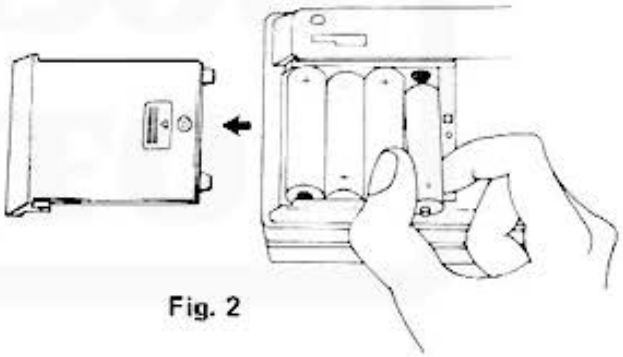
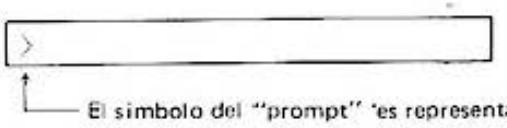


Fig. 2

4. Empuje la cubierta de las pilas ligeramente mientras reemplace el tornillo.
5. **Para proseguir, pulse las teclas [ON] y [CL]. Escriba NEW Ø y pulse la tecla [ENTER]**
6. Compruebe el visor.



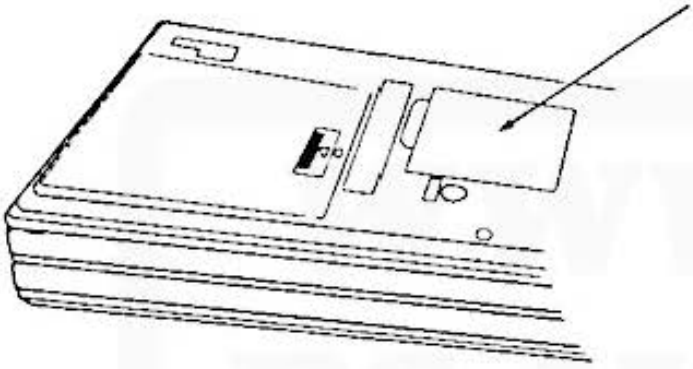
El simbolo del "prompt" 'es representado.

Si el visor es blando o representa otro simbolo que no sea " > ' remueve las pilas y pónga las otra vez, y comprueba la representación visual.

Nota:

- Si la Computadora no va a ser utilizada por bastante tiempo, retire las pilas para evitar el daño que causarín fugas de las pilas.
- Conservando una pila agotada puede resultar dañada la computadora debido a la fuga del compuesto químico de la pila. Remueve las pilas agotadas inmediatamente.
- La batería recargable no puede ser utilizada como una batería para la PC-1500.
- El adaptador de CA. EA-150 para la CE-150 se puede utilizar como un adaptador para la PC-1500.

  (No conecte la EA-150 a la computadora PC-1500 cuando la PC-1500 es conectada al impresor/interfaz CE-150).



Compartimiento modular. Este es el lugar donde un módulo de memoria opcional se instala.

No toque ningún conector dentro del compartimiento. Esto puede afectar adversamente la computadora debido a la electricidad estática.

C. CODICO DE CARACTERES ASCII PARA LA PC-1500

Posición de bit de la parte inferior b4, b3, b2, b1 ↓	Hexa-decimal	Posición de bit de la parte superior b7, b6, b5 → 000	001	010	011	100	101	110	111
0000	0			SPACE	0	@	P		p
0001	1			!	1	A	Q	a	q
0010	2			"	2	B	R	b	r
0011	3			#	3	C	S	c	s
0100	4			$	4	D	T	d	t
0101	5			%	5	E	U	e	u
0110	6			&	6	F	V	f	v
0111	7			'	7	G	W	g	w
1000	8			(	8	H	X	h	x
1001	9			)	9	I	Y	i	y
1010	A			*	:	J	Z	j	z
1011	B			+	;	K	√	k	{
1100	C			,	<	L	¥	l	¦
1101	D			-	=	M	π	m	}
1110	E			.	>	N	^	n	~
1111	F			/	?	O	_	o	■

E: PC-1500 LISTA DE CODIGOS DE ERRORES

Código Del Error **Explicación**

1. Error sintáctico: Este error resulta al escribir información como:

```
10: GOTO
```

o información imcompleta:

```
10: GOTO
```

o mandatos inválidos

```
10: NEW
```

(debido a que NEW no puede ser una sentencia)

< Representación visual > ERROR 1 IN 10

2. Este error ocurre cuando no hay correspondencia entre comandos FOR/NEXT ó GOSUB/RETURN.

Ej.

```
10:  FOR A = 1 TO 10
      .
      .
      .
100: NEXT B
```

< Representación visual > ERROR 2 IN 100

4. Este error ocurre cuando no hay datos correspondientes al comando READ.

Ej.

```
10: READ X, Y
20: DATA 10
30: END
```

< Representación visual > ERROR 4 IN 10

5. Este error ocurre cuando una variable array de declara con el nombre de una variable existente.

Ej.

```
10: DIM A (10, 10)
20: DIM A (5)
```

< Representación visual > ERROR 5 IN 20

PC-1500 LISTA DE CODIGOS DE ERRORES

Código Del Error **Explicación**

6. Este error ocurre cuando una variable array ha sido utilizada sin una sentencia DIM (dimensión)

Ej.
```
10: CLEAR
20: A(3) = 1
```

< Representación visual > `ERROR 6 IN 20`

7. Este error ocurre cuando el nombre de la variable no es apropiado.

Ej.
```
10: A$ = 10 o
10: FOR A$ = 1 TO 10
```

< Representación visual > `ERROR 7 IN 10`

8. Este error ocurre cuando la dimensión excede 3 en la declaración de una variable array.

Ej.
```
10: DIM A (3, 4, 5, 6)
```

< Representación visual > `ERROR 8 IN 10`

9. Este error ocurre cuando el número subscrito de una variable array excede el tamaño de la array presentada del comando DIM.

Ej.
```
10: DIM A (3)
20: A (4) = 1
```

< Representación visual > `ERROR 9 IN 20`

10. Este error ocurre cuando no hay bastante memoria disponible para crear más variables.

Ej.

pulsaciones la tecla	representación visual
MEM [ENTER]	7
AB = 10 [ENTER]	ERROR 10

PC-1500 LISTA DE CODIGOS DE ERRORES

Código Del Error **Explicación**

11. Este error ocurre cuando la línea especificada no existe en el programa.

Ej. 10: INPUT "X =" ;X: GOTO 5

< Representación visual > ERROR 11 IN 10

12. Este error ocurre cuando el mandato USING especifica un formato con especificaciones incorrectas.

Ej. 100: PRINT USING "# # #, A#"; 10

< Representación visual > ERROR 12 IN 100

13. Este error ocurre cuando un programa excede la capacidad de memoria de la PC-1500 o cuando la tecla reservada para especificación excede la capacidad del área reservable.

Ej.	pulsaciones	representación visual
	MEM [ENTER]	7
	15 A = A + 1 [ENTER]	ERROR 13

14. (1) Sentencias FOR que han sido demasiado anidadas y la capacidad del "stack" (pila) interno ha sido excedida.
(2) Al construir una expresión, el registro intermedio para expresiones ha sido excedido.

15. (1) Sentencias GOSUB que han sido anidadas demasiado y la capacidad del "stack" (pila) interno ha sido excedida.
(2) Al construir una expresión, el registro intermedio para la serie ha sido excedido.

16. (1) El valor especificado es mayor que 10 E100 o menor que −10 E100

Ej. 12 E99

(2) El valor fijado por hexadecimales excede 65535

Ej. &1FFAB

PC-1500 LISTA DE CODIGOS DE ERRORES

Código Del Error **Explicación**

17. El tipo de datos (numerales, serie de caracteres) es inapropiado para calculo de expresiones.

Ej. 1 + "A" ENTER

18. Número de argumentos inapropiado para la expresión.

Ej. LEFT$ ("ABC") ENTER
SIN (30, 60) ENTER

19. Valor numérico especificado fuera del rango permitido.

Ej. 10: DIM A(256)

< Representación visual > ERROR 19 IN 10

20. Cuando las variables arrays de memoria fija son especificadas, no hay '(' siguiendo '@' o '@$'.

Ej. 100: @$ = "A"

< Representación visual > ERROR 20 IN 100

21. Una variable es requerida en la expresión.

Ej. 10: FOR 1 = 0 TO 10

< Representación visual > ERROR 21 IN 10

22. Este error ocurre cuando el programa está cargado, y no hay espacio en la memoria disponible para más información.

23. Este error ocurre cuando la hora es fijada incorrectamente.

Ej. TIME = 131005.10 ENTER

PC-1500 LISTA DE CODIGOS DE ERRORES

Código Del Error **Explicación**

26. Este error ocurre cuando el comando no puede ser ejecutado en el modo corriente.

Ej. < RUN MODE > NEW [ENTER]

27. Este error ocurre cuando no hay ningún programa que corresponde al rótulo dado.

Ej. pulsaciones representación

[DEF] 1 [ENTER] ERROR 27

28. Este error ocurre cuando un mandato o código de función ha sido insertado dentro de " ", o cuando usted intente substituir comandos INPUT o comandos AREAD por variables de caracteres.

Ej. 10 INPUT A$

pulsaciones representación visual

[DEF] W [ENTER] ERROR 28

30. Este error ocurre cuando un número de línea excede 65539.
(65280 ~ 65539 : ERROR 1)

Ej. 102235 A = 10 [ENTER]

32. Este error ocurre cuando el cursor gráfico está entre las columnas 152–155 durante la ejecución de comandos de entrada (el dato de entrada no puede ser representado gráficamente).

Ej. 100: GCURSOR 152
110: INPUT X

< Representación visual > ERROR 32 IN 110

34. Este error ocurre cuando no hay ningún mecanismo periférico para el nombre del mecanismo.

35. El número del mecanismo periférico especificado por la expresión en una expresión PRINT#, expresión INPUT#, no es consistente o incluso si es consistente, el mecanismo de periférico no puede operar con los comandos de entrada o salida dados.

177 ~ 181 Durante la cración del programa, el programa se ha extendido más allá del área para datos. Y ha tenido lugar la superposición de estas dos áreas.

PC-1500 LISTA DE CODIGOS DE ERRORES

Código Del Error	Explicación

0, 224 ~ 241 Durante la ejecución de los comandos INPUT o comandos AREAD, se han dado datos incorrectos de entrada.

Ej. 10: INPUT A

pulsaciones	representación visual
123 PRINT [ENTER]	ERROR 240

36. Datos o caracteres no pueden ser representados gráficamente de acuerdo con el formato especificado por el (USING) vigente.

Ej. 10: USING "# # # #. # #"
20: PRINT 12345

La sección de números enteros junto con su signo ha excedido 4 espacios enteros.

37. Este error ocurre en cálculos numéricos, cuando el resultado del cálculo ha excedido 9,999999999E 99.

38. Este error ocurre cuando en unz división se ha utilizado 0 como denominador.

Ej. 10: PRINT 5/0

39. Este error ocurre cuando un cálculo ilógico ha sido hecho:

- Cálculo logarítmico de un número negativo — Ex. LN (−10)
- ASN, ACS en el caso de $|X| > 1$ — Ex. ASN (1,5) ACS (100)
- Raíz cuadrada de números negativos — Ex. SQR (−10)

Errores relacionados con el cassette

40. Especificación inapropiada para la expresión.

41. SAVE y LOAD fueron especificados para el área ROM.

42. El archivo de datos del cassette es demasiado largo y no puede ser cargado (LOAD).

PC-1500 LISTA DE CODIGOS DE ERRORES

Código Del Error	Explicación

43. Al verificar datos utilizando el comando CLOAD?, el formato de datos a ser verificados no iguala al formato del archivo.

44. Un error de CHECK SUM ha ocurrido.

Comandos relacionados con el impresor

70. (1) La pluma ha excedido el rango de las coordenadas de –2048 <= X, Y <= 2047.
(2) La pluma excederá el rango al ejecutar más comandos.

71. (1) El papel ha corrido hacia atrás más de 10,24 cm.
(2) El papel retrocederá más de 10,24 cm. al ejecutar más comandos. (en el modo TEXT)

72. El valor dado no es apropiado para el valor de TAB o LCURSOR.

73. Un comando ha sido utilizado en el modo de impresor erróneo (GRAPH o TEXT).

74. El número de comas (,) en los comandos LINE o RLINE es demasiado grande.

Nota: Una entrada de más de siete comas ocasionará error. También, si la primera coma es omitada, más de seis comas causarán un error.

76. Para LPRINT, cuando la impresión de los resultados del cálculo no puede ser hecha en una línea (en el modo TEXT).

78. (1) Las plumas están en el proceso de ser cambiadas.
(2) El estado de batería baja no ha sido corregido. (vea Nota 1).
Este error ocurre cuando, por cualquiera de estas de razones, los comandos que mueven la pluma (como LPRINT y LINE) no pueden ser ejecutados.

79. La señal de color no ha aparecido (vea NOTA 2).

80. Batería baja (vea NOTA 3)

NOTAS:

(1) Si el error 78 es debido al estado de LOW BATTERY, apague la CE-150. Después de recargar el impresor, encienda la CE-150. Usted puede continuar ahora.

(2) La señal de color es para COLOR y se genera sólo cuando la pluma viene al lado izquierdo. Cuando la pluma está en esta posición, es posible conocer el color de la pluma actual.

(3) Después de recargar, encienda el interruptor de la CE-150 inmediatamente otra vez y empiece la operación.

F: LECTURAS SUGERIDAS

Programación en BASIC

Problem Solving and Structured Programming in BASIC by Elliot Koffman and Frank Friedman. (Addison-Wesley Publishing Co., Reading, Massachusetts). 1979.

basic BASIC by Donald M. Monro. (Winthrop Publishers, Inc. Cambridge, Massachusetts), 1978.

BASIC With Business Applications by Richard W. Lott. (John Wiley & Sons, New York, New York), 1977.

Practical BASIC Programs edited by Lon Poole. (OSBORE/McGraw-Hill, Berkeley, California.), 1980.

Consulta General

Introduction to Computers and Data Processing by Gary B. Shelley and Thomas J. Cashman. (Anaheim Publishing Co., Fullerton, California), 1980.

O: ORDEN DE EVALUACION DE LA EXPRESION

Los cálculos se ejecutan de acuerdo con la jerarquía siguiente: expresiones en paréntesis tienen la mayor prioridad y operaciones lógicas tienen la menor. Si dos o más operaciones de la misma prioridad son encontradas en la misma expresión, son evaluadas de izquierda a derecha.

1) Expresiones en paréntesis

2) Recuperación de valores de las variables, TIME, PI, MEM, INKEY$

3) Funciones (SIN, COS, LOG, EXP, etc.)

4) Potenciación (Ejemplo: 2A ^ 3 = 2 * (A ^ 3))

5) Signo Aritmético (+, −)

6) Multiplicación, División (*, /)

7) Adición, Sustracción (+, −)

8) Operadores de comparación (<, <=, =, >=, >, <>)

9) Operadores Lógicos (AND, OR, NOT)

NOTAS:

Cuando ambos signos aritméticos y potencias son utilizados en la misma expresión, la potencia es ejecutada antes que el signo.

Ejemplo: −5 ^ 4 es evaluado a −625 en vez de 625.

Los cálculos dentro de paréntesis serán ejecutados primero. En expresiones con varios niveles de paréntesis, los cálculos empiezan con el par más interior y prosiguen hasta el paréntesis más exterior.

Ejemplos de Resolución

$$7 \wedge 2 + 3 * \sqrt{144} / \sqrt{81} + \text{SIN}\,(120 + 150) * -3$$

$$7 \wedge 2 + 3 * \sqrt{144} / \sqrt{81} + \text{SIN}\,(270) * -3$$

$$7 \wedge 2 + 3 * 12 / 9 + -1 * -3$$

$$49 + 3 * 12 / 9 + -1 * -3$$

$$49 + 36 / 9 + 3$$

$$49 + 4 + 3$$

$$53 + 3$$

$$56$$

(RANGO DE CALCULO)

Funciones	Rango dinámico
$y \wedge x$ (y^x)	$-1 \times 10^{100} < x \log y < 100$ ($y = 0, x \leqq 0$: ERROR 39; $y = 0, x > 0$: 0; $y < 0, x \neq$ entero : ERROR 39)
SIN x COS x TAN x	DEG: $\lvert x \rvert < 1 \times 10^{10}$ RAD: $\lvert x \rvert < \frac{\pi}{180} \times 10^{10}$ GRAD: $\lvert x \rvert < \frac{10}{9} \times 10^{10}$ En TAN x sin embargo, los siguientes casos son excluidos. DEG: $\lvert x \rvert = 90(2n-1)$ RAD: $\lvert x \rvert = \frac{\pi}{2}(2n-1)$ GRAD: $\lvert x \rvert = 100(2n-1)$ (n: numero entero)
$SIN^{-1}x$ $COS^{-1}x$	$-1 \leqq x \leqq 1$
$TAN^{-1}x$	$\lvert x \rvert < 1 \times 10^{100}$
LNx LOGx	$1 \times 10^{-99} \leqq x < 1 \times 10^{100}$
EXPx	$-1 \times 10^{100} < x \leqq 230.2585092$
$\sqrt{x}$	$0 \leqq x < 1 \times 10^{100}$

Otras funciones no mostradas arriba sólo pueden ser calculadas cuando x está dentro del siguiente rango.

$$1 \times 10^{-99} \leqq \lvert x \rvert < 1 \times 10^{100} \text{ and } 0$$

(Ex.)

Ø ^ Ø [ENTER]	→	ERROR 39
Ø ^ 5 [ENTER]	→	Ø
(−4) ^ Ø.5 [ENTER]	→	ERROR 39
−4 ^ Ø.5 [ENTER]	→	−2

- Como regla, el error de cálculos funcionales es menor que ±1 en el número más bajo de un valor numérico mostrado en el visor (al dígito más bajo de la mantisa en el caso de notación científica) dentro del rango de cálculos mostrado arriba.

X. COMPARACION DE COMANDOS ENTRE PC-1211 Y PC-1500

X. 1 Las instrucciones de la PC-1211 que están disponibles en la PC-1500 son:

1. Funciones

ABS
ACS
ASN
ATN
COS
DEG
DMS
EXP
INT
LOG
LN
π (PI)
SGN
SIN
√ (Raíz cuadrada)
TAN
^ (potenciación)

2. Sentencias

AREAD
USING
CLEAR
DEGREE
END
FOR-TO-STEP
GOSUB
GOTO
GRAD
IF
INPUT
LET
MEM
NEXT
PAUSE
PRINT
RADIAN
REM
RETURN
STOP
THEN
USING

3. Comandos

CONT
LIST
NEW
RUN

4. Comandos Del Cassette

CHAIN
CLOAD
CLOAD?
CSAVE
INPUT #
PRINT #

X-2 COMANDOS UNICOS DE LA PC-1500

1. Funciones

AND
ASC
CHR$
INKEY$
LEFT$
LEN
MID$
NOT
OR
POINT
RIGHT$
RND
STATUS
STR$
TIME
VAL

2. Sentencias

ARUN
BEEP (not PC-1211)
CLS
CURSOR
DATA
DIM
LOCK
ON ERROR
ON GOSUB
ON GOTO
POINT
RANDOM
READ
RESTORE
TRON
TROFF
UNLOCK
WAIT

3. Comandos

(Iguales que la PC-1211)

4. Instrucciones de Cassette

MERGE
RMT OFF
RMT ON

5. Instrucciones del Impresor

COLOR
CSIZE
GCURSO
GLCURSOR
GLCURSOR
GPRINT
GRAPH
LCURSOR
LF
LINE
LLIST
LPRINT
RLINE
ROTATE
SORGN
TAB
TEST
TEXT

Z. TABLA DE REFERENCIAS DE LOS COMANDOS

1. Funciones

Función	Abreviaturas	Observaciones
ABS	AB.	Valor absoluto
ACS	AC.	$\cos^{-1}$
AND	AN.	exp. AND exp. [And de Boole]
ASN	AS.	$\sin^{-1}$
ATN	AT.	$\tan^{-1}$
CHRS	CH. CHR.	Convierte los caracteres al código ASCII CHR$ ASCII valor numérico de código decimal.
COS		
DEG		Convierte grados, minutos, segundos a grados decimales.
DMS	DM.	Convierte grados decimales a grados, minutos, segundos.
EXP	EX.	e^x
INKEYS	INK. INKE. INKEY.	Variable de caracteres = INKEYS Si una tecla es pulsada durante la ejecución del mandato INKEY$, el carácter ASCII será leído dentro de la variable de caracteres.
INT		Convierte un valor fraccionado a un número entero: INT (10/3) [ENTER] < Display > 3
LEFT$	LEF. LEFT.	LEFT$ (variable carácter, expresión numérica) Toma el número especifico de caracteres del lado izquierdo de la serie de caracteres especificados.
LEN		LEN "carácter" variable carácter Da el número de caracteres en la series de caracteres especificada
LOG	LO.	$\log_{10} X$
LN		$\log_e X$
MEM	M	Representa gráficamente el número restante de pasos disponibles en la memoria como el mandato STATUS 0.

Función	Abreviaturas	Observaciones
MID$	MI. MID.	MID$ (variable cáracter, numérico-exp1, numérico-exp2); Toma caracteres de medio de la serie de caracteres especificada.
NOT	NO.	NOT exp. [Negación de Boole]
OR		exp. OR exp. [OR de Boole]
π (PI)		Especifica la relación de la circunferencia: (= 3,141892654)
POINT	POI. POIN.	Devuelve un número que representa la estructura de puntos activados dentro de la columna dada. POINT expresión numérica
RIGHT$	RI. RIG. RIGH. RIGHT.	RIGHT$ (carácter-variable, exp-numérica) Toma el número especifico de caracteres del lado derecho de la serie de caracteres especificados.
RND	RN.	RND expresión Genera números al azar
SGN	SG.	Función del signo
SIN	SI.	Seno de una expresión.
$\sqrt{\ }$ (SQR)	SQ.	Raíz cuadrada de una expresión.
STATUS	STA. STAT. STATU.	STATUS 0 ó STATUS 1 (0) Número de pasos de programa disponibles. (1) Número de pasos de programa utilizados.
STR$	STR.	STR$ expresión numérica Convierte números a la serie de caracteres.
TAN	TA.	Tangente de una expresión.
TIME	TI. TIM.	(1) TIME = MMDDHH. MMSS (2) TIME [anuncia fecha y hora] La función del tiempo fija y recupera el mes (MM), día(DD), hora(HH), minuto(MM), y segundos(SS).
VAL	V. VA.	VAL { "carácter" / variable carácter } Convierte series de caracteres a números
^		Función de exponenciación.

2. Sentencias

Sentencia	Abreviaturas	Observaciones
AREAD (lectura automática)	A. AR. ARE. AREA.	AREAD variable Cuando ejecute programas a través de teclas definidas, AREAD introduce los contenidos de la representación visual en la variable especificada.
ARUN (auto run)	ARU.	ARUN Comando para empezar la ejecución del programa automáticamente cuando se enciende la PC-1500.
BEEP	B. BE. BEE.	BEEP exp1, exp2, exp3 Comando de sonido. Enciende las funciones generadoras de sonido ON/OFF, especifica el tono y la duración de sonidos.
CLEAR	CL. CLE. CLEA.	Comando para borrar todos los valores de las variables.
CLS (clears)		Borra la representación visual.
CURSOR	CU. CUR. CURS. CURSO.	(1) CURSOR exp. (0 < = exp. < = 255) especificación de la posición inicial de la representación visual (2) CURSOR cancela la especificación previa.
DATA	DA. DAT.	DATA exp, exp, ... Especifica datos dentro de un programa utilizados con la sentencia READ.
DEGREE	DE. DEG. DEGR. DEGRE.	Especificación del modo angular. El grado [°] es designado.
DIM (dimensión)	D. DI.	(1) DIM nombre de la variable (exp) (2) DIM nombre de la variable (exp)*exp3 (3) DIM nombre de la variable (exp1, exp2) nombres variables: A, B, C$, D$, etc . . . ( ): especifica el tamaño y la dimensión de un array. exp3: especificación del espacio de dígitos
END	E. EN.	Señala el final del programa.
FOR TO STEP	F. FO. STE.	(1) FOR variable-numérica = exp 1 TO exp 2 (2) FOR variable numérica = exp 1 TO exp 2 STEP exp 3 Comienxa el bucle FOR-NEXT. Utilizado en correspondencia con el comando NEXT. exp1: expresión inicial exp2: valor final exp3: intervalo para incrementar con cada bucle

Sentencia	Abreviaturas	Observaciones
GCURSOR	GC. GCU. GCUR. GCURS. GCURSO.	Especifica la posición del cursor gráfico espresión GCURSOR (0 <= expresión <= 155) ó (&0 <= expresión <= &9B)
GOSUB	GOS. GOSU.	GOSUB { expresión / "carácter" / variable caracter } Empieza la ejecución de una subrutina en la línea o rótulo especificado.
GOTO	G. GO. GOT.	Transfiere control a una línea o rótulo especificado.
GPRINT	GP. GPR. GPRI. GPRIN.	Manipula la representación visual LCD para gráficas. (1) GPRINT "0000 . . . " (" " son números hexadecimales) (2) GPRINT 0; 0; . . . (3) GPRINT 0, 0, . . .
GRAD	GR. GRA.	Selecciona grados decimales como el modo angular.
IF		(1) IF expresión condicional (THEN) expresión (2) IF expresión aritmética (THEN) expresión Evalúa condiciones dadas y o mueva la ejecución a la próxima línea o ejecuta. La cláusula THEN es opcional.
INPUT	I. IN. INP. INPU.	(1) INPUT nombre-variable (2) INPUT "serie" variable, "serie", variable (3) INPUT "serie"; variable, "serie"; variable
LET	LE.	(1) LET variable-numérica = expresión (2) LET variable-numérica = variable-numérica (3) LET variable carácter = "caracteres" (4) LET variable carácter = variable carácter LET es opcional excepto dentro de frase IF.
LOCK	LOC.	Traba la computadora dentro del modo corriente.
NEXT	N. NE. NEX.	NEXT nombre-variable Señala el final del bucle FOR-NEXT. El nombre de la variable en la sentencia FOR.
ON ERROR	O. ER ERR. ERRO.	ON ERROR GOTO expresión Sentencia para detectar errores.

Sentencia	Abreviaturas	Observaciones
ON GOSUB	O. GOS. GOSU.	On expresión GOSUB exp1, exp2, exp3 . . . . Forma automática de la sentencia GOSUB. Empieza la ejecución de las subrutinas especificadas por exp1, exp2, exp3, etc. dependiendo en el valor de la variable dada.
ON GOTO	O. G. GO. GOT.	ON expresión GOTO exp1, exp2, exp3 Transfiere control a uno de las líneas registradas como exp1, exp2, exp3, etc., dependiendo de valor de la variable dada.
PAUSE	PA. PAU, PAUS.	Misma forma que el comando PRINT. Representa gráficamente la información especificada por cerca de 0,85 segundos, después continúa el programa.
POINT	POI. POINT.	POINT expresión (0 < = expresión < = 155) (&0 < = expresión < = &9B) Devuelve un número que representa la estructura de puntos activados en la columna indicada de la representación visual.
PRINT	P. PR. PRI. PRIN.	(1) PRINT expresión (2) PRINT expresión; (3) PRINT expresión; expresión2 (4) PRINT expresión; expresión2; . . . .
RADIAN	RAD. RADI. RADIA.	Selecciona radianes como el modo angular.
RANDOM	RA. RAN. RAND. RANDO.	Cambia el origen utilizado por el RND para generar números al azar.
READ	REA.	READ variable, variable2, . . . etc. Introduce datos desde la sentencia DATA dentro de variables especificadas.
REM (REMarque)		REM información explicadora Especifica los comentarios en el cuerpo del programa.
RESTORE	RES. REST. RESTO.	(1) RESTORE explicatiua —Cambia el orden de datos utilizados por la sentencia READ. (2) RESTORE —Restaura los datos a la primera sentencia DATA.
RETURN	RE. RET. RETU. RETUR.	Continúa la ejecución en la sentencia siguiente a la sentencia GOSUB que invocó la subrutina actual.

Sentencia	Abreviaturas	Observaciones
STOP	S. ST. STO.	Comando para detener le ejecución del programa.
THEN	T. TH. THE.	Utilizado en la sentencia IF para separar una expresión de la prueba condicional que determina si la expresión será resulta THEN { expresión / "carácter" / carácter-variable }
TRON (trace on)	TR. TRO.	Enciende el modo Trace para detectar los problemas en el programa.
TROFF	TROF.	Apaga el modo Trace.
UNLOCK	UN. UNL. UNLO. UNLOC.	Cancela 'Lock' permitiendo al usuario cambiar modos.
USING	U. US. USI. USIN.	Especifica el formato para presentación de la información por un programa.
WAIT	W. WA. WAI.	WAIT expresión (0 < = expresión < = 65535) Especifica la duración de información en la representación visual cuando utilice comandos PRINT. WAIT sin argumentos cancela la especificación previa (duración infinita).

3. Comandos

Comando	Abreviaturas	Observaciones
CONT (CONTinuar)	C. CO. CON.	Continúa otra vez la ejecución de programas que han sido temporalmente parados. Es efectivo en el modo RUN.
LIST	L. LI. LIS.	Efectúa un listado del programa. Efectiva en el modo PROgrama.
NEW		(1) NEW (2) NEWO En el modo PRO, borra el programa y todas las variables. En el modo RESERVE, borra el contenido del área de reserva.
RUN	R. RU.	(1) RUN (2) RUN expresión (3) RUN "carácter" variable-carácter Efectivo en el modo RUN.

4. Comandos del cassette

Comando	Abreviaturas	Observaciones
CHAIN	CHA. CHAI.	Lee un programa de cinta magnética y lo ejecuta. (1) CHAIN "nombre de archivo" (CHAIN-1 "nombre de archivo" (2) CHAIN "nombre archivo", expresión (CHAIN-1 "nombre archivo", expresión) (3) Formas abreviadas del "nombre archivo" de (1) y (2).
CLOAD	CLO. CLOA.	Guarda los contenidos del programa o memoria reservada en la cinta magnética. (1) CLOAD (CLOAD-1) (2) CLOAD "nombre clave" (CLOAD-1 "nombre clave")

Comando	Abreviaturas	Observaciones
CLOAD?	CLO.? CLOA.?	Comandos de comparación. Este comando compara un programa en memoria con un programa grabado en cinta magnética. Puede también ser utilizado para compara los contenidos de la memoria de reserva. (1) CLOAD? (CLOD?-1) (2) CLOAD? "nombre archivo" (CLOAD?-1 "nombre archivo")
CSAVE (guardar en cassette)	CS. CSA. CSAV.	Este comando graba en la cinta el contenido del programa y la memoria de reserva. (1) CSAVE (CSAVE-1) (2) CSAVE "nombre de archivo" (CSAVE-1 "nombre de archivo")
INPUT #	I. # IN. # INP. # INPU. #	Este comando transfiere datos grabados en cinta a las variables especificadas. Tiene la misma forma que el comando PRINT #.
MERGE	MER. MERG.	Este comando carga en memoria programas almacenados previamente en cinta magnética sin escribir caracteres o borrar el programa en la memoria. Formato similar a CLOAD.
PRINT #	P. # PR. # PRI. # PRIN. #	Comando de grabación de datos. Este comando graba en cinta los datos almacenados en la PC-1500. (1) PRINT # nombre variable, nombre variable, . . . (2) PRINT # "nombre archivo", nombre variable, . . . PRINT #-1 "nombre archivo", nombre variable, . . .
RMT OFF (remoto apagado)	RM. OF. RMTOF.	Este comando cancela la función remota del terminal REM 1 (para el segundo grabador de cinta).
RMT ON (remoto encendido)	RM. O. RMTO.	Este comando repone la función remota del terminal REM 1 (para el segundo grabador de cinta)

5. Comandos para el impresor

Comando	Abreviaturas	Observaciones
COLOR	COL. COLO.	Especifica el color para imprimir COLOR expresión (0 <= expresión <= 3)
CSIZE (tamaño de carácter)	CSI. CSIZ.	Especifica el tamaño de los caracteres impresos con la expresión CSIZE (1 <= expresión <= 9)
GLCURSOR	GL. GLC. GLCU. GLCUR. GLCURS. GLCURSO.	Mueve la posición de la pluma de un punto inicial a una coordenada X-Y. Es válido solamente en el modo graph. GLCURSOR (exp1, exp2)

Comando	Abreviaturas	Observaciones
GRAPH	GRAP.	Fija el modo a ser utilizado para dibujar gráficos e ilustraciones.
LCURSOR	LC. LCU. LCUR. LCURS. LCURSO.	Mueve la pluma del impresor a una de las posiciones de los caracteres disponibles.
LF		Mueve el papel hacia adelante el número de renglones indicados por la expresión. LF expresión
LINE	LIN.	Comando para dibujar líneas. (1) LINE (exp1, exp2) – (exp3, exp4) (2) LINE (exp1, exp2) – (exp3, exp4), exp5, exp6 (3) LINE (exp1, exp2) – (exp3, exp4), exp5, exp6, B exp5: especifica el tipo de línea (sólida, quebrada) exp6: especifica el color de la línea.
LLIST	LL. LLI. LLIS.	Lista una renglón o renglones del programa corriente. (1) LLIST (2) LLIST exp1 (3) LLIST, exp1 (4) LLIST exp1, (5) LLIST exp1, exp2 (6) LLIST "rótulo" (7) LLIST "rótulo",
LPRINT	LP. LPR. LPRI. LPRIN.	Imprime por impresor información especificada. Válido para el modo TEXT solamente. Tiene las mismas formas que el comando PRINT.
RLINE (línea relativa)	RL. RLI. RLIN.	Comando que traza líneas con la posición de las plumas como punto de partida. Válido para el modo GRAPH solamente. Tiene la misma forma que los comandos LINE.
ROTATE	RO. ROT. ROTA. ROTAT.	Especifica la dirección de impresión de los caracteres. Válido solo en modo GRAPH. ROTATE expresión (0 <= expresión <= 3)
SORGN (set origin)	SO. SOR. SORG.	Este comando especifica la posición presente de la pluma como el punto de partida (punto de origen). Válido para el modo GRAPH solamente.
TAB		TAB igual que el LCURSOR, pero puede ser utilizado en sentencias LPRINT: Válido solamente en modo TEXT (1) Expresión LPRINT TAB; . . . (2) Expresión TAB;
TEST	TE. TES.	Comprueba el impresor dibujardo cuadrados, de 5mm por 5mm en cada color.
TEXT	TEX.	Especificación del modo TEXT. Este modo se utiliza para imprimir caracteres y numerales.

SHARP CORPORATION

OSAKA, JAPAN

Printed in Japan
2F4T(TINSS3523CCZZ)