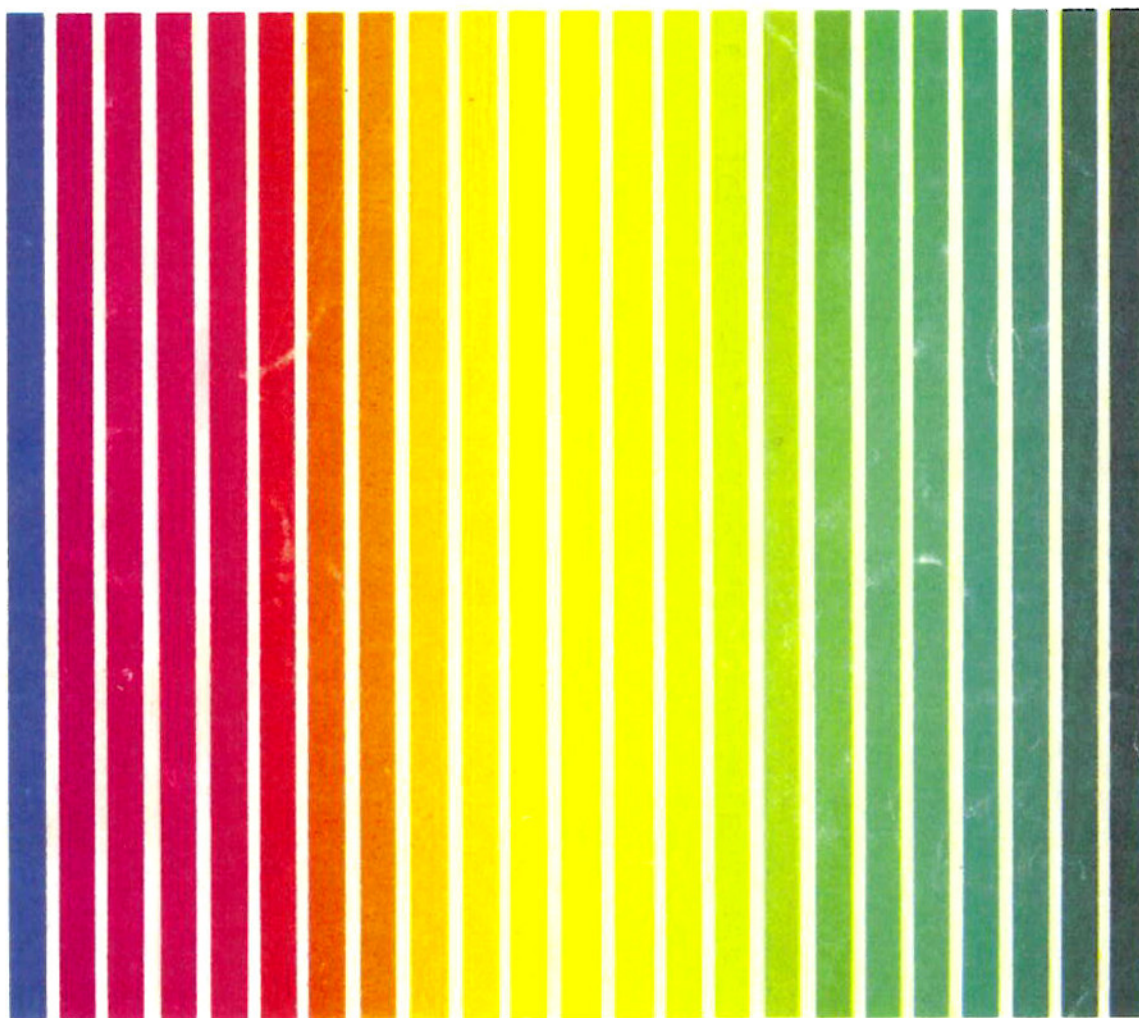


MICRO COMPUTADORES



**MANUAL DE REFERENCIA
ATARI BASIC**

(BASIC - MAN)



COMPONENTES ELECTRONICOS S.A.

MANUAL DE REFERENCIA
BASIC

MICROCOMPUTADORES



COELSA S. A.
1985

CODIGO DE ERRORES

CODIGO	MENSAJE
2	Memoria insuficiente
3	Error de valor
4	Demasiadas variables
5	Error de largo de string
6	Falta de datos
7	Número mayor que 32767
8	Error de instrucción de entrada
9	Error de dimensionamiento (DIM) de string
10	Stack de parámetros excedido
11	Rango de coma flotante excedido
12	Línea inexistente
13	Falta instrucción FOR
14	Línea demasiado larga
15	Línea GOSUB o FOR eliminada
16	Error de retorno (RETURN)
17	Línea incomprensible
18	Carácter de string inadmisible

NOTA: Los siguientes son errores de entrada/salida (INPUT/OUTPUT) que pueden resultar de la interacción con dispositivos periféricos como: unidades de disco, impresores, etc.

19	Programa de carga (LOAD) demasiado largo
20	Número de dispositivo (#) mayor que 7
21	El programa archivado no está codificado (SAVE/LOAD)
128	Detención por BREAK
129	Bloque de control (IOCB) ya abierto
130	Dispositivo inexistente
131	Bloque de control (IOCB) abierto para salida solamente
132	Comando inválido
133	El dispositivo o archivo no está abierto
134	Número de bloque de control (IOCB) inválido
135	Bloque de control (IOCB) abierto para entrada solamente
136	Fin de archivo (EOF)
137	Registro trunco
138	Tiempo del dispositivo excedido
139	El dispositivo no responde (NAK)
140	Error del bus serial
141	Cursor fuera de rango
142	Campo de datos del bus serial excedido
143	Error en comprobación del campo de datos del bus serial
144	Error en el dispositivo periférico
145	Error en comparación de lectura después de grabación
146	Función inexistente
147	Memoria (RAM) insuficiente
160	Error de número de unidad de disco
161	Demasiados archivos abiertos
162	Disco lleno
163	Error irreparable de dato del sistema en entrada/salida (INPUT/OUTPUT)
164	Número de archivo no corresponde
165	Error de nombre de archivo
166	Error de longitud de datos en POINT
167	Archivo protegido (LOCKED)
168	Comando inválido
169	Directorio lleno (64 archivos)
170	Archivo no ubicable
171	Comando POINT Inválido
172	Intento de combinar archivos de DOS I y DOS II
173	Sector con defectos durante formateo

CONTENIDO

PREFACIO	
1	INFORMACION GENERAL
	Terminología 1
	Notaciones especiales usadas en este manual 3
	Abreviaciones usadas en este manual 4
	Modos de operación 5
	Teclas de funciones especiales 6
	Operadores aritméticos 7
	Precedencia de operadores 8
	Funciones incluidas 8
	Gráficos 9
	Sonidos y juegos 9
	Pantalla envolvente y repetición de teclado 9
	Mensajes de error 9
2	COMANDOS
	BYE 10
	CONT 10
	END 10
	LET 11
	LIST 11
	NEW 11
	REM 12
	RUN 12
	STOP 12
3	CARACTERISTICAS DE EDICION
	Edición de pantalla 13
	Tecla CONTROL (CTRL) 13
	Tecla SHIFT 13
	Funciones de dos teclas 14
	Teclas de control de cursor 14
	Teclas que se usan con la tecla CONTROL 14
	Teclas que se usan con la tecla SHIFT 14
	Teclas de funciones especiales 14
	Tecla Alto (BREAK) 14
	Tecla Escape (ESC) 14
4	SENTENCIAS DE PROGRAMA
	FOR/NEXT/STEP 15
	GOSUB/RETURN 16
	GOTO 17
	IF/THEN 18
	ON/GOSUB 20
	ON/GOTO 20
	POP 21
	RESTORE 21
	TRAP 22

5 COMANDOS DE ENTRADA/SALIDA

Dispositivos de Entrada/Salida	23
CLOAD	24
CSAVE	24
DOS	25
ENTER	25
INPUT	26
LOAD	26
LPRINT	26
NOTE	27
OPEN/CLOSE	27
POINT	28
PRINT	29
PUT/GET	29
READ/DATA	29
SAVE	30
STATUS	30
X10	30
Concatenación de programas	31

6 BIBLIOTECA DE FUNCIONES

Funciones aritméticas	34
ABS	34
CLOG	34
EXP	34
INT	34
LOG	35
RND	35
SGN	35
SQR	35
Funciones Trigonométricas	35
ATN	35
COS	36
SIN	36
DEG/RAD	36
Funciones de propósitos especiales	36
ADR	36
FRE	36
PEEK	37
POKE	37
USR	37

7 STRINGS

ASC	39
CHR\$	39
LEN	40
STR\$	40
VAL	40
Manejo de strings	41

8 AGRUPAMIENTOS Y MATRICES

DIM	43
CLR	45

9 MODOS Y COMANDOS GRAFICOS

GRAPHICS	46
----------	----

Modos gráficos	47
Modo 0	47
Modos 1 y 2	48
Modos 3, 5 y 7	49
Modos 4 y 6	49
Modo 8	49
Modos 9, 10 y 11	50
Modos 12 y 13	51
Modos 14 y 15	53
COLOR	53
DRAWTO	54
LOCATE	54
PLOT	54
POSITION	55
GET/PUT	55
SETCOLOR	55
X10 (Aplicación especial: relleno)	59
Asignando colores a los modos de texto	60
Caracteres de control gráfico	62
10 SONIDOS Y CONTROLADORES DE JUEGO	
SOUND	63
PADDLE	65
PTRIG	65
STICK	65
STRING	66
11 TECNICAS DE PROGRAMACION AVANZADA	
Conservación de memoria	67
Programando en lenguaje de máquina	68
APENDICE A	
PALABRAS RESERVADAS DEL BASIC	A-1
APENDICE B	
MENSAJES DE ERROR	B-1
APENDICE C	
JUEGO DE CARACTERES ATASCII	C-1
APENDICE D	
MAPAS DE MEMORIA 400/800; XL	D-1
APENDICE E	
FUNCIONES DERIVADAS	E-1
APENDICE F	
VERSIONES IMPRESAS DE LOS CARACTERES DE CONTROL	F-1
APENDICE G	
GLOSARIO	G-1
APENDICE H	
PROGRAMAS DE APLICACION	H-1
APENDICE I	
UBICACIONES DE MEMORIA	I-1

Este manual supone que el usuario ha leído "BASIC Atari — Una Guía de Auto-enseñanza" o algún otro libro de BASIC. Este manual no pretende "enseñar" BASIC. Es una guía de referencia para los comandos, sentencias, instrucciones, funciones y aplicaciones especiales de BASIC Atari.

Los programas y los ejemplos de programación parciales que se usan en este manual son fotocopias de listados impresos en un impresor de línea. Algunos de los símbolos especiales del juego de caracteres Atari no aparecen al igual en el impresor; o simplemente no aparecen. Los ejemplos del texto, que se eligieron para ilustrar una función en particular, no representan necesariamente técnicas de buena programación.

Cada una de las secciones contiene grupos de comandos, funciones o sentencias que tratan de algún aspecto particular del BASIC Atari. Así, la sección 9 contiene todas las sentencias relacionadas con las capacidades gráficas únicas de Atari. Los apéndices incluyen referencias rápidas a términos, mensajes de error, palabras claves del BASIC, ubicaciones de memoria, y el juego de caracteres ATASCII.

Como no existe una aplicación específica para el Sistema de Computación Personal Atari, este manual se orienta a aplicaciones generales y al usuario general. El apéndice H contiene programas que ilustran algunas de las posibilidades del sistema Atari.

INFORMACION GENERAL

Esta sección explica la terminología BASIC, notaciones especiales y abreviaciones usadas en este manual y las teclas especiales de los teclados de los Sistemas de Computación Personal ATARI. También hace referencia a otras secciones en las que los comandos BASIC dicen relación con aplicaciones específicas.

TERMINOLOGIA

BASIC: Beginner's All - purpose Symbolic Instruction Code (Código de Instrucción Simbólica de todo propósito para Principiantes).

PALABRA CLAVE BASIC: Cualquier palabra reservada que es "legal" en el lenguaje BASIC. Puede usarse en una sentencia, como un comando, o para cualquier otro propósito. (Vea el apéndice A para una lista de todas las "palabras reservadas" o palabras claves de BASIC Atari.

SENTENCIA BASIC: Generalmente comienza con una palabra clave, como LET, PRINT, o RUN. El término "instrucción" es equivalente a la palabra "sentencia" en Basic.

CONSTANTE: Una constante es un valor expresado como número y no representado por el nombre de una variable. Por ejemplo, en la sentencia $X=100$, X es una variable y 100 una constante (Vea variable).

CADENA DE COMANDOS: Comandos o sentencias de programas múltiples, puestos en la misma línea numerada y separados por doble punto.

EXPRESION: Expresión es cualquier combinación legal de variables, constantes, operadores y funciones usadas en conjunto para computar un valor. Las expresiones pueden ser ya sea aritméticas, lógicas, o string *1.

FUNCION: Una función es un cálculo incluído en el computador de modo que pueda ser llamada por el programa del usuario. Una función **NO ES UN SENTENCIA:** es parte de una expresión. En realidad es una subrutina usada para calcular un valor que enseguida es entregado al programa principal cuando vuelve la sub-rutina. COS (coseno), RND (Aleatorio), FRE (Memoria disponible), e INT (Integro) son ejemplos de funciones. En muchos casos el valor simplemente se asigna a una variable (se almacena en una variable) para uso posterior. En otros casos puede que sea impreso de inmediato en la pantalla. Vea la sección 6 para más información sobre funciones. Los siguientes son ejemplos de funciones tal como podrían aparecer en un programa:

10 PRINT RND(0)

(Imprime el número aleatorio)

10 X=100+COS(45)

(Agregar el valor entregado a 100 y almacenar el total en la variable X)

LINEA LOGICA: Una línea lógica consiste de una a tres líneas físicas y termina con **RETURN** o automáticamente cuando se ha alcanzado el límite máximo de una línea lógica. Cada línea numerada en un programa BASIC consiste de una línea lógica cuando se despliega en la pantalla. Al ingresar una línea que es más larga que una línea física, el cursor automáticamente irá al comienzo de la siguiente línea física cuando alcance el final de la línea física actual. Si no se ingresa **RETURN**, ambas líneas físicas formarán parte de la misma línea lógica.

*1. El término string se ha traducido por cadena, cuerda, variable alfa-numérica, variable literal, etc. Dada la diversidad de traducciones que ha tenido y habiéndose mantenido por otro lado la misma palabra string en la jerga de computación, preferimos usar esta última forma.

OPERADOR: Los operadores se usan en expresiones. Los operadores incluyen adición (+), sustracción (—), multiplicación (X), división (/), exponenciación (^), mayor que (>), menor que (<), igual (=), mayor o igual (>=), menor o igual (<=), diferente (<>). Las palabras claves lógicas AND, NOT y OR también son operadores. Los operadores + y — también pueden usarse como operadores unitarios; por ejemplo, —3. No ponga varios operadores unitarios uno a continuación de otro; por ejemplo: ——3, ya que el computador los interpretará en forma incorrecta.

LINEA FISICA: Una línea de caracteres tal como se despliega en la pantalla del televisor.

STRING: Un string es un grupo de caracteres encerrados entre comillas; "ABRA CADABRA" es un string. También lo son "ATARI FABRICA COMPUTADORES MAGNIFICOS" y "123456789". Un string se parece mucho a una constante; al igual que ésta, puede almacenarse en una variable. Una variable string se diferencia en que su nombre debe terminar con un carácter \$. Por ejemplo, el string "ATARI 800" puede asignarse a una variable llamada A\$ usando el LET (opcional); así:

10 LET A\$="ATARI 800" (Note las comillas)

10 A\$="ATARI 800" (LET es opcional; las comillas son obligatorias)

Las comillas no pueden usarse al interior de un string. Sin embargo, puede omitirse la comilla de cierre si constituye el último carácter de una línea lógica (Vea la Sección 7—STRINGS)

VARIABLES: Una variable es el nombre para una cantidad numérica, o de otro tipo, que puede (o puede no) variar. Los nombres de las variables pueden tener hasta 120 caracteres.

Sin embargo, toda variable debe comenzar con una letra alfabética y puede contener solamente letras mayúsculas y dígitos numéricos.

Es recomendable no utilizar palabras claves como nombres de variables o como la primera parte del nombre de una variable, ya que puede ser interpretado erróneamente. Ejemplos de como almacenar un valor en una variable:

```
LETC123DVB=1.234
LETVARIABLE112=267.543
LETA=1
LETQUINTO=6.5
LETESTENUMERO 59.809
```

NOTA: LET es opcional y puede omitirse.

LIMITE DE NOMBRES DE VARIABLE: ATARI BASIC limita al usuario a no más de 128 nombres de variables. Para obviar este problema, use elementos individuales de un agrupamiento en vez de usar nombres de variables independientes. BASIC mantiene todas las referencias a una variable que haya sido eliminada de un programa y su nombre permanecerá siempre en la tabla de nombres de variables.

Si en la pantalla aparece un mensaje de ERROR-4 (demasiadas variables), use el siguiente procedimiento para hacer lugar para nuevos nombres de variables.:

```
LIST esparchivo
NEW
ENTER esparchivo
```

El LIST esparchivo escribe la versión no codificada del programa en disco o cassette. NEW limpia el programa y el área de variables. Enseguida se reingresa el programa, se recodifica y se construye una nueva tabla de variables. (La versión codificada es el formato interno del BASIC Atari. La versión no codificada está en ATASCII y es la versión que se despliega en la pantalla).

AGRUPAMIENTOS Y VARIABLES DE GRUPO: Un agrupamiento es una lista de lugares en los que se pueden archivar datos para uso futuro. Cada uno de estos lugares se llama elemento, y el agrupamiento completo o cualquier elemento constituye una variable de grupo. Así por ejemplo, definamos "Agrupamiento A" con 6 elementos.

Se hace referencia a estos elementos usando variables indexadas tales como A(2), A(3), A(4), etc. En cada elemento puede almacenarse un número, lo que puede hacerse elemento por elemento (usando la sentencia LET), o como parte de un bucle FOR/NEXT (Vea el capítulo 8).

NOTA: Jamás deje espacios entre el número entre parentesis que corresponde al elemento y el nombre del agrupamiento.

CORRECTO	INCORRECTO
A(23)	A (23)
GRUPO(3)	GRUPO (3)
X123(38)	X123 (38)

NOTACIONES ESPECIALES QUE SE USAN EN ESTE MANUAL

FORMATO DE LINEA: El formato de una línea en un programa BASIC incluye un número de línea (abreviado como nolínea) al comienzo de la línea, seguido de la palabra clave de la sentencia; a continuación el cuerpo de la sentencia o instrucción y finalmente un terminal de comando (tecla **RETURN**).

En un programa real, estos cuatro elementos podrían verse así:



Pueden escribirse varias sentencias en la misma línea siempre que estén separadas por doble punto (:). Vea IF/THEN en la Sección 5 y en la Sección 11.

LETRAS MAYUSCULAS: Se usan en este libro para indicar palabras claves que deben ser ingresadas exactamente en la misma forma por el usuario. Los caracteres en video-inverso no servirán salvo el caso del comando RUN. He aquí algunos ejemplos:

PRINT INPUT LIST END GOTO GOSUB FOR NEXT IF

LETRAS MINUSCULAS: En este manual, las letras minúsculas se usan para indicar diversas clases de ítem que pueden ser usados en un programa, tales como variables (var), expresiones (exp), etc. Las abreviaciones usadas para este tipo de ítems se indican en la tabla 1.1.

ITEMS ENTRE PARENTESIS CUADRADOS: Los paréntesis cuadrados, [], contienen ítems opcionales que pueden usarse, pero no son obligatorios. Si un ítem encerrado en paréntesis cuadrado es seguido por tres puntos [exp,...] significa que cualquier número de expresiones puede ingresarse pero no se requiere ninguna.

ITEMS ALINEADOS VERTICALMENTE ENTRE PARENTESIS DE LLAVE: Los items almacenados verticalmente en paréntesis de llave indican que cualquiera de los items indicados pueden usarse, pero que solo se permite uno a la vez. En el ejemplo que sigue, escriba GOTO o GOSUB.

$$100 \quad \left\{ \begin{array}{l} \text{GOTO} \\ \text{GOSUB} \end{array} \right\} \quad 2000$$

ABREVIACIONES DE COMANDOS EN LOS ENCABEZAMIENTOS: Si algún comando o alguna sentencia tiene asociada alguna forma de abreviación, la abreviación se colocará siguiendo la expresión completa del comando del encabezamiento; por ejemplo: LIST (L.).

ABREVIACIONES QUE SE USAN EN ESTE MANUAL:

La siguiente tabla explica las abreviaciones que se usan en toda la extensión de este manual:

TABLA 1.1 ABREVIACIONES

vara - VARIABLE ARITMETICA:

Una ubicación en la cual se almacena un valor numérico. Los nombres de variables pueden tener de uno a 120 caracteres alfanuméricos; sin embargo deben iniciarse con un carácter alfabético, y todos los caracteres deben ser mayúsculas y no inversos.

vars - VARIABLE STRING:

Una ubicación en la cual puede almacenarse un string, una cadena de caracteres. Se aplican las mismas reglas de nombre correspondiente a vara, excepto que el último carácter del nombre de la variable debe ser \$. Las variables string pueden llevar sub - índices. Vea la Sección 7, STRINGS.

varm - VARIABLES MATRICIALES

También llamadas variables indexadas. Es un elemento de un agrupamiento o de una matriz. El nombre de la variable para el agrupamiento o matriz como un todo puede ser cualquier nombre de variable legal tal como A, X, Y, ZIP, o K. La variable indexada (nombre de un elemento en particular) se inicia con el nombre de la matriz, y enseguida emplea un número, una variable o expresión entre paréntesis siguiendo inmediatamente la variable de grupo o matriz. Por ejemplo: A(Línea), B(1), A(X+1),

var — VARIABLE:

Cualquier variable. Puede ser varm, vara, o vars.

opa — OPERADOR ARITMETICO.

opl — OPERADOR LOGICO.

expa — EXPRESION ARITMETICA:

Generalmente compuesta por una variable, función, constante, o dos expresiones aritméticas separadas por un operador aritmético.

expl — EXPRESION LOGICA:

Generalmente compuesta de dos expresiones aritméticas o string, separadas por un operador lógico. Una expresión de este tipo da como resultado ya sea un 1 (verdadero lógico) o un 0 (falso lógico).

Por ejemplo, la expresión $1 < 2$ es igual a 1 (verdadero) mientras que la expresión "LIMON"="NARANJA" es igual a cero (falso) ya que los dos strings no son iguales.

exps — EXPRESION STRING:

Puede consistir de una variable string, de una constante, o de una función que entrega un valor string.

exp — Una expresión, ya sea exps o expa.

no línea NUMERO DE LINEA:

Una constante que identifica una línea particular dentro de un programa del modo BASIC diferido. Debe corresponder a un entero entre 0 y 32.767. La numeración de las líneas determina el orden de la ejecución del programa.

data — DATO ATASCII:

Cualquier carácter ATASCII excluyendo comas y retornos de carro. (Vea apéndice C.).

esparchivo — ESPECIFICACION DE ARCHIVO:

Una expresión string que se refiere a un dispositivo tal como el teclado o un archivo de disco. Contiene información acerca del tipo de dispositivo de entrada y salida, su número, un doble punto, un nombre de archivo opcional, y un extensor de nombre de archivo opcional. (Vea OPEN, en la Sección 5).

EJEMPLO DE ESPARCHIVO: "D1:NATALIA.ED".

MODOS DE OPERACION**MODO DIRECTO:**

No utiliza número de línea y ejecuta la instrucción inmediatamente después de haberse presionado la tecla **RETURN**.

MODO DIFERIDO:

Emplea números de línea y demora la ejecución de la o las instrucciones hasta que se ingrese el comando RUN.

MODO DE EJECUCION:

También llamado ocasionalmente modo Run. Una vez ingresado el comando RUN, cada línea de programa se procesa y se ejecuta.

MODO MEMO PAD:

Un modo no programable que permite al usuario experimentar con el teclado o dejar un mensaje sobre la pantalla. (Sólo existe en los modelos 400 y 800).

MODO AUTOCOMPROBACION

Un modo programado en los modelos XL, que permite al usuario verificar el correcto funcionamiento de teclado, memoria, etc.

TECLAS DE FUNCIONES ESPECIALES



TECLA DE VIDEO INVERSO, o "TECLA DE LOGO ATARI" (modelos 400 y 800). Presionando esta tecla el texto que se escriba a continuación se invierte sobre la pantalla (texto obscuro sobre fondo blanco). Presione esta tecla por segunda vez para volver a texto normal.

CAPS/LOWR

CAPS

TECLA DE MINUSCULAS: Presionando esta tecla los caracteres que se digiten a continuación, aparecerán en la pantalla en minúsculas. Para volver a caracteres de mayúsculas, presione simultáneamente las teclas **SHIFT** y **CAPS/LOWR** (Computadores modelo 400 y 800), o simplemente la tecla **CAPS** (Computadores XL).

ESC

TECLA ESCAPE: El presionar esta tecla permite ingresar un comando para su posterior ejecución en un programa.

Por ejemplo: Para limpiar la pantalla debe ingresarse:

```
10 PRINT " ESC CTRL CLEAR "
y presionar RETURN .
```

La tecla escape también se utiliza en conjunto con otras teclas para imprimir caracteres de control gráficos especiales. Vea el Apéndice F para las teclas específicas y su representación de carácter de pantalla.

BREAK

TECLA BREAK (ALTO): Presionando esta tecla durante la ejecución de un programa, tiene como consecuencia la detención de la ejecución. La ejecución continua escribiendo CONT seguido de **RETURN** .

RESET

TECLA SYSTEM RESET (REPOSICION DEL SISTEMA): Similar a **BREAK** en el sentido que la presión sobre esta tecla detiene la ejecución del programa. Además lleva el despliegue de pantalla al modo gráfico 0, limpia la pantalla y repone los márgenes y otros valores de variables a sus valores asumidos.

TAB

TECLA DE TABULACION: Presione simultáneamente las teclas **SHIFT** y **TAB** para poner un punto de tabulación. Para eliminar un punto de tabulación, presione simultáneamente las teclas **CTRL** y **TAB** . Usada por si sola, **TAB** hace avanzar el cursor a la siguiente posición de tabulación. En el modo diferido, disponga y elimine puntos de tabulación precediendo lo anterior con un número de línea, el comando PRINT, comillas y presionando la tecla **ESC** .

Ejemplos:

```
100 PRINT " ESC SHIFT SET-CLR-TAB "
200 PRINT " ESC CTRL SET-CLR-TAB "
```

Los puñtos de tabulación asumidos están ubicados en las columnas 7, 15, 23, 31 y 39.

INSERT

TECLA DE INSERCIÓN: Presione simultáneamente las teclas **SHIFT** e **INSERT** para insertar una línea. Para insertar un solo carácter, presione simultáneamente las teclas **CTRL** e **INSERT** .

DELETE

TECLA DE ELIMINACIÓN: Presiones simultáneamente las teclas **SHIFT** y **DELETE** para eliminar una línea. Para eliminar un solo carácter, presione simultáneamente las teclas **CTRL** y **DELETE** .

DELETE	TECLA DE RETROCESO: Al presionar esta tecla se reemplaza el carácter a la izquierda del cursor con un espacio y retrocede el cursor en un espacio.
CLEAR	TECLA BORRADOR: El presionar esta tecla mientras se mantiene presionado SHIFT o CTRL limpia la pantalla y ubica el cursor en el rincón superior izquierdo.
RETURN	TECLA RETURN: Terminal para indicar el final de una línea BASIC. El presionar esta tecla hará que una línea numerada sea interpretada y agregada al programa BASIC en la memoria. Una línea no numerada (en modo directo) se interpreta y se ejecuta de inmediato. Cualquier variable que aparezca será ingresada en la tabla de variables.

OPERADORES ARITMETICOS

El Sistema de Computación Personal Atari usa cinco operadores aritméticos:

- + adición (también el "más" unitario; por ejemplo +5).
- substracción (también el "menos" unitario; por ejemplo —5).
- X multiplicación
- / división
- ^ exponenciación

OPERADORES LOGICOS

Los operadores lógicos son de dos tipos: unitarios y binarios. El operador unitario es NOT. Los operadores binarios son:

- AND Y lógico
- OR O lógico

Ejemplos:

10 IF A=12 AND T=0 THEN PRINT "BIEN"	Ambas expresiones deben ser verdaderas para que se imprima BIEN.
10 A=(C > 1) AND (N < 1)	Si ambas expresiones son verdaderas, A=+1; de otro modo A=0.
10 A=(C+1) OR (N—1)	Si una de ambas expresiones es verdadera, A=31; de otro modo A=0.
10 A= NOT (C+1)	Si la expresión es falsa, A=+1; de otro modo A=0.

Los demás operadores binarios son relacionales.

- < La primera expresión es menor que la segunda expresión.
- > La primera expresión es mayor que la segunda.

- = Ambas expresiones son iguales entre si.
- <= La primera expresión es menor o igual a la segunda.
- >= La primera expresión es mayor o igual a la segunda.
- < > Las dos expresiones no son iguales entre si.

Estos operadores se emplean con mayor frecuencia en las sentencias IF/THEN y la aritmética lógica.

PRECEDENCIA DE OPERADORES

Las operaciones al interior de los paréntesis más internos son las que se realizan primero y a continuación se procede hacia afuera a los niveles siguientes. Cuando un par de paréntesis se encuentra encerrado por otro par, se dice que está "anidado". Las operaciones al mismo nivel de anidamiento se realizan en el siguiente orden:

Precedencia mayor

<, >, =, <=, >=, <>

Operadores relacionales usados en expresiones de string. Tienen la misma precedencia y se realizan de izquierda a derecha.

—

Menos unitario

^

Exponenciación

*, /

Multiplicación y división tienen el mismo nivel de precedencia y se realizan de izquierda a derecha.

+, -

Adición y sustracción tienen el mismo nivel de precedencia y se realizan de izquierda a derecha.

<, >, =, <=, >=, <>

Operadores relacionales en expresiones numéricas tienen el mismo nivel de precedencia y se efectúan de izquierda a derecha.

NOT

Operador unitario.

AND

Y lógico.

OR

O lógico

Precedencia menor

FUNCIONES INCLUIDAS

La sección titulada BIBLIOTECA DE FUNCIONES explica las funciones aritméticas especiales incluidas en BASIC Atari.

GRAFICOS

Los gráficos Atari comprenden doce modos en los modelos 400 y 800 y 16 modos en los modelos XL. Los comandos han sido diseñados para dar el máximo de flexibilidad en la elección de color y variedad de dibujo. La Sección 9 explica cada comando y da ejemplos de las muchas formas de usar cada uno de ellos.

SONIDO Y CONTROLADORES DE JUEGOS

El computador Personal Atari es capaz de emitir una gran variedad de sonidos, incluyendo explosiones simuladas, música electrónica, y "bufidos". La Sección 10 define los comandos para usar la función SOUND y para los controladores de paleta, bastón y teclado.

PANTALLA ENVOLVENTE Y TECLADO DE REPETICION

El sistema de Computación Personal ATARI tiene pantalla envolvente dando así mayor flexibilidad al cursor. También permite al usuario ingresar una tecla en avance. Si el usuario presiona y mantiene una tecla, ésta comenzará a repetirse. (En los modelos XL el tiempo al cabo del cual se inicia la repetición y la velocidad de ésta son programables).

MENSAJES DE ERROR

Si llegara a ocurrir un error de ingreso de dato, la pantalla mostrará una reimpresión de la línea precedida del mensaje ERROR y recalcando el carácter en discordia. Una vez corregido el carácter en la línea original, elimine la línea que contenga el ERROR, antes de presionar RETURN. El apéndice B contiene una lista de todos los mensajes de error y sus respectivas definiciones.

Siempre cuando el cursor (☐) esté sobre la pantalla, el computador está listo para aceptar ingresos. Escriba el comando (ya sea en el modo Directo o el Diferido), y presione **RETURN**. Esta sección describe los comandos empleados para limpiar la memoria del computador y también otros comandos de control útiles.

Los comandos explicados en esta sección son los siguientes:

BYE	NEW
CONT	REM
END	RUN
LET	STOP
LIST	

BYE (B.)

Formato: BYE

Ejemplo: BYE

La función normal de un comando BYE en los modelos 400 y 800 es egresar de BASIC y poner el computador en el modo de borrador (Memo Pad). Esto permite al usuario experimentar con el teclado o dejar mensajes sobre la pantalla sin alterar un programa BASIC que pueda encontrarse en memoria. Para retornar a BASIC, presione RESET. En los modelos XL, BYE, lleva a la prueba inicial (Self Test o Autocomprobación) El programa en memoria no se conserva.

CONT (CON.)

Formato: CONT

Ejemplo: CONT

Escribiendo este comando seguido de **RETURN** produce la reanudación en la ejecución de un programa. Si el programa encuentra un **BREAK**, STOP, o END se detendrá hasta que se ingrese CONT RETURN. La ejecución se reanuda a partir del número de línea siguiente a la sentencia en la cual el programa se detuvo.

NOTA:

Si la sentencia en la cual el programa se detuvo tiene otros comandos en la misma línea numerada, que no se hayan ejecutado al instante de **BREAK**, STOP, o END, ellos no serán ejecutados. A consecuencia de CONT la ejecución se reanuda en la siguiente línea numerada. Un bucle puede ser ejecutado incorrectamente si el programa se detiene antes de completar la ejecución del bucle.

Este comando no produce efecto en el modo de programa diferido.

END

Formato: END

Ejemplo: 1000 END

Este comando finaliza la ejecución de un programa y se usa en el modo diferido. En BASIC Atari no se requiere un END al final de un programa. Cuando se alcanza el final de un programa, el BASIC Atari automáticamente cierra todos los archivos y apaga los sonidos que pueda haber. END también puede utilizarse en el modo directo para cerrar archivos o acallar sonidos.

LET (LE.)

Formato: [LET]var = exp

Ejemplo: LET X = 3.142*16
LET X = 2

Esta sentencia es opcional en la definición de variables. Igualmente puede dejarse fuera de la sentencia. Puede usarse, sin embargo, para igualar el nombre de una variable a un valor.

LIST (L.)

Formato: LIST [nolínea[,no línea]]
LIST [esparchivo[,nolínea[,nolínea]]]

Ejemplos: LIST
LIST 10
LIST 10,100
LIST "P:",20,100
LIST "P"
LIST "D:DEMO.LST"

Este comando hará que el computador despliegue la versión fuente de todas las líneas en memoria, si se da sin número de líneas, o la o las líneas específicas. Por ejemplo, LIST 10,100 **RETURN** desplegará las líneas de 10 a 100 sobre la pantalla. Si el usuario no ha escrito las líneas en el computador siguiendo el orden numérico, un LIST las pondrá en orden automáticamente.

Escribiendo L. "P: hará que el programa residente en memoria sea impreso por el impresor.

LIST puede utilizarse en el modo diferido como parte de una rutina para atrapar errores (Vea TRAP en la Sección 4).

El comando LIST se usa igualmente para grabar programas en cinta cassette. Se utiliza el segundo formato y se ingresa una esparchivo. (Vea la Sección 5 para más detalles acerca de los dispositivos periféricos). Si se desea listar todo el programa en cinta, no es necesario especificar números de línea.

Ejemplos: LIST "C1"
1000 LIST "C1"

NEW

Formato: NEW

Ejemplo: NEW

Este comando borra el programa almacenado en la memoria. Por ello, antes de escribir NEW, grabe con SAVE o con CSAVE todo programa para ser recuperado y usado más adelante.

NEW limpia la tabla interna de símbolos del BASIC, de modo que no queda ninguna definición de agrupamientos (Vea la Sección 8) ni strings (Vea la Sección 7); se usa en modo directo.

REM (R. o .ESPACIO)

Formato: REM texto

Ejemplo: 10 REM RUTINA PARA CALCULAR X

Este comando y el texto que lo siga, solamente sirven de información al usuario. El computador lo ignora. Sin embargo, se incluye en un LIST conjuntamente con las demás líneas numeradas. Toda sentencia que esté en la misma línea numerada y que se presente con posterioridad a una sentencia REM será ignorada.

RUN (RU.)

Formato: RUN [Esparchivo]

Ejemplos: RUN

RUN "D:MENU"

Este comando hace que el computador inicie la ejecución de un programa. Si no se especifica un archivo, el programa residente en RAM inicia su ejecución. Si se incluye un esparchivo, el computador encuentra el programa especificado y codificado del archivo especificado y lo ejecuta.

Todas las variables se colocan en cero y todos los archivos y periféricos abiertos se cierran. Todos los agrupamientos, strings y matrices se eliminan y todos los sonidos se apagan. A no ser que haya usado el comando TRAP, se desplegará un mensaje de error si se detecta un error durante la ejecución, con lo cual el programa se detiene.

RUN puede usarse en el modo Diferido.

Ejemplos: 10 PRINT "UNA Y OTRA VEZ DE NUEVO"
20 RUN

Escriba RUN y presione **RETURN** ; para terminar, presione **BREAK** .

Para iniciar la ejecución de un programa en un punto que no sea el primer número de línea, escriba GOTO seguido del número de línea específico, enseguida presione RETURN.

STOP (STO.)

Formato: STOP

Ejemplo: 100 STOP

Al ejecutarse el comando STOP en un programa, BASIC despliega un mensaje STOPPED AT LINE (detenido en la línea) ————, termina con la ejecución del programa, y retorna al modo directo. El comando STOP no cierra archivos ni apaga sonidos, de modo que el programa puede reanudarse escribiendo simplemente CONT **RETURN** .

CARACTERISTICAS DE EDICION

Adicionalmente a las teclas de funciones especiales descritas en la sección 1, hay otras teclas de control de cursor que permiten acceder a capacidades de edición inmediatas. Estas teclas se usan en conjunto con las teclas **SHIFT** o **CTRL**

En esta sección se describen las siguientes funciones de teclado:

CTRL	CTRL INSERT	CTRL 1
SHIFT	CTRL DELETE	CTRL 2
CTRL ↑	SHIFT INSERT	CTRL 3
CTRL ↓	SHIFT DELETE	BREAK
CTRL →	SHIFT CAPS/LOWR	ESC
CTRL ←		

EDICION DE PANTALLA

El teclado y la pantalla están combinados en forma lógica para un modo de operación conocido como edición de pantalla, cada vez que se complete un cambio en la pantalla debe aprimirse la tecla **RETURN**. De otro modo el cambio no se produce en el programa en la memoria.

Ejemplo:

```
10 REM PRESIONE [ RETURN]DESPUES DE EDITAR LA LINEA.
20 PRINT ;PRINT
30 PRINT "ESTA ES LA LINEA 1 SOBRE LA PANTALLA. "
```

Para eliminar la línea 20 del programa, escriba el número de línea y presione **RETURN**. Eliminando simplemente la línea de la pantalla no la elimina del programa.

Pantalla y teclado como dispositivos de entrada y salida se describen en la sección 5.

CTRL Tecla Control:

Presionando esta tecla en conjunto con las teclas de flecha se producen las funciones de control de cursor que permiten al usuario mover el cursor a cualquier lugar de la pantalla sin alterar ningún carácter ya escrito sobre ella. Otras combinaciones de teclas controlan la colocación y anulación de posiciones de tabulación, el detener y reiniciar el listado de un programa y los símbolos de control gráficos.

Presionar una tecla mientras se mantiene oprimida la tecla **CTRL** dará origen al símbolo superior izquierdo de aquellas teclas que tengan tres funciones.

SHIFT Tecla Shift:

Esta tecla se usa en conjunto con las teclas numéricas para desplegar los símbolos que se indican en la parte superior de estas teclas. Se usan también en conjunto con otras teclas para insertar y eliminar líneas, volver al despliegue de letras normales mayúsculas, desplegar los

símbolos de función sobre los operadores de resta, igual, adición y multiplicación como también los paréntesis cuadrados [], y el signo de interrogación ?.

FUNCIONES DE DOS TECLAS

Teclas de Control de Cursor

CTRL ↑	Mueve el cursor una línea física hacia arriba, sin alterar ni el programa ni el contenido de la pantalla.
CTRL →	Mueve el cursor en un espacio hacia la derecha sin alterar ni programa ni pantalla.
CTRL ↓	Mueve el cursor hacia abajo en una línea física sin alterar ni el programa ni la pantalla.
CTRL ←	Mueve el cursor un espacio hacia la izquierda, sin cambiar ni el programa ni la pantalla.

Al igual que las demás teclas del teclado Atari, mantener las teclas de control de cursor apretadas por un rato, hace que las teclas comiencen a repetirse.

Teclas que se usan con **CTRL**

CTRL INSERT	Inserta el espacio correspondiente a un carácter.
CTRL DELETE	Elimina un carácter o espacio.
CTRL 1	Detiene temporalmente y reinicia el despliegue de pantalla sin ruptura del programa.
CTRL 2	Hace sonar el zumbador.
CTRL 3	Indica fin de archivo.

Teclas usadas con **SHIFT**

SHIFT INSERT	Inserta una línea física
SHIFT DELETE	Elimina una línea física.
SHIFT CAPS/LOWR	Retorna los ingresos a pantalla a caracteres alfabéticos mayúsculas. (modelos 400 y 800).

TECLAS DE FUNCIONES ESPECIALES

BREAK	Detiene la ejecución o listado del programa, imprime READY sobre la pantalla, y despliega el cursor.
ESC	Permite poner los comandos que normalmente se usan en modo Directo en el modo Diferido; por ejemplo, en el modo Directo CTRL CLEAR limpia la pantalla. Para limpiar la pantalla en modo Diferido, escriba lo siguiente después del número de línea del programa. Presione primero ESC y a continuación CTRL y CLEAR simultáneamente. PRINT " ESC CTRL CLEAR "

Esta sección explica los comandos asociados con bucles, desvíos condicionales e incondicionales, trampas de error, sub-rutinas y sus ubicaciones. También explica la forma de acceder datos y el comando opcional usado para definir variables.

En la sección se describen los siguientes comandos:

FOR, TO, STEP/NEXT	IF/THEN	POP
GOSUB/RETURN	ON, GOSUB	RESTORE
GOTO	ON, GOTO	TRAP

FOR (F.), TO, STEP/NEXT (N.)

Formato: FOR vara = expa1 TO expa2 [STEP expa3]
NEXT vara

Ejemplos: FOR X = 1 TO 10
NEXT X
FOR Y = 10 TO 20 STEP 2
NEXT Y
FOR INDEX = Z TO 100 * Z
NEXT INDEX

Este comando establece un bucle y determina cuantas veces debe ejecutarse este bucle. La variable de bucle (vara) se inicializa al valor expa1.

Cada vez que se encuentra la sentencia NEXT vara, la variable del bucle se incrementa por la cantidad expa3 de la sentencia STEP. La expa3 puede ser un entero positivo o negativo, decimal, o un número fraccional. Si no existe el comando STEP expa3, el bucle se incrementa en uno. Cuando el bucle completa el límite definido por expa2, se detiene y el programa procede a la sentencia inmediatamente posterior a la sentencia NEXT; puede encontrarse en la misma línea o la siguiente.

Los bucles pueden estar anidados, uno dentro de otro. En este caso, el bucle más interno se completa primero, antes de llegar al bucle más externo. El siguiente ejemplo ilustra un programa de bucles anidados.

```
10 FOR X=1 TO 3
20 PRINT "BUCLE EXTERIOR"
30 Z=0
40 Z=Z+2
50 FOR Y=1 TO 5 STEP Z
60 PRINT "      BUCLE INTERIOR"
70 NEXT Y
80 NEXT X
90 REM END
```

Figura 4-1. Programa de Bucle anidado

En la figura 4-1, el bucle exterior completa tres pasos (X=1 a 3). Sin embargo, antes de que el bucle alcance la sentencia NEXT X, el programa pasa el control al bucle interior. Note

que la sentencia NEXT para el bucle interior debe preceder la sentencia NEXT del bucle exterior. En el ejemplo, la cantidad de pasos del bucle interior se determina por la sentencia STEP (STEP Z). En este caso, Z ha sido definido como 0, y a continuación redefinido como Z+2. Usando estos datos, el computador debe completar tres pasos a través del bucle interior antes de retornar al bucle exterior. La expa3 en la sentencia STEP igualmente podría haberse definido con el valor numérico 2.

La ejecución del programa se ilustra en la figura 4-2.

```

BUCLE EXTERIOR
  BUCLE INTERIOR
  BUCLE INTERIOR
  BUCLE INTERIOR
BUCLE EXTERIOR
  BUCLE INTERIOR
  BUCLE INTERIOR
  BUCLE INTERIOR
BUCLE EXTERIOR
  BUCLE INTERIOR
  BUCLE INTERIOR
  BUCLE INTERIOR

```

FIGURA 4-2. Ejecución de Bucles Anidados

Las direcciones de retorno para los bucles se ubican en un grupo especial de direcciones de memoria llamadas stack (pila). La información se empuja sobre la pila, y al usarla, se elimina del stack (vea POP).

```

GOSUB  ( GOS. )
RETURN ( RET. )

```

Formato: GOSUB nolínea
 nolínea
 RETURN

Ejemplo: 100 GOSUB 2000
 2000 PRINT "SUBROUTINA"
 2010 RETURN

Una subrutina * es un programa o rutina usada para computar un cierto valor, etc. Generalmente se emplea cuando una operación debe ser realizada varias veces dentro de una secuencia de programa usando el mismo o diferentes valores. El comando permite al usuario llamar la subrutina, en caso necesario. La última línea de una subrutina siempre debe contener una sentencia RETURN. La sentencia a RETURN retorna a la línea física que sigue a la sentencia GOSUB.

Al igual que el comando FOR/NEXT, precedente, el comando GOSUB/RETURN usa una pila para su dirección de retorno. Si a la subrutina no se le permite completar su ejecución normalmente, es decir, si hay un GOTO nolínea antes del RETURN, la dirección de GOSUB debe eliminarse del stack (vea POP) o podría crear futuros errores en la ejecución del programa.

*Generalmente, una sub-rutina puede hacer cualquier cosa que pueda hacerse en un programa. Se usa para ahorrar memoria y tiempo de ingreso de programa, y facilitar la lectura y corrección del programa.

Para prevenir la ejecución accidental de una subrutina (que normalmente sigue al programa principal), coloque una sentencia END antes de la subrutina. El siguiente programa demuestra el uso de subrutinas.

```

10 PRINT ">"
20 REM Ejemplo de GOSUB/RETURN
30 X=100
40 GOSUB 1000
50 X=120
60 GOSUB 1000
70 X=50
80 GOSUB 1000
90 END
1000 Y=3*X
1010 X=X+Y
1020 PRINT X,Y
1030 RETURN

```

Figura 4-3. Listado de un programa GOSUB/RETURN

En el programa precedente, la subrutina, que comienza en la línea 1000, se llama tres veces para computar e imprimir diferentes valores de X y de Y. La figura 4-4 ilustra los resultados de la ejecución de este programa.

400	300
480	360
200	150

Figura 4-4. Ejecución del programa GOSUB/RETURN

GOTO (G.)

Formato: $\left\{ \begin{array}{l} \text{GO TO} \\ \text{GOTO} \end{array} \right\} \text{expa}$

Ejemplos: 100 GOTO 50
500 GOTO (X+Y)

El comando GOTO es una sentencia de desvío incondicional igual que el comando GOSUB. Ambas transfieren de inmediato el control del programa al número de línea meta o expresión arbitraria. Sin embargo, el usar cualquier otra cosa que no sea una constante, hará difícil reenumerar el programa. Si el número de línea meta no existe, se produce un error. Toda sentencia GOTO que se desvíe hacia una línea precedente puede convertirse en un bucle sin fin. Las sentencias que siguen a una sentencia GOTO no se ejecutarán. Note que una sentencia de desvío condicional (vea IF/THEN) puede usarse para salir de un bucle GOTO. El siguiente programa ilustra dos aplicaciones del comando GOTO.


```

10 PRINT
20 PRINT :PRINT "UNO"
30 PRINT "DOS"
40 PRINT "TRES"
50 PRINT "CUATRO"
60 PRINT "CINCO"
65 GOTO 100
70 PRINT "#####"
80 PRINT "%%%%%%%%%"
90 PRINT "?????????"
95 END
100 PRINT "SEIS"
110 PRINT "SIETE"
120 PRINT "OCHO"
130 PRINT "NUEVE"
140 PRINT "DIEZ"
150 GOTO 70

```

Figura 4-5. Listado del programa GOTO

Una vez ejecutado, primero aparecerán los números del listado superior, seguidos de las tres líneas de símbolos. Los símbolos listados en las líneas 70, 80 y 90 serán ignorados temporalmente mientras el programa ejecuta el comando GOTO 100. Procede entonces a imprimir los números del "SEIS" AL "DIEZ", enseguida ejecuta la segunda sentencia GOTO que transfiere el control del programa de vuelta a la línea 70. (Esto es simplemente un ejemplo. Este programa igualmente podría escribirse sin recurrir a las sentencias GOTO.) El programa, al ejecutarse, se ve como sigue:

```

UNO
DOS
TRES
CUATRO
CINCO
SEIS
SIETE
OCHO
NUEVE
DIEZ
#####
%%%%%%%%%
?????????

```

Figura 4-6. Ejecucion del programa GOTO

IF / THEN

Formato: IF expa THEN $\left\{ \begin{array}{l} \text{nolínea} \\ \text{sentencia[:sentencia...]} \end{array} \right\}$

Ejemplos IF X=100 THEN 150.
 IF A\$="ATARI" THEN 200
 IF AA=145 AND BB=1 THEN PRINT AA,BB
 IF X=100 THEN X=0

La sentencia IF/THEN es una sentencia de desvío condicional. Este tipo de desvío se produce sólo si se cumplen determinadas condiciones. Estas condiciones pueden ser ya sea aritméticas o lógicas. Si la expa que sigue a la sentencia IF es verdadera (no cero), el progra-

ma ejecuta la parte THEN de la sentencia. Si, sin embargo, la expa es falsa (o un 0 lógico), el resto de la sentencia se ignora y el control del programa pasa a la línea de la siguiente numeración.

En el formato: IF expa THEN nolínea, nolínea debe ser una constante y no una expresión y especifica el número de línea al cual debe irse si la expresión es verdadera.

Si ocurren algunas sentencias después de THEN, separadas por dobles puntos, ellas serán ejecutadas, solamente si la expresión es verdadera. Algunas sentencias IF pueden anidarse en una misma línea. Por ejemplo:

```
100 IF X=5 THEN IF Y=3 THEN R=9:GOTO 200
```

La sentencia R=9: GOTO 200 se ejecutará sólo si X=5 e Y=3. La sentencia Y=3 se ejecutará si X=5.

El siguiente programa demuestra las sentencias IF/THEN.

```
5 GRAPHICS 0:PRINT :PRINT "DEMOSTRACION DE  'IF'"
10 ? "INGRESE A?":INPUT A
20 IF A=1 THEN 40:REM NINGUNA INSTRUCCION MULTIPLE, PUESTA AQUI,SE EJECUTA!
30 ? "A NO VALE 1. LA EJECUCION CONTINUA AQUI CUANDO LA EXPRESION ES FALSA."
40 IF A=1 THEN ? :? "A=1":? "SI, REALMENTE VALE 1.":REM INSTRUCCIONES MULTIPLES
AQUI,SOLO SE EJECUTARAN SI A=1!
50 ? :? "LA EJECUCION CONTINUA AQUI SI A<>1 O DESPUES DE APARECER EN LA PANTA-
- LLA 'SI, REALMENTE VALE 1'"
60 PRINT
70 GOTO 10
```

Figura 4-7 Programa IF/THEN

DEMOSTRACION DE 'IF'	
INGRESE A?2	
A NO VALE 1. LA EJECUCION CONTINUA AQUI CUANDO LA EXPRESION ES FALSA.	(ingresó 2)
LA EJECUCION CONTINUA AQUI SI A<>1 O DESPUES DE APARECER EN LA PANTA- LLA 'SI, REALMENTE VALE 1'	(ingresó 1)
INGRESE A?1	
A=1 SI, REALMENTE VALE 1.	
LA EJECUCION CONTINUA AQUI SI A<>1 O DESPUES DE APARECER EN LA PANTA- LLA 'SI, REALMENTE VALE 1'	

Figura 4-8 Ejecución del Programa IF/THEN.

ON /GOSUB / RETURN ON / GOTO

Formato: ON expa $\left\{ \begin{array}{l} \text{GOTO} \\ \text{GOSUB} \end{array} \right\}$ nolínea [,nolínea...]

Ejemplos: 100 ON X GOTO 200, 300, 400
 100 ON A GOSUB 1000, 2000
 100 ON SQR (X) GOTO 30, 10, 100

Nota: GOSUB y GOTO no pueden abreviarse.

Estas dos sentencias también constituyen desvíos condicionales al igual que la sentencia IF/THEN. Sin embargo, estas dos son más poderosas. La expa debe dar un número positivo que enseguida se redondea el entero positivo más próximo, hasta 255. Si el número resultante es 1, el control del programa pasa al primer nolínea en la lista que sigue al GOSUB o GOTO. Si el número resultante es 2, el control de programa pasa al segundo nolínea de la lista, y así sucesivamente. Si el número resultante es 0 o mayor que el número de nolíneas de la lista, las condiciones no se cumplen y el programa pasa a la sentencia siguiente que puede o puede no estar ubicada en la misma línea. Con ON/GOSUB, la subrutina seleccionada se ejecuta y enseguida el control pasa a la siguiente sentencia.

La siguiente rutina demuestra la sentencia ON/GOTO:

```
10 X=X+1
20 ON X GOTO 100,200,300,400,500
30 IF X>5 THEN PRINT "TERMINADO":END
40 GOTO 10
50 END
100 PRINT "TRABAJANDO EN LA LINEA 100 AHORA":GOTO 10
200 PRINT "TRABAJANDO EN LA LINEA 200 AHORA":GOTO 10
300 PRINT "TRABAJANDO EN LA LINEA 300 AHORA":GOTO 10
400 PRINT "TRABAJANDO EN LA LINEA 400 AHORA":GOTO 10
500 PRINT "TRABAJANDO EN LA LINEA 500 AHORA":GOTO 10
```

Figura 4-9 Listado del Programa ON/GOTO

Al ejecutarse el programa, el aspecto será el siguiente:

```
TRABAJANDO EN LA LINEA 100 AHORA
TRABAJANDO EN LA LINEA 200 AHORA
TRABAJANDO EN LA LINEA 300 AHORA
TRABAJANDO EN LA LINEA 400 AHORA
TRABAJANDO EN LA LINEA 500 AHORA
TERMINADO
```

Figura 4-10 Ejecución del Programa ON/GOTO

POP

Formato: POP
Ejemplo: 1000 POP

En la descripción de la sentencia FOR/NEXT, la pila o stack se definió como un grupo de direcciones de memoria reservadas para direcciones de retorno. El ingreso superior de la pila controla el número de bucles que deben ejecutarse y la línea meta de RETURN para un GOSUB. Si la subrutina no termina por medio de una sentencia RETURN, la ubicación de memoria superior de la pila siempre estará cargada con algún número. Si enseguida se ejecuta otra GOSUB, esta ubicación superior debe limpiarse. Para preparar la pila para un nuevo GOSUB, use un POP para limpiar los datos de la ubicación superior de la pila.

El comando POP debe usarse de acuerdo a las siguientes reglas:

1. Debe estar en el paso de ejecución del programa.
2. Debe seguir a la ejecución de una sentencia GOSUB que no ha vuelto al programa principal a través de una sentencia RETURN.

El siguiente ejemplo muestra el uso del comando POP con GOSUB cuando no se ejecuta RETURN:

```
10 GOSUB 1000
15 REM LA LINEA 20 NO SERA EJECUTADA
20 PRINT "RETORNO NORMAL PRESENTA ESTE MENSAJE"
30 PRINT "RETORNO ANORMAL PRESENTA ESTE MENSAJE"
40 POP
999 END
1000 PRINT "AHORA SE ESTA EJECUTANDO LA SUBROUTINA"
1010 GOTO 30
1020 RETURN
```

Figura 4-11 Sentencia GOSUB con POP.

RESTORE (RES.)

Formato: RESTORE [expa]
Ejemplo: 100 RESTORE

El sistema de Computación Personal Atari contiene un "puntero" interno que hace el seguimiento del ítem de sentencia DATA próximo a leerse. Usada sin la expa opcional, la sentencia RESTORE repone el puntero al primer ítem de DATA del programa. Usada con la expa opcional, la sentencia RESTORE repone el puntero al primer ítem de DATA de la línea especificada por el valor de la expa. Esta sentencia permite el uso repetitivo de los mismos datos.

```

10 FOR N=1 TO 2
20 READ A
30 RESTORE 30
40 READ B
50 M=A+B
60 PRINT "EL TOTAL VALE ";M
70 NEXT N
80 END
90 DATA 30,15

```

Figura 4 - 12. Listado de un Programa RESTORE

En el primer paso a través del bucle, A será 30 y B será 30 de modo que el total de la línea 50 imprimirá EL TOTAL VALE 60; pero en la segunda pasada, A valdrá 15 y B, a consecuencia de la sentencia RESTORE, siempre valdrá 30. Por ello, la sentencia PRINT de la línea 60 desplegará EL TOTAL VALE 45.

TRAP (T.)

Formato: TRAP expa

Ejemplo: 100 TRAP 120

La sentencia TRAP se emplea para dirigir el programa a un número de línea específico si se detecta un error. Sin la sentencia TRAP, el programa detiene su ejecución cuando encuentra un error y despliega el mensaje de error sobre la pantalla.

La sentencia TRAP funciona con cualquier error que pueda ocurrir una vez que haya sido ejecutada, pero una vez que se haya detectado un error y se le haya atrapado, es necesario reponer la trampa con otro comando TRAP. Este comando TRAP puede ubicarse al comienzo de la sección del programa que maneje la entrada desde el teclado, de modo que la trampa sea repuesta después de cada error. PEEK (195) entregará un mensaje de error (vea el apéndice B). $256 * \text{PEEK}(187) + \text{PEEK}(186)$ dará el número de la línea en el cual ocurrió el error. TRAP puede limpiarse ejecutando una sentencia TRAP con una expa cuyo valor esté comprendido entre 32768 y 65535 (por ejemplo, 40000).

COMANDOS Y DISPOSITIVOS DE ENTRADA Y SALIDA

Esta sección describe los dispositivos de entrada/salida y como los datos se mueven entre ellos. Los comandos explicados en esta sección son aquellos que permiten el acceso a los dispositivos de entrada/salida. Los comandos de entrada son los asociados con la obtención de datos para la memoria y los dispositivos apropiados para aceptar entradas. Los comandos de salida son aquellos asociados con la búsqueda de datos de la memoria y los dispositivos adecuados para generar salidas.

Los comandos descritos en esta sección son:

CLOAD	INPUT	OPEN/CLOSE	READ/DATA
CSAVE	LOAD	POINT	SAVE
DOS	LPRINT	PRINT	STATUS
ENTER	NOTE	PUT/GET	X10

DISPOSITIVOS DE ENTRADA / SALIDA

La configuración física de cada uno de los dispositivos siguientes se ilustra en los manuales individuales que se entregan con cada uno de ellos. El subsistema central de entrada/salida (Central Input/Output (CIO)) provee al usuario con una sola interfaz para acceder todos los dispositivos periféricos del sistema en gran medida independientemente de éstos.

Esto significa que existe un solo punto de entrada y una secuencia de llamada independiente del dispositivo. Cada dispositivo tiene su nombre de dispositivo simbólico utilizado para identificarlo; por ejemplo, K: para el teclado (Keyboard).

Cada dispositivo debe ser abierto antes de su acceso y a cada uno se le debe asignar un bloque de control de entrada/salida IOCB (Input/Output Control Block). De ahí en adelante, se hace referencia al dispositivo a través de su número IOCB.

BASIC ATARI contiene 8 bloques en la memoria que identifican al Sistema Operativo la información requerida para realizar una operación Entrada/Salida. Esta información incluye el comando, el largo de la memoria de transferencia, la dirección de la memoria de transferencia y dos variables de control auxiliares. BASIC ATARI se encarga de preparar el IOCB, pero el usuario debe especificar cual de ellos usará. BASIC reserva el IOCB #0 para Entrada/Salida al Editor de Pantalla, por lo cual el usuario no puede requerir el IOCB #0. Las sentencias gráficas (vea la Sección 9) abren IOCB #6 para entrada y salida hacia la pantalla. (Es ésta la ventana gráfica S:). IOCB #7 es usado por BASIC para los comandos LPRINT, CLOAD, y CSAVE. También se puede hacer referencia al número del IOCB como el número de dispositivo o archivo. Los IOCB del 1 al 5 se usan para abrir otros dispositivos para operaciones de entrada/salida. Si el IOCB # 7 está en uso, será imposible realizar LPRINT o algunas de las otras sentencias BASIC de Entrada/salida.

TECLADO: (K:) (Keyboard):

Dispositivo de Entrada solamente. El teclado permite al usuario leer los datos de teclado convertidos (ATASCII) a medida que se presiona cada tecla.

IMPRESOR DE LINEA: (P:) (Line Printer):

Dispositivo de Salida solamente. El impresor de línea imprime los caracteres ATASCII de una línea por vez. En general no reconoce los caracteres de control.

GRABADORA DE PROGRAMAS: (C:) (Program Recorder):

Dispositivo de Entrada y Salida. La grabadora es un dispositivo de escritura/lectura que puede usarse para ambas funciones, pero jamás ambas simultáneamente.

El cassette tiene dos pistas para sonido y grabación de programas respectivamente. La pista de audio no puede grabarse desde un sistema ATARI, pero puede ser reproducida a través del parlante del televisor.

UNIDADES DE DISCO: (D1:, D2:, D3:, D4:) (Disk Drives):

Dispositivos de entrada y salida. Si se dispone de 16KB de memoria de acceso aleatorio RAM, el computador ATARI puede manejar de una a cuatro unidades de disco. Si solamente se encuentra conectada una unidad de disco, no es necesario agregar el número después del código simbólico del dispositivo D.

EDITOR DE PANTALLA: (E:) (Screen Editor):

Dispositivo de Entrada y Salida. Este dispositivo emplea el teclado y la pantalla (vea monitor TV) para simular un terminal de edición de pantalla. Escribir en este dispositivo hace que los datos aparezcan sobre la pantalla a partir de la posición presente del cursor. La lectura de ese dispositivo activa el proceso de edición de pantalla y permite al usuario ingresar y editar datos. Cada vez que se presione la tecla **RETURN**, toda la línea lógica dentro de la cual se encuentra el cursor se selecciona como la información a ser transferida por el CIO al programa del usuario. (vea la Sección 9).

MONITOR DE TV: (S:)

Dispositivo de Entrada y Salida. Este dispositivo permite al usuario leer caracteres desde y escribir caracteres a la pantalla, usando el cursor como el mecanismo de direccionamiento de pantalla. Soporta tanto operaciones de texto como gráficas. Vea la Sección 9 para una descripción completa de los modos gráficos.

INTERFAZ, RS-232: (R:)

El dispositivo RS-232 permite al sistema ATARI interactuar con dispositivos compatibles RS-232 tales como impresores, terminales, y trazadores. Contiene una puerta paralela del tipo Centronic, usada principalmente para la conexión de impresores.

CLOAD (CLOA.)

Formato: CLOAD

Ejemplos: CLOAD
100 CLOAD

Este comando puede usarse ya sea en el modo Directo o en el Diferido para cargar un programa desde cinta cassette a la memoria para su ejecución. Al ingresar CLOAD, suena un campanillazo para indicar que debe apretarse la tecla PLAY y enseguida **RETURN**.

Sin embargo, no presione PLAY hasta que la cinta haya sido correctamente ubicada. Las instrucciones específicas para el CLOAD de un programa se encuentran en los manuales de las Grabadoras de Programa ATARI. Los pasos necesarios para cargar programas excepcionalmente grandes se incluyen en los párrafos bajo CONCATENACION DE PROGRAMAS al final de esta sección.

CSAVE (CS.)

Formato: CSAVE

Ejemplos: CSAVE
100 CSAVE
100 CS.

Este comando se usa habitualmente en el modo Directo para grabar un programa residente en memoria en cinta cassette. CSAVE graba la versión codificada del programa. Al ingresar CSAVE suenan dos campanillazos para indicar que deben apretarse las teclas PLAY y RECORD seguido de **RETURN**. Sin embargo, no presione estas teclas hasta que la cinta haya sido correctamente ubicada. Es más rápido grabar un programa usando este comando que SAVE "C" (vea SAVE) porque se usan intervalos de grabación cortos.

Notas:

Cintas grabadas bajo los dos comandos, SAVE y CSAVE, no son compatibles.

Puede resultar necesario ingresar un LPRINT (vea LPRINT) antes de recurrir a CSAVE. De otra forma es posible que CSAVE no opere adecuadamente.

Para instrucciones específicas de como conectar y operar los equipos, ubicar la cinta, etc., vea el Manual de su Grabadora de Programa ATARI.

DOS (DO.)

Formato: DOS

Ejemplo: DOS

El comando DOS se emplea para pasar de BASIC al Sistema Operativo de Disco (DOS). Si el Sistema Operativo de Disco no ha sido cargado en memoria, el computador pasará al Modo Borrador (Memo Pad) o al logo ATARI (en el caso de los modelos XL). El usuario debe presionar **Reset** para volver al modo directo. Si el Sistema Operativo de Disco había sido cargado, el Menú del DOS aparece en la pantalla. Para limpiar el Menú del DOS de la pantalla, presione **Reset**.

El control pasará así a BASIC. También puede retornarse el Control a BASIC seleccionando B (Run Cartridge) del Menú del DOS.

El comando DOS normalmente se emplea en el Modo Directo; sin embargo, puede utilizarse igualmente en un programa. Para mayores detalles sobre esto, vea el Manual DOS de ATARI.

ENTER (E.)

Formato: ENTER esparchivo

Ejemplos: ENTER "C
ENTER "D:DEMOPR. INS"

Con esta sentencia la grabadora cassette reproducirá un programa originalmente grabado usando LIST (vea la Sección 2, LIST).

El programa se ingresa en forma no procesada (sin codificar), y se interpreta a medida que los datos se reciben. Una vez completa la carga, puede correrse en la forma normal. El comando ENTER también puede emplearse con la unidad de disco.

Note que tanto LOAD y CLOAD (vea la Sección 2) limpian el programa antiguo de la memoria antes de cargar uno nuevo. ENTER refunde los programas antiguo y nuevo. Esta sentencia ENTER generalmente se emplea en el Modo Directo.

INPUT (I.)

Formato: INPUT $\left[\#expa \left\{ ; \right\} \left\{ \begin{matrix} \text{vara} \\ \text{vars} \end{matrix} \right\} \left[, \left\{ \begin{matrix} \text{vara} \\ \text{vars} \end{matrix} \right\} \dots \right] \right]$

Ejemplos: 100 INPUT X
 100 INPUT N\$
 100 PRINT " INGRESE EL VALOR DE X "
 110 INPUT X

Esta sentencia requiere al usuario ingresar datos desde el teclado. Durante su ejecución, el computador despliega un signo de interrogación ? cuando el programa encuentra una sentencia INPUT. Normalmente está precedido por una sentencia PRINT que indica al usuario el tipo de información que se le solicita.

Las variables de string o literales se permiten solamente si no están indexadas. No se permiten variables matriciales.

El # expa es opcional y se emplea para especificar un número de dispositivo o archivo del cual debe ingresarse los datos (vea dispositivos de Entrada/Salida). Si no se especifica el # expa, el ingreso proviene del editor de pantalla (E:).

Si deben ingresarse varios strings desde el editor de pantalla, digite un string, presione **RETURN**, digite el string siguiente, **RETURN**, etc. Números aritméticos pueden digitarse en la misma línea separándolos por comas.

```
10 PRINT "INGRESE 5 NUMEROS PARA SUMAR"
20 FOR N=1 TO 5
30 INPUT X
40 C=C+X
50 NEXT N
60 PRINT "LA SUMA DE SUS NUMEROS ES ";C
70 END
```

Figura 5-1 Listado de Programa con Input.

El listado de más arriba ilustra un programa que sumará 5 números ingresados por el usuario.

LOAD (L.)

Formato: LOAD esparchivo

Ejemplo: LOAD "D1 :FACTURAS. TOT"

Este comando es similar a CLOAD, excepto que se puede emplear el sistema de nombre de archivo completo. LOAD emplea espacios inter-bloques largos en la cinta (vea CLOAD) y la versión codificada del programa. Al emplear solamente una unidad de disco, no es necesario especificar un número después de la "D" debido a que la unidad de disco asumida será siempre # 1.

LPRINT (LP.)

Formato: LPRINT

Ejemplo: LPRINT "PROGRAMA PARA CALCULAR X"
 100 LPRINT X; " ",Y;" ";Z

Con esta sentencia el computador imprime datos en el impresor de línea, en vez de hacerlo en la pantalla. Puede usarse tanto en el modo Directo como en el Diferido. No requiere especificador de dispositivos ni sentencias de OPEN o CLOSE (BASIC usa el IOCB #7).

Para imprimir un listado de programa en un impresor de línea, vea LIST.

NOTE (NO.)

Formato: NOTE #expa, vara, vara

Ejemplo: 100 NOTE # 1, X, Y

Este comando se emplea para almacenar en la primera vara el número del sector de disco en uso y en la segunda vara el número del byte de ese sector. Es ésta la posición de lectura o escritura del archivo especificado donde se leerá o escribirá el próximo byte. Este comando NOTE se emplea al escribir datos en un archivo de disco (vea POINT). La información del comando NOTE se escribe en un segundo archivo, que se usa enseguida como índice para el primer archivo.

OPEN (O.)

CLOSE (CL.)

Formatos: OPEN # expa, expa1, expa2, esparchivo
CLOSE #expa

Ejemplos: 100 OPEN #2,8,0, "D1:ATARI800.BAS"
100 A\$="D1:ATARI.BAS"
110 OPEN #2,8,0,A\$
150 CLOSE #2

Antes de que se pueda acceder a un dispositivo, éste debe ser abierto. Este proceso de apertura enlaza a un IOCB específico al administrador de dispositivo apropiado, inicializa las variables de control relacionadas con el sistema central de entradas y salidas CIO y comunica las opciones específicas del dispositivo al administrador del dispositivo. Los parámetros para un comando OPEN se definen como sigue:

#	Carácter obligatorio que debe ser ingresado por el usuario.
expa	Número de referencia del IOCB para uso futuro (como por ejemplo en el comando CLOSE). El número puede ir del 1 hasta el 7.
expa1	Número de código para determinar si se trata de una operación de entrada o salida.
Código	4 = Operación de entrada
	8 = Operación de salida
	12 = Operación de entrada y salida
	6 = Operación de entrada de directorio de disco (En este caso la esparchivo es la especificación de búsqueda).
	9 = Operación de apéndice de fin de archivo (EOF) (salida).
	Apéndice se usa también para un modo especial de entrada del editor de pantalla. Este modo le permite a un programa ingresar la próxima línea desde E: sin esperar que el usuario presione RETURN .

- expa2** Código auxiliar dependiente del dispositivo. Un 83 en este parámetro indica impresión lateral en el impresor ATARI modelo 820 (vea los manuales correspondientes para éstos códigos de control).
- esparchivo** Designación específica del archivo. Debe estar encerrada entre comillas. El formato para el parámetro esparchivo se muestra en la figura 5-2.

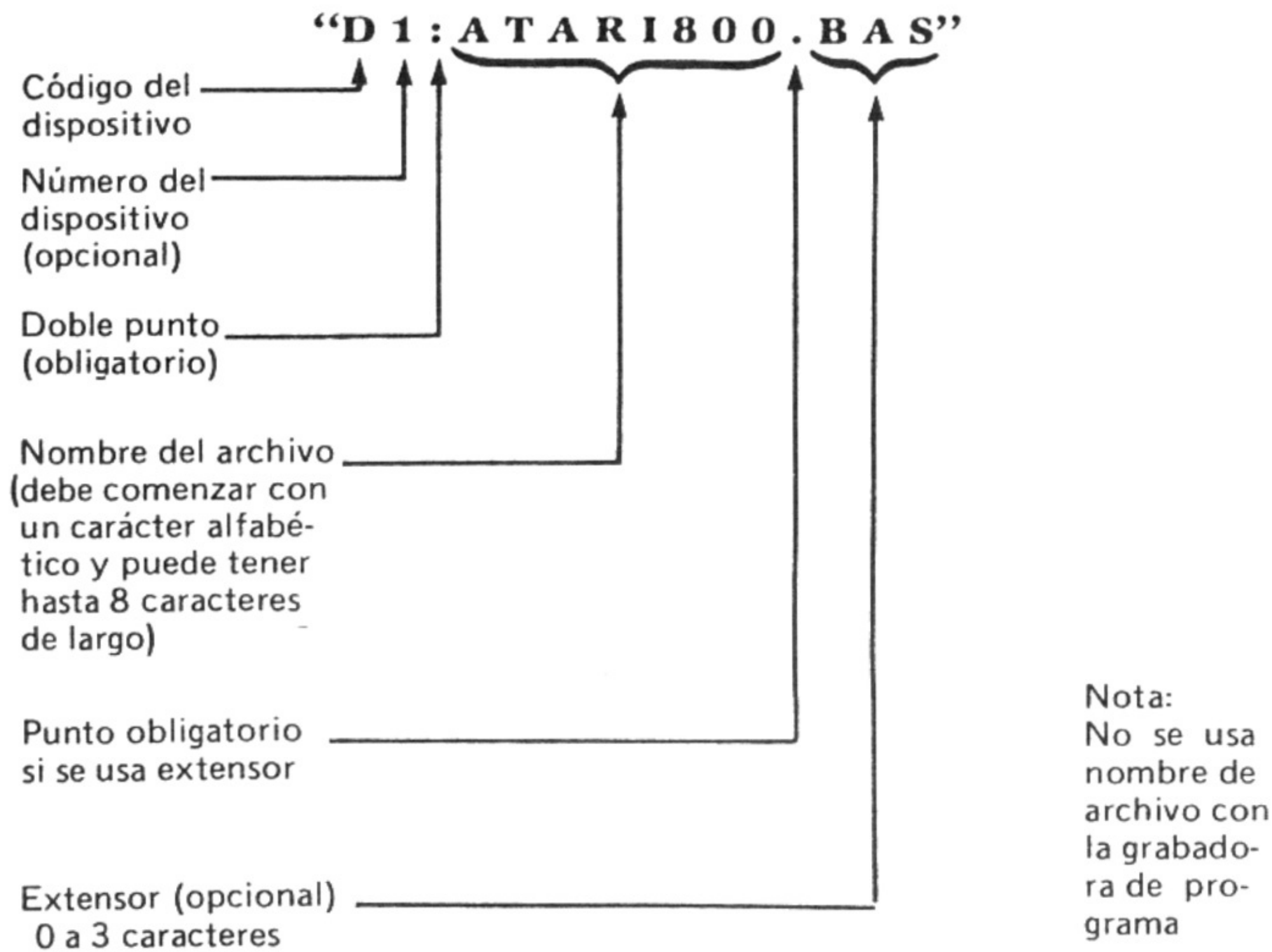


Figura 5-2 Detalle de la Especificación de Archivo (Esparchivo)

El comando CLOSE simplemente cierra el archivo que había sido abierto previamente con un comando OPEN. En el ejemplo note que la expa que sigue al carácter obligatorio # debe ser la misma expa que hace de número de referencia en la sentencia OPEN.

POINT (P.)

Formato: POINT # expa, vara, var

Ejemplo: 100 POINT #2,A,B

Este comando se emplea al leer un archivo a la memoria. La primera vara especifica el número de sector y la segunda vara especifica el byte dentro de ese sector donde se leerá o grabará el próximo byte. Esencialmente, el comando mueve un puntero controlado por software hasta la ubicación especificada del archivo. Esto da al usuario la posibilidad de acceder al azar los datos almacenados en el archivo de disco. Los comandos POINT y NOTE se discuten con mayor detalle en el Manual DOS.

PRINT (PR o ?)

Formato: PRINT [#expa] { ; } [exp] [,exp...]

Ejemplos: PRINT X,Z,A\$
 100 PRINT "EL VALOR DE X ES ";X
 100 PRINT "LAS COMAS", "SEPARAN", "POR", "COLUMNAS"
 100 PRINT #3, A\$

El comando PRINT puede emplearse ya sea en el modo Directo o en el Diferido. En el modo Directo, este comando imprime cualquier información contenida entre dos pares de comillas exactamente tal como aparece. En el primer ejemplo, PRINT X,Y,Z,A\$, la pantalla desplegará los valores actuales de X,Y,Z,y A\$ tal como aparecen en el programa residente en la memoria. En el último ejemplo, PRINT #3,A\$, el #3 es el especificador de archivo (puede ser cualquier número entre 1 y 7) y que controla hacia cual dispositivo se imprimirá el valor de A\$. (Vea dispositivos de Entrada/Salida.)

Una coma produce el desplazamiento hasta el siguiente tope de tabulación. Un punto y coma hace que la siguiente expa o exps sea colocada inmediatamente después de la expresión precedente, sin esparciamiento. Por ello, en el segundo ejemplo se pone un espacio antes de las comillas finales, de modo que el valor de X no aparezca inmediatamente después de la palabra "ES". Si no se usa coma o punto y coma al final de una sentencia de PRINT, el computador agregará un **RETURN** ; el PRINT siguiente comenzará en la próxima línea.

PUT (PU.) / GET (GE.)

Formato: PUT #expa, expa
 GET #expa, vara

Ejemplos: 100 PUT #6, ASC("A")
 200 GET #1,X

PUT y GET son opuestos entre sí. El comando PUT entrega un byte comprendido entre 0 y 255 al archivo especificado por #expa. (# es un carácter obligatorio en ambos comandos). El comando GET lee un byte comprendido entre 0 y 255 (usando #expa para definir el archivo, etc. del diskette o donde sea) y enseguida almacena el byte en la variable vara.

**READ (REA.)
DATA (D.)**

Formatos: READ var [, var...]
 DATA dat [, dat...]

Ejemplos: 100 READ A,B,C,D,E
 110 DATA 12,13,14,15,16
 100 READ A\$,B\$,C\$,D\$,E\$
 110 DATA MB, EVELYN, CARLA, CORINA, BARBARA

Estos dos comandos siempre se usan juntos y la sentencia DATA siempre se usa en el modo Diferido (1). Las sentencias DATA pueden estar ubicadas en cualquier parte del pro-

(1) UN READ en Modo Directo solamente leerá datos si se ha ejecutado una instrucción DATA en el programa.

grama, pero deben tener tantas piezas de datos como se hayan definido en las sentencias READ. De otro modo se despliega un error "out of data" ("sin datos") en la pantalla.

Las variables String o literales que se lean en las sentencias READ deben dimensionarse y no pueden ser indexadas. (Vea en la Sección STRINGS). Tampoco pueden usarse variables de agrupamiento en una sentencia READ.

La sentencia DATA contiene una serie de datos string para ser accedidos por la sentencia READ. No puede contener operaciones aritméticas ni funciones, etc. Adicionalmente, el tipo de datos de la sentencia DATA debe coincidir con el tipo de variable definida en la sentencia READ correspondiente.

El siguiente programa suma una lista de números de una sentencia DATA.

```
10 FOR N=1 TO 5
20 READ D
30 M=M+D
40 NEXT N
50 PRINT "LA SUMA ES IGUAL A ";M
60 END
70 DATA 30,15,106,17,87
```

Figura 5-3 Listado de un Programa Read/Data

Este programa al ser ejecutado, imprimirá la siguiente sentencia:

LA SUMA ES IGUAL A 255

SAVE (S.)

Formato: SAVE esparchivo

Ejemplo: SAVE "D1:PRECIOS.NET"

El comando SAVE es similar al comando CSAVE, excepto en que se puede usar el nombre de archivo completo. El código numérico del dispositivo es opcional al usar solamente una unidad de disco. La unidad de disco asumida siempre es la #1. SAVE como LOAD, usa el espacio interbloque de grabación largo en el cassette (vea CSAVE) y la versión codificada del programa.

STATUS (ST.)

Formato: STATUS #expa, vara

Ejemplo: 350 STATUS #1,Z

El comando STATUS llama la rutina STATUS para el dispositivo especificado (expa). El estado del comando STATUS (vea MENSAJES DE ERROR, Apéndice B) se almacena en la variable especificada (vara). Esto puede ser útil para conocer el éxito o fracaso de una operación de entrada/salida (I/O).

XIO (X.)

Formato: XIO nocdo, #expa, expa1, expa2, esparchivo

Ejemplo: XIO 18, #6,0,0, "S:"

El comando XIO es una sentencia de entrada/salida generalizada que se emplea para operaciones especiales. Un ejemplo es su uso para llenar (fill) de color un área de la pantalla entre puntos y líneas ya dibujados (vea la Sección 9). Los parámetros para este comando se definen como sigue:

nocdo: Número que corresponde al comando particular a ejecutar.

nocdo	OPERACION	EJEMPLO
3	OPEN	Lo mismo que OPEN en BASIC
5	GET REGISTRO	Estos 4 comandos son iguales a INPUT, GET, PRINT y PUT en BASIC respectivamente
7	GET CHARACTER	
9	PUT REGISTRO	
11	PUT CHARACTER	
12	CLOSE	igual a CLOSE en BASIC
13	REQUERIMIENTO DE STATUS	igual a STATUS en BASIC
17	TRAZAR LINEA	igual al DRAWTO en BASIC
18	FILL	ver Sección 9
32	RENAME (RENOMBRAR)	XIO32, #1,0,0, "D:TEM.CAROL"
33	DELETE (ELIMINAR)	XIO33, #1,0,0, "D:TEM.BAS"
35	LOCK FILE (PROTEGER ARCHIVO)	XIO35, #1,0,0, "D:TEM.BAS"
36	UNLOCK FILE (DESPROTEGER ARCHIVO)	XIO36, "1,0,0, "D:TEM.BAS"
37	POINT (APUNTAR A)	igual a POINT en BASIC
38	ANOTAR	igual a NOTE en BASIC
254	FORMATEAR	XIO254, #1,0,0, "D2:"

expa: Número del dispositivo (igual como en OPEN). La mayor parte del tiempo se ignora, pero debe ser precedido por #.

expa1

expa 2: Dos bytes de control auxiliares. Su uso depende del dispositivo en particular y del comando. En la mayoría de los casos no se emplean y se igualan a 0.

esparchivo: Expresión string o literal que especifica el dispositivo. Debe estar entre comillas. Aunque algunos comandos como fill (rellenar) (Sección 9), no se fijan en esta esparchivo, siempre debe estar incluida en la sentencia.

CONCATENACION DE PROGRAMAS

Si un programa requiere más memoria que la disponible, siga los siguientes pasos para encadenar programas, cada uno de los cuales requiera menos memoria de la disponible.

1. Digite la primera parte del programa en la forma normal
2. La última línea de esta primera parte del programa debería contener solamente el número de línea y el comando RUN"C:"
3. Embobine la cinta hasta una parte virgen. Anote el número de contador de programa para propósito de RUN posteriores. Presione PLAY y RECORD en la grabadora de modo que ambas teclas permanezcan abajo.
4. Digite SAVE "C:" y presione **RETURN**.
5. Cuando suene el doble pito, vuelva a presionar **RETURN**.

6. Cuando la pantalla despliegue "READY" no mueva la cinta. Digite NEW **RETURN**.
7. Repita las instrucciones superiores para la segunda parte del programa.
8. Siendo la segunda parte del programa esencialmente un programa totalmente nuevo, es posible volver a usar los números de líneas usados ya en la primera parte.
9. Si hay una tercera parte del programa, asegúrese que la última línea de la segunda parte sea un comando RUN"C:".

Para ejecutar un programa encadenado, siga los siguientes pasos:

1. Lleve la cinta al comienzo de la parte 1 del programa.
2. Presione la tecla PLAY de la grabadora.
3. Digite RUN"C:" **RETURN**
4. Cuando suene el pito, presione **RETURN** nuevamente.

El computador automáticamente cargará la primera parte del programa, la ejecutará y hará sonar un pito para indicar que es el momento de presionar **RETURN** o la barra de espacio para iniciar el movimiento del motor de la grabadora para la segunda parte de LOAD/RUN. La carga demora algunos segundos.

NOTA: También un programa de una sola parte puede grabarse y volver a cargarse en esta misma forma o igualmente pueden usarse CSAVE y CLOAD.

NOTA: Recuerde de cargar el DOS antes de digitar su programa.

MODIFICACION DE UN PROGRAMA BASIC EN DISCO

El procedimiento para la modificación de un programa BASIC ya existente, almacenado en un diskette se demuestra a manos de los siguientes pasos:

1. Apague el computador ATARI (e inserte el cartridge de BASIC en el caso de los modelos 400 y 800).
2. Conecte la unidad de disco y préndala sin insertar un diskette.
3. Espere que la luz Busy (ocupado) se apague y se detenga el motor de la unidad. Abra la puerta de la unidad de disco.
4. Inserte un diskette con DOS y cierre la puerta.
5. Prenda su computador. El DOS debería cargarse en él y la pantalla mostrar ahora READY
6. Para cargar un programa desde el disco, digite LOAD"D:archivo. ext
7. Modifique el programa o digite un programa nuevo.
8. Para grabar el programa en diskette, digite SAVE"D:archivo.ext
9. Siempre espere que se apague la luz Busy antes de sacar un diskette.
10. Para obtener un listado del Directorio, no saque todavía el diskette y digite DOS

Apriete **RETURN** ; una vez que aparezca el Menú del DOS escoja la letra de comando A, dígitela y presione **RETURN** dos veces para obtener el Directorio en la pantalla; o digite A seguido de **RETURN** y enseguida P: **RETURN** para listar el Directorio por el impresor.

11. Para volver a BASIC, digite B **RETURN** o presione **Reset**.

Esta sección describe las funciones aritméticas, trigonométricas, y de propósitos especiales que se encuentran incorporadas en BASIC ATARI. Una función realiza un cálculo y devuelve el resultado (por lo común un número) ya sea para impresión o uso computacional posterior. Entre las funciones trigonométricas se incluyen dos sentencias, radianes (RAD) y grados (DEG), que se usan frecuentemente con las funciones trigonométricas. Cada una de las funciones descritas en esta sección puede usarse ya sea en el modo Directo o en el modo Diferido. Son perfectamente legales las funciones múltiples.

En esta sección se describen las siguientes funciones y sentencias:

ABS	ATN	ADR
CLOG	COS	FRE
EXP	SIN	PEEK
INT	DEG/RAD	POKE
LOG		USR
RND		
SGN		
SQR		

FUNCIONES ARITMETICAS

ABS

Formato: ABS (expa)
Ejemplo: 100 AB=ABS (-190)

Entrega el valor absoluto de un número sin atención respecto de si éste es positivo o negativo. El valor entregado siempre es positivo.

CLOG

Formato: CLOG (EXPA)
Ejempló: 100 C=CLOG(83)

Entrega el logaritmo en base 10 de la variable o expresión entre paréntesis. CLOG (0) da error y CLOG(1) es 0.

EXP

Formato: EXP (expa)
Ejemplo: 100 PRINT EXP (3)

Retorna el valor de e (aproximadamente 2. 71828283), elevado a la potencia especificada por la expresión entre paréntesis. En el ejemplo de más arriba, el número entregado es 20.0855365. En algunos casos, EXP es exacto solamente hasta seis cifras significativas.

INT

Formato: INT (expa)
Ejemplos: 100 I = INT (3.445) (se almacena 3 en I)
100 X = INT (-14.66778) (se almacena -15 en X)

Entrega el mayor número entero que sea menor o igual al valor de la expresión. Esto se cumple ya sea que el valor resultante de la expresión sea un número positivo o negativo. Así, en nuestro primer ejemplo de más arriba, I se usa para almacenar el número 3.

En el segundo ejemplo, X se usa para almacenar el número -15 (el primer número entero menor o igual a -14,66778). Esta función INT no debe confundirse con la función usada en las calculadoras y que simplemente trunca o corta todos los lugares decimales.

LOG

Formato: LOG (expa)

Ejemplo: 100 L = LOG (67.89/2.57)

Entrega el logaritmo natural del número o de la expresión entre paréntesis. LOG(0) da un error y LOG(1) es igual a 0.

RND

Formato: RND (expa)

Ejemplo: 10 A= RND (0)

Entrega un número aleatorio generado circuitalmente y comprendido entre 0 y 1, pero jamás entrega 1. La variable o expresión entre paréntesis que sigue a RND es un relleno y no tiene efecto sobre el número que se obtiene. Sin embargo, esta variable de relleno debe emplearse. Generalmente la función RND se usa en combinación con otras sentencias o funciones BASIC para entregar un número para juegos, tomas de decisiones y cosas parecidas. He aquí una rutina simple que entrega un número al azar entre 0 y 999.

```
10 X=RND(0)           (0 es una variable de relleno)
20 RX=INT(1000*X)
30 PRINT RX
```

SGN

Formato: SGN (expa)

Ejemplo: 100 X= SGN(-199) (X toma el valor -1)

Entrega un -1 si expa es de valor negativo; un 0 si expa da 0, o un 1 si expa tiene resultado positivo.

SQR

Formato: SQR(expa)

Ejemplo: 100 PRINT SQR(100)

Entrega la raíz cuadrada de la expa, la cual debe ser positiva.

FUNCIONES TRIGONOMETRICAS

ATN

Formato: ATN (expa)

Ejemplo: 100 X= ATN(65)

Entrega arcotangente de la variable o expresión entre paréntesis.

COS

Formato: COS (expa)

Ejemplo: 100 C = COS(X+Y+ Z)

NOTA: Presupone que X,Y,Z, han sido definidos previamente.

Entrega el coseno trigonométrico de la expresión entre paréntesis.

SIN

Formato: SIN (expa)

Ejemplo: 100 X= SIN(Y)

NOTA: Se supone que Y ha sido definido previamente.

Entrega el seno trigonométrico de la expresión entre paréntesis.

DEG/RAD

Formato: DEG
RAD

Ejemplo: 100 DEG
100 RAD

Estas dos sentencias permiten al programador especificar grados o radianes para el cálculo de funciones trigonométricas. El computador asume radianes a no ser que se especifique DEG. Una vez que una sentencia DEG se ejecute, debe emplearse RAD para volver a radianes.

Vea el Apéndice E para funciones trigonométricas adicionales que puedan derivarse.

FUNCIONES DE PROPOSITOS ESPECIALES

ADR

Formato: ADR(vars)

Ejemplo: ADR(A\$)

Entrega la dirección de memoria decimal del string especificado por la expresión entre paréntesis. El conocer esta dirección permite al programador pasar información a rutinas USR, etc. (Vea USR y el apéndice D).

FRE

Formato: FRE (expa)

Ejemplos: PRINT FRE (0)
100 IF FRE (0) <1000 THEN PRINT "MEMORIA CRITICA".

Esta función entrega la cantidad de bytes de memoria RAM de usuario disponible. Su uso principal está en el Modo Directo con una variable de relleno (0) para informar al programador de cuanto espacio disponible le queda en la memoria para terminar su programa. Naturalmente FRE también puede utilizarse dentro de un programa BASIC en el Modo Diferido.

PEEK

Formato: PEEK (expa)

Ejemplos: 1000 IF PEEK (4000)=255 THEN PRINT "255"
100 PRINT "EL MARGEN IZQUIERDO ES"; PEEK (82)

Entrega el contenido de una ubicación de dirección de memoria especificada (expa). La dirección especificada debe ser un número entero o una expresión aritmética cuyo valor resultante sea un entero comprendido entre 0 y 65535 y representa la dirección de memoria en notación decimal (no hexadecimal). El número entregado también será un número decimal íntegro en el rango comprendido entre 0 y 255. Esta función permite al usuario examinar tanto las ubicaciones de RAM como las de ROM. En el ejemplo, de más arriba, PEEK se usa para determinar si en la locación 4000 decimal hay el número 255. En el segundo ejemplo la función PEEK se emplea para examinar el valor del margen izquierdo.

POKE

Formato: POKE expa1, expa2

Ejemplos: POKE 82, 10
100 POKE 82, 20

A pesar de que no se trata de una función, se le incluye en esta sección porque está asociado muy de cerca con la función PEEK. Este comando POKE inserta datos en las ubicaciones de memoria o modifica datos que ya se encuentran allí. En el formato descrito más arriba, expa1 es la dirección decimal de una ubicación que va a ser cargada y expa2 es el dato con el cual se carga. Note que este número es un número decimal comprendido entre 0 y 255. POKE no puede usarse para alterar las ubicaciones de ROM. Para adquirir familiaridad con este comando, se recomienda mirar el contenido de las ubicaciones de memoria y anotar el contenido de cada ubicación, enseguida si el comando POKE no causa el efecto anticipado, el contenido original puede volver a cargarse en las ubicaciones respectivas.

El ejemplo de Modo Directo de más arriba cambia el margen de pantalla izquierdo de su valor asumido de la posición 2 a la nueva posición 10. En otras palabras, el nuevo margen estará a 10 espacios hacia la derecha. Para restituir el margen a su valor asumido normal, presione **Reset**.

USR

Formato: USR (expa1 [,expa2][,expa3...])

Ejemplo: 100 RESULTADO=USR(ADD1,A*2)

Esta función entrega el resultado de una subrutina en lenguaje de máquina. La primera expresión expa1, debe corresponder a un número entero o una expresión aritmética cuyo resultado sea un valor entero que represente la dirección de memoria decimal de la rutina de máquina que deba ejecutarse. Los argumentos de entrada expa2, expa3, etc. son opcionales.

Ellos deben corresponder a expresiones aritméticas con un valor decimal en el rango de 0 a 65535. Pueden emplearse valores no íntegros; sin embargo, un valor no íntegro será redondeado al número entero más próximo.

Estos valores serán convertidos en formato de número decimal codificado binario (BCD) en coma flotante del BASIC a número binario de dos bytes; enseguida serán puestos sobre

el stack compuesto por un grupo de ubicaciones de memoria RAM bajo control directo del microprocesador 6502. La figura 6-1 ilustra la estructura de este stack.

N (Cantidad de argumentos sobre el stack; puede ser igual a 0)

X1 (Byte superior del argumento X)

X2 (Byte inferior del argumento X)

Y1 (Byte superior del argumento Y)

Y2 (Byte inferior del argumento Y)

Z1 (Byte superior del argumento Z)

Z2 (Byte inferior del argumento Z)

.

R1 (Byte inferior de la dirección de regreso)

R2 (Byte superior de la dirección de regreso)

Figura 6-1. Definición del Stack

NOTA: X es el argumento que sigue a la dirección de la rutina, Y es el segundo, Z el tercero, etc. Hay N pares de bytes.

Vea la Sección 11 para una descripción de la función USR en la programación en lenguaje de máquina. El apéndice D define los bytes disponibles en la memoria RAM para la programación en lenguaje de máquina.

Esta sección describe los strings o variables literales y las funciones asociadas con el manejo de strings.

Cada string debe dimensionarse (vea la sentencia DIM, en la sección 8) y cada variable string debe finalizar con \$. Un string de por sí es un grupo de caracteres atados uno con el otro. Los caracteres individuales pueden ser letras, números o símbolos (incluyendo los símbolos Atari especiales del teclado). Un substring es una parte de un string mayor; cualquier substring es accesible desde BASIC Atari si el string ha sido correctamente dimensionado (vea al final de esta sección). Los caracteres de un string están indexados de 1 hasta el largo total, que siempre es igual o menor que el largo dimensionado del string.

Las funciones string descritas en esta sección son:

ASC	STR\$
CHR\$	VAL
LEN	

ASC

Formato: ASC(exps)

Ejemplos: 100A=ASC(A\$)

Esta función entrega el número de código ATASCII del primer carácter de la expresión string (exps). Esta función puede emplearse ya sea en el modo Directo o en el Diferido. La figura 7-1 muestra un corto programa que ilustra el uso de la función ASC.

```
10 DIM A$(3)
20 A$="E"
30 A=ASC(A$)
40 PRINT A
```

Figura 7-1. Programa de la Función ASC

Al ejecutarse este programa, imprime un 69, que es el código ATASCII de la letra "E". Note que cuando se usa el string por sí mismo, debe estar encerrado entre comillas.

CHR\$

Formato: CHR\$ (expa)

Ejemplos: 100 PRINT CHR\$ (65)
100 A\$ = CHR\$ (65)

Esta función string de carácter entrega el carácter en formato string, representado por el número o los números de código ATASCII entre paréntesis. Solamente entrega un carácter. En los ejemplos de más arriba, entrega la letra A. Usando las funciones ASC y CHR\$, el siguiente programa imprime las letras mayúsculas y minúsculas del alfabeto.

```
10 FOR I=0 TO 25
20 PRINT CHR$(ASC("A")+I),CHR$(ASC("a")+I)
30 NEXT I
```

Figura 7-2. Ejemplo de Programa ASC y CHR\$

LEN

Formato: LEN (exps)

Ejemplo: 100 PRINT LEN (A\$)

Esta función entrega el largo en bytes del string designado. Esta información puede imprimirse a continuación o usarse en un programa. El largo de una variable string es simplemente el índice del carácter que en este momento se encuentra al final del string. Los strings tienen un largo 0 hasta que se hayan almacenado caracteres en ellos.

Es posible almacenar caracteres en la parte central de un string usando subcriptores. Sin embargo, el comienzo de un string contendrá basuras, salvo que algo se haya almacenado allí previamente.

La siguiente rutina ilustra el uso de la función LEN:

```
10 DIM A$(10)
20 A$="ATARI"
30 PRINT LEN(A$)
```

Figura 7-3. Ejemplo de la función LEN

El resultado de correr el programa de más arriba sería 5.

STR\$

Formato: STR\$ (expa)

Ejemplo: A\$=STR\$ (65)

Esta función de string de número entrega la forma string del número entre paréntesis. EL ejemplo de más arriba entregaría el número 65, pero sería reconocido por el computador como un string.

NOTA: Solamente puede haber un STR\$ y un CHR\$ en una comparación lógica.

Por ejemplo, A=STR\$ (1) > STR\$ (2) no es válido y no funcionará correctamente.

VAL

Formato: VAL (exps)

Ejemplo: 100 A= VAL(A\$)

Esta función entrega un número del mismo valor que el número almacenado como string. Esto corresponde a lo opuesto de la función STR\$. Usando esta función, el computador puede realizar operaciones aritméticas con string, tal como se muestra en el ejemplo de programa siguiente:

```
10 DIM B$(5)
20 B$="10000!"
30 B=SQR(VAL(B$))
40 PRINT "LA RAIZ CUADRADA DE ";B$;" ES ";B
```

Figura 7-4 Programa para la función VAL

Una vez ejecutado, la pantalla mostrará el mensaje LA RAIZ CUADRADA DE 10000 ES 100.

No es posible usar la función VAL con un string que no comience con un número o que no pueda ser interpretado por el computador como un número. Puede, sin embargo, interpre-

tar números en notación de coma flotante; por ejemplo VAL ("1E9") entregaría el número 1.000.000.000.

MANEJO DE STRINGS

Los strings se pueden manejar en una variedad de formas. Pueden ser partidos, concatenados, reagrupados y ordenados. Los siguientes párrafos describen las diferentes manipulaciones.

CONCATENACION DE STRINGS

Concatenación significa poner dos o más strings juntos para formar un string más largo. Cada string que se incluye en el string mayor se llama substring. Cada substring debe dimensionarse (ver DIM). En BASIC Atari, un substring puede contener hasta 99 caracteres (incluyendo espacios). Una vez concatenados los substrings pueden almacenarse en otra variable string, pueden ser impresos o usados en secciones posteriores del programa. La figura 7-5 muestra un programa de concatenación de strings. En este programa se concatena a A\$, B\$ y C\$ y se le pone en el lugar de A\$.

```
10 DIM A$(100),B$(100),C$(100)
20 A$="LOS STRINGS Y SUBSTRINGS SE DISCUTEN "
30 B$="EN EL CAPITULO No.9 "
40 C$="DE 'BASIC ATARI--UNA GUIA DE AUTO-ESTUDIO'"
50 A$(LEN(A$)+1)=B$
60 A$(LEN(A$)+1)=C$
70 PRINT A$
```

Figura 7-5. Ejemplo de Concatenación de Strings

PARTICION DE STRINGS

El formato de una variable string con subcriptores es como sigue:

nombre vars (expa1[,expa2])

El nombre vars se emplea para indicar el nombre de la variable string sin subcriptores (con \$); expa1 indica la ubicación de partida del substring y expa2, si es que se emplea, indica la ubicación final del substring. Si no se especifica expa2, entonces el final del substring coincide con el final del string. La ubicación de partida no puede ser mayor que el largo total del string. Los dos programas ejemplos de la figura 7-6 ilustran un string partido sin indicación de ubicación final y un string partido con su ubicación final indicada.

```
10 DIM S$(5)
20 S$="ABCD#"
30 PRINT S$(2)
40 END
```

El resultado es BCD
(sin indicación de final)

```
10 DIM S$(20)
20 S$="BASIC ATARI 800/XL"
30 PRINT S$(13,15)
40 END
```

El resultado es 800
(con indicación final)

Figura 7-6. Ejemplos de Partición de String

COMPARACION Y ORDENAMIENTO DE STRINGS

En las comparaciones de strings, los operadores lógicos se usan en la misma forma que con números. El segundo programa del apéndice H da un ejemplo simple de un ordenamiento de burbuja.

Al usar operadores lógicos, recuerde que a cada letra, número y símbolo se le asigna un código de número ATASCII. Hay algunas reglas generales que se aplican a estos códigos.

1. Los códigos ATASCII para números están ordenados en la misma forma que los valores reales de los números y siempre son menores que los códigos correspondientes a las letras. (vea el apéndice C).
2. Las letras mayúsculas tienen valores numéricos más bajos que las letras minúsculas. Para obtener el código ATASCII de una letra minúscula, sabiendo el de la mayúscula, agregue 32 al código de la mayúscula.

NOTA: El sistema de manejo de memoria del BASIC Atari mueve los strings a través de la memoria para ir haciendo lugar para nuevas sentencias. Esto hace que la dirección del string cambie si un programa se modifica o se emplea el modo Directo.

Un agrupamiento es una lista unidimensional de números, asignada a variables con subcriptores; por ejemplo: A(0), A(1), A(2). Los subcriptores pueden ir desde 0 hasta el valor dimensionado. La figura 8-1 ilustra un agrupamiento de 7 elementos.

A(0)
A(1)
A(2)
A(3)
A(4)
A(5)
A(6)

Figura 8-1. Ejemplo de un agrupamiento

En este contexto, una matriz es una tabla de dos dimensiones que tiene líneas y columnas. Las líneas van horizontalmente y las columnas verticalmente. BASIC almacena los elementos de una matriz en un orden de precedencia de líneas. Esto significa que todos los elementos de la primera línea se almacenan primero seguidos de los elementos de la segunda línea, etc. La Figura 8-2 ilustra una matriz de 7x4 elementos.

		Columnas			
Líneas	M(0, 0)	M(1, 0)	M(2, 0)	M(3, 0)	
	M(0, 1)	M(1, 1)	M(2, 1)	M(3, 1)	
	M(0, 2)	M(1, 2)	M(2, 2)	M(3, 2)	
	M(0, 3)	M(1, 3)	M(2, 3)	M(3, 3)	
	M(0, 4)	M(1, 4)	M(2, 4)	M(3, 4)	
	M(0, 5)	M(1, 5)	M(2, 5)	M(3, 5)	
	M(0, 6)	M(1, 6)	M(2, 6)	M(3, 6)	

Figura 8-2. Ejemplo de una matriz

Esta sección describe los dos comandos asociados con agrupamientos, matrices y strings y como cargar, tanto agrupamientos como matrices. Los comandos de esta sección son:

DIM
CLR

DIM (DI)

Formato: DIM $\left\{ \begin{array}{l} \text{vars(expa)} \\ \text{varm(expa[,expa])} \end{array} \right\} \left[\left\{ \begin{array}{l} \text{vars(expa)} \\ \text{varm(expa[,expa])} \end{array} \right\} \right]$

Ejemplos: DIM A(100)
 DIM M(6,3)
 DIM B\$(20) usado con STRINGS.

La sentencia DIM se emplea para reservar un cierto número de ubicaciones de memoria para un string, un agrupamiento o una matriz. Un carácter de un string toma un byte de la memoria, en cambio un número de un agrupamiento toma seis bytes. El primer ejemplo reserva 101 ubicaciones para el agrupamiento llamado A. El segundo ejemplo reserva 7 líneas por cuatro columnas para una matriz o agrupamiento bidimensional designado M. El tercer ejemplo reserva 20 bytes llamados B\$. **TODOS LOS STRINGS, AGRUPAMIENTOS Y MATRICES DEBEN ESTAR DIMENSIONADOS.** Es un buen hábito poner todas las sentencias DIM al comienzo de un programa. Note que en la Figura 8-1 a pesar de haberse dimensionado el agrupamiento con DIM A(6), de hecho existen 7 elementos en el agrupamiento debido a que existe el elemento de orden 0. A pesar de que en la Figura 8-2 se dimensiona como DIM M(6,3), se reservan 28 ubicaciones.

Nota: El Computador Personal ATARI no inicializa automáticamente las variables de agrupamiento o de matriz en 0 a la partida de una ejecución de programa. Para inicializar en 0 los elementos de un agrupamiento o de una matriz, siga los siguientes pasos:

```
250 DIM A(100)
300 FOR E=0 TO 100
310 A(E)=0
320 NEXT E
```

Los agrupamientos y matrices se rellenan con datos usando sentencias FOR/NEXT, READ/DATA y comandos de entrada. La Figura 8-3 muestra la construcción de parte de un agrupamiento usando el bucle FOR/NEXT y la Figura 8-4 construye un agrupamiento usando sentencias READ/DATA.

```
10 DIM A(100)
20 X=20
30 FOR E=1 TO 90
40 X=X+1
50 A(E)=X
60 NEXT E
70 FOR E=1 TO 90
80 PRINT E, A(E)
90 NEXT E
```

Figura 8-3. Uso de FOR/NEXT para construir un agrupamiento

```
10 DIM A(3)
20 FOR E=1 TO 3
30 READ X
40 A(E)=X
50 PRINT A(E)
60 NEXT E
70 END
80 DATA 33,45,12
```

Figura 8-4 Uso de READ/DATA para construir un Agrupamiento.

La Figura 8-5 muestra un ejemplo de la construcción de una matriz de 6x3.


```

10 DIM M(5,3)
20 FOR LINEA=0 TO 5
30 FOR COLUMNA=0 TO 3
40 M(LINEA,COLUMNA)=INT(RND(0)*1000)
50 NEXT COLUMNA:NEXT LINEA
60 FOR LINEA=0 TO 5
70 FOR COLUMNA=0 TO 3
80 PRINT M(LINEA,COLUMNA);
100 NEXT COLUMNA:PRINT :NEXT LINEA

```

Figura 8-5. Construyendo una Matriz

Note que las palabras, LINEA Y COLUMNA no son ni comandos ni sentencias ni funciones ni palabras claves de BASIC. Son simplemente nombres de variables que se emplean aquí para indicar cual es la función de bucle que se realiza primero. Igualmente el programa podría haberse escrito usando como nombres de variables X e Y.

CLR

Formato: CLR

Ejemplo: 200 CLR

Este comando limpia la memoria de todos los strings, agrupamientos o matrices dimensionados que haya en la memoria, de modo que los nombres de variables puedan ser usados para otros propósitos. También limpia los valores almacenados en variables no dimensionadas. Si una matriz, un string o un agrupamiento se requieren después de un comando CLEAR, deben ser redimensionados con un comando DIM.

MODOS Y COMANDOS GRAFICOS

Esta sección describe los comandos de BASIC Atari y los diferentes modos gráficos del Computador Personal Atari. Usando estos comandos, es posible la creación gráfica para juegos, dibujos y figuras.

Los comandos que se describen en esta sección son:

GRAPHICS	LOCATE	PUT/GET
COLOR	PLOT	SETCOLOR
DRAWTO	POSITION	XIO

Los comandos PUT/GET y XIO explicados en esta sección son aplicaciones especiales de los mismos comandos ya descritos en la sección 5.

GRAPHICS (GR.)

Formato: GRAPHICS expa

Ejemplo: GRAPHICS 2

Este comando se usa para seleccionar uno de los modos gráficos (nueve modos en los modelos 400 y 800 equipados con CTIA, 12 en los equipados con GTIA y 16 modos en los modelos XL). La tabla 9-1 resume estos modos y las características de cada uno. El comando GRAPHICS automáticamente abre la pantalla , S: (la ventana gráfica), como dispositivo #6.

Así, al imprimir texto en la ventana de texto, no es necesario especificar el código del dispositivo. La expa debe ser positiva redondeada al entero más próximo. El modo gráfico 0 es un despliegue de pantalla completa, mientras que los modos 1 al 8 son de pantalla partida. Los modos nueve a once son de pantalla completa y los de 12 a 15 de pantalla partida. Para anular la partición de pantalla agregue +16 al número de modo (expa) en el comando GRAPHICS. El agregar 32 previene al comando gráfico de limpiar la pantalla.

Para volver al modo gráfico 0 en modo Directo, presione **Reset** o digite GR.0 y presione **RETURN**.

TABLA 9.1 — MODOS GRAFICOS Y FORMATOS DE PANTALLA

Modo	Tipo	<i>PLOT</i>	<i>expa 1</i>	<i>expa 2</i>	Colores	RAM requer. (Bytes)	
		Hbr. (Col.)	part.	comp.		part.	comp.
0	texto	40	—	24	1 1/2	—	992
1	texto	20	20	24	5	674	672
2	texto	20	10	12	5	424	420
3	gráfico	40	20	24	4	434	432
4	gráfico	80	40	48	2	694	696
5	gráfico	80	40	48	4	1174	1176
6	gráfico	160	80	96	2	2174	2184
7	gráfico	160	80	96	4	4190	4200
8	gráfico	320	160	192	1 1/2	8112	8138
9	gráfico	80	—	192	1 *	—	8138
10	gráfico	80	—	192	9	—	8138
11	gráfico	80	—	192	16	—	8138
12**	texto ***	40	20	24	5	1154	1152
13**	texto ***	40	10	12	5	664	660
14**	gráfico	160	160	192	2	4270	4296
15**	gráfico	160	160	192	4	8112	8138

* 16 luminancias.

** Solamente en los modelos XL.

*** Especialmente apto para uso con juegos de caracteres de creación específica.

MODOS GRAFICO 0

Este modo es de 1 color y 2 luminancias (brillos) y es el modo asumido por el Computador Personal ATARI. Contiene una matriz de pantalla de 24 por 40 caracteres. Sus posiciones de margen asumidos son 2 y 39, lo que permite 38 caracteres por línea. Los márgenes pueden cambiarse cargando LMARGN y RMARG (82 y 83). Vea el apéndice I. Algunos sistemas tienen diferentes valores asumidos de margen. El color de los caracteres está determinado por el color de fondo. Sólo puede cambiarse la luminancia de los caracteres.

Este despliegue de pantalla completa tiene un color azul rodeado de un borde negro (a no ser que el borde se haya especificado de otro color). Para desplegar caracteres en una ubicación específica use uno de los siguientes dos métodos.

Método 1

nolínea POSITION expa1,expa2

Coloca el cursor en la ubicación especificada por expa1 y expa2

nolínea PRINT exps

Imprime exps

Método 2

nolínea GR.0

Especifica modo gráfico

nolínea POKE 752,1

Suprime el cursor

nolínea PRINT CHR\$(125)

nolínea COLOR ASC (exps)

Especifica carácter a imprimir

nolínea PLOT expa1, expa2

Especifica donde imprimir el carácter

nolínea GOTO nolínea

Inicia bucle para evitar que se imprima READY (GOTO mismo nolínea)

Presione **BREAK** para terminar.

GRAPHICS 0 también se usa como comando para limpiar la pantalla ya sea en el modo Directo o en el Diferido. Termina con cualquier modo gráfico seleccionado previamente y vuelve la pantalla al modo asumido (GRAPHICS 0).

MODOS GRAFICOS 1 Y 2

Tal como se define en la Tabla 9-1, estos dos modos de 5 colores son modos de texto. Sin embargo, ambos son de pantalla partida (vea la Figura 9-1). Los caracteres impresos en el modo Gráfico 1 tienen el doble ancho de los impresos en el modo Gráfico 0, pero tienen su misma altura. Los caracteres impresos en el modo Gráfico 2 tienen doble ancho y doble altura con respecto a los del modo Gráfico 0. En el modo de pantalla partida, el comando PRINT se usa para desplegar caracteres tanto en la ventana de texto como en la ventana del modo gráfico correspondiente. Para imprimir caracteres en la ventana del modo gráfico, especifique dispositivo #6 después del comando PRINT.

Ejemplo: 100 GR. 1
 100 PRINT #6;"ATARI"

Los colores asumidos dependen del tipo de caracteres ingresados. La Tabla 9-2 define los colores asumidos y los registros de color usados para cada tipo.

TABLA 9-2. Colores asumidos para tipos de entrada específicos

Tipo de carácter	Reg. de color	Color asumido
alfabético mayúscula	0	naranja
alfabético minúscula	1	verde claro
alfabético mayúscula v. inverso	2	azul oscuro
alfabético minúscula v. inverso	3	rojo
números	0	naranja
números v. inverso	2	azul oscuro

Nota: Vea SETCOLOR para cambiar los colores de los caracteres

A no ser que especifique de otra forma, todos los caracteres se despliegan como mayúsculas no inversas. Para imprimir minúsculas y caracteres gráficos, use POKE 756, 226. Para volver a mayúsculas, use POKE 756, 224.

En los modos gráficos 1 y 2, no hay video inverso, pero es posible obtener todos los caracteres restantes en cuatro diferentes colores (Vea al final de esta sección).

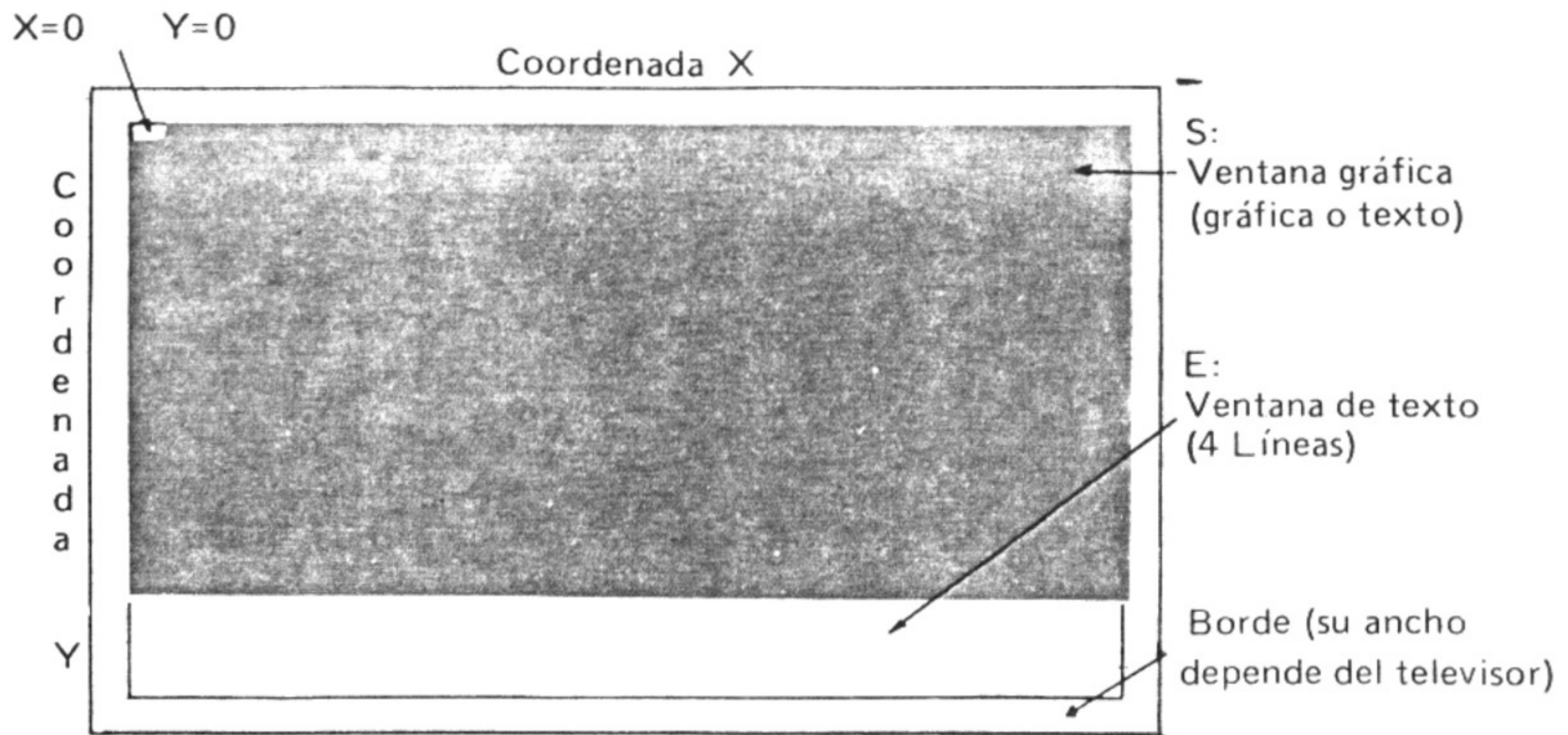


Figura 9-1. Pantalla Partida para Modos Gráficos 1 y 2

Las coordenadas X e Y comienzan en 0 (en la esquina superior izquierda de la pantalla). Los valores máximos son los números de líneas y columnas respectivamente menos 1 (vea la Tabla 9-1).

Esta configuración de pantalla partida puede cambiarse a pantalla total agregando los caracteres +16 al número de modo.

Ejemplo: GRAPHICS 1 + 16

MODOS GRAFICOS 3, 5 y 7

Estos tres modos gráficos de cuatro colores también dan despliegue de pantalla partida en su modo asumido, pero pueden cambiarse a pantalla completa agregando +16 al número de modo.

Los modos 3, 5, 7 son similares, excepto en que los modos 5 y 7 usan una mayor cantidad de puntos (pixels) al trazar, dibujar y ubicar el cursor; los puntos son más pequeños, por lo cual dan una resolución mucho mayor.

MODOS GRÁFICOS 4 y 6

Estos dos modos gráficos de 2 colores, son de pantalla partida y pueden desplegar solamente dos colores, mientras que los otros modos pueden desplegar 4 y 5 respectivamente. La ventaja de un modo de dos colores es que requiere menos espacio de memoria RAM (vea Tabla 9-1). Por ello se usan cuando solamente se requieren dos colores y la RAM se está haciendo escasa. Estos dos modos tienen también una mayor resolución, lo cual significa puntos más pequeños que el modo gráfico 3.

MODOS GRAFICO 8

Este modo gráfico es el que da la mayor resolución de entre todos los demás modos. Como toma una gran cantidad de RAM para obtener este grado de resolución, puede acomodar solamente un máximo de un color a dos luminancias diferentes.

MODOS GRAFICOS 9, 10, 11

Son los llamados modos GTIA, por ser accesibles solamente por los computadores ATARI que tienen este circuito integrado GTIA. Los modelos XL tienen GTIA; los modelos 400 y 800 lo tienen desde 1982 aproximadamente. La modificación es simple. basta sustituir el circuito CTIA por el GTIA.

Los tres modos GTIA tienen la misma resolución (ver Tabla 9-1) y son de pantalla completa solamente. Ello hace que para mantener uno de ellos en pantalla, una vez ejecutado un programa gráfico, debe efectuarse un bucle de espera del tipo: `nolínea n GOTO nolínea n`, con el fin de evitar que el computador retorne al modo directo, en el cual forzosamente requiere una ventana de texto (GR.0).

El modo 9 posee 1 sólo color en 16 luminancias dadas por el `COLOR expa`. (ver tabla 9-5). El color y la luminancia del fondo quedan dadas por `SETCOLOR 4, expa2, expa3`.

El siguiente listado corresponde a un programa que muestra un abanico de las luminancias del modo 9.

```
100 GRAPHICS 9
110 SETCOLOR 4,15,0
120 FOR LUM=0 TO 15
130 COLOR LUM
140 FOR X=0 TO 4
150 PLOT 5*LUM+X,0
160 DRAWTO 40,150
170 NEXT X
180 NEXT LUM
190 GOTO 190
```

Figura 9-2. La luminancia del modo 9

El modo 10 permite el uso de 9 registros de color, es decir 9 pares de color y luminancia. Sin embargo, BASIC no permite referencias a más de 5 registros de color (0 a 4) por lo que los 4 restantes deben cargarse directamente en sus ubicaciones de memoria. (también los 5 registros tradicionales pueden cargarse en esta misma forma). Ello se logra con: `POKE expa1, expa2+expa3` en que `expa1` está comprendido entre 704 y 712 y corresponde naturalmente a la dirección de memoria respectiva. Los números de registros de 0 a 4, corresponden a las ubicaciones de memoria del 708 al 712 en el mismo orden.

`expa 2` es el número del color multiplicado por 16 y `expa 3` es la luminancia del registro. (Tanto el número de color como la luminancia están comprendidos entre 0 y 15).

El siguiente programa, al ser ejecutado, muestra una gama de colores posibles con este modo.


```

100 GRAPHICS 10
110 FOR I=704 TO 712
120 READ COL,LUM
130 POKE I,16*COL+LUM
140 NEXT I
150 FOR X=1 TO 8
160 COLOR X:REM 9 ES COLOR DE FONDO
170 REM COMIENZO RUTINA DE RELLENO
180 POKE 765,X
190 PLOT X*8+5,0:DRAWTO X*8+5,159
200 PLOT X*8+1,159:POSITION X*8+1,0
210 XIO 18,#6,0,0,"S:"
220 REM FIN RUTINA DE RELLENO
230 NEXT X
240 GOTO 240
250 DATA 1,3,3,5,5,1,7,9,10,12,12,7,0,12,15,6,4,15

```

Figura 9-3. Listado de un Programa que muestra los nueve Registros de color del Modo 10

El modo 11 es complementario al 9 en el sentido de tener 16 colores diferentes, todos con la misma luminancia. Su manejo es igual; cambiando la línea 100 del programa de Fig. 9-2 por GRAPHICS11 y la 110 por SETCOLOR 0,0,6 se obtiene una apreciación de sus posibilidades.

MODOS GRAFICOS 12 y13

Estos dos modos son de texto con posibilidades de pantalla partida como los modos 1 y 2; también el tamaño de los caracteres y la forma de acceso, a través de #6 son iguales en estos modos. Sin embargo, cada pixel puede tener uno de cuatro colores. Como los juegos de caracteres incorporados al computador ATARI están concebidos para modos con caracteres monocromáticos (modo 0, 1 y 2) su uso en los modos 12 y 13 origina textos de muy mala legibilidad. Lo interesante en estos modos es la posibilidad de definir nuevos juegos de caracteres multicolores, no necesariamente alfabéticos, para mapas, juegos, programas educativos, etc.

Los programas de las Figuras 9-4 y 9-5 muestran un ejemplo para cada uno de estos modos.

```

10 GOSUB 1000
20 DIM CAR1$(3),CAR2$(3)
30 GRAPHICS 12
40 SETCOLOR 0,4,6:SETCOLOR 2,4,2:SETCOLOR 3,12,8:SETCOLOR 4,0,10
50 POKE 756,DIRC/256
60 CAR1$="ab " :REM a,b en v.inv.,espacio
70 CAR2$="cd "
80 PRINT #6
90 FOR LINEA=1 TO 4
100 PRINT #6;" ";
110 FOR COLUMNA=1 TO 10
120 PRINT #6;CAR1$;
130 NEXT COLUMNA
140 PRINT #6
150 PRINT #6;" ";
160 FOR COLUMNA=1 TO 10
170 PRINT #6;CAR2$;
180 NEXT COLUMNA
190 PRINT #6
200 NEXT LINEA
210 PRINT :PRINT "          MODO GRAFICO 12 CON          CARACTERES ESPECIA
LES."
220 GOTO 220
1000 POKE 106,PEEK(106)-5:GRAPHICS 0
1010 DIRC=256*(PEEK(106)+1)
1020 PRINT :PRINT " Transpaso del juego de caracteres
1030 FOR I=0 TO 1023:POKE DIRC+I,PEEK(57344+I):NEXT I
1040 PRINT :PRINT " Ahora se redefinen 4 caracteres."
1050 FOR I=0 TO 31
1060 READ BYTE:POKE 776+DIRC+I,BYTE
1070 NEXT I
1080 DATA 3,63,247,255,255,61,223,61
1090 DATA 240,124,252,223,253,255,252,244
1100 DATA 3,3,3,15,63,0,0,0
1110 DATA 192,192,192,240,254,0,0,0
1190 RETURN

```

Documenta 5 a RAMTOP
 $DIRC = 256 * RAMTOP + 1$
 de ROM a RAM."
 $DIRC \leftarrow 56k \dots 8x128 \text{ vcc.}$
 $DIRC + 0x97 \dots \leftarrow \text{BYTE}$

Figura 9 - 4. Uso de Caracteres Especiales con modo 12

```

10 GOSUB 1000
30 GRAPHICS 13
40 SETCOLOR 0,4,6:SETCOLOR 2,0,14:SETCOLOR 1,9,6:SETCOLOR 3,14,12:SETCOLOR 4,0,1
0
100 POKE 756,DIRC/256
140 POSITION 15,3:? #6;"abcdddd":POSITION 15,4:? #6;"eeeeeee"
150 PRINT "          MODO GRAFICO 13 CON          CARACTERES ESPECIALES"
990 GOTO 990
1000 POKE 106,PEEK(106)-5:GRAPHICS 0:PRINT "Transpaso de caracteres de ROM a RAM
."
1010 DIRC=256*(PEEK(106)+1)
1020 FOR I=0 TO 1023:POKE DIRC+I,PEEK(57344+I):NEXT I
1030 ? :? "Redefinición de 5 caracteres."
1040 FOR I=0 TO 39:READ BYTE:POKE 776+DIRC+I,BYTE:NEXT I
1050 DATA 170,170,170,175,171,170,171,170
1060 DATA 170,186,186,255,255,254,171,170
1070 DATA 170,170,170,234,170,170,170,170
1080 DATA 255,255,255,255,255,255,255,255
1090 DATA 85,85,85,85,85,85,85,85
1990 RETURN

```

Figura 9 - 5. Generación de Caracteres Multicolores para Crear Efectos Especiales en el Modo 13

MODOS 14 y 15

Las características y aplicaciones de estos modos son similares a las de los modos 6 y 7 respectivamente, salvo en lo que respecta a resolución vertical: los modos 14 y 15 tienen 192 líneas a pantalla completa, mientras que los modos 6 y 7 tienen solamente 96. Naturalmente ello se refleja en los requerimientos de memoria que con la mayor resolución aumentan casi al doble (ver tabla 9-1).

COLOR

Formato: COLOR expa

Ejemplos: 110 COLOR ASC("A")
110 COLOR 3

El valor de la expresión en la sentencia COLOR determina el dato que se almacena en la memoria de despliegue para todos los comandos PLOT y DRAWTO siguientes hasta que se ejecute la próxima sentencia COLOR. El valor debe ser positivo y generalmente es un número entero comprendido entre 0 y 255. Los números no enteros se redondean al entero más próximo. Los circuitos de despliegue gráfico interpretan los datos de manera diferente, dependiendo del modo gráfico escogido. En los modos de texto 0 a 2, el número puede ser de 0 a 255 (8 bits) en el modo 0 determina el carácter que se desplegará, en los modos 1 y 2 tanto el carácter como el color de éste. (Los dos bits más significativos determinan el color). Por esta razón sólo se dispone de 64 caracteres diferentes en estos modos, en vez del juego de caracteres completo de 256).

Las tablas 9-6 y 9-7 al final de esta sección ilustran el juego de caracteres internos y la asignación de carácter/color. La tabla 9-2 es una tabla simplificada que permite fácilmente la generación de alguno de los colores. Por ejemplo, COLOR ASC("A") : PLOT 5,5 desplegará un carácter A de color naranja en el modo gráfico 1 o el modo gráfico 2, en la ubicación 5,5.

Los modos gráficos del 3 al 8 no son modos de texto, de modo que los datos almacenados en la memoria de despliegue simplemente determinan el color de cada pixel. Los modos de dos colores o de dos luminancias requieren ya sea un 0 o un 1 (1 bit) y los modos de cuatro colores requieren 0,1,2, ó 3. (la expresión en la sentencia COLOR puede tener un valor mayor que 3, pero solamente uno o dos bites serán usados). El color que definitivamente se despliegue dependerá del valor que exista en el registro de color que corresponda a los datos de 0, 1, 2 ó 3 en el modo gráfico particular en uso. Esto puede determinarse mirando la tabla 9-5, que da los valores asumidos y los registros de color correspondientes. Los colores pueden cambiarse usando SETCOLOR.

Note que cuando BASIC se inicia, el dato de colores es 0 y cuando se ejecuta un comando GRAFICO (sin +32) todos los pixels se ponen en 0. Por ello nada parece pasar al ejecutar comandos PLOT Y DRAWTO en los modos GRAFICOS 3 a 7 si no se ha ejecutado previamente una sentencia COLOR. Corrija esto haciendo un COLOR 1 primero.

DRAWTO (DR.)

Formato: DRAWTO expa1, expa2

Ejemplo: 100 DRAWTO 10,8

Con esta sentencia se traza una línea desde el último punto desplegado por efecto de un PLOT (Vea PLOT) hasta la ubicación especificada por expa1 y expa2. La primera expresión representa la coordenada X y la segunda la coordenada Y (vea la figura 9-1). El color de la línea es el mismo del punto desplegado por PLOT.

LOCATE (LOC.)

Formato: LOCATE expa1, expa2, var

Ejemplo: 150 LOCATE 12, 15, X

Este comando ubica el cursor gráfico invisible en la posición especificada en la ventana gráfica, encuentra el dato de ese pixel y lo almacena en la variable aritmética especificada. Esto dará un número comprendido entre 0 y 255 para los modos de texto; 0 ó 1 para los modos gráficos de 2 colores; y 0, 1, 2 ó 3 para los modos de 4 colores. Las dos expresiones aritméticas especifican las coordenadas X e Y del punto. LOCATE es equivalente a:

POSITION expa1, expa2:GET#6, vara.

Realizar un PRINT después de LOCATE ó GET desde la pantalla, puede hacer que el dato del pixel que se haya examinado se modifique. Este problema se evita reubicando el cursor y colocando el dato que se leyó de vuelta en el pixel antes de efectuar el PRINT. El siguiente programa ilustra el uso del comando LOCATE.

```
10 GRAPHICS 3+16
20 COLOR 1
30 SETCOLOR 2,10,8
40 PLOT 10,15
50 DRAWTO 15,15
60 LOCATE 15,15,X
70 PRINT X
```

Figura 9-6. Programa Ejemplo que usa LOCATE.

En ejecución, el programa imprime el dato (1) determinado por la sentencia COLOR que se había almacenado en el pixel 12, 15.

PLOT

Formato: PLOT expa1, expa2

Ejemplo: 100 PLOT 5,5

El comando PLOT se usa en los modos gráficos del 3 al 8 para desplegar un punto en la ventana gráfica. La expa1 especifica la coordenada X y la expa2 la coordenada Y. El color del punto trazado se determina por el matiz y la luminancia correspondientes al registro COLOR de la última sentencia COLOR ejecutada. Para cambiar el registro de color y el color del punto trazado, use SETCOLOR. La cantidad de puntos que pueden trazarse sobre

la pantalla depende del modo gráfico en uso. El rango de los puntos comienza en 0 y se extiende hasta una unidad menos que el número total de líneas (para la coordenada X) o columnas (para la coordenada Y) y que se muestra en la Tabla 9-1.

POSITION (POS.)

Formato: POSITION expa1, expa2

Ejemplo: 100 POSITION 8, 12

La sentencia POSITION se usa para ubicar el cursor invisible de la ventana gráfica en un lugar específico sobre la pantalla (Generalmente precede a una sentencia PRINT). Esta sentencia puede usarse en todos los modos. Note que el cursor de hecho no se moverá hasta que se ejecute un comando de entrada/salida y/o que tenga relación con la pantalla.

PUT/GET (PU. /GE.)

Formatos: PUT #expa expa
GET #expa, vara

Ejemplos: 100 PUT #6, ASC ("A")
200 GET #1, X

En operaciones gráficas, PUT se usa para ubicar datos sobre la pantalla. Esta sentencia trabaja mano a mano con la sentencia POSITION. Después de un PUT o GET el cursor se mueve a la próxima ubicación de pantalla. Hacer un PUT al dispositivo #6 hace que el ingreso de un byte (la segunda expa) se despliegue en la posición del cursor.

Este byte es o el byte de código ATASCII para un carácter particular (modos de texto) o el dato de color (modos gráficos).

GET se emplea para ingresar el byte código de un carácter desplegado en la posición del cursor a una variable aritmética especificada. Los valores que se usan en PUT y GET corresponden a los valores de las sentencias de color. (PRINT e INPUT también pueden usarse).

Nota: Al hacer un PRINT después de LOCATE o GET de la pantalla puede modificar el dato del pixel que se haya examinado. Para evitar este problema, reubique el cursor y reponga el dato que se había leído de vuelta en el pixel, antes de ejecutar PRINT.

SETCOLOR (SE.)

Formato: SETCOLOR expa1, expa2, expa3

Ejemplo: 100 SETCOLOR,0, 1, 4

Esta sentencia se usa para escoger un matiz y una luminancia particulares para ser almacenados en el registro de color especificado. Los parámetros de la sentencia SETCOLOR se definen como sigue:

expa1 = Registro de color (0 a 4 dependiendo del modo gráfico).

expa2 = Número de matiz de color (0 a 15. Vea la Tabla 9-3).

expa3 = Luminancia de color (debe ser un número par entre 0 y 14; mientras mayor el número, más brillante será el despliegue. 14 es blanco casi puro).

TABLA 9-3. LOS COLORES DE COMANDO SETCOLOR Y NUMEROS CORRESPONDIENTES

COLOR	NUMEROS (expa2) SETCOLOR
GRIS	0
NARANJA CLARO (DORADO)	1
NARANJA	2
ROJO ANARANJADO	3
ROSADO	4
PURPURA	5
PURPURA AZULADO	6
AZUL	7
AZUL	8
CELESTE	9
TURQUESA	10
VERDE AZULADO	11
VERDE	12
AMARILLO VERDOSOSO	13
NARANJA VERDOSOSO	14
NARANJA CLARO	15

Nota: Los colores variarán según el tipo de televisor o monitor usado y su ajuste.

Los circuitos de despliegue ATARI contienen cinco registros de color numerados del 0 al 4. El sistema Operativo (OS) tiene cinco ubicaciones RAM (COLOR 0 A COLOR 4, vea el apéndice I - Ubicaciones de Memoria) donde se anotan los valores de los colores en uso. La sentencia SETCOLOR se usa para cambiar los valores en estas ubicaciones de memoria. (El Sistema Operativo transfiere estos valores a los registros circuitales en cada cuadro de televisión). La sentencia SETCOLOR requiere de un valor comprendido entre 0 y 4 para especificar un registro de color. La sentencia COLOR usa números diferentes porque especifica datos que solamente en forma indirecta corresponden a los registros de color. Esto puede aparecer confuso, por lo que se recomienda una cuidadosa experimentación y un detallado estudio de las diversas tablas de esta sección.

No se requieren comandos SETCOLOR si se usan los cinco colores asumidos.

A pesar de que son posibles 128 combinaciones de color-luminancia, no más de nueve pueden desplegarse a un tiempo en BASIC. El propósito de los registros de color y de la sentencia SETCOLOR es especificar estos colores.

TABLA 9 - 4. COLORES ASUMIDOS DE LOS REGISTROS

SETCOLOR (Registro de color)	VALORES ASUMIDOS		COLOR
	COLOR	LUMINANCIA	
0	2	8	NARANJA
1	12	10	VERDE
2	9	4	AZUL OSCURO
3	4	6	ROSADO O ROJO
4	0	0	NEGRO

Los valores (y colores) asumidos son los presentes en ausencia de instrucciones SETCOLOR

Nota : Los colores variarán según el tipo de televisor o monitor usado y su ajuste.

Un programa que ilustra el modo gráfico 3 y los comandos explicados hasta aquí en esta sección se muestra a continuación:

```

10 GRAPHICS 3
20 SETCOLOR 0,2,8:COLOR 1
30 PLOT 17,1:DRAWTO 17,10:DRAWTO 9,18
40 PLOT 19,1:DRAWTO 19,18
50 PLOT 20,1:DRAWTO 20,18
60 PLOT 22,1:DRAWTO 22,10:DRAWTO 30,18
70 POKE 752,1
80 PRINT :PRINT "    COMPUTADORES PERSONALES ATARI"
90 GOTO 90

```

Las sentencias SETCOLOR y COLOR determinan el color de los puntos que se trazarán (vea la Tabla 9.5). El comando SETCOLOR carga el registro de color 0 con el matiz 2 (naranja) y la luminancia 8 ("normal"). Las siguientes 4 líneas trazan los puntos para el despliegue. La línea 70 suprime el cursor y la línea 80 imprime la expresión literal COMPUTADORES PERSONALES ATARI en la ventana de texto (a 5 espacios del margen).

Note que en ningún momento se especificó el color de fondo porque el asumido corresponde al deseado (negro).

Si se ejecuta este programa, imprimirá el logo Atari en la ventana gráfica y la expresión literal en la ventana de texto tal como en la Figura 9-7.

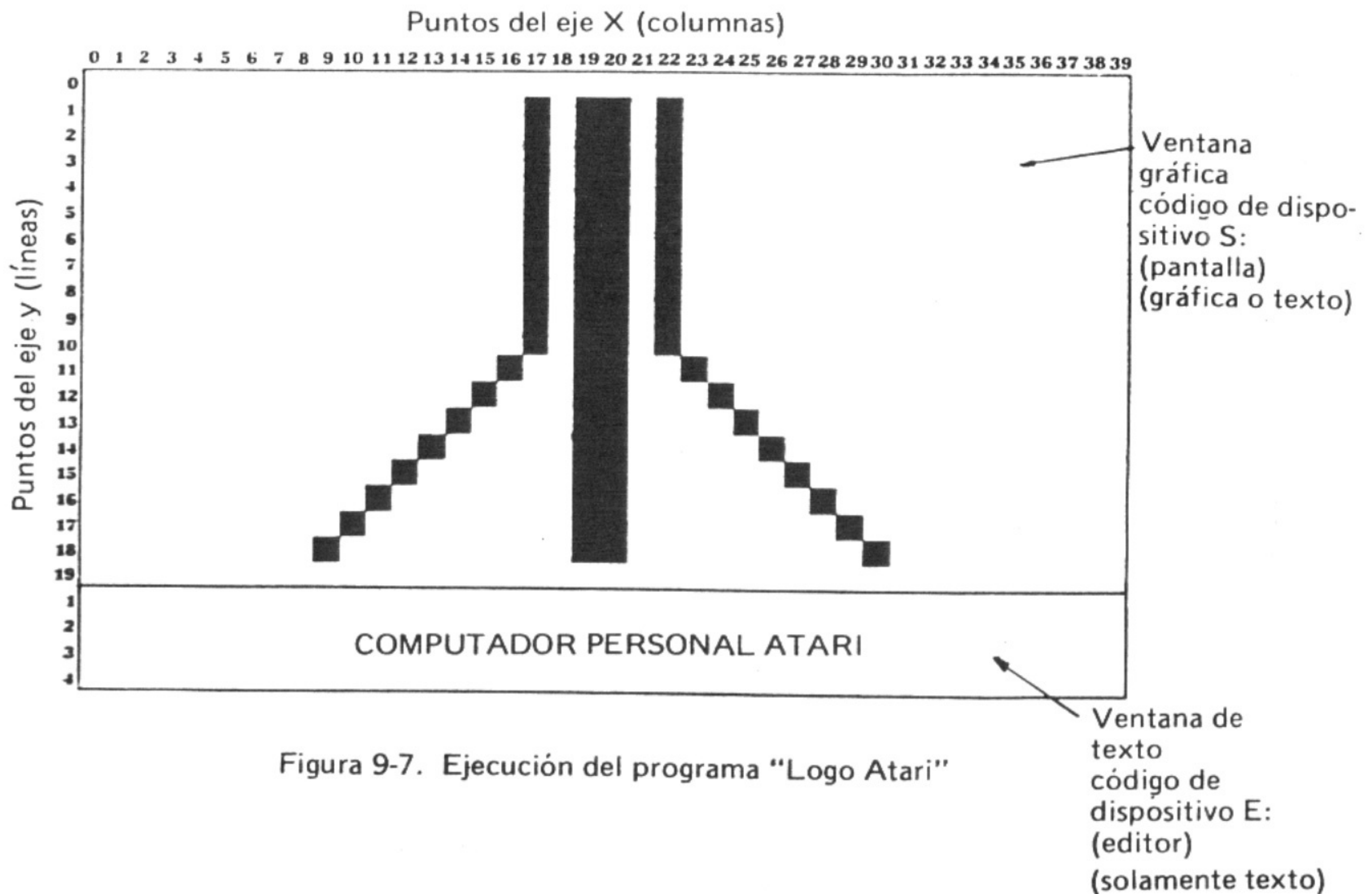


Figura 9-7. Ejecución del programa "Logo Atari"

TABLA 9 - 5. RELACIONES ENTRE SETCOLOR Y COLOR SEGUN MODOS

Modo	Registro SETCOLOR (expa 1)	COLOR (expa)	Descripción y comentarios
Modo 0 y todas las ventanas de texto	0 1 2 3 4	Los datos de COLOR determinan el carácter a desplegar	Lumin. caract. (color igual al fondo) Fondo Borde
Modos 1 y 2 (Modos de texto)	0 1 2 3 4	Los datos de COLOR determinan el carácter a desplegar	Caracteres-Ver tabla 9-2 Caracteres " Caracteres " Caracteres " Fondo, Borde "
Modos 3, 5 y 7 (Modos de 4 colores)	0 1 2 3 4	1 2 3 - 0	Puntos gráficos Puntos gráficos Puntos gráficos - Ptos. gráficos (fondo asumido), borde
Modos 4 y 6 (Modos de 2 colores)	0 1 2 3 4	1 - - - 0	Puntos gráficos - - - Ptos. Gráficos (fondo asumido), borde
Modo 8 (1 color y 2 luminan.)	0 1 2 3 4	- 1 0 0 -	- Lumin. ptos. gráficos (mismo color del fondo) Ptos. gráficos (fondo asumido) - borde
9 Modo gráfico de 1 color y 16 luminancias	1	de 0 a 15 fijan los 16 valores de luminancia de los puntos gráficos	SETCOLOR4, expa2, 0 expa2, comprendido entre 0 y 15 determina el color de la pantalla (Vea la Tabla 9-3)
10 Modo gráfico DE 9 colores	(704) (705) (706) (707) 0 (708) 1 (709) 2 (710) 3 (711) 4 (712)	0 1 2 3 4 5 6 7 8	Para cargar los registros correspondientes a COLOR 0 hasta COLOR 3, debe aplicarse la instrucción POKE con 16*expa2 + expa3 a la ubicación de memoria indicada entre paréntesis. Para COLOR 4 a 8 puede usarse alternativamente la sentencia SETCOLOR

11 Modo gráfico de 16 colores en 1 luminancia	4	De 0 a 15 fijan los 16 diferentes colores de los ptos. gráficos.	SETCOLOR 4,0, expa3 expa3, comprendido entre 0 y 15 determina la luminancia de la pantalla
Modos 12 y 13	0 1 2	(1) (2) (3)	Colores y forma de los caracteres están determinados por el mapa de pares de bits del juego de caracteres (colores 0,1,2 y 3)
(Modo de texto)	3	(4)	
de 5 colores	4	(0)	
14 Modo grafico	0 1	1	puntos gráficos brillo letras ventana texto
de 2 colores	2		color ventana texto
	3		
	4	0	fondo, ptos. gráficos
15 Modo gráfi- co	0 1 2	1 2 3	puntos gráficos p. gráf., brillo letras v. texto " ", col. ventana texto
de 4 colores	3 4		fondo, puntos gráficos

XIO (X.)

Formato: XIO 18, #expa, expa1, expa2, esparchivo

Ejemplo: 100 XIO 18, #6,0,0, "S:"

Esta aplicación especial de la sentencia XIO rellena un área de la pantalla entre puntos y líneas trazadas de color no igual a 0. Se usan variables de relleno (0) para expa1 y expa2.

Los siguientes pasos ilustran el proceso de relleno:

1. PLOT del punto inferior derecho (punto 1).
2. DRAWTO al punto superior derecho (punto 2). Esto perfila el límite derecho del área a rellenar.
3. DRAWTO a la esquina superior izquierda (punto 3).
4. POSITION del cursor en el punto inferior izquierdo (punto 4).
5. POKE de la dirección 765 con el dato de color de relleno.
6. Este método se usa para rellenar cada línea horizontal desde arriba hasta abajo del área especificada. El relleno parte de la izquierda y avanza a través de la línea hacia la derecha hasta llegar a un pixel que contenga un dato no cero (dando la vuelta a la pantalla en caso necesario). Esto significa que el relleno no puede usarse para cambiar el color de un área que anteriormente ya se haya rellenado con un valor diferente de 0, ya que el relleno se detendrá. El comando de relleno ingresará a un bucle infinito si se intenta rellenar con cero (0) una línea que no tiene pixels diferentes de 0. Puede presionarse **BREAK** o **Reset** para detener el relleno, si esto llegara a suceder.

El siguiente programa crea una forma y la rellena con datos de color 3. Note que el comando XIO traza las líneas de la izquierda y de la base de la figura:

```

10 GRAPHICS 5+16
20 COLOR 3
30 PLOT 70,45
40 DRAWTO 50,10
50 DRAWTO 30,10
60 POSITION 10,45
70 POKE 765,3
80 XIO 18,#5,0,0,"S:"
90 GOTO 90

```

Figura 9-8. Ejemplo de un Programa de Relleno

ASIGNANDO COLORES A LOS CARACTERES DE LOS MODOS DE TEXTO 1 y 2

Este procedimiento describe el método para asignar colores al juego de caracteres Atari. Primero encuentre el número del carácter en la Tabla 9-6. Enseguida vea en la Tabla 9-7 para obtener la conversión de este número, requerida para asignar un registro de color.

Ejemplo: Asignar SETCOLOR 0 a la letra "r" en el modo 2, cuyo valor está determinado por el registro 0.

1. En la Tabla 9-6, encuentre la columna y el número correspondiente a "r" (114 columna 4).
2. Usando la Tabla 9-7 ubique la columna 4. La conversión da el número de carácter menos 32 ($114 - 32 = 82$).
3. Haga un POKE de la dirección de la base de caracteres (CHBAS) con 226 para especificar letras minúsculas o caracteres gráficos especiales; por ejemplo:

```

POKE 756, 226
      O
CHBAS = 756
POKE CHBAS, 226

```

Para volver a letras mayúsculas, números y marcas de puntuación haga POKE CHBAS con 224.

4. Una sentencia PRINT que use el número convertido (82) asigna la "r" a SETCOLOR 0 en el modo 2 (vea la tabla 9-5).

TABLA 9-6. JUEGO INTERNO DE CARACTERES

Columna 1				Columna 2				Columna 3				Columna 4			
#	Car.	#	Car.	#	Car.	#	Car.	#	Car.	#	Car.	#	Car.	#	Car.
0	Space	16	0	32	@	48	P	64		80		96		112	p
1	!	17	1	33	A	49	Q	65		81		97	a	113	q
2	”	18	2	34	B	50	R	66		82		98	b	114	r
3	#	19	3	35	C	51	S	67		83		99	c	115	s
4	\$	20	4	36	D	52	T	68		84		100	d	116	t
5	%	21	5	37	E	53	U	69		85		101	e	117	u
6	&	22	6	38	F	54	V	70		86		102	f	118	v
7	,	23	7	39	G	55	W	71		87		103	g	119	w
8	(24	8	40	H	56	X	72		88		104	h	120	x
9)	25	9	41	I	57	Y	73		89		105	i	121	y
10	*	26	:	42	J	58	Z	74		90		106	j	122	z
11	+	27	;	43	K	59	[75		91		107	k	123	
12	,	28	<	44	L	60	\	76		92		108	l	124	l
13	—	29	=	45	M	61]	77		93		109	m	125	
14	—	30	>	46	N	62	^	78		94		110	n	126	
15	/	31	?	47	O	63	—	79		95		111	o	127	

1. En el modo 0, estos caracteres deben estar precedidos de un escape, CHR\$(27), para ser impresos

CARACTERES DE CONTROL GRAFICOS

Estos caracteres se producen al presionar la tecla **CTRL** junto con una de caracteres alfabéticos tal como se muestra en la tabla de caracteres gráficos. Estos caracteres pueden usarse para trazar figuras, etc., en el modo 0 y en los modos 1 y 2 si se cambia CHBAS.

TABLA 9-7. ASIGNACIONES CARACTER/COLOR

		CONV. 1	CONV. 2	CONV. 3	CONV. 4
MODO 0	² SETCOLOR 2	# + 32 <i>ASCII</i>	# + 32 <i>ASCII</i>	# - 32 <i>ASCII + 32</i>	ninguna <i>ASCII</i>
		POKE 756,224		POKE 756,226	
MODOS 1 o 2	SETCOLOR 0	# + 32 <i>ASCII</i>	# + 32 <i>ASCII</i>	# - 32 <i>ASCII + 32</i>	# - 32 <i>ASCII - 32</i>
	SETCOLOR 1	ninguna <i>ASCII - 32</i>	# + 64 <i>ASCII + 32</i>	# - 64 <i>ASCII</i>	ninguna <i>ASCII</i>
	SETCOLOR 2	# + 160 <i>ASCII + 128</i>	# + 160 <i>ASCII + 128</i>	# + 96 <i>ASCII + 160</i>	# + 96 <i>ASCII + 96</i>
	SETCOLOR 3	# + 128 <i>ASCII + 96</i>	# + 192 <i>ASCII + 160</i>	# + 64 <i>ASCII + 128</i>	# + 128 <i>ASCII + 128</i>

2. Luminancia controlada por SETCOLOR 1,0,LUM.

SONIDO Y CONTROLADORES DE JUEGO

Esta sección describe la sentencia usada para generar notas musicales y sonidos a través del sistema de audio del monitor o del televisor. Se pueden crear acordes tocando hasta cuatro sonidos diferentes simultáneamente. La sentencia **SOUND** también puede emplearse para simular explosiones, pitos, y otros efectos sonoros interesantes. Los demás comandos descritos en esta sección dicen relación con las funciones usadas para manipular los controladores de teclado, bastón y paleta. Estas funciones permiten que dichos controladores se enchufen y se usen en programas de BASIC como juegos, etc.

Los comandos y funciones cubiertas en esta sección son:

SOUND	PADDLE	STICK
	PTRIG	STRIG

SOUND (SO.)

Formato: **SOUND** expa1, expa2, expa3, expa4

Ejemplo: 100 **SOUND** 2, 204, 10, 12

Con la sentencia **SOUND**, la nota especificada comienza a sonar tan pronto como se ejecute la sentencia. La nota continuará sonando hasta que el programa encuentre otra sentencia **SOUND** con la misma expa1 o una sentencia **END**. Este comando puede usarse en el modo directo y en el modo diferido.

Los parámetros de **SOUND** se describen como sigue:

- | | |
|---------|--|
| expa1 = | Voz. Comprendida entre 0 y 3, pero cada vez requiere de una sentencia SOUND diferente. |
| expa2 = | Altura. Puede ser cualquier número entre 0 y 255. Mientras mayor el número, menor será la altura. La tabla 10-1 define los números de altura para las diferentes notas musicales que abarcan dos octavas por arriba del do central y una octava por debajo del do central. |
| expa3 = | Distorsión. Puede ser cualquier número par entre 0 y 14. Se usa para crear efectos de sonido. El 10 se usa para crear un tono puro, mientras que el 12 da un zumbido, muy interesante. Un zumbido como el de las máquinas en una pista de carreras puede producirse usando dos comandos SOUND separados con un valor de distorsión (expa3) alternando entre 0 y 1. El valor 1 se usa para asegurar la salida al parlante usando el volumen especificado (vea expa4). El resto de los números se usa para otros efectos especiales, generación de ruidos y uso experimental. |
| expa4 = | Control de Volumen. Puede estar entre 1 y 15. Con 1 se crea un sonido apenas audible, mientras que 15 es fuerte. Un valor de 8 se considera normal. Al usar más de una sentencia SOUND , el volumen total no debería exceder de 32. De lo contrario se producirá un sonido desagradablemente recortado. |

Usando valores de nota de la Tabla 10-1, el ejemplo siguiente demuestra como escribir un programa que tocará la escala del do.

```

10 READ A
20 IF A=256 THEN END
30 SOUND 0,A,10,10
40 PRINT A
50 FOR T=0 TO 400:NEXT T
60 GOTO 10
70 END
80 DATA 29, 31, 35, 40, 45, 47, 53, 60, 64, 72, 81, 91, 96, 121
90 DATA 128, 144, 162, 182, 193, 217, 243, 256

```

Figura 10-1. Programa de la Escala Musical

Note que las sentencias DATA en la línea 80 terminan con un 256 que esta fuera del rango permitido. El 256 se emplea como un marcador de fin de datos.

TABLA 10.1. TABLA DE VALORES DE ALTURA PARA LAS NOTAS MUSICALES

NOTAS AGUDAS	Do	29
	Si	31
	La sostenido o Si bemol	33
	La	35
	Sol sostenido o La bemol	37
	Sol	40
	Fa sostenido o Sol bemol	42
	Fa	45
	Mi	47
	Re sostenido o Mi bemol	50
	Re	53
	Do sostenido o Re bemol	57
	Do	60
	Si	64
	La sostenido o Si bemol	68
	La	72
	Sol sostenido o La bemol	76
DO CENTRAL	Sol	81
	Fa sostenido o Sol bemol	85
	Fa	91
	Mi	96
	Re sostenido o Mi bemol	102
	Re	108
	Do sostenido o Re bemol	114
	Do	121
	Si	128
	La sostenido o Si bemol	136
NOTAS GRAVES	La	144
	Sol sostenido o La bemol	153
	Sol	162
	Fa sostenido o Sol bemol	173
	Fa	182
	Mi	193
	Re sostenido o Mi bemol	204
	Re	217
	Do sostenido o Re bemol	230
	Do	243

FUNCIONES DE LOS CONTROLES DE JUEGO

La figura 10-2 ilustra los tres controles usados con los computadores personales Atari. Los controladores pueden enchufarse directamente al computador Personal Atari o a dispositivos mecánicos externos, de manera que eventos externos puedan ser alimentados directamente al computador para propósitos de procesamiento y control.

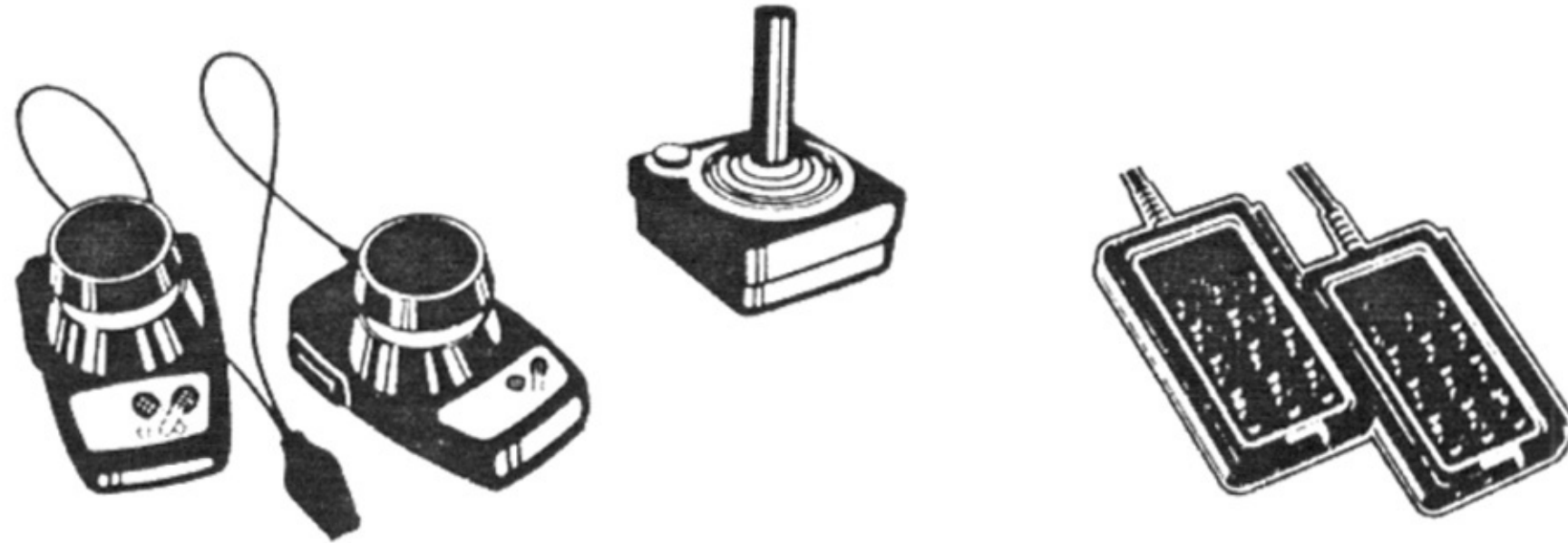


Figura 10-2. Controles de Juego

PADDLE

Formato: PADDLE (expa)

Ejemplo: PRINT PADDLE (3)

Esta función entrega el estado de un control en particular. Los controladores de paleta están numerados de 0 a 7 de izquierda a derecha (en el caso de los modelos 400 y 800 y de 0 a 3 en el caso de los modelos XL). Esta función puede usarse en conjunto con otras funciones o comandos para producir acciones posteriores tales como sonido, controles gráficos, etc. Por ejemplo, la sentencia `IF PADDLE (3) = 14 THEN PRINT "PALETA EN ACCION."` Note que la función PADDLE entrega un número entre 1 y 228, con números crecientes a medida que la perilla del control se gira en el sentido contrario a los punteros del reloj (hacia la izquierda).

PTRIG

Formato: PTRIG (expa)

Ejemplo: 100 IF PTRIG (4)=0 THEN PRINT"PROYECTILES DISPARADOS! "

La función PTRIG entrega un estado 0 si el botón disparador del control designado se encuentra presionado. De otra forma, retorna un valor de 1. La expa debe ser un número comprendido entre 0 y 7 (en el caso de los modelos 400 y 800) o 0 y 3 (para los modelos XL) ya que designa a un control.

STICK

Formato: STICK (expa)

Ejemplo: 100 PRINT STICK (3)

Esta función opera de la misma forma que el comando PADDLE, pero se usa con los controles de bastón. Los controles de bastón están numerados de 0 a 3 de izquierda a derecha (para el caso de los modelos 400 y 800. Los modelos XL tienen dos controles de bastón 0 y 1).

Control 1 = STICK (0)
 Control 2 = STICK (1)
 Control 3 = STICK (2) (solamente modelos 400 y 800)
 Control 4 = STICK (3) (solamente modelos 400 y 800)

La figura 10-3 indica los números que entregará esta función a medida que el bastón se mueve en cualquier dirección.

STRIG

Formato: STRIG (expa)

Ejemplo: 100 IF STRIG (3) =0 THEN PRINT "DISPAREN TORPEDO"

La función STRIG opera al igual que la función PTRIG. Puede usarse tanto con los bastones como con los controles de teclado.

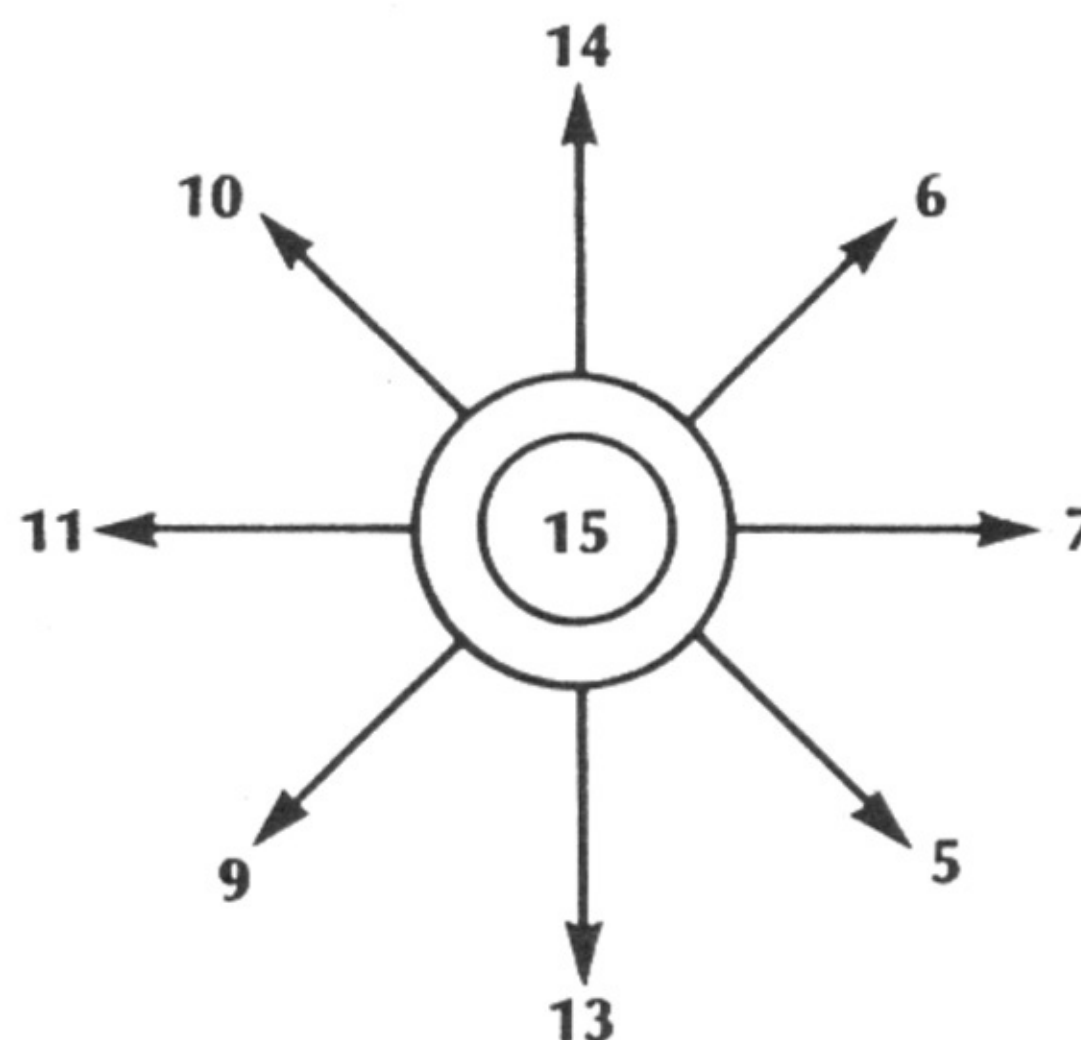


Figura 10-3. Movimiento del control de bastón

Esta sección incluye indicaciones para aumentar la eficiencia de programación, conservar memoria y combinar programas en lenguaje de máquina con programas BASIC Atari. Esta sección no incluye el juego de instrucciones del microprocesador 6502 ni da instrucciones respecto a la programación en lenguaje de máquina. Se recomienda sinceramente la adquisición del cartridge del editor de Assembler Atari y un estudio cuidadoso del manual del mismo Editor de Assembler Atari.

CONSERVACION DE MEMORIA

Estas indicaciones muestran algunos caminos para conservar memoria. Algunos de los métodos hacen que los programas sean menos legibles y más difíciles para modificar, pero hay veces que ello es necesario debido a limitaciones de memoria.

1. En muchos microcomputadores, el eliminar los espacios en blanco entre palabras y caracteres a medida que se digitan en el teclado puede ahorrar memoria. Esto no es el caso de los computadores Atari, que remueven los espacios extra. Las sentencias siempre se indican de la misma forma, independientemente de cuantos espacios se hayan usado al ingresar el programa. Los espacios deben usarse al igual que al escribir en una máquina convencional entre palabras claves sucesivas y entre palabras y nombres de variables. He aquí un ejemplo:

```
10 IF A = 5 THEN PRINT A
```

Note el espacio entre IF y A entre THEN y PRINT. En la mayoría de los casos, una sentencia se interpretará correctamente aun si no se dejan espacios, pero esto no siempre es cierto. Use los espacios en la forma convencional.

2. Cada número de línea nuevo representa el comienzo de lo que se llama una nueva línea lógica. Cada línea lógica toma 6 bytes de encabezamiento, ya sea que se use en toda su capacidad o no. Agregando sentencias BASIC adicionales usando doble punto (:) para separar cada par de sentencias sobre la misma línea toma solamente 3 bytes.

Si necesita ahorrar memoria, evite programas como el que sigue:

```
10 X=X+1
20 Y=Y+1
30 Z=X+Y
40 PRINT Z
50 GOTO 10
```

y consolide líneas así:

```
10 X=X+1:Y=Y+1:Z=X+Y:PRINT Z:GOTO 10
```

Esta consolidación ahorra 12 bytes.

3. Las variables y constantes también deberían manejarse en forma ahorrativa. Cada vez que se emplea una constante (4, 5, 16, 3. 14159, etc.) se usan 7 bytes. Definir una nueva variable requiere 8 bytes más el largo del nombre de la variable (en caracteres). Pero cada vez que vuelva a usarse una vez definida, solamente se requiere 1 byte, independientemente de su largo. Así, si una constante (tal como 3. 14159) se usa más de una vez o dos en un programa, debería definirse como variable y el nombre de la variable usarse a través del programa. Por ejemplo:


```
10 PI=3.14159
20 PRINT "EL AREA DE UN CIRCULO ES ";PI;" POR EL CUADRADO DEL RADIO."
```

4. Los strings literales requieren 2 bytes de encabezamiento y 1 byte para cada carácter del string (incluyendo todos los espacios).
5. Las variables literales o string requieren 9 bytes cada uno, más el largo del nombre de la variable (incluyendo espacios), más lo consumido por la sentencia DIM más el tamaño del string propiamente tal (1 byte por carácter incluyendo espacios) cuando se le define. Obviamente, el uso de las variables string es muy costoso en términos de RAM.
6. La definición de una matriz nueva requiere 15 bytes, más el largo del nombre de la variable matriz, más el espacio requerido por la sentencia DIM, más 6 veces el tamaño de la matriz (producto del número de líneas por el número de columnas); así una matriz de 25 líneas por 4 columnas requeriría 15 más aproximadamente 3 (para el nombre de la variable) más aproximadamente 10 (para la sentencia DIM) + 6 por 100 (el tamaño de la matriz), o alrededor de 630 bytes.
7. Cada carácter después de REM requiere un byte de memoria. Las notas son de gran ayuda para gente que trata de entender un programa, pero a veces es necesario sacrificarlas para ahorrar memoria.
8. Las subrutinas pueden ahorrar memoria, porque una subrutina y varias llamadas cortas a estas subrutinas toman menos memoria que duplicar esta parte del programa varias veces. Por otro lado, una subrutina que solamente se llama una vez, requiere bytes extra para las sentencias GOSUB y RETURN.
9. Los paréntesis toman un byte cada uno. Los paréntesis extra son una buena idea en algunos casos cuando hacen más comprensible para el programador una expresión; sin embargo, retirar los paréntesis innecesarios y confiar en la precedencia de operadores podrá ahorrar algunos bytes.

PROGRAMANDO EN LENGUAJE DE MAQUINA

El lenguaje de máquina está escrito en su totalidad en código binario. El Computador Personal Atari contiene un microprocesador 6502 y es posible llamar subrutinas de código de máquina 6502 desde BASIC usando la función USR. En caso necesario puede incluso ingresarse rutinas ensambladas manualmente en los programas.

Antes de retornar a BASIC la rutina de lenguaje Assembler debe ejecutar una instrucción de paso al acumulador (PLA) para retirar el número (N) de argumentos de entrada desde el stack. Si este número no es igual a 0, entonces todos los argumentos de entrada deben ser desplazados del stack usando la misma instrucción PLA en forma repetida. (Vea la figura 6-1).

La subrutina debería finalizar colocando el byte bajo de su resultado en la ubicación 212 de la Memoria, y el byte alto en la ubicación 213 (ambas decimales) y enseguida volver a BASIC utilizando una instrucción RTS (Retorno desde Subrutina). El intérprete BASIC convertirá el número binario de 2 bytes almacenado en las ubicaciones 212 y 213 en un número entero comprendido entre 0 y 65535 en formato de coma flotante para obtener el valor entregado por la función USR.

La función ADR puede usarse para pasar datos que están almacenados en agrupamientos o strings a una subrutina de lenguaje de máquina. Use la función ADR para obtener la dirección de este agrupamiento o string, y enseguida use esta dirección como uno de los argumentos de entrada de USR.

El siguiente programa, un cargador hexadecimal, entrega los medios de ingresar códigos hexadecimales, convertir cada número hexadecimal a un decimal y almacenar el número decimal en un agrupamiento.

El agrupamiento enseguida se ejecuta como una subrutina en lenguaje Assembler. (Se usa un agrupamiento para asignar espacio en la memoria para la subrutina).

1. Para usar este programa, primero ingréselo. Una vez ingresado, grábalo en disco o cassette para uso futuro.

```

10 GRAPHICS 0:PRINT "CARGADOR HEXADECIMAL":PRINT
20 REM ALMACENA EQUIVALENTES DECIMALES EN AGRUPAMIENTO A;LOS ENTREGA COMO 'SEN-
21 REM TENCIAS DATA' IMPRESAS EN LA LINEA NO. 1500
30 REM EL USUARIO A CONTINUACION UBICA EL CURSOR SOBRE LA LA LINEA DE SALIDAIM-
31 REM PRESA,PRESIONA "RETURN" E INGRESA EL RESTO DEL PROGRAMA BASIC, INCLUYEN-
32 REM DO LA SENTENCIAUSR.
40 DIM A(50),HEX$(5)
50 REM INGRESO,CONVERSION Y ALMACENAMIENTO DE DATOS.
60 N=0:PRINT "INGRESE UN CODIGO HEXADECIMAL;SI YA INGRESO EL ULTIMO,DIGITE 'FIN'
  .";
70 INPUT HEX$
80 IF HEX$="FIN" THEN N=999:GOTO 130
90 FOR I=1 TO LEN(HEX$)
100 IF HEX$(I,I)<="9" THEN N=N*16+VAL(HEX$(I,I)):GOTO 120
110 N=N*16+ASC(HEX$(I,I))-ASC("A")+10
120 NEXT I
130 PRINT N:C=C+1
140 A(C)=N
150 IF N<>999 THEN GOTO 60
190 REM IMPRESION DE SENTENCIA DATA EN LINEA ,1500
200 GRAPHICS 0:PRINT "1500 DATA ";
210 C=0
220 C=C+1
230 IF A(C)=999 THEN PRINT "999":GOTO 270
240 PRINT A(C);", ";
250 A(C)=0
260 GOTO 220
270 STOP
300 PRINT "INDIQUE EL NUMERO EXACTO DE BYTES HEXADECIMALES EN LA LINEA 1000.":LI
ST 1000:STOP :REM LINEA TRAP

```

```

999 REM ** MODULO DE EJECUCION **
1000 CLR :BYTES=0
1010 TRAP 300:DIM E$(1),E(INT(BYTES/5)+1)
1030 FOR I=1 TO BYTES
1040 READ A:IF A>255 THEN GOTO 1060
1050 POKE ADR(E$)+I,A
1060 NEXT I
1070 REM SIGUE AHORA EL PROGRAMA BASIC DEL USUARIO.

```

Figura 11-1 Programa de Ingreso del Cargador Hexadecimal

2. Ahora agregue la parte de su programa en lenguaje BASIC comenzando por la línea 1080 incluyendo la función **USR** que llamará la subrutina de lenguaje de máquina (Vea el ejemplo más abajo).
3. Cuente el número total de códigos hexadecimales que deben ingresarse e ingrese este número en la línea 1000 cuando así sea requerido. Si ya hay allí otro número simplemente reemplácelo.
4. Ejecute el programa e ingrese los códigos hexadecimales a nivel de subrutina de lenguaje de máquina, presionando **RETURN** después de cada ingreso. Después del último ingreso digite **FIN** y presione **RETURN**.
5. Ahora aparecerá la línea de DATA (1500) en la pantalla. Ella no será ingresada al programa hasta que el cursor sea movido hasta la línea de DATA y se presione **RETURN**.
6. Añada una línea de programa del tipo 5 GOTO 1000 para evitar el cargador hexadecimal o elimine el cargador hexadecimal hasta la línea 260. Ahora grabe el programa ya terminado usando **CSAVE** o **SAVE**. Es importante hacer esto antes de ejecutar la parte del programa que tenga la llamada **USR**. Un error en la rutina de lenguaje de máquina puede producir un bloqueo del sistema. Si el sistema de bloqueo, presione **RESET**. Si el sistema no responde apague el equipo y vuelva a encenderlo. Recargue el programa y corrija.

Nota: Este método sólo opera con rutinas de lenguaje de máquina que sean reubicables.

Los dos programas de ejemplos que siguen, pueden cada uno ingresarse en el cargador hexadecimal. El primer programa imprime **NADA SE MUEVE** mientras el programa de máquina hace cambiar los colores. El segundo programa muestra un diseño de gráficos BASIC y enseguida cambia los colores.

```

1080 GRAPHICS 1+16
1090 FOR I=1 TO 6
1100 PRINT #6;"nada se mueve!"
1110 PRINT #6;"NADA SE MUEVE!"
1120 PRINT #6;"nada se mueve!":REM VIDEO INVERSO
1130 PRINT #6;"NADA SE MUEVE!":REM VIDEO INVERSO
1140 NEXT I
1150 Q=USR(ADR(E$)+1)
1160 FOR I=1 TO 25:NEXT I:GOTO 1150

```

Una vez ingresado el programa compruebe que la línea 1000 sea:

```
1000 CLR:BYTES = 21
```

Digite **RUN RETURN**.

Ahora ingrese los códigos hexadecimales tal como se indica, columna por columna.

68	2
A2	E8
0	E0
AC	3
C4	90
2	F5
BD	8C
C5	C7
2	2
9D	60
C4	

BYTES = 21

Una vez hecho, digite FIN y presione **RETURN**. Ahora ubique el cursor tras el último ingreso (999) en la línea de DATA y presione **RETURN**.

Ahora ejecute el programa digitando GOTO 1000 y presionando **RETURN**, o si agregó la línea 5 digite RUN **RETURN**. Presione **BREAK** para detener el programa y eliminar la línea 5.

El segundo programa, que es el que sigue, debe ingresarse en lugar del programa NADA SE MUEVE. Asegúrese de comprobar la cuenta de BYTES = ____ _ _ _ en la línea 1000. Siga los pasos del 2 al 6.

```

1080 GRAPHICS 7+16
1090 SETCOLOR 0,9,4
1100 SETCOLOR 1,9,8
1110 SETCOLOR 2,9,4
1120 CR=1
1130 FOR X=0 TO 159
1140 COLOR INT(CR)
1150 PLOT 80,0
1160 DRAWTO X,95
1170 CR=CR+0.125
1180 IF CR=4 THEN CR=1
1190 NEXT X
1200 X=USR(ADR(E$)+1)
1210 FOR I=1 TO 15:NEXT I
1220 GOTO 1200

```

Digite RUN **RETURN**.

Ingresa los códigos hexadecimales para este programa columna por columna.

68	2
A2	E8
0	E0
AC	2
C4	90
2	F5
BD	8C
C5	C6
2	2
9D	60
C4	

BYTES = 21

Una vez hecho, escriba FIN y presione **RETURN**. Ahora ubique el cursor detrás del último ingreso de datos (999) y presione **RETURN**.

Ejecute ahora el programa digitando GOTO 1000 y presionando **RETURN**, o agregue la línea 5 GOTO 1000 y digite RUN **RETURN**. Presione **BREAK** para detener el programa y eliminar la línea 5.

La figura 11-2 ilustra una subrutina Assembler empleada para rotar colores y que puede ser muy útil. Se incluye aquí para información del usuario.

Subrutina Assembler para rotar colores

Dir.	Cód. O.	No. Lí.	Etiqueta	Mnem.	Datos	Comentarios
		0100				Rutina para rotar
		0110				datos de COLOR de
		0120				un registro a
		0130				otro. Se rotan 4
						colores.
		0140				Dir. sist. operativo
02C4		0150				COLOR 0 = \$02C4
02C5		0160				COLOR 1 = \$02C5
02C6		0170				COLOR 2 = \$02C6
02C7		0175				COLOR 3 = \$02C7
		0180				
		0190		*=	\$6000	Dir. comienzo
						prog. máquina. *
6000	68	0200		PLA		Tirar stack
6001	A200	0210		LDX	#0	Poner en 0 reg. X
6003	ACC402	0220		LDY	COLOR0	Guardar COLOR 0
6006	BDC502	0230	BUCLE	LDA	COLOR1, X	
6009	9DC402	0240		STA	COLOR0, X	
600C	E8	0250		INX		Incr. registro X
600D	E002	0260		CPX	#3	Comparar reg. X
						con 2
600F	90F5	0270		BCC	BUCLE	Bucle si reg. X < 2
6011	8CC602	0280		STY	COLOR3	Guardar COLOR 0
						en COLOR3
6014	60	0290		RTS		Regreso desde nivel
						de rut. máq.
El Assembler		Esta parte es información fuente ingresada por el programador				
imprime esto		con el cartridge de assembler ATARI				

indica datos (fuente)

* la rutina es reubicable

\$ indica número hexadecimal

Figura 11-2. Subrutina assembler para rotar colores

A P E N D I C E A

 INDICE ALFABETICO
 DE PALABRAS RESERVADAS DEL BASIC

Nota: El punto es obligatorio en todas las palabras claves abreviadas.

PALABRA RESERV.	ABREV.	BREVE RESUMEN SENTENCIA BASIC
ABS		La función entrega el valor absoluto de la variable o expresión.
ADR		La función entrega la dirección de memoria de un string.
AND		Operador lógico. La expresión es verdadera solamente si las dos sub-expresiones unidas por AND son verdaderas.
ASC		Función string que entrega el valor numérico de un carácter string.
ATN		La función da el arcotangente del número o de la expresión en radianes o grados.
BYE	B.	Egreso de BASIC a "MEMO PAD" o "SELF TEST" según modelo.
CLOAD	CLOA.	Carga datos desde la grabadora de programas a la RAM.
CHR\$		Función string, entrega el carácter equivalente en código ATASCII de un byte comprendido entre 0 y 255.
CLOG		Función que entrega el logaritmo en base 10 de una expresión.
CLOSE	CL.	Sentencia de entrada/salida (I/O) usada para cerrar un archivo al finalizar una operación de entrada/salida.
CLR		Lo opuesto de DIM: elimina el dimensionamiento de todos los strings y matrices
COLOR	C.	Escoge el registro de color a usar en el procesamiento gráfico.
COM		Lo mismo que DIM.
CONT	CON.	Continuar. Reinicia la ejecución de un programa a partir de la línea siguiente al uso de BREAK o encuentro de STOP.
COS		Función que da el coseno trigonométrico de la variable o expresión (en radianes o grados).

CSAVE		Entrega datos desde la RAM a la grabadora de programas para su almacenamiento en cinta.
DATA	D.	Parte de la combinación READ/DATA. Se emplea para identificar los sucesivos items (debidamente separados por comas), como items de DATA individuales.
DEG	DE.	La sentencia DEG indica al computador que debe realizar el cálculo de funciones trigonométricas en grados y no radianes (lo asumido son radianes).
DIM	DI.	Reserva la cantidad especificada de memoria para una matriz, un agrupamiento, o un string. Toda variable de string, agrupamiento y matriz debe ser dimensionado con una sentencia DIM.
DOS	DO.	Palabra reservada para usuarios de disco. Hace desplegar el menú del DOS (vea el manual del DOS).
DRAWTO	DR.	Traza una línea recta entre un punto marcado PLOT y el punto especificado.
END		Detiene la ejecución del programa; cierra los archivos, apaga los sonidos. El programa puede reiniciarse usando CONT. (Nota: END puede usarse más de una vez en un programa).
ENTER	E.	Comando de entrada/salida (I/O) que se usa para almacenar datos o programas en forma no codificada (forma fuente).
EXP		Función que entrega e (2. 7182818) elevado a la potencia especificada.
FOR	F.	Se usa con NEXT para establecer bucles FOR/NEXT. Introduce el rango dentro del cual operará la variable de bucle durante la ejecución de éste.
FRE		La función entrega la cantidad de memoria de usuario RAM disponible (en bytes).
GET	GE.	Se usa principalmente en operaciones con disco para ingresar un solo byte de datos.
GOSUB	GOS.	Ramificación a la subrutina que comienza en el número de línea especificado.
GOTO	G.	Ramificación, incondicional a un número de línea especificado.
GRAPHICS	GR.	Especifica cual de los quince modos gráficos se va a usar. GR. 0 puede usarse para limpiar la pantalla.
IF		Se usa para lograr una ramificación condicional o para ejecutar otra sentencia en la misma línea (solamente si la primera expresión es verdadera).

INPUT	I.	Hace al computador consultar por un ingreso desde el teclado. La ejecución continúa sólo una vez que se haya presionado la tecla RETURN después de ingresar los datos.
INT		La función entrega el número entero menor más próximo al valor especificado. El redondeo siempre es hacia abajo, aun cuando el número sea negativo.
LEN		Función string que da el largo en bytes del string especificado (1 byte contiene 1 carácter).
LET	LE.	Asigna un valor a la variable especificada. LET es opcional en BASIC Atari y simplemente puede omitirse.
LIST	L.	Despliega o entrega en alguna otra forma un listado de programa.
LOAD	LO.	Ingresa desde disco, etc. al computador.
LOCATE	LOC.	Gráficos: almacena en una variable especificada el valor que controla un punto gráfico específico.
LOG		La función entrega el logaritmo natural del número.
LPRINT	LP.	Ordena al impresor de línea la impresión del mensaje especificado.
NEW	N.	Borra todo lo contenido en la memoria de usuario.
NEXT	N.	Hace que el bucle FOR/NEXT termine o siga, dependiendo de las variables o expresiones en particular. Todos los bucles se ejecutan al menos una vez.
NOT		Entrega un "1" solamente si la expresión no es verdadera. Si es verdadera entrega un "0".
NOTE	NO.	Vea el manual del Sistema Operativo de Disco... se usa solamente en operaciones de disco.
ON		Se usa con GOTO o GOSUB para propósitos de ramificación. Son posibles ramificaciones múltiples a diferentes números de líneas dependiendo del valor de la variable o expresión ON.
OPEN	O.	Abre el archivo especificado para operaciones de ingreso o salida.
OR		Operador lógico que se usa entre dos expresiones. Si cualquiera de éstas es verdadera, su valor es "1". Resulta un 0 solamente si ambas expresiones son falsas.
PADDLE		La función entrega la posición del controlador de paleta.

PEEK		La función entrega en forma decimal el contenido de la ubicación de memoria especificada (RAM o ROM).
PLOT	PL.	Marca un punto en la pantalla en la posición X, Y, especificada.
POINT	P.	Solamente se usa en operaciones de disco.
POKE	POK.	Introduce el byte especificado en la ubicación de memoria especificada. Solamente puede usarse en RAM. No trate de hacer POKE en ROM u obtendrá un error.
POP		Elimina la variable de bucle del stack GOSUB. Se usa cuando se abandona el bucle por una vía diferente de la normal.
POSITION	POS.	Ubica el cursor en la posición de pantalla especificada.
PRINT	PR. o ?	Comando de entrada/salida (I/O) que produce una salida del computador al dispositivo de salida especificado
PTRIG		La función entrega el estado del botón de disparo de un controlador de paleta.
PUT	PU.	Provoca la entrega de un byte de dato del computador al dispositivo especificado
RAD		Especifica que la información numérica a usar en funciones trigonométricas se encuentra en radianes y no en grados. El valor asumido es RAD (vea DEG).
READ	REA.	Lee el ítem siguiente de la lista de DATA y lo asigna a la variable especificada.
REM	R. o .ESPACIO	Notas. Esta sentencia no hace nada, pero permite imprimir comentarios dentro del listado de un programa para referencia futura del programador. Las sentencias que se encuentran en una línea que comienza con REM no son ejecutadas.
RESTORE	RES.	Permite que los datos de DATA puedan ser leídos más de una vez.
RETURN	RET.	Retorno de subrutina a la sentencia inmediatamente posterior a la que aparecía GOSUB.
RND		La función entrega un número aleatorio comprendido entre 0 y 1. Nunca alcanza a ser 1.
RUN	RU.	Ejecuta el programa. Repone las variables a 0, elimina dimensionamiento de agrupamientos y strings.
SAVE	S.	Sentencia de entrada/salida (I/O) que graba datos o programas en disco bajo la esparchivo suministrada con SAVE.

SETCOLOR SE.	Almacena los datos de matriz y luminancia de color en un registro de color especificado.
SGN	La función entrega +1 si el valor es positivo, 0 si es nulo y -1 si es negativo.
SIN	La función entrega el seno trigonométrico del valor dado en grados DEG o radianes RAD.
SOUND SO.	Controla el registro, altura, distorsión y volumen de un tono o nota.
SQR	La función entrega la raíz cuadrada del valor especificado.
STATUS ST.	Llama la rutina de estado o STATUS del dispositivo especificado.
STEP	Se usa con FOR/NEXT. Determina la magnitud del salto entre cada par de valores de la variable del bucle.
STICK	La función entrega la posición del controlador de bastón.
STRIG	La función da un 1 si el botón disparador del bastón no está presionado, 0 si está presionado.
STOP STO.	Detiene la ejecución, sin embargo no cierra archivos ni apaga sonidos.
STR\$	La función entrega un string de carácter igual al valor numérico dado. Por ejemplo: STR\$(65) da 65 como string.
THEN	Se usa con IF: si la expresión es verdadera entonces se ejecutan las sentencias THEN. Si la expresión es falsa, el control pasa a la línea siguiente.
TO	Se usa con FOR como en "FOR X = 1 TO 10". Separa las expresiones de rango del bucle.
TRAP T.	Toma control del programa en caso de producirse un error INPUT y dirige la ejecución hacia el número de línea especificado.
USR	La función entrega el resultado de una subrutina en lenguaje de máquina.
VAL	La función entrega el valor numérico equivalente a un string.
XIO X.	Sentencia de entrada/salida (I/O) general, que se usa con operaciones de disco (vea el manual del DOS) y en trabajo gráfico (relleno).

MENSAJES DE ERROR

CODIGO DE ERROR No.	MENSAJE DEL CODIGO DE ERROR
2	Memoria insuficiente para almacenar la sentencia o el nuevo nombre de variable o DIM de una nueva variable string.
3	Error de Valor: Un valor que se esperaba fuera un entero positivo resultó negativo, o un valor que se esperaba estuviera dentro de un determinado rango no lo está.
4	Demasiadas Variables: Se permite un máximo de 128 diferentes nombres de variables (Vea límite de nombres de variables).
5	Error de Largo de String: Intento de almacenar más allá del largo dimensionado de string.
6	Error de Falta de Datos: Una sentencia READ que requiere más ítems de datos que los suministrados por la sentencia DATA.
7	Número mayor que 32767. El valor no es entero positivo o es mayor que 32767.
8	Error de Sentencia de Entrada: Intento de hacer un INPUT de un valor no numérico en una variable numérica
9	Error DIM de Agrupamiento o string: Tamaño de DIM mayor que 32767 o la referencia de un agrupamiento o matriz está fuera del rango de su tamaño dimensionado o el agrupamiento/matriz o string ya ha sido dimensionado, o se ha hecho referencia a un agrupamiento o string no dimensionado.
10	Stack de Parámetros Excedido: Hay demasiados GOSUB o una expresión demasiado larga.
11	Rango de Coma Flotante excedido: Intento de dividir por 0 o hacer referencia a un número mayor que 1×10^{98} o menor que 1×10^{-99} .
12	Línea Inexistente: Un GOSUB, GOTO o THEN que hace referencia a un número de línea no existente.
13	Falta instrucción FOR: Hay un NEXT sin un FOR previo o sentencias FOR/NEXT anidadas que no se complementan como corresponde. (El error se reporta en la sentencia NEXT, no en FOR).
14	Error de línea demasiado larga: La sentencia es demasiado compleja o excesivamente larga para ser manejada por BASIC.
15	Línea GOSUB o FOR eliminada: Hay una sentencia NEXT o RETURN y se ha eliminado el FOR o GOSUB correspondiente desde el último RUN.
16	Error de RETURN: Hay un RETURN sin su GOSUB correspondiente.
17	Línea incomprensible: Se intentó la ejecución de "basura" (bits malos en la RAM). Este código de error puede indicar un problema de circuitos, pero también puede ser el resultado de mal uso de POKE. Intente digitando NEW o

apagando el computador, reingresando a continuación el programa sin ningún comando POKE.

- 18 Carácter String Inválido: El string no se inicia con un carácter válido, o el string en una sentencia VAL no corresponde a un string numérico.

Nota: Los siguientes son errores de entrada/salida (INPUT/OUTPUT) que pueden resultar de la interacción con dispositivos periféricos como unidades de disco, impresores, etc. Se da mayor información en los manuales correspondientes.

- 19 Programa en Carga LOAD demasiado Largo: Queda poca memoria para completar el LOAD.
- 20 Número de Dispositivo mayor que 7 o igual a 0.
- 21 Error de Archivo LOAD: Intento de hacer la carga LOAD de un archivo que no está en formato LOAD codificado.
- 128 Detención por BREAK: El usuario presionó la tecla BREAK durante una operación de entrada/salida.
- 129 Bloque de Control IOCB ya abierto. (IOCB se refiere al Bloque de control entrada/salida). El número del dispositivo es el mismo del bloque IOCB.
- 130 Dispositivo Inexistente.
- 131 Bloque de Control IOCB abierto para salida solamente. Hubo un comando READ en un dispositivo de escritura solamente (Impresor).
- 132 Comando Inválido: El comando es inválido para este dispositivo.
- 133 El Dispositivo o Archivo no está abierto: No se ha especificado OPEN para el dispositivo.
- 134 Número IOCB erróneo: Número de dispositivo ilegal.
- 135 Error de IOCB de Lectura solamente: Comando de escritura para un dispositivo destinado a lectura exclusivamente.
- 136 EOF: Fin de Archivo. (Nota: Este mensaje puede ocurrir al usar archivos en cassette).
- 137 Registro Trunco: Intento de leer un registro de más de 256 caracteres.
- 138 Tiempo del Dispositivo Excedido: El dispositivo no responde.
- 139 El Dispositivo no responde NAK: Basura en la puerta serial o unidad de disco defectuosa.
- 140 Error del Campo de Datos del Bus Serial.
- 141 Cursor fuera de Rango para este modo particular.
- 142 Campo de Datos del Bus Serial excedido.
- 143 Error en Comprobación del Campo de Datos del Bus Serial.
- 144 Error del Dispositivo (byte "cumplido" inválido): Intento de escribir en un disquette con protección.

145	Error de Comprobación de Lectura después de grabación (Administración del disco o Administrador del Modo de pantalla con problemas).
146	Función no implementada en el Administrador.
147	RAM insuficiente para el modo gráfico seleccionado.
160	Error del número de la Unidad de Disco.
161	Demasiados Archivos abiertos (no hay disponibilidad de memoria de sector).
162	Disco Lleno (no hay sectores libres).
163	Error de entrada/salida de datos de sistema irrecuperable.
164	Descompaginación de número de Archivo: Los punteros del disco están desordenados.
165	Error de nombre de Archivo.
166	Error en el largo de los datos POINT.
167	Archivo protegido.
168	Comando Inválido (Código operacional especial).
169	Directorio lleno (64 archivos).
170	Archivo no ubicable.
171	Comando POINT inválido.
172	Apéndice ilegal: intento de combinar archivos DOS I y DOS II.
173	Sector defectuoso al formatear diskette.

APÉNDICE C






JUEGO DE CARACTERES ATASCII

CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER
0	0	á	13	D	ú	26	1A	² L ⁴ A
1	1	ù	14	E	ó	27	1B	ESC E
2	2	ñ	15	F	ö	28	1C	↑
3	3	É	16	10	ü	29	1D	↓
4	4	ξ	17	11	â	30	1E	←
5	5	ô	18	12	û	31	1F	→
6	6	ò	19	13	ÿ	32	20	Espacio
7	7	ò	20	14	é	33	21	!
8	8	ò	21	15	è	34	22	"
9	9	ï	22	16	ñ	35	23	#
10	A	ü	23	17	ë	36	24	\$
11	B	ä	24	18	à	37	25	%
12	C	ò	25	19	à	38	26	&

CHBAS 22h
CHBAS 20x



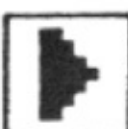
salto

CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER
39	27	,	55	37	7	71	47	G
40	28	(56	38	8	72	48	H
41	29)	57	39	9	73	49	I
42	2A	*	58	3A	:	74	4A	J
43	2B	+	59	3B	;	75	4B	K
44	2C	,	60	3C	<	76	4C	L
45	2D	-	61	3D	=	77	4D	M
46	2E	.	62	3E	>	78	4E	N
47	2F	/	63	3F	?	79	4F	O
48	30	0	64	40	@	80	50	P
49	31	1	65	41	A	81	51	Q
50	32	2	66	42	B	82	52	R
51	33	3	67	43	C	83	53	S
52	34	4	68	44	D	84	54	T
53	35	5	69	45	E	85	55	U
54	36	6	70	46	F	86	56	V

CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER
87	57	W	103	67	g	119	77	w
88	58	X	104	68	h	120	78	x
89	59	Y	105	69	i	121	79	y
90	5A	Z	106	6A	j	122	7A	z
91	5B	[107	6B	k	123	7B	
92	5C	\	108	6C	l	124	7C	
93	5D]	109	6D	m	125	7D	 <i>salta</i>
94	5E	^	110	6E	n	126	7E	 <i>retro</i>
95	5F	_	111	6F	o	127	7F	 <i>inserte</i>
96	60		112	70	p	128	80	
97	61	a	113	71	q	129	81	
98	62	b	114	72	r	130	82	
99	63	c	115	73	s	131	83	
100	64	d	116	74	t	132	84	
101	65	e	117	75	u	133	85	
102	66	f	118	76	v	134	86	

CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER
135	87		151	97		167	A7	
136	88		152	98		168	A8	
137	89		153	99		169	A9	
138	8A		154	9A		170	AA	
139	8B		155	9B	(EOL) RETURN	171	AB	
140	8C		156	9C	⬆	172	AC	
141	8D		157	9D	⬇	173	AD	
142	8E		158	9E	⬅	174	AE	
143	8F		159	9F	➡	175	AF	
144	90		160	A0		176	B0	
145	91		161	A1		177	B1	
146	92		162	A2		178	B2	
147	93		163	A3		179	B3	
148	94		164	A4		180	B4	
149	95		165	A5		181	B5	
150	96		166	A6		182	B6	

CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER
183	B7		199	C7		215	D7	
184	B8		200	C8		216	D8	
185	B9		201	C9		217	D9	
186	BA		202	CA		218	DA	
187	BB		203	CB		219	DB	
188	BC		204	CC		220	DC	
189	BD		205	CD		221	DD	
190	BE		206	CE		222	DE	
191	BF		207	CF		223	DF	
192	C0		208	D0		224	E0	
193	C1		209	D1		225	E1	
194	C2		210	D2		226	E2	
195	C3		211	D3		227	E3	
196	C4		212	D4		228	E4	
197	C5		213	D5		229	E5	
198	C6		214	D6		230	E6	

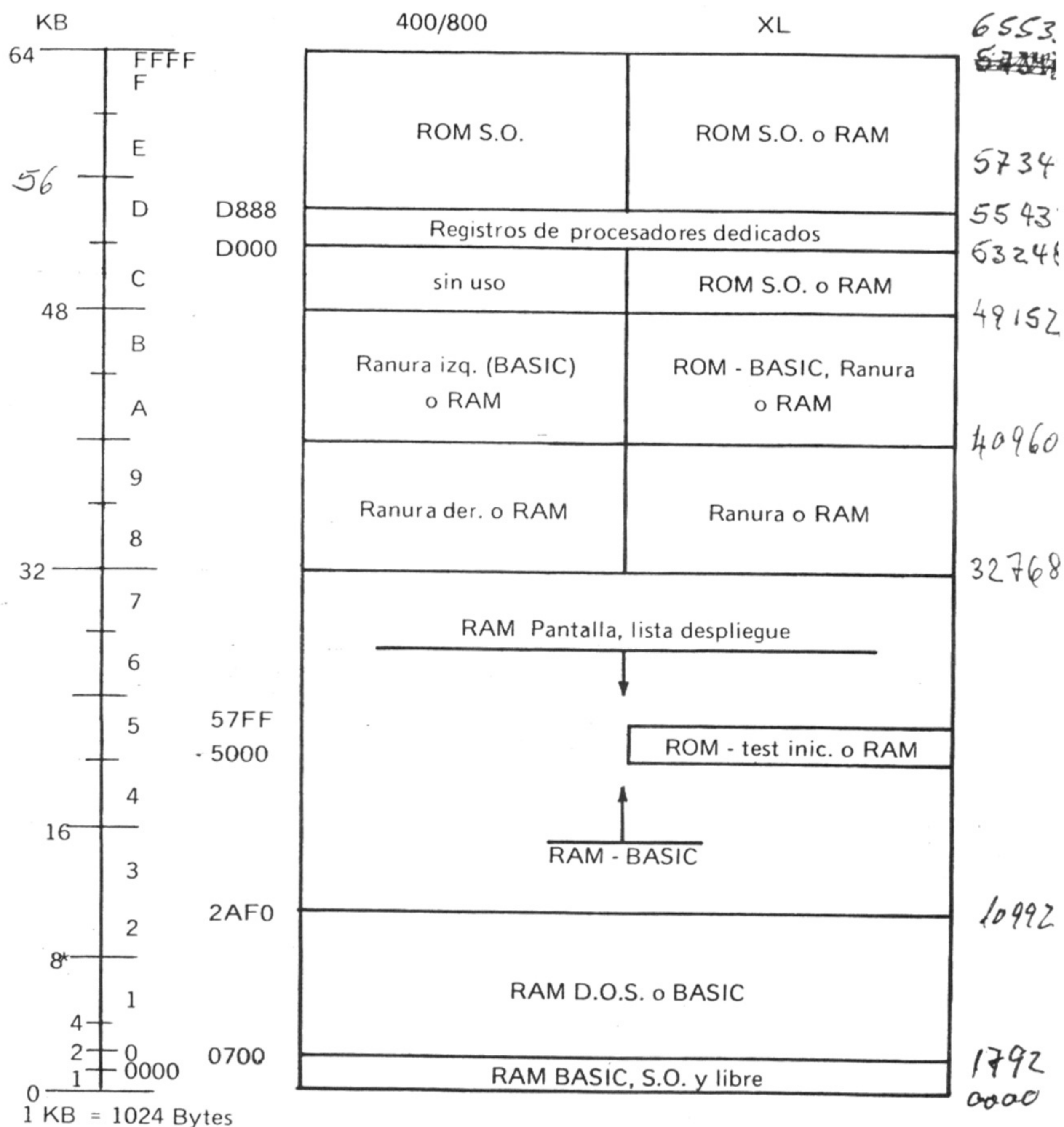
CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER
231	E7		240	F0		249	F9	
232	E8		241	F1		250	FA	
233	E9		242	F2		251	FB	
234	EA		243	F3		252	FC	
235	EB		244	F4		253	FD	 (Zumbador) ✓
236	EC		245	F5		254	FE	 (Eliminar caracter) 7
237	ED		246	F6		255	FF	 (Insertar caracter) 7
238	EE		247	F7				
239	EF		248	F8				

Vea en el Apéndice H un programa que hace la conversión decimal/hexadecimal.

Notas:

1. ATASCII significa "ATARI-ASCII". Letras y números tienen los mismos valores del ASCII, pero algunos caracteres especiales son diferentes.
2. Excepto los casos indicados, los caracteres del 128 al 255 son de color inverso a los del 1 al 127.
3. Sume 32 al código de una mayúscula, para obtener el código de la minúscula correspondiente.
4. Para obtener el código ATASCII, pida al computador (en modo directo), que imprima: PRINT ASC (" "). Llene el espacio con la letra, el carácter o el número en cuestión. Las comillas son obligatorias.
5. En las primeras páginas (C1-C3) de este apéndice, los caracteres gráficos normales se han representado como símbolos blancos sobre fondo negro; en las últimas páginas (C4-C6), los símbolos del teclado de video inverso como símbolos negros sobre fondo blanco.

MAPAS DE MEMORIA
400 / 800 ; XL



Como las direcciones correspondientes a los extremos de RAM, S.O., BASIC y D.O.S. dependen de la cantidad de RAM, ellas se indican por medio de punteros. Las direcciones de estos punteros se dan en el APENDICE I.

Funciones derivadas	Funciones derivadas en términos de funciones BASIC ATARI
Secante	$\text{SEC}(X) = 1/\text{COS}(X)$
Cosecante	$\text{CSC}(X) = 1/\text{SIN}(X)$
Arcoseno	$\text{ARCSIN}(X) = \text{ATN}(X/\text{SQR}(-X*X+1))$
Arcocoseno	$\text{ARCCOS}(X) = -\text{ATN}(X/\text{SQR}(-X*X+1)) + \text{CONSTANTE}$
Arcosecante	$\text{ARSEC}(X) = \text{ATN}(\text{SQR}(X*X-1)) + (\text{SGN}(X-1) * \text{CONSTANTE})$
Arcocosecante	$\text{ARCCSC}(X) = \text{ATN}(1/\text{SQR}(X*X-1)) + (\text{SGN}(X-1) * \text{CONSTANTE})$
Arcocotangente	$\text{ARCCOT}(X) = \text{ATN}(X) + \text{CONSTANTE}$
Seno hiperbólico	$\text{SINH}(X) = (\text{EXP}(X) - \text{EXP}(-X)) / 2$
Coseno hiperbólico	$\text{COSH}(X) = (\text{EXP}(X) + \text{EXP}(-X)) / 2$
Tangente hiperbólica	$\text{TANH}(X) = -\text{EXP}(-X) / (\text{EXP}(X) + \text{EXP}(-X)) * 2 + 1$
Secante hiperbólica	$\text{SECH}(X) = 2 / (\text{EXP}(X) + \text{EXP}(-X))$
Cosecante hiperbólica	$\text{CSCH}(X) = 2 / (\text{EXP}(X) - \text{EXP}(-X))$
Cotangente hiperbólica	$\text{COTH}(X) = \text{EXP}(-X) / (\text{EXP}(X) - \text{EXP}(-X)) * 2 + 1$
Arco seno hiperbólico	$\text{ARCSINH}(X) = \text{LOG}(X + \text{SQR}(X*X+1))$
Arco coseno hiperbólico	$\text{ARCCOSH}(X) = \text{LOG}(X + \text{SQR}(X*X-1))$
Arcotang. Hiperbólico	$\text{ARCTANH}(X) = \text{LOG}((1+X) / (1-X)) / 2$
Arco Sec. hiperbólica	$\text{ARCSECH}(X) = \text{LOG}((\text{SQR}(-X*X+1)+1)/X)$
Arco cosec. hiperbólica	$\text{ARCCSCH}(X) = \text{LOG}((\text{SGN}(X) * \text{SQR}(X*X+1) + 1)/X)$
Arco Cotang. hiperbólica	$\text{ARCCOTH}(X) = \text{LOG}((X+1) / (X-1)) / 2$

Notas:

1. En el modo RAD (asumido), **CONSTANTE=1,57079633**
En el modo DEG, **CONSTANTE=90**

2. En este cuadro X (entre paréntesis) representa la variable o expresión cuya función derivada se está evaluando. Obviamente se permite cualquier nombre de variable, mientras represente el valor o la expresión a evaluar.

APÉNDICE F

VERSIONES IMPRESAS DE LOS CARACTERES DE CONTROL

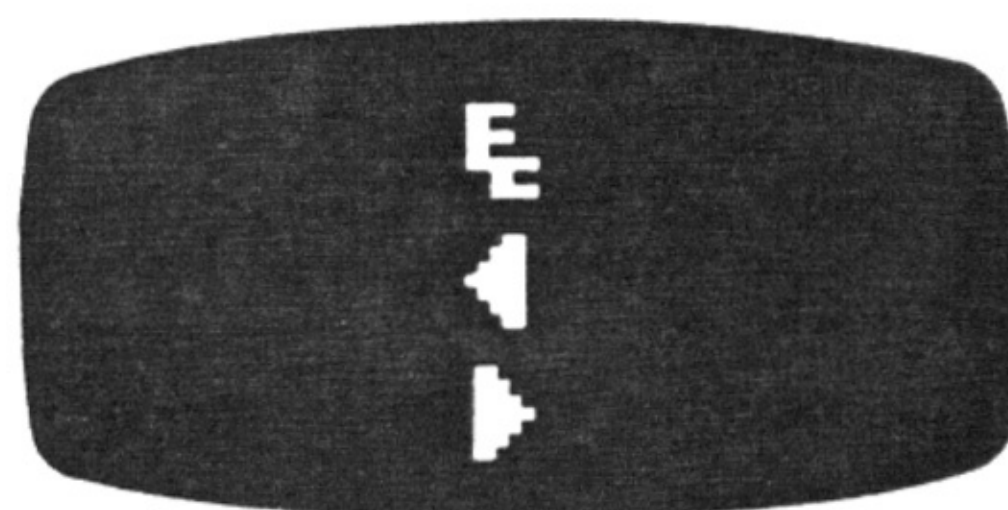
Los caracteres de control de cursor y de pantalla pueden ponerse en un string o en un programa, lo mismo que pueden usarse instrucciones de modo directo, presionando la tecla **ESC** antes de ingresar el carácter al teclado. Con esto se presentan los siguientes símbolos en la pantalla (Vea la Sección 1 tecla **ESC**).

VEA ESTO

PRESIONE



PRESIONE

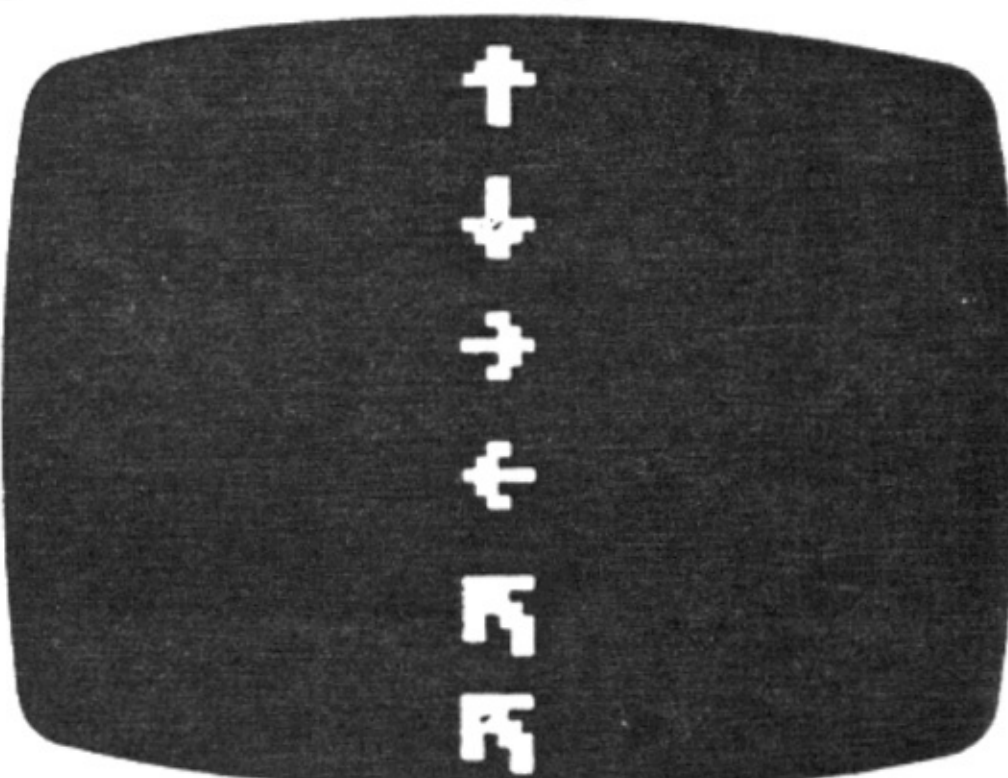
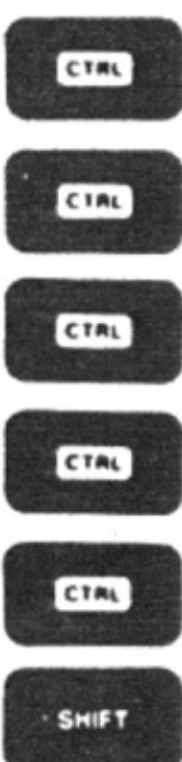


PRESIONE

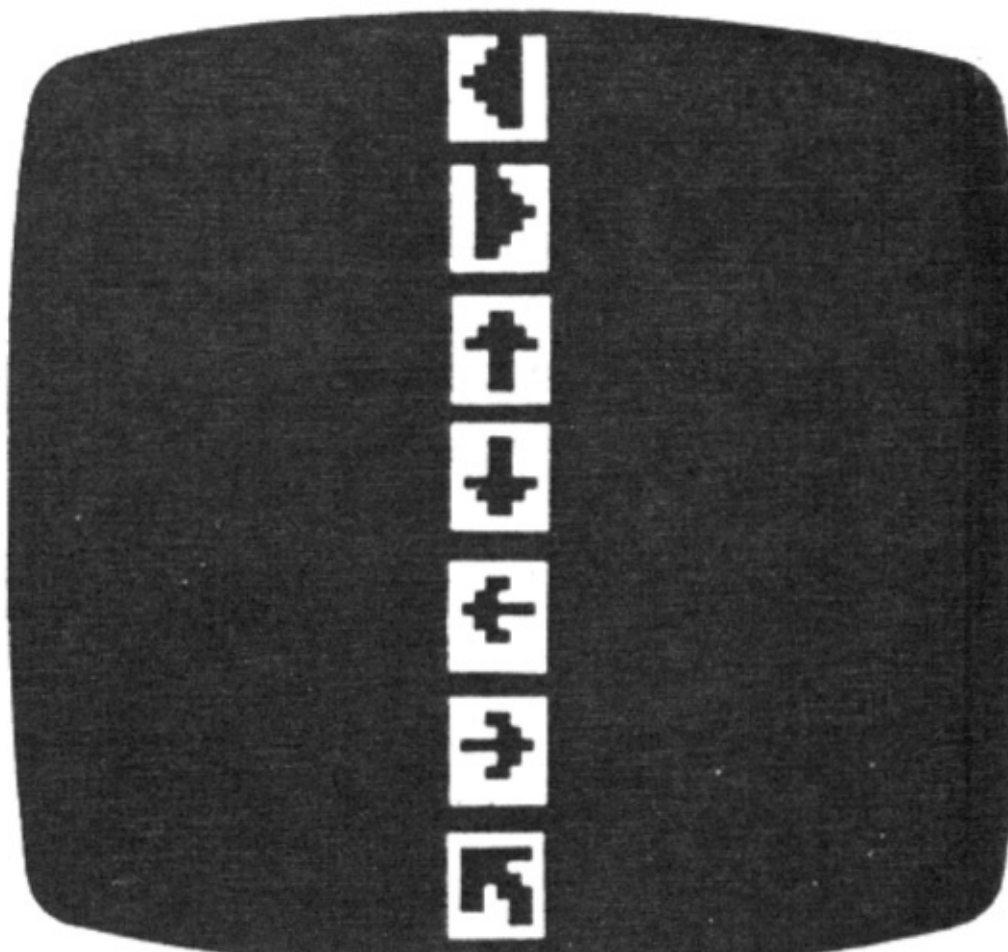


PRESIONE Y

MANTENGA PRESIONE



O



AGRUPAMIENTO	Una lista de valores numéricos almacenados en una serie de ubicaciones de memoria precedidas por una sentencia DIM. Se le identifica por una variable de grupo; sus elementos individuales se designan por el nombre de la variable con subcriptores.
ALFANUMERICO	Una letra del alfabeto (de A a Z), una cifra (de 0 a 9) o algún símbolo. (Ni los símbolos de puntuación ni los gráficos).
ASUMIDO	Modo o condición supuesta por el computador, mientras no se le instruya de lo contrario. Por ejemplo, asumirá comunicación a través de la pantalla y del teclado a no ser que se le indiquen otros dispositivos de entrada/salida.
ATASCII	Sigla de Código Americano Normalizado para Intercambio de Información (ASCII) ATARI.
BASIC	Lenguaje de programación de alto nivel. Sigla de Código de Instrucción de todo Propósito para Principiantes. Desarrollado por los señores Kemeny y Kurtz en el Dartmouth College en 1963.
BINARIO	Sistema numérico de base dos. Así los dos únicos dígitos posibles son 0 y 1, los que pueden usarse en el computador para representar "verdadero" y "falso", "sí" y "no" etc.
BIT	Contracción de binary digit (dígito binario). Puede imaginarse el bit como representando "verdadero" o "falso", conexión o desconexión de un circuito, o cualquier otro tipo de concepto de dos alternativas. El bit es la menor unidad de información con la que puede trabajar un computador.
BYTE	Por lo común ocho bits, (lo suficiente para representar el número 255 decimal o 11111111 en notación binaria). Un byte de información permite representar un número comprendido entre 0 y 255 o un carácter ATASCII.
CARACTER DE CONTROL	Carácter producido al mantener presionada la tecla CONTROL mientras simultaneamente se pulsa otra.
CARACTER ESPECIAL	Un carácter que puede ser desplegado por el computador, pero que no es ni una letra ni un número. Los caracteres gráficos ATARI son caracteres especiales, como lo son también los caracteres de puntuación.
CARGAR	Traspasar desde la grabadora de programa o la unidad de disco, a la memoria del computador algún tipo de información, en especial programas o datos.
CPU	Vea UCP, Unidad Central de Procesamiento.
CODIGO	Instrucciones escritas en un lenguaje comprensible para el computador.

COMANDO	Una instrucción al computador que se ejecuta de inmediato. Un buen ejemplo es el comando de BASIC: RUN (vea sentencia).
COMPUTADOR	Un dispositivo que puede recibir y a continuación seguir instrucciones para manejar información. Tanto las instrucciones como la información (los datos) pueden cambiar de un momento a otro. La diferencia entre una calculadora programable y un computador esta en la capacidad de éste de manejar tanto números como texto; las calculadoras normalmente sólo procesan números.
CONCATENACION	El proceso de unir dos o más strings, programas o archivos para formar uno más largo.
CONSULTA	Símbolo que aparece sobre la pantalla, indicando que el computador esta listo para aceptar un ingreso de teclado. En Basic Atari tiene la forma de la palabra READY (listo). También se usa "?" como consulta para solicitar al usuario para que ingrese (INPUT) un dato o tome otro tipo de acción apropiada.
CURSOR	Un pequeño cuadrado desplegado en la pantalla y que muestra donde aparecerá el próximo carácter que se digite.
DESVIO	Vea ramificación.
DIGITAL	Información que puede representarse por un conjunto de bits. Virtualmente todos los computadores, y en especial los micros, recurren a la representación digital.
DISKETTE	Un pequeño disco. Constituye un medio de grabación/reproducción similar a la cinta, pero realizado con la forma de un disco plano y flexible al interior de un sobre más rígido que lo protege mecánicamente. La gran ventaja del diskette sobre la cinta es la accesibilidad practicamente inmediata que permite a cualquier dato sobre su superficie. El Sistema de Computación Personal ATARI, puede controlar hasta cuatro unidades de disco simultaneamente. En este manual, se usan indistintamente los términos diskettes y disco, aunque en rigor debería decirse minidiskette flexible de 5 1/4 pulgadas.
D.O.S.	Abreviación de Sistema Operativo de Disco. Es el software o conjunto de programas que permiten el uso de un sistema de disco. El Sistema Computacional ATARI puede emplear diferentes sistemas operativos de disco. Los más conocidos son DOS 2.0 y DOS 3.0 de ATARI y OS+versiones 2 y 4 de O.S.S. En general estos sistemas operativos de disco no son compatibles entre sí, aunque existen programas de transferencia entre unos y otros.
EDITAR	Hacer correcciones o modificaciones en un programa o en datos.
EJECUTAR	Hacer lo que especifica un comando o programa; correr un programa total o parcialmente.
ENTRADA	Vea ingreso.

ENTRADA/SALIDA	Característica simultanea y/o alternativa de los periféricos para entregar y/o recibir información desde o hacia el computador o usuario.
EXPRESION	Una combinación de variables, números y operadores (como +, —, etc.) que pueden ser reducidos a una sola cantidad. Esta cantidad puede ser un string o un número.
FORMATO	La especificación de la forma en que algún tipo de información debe aparecer en la pantalla, el impresor, la cinta o el diskette.
GENERADOR DE NUMEROS ALEATORIOS	Puede ser un circuito (como en el caso ATARI) o un programa que suministra secuencias de números cuyos valores son difíciles de predecir. Se usa principalmente para la toma de decisiones en programas de juego etc.
GRABAR	Copiar un programa u otro tipo de datos (desde la memoria RAM) a un medio de almacenamiento secundario como diskette o cinta magnética.
HARDWARE	Circuitería; todo el aparataje físico, mecánico y electrónico de que esta constituido el computador.
INCREMENTAR	Aumentar la magnitud o el valor agregando (por lo común) uno; muy usado para contar. (Por ejemplo la cantidad de veces que se ha recorrido un bucle).
INGRESO	Transferencia de información al computador. Salida es la transferencia inversa: desde el computador hacia afuera. En este manual ingreso y salida se entienden siempre relativos al computador.
INSTRUCCION	Vea sentencia.
INTERACTIVO	Un sistema que responde rápidamente al usuario, normalmente dentro de 1 ó 2 segundos. Todos los computadores personales son interactivos.
INTERFAZ	La circuitería (que puede ser muy compleja y comprender incluso un microprocesador dedicado) que permite la intercomunicación entre diferentes dispositivos de un sistema computacional. Entre microcomputadores y sus periféricos es muy común el uso de las interfaces RS-232C o V-24 (especialmente para comunicaciones por modems y líneas telefónicas) y las tipo Centronics para impresores.
I/O	Vea entrada/salida.
IOCB	Bloque de Control de Entrada/Salida. Un bloque de datos en RAM que contiene la información necesaria para que el sistema operativo (O.S.) efectúe las operaciones de entrada/salida.
K	Kilo; se usa en la jerga computacional para designar la cantidad de $1024=2^{10}$. Así, un kilobyte o KB tiene 1024 bytes. Normalmente la letra K corresponde también al código de dispositivo del teclado (keyboard).

LENGUAJE	Un conjunto de convenciones que especifican como indicarle al computador lo que debe hacer.
LOAD	Vea cargar.
MEMORIA	La parte del computador que almacena información, datos, programas, instrucciones, etc. La memoria RAM (memoria de acceso aleatorio) es alterable, es decir, su contenido puede ser cambiado. En ella se almacenan datos, programas, etc. en forma temporal (por ejemplo durante la ejecución de un programa). La memoria ROM (memoria de lectura solamente) no es alterable y normalmente contiene las instrucciones y datos invariables: sistema operativo, juego de caracteres, etc.
MENU	Una lista de opciones de entre las cuales debe elegir el usuario.
MICROCOMPUTADOR	Un computador construido alrededor de un microprocesador realizado en un sólo circuito integrado. En el caso del microcomputador ATARI el microprocesador es de modelo 6502.
OUTPUT	Ver salida.
PALABRA CLAVE	Palabra que tiene un significado como instrucción o comando en un lenguaje computacional y que por lo tanto no debe usarse como nombre de variable o como primera parte del nombre de una variable.
PALABRA RESERVADA	Vea palabra clave.
PANTALLA	El dispositivo de imagen, generalmente el televisor. En Basic ATARI el dispositivo de imagen obedece al código de dispositivo "S".
PARALELO	El acontecer simultáneo de dos o más cosas. Normalmente se usa en relación a la comunicación de datos en forma binaria: Los 8 bits de cada byte se transfieren simultáneamente por 8 conductores (alambre, pistas de circuito impreso, etc.). Lo contrario de seriado.
PERIFERICO	Dispositivo de entrada/salida de datos del computador como: pantalla, teclado, impresor, etc.
PIXEL	Elemento de imagen. Un punto sobre la pantalla. Su tamaño y cantidad de colores que puede asumir depende del modo gráfico escogido.
PRECEDENCIA	Reglas que determinan la prioridad con que se conducen las operaciones, especialmente en relación a los operadores lógicos y aritméticos.
PROMPT	Vea consulta.
RAM	Memoria de acceso aleatorio. La memoria primaria en la mayoría de los computadores. La memoria RAM se usa para almacenar datos y programas.
RAMIFICACION	Las instrucciones de programas y consecuencias relacionadas con la alteración intencional de la secuencia ascendente de los

números de líneas de programa, según la cual normalmente se ejecuta un programa. Las ramificaciones o desvíos pueden ser incondicionales (se efectúan siempre) o condicionales (sólo se efectúan bajo ciertas condiciones especificadas por el programa).

ROM	Memoria de lectura solamente, inalterable por el usuario. Contiene la información permanente (datos y programas) necesaria para el funcionamiento del computador.
RUTINA	Vea subrutina.
SALIDA	Entrega de información desde el computador hacia el usuario o un periférico. Vea también entrada y entrada/salida.
SAVE	Vea grabar.
SENTENCIA	Una instrucción para el computador. Vea también comando. Si bien todos los comandos son sentencias, no todas las sentencias constituyen comandos. Una sentencia contiene un número de línea (en el modo diferido), una palabra clave, el valor sobre el que opera y el comando RETURN.
SERIADO	Serial o serie, lo contrario de paralelo, es la forma de transferencia de datos bit por bit secuencialmente por un conductor.
SOFTWARE	En oposición a "hardware", constituye todo lo que son programas y datos.
STRING	Una secuencia de letras, números y otros caracteres. Puede almacenarse en una variable string o literal. El nombre de una variable debe terminar en \$.
STRING VACIO	Un string que no contiene ningún carácter.
SUBROUTINA	Parte de un programa que puede ejecutarse a consecuencia de una instrucción BASIC especial: GOSUB. De hecho esto da a una sentencia el poder de un programa completo.
TRC	Tubo de Rayos Catódicos. Es la pantalla o tubo de imagen del Televisor o monitor. En la práctica, la sigla TRC o su equivalente inglés CTR, frecuentemente se usa para referirse al receptor de TV o al monitor que sirve de dispositivo de despliegue de salida del computador.
UCP	Unidad Central de Procesamiento. En microcomputadores como los ATARI la UCP o CPU (en inglés) llamada también MPU y realizada en un sólo circuito integrado, es aquella parte del computador que controla la memoria y los periféricos. Los sistemas de computación ATARI emplean el microprocesador 6502.
VARIABLE	Una variable puede imaginarse como una caja en la cual puede almacenarse un valor. Estos valores típicamente están constituidos por números y caracteres (strings).
VENTANA	Parte de la pantalla de imagen, dedicada a un propósito específico como por ejemplo al despliegue de texto o gráficos.

 A P E N D I C E H

 PROGRAMAS DE USUARIO

Este apéndice contiene programas y rutinas que demuestran las diversas capacidades de los sistemas de computación personal ATARI. Se incluye en este apéndice un programa de conversión decimal/hexadecimal para aquellos usuarios que escriban programas que requieren este tipo de conversión.

SALDADOR DE CHEQUERA

Este es uno de los programas tradicionales que todo programador novicio escribe; permite el ingreso de cheques pendientes y depósitos no registrados, lo mismo que cheques cobrados y depósitos acreditados.

```

10 DIM R$(1),TEX$(40),TEX1$(33),TEX2$(33),TEX3$(33),TEX4$(33),TEX5$(33),TEX6$(33)
20 PENDIENTE=0
30 GRAPHICS 0:?:?:? "          SALDADOR DE CUENTA  ":?
40 ? "I Para hacer correcciones, basta I      I  Ingresar valores negativos.  I":
PRINT
50 TEX1$="CHEQUE ANTERIOR PENDIENTE          "
60 TEX2$="DEF.ANTERIOR SIN ACREDITAR        "
70 TEX3$="CHEQUE ANTERIOR RECIEN REGISTRADO"
80 TEX4$="DEPOS. ANTERIOR RECIEN ACREDITADO"
90 TEX5$="NUEVO CHEQUE (O CARGO SERVICIO)  "
100 TEX6$="NUEVO DEPOSITO  (O INTERESES)    "
150 TRAP 150:?: "Ingrese el saldo de su chequera":INPUT SUSALDO:PRINT
160 TRAP 160:?: "Ingrese el saldo segun cartola ":INPUT SALDOBA:PRINT
165 TRAP 40000
170 GOTO 190
180 CLOSE #1:?: "EL IMPRESOR NO ESTA OPERATIVO."
185 ? "POR FAVOR,REVISE LAS CONEXIONES."
190 PERM=0
200 ? "Desea registro por impresor ":INPUT R$:PRINT
210 IF LEN(R$)=0 THEN 200
220 IF R$(1,1)="N" THEN 400
230 IF R$(1,1)<>"S" THEN 200
240 TRAP 190
250 LPRINT :REM PRUEBA IMPRESOR
260 PERM=1
280 LPRINT "SU SALDO INICIAL ES $";SUSALDO;","
290 LPRINT "SEGUN EL BANCO ES $";SALDOBA:LPRINT
400 TRAP 400:?:?: "Escoja uno de los siguientes":PRINT
410 ? "(1) ";TEX1$

```



```

415 ? "(2) ";TEX2$
420 ? "(3) ";TEX3$
425 ? "(4) ";TEX4$
430 ? "(5) ";TEX5$
435 ? "(6) ";TEX6$
440 ? "(7) SALDO"
490 PRINT
500 INPUT N:IF N<1 OR N>7 THEN 400
505 TRAP 40000
510 ON N GOSUB 1000,2000,3000,4000,5000,6000,7000
520 TEX$="SU NUEVO SALDO ES ":CANTIDAD=SUSALDO:GOSUB 8000
530 TEX$="EL NUEVO SALDO DEL BANCO ES":CANTIDAD=SALDOBA:GOSUB 8000
540 TEX$="CHEQUES - DEPOSITOS PENDIENTES =":CANTIDAD=PENDIENTE:GOSUB 8000
545 IF PERM THEN LPRINT
550 GOTO 400
1000 REM CHEQUE ANTERIOR PENDIENTE
1010 TEX$=TEX1$:GOSUB 8100
1020 PENDIENTE=PENDIENTE+CANTIDAD
1030 RETURN
2000 REM DEP.ANTERIOR SIN ACREDITAR
2010 TEX$=TEX2$:GOSUB 8100
2020 PENDIENTE=PENDIENTE-CANTIDAD
2030 RETURN
3000 REM CHEQUE ANTERIOR COBRADO
3010 TEX$=TEX3$:GOSUB 8100
3020 SALDOBA=SALDOBA-CANTIDAD
3030 RETURN
4000 REM DEP.ANTERIOR ACREDITADO
4010 TEX$=TEX4$:GOSUB 8100
4020 SALDOBA=SALDOBA+CANTIDAD
4030 RETURN
5000 REM NUEVO CHEQUE (O CARGO SERVICIO) COBRADO
5010 TEX$=TEX5$:GOSUB 8100
5020 SUSALDO=SUSALDO-CANTIDAD
5030 ? "NUEVO CHEQUE, AUN SIN COBRAR ";:INPUT R$
5040 IF LEN(R$)=0 THEN 5030
5050 IF R$(1,1)<>"N" THEN 5060
5055 SALDOBA=SALDOBA-CANTIDAD
5057 IF PERM THEN LPRINT "CHEQUE COBRADO."
5058 RETURN
5060 IF R$(1,1)<>"S" THEN 5030
5070 PENDIENTE=PENDIENTE+CANTIDAD
5075 IF PERM THEN LPRINT "CHEQUE AUN SIN COBRAR."
5080 RETURN
6000 REM NUEVO DEP. (O INTERES) ACREDITADO
6010 TEX$=TEX6$:GOSUB 8100
6020 SUSALDO=SUSALDO+CANTIDAD
6030 ? "FUE ACREDITADO SU ULTIMO DEPOSITO ";:INPUT R$
6040 IF LEN(R$)=0 THEN 6030
6050 IF R$(1,1)<>"S" THEN 6060
6052 SALDOBA=SALDOBA+CANTIDAD
6053 IF PERM THEN LPRINT "DEPOSITO ACREDITADO."
6055 RETURN
6060 IF R$(1,1)<>"N" THEN 6030
6070 PENDIENTE=PENDIENTE-CANTIDAD
6075 IF PERM THEN LPRINT "DEPOSITO SIN ACREDITAR."
6080 RETURN
7000 REM SALDO
7010 ? "EL SALDO DEL BANCO MENOS (CHEQUES SIN COBRAR-DEPOSITOS SIN ACRED.) DEBER
IA SER IGUAL AL SALDO SUYO."
7020 DIF=SUSALDO-(SALDOBA-PENDIENTE)
7030 IF DIF<>0 THEN 7040
7035 ? "ES $";SALDOBA;" EL SALDO FINAL DE SU CARTOLA DEL BANCO ";:INPUT R$
7036 IF LEN(R$)=0 THEN 7035

```

```

7037 IF R$(1,1)="S" THEN ? "FELICITACIONES:SU SALDO ESTA DE ACUER-DO CON EL BANC
0!":END
7038 GOTO 7060
7040 IF DIF>0 THEN ? "EL TOTAL DE SU SALDO ESTA $";DIF;"          POR SOBRE EL
SALDO DEL BANCO.":GOTO 7060
7050 ? "EL TOTAL DE SU SALDO ESTA $";-DIF;"          BAJO EL SALDO DE SU BANCO."
7060 ? "DESEA HACER UNA CORRECCION AHORA ?"
7070 ? "RECUERDE, QUE PARA LAS CORRECCIONES    PUEDE RECURRIR A VALORES NEGATIVOS
."
7080 ? "INGRESE 'S' O 'N'";:INPUT R$
7090 IF LEN(R$)=0 THEN END
7100 IF R$(1,1)="S" THEN RETURN
7110 IF R$(1,1)<>"N" THEN 7080
7120 END
7999 REM Rutina de impresion de mensajes
8000 ? TEX$;" $";CANTIDAD
8010 IF PERM=1 THEN LPRINT TEX$;" $";CANTIDAD
8020 RETURN
8100 REM Rutina de ingresos
8110 TRAP 8110: ? "ANOTE LA CANTIDAD DEL ";TEX$;:INPUT CANTIDAD
8120 TRAP 40000
8130 IF PERM=1 THEN LPRINT TEX$;" $";CANTIDAD
8140 RETURN

```

ORDENAMIENTO DE BURBUJA

Este programa usa el operador de comparación de strings " < = " que ordena los strings de acuerdo a los valores ATASCII de los diversos caracteres. Como el BASIC ATARI no contempla agrupamientos de strings, todos los strings que se usan en este programa constituyen substrings de un solo string mayor. Un ordenamiento de burbuja, aunque relativamente lento cuando los items a ordenar son muchos, es fácil de escribir, bastante corto y más fácil de entender que otros ordenamientos más complejos.

```


10 DIM R$(1)
20 GRAPHICS 0: ? "          ORDENAMIENTO DE STRINGS": ?
30 TRAP 30: ? "Ingrese el largo de string maximo.": ? : INPUT LASM: LASM1=LASM-1
35 IF LASM<1 OR INT(LASM) <> LASM THEN ? "POR FAVOR,INGRESE UN ENTERO POSITIVO": GOTO 30
40 TRAP 40: ? : ? "Ingrese la cantidad maxima de strings."
41 ? "(STRINGS MAS CORTOS QUE EL MAXIMO SE RELLENAN CON ESPACIOS EN BLANCO.):": F
RINT
42 INPUT INGRESOS
45 IF INGRESOS<2 OR INT(INGRESOS) <> INGRESOS THEN ? "POR FAVOR,INGRESE UN ENTERO
POSITIVO >1": GOTO 40
47 TRAP 40000
50 DIM A$(LASM*INGRESOS),TEMP$(LASM)
60 ? : ? "Ingrese los strings de a uno cada vez."
70 ? : ? "Ingrese un string vacio para terminar.(simplemente presione RETURN.)"
75 ? : ? "POR FAVOR,ESPERE MIENTRAS SE LIMPIAN LOS STRINGS..."
80 FOR I=1 TO LASM*INGRESOS: A$(I,I)=" ": NEXT I
85 ? : ?
90 I=1
100 FOR J=1 TO INGRESOS
110 ? "#";J;" ": INPUT TEMP$
120 IF LEN(TEMP$)=0 THEN INGRESOS=J-1: GOTO 190
130 A$(I,I+LASM1)=TEMP$
140 I=I+LASM
150 NEXT J
190 ? : ? : ? "POR FAVOR,ESPERE MIENTRAS SE HACE EL ORDENAMIENTO...":
200 GOSUB 1000: REM IR A SUBROUTINA DE ORDENAMIENTO
202 ? : ?
205 I=1
210 FOR K=1 TO INGRESOS
220 ? "#";K;" ": A$(I,I+LASM1)
225 I=I+LASM
230 NEXT K
240 TRAP 300: ? : ? "DESEA UNA COPIA IMPRESA": INPUT R$
250 IF R$(1,1)="S" THEN 400
300 END
400 I=1: LPRINT : FOR K=1 TO INGRESOS
420 LPRINT "#";K;" ": A$(I,I+LASM1)
430 I=I+LASM: NEXT K: END
1000 REM RUTINA DE ORDENAMIENTO DE BURBUJA PARA STRINGS
1010 REM INGRESE: A$,LASM,INGRESOS
1015 REM TEMP$ DEBE TENER LA DIMENSION LASM
1020 LASM1=LASM-1: MAX=LASM*(INGRESOS-1)+1
1040 FOR I=1 TO MAX STEP LASM
1050 HECHO=1

```



```
1060 FOR K=1 TO MAX-I-LASM1 STEP LASM
1070 KLASM1=K+LASM1:KLASM=K+LASM:KLASMLASM1=KLASM+LASM1
1080 IF A$(K,KLASM1)<=A$(KLASM,KLASMLASM1) THEN GOTO 1110
1090 HECH0=0
1100 TEMP$=A$(K,KLASM1):A$(K,KLASM1)=A$(KLASM,KLASMLASM1):A$(KLASM,KLASMLASM1)=T
EMP$
1110 NEXT K
1120 IF HECH0 THEN RETURN
1130 NEXT I
1140 RETURN
```

DEMOSTRADOR DE CARACTERES DE TEXTO

Este programa imprime en la pantalla todos los caracteres en sus colores asumidos en los modos 0, 1 y 2. Al ingresar este programa recuerde que el símbolo para limpiar pantalla (clear ) aparece en el listado de impresor como " }".

```

1 DIM R$(1)
5 ? ">":REM LIMPIA PANTALLA(CLEAR)
10 ? "      Demostracion de"
20 ? "      Modos de texto GR.0,1 y 2 "
30 ? " Despliega el juego de caracteres                                de cada modo." :? :?
60 ESPERA=1000:REM NO. DE LINEA DE SUBROUTINA
70 CHBAS=756:REM DIRECCION DE LA BASE DE CARACTERES EN ROM
80 MAY=224:REM VALOR ASUMIDO DE CHBAS
90 MIN=225:REM MINUSCULAS Y CAR.GRAFICOS
95 GOSUB ESPERA
100 FOR L=0 TO 2
112 REM USE E: PARA GR.0
115 IF L=0 THEN OPEN #1,8,0,"E:":GOTO 118
116 REM USE S: PARA GR.1 Y 2
117 OPEN #1,8,0,"S:"
118 GRAPHICS L
120 PRINT "      GR. ";L:
130 FOR J=0 TO 7:REM 8 LINEAS
140 FOR I=0 TO 31:REM 32 CAR/LINE
150 K=32*I+J
155 REM NO DESPLIEGUE (CLEAR) NI (RETURN).
160 IF K=ASC(">") OR K=155 THEN 180
165 IF L=0 THEN PUT #1,ASC(">"):REM ESCAPE
170 PUT #1,K:REM DESPLIEGA CAR.
180 NEXT I
190 PRINT #1;" " :REM FIN DE LINEA
200 IF L<>2 OR J<>3 THEN 240
210 REM PANTALLA LLENA
220 GOSUB ESPERA
230 PRINT #1;">":REM (CLEAR)
240 NEXT J
250 GOSUB ESPERA
260 PRINT "MINUSCULAS Y GRAFICOS"
270 IF L<>0 THEN POKE CHBAS,MIN:GOSUB ESPERA
275 CLOSE #1
280 NEXT L
300 GRAPHICS 0:END
1000 REM ESPERA PARA SEGUIR
1010 ? :PRINT " PRESIONE RETURN PARA SEGUIR."
1020 INPUT R$
1030 RETURN

```

LUCES

Este programa demuestra otro aspecto de la gráfica ATARI. Usa el modo de alta resolución 7 para líneas mediante instrucciones PLOT y DRAWTO. En la línea 20, el título puede resultar más impresionante si se ingresa en video inverso.

```

10 FOR ST=1 TO 8:GRAPHICS 7
15 POKE 752,1
20 ? :? :? "          L U C E S   A T A R I":SETCOLOR 2,0,0
30 SETCOLOR 1,2*ST,8:COLOR 2
40 FOR DR=0 TO 400 STEP ST:TRAP 60
50 PLOT 0,0:DRAWTO 125,DR
60 NEXT DR:FOR N=1 TO 800:NEXT N:NEXT ST
70 FOR N=1 TO 2000:NEXT N:GOTO 10

```


GAVIOTA SOBRE EL OCEANO

Este programa combina gráfica con sonido. Los sonidos no son puros, sino que simulan el ruido del mar y los graznidos de la gaviota. Los símbolos gráficos usados para representar a la gaviota no son reproducibles en el impresor. Ingrese en la línea 20 los caracteres descritos en la línea 25, presionando simultaneamente la tecla CONTROL y la de la letra correspondiente.

```

10 DIM GAVIOTA$(4)
20 GAVIOTA$=""
25 REM GAVIOTA$="[CTRL G][CTRL F][CTRL R][CTRL R]"
30 ALAS=1:LIN=10:COL=10
40 GRAPHICS 1:POKE 756,226:POKE 752,1
50 SETCOLOR 0,0,0:SETCOLOR 1,8,14
60 PRINT #6;"      el oceano"
70 R=INT(RND(0)*11)
80 POSITION 17,17
90 FOR T=0 TO 10
100 SOUND 0,T,8,4
110 FOR A=1 TO 50:NEXT A
120 IF RND(0)>0.8 THEN FOR D=10 TO 5 STEP -1:SOUND 1,0,10,INT(RND(0)*10):NEXT D:
SOUND 1,0,0,0
130 GOSUB 200
140 NEXT T
150 FOR T=10 TO 0 STEP -1
160 SOUND 0,T,8,4
170 FOR A=1 TO 50:NEXT A
175 IF RND(0)>0.8 THEN FOR D=10 TO 5 STEP -1:SOUND 1,D,10,8:NEXT D:SOUND 1,0,0,0
180 GOSUB 200
190 NEXT T
195 GOTO 70
200 GOSUB 300
210 POSITION COL,LIN
220 PRINT #6;GAVIOTA$(ALAS,ALAS+1)
230 ALAS=ALAS+2:IF ALAS=5 THEN ALAS=1
240 RETURN
300 IF RND(0)>0.5 THEN RETURN
310 POSITION COL,LIN
320 PRINT #6;"  "
330 A=INT(RND(0)*3-1)
340 B=INT(RND(0)*3-1)
350 LIN=LIN+A
360 IF LIN=0 THEN LIN=1
370 IF LIN=20 THEN LIN=19
380 COL=COL+B
390 IF COL=0 THEN COL=1
400 IF COL>18 THEN COL=18
410 RETURN

```

LAPICES

Este programa requiere un bastón de control para cada jugador. Cada bastón tiene asociado un color. Maniobrando con los bastones, pueden crearse las más diversas figuras sobre la pantalla. Note el uso de los comandos STICK y STRIG.

```

1 GRAPHICS 0
2 ? "      VIDEO-LAPICES"
3 REM AGRUPAMIENTOS X e Y SON COORD.
6 REM PARA HASTA CUATRO JUGADORES.
7 REM AGUP. COLR DA COLORES.
10 DIM R$(1),X(3),Y(3),COLR(3)
128 ? :? "USE BASTONES PARA TRAZAR FIGURAS."
129 ? :? "PRESIONE BOTONES PARA CAMBIAR COLOR."
130 ? "  COLORES INICIALES":?
131 ? "    BASTON 1 : ROJO"
132 ? "    BASTON 2 : BLANCO"
133 ? "    BASTON 3 : AZUL"
134 ? "    BASTON 4 : NEGRO (FONDO)"
135 ? :? " LA POSICION NEGRA SE INDICA POR UN      DESTELLO ROJO."
136 ? :? " EN MODO 8, LOS BASTONES 1 Y 3 SON      BLANCOS Y 4 ES AZUL."
138 ? :? " CUANTOS JUGADORES (1-4) ";
139 INPUT R$:IF LEN(R$)=0 THEN R$="1"
140 BASMAX=VAL(R$)-1
145 IF BASMAX<0 OR BASMAX>=4 THEN 138
147 ? :PRINT " GRAPHICS 3 (40*24), 5 (80*48),"
150 PRINT " 7 (160*96) u 8 (320*192)";
152 INPUT R$:IF LEN(R$)=0 THEN R$="3"
153 R=VAL(R$)
154 IF R=3 THEN XMAX=40:YMAX=24:GOTO 159
155 IF R=5 THEN XMAX=80:YMAX=48:GOTO 159
156 IF R=7 THEN XMAX=160:YMAX=96:GOTO 159
157 IF R=8 THEN XMAX=320:YMAX=192:GOTO 159
158 GOTO 147:REM R INVALIDO
159 GRAPHICS R+16
160 FOR I=0 TO BASMAX:X(I)=XMAX/2:Y(I)=YMAX/2:NEXT I:REM PARTIDA EN EL CENTRO DE
  LA PANTALLA
161 IF A<>8 THEN 166
162 FOR I=0 TO 2:COLR(I)=I+1:NEXT I
163 SETCOLOR 1,9,14:REM CELESTE
165 GOTO 180
166 FOR I=0 TO 2:COLR(I)=I+1:NEXT I
167 SETCOLOR 0,4,6:REM ROJO
168 SETCOLOR 1,0,14:REM BLANCO
180 COLR(3)=0
295 FOR J=0 TO 3
300 FOR I=0 TO BASMAX:REM REVISION BASTONES
305 REM REVISION BOTONES

```

```

310 IF STRIG(I) THEN 321
311 FOR T=0 TO 100:NEXT T:IF R<>8 THEN 320
312 COLR(I)=COLR(I)+1:IF COLR(I)=2 THEN COLR(I)=0:REM MOD0 DE 2 COLORES
313 GOTO 321
320 COLR(I)=COLR(I)+1:IF COLR(I)>=4 THEN COLR(I)=0:REM MOD0 DE 4 COLORES
321 IF J>0 THEN COLOR COLR(I):GOTO 325
322 IF COLR(I)=0 THEN COLOR 1:GOTO 325
323 COLOR 0:REM PARPADEO DEL CURSOR
325 PLOT X(I),Y(I)
330 LEBAS=STICK(I):REM LECTURA BASTON
340 IF LEBAS=15 THEN 530:REM SIN MOVIMIENTO
342 COLOR COLR(I):REM CONFIRMACION COLOR
344 PLOT X(I),Y(I)
350 IF LEBAS>=8 THEN 390
360 X(I)=X(I)+1:REM A LA DERECHA
365 REM SI FUERA DE RANGO VUELVE POR ATRAS
370 IF X(I)>=XMAX THEN X(I)=0
380 GOTO 430
390 IF LEBAS>=12 THEN 430
400 X(I)=X(I)-1:REM A LA IZQUIERDA
410 IF X(I)<0 THEN X(I)=XMAX-1
430 IF LEBAS<>5 AND LEBAS<>9 AND LEBAS<>13 THEN 470
440 Y(I)=Y(I)+1:IF Y(I)>=YMAX THEN Y(I)=0:REM HACIA ABAJO
460 GOTO 500
470 IF LEBAS<>6 AND LEBAS<>10 AND LEBAS<>14 THEN 500
480 Y(I)=Y(I)-1:IF Y(I)<0 THEN Y(I)=YMAX-1:REM HACIA ARRIBA
500 PLOT X(I),Y(I)
530 NEXT I
535 NEXT J
540 GOTO 295

```


TECLADO MUSICAL

Este programa asigna notas musicales a la corrida de teclas superior. Presione sólo una tecla a un tiempo.

Tecla	Nota
INSERT	Si
CLEAR	Si bemol (o La sostenido)
0	La
9	La bemol (o Sol sostenido)
8	Sol
7	Fa sostenido (o Sol bemol)
6	Fa
5	Mi
4	Mi bemol (o Re sostenido)
3	Re
2	Re bemol (o Do sostenido)
1	Do

```

10 DIM ESCALA(37), TONO(12)
20 GRAPHICS 0: ? "PROGRAMA    TECLADO MUSICAL"
25 ? : ? "PRESIONE LAS TECLAS 1-9, 0, <, > PARA    LAS NOTAS.";
27 ? "SUELTE UNA TECLA ANTES DE    APRETAR LA SIGUIENTE; ";
28 ? "DE OTRO MODO    PUEDE HABER RETARDOS."
30 FOR X=1 TO 37: READ A: ESCALA(X)=A: NEXT X
40 FOR X=1 TO 12: READ A: TONO(X)=A: NEXT X
50 OPEN #1, 4, 0, "K: "
55 ANTERIOR=-1
60 A=PEEK(764): IF A=255 THEN 60
63 IF A=ANTERIOR THEN 100
65 ANTERIOR=A
70 FOR X=1 TO 12: IF TONO(X)=A THEN SOUND 0, ESCALA(X), 10, 8: GOTO 100
80 NEXT X
100 I=INT(PEEK(53775)/4): IF (I/2)=INT(I/2) THEN 60
110 POKE 764, 255: SOUND 0, 0, 0, 0: ANTERIOR=-1: GOTO 60
200 DATA 243, 230, 217, 204, 193, 182, 173, 162, 153, 144, 136, 128, 121, 114, 108, 102, 96, 91, 8
5, 81, 76, 72, 68, 64, 60
210 DATA 57, 53, 50, 47, 45, 42, 40, 37, 35, 33, 31, 29
220 DATA 31, 30, 26, 24, 23, 27, 51, 53, 48, 50, 54, 55

```

BLUES ELECTRONICOS

Este programa genera notas musicales al azar, para "escribir" algunas melodías bien interesantes para el bajo programado.

```

1 GRAPHICS 0:?:?:? "          BLUES ELECTRONICOS":?
2 PTR=1
3 LAN=1
5 ACORDE=1
6 ? "RITMO DEL BAJO ( 1 = MUY RAPIDO )";
7 INPUT RITMO
8 GRAPHICS 2+16:GOSUB 2000
10 DIM BASE(3,4)
20 DIM GRAVE(3)
25 DIM LINEA(16)
26 DIM JAM(3,7)
30 FOR X=1 TO 3
40 FOR Y=1 TO 4
50 READ A:BASE(X,Y)=A
60 NEXT Y
70 NEXT X
80 FOR X=1 TO 3:READ A:GRAVE(X)=A
90 NEXT X
95 FOR X=1 TO 16:READ A:LINEA(X)=A:NEXT X
96 FOR X=1 TO 3
97 FOR Y=1 TO 7
98 READ A:JAM(X,Y)=A:NEXT Y:NEXT X
100 GOSUB 500
110 T=T+1
115 GOSUB 200
120 GOTO 100
200 REM PROCESAR LO AGUDO
205 IF RND(0)<0.25 THEN RETURN
210 IF RND(0)<0.5 THEN 250
220 NT=NT+1
230 IF NT>7 THEN NT=7
240 GOTO 260
250 NT=NT-1
255 IF NT<1 THEN NT=1
260 SOUND 2,JAM(ACORDE,NT),10,NT*2
270 FOR T=0 TO RITMO:NEXT T
280 RETURN
500 REM PROCESAR LO GRAVE
510 IF BAJO=1 THEN 700
520 DURB=DURB+1
530 IF DURB<>RITMO THEN 535
531 BAJO=1:DURB=0
535 SOUND 0,GRAVE(ACORDE),10,2
540 SOUND 1,BASE(ACORDE,LAN),10,2
550 RETURN

```

```
700 SOUND 0,0,0,0
710 SOUND 1,0,0,0
720 DURB=DURB+1
730 IF DURB<>1 THEN 800
740 DURB=0:BAJO=0
750 LAN=LAN+1
760 IF LAN<>5 THEN 800
765 LAN=1
770 PTR=PTR+1
780 IF PTR=17 THEN PTR=1
790 ACORDE=LINEA(PTR)
800 RETURN
1000 DATA 162,144,136,144,121,108,102,108,108,96,91,96
1010 DATA 243,182,162
1020 DATA 1,1,1,1,2,2,2,2,1,1,1,1,3,2,1,1
1030 DATA 60,50,47,42,40,33,29
1040 DATA 60,50,45,42,40,33,29
1050 DATA 81,68,64,57,53,45,40
2000 PRINT #6:PRINT #6:PRINT #6
2005 PRINT #6;"      Blues"
2006 PRINT #6
2010 PRINT #6;"      Electronicos"
2030 RETURN
```


PROGRAMA DE CONVERSION DECIMAL/HEXADECIMAL

Este programa convierte números hexadecimales en decimales y vice versa.

```

10 DIM A$(9),AD$(1)
20 GRAPHICS 0:?:?:? " CONVERSION DE NUMEROS HEXADECIMALES":?
30 ? :? "Ingrese 'D' para conversion DECHEX.":? "ingrese 'H' para conversion HEX
DEC.":INPUT A$
40 IF LEN(A$)=0 THEN 30
50 IF A$="H" THEN 300
60 IF A$<>"D" THEN 30
90 TRAP 90
100 ? :? " INGRESE UN NUMERO DECIMAL ENTRE      0 Y 9999999999."
110 ? "DEC:":INPUT N
120 IF N<0 OR N>1E+10 THEN GOTO 100
130 I=9
140 TEMP=N:N=INT(N/16)
150 TEMP=TEMP-N*16
160 IF TEMP<10 THEN A$(I,I)=STR$(TEMP):GOTO 180
170 A$(I,I)=CHR$(TEMP-10+ASC("A"))
180 IF N<>0 THEN I=I-1:GOTO 140
190 ? "HEX: ";A$(I,9)
200 GOTO 110
300 TRAP 300
310 ? :? " INGRESE UN NUMERO HEXADECIMAL ENTRE  0 Y FFFFFFFF."
320 ? "HEX:":INPUT A$
330 N=0
340 FOR I=1 TO LEN(A$)
345 AD$=A$(I,I):IF AD$<"0" THEN 300
350 IF A$(I,I)<="9" THEN N=N*16+VAL(AD$):GOTO 370
355 IF AD$<"A" THEN 300
357 IF AD$>"F" THEN 300
360 N=N*16+ASC(AD$)-ASC("A")+10
370 NEXT I
380 ? "DEC: ";N:
390 GOTO 320
400 END

```

APÉNDICE I

UBICACIONES DE MEMORIA

Nota: Muchas de estas ubicaciones son de interés primordial para programadores expertos y se incluyen aquí para su conveniencia. Las etiquetas (labels) dadas, son los usados por los programadores de ATARI, para hacer más legibles los programas

ETIQUETA	UBICACION		COMENTARIOS Y DESCRIPCION
	DECIMAL	HEXADECIMAL	
APPMHI	14, 15	D, E	Ubicación más alta usada por BASIC (LSB, MSB)
RTCLOK	18, 19, 20	12, 13, 14 ✓	Contador de cuadros de TV 1/60 seg.) (LSB, NSB, MSB)
SOUNDR	65	41	Bandera de I/O ruidoso (0= silencioso)
	77		Bandera de rotación de colores (128=modo de rotación), <i>0 no rota col. al estar en reposo.</i>
LMARGIN, RMARGIN	82, 83	52, 53	Márgenes izquierdo, derecho 2, 38 asumidos)
ROWCRS	84	54	Línea actual del cursor (ventana gráfica)
COLCRS	85, 86 <i>88, 89</i>	55, 56	Columna actual del cursor (ventana gráfica) <i>dirección izquierda sup-129</i>
OLDROW	90	5A	Línea previa del cursor (ventana gráfica)
OLDCOL	91, 92	5B	Columna previa del cursor (ventana gráfica)
	93	5C	Dato bajo el cursor (ventana gráfica, excepto en modo 0)
	<i>94, 95</i>	<i>5E, 5F</i>	<i>máxima dirección cursor</i>
NEWROW	96	60	Línea del cursor a la cual irá DRAWTO (400/800) (757 2F5 en modelos XL)
NEWCOL	97, 98	61, 62	Columna de cursor a la cual irá DRAWTO (400/800) (758, 759 2F6, 2F7 en modelos XL)
RAMTOP	106	6A	Límite superior de la memoria (cantidad de páginas)
LOMEM	128, 129	80, 81	Puntero de límite inferior de memoria BASIC
MEMTOP	144, 145	90, 91	Puntero de límite superior de memoria BASIC
STOPLN	186, 187	BA, BB	Número de línea en la que ocurrió STOP o TRAP (número binario de dos bytes)
ERRSAV	195	C3	Número de error
PTABW	201	C9	Paso de tabulación (valor asumido 10)

FRO	212, 213	D4, D5	Bytes bajo y alto retornados a Basic como resultado de la función USR
RADFLG	251	FB	Bandera RAD/DEG (0=radianes, 6=grados)
LPENH	559, 560, 561 564	22F (ver atcas) 234	0: Cuadro apagado, 34: encendido Valor horizontal de lápiz de luz 46: las doble
LPENV	565	235	Valor vertical de lápiz de luz
TXTRW	622 (623) 656	24E (prioridad) 290	0: coarse scroll, 255: fine scroll Línea del cursor (ventana de texto)
TXTCOL	675-699 657, 658 704-707	291, 292	Posiciones del tabulador Columna del cursor (ventana de texto)
COLOR0	708	2C4	Registro color objeto 0-3 Registro de color 0
COLOR1	709	2C5	Registro de color 1
COLOR2	710	2C6	Registro de color 2
COLOR3	711	2C7	Registro de color 3
COLOR4	712	2C8	Registro de color 4
MEMTOP	741, 742	3E5, 2E6	Puntero del límite superior de memoria de usuario del sistema operativo (LSB, MSB)
MEMLO	743, 744	2E7, 2E8	Puntero de límite inferior de memoria del sistema operativo (LSB, MSB)
CRSINH	752	2F0	Inhibición del cursor (0=cursor visible, 1= cursor apagado)
CHACT	755	2F3	Registro de modo de carácter (4=especular vertical, 2=normal, 1=en blanco, 0=normal inverso, 3: inversos (leucos)
CHBAS	756, 757	2F4 (ver 96)	Registro de la base de caracteres (asume 224) 224=mayúsculas, 206=minúsculas
NEWCOL	758-759	(ver 97-98)	57344-58367 52224-53247
ATACHR	763	2FB	Ultimo carácter ATASCII
CH	764	2FC	Ultima tecla presionada (código interno; 255 borra carácter)
FILDAT	765	2FD	Dato de relleno gráfico (XIO)
DSPFLG	766	2FE	Bandera de despliegue (1=despliegue de caracteres de control)
SSFLAG	767	2FF	Bandera partida (detención para paginación (0=listado normal) puesta por CONTROL 1
HATABS	794-831	31A	Tabla de direcciones de los administradores (3 bytes/administrador)
IOCB	832-1151	340	Bloques de control I/O (16 bytes /IOCB)
	1634-1791	680-6FE	RAM reserva

16: one line player serv.
32: Enable instr. to fetch data