A diagonal rainbow graphic consisting of multiple parallel lines in the colors of the spectrum, from blue at the top to red at the bottom.

ATARI[®]
PERSONAL COMPUTERS

**MANUAL
DE
BASIC**

ATARI

MANUAL DE BASIC

Distribuidor Exclusivo



Unimport Ibérica, S.A.

c/Dos Amigos, 3 28015 MADRID

ISBN: 84-398-0987-5
Depósito Legal: M. 17636-1985

Impreso en España / Printed in Spain by
Gráficas Lormo - Isabel Méndez, 15 - 28038 Madrid

ATARI-1

CODIGO DE ERRORES

CODIGO DE ERROR	MENSAJE DEL CODIGO DE ERRORES
2	Memoria insuficiente
3	Error de valor
4	Demasiadas variables
5	Error de longitud de la cadena
6	Error por falta de datos
7	Número mayor que 32767
8	Error en la declaración de entrada de datos
9	Error de DIM de la colección o de cadena
10	Desborde de la pila de argumentos
11	Error de desborde o insuficiencia
12	Línea no hallada
13	Declaracion FOR no correspondida
14	Error de una línea demasiado extensa
15	Error de una línea GOSUB o FOR suprimida
16	Error de RETURN
17	Error de GARBAGE (BASURA)
18	Error en la función VAL

Nota: Los siguientes errores son de INPUT/OUTPUT (I/O) (ENTRADA/SALIDA) que resultan durante la utilización de las unidades de discos, impresoras u otros dispositivos. Mayor información se provee con los HARDWARE auxiliares.

- 19 El programa LOAD (Cargar) es demasiado extenso
- 20 Número de dispositivo no válido
- 21 Error en la carga del fichero
- 128 Aborto de la tecla BREAK
- 129 IOCB
- 130 Dispositivo no existente
- 131 IOCB escritura únicamente
- 132 Orden no válida
- 133 Dispositivo o fichero no abierto
- 134 Mal número de IOCB
- 135 Error de lectura de IOCB únicamente
- 136 EOF (Final del fichero)
- 137 Registro truncado
- 138 Dispositivo desconectado
- 139 Dispositivo NAK
- 140 Error del bus serie
- 141 Cursor fuera de alcance
- 142 Error de Overrun en el bus serie
- 143 Error en la comprobación de los datos enviados
- 144 Error del dispositivo DONE
- 145 Error de la comprobación de lectura después de escribir
- 146 Función no permitida
- 147 Insuficiente RAM
- 160 Error del número de la unidad de disco
- 161 Demasiados ficheros abiertos
- 162 Disco completo
- 163 Error del sistema de entrada/salida del DOS
- 164 Número de fichero no correspondiente
- 165 Error de nombre de fichero
- 166 Error de longitud del dato POINT
- 167 Fichero bloqueado
- 168 Orden no válida
- 169 Directorio lleno
- 170 Fichero no hallado
- 171 POINT no válido

PREFACIO

Este manual asume el que el que lo utiliza haya leído el libro "TODO SOBRE ATARI", Libro Primero, o cualquier otro libro acerca del BASIC. La intención de este manual no es la de "enseñar" BASIC. Es una guía de referencia de las órdenes, declaraciones, funciones, y aplicaciones especiales del BASIC ATARI.

Algunos de los símbolos especiales en el conjunto de caracteres Atari no aparecen iguales en la impresora: por ejemplo, el símbolo de despejar la pantalla "⌈" aparece como un paréntesis, "}". Los ejemplos del texto fueron elegidos para ilustrar una función particular —no necesariamente "buenas" técnicas de programación.

Cada una de las secciones contiene grupos de órdenes, funciones o declaraciones que tratan de un aspecto particular del BASIC Atari. Por ejemplo, la Sección 9 contiene todas las declaraciones pertenecientes a las capacidades especiales gráficas del Atari. Los apéndices incluyen referencias rápidas a términos, mensajes de error, palabras claves del BASIC, posiciones de memoria, y el conjunto de caracteres ATASCII.

Debido a que no hay en él ninguna aplicación específica para el ordenador personal ATARI, este manual se dirige hacia aplicaciones generales y al público en general. El Apéndice H contiene programas que ilustran algunas de las capacidades del sistema ATARI.

CONTENIDOS

PREFACIO

1. INFORMACION GENERAL

Terminología	15
Anotaciones Especiales utilizadas en este Manual	18
Abreviaturas utilizadas en este Manual	19
Modos Operativos	21
Teclas de Funciones Especiales	21
Operadores Aritméticos	23
Operadores Lógicos	23
Precedencia del Operador	24
Funciones Incorporadas	25
Gráficos	25
Sonidos y Controladores de Juegos	25
Repetición de Teclado	25
Mensajes de Error	25

2. ORDENES

BYE	27
CONT	27
END	28
LET	28
LIST	28
NEW	29
REM	29
RUN	30
STOP	30

3. CARACTERISTICAS DE EDITADO

Editado de Pantalla	31
Teclas de Doble Función	32

Teclas de Control de Cursor	32
Teclas utilizadas con CONTROL	33
Teclas utilizadas con SHIFT	33
Teclas de Funciones Especiales	33
<hr/>	
4. DECLARACIONES DE PROGRAMA	
<hr/>	
FOR/TO/STEP/NEXT	35
GOSUB/RETURN	37
GOTO	38
IF/THEN	39
ON/GOSUB	41
ON/GOTO	41
POP	42
RESTORE	43
TRAP	44
<hr/>	
5. ORDENES Y DISPOSITIVOS DE ENTRADA Y SALIDA (I/O)	
<hr/>	
Dispositivos de Entrada/Salida	45
CLOAD	47
CSAVE	47
DOS	48
ENTER	48
INPUT	48
LOAD	49
LPRINT	50
NOTE	50
OPEN	50
POINT	52
PRINT	53
PUT	53
READ	54
SAVE	55
STATUS	55
XIO	55
Encadenado Programas	56
Modificando un Programa en un Disco	57
<hr/>	
6. BIBLIOTECA DE FUNCIONES	
<hr/>	
Funciones Aritméticas	59

ABS	59
CLOG	59
EXP	60
INT	60
LOG	60
RND	60
SGN	61
SQR	61
Funciones Trigonométricas	61
ATN	61
COS	61
SIN	61
DEG/RAD	61
Funciones de Propósitos Especiales	62
ADR	62
FRE	62
PEEK	62
POKE	63
USR	63

7. CADENAS

ASC	65
CHR\$	66
LEN	66
STR\$	67
VAL	67
Manipulación de Cadenas	67
Concatenación de Cadenas	68
Seccionando Cadenas	68
Comparaciones de Cadena y Clasificadores	69

8. TABLAS Y MATRICES

DIM	70
CLR	71

COMANDOS Y MODOS GRAFICOS

GRAPHICS	75
Modo Gráfico 0	76

Modos Gráficos 1 y 2	77
Modos Gráficos 3, 5 y 7	79
Modos Gráficos 4 y 6	79
Modo Gráfico 8	79
Modos Gráficos 9, 10 y 11	79
Modos Gráficos 12 y 13	79
Modos Gráficos 14 y 15	79
COLOR	79
DRAWTO	80
LOCATE	80
PLOT	81
POSITION	82
PUT/GET	82
SETCOLOR	82
XIO	85

10. SONIDOS Y CONTROLADORES DE JUEGO

SOUND	91
PADDLE	94
PTRIG	94
STOCK	95
STRIG	97

11. TECNICAS DE PROGRAMACION AVANZADAS

Conservación de Memoria	97
Programando en el Lenguaje de Máquina	99

APENDICE A

Guía Alfabética de las Palabras Reservadas del Basic	105
--	-----

APENDICE B

Mensaje de Error	113
------------------------	-----

APENDICE C

Conjunto de Caracteres ATASCII con Posiciones Decimales y Hexadecimales	119
---	-----

APENDICE D

Mapa de Memoria del ATARI 400/800 125

APENDICE E

Funciones Derivadas 128

APENDICE F

Versiones Imprimidas de los Caracteres de Control 131

APENDICE G

Glosario 133

APENDICE H

Programas para el Usuario 141

APENDICE I

Posiciones de Memoria 159

1 INFORMACION GENERAL

Esta sección explica la terminología BASIC, anotaciones especiales y abreviaturas utilizadas en este manual, y las teclas especiales del teclado del Sistema de Ordenadores Personales ATARI. También señala otras secciones en donde las órdenes BASIC tratan con aplicaciones especiales y específicas.

TERMINOLOGIA

BASIC: Código de Instrucciones Simbólicas de Uso General para Principiantes.

Palabras Clave del BASIC: Es cualquier palabra reservada que sea "legal" en el lenguaje BASIC. Puede utilizarse en una declaración, como una orden, o para cualquier otro propósito. (Ver el Apéndice A para una lista de todas las palabras reservadas o palabras clave en el BASIC ATARI.)

Declaraciones BASIC: Generalmente empieza con una palabra clave como LET, PRINT o RUN.

Constante: Una constante es un valor expresado como un número en vez de ser representado por un nombre de variable. Por ejemplo, en la declaración $X = 100$, X es una variable; 100 es una constante (Ver Variables).

Ordenes en cadena: Son órdenes múltiples (o declaraciones de programas) puestas en la misma línea enumerada y separada por dos puntos.

Expresión: Una expresión es cualquier combinación "legal" de variables, constantes, operadores y funciones utilizadas juntas para computar un valor. Las expresiones pueden ser aritméticas, lógicas o en cadena.

Función: Una función es una computación incorporada en el ordenador a fin de que ésta pueda ser llamada por el programa de quien lo utilice.

Una función no es una declaración; es parte de una expresión. Realmente es una sub-rutina utilizada para computar un valor que entonces es "devuelto" al programa central cuando la sub-rutina "regresa". COS (coseno), RND (Random (al azar)), FREE (espacio de memoria no utilizada), e INT (número entero)

son ejemplos de funciones. En muchos casos el valor es simplemente asignado a una variable (almacenado en una variable) para uso posterior. En otros casos pueden ser inmediatamente impresos en la pantalla. Ver la Sección 6 para mayor detalle acerca de las funciones. Ejemplos de funciones como pueden aparecer en los programas son:

10 PRINT RND(0)

(Imprime el número al azar que es "devuelto")

10 X=100+COS(45)

(Agregue el valor "devuelto" a 100 y almacene el total en la variable X)

Línea Lógica: Una línea lógica se compone de una a tres líneas físicas y se termina al oprimir la tecla RETURN o automáticamente cuando se alcanza el fin de la tercera línea. Cuando es expuesta en la pantalla, cada línea numerada en un programa BASIC se compone de una línea lógica. Cuando esté tecleando una línea que es más larga que una línea física, el cursor automáticamente irá al principio de la próxima línea física, cuando se alcanza el final de la línea física actual. Si no se pulsa la tecla RETURN, entonces ambas líneas físicas serán parte de la misma línea lógica.

Operador: Los operadores son utilizados en expresiones. Los operadores incluyen la *adición (+)*, *resta (-)*, *multiplicación (x)*, *división (/)*, *potenciación (^)*, *mayor que (>)*, *menor que (<)*, *mayor que o igual que (>=)*, *menor que o igual que (<=)*, y *no igual a (<>)*. Las palabras claves lógicas AND, NOT y OR también son operadores. Los operadores (+) y (-) también pueden ser utilizados como operadores unitarios; por ejemplo, -3. No ponga varios operadores unitarios en fila; por ejemplo, --3, puesto que el ordenador lo interpretará incorrectamente.

Línea Física: Es una línea de caracteres expuestos en una pantalla.

Cadena: Una cadena es un grupo de caracteres encerrados entre comillas. "ABACADABRA", "ATARI HACE GENIALES ORDENADORES" y "1234567890" son ejemplos de cadenas. Una cadena es muy parecida a una constante, y como ella, también puede ser almacenada en una variable. Una variable en cadena es diferente, en que su nombre debe terminar con el carácter: \$. Por ejemplo, la cadena "ATARI 800" puede ser asignada a una variable llamada A\$ utilizando LET así:

10 LET A\$="ATARI 800"

(Obsérvese las comillas)

10 A\$="ATARI 800"

(El LET es opcional, las comillas ¡NO!)

Las comillas no pueden ser utilizadas dentro de una cadena. Sin embargo, la comilla final puede ser omitida si es el último carácter en una línea lógica. (Ver Sección 7 CADENAS.)

Variables: Una variable es el nombre para una cantidad numérica o de otra clase que puede (o no) cambiar. Los nombres de las variables pueden ser de hasta 120 caracteres de longitud. Sin embargo, un nombre de una variable debe comenzar con una letra, y únicamente puede consistir de letras mayúsculas y dígitos numéricos. Se aconseja NO utilizar una palabra clave como un nombre de la variable o como la primera parte de un nombre de una variable puesto que, puede ser mal interpretada. Ejemplos de almacenamiento de un valor en una variable son:

```
LET C123DVB=1.234
LET VARIABLE112=267.543
LET A=1
LET F5TH=6.5
LET ESTENUM=59.809
```

Nota: El LET es opcional y puede ser omitido.

Límite del Nombre de una Variable: El BASIC ATARI limita, a quien lo utiliza, a 128 nombres de las variables. Para evitar este problema, utilice elementos individuales de un vector (array) o de una matriz en lugar de tener los nombres de las variables separadas. El BASIC guarda todas las referencias de una variable que ha sido suprimida de un programa, y el nombre aún permanece en la tabla de nombres de variables. Si la pantalla expone el ERROR -4, utilice el siguiente procedimiento para hacer lugar para los nuevos nombres de las variables:

```
LIST nombre de fichero
NEW
ENTER nombre de fichero
```

El LIST nom. de fichero escribe la versión no señalizada del programa en un disco o cassette. El NEW despeja el programa y las áreas de la tabla. Entonces el programa es re-ingresado, re-señalizado y una nueva tabla de variables queda construida. La versión señalizada es el formato interno del BASIC ATARI. Las versiones no señalizadas son del ATASCII, que es la versión expuesta en la pantalla.

Vectores: Un vector es un conjunto de posiciones donde se pueden almacenar datos para una futura utilización. A cada una de estas posiciones se la llama elemento del vector (a éste se le da un nombre de variable para referirse a él). Por ejemplo, podemos definir el "Vector A" para que contenga 6 ele-

mentos. Se refieren a estos elementos por medio de la utilización de variables suscritas como pueden ser A(2), A(3), A(4), etc. Un número puede ser almacenado en cada elemento. Esto puede ser llevado a cabo elemento por elemento utilizando la declaración LET), o como parte de un lazo FOR/NEXT (Ver Sección 8).

Nota: Nunca deje espacios entre el número de elementos entre paréntesis y el nombre del vector.

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

ANOTACIONES ESPECIALES UTILIZADAS EN ESTE MANUAL

Formato de Línea: El formato de una línea en un programa de BASIC incluye un número de línea (abreviado a: lineno) al principio de la línea, seguido por una palabra clave de declaración, seguido por el cuerpo de la declaración y terminado con una orden de finalización de la línea (la tecla RETURN). En un programa real, los cuatro elementos pueden aparecer así:

DECLARACION

N.º de Línea	Palabra clave	Cuerpo	Finalización
100	PRINT	A/X*(Z+4.567)	RETURN

Varias declaraciones pueden ser escritas en la misma línea siempre que estén separadas por dos puntos (:). Ver IF/THEN en la Sección 5 y 11.

Letras Mayúsculas: Las palabras clave (instrucciones, comandos, etc.) han de ser escritas en letras mayúsculas exactamente como están impresas en este texto. Los caracteres en video-inverso no funcionarán excepto en el caso de la orden RUN. Aquí hay varios ejemplos:

PRINT INPUT LIST END GOTO GOSUB FOR NEXT IF

Letras Minúsculas: En este manual, las letras en minúsculas se utilizan para denotar las distintas clases de artículos que pueden ser utilizados en un programa, tal como variables (var), expresiones (exp), y parecidos. Las abreviaturas que se utilizan para estas clases de artículos se demuestran en la Tabla 1.1.

Artículos en Corchetes: Los corchetes, [], contienen artículos opcionales que pueden ser utilizados, pero que no son necesarios.

Si el artículo encerrado en corchetes es seguido por tres puntos [exp, ...], esto quiere decir que cualquier número de expresiones pueden ser tecleadas, pero que ninguna es necesaria.

Elementos apilados verticalmente en llaves: Los elementos apilados verticalmente en llaves indican que cualquiera de los elementos apilados pueden ser utilizados, pero únicamente uno a la vez es permitido. En el ejemplo de abajo, teclee el GOTO ó el GOSUB.

$$100 \left\{ \begin{array}{l} \text{GOTO} \\ \text{GOSUB} \end{array} \right\} 200$$

Abreviaturas de órdenes en encabezamientos: Si una orden o una declaración tiene asociado con ello una abreviatura, la abreviatura se pone siguiendo al nombre de la orden en el encabezamiento; por ejemplo, LIST (L.).

ABREVIATURAS UTILIZADAS EN ESTE MANUAL

La siguiente tabla explica las abreviaturas utilizadas a lo largo de este manual.

TABLA 1.1. ABREVIATURAS

avar	Variable Aritmética: Es la ubicación en donde un valor numérico es almacenado. Los nombres de las variables pueden tener desde 1 hasta 120 caracteres alfanuméricos, pero deben comenzar con un carácter alfabético, y todos los caracteres alfa deben ser no-invertidos y en letras mayúsculas.
svar	Variable en Cadena: Es la ubicación en donde una cadena de caracteres puede ser almacenada. Las mismas reglas para nombres como las del avar, son aplicadas, excepto que el último carácter del nombre de la variable tiene que ser un \$. Las variables en cadena pueden ser suscritas. Ver la Sección 7, CADENAS.
mvar	Variable de matriz: Es también llamada una variable suscrita. Es un elemento de un vector o de una matriz. El nombre de un vector o de una matriz puede ser cualquier nombre legal de una variable tal como A, X, Y, ZIP ó K. La variable suscrita (nombre para elemento particular) empieza con la variable de la matriz, y luego utiliza un nú-


mero, variable o expresión en paréntesis inmediatamente siguiendo la variable de colección o de matriz. Por ejemplo, A (FILA), A (1) ó A (X + 1).


- var** Variable: Es cualquier variable. Puede ser mvar, avar ó svar.
- aop** Operador aritmético.
- lop** Operador lógico.
- aexp** Expresión aritmética: Está generalmente compuesta de una variable, función, constante, o dos expresiones aritméticas separadas por un operador aritmético.
- lexp** Expresión lógica: Está generalmente compuesta de dos expresiones aritméticas o en cadena separadas por un operador lógico. Tal expresión se evalúa a un 1 (verdadero lógico) o a un 0 (falso lógico).
Por ejemplo, la expresión $1 < 2$ se evalúa al valor 1 (verdadero) mientras que la expresión "LIMON" = "NARANJA" se evalúa 0 (falso) puesto que las dos cadenas no son iguales.
- sexp** Expresión en cadena: Puede ser una variable de cadena (A\$), una cadena literal o constante ("LIMON") o una función que "devuelve" una cadena (STR\$(123)).
- exp** Esta es cualquier expresión, sea sexp ó aexp.
- lineno** Número de línea: Es una constante que identifica una línea particular de programa en un programa BASIC. Tiene que ser cualquier número entero del 0 al 32767. La enumeración de la línea determina el orden de la ejecución del programa.
- adata** Datos ATASCII: Es cualquier carácter ATASCII excluyendo las comas y el regreso del carro. (Ver Apéndice C.)
- filespec** Especificación del fichero: Es una expresión en cadena que se refiere a un dispositivo como es el teclado o un fichero de disco. Este contiene información acerca del tipo de dispositivo I/O, su número, dos puntos, un nombre opcional del fichero y un extensor opcional del nombre del fichero. (Ver, OPEN, en la Sección 5.)
Ejemplar filespect: "D1:NATALIA.ED".


MODOS OPERATIVOS

- Modo Directo:** Este no utiliza números de línea y ejecuta instrucciones inmediatamente después que la tecla **RETURN** es oprimida.
- Modo Diferido:** Este utiliza números de líneas y retrasa la ejecución de instrucciones hasta que la orden **RUN** es tecleada.
- Modo Ejecutar:** Este es a veces llamado Modo **RUN**. Después que una orden **RUN** es tecleada, cada línea del programa es procesada y ejecutada.

TECLAS DE FUNCIONES ESPECIALES

 Tecla de video-inverso: o "ATARI LOGO KEY". Oprimiendo esta tecla hace que el texto en la pantalla se invierta (texto oscuro-fondo claro). Oprima la tecla por segunda vez para volver al texto normal.

 Tecla de Minúsculas: Oprimiendo esta tecla cambia los caracteres de la pantalla de mayúsculas a minúsculas. Para volver los caracteres a mayúsculas, oprima la tecla **CAPS**

 Tecla de **ESCAPE**: Oprimiendo esta tecla hace que una orden sea entrada en un programa para su ejecución posterior.

Ejemplo: Para despejar la pantalla, teclearía:

```
10 PRINT "ESC CTRL CLEAR"
```

y oprima la tecla **RETURN**.

El **ESCAPE** es también utilizado en conjunción con otras teclas para imprimir caracteres especiales de control gráfico. Ver el Apéndice F y la contratapa posterior para ver las teclas específicas y la representación de sus caracteres en la pantalla.

BREAK

Tecla de **BREAK**: Oprimiendo esta tecla durante la ejecución del programa hace que la ejecución se detenga. La ejecución puede ser recomenzada escribiendo **CONT** seguido por la opresión de la tecla **RETURN**.

RESET

Tecla para **Recomponer el sistema**: Esta es similar al **BREAK** en que al primirlo detiene la ejecución del programa. También devuelve lo expuesto en la pantalla al Modo Gráfico 0, despeja la pantalla y devuelve los márgenes y otras variables a su valor de omisión.

SET-CLR-TAB

Tecla **TAB**: Oprima la tecla **SHIFT** y la tecla **SET-CLR-TAB** simultáneamente para fijar un sangrado. Para despejar un sangrado, oprima las teclas **CTRL** y **SET-CLR-TAB** simultáneamente. Utilizada sola la tecla **SET-CLR-TAB** avanza el cursor a la próxima posición del sangrado. En el Modo Diferido, fije y despeje los sangrados procediendo lo de arriba con un número de línea, la orden **PRINT**, una comilla y oprima la tecla **ESC**.

Ejemplos:

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

Las fijaciones de los sangrados ausentes están situados en las columnas 7, 15, 23, 31 y 39.

INSERT

Tecla de **Introducción**: Oprima las teclas **SHIFT** e **INSERT** simultáneamente para introducir una línea. Para introducir un carácter, oprima la tecla **CTRL** y la tecla **INSERT** simultáneamente.

DELETE BACK S

Tecla de **Supresión**: Oprima las teclas **SHIFT** y **DELETE** simultáneamente para suprimir una línea. Para suprimir un carácter oprima las teclas **CTRL** y **DELETE** simultáneamente.

DELETE BACK S

Tecla de **Retroceso**: Oprimiendo esta tecla ubica al carácter a la izquierda del cursor con un espacio y mueve el cursor un espacio hacia atrás.

CLEAR

Tecla de **Despeje**: Oprimiendo esta tecla mientras sostiene las teclas **SHIFT** o **CTRL** hace que la pantalla se despeje totalmente y sitúa al cursor en la esquina izquierda superior.

RETURN

Tecla de Regreso: Es un finalizador para indicar el fin de una línea del BASIC. Oprimiendo esta tecla hace que una línea enumerada sea interpretada y agregada al RAM del programa BASIC. Una línea no enumerada (en Modo Directo) es interpretada y ejecutada inmediatamente. Cualquier variable es ubicada en una tabla de variables.

OPERADORES ARITMETICOS

El Sistema del Ordenador Personal ATARI utiliza 5 operadores aritméticos:

- + Adición (también suma unitaria; ej., 5)
- Sustracción (también resta unitaria; ej., - 5)
- * Multiplicación
- / División
- ^ Potenciación

OPERADORES LOGICOS

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

- AND AND Lógico
- OR OR Lógico

Ejemplo:

10 IF A = 12 AND T = 0 THEN PRINT "BIEN" *Ambas expresiones deben ser verdaderas antes que el "BIEN" se imprima.*

10 A = (C > 1) AND (N < 1) *Si ambas expresiones son verdadera, A = 1 ó si no A = 0.*

10 A = (C + 1) OR (N - 1) *Si cualquiera expresión es verdadera, A = 1; si no A = 0.*

10 A = NOT(C + 1) *Si la expresión es falsa, A = 1; si no A = 0.*

El resto de los operadores binarios están relacionados.

- > La primera expresión es mayor que la segunda expresión.
- < La primera expresión es menor que la segunda expresión.
- = Las expresiones se igualan.
- <= La primera expresión es menor o igual a la segunda expresión.
- >= La primera expresión es mayor o igual a la segunda expresión.
- <> Las dos expresiones no se igualan.

Estos operadores se utilizan más frecuentemente en declaraciones IF/THEN y en aritmética lógica.

PRECEDENCIA DEL OPERADOR

Las operaciones que se encuentran en el conjunto más central de los paréntesis se realizan primero, y preceden al siguiente nivel. Cuando los conjuntos de paréntesis se encuentran encerrados en otro conjunto, se dicen que están "anidados". Las operaciones en el mismo nivel de nido se realizan en el siguiente orden:

Precedencia Mayor <, >, =, <=, >=, <>

Los operadores en relación utilizados en expresiones en cadena tienen la misma precedencia y se realizan de izquierda a derecha.

-

Resta unitaria.

*, /

La multiplicación y la división tienen el mismo nivel de precedencia y se realizan de izquierda a derecha.

+, -

La adición y la sustracción tienen el mismo nivel de precedencia y se realizan de izquierda a derecha.

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

Las operaciones en relación en expresiones numéricas tienen el mismo nivel de precedencia y se realizan de izquierda a derecha.

	NOT	Operador unitario.
	AND	AND lógico.
Precedencia Menor	OR	OR lógico.

FUNCIONES INCORPORADAS La sección titulada BIBLIOTECA DE FUNCIONES explica las funciones especiales y aritméticas incorporadas en el BASIC ATARI.

GRAFICOS Los gráficos ATARI incluyen 16 modos gráficos. Las órdenes han sido desarrolladas para permitir la máxima flexibilidad en la variedad de patrones y selección de colores. La Sección 9 explica cada orden y da ejemplos de muchas maneras de utilizar cada una de ellas.

SONIDOS Y CONTROLADORES DE JUEGOS El Ordenador Personal ATARI es capaz de emitir una gran variedad de sonidos incluyendo explosiones simuladas, música electrónica, y "rasberries" (un sonido despectivo, ofensivo). La Sección 10 define las órdenes para utilizar la función SOUND y para controlar la paleta (paddle), palanca de mando (joystick) y los controles del teclado.

REPETICION DE TECLADO El Sistema de Ordenadores Personales ATARI permiten a quien lo utilice escribir una tecla de antemano. Si quien lo utiliza oprime y sostiene cualquier tecla, ésta comenzará a repetirse después de 1/2 segundo.

MENSAJES DE ERROR Si se comete un error de entrada de datos, lo expuesto en la pantalla manifiesta la línea re-imprimida precedida por el mensaje ERROR —y se resalta el carácter erróneo. Después de corregir el carácter en la línea original, suprime la línea que contiene el error antes de oprimir la tecla RETURN. El Apéndice B contiene una lista de todos los mensajes de error con sus correspondientes definiciones.

2 ORDENES

Siempre que el cursor es expuesto en la pantalla, el ordenador está preparado para recibir entradas. Teclee la orden (en Modo Directo), y oprima la tecla **RETURN**. Esta sección describe las órdenes utilizadas para despejar la memoria del ordenador, y otras órdenes útiles de control.

Las órdenes explicadas en esta sección son las siguientes:

BYE CONT END LET LIST NEW REM RUN STOP

BYE (B.)

Formato: **BYE**

Ejemplo: **BYE**

La función de la orden **BYE** es la de salir del **BASIC** y poner al ordenador en el modo Autoensayo (Self Test). Para volver al **BASIC**, oprima la tecla **RESET**.

CONT (CON.)

Formato: **CONT**

Ejemplo: **CONT**

Tecleando esta orden seguida por la opresión de la tecla **RETURN** recomienza la ejecución del programa. Si se encuentra con la tecla **BREAK** o las órdenes **STOP** ó **END**, el programa se detendrá hasta que se teclee **CONT** y se oprima la tecla **RETURN**. La ejecución recomienza en el próximo número de línea secuencial, seguida de la declaración en donde se detuvo el programa.

Nota: Si la declaración en donde se detuvo el programa tiene otras órdenes en la misma línea enumerada que no se ejecutaron en el momento que se oprimió la tecla **BREAK**, o las órdenes **END** ó **STOP**, no se ejecutarán. En la orden **CONT**,

la ejecución recomienza en la próxima línea enumerada. Un lazo puede ser incorrectamente ejecutado si el programa es detenido antes que el lazo complete la ejecución.

Esta orden no tiene ningún efecto en un programa en el Modo Diferido.

END

Formato: END

Ejemplo: 1000 END

Esta orden finaliza la ejecución del programa y es utilizada en el Modo Diferido. En el BASIC ATARI, un END no es necesario al final del programa. Cuando se alcanza el final del programa, el BASIC ATARI automáticamente cierra todos los ficheros y suprime los sonidos (si los hay). El END puede también ser utilizado para cerrar los ficheros y suprimir los sonidos (en el Modo Directo).

LET (LE.)

Formato: [LET] var = exp

Ejemplo: LET X = 3.124 * 16
LET X = 2

Esta declaración es opcional cuando está definiendo las variables. Tranquilamente se puede omitir el LET de la declaración. Se puede utilizar, sin embargo, para fijar un nombre de una variable igual a un valor.

LIST (L.)

Formato: LIST [lineno [, lineno]]
LIST [nombre de fichero [, lineno , lineno]]]

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

Esta orden hace que el ordenador exponga la versión de origen de todas las líneas que actualmente se encuentran en la memoria si la orden es tecleada sin los números de líneas, o para exponer una línea o líneas especificada(s). Por ejemplo, LIST 10,100, seguido por la opresión de la tecla RETURN, visualiza desde la línea 10 hasta la línea 100 inclusive. Si el usuario no ha enumerado las líneas de entrada en orden, la orden LIST lo hará.

Teclear L."P imprimirá el programa resident en RAM en la impresora.

El LIST puede ser utilizado en el Modo Diferido como parte de una rutina que atrapa errores (Ver TRAP, en la Sección 4).

La orden LIST también se utiliza cuando se graban programas en cassettes. Se utiliza el segundo formato y el nombre del fichero es tecleado. (Ver la Sección 5 para mayor detalle sobre los dispositivos periféricos.) Si se quiere inscribir el programa entero en la cinta no hace falta especificar los números de las líneas.

Ejemplo: LIST "CL"
1000 LIST "CL"

NEW

Formato: NEW

Ejemplo: NEW

Esta orden borra el programa almacenado en el RAM. Por lo tanto, antes de escribir NEW, teclee SAVE ó CSAVE para así salvar cualquier programa que ha de ser recuperado y utilizado luego. La orden NEW despeja la tabla de símbolos internos del BASIC para que ninguna colecciones o cadenas sean definidos. Es utilizado en el Modo Directo.

REM (R. o la tecla SPACE)

Formato: REM texto

Ejemplo: 10 REM Rutina para calcular X

Esta orden y el texto que la sigue son para la información de quien lo utilice únicamente. Es ignorado por el ordenador. Sin embargo, se incluye en un LIST junto con las otras líneas enumeradas. Cualquier declaración en la misma línea numerada que ocurre después de una declaración REM se ignorará.

RUN (RU.)

Formato: RUN [nombre de fichero]

Ejemplos: RUN
RUN "D:MENU"

Esta orden acciona el ordenador para comenzar la ejecución de un programa. Si ningún nombre de fichero es especificado, el actual programa residente-RAM comienza la ejecución. Si se incluye un nombre de fichero, el ordenador recoge el programa especificado y señalado del fichero especificado y lo ejecuta.

El RUN se puede utilizar en el Modo Diferido.

Ejemplo: 10 PRINT "vez tras vez"
20 RUN

Escriba RUN y oprima la tecla RETURN. Para finalizar, oprima la tecla BREAK.

Para empezar la ejecución en un punto que no sea la primera línea, escriba: GOTO seguido del número de línea en que quiera que empiece el programa, luego oprima RETURN.

STOP (STO.)

Formato: STOP.

Ejemplo: 100 STOP.

Cuando la orden STOP es ejecutada en un programa, el BASIC escribe el mensaje STOPPED AT LINE _____, termina la ejecución y vuelve al modo directo. La orden STOP no cierra archivos, ni desconecta sonidos, de este modo puede continuarse la ejecución del programa tecleando CONT RETURN.

3 CARACTERISTICAS DE EDITADO

En adición a las teclas de funciones descrita en la Sección 1, hay teclas de control de cursor que permiten capacidades inmediatas de corrección. Estas teclas son utilizadas en conjunción con la tecla **SHIFT** o la tecla **CTRL**.

Las siguientes funciones de teclas se describirán en esta sección:

CONTROL	CONTROL INSERT	CONTROL 1
SHIFT	CONTROL DELETE	CONTROL 2
CONTROL ↑	CONTROL INSERT	CONTROL 3
CONTROL ↓	CONTROL DELETE	BREAK
CONTROL →	SHIFT CAPS	CONTROL
CONTROL ←		

EDITADO DE PANTALLA

El teclado y lo expuesto están lógicamente combinados para un modo de operación conocido como editado de pantalla. Cada vez que un cambio se completa en la pantalla, la tecla **RETURN** debe ser oprimida. De otra manera el cambio no se efectúa en el programa del RAM.

Ejemplo:

```
10 REM OPRIMA RETURN DESPUES DE EDITADO EN PANTALLA
20 PRINT :PRINT
30 PRINT "ESTA ES LA LINEA NO. 1 EN LA PANTALLA."
```

Para suprimir la línea 20 del programa, teclee el número de línea y luego oprima la tecla **RETURN**. Simplemente suprimiendo la línea de la pantalla **no** lo suprime del programa

La pantalla y el teclado utilizados como dispositivos de entrada y salida (I/O) se detallan en la Sección 5.

CONTROL

Tecla de Control: Pulsando esta tecla en conjunción con las teclas de las flechas direccionales ocasionan las funciones del control del cursor, las cuales permiten, a quien lo utiliza, mover el cursor a cualquier parte de la pantalla sin cambiar ninguno de los caracteres ya existentes en la pantalla. Otras combinaciones de teclas controlan la fijación y la supresión de los sangrados, el cese y recomienzo de las listas de programas y los símbolos de control gráfico. (Pulsando una tecla mientras que se sostiene la tecla **CTRL** producirá el símbolo superior de la izquierda en aquellas teclas que tienen tres funciones.)

SHIFT

Tecla de Mayúsculas: Esta tecla es utilizada en conjunción con las teclas numéricas para exponer los símbolos que están en la mitad superior de esas teclas. También es utilizada en conjunción con otras teclas para introducir o suprimir líneas, volver a una exposición de letras mayúsculas normales, y para visualizar los símbolos de funciones arriba de los operadores de resta, igualdad, adición y multiplicación, como también los corchetes, [], y los signos de interrogación, ?.

TECLAS DE DOBLE FUNCION

Teclas de control del cursor

- | | | |
|---------|---|---|
| CONTROL | ⏮ | Mueve el cursor una línea física hacia arriba, sin cambiar el programa ni el listado. |
| CONTROL | ⏪ | Mueve el cursor un espacio hacia la derecha sin cambiar el programa ni el listado. |
| CONTROL | ⏩ | Mueve el cursor una línea física hacia abajo, sin cambiar el programa ni el listado. |
| CONTROL | ⏭ | Mueve el cursor un espacio hacia la izquierda sin cambiar el programa ni el listado. |

Como las otras teclas del teclado ATARI, apretando las teclas de control del cursor, durando más de 1/2 segundo, para que la tecla se repita.

Teclas utilizadas con Control

CONTROL	INSERT	Inserta un espacio.
CONTROL	DELETE	Borra un espacio con carácter.
CONTROL	1	Para temporalmente y devuelve la representación en pantalla sin "romper" el programa.
CONTROL	2	Hace sonar un pitido.
CONTROL	3	Indica final de archivo.

Teclas utilizadas con SHIFT

SHIFT	INSERT	Introduce una línea física.
SHIFT	DELETE	Suprime una línea física.
SHIFT	CAPS	Devuelve lo expuesto en la pantalla a caracteres alfabéticos en mayúsculas.

Teclas de Funciones Especiales

BREAK	Esta tecla detiene la ejecución o lista del programa, imprime la palabra READY en la pantalla y lo expone al cursor.
--------------	---

ESC	Esta tecla permite a las órdenes normalmente utilizadas en Modo Directo de ser puestas en el Modo Diferido; ej., en el Modo Directo las teclas CTRL y CLEAR despejan lo expuesto en la pantalla. Para despejar la pantalla en el Modo Diferido, teclee lo siguiente después del número de la línea del programa. Oprima la tecla ESC y luego oprima las teclas CTRL y CLEAR juntas.
------------	--

PRINT "ESC CTRL CLEAR"

4 DECLARACIONES DE PROGRAMA

Esta sección explica las órdenes asociadas con lazos, saltos condicionales y no condicionales, trampas de errores y subrutinas y sus recuperaciones. También explica la manera de adquirir datos y las órdenes opcionales que se utilizan para definir las variables.

Las siguientes órdenes están descritas en esta sección:

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

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

Formato: FOR avar = aexp1 TO aexp2 (STEP aexp3)
NEXT avar

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

Esta orden fija un lazo y determina cuántas veces ese lazo es ejecutado. La variable del lazo (avar) es marcado con iniciales al valor aexp1. Cada vez que se encuentra con la declaración NEXT avar, la variable del lazo es incrementada por el aexp3 en la declaración STEP. Los aexp3 pueden ser números enteros positivos o negativos, o decimales o números fraccionarios. Si no hay *ninguna* orden STEP aexp3, el lazo se incrementa por 1. Cuando el lazo complete el límite definido por el aexp2, se detiene y el programa sigue a la declaración que inmediatamente sigue la declaración NEXT; se puede encontrar en la misma línea o en la próxima línea secuencial.

Los lazos se pueden "anidar", uno dentro del otro. En este caso, el lazo más

interno se completa antes de volver al lazo externo. El siguiente ejemplo ilustra un programa de un lazo anidado.

```
10 FOR X=1 TO 3
20 PRINT "LAZO EXTERNO"
30 Z=0
40 Z=Z+2
50 FOR Y=1 TO 5 STEP Z
60 PRINT "LAZO INTERNO"
70 NEXT Y
80 NEXT X
90 END
```

Figura 4-1. Lazo anidado.

En la Figura 4-1, el lazo externo completará tres ejecuciones ($X = 1$ TO 3). Sin embargo, antes que este primer lazo llegue a su declaración NEXT X, el programa da control al lazo interno. Observe que a la declaración NEXT de el lazo interno debe preceder la declaración NEXT para el lazo externo. En el ejemplo, el número de ejecuciones del lazo interno es determinado por la declaración STEP (STEP Z). En este caso Z ha sido definido como 0, y después redefinido como $Z + 2$. Utilizando estos datos, el ordenador debe completar tres ejecuciones a través del lazo interno antes de volver al lazo externo. El aexp3 en la declaración STEP podría haber sido definido como el valor numérico 2.

La realización del programa se ilustra en la Figura 4-2.

El remitente para los lazos se sitúa en un grupo especial de direcciones de memoria, se refiere a ello como "pila" "STACK". La información es "empujada" encima de la pila y cuando es utilizada, la información es "saltada" fuera de la pila.

```
LAZO EXTERNO
  LAZO INTERNO
  LAZO INTERNO
  LAZO INTERNO
LAZO EXTERNO
  LAZO INTERNO
  LAZO INTERNO
  LAZO INTERNO
LAZO EXTERNO
  LAZO INTERNO
  LAZO INTERNO
  LAZO INTERNO
```

Figura 4-2. Ejecución de lazo anidado.

GOSUB (GOS.)
RETURN (RET.)

Formato: GOSUB lineno
lineno
RETURN

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

Una subrutina es un programa o una rutina utilizada para computar un cierto valor, etc. Generalmente se utiliza cuando una operación debe ser reemplazada varias veces dentro de una secuencia de programa utilizando los mismos o diferentes valores. Esta orden permite a quien lo utilice a "llamar" a la subrutina, si es necesario. La última línea de una subrutina debe contener una declaración RETURN. La declaración RETURN vuelve a la línea física *siguiendo* la declaración GOSUB.

Como la orden FOR/NEXT precedente, la orden GOSUB/RETURN utiliza a la pila para su remitente. Si no se le permite a la subrutina completarse normalmente; ej., un GOTO lineno antes de un RETURN, la dirección GOSUB tendrá que ser "saltada" fuera de la pila (Ver POP) o podría causar futuros errores.

Generalmente, una subrutina puede hacer cualquier cosa que se pueda hacer en un programa. Se utiliza para ahorrar memoria y tiempo de entradas de programas y para facilitar la lectura y la "puesta a punto" de los programas".

Para prevenir un desencadenamiento accidental de una subrutina (que normalmente sigue al programa principal), ponga una declaración END precediendo la subrutina. El siguiente programa demuestra el uso de las subrutinas.

```
10 PRINT "}" (Despejar Pantalla)
20 REM USO EJEMPLAR EL 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 programa.

En el programa de arriba, la subrutina, comenzando por la línea 1000, es llamada tres veces para computar e imprimir los distintos valores de X e Y. La figura 4-4 ilustra los resultados al ejecutar este programa.

400	300
480	360
200	150

Figura 4-4. Ejecución de programa GOSUB/RETURN.

GOTO (G.)

Formato: { GOTO }
 { GO TO } aexp

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

La orden GOTO es una declaración de salto incondicional tal como la orden GOSUB. Ambos inmediatamente transfieren el control del programa al número de la línea seguida o a una expresión arbitraria. No obstante, utilizar cualquier cosa que no sea una constante, dificultará la re-enumeración del programa. Si el número de la línea seguida es inexistente da por resultado un error. Cualquier declaración GOTO que se ramifica a una línea precedente puede resultar en un lazo sin final. Las declaraciones que siguen a una declaración GOTO no serán ejecutadas. Observe que una declaración ramificante condicional (Ver IF/THEN) se puede utilizar para salir de un lazo GOTO. El siguiente programa ilustra dos utilizaciones de la orden GOTO.

```

10 PRINT
20 PRINT: PRINT "UNO"
30 PRINT "DOS"
40 PRINT "TRES"
50 PRINT "CUATRO"
60 PRINT "CINCO"
70 GOTO 120
80 PRINT "$$$$$$$$$$$$$"
90 PRINT ":/:/:/:/:/:/:/:/:/:/:/:/:/:/:/"
100 PRINT "?????????????????"
110 END
120 PRINT "SEIS"
130 PRINT "SIETE"

```


La declaración IF/THEN es una declaración de salto condicional. Este tipo de saltos ocurren únicamente si ciertas condiciones se reúnen. Estas condiciones pueden ser aritméticas o lógicas. Si el aexp que sigue a la declaración IF es acertada (no-cero), el programa ejecuta la parte THEN de la declaración. Si, sin embargo, el aexp es falso (un 0 lógico), el resto de la declaración es ignorada y el control del programa pasa a la próxima línea enumerada.

En el formato, IF aexp THEN lineno, el lineno debe ser una constante, no una expresión y que especifique el número de la línea a la cual ir si la expresión es verdadera. Si varias declaraciones ocurren después del THEN, separadas por dos puntos, entonces serán ejecutadas únicamente si la expresión es verdadera. Varias declaraciones IF se pueden anidar en la misma línea. Por ejemplo:

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

Las declaraciones R = 9: GOTO 200 se ejecutarán únicamente si X = 5 e Y = 3 se ejecutarán si X = 5.

El siguiente programa demuestra la declaración IF/THEN:

```
5 GRAPHICS 0:?:? :?"
10 ? :? "TECLEE A"; :INPUT A
20 IF A=1 THEN 40:REM DECLARACIONES
  MULTIPLES AQUI NUNCA SERAN EJECU-
  TADAS!!
30 ? :? "A NO ES 1. EJECUCION CONTINUA
  AQUI CUANDO LA EXPRESION ES FALSA."
40 IF A=1 THEN ? :? "A=1":? "SI, ES REAL-
  MENTE 1.":REM DECLARACIONES MULTI-
  PLES AQUI UNICAMENTE SERAN EJECUTA-
  DAS SI A=1!!
50 ? :? "EJECUCION CONTINUA AQUI SI A<>1
  O DESPUES DE 'SI, ES REALMENTE 1' ES
  EXPUESTO
60 GOTO 10
```

Figura 4-7. Programa IF/THEN.

```
TECLEE A
A NO ES 1. EJECUCION CONTINUA AQUI
CUANDO LA EXPRESION ES FALSA.
EJECUCION CONTINUA AQUI SI A<>1 O DES-
PUES DE QUE 'SI, ES REALMENTE 1' ES EX-
PUESTO.
```

(Entrada 2)

TECLEE A

(Entrada 1)

A=1

'SI, ES REALMENTE 1'

EJECUCION CONTINUA AQUI SE A<>1 O DES-
PUES QUE 'SI, ES REALMENTE 1' ES EX-
PUESTO.

TECLEE A

Figura 4-8. Ejecución de Programa IF/THEN.

ON/GOSUB/RETURN
ON/GOTO

Formato: ON aexp { GOTO } lineno [,lineno...]
 { GOSUB }

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 ser abreviados.

Estas dos declaraciones también son declaraciones de salto condicional como la declaración IF/THEN. No obstante, estas dos son más poderosas. El aexp debe evaluarse a un número positivo, el cual es entonces redondeado al valor del número entero positivo más cercano hasta 255. Si el número que resulta es 2, el control del programa pasa al segundo lineno de la lista, y continuando así sucesivamente. Si el número que resulta es 0 ó es mayor que el número de lineos en la lista, las condiciones no se reúnen y el control del programa pasa a la declaración siguiente, la cual puede o no estar situada en la misma línea. Con el ON/GOSUB, la subrutina seleccionada es ejecutada y entonces el control pasa a la próxima declaración.

La siguiente rutina demuestra la declaración ON/GOTO.

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



```

300 PRINT "AHORA TRABAJANDO EN LA LINEA 300": GOTO 10
400 PRINT "AHORA TRABAJANDO EN LA LINEA 400": GOTO 10
500 PRINT "AHORA TRABAJANDO EN LA LINEA 500": GOTO 10

```

Figura 4-9. Listado de programa ON/GOTO.

Cuando el programa se ejecuta, se parece a lo siguiente:

```

AHORA 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
COMPLETO

```

Figura 4-10. Ejecución de programa ON/GOTO.

POP

Formato: POP

Ejemplo: 1000 POP

En la descripción de la declaración FOR/NEXT, la pila fue definida como un grupo de direcciones de memoria reservadas para remitentes. La entrada en la parte superior de la pila controla el número de lazos que serán ejecutados y la línea RETURN seguida para un GOSUB. Si una subrutina no es finalizada con una declaración RETURN, la ubicación superior de la memoria de la pila está aún cargada con algunos números. Si se ejecuta otra declaración GOSUB, esa posición debe ser despejada. Para preparar la pila para una nueva declaración GOSUB, utilice un POP para despejar los datos de la posición superior de la pila.

La orden POP debe ser utilizada de acuerdo a las siguientes reglas:

1. Debe estar en el camino de ejecución del programa.
2. Debe seguir la ejecución de cualquier declaración GOSUB que no vuelva al programa principal por medio de una declaración RETURN.

El siguiente ejemplo demuestra la utilización de una orden POP con un GOSUB cuando la declaración RETURN no es ejecutada.

```

10 GOSUB 1000
15 REM LINEA 20 NO SERA EJECUTADA
20 PRINT "REGRESO NORMAL DEVUELVE ESTE MENSAJE"

```

```

30 PRINT "REGRESO ANORMAL DEVUELVE ESTE MENSAJE"
40 POP
999 END
1000 PRINT "AHORA EJECUTANDO SUBROUTINA"
1010 GOTO 30
1020 RETURN

```

Figura 4-11. Declaración GOSUB con POP.

RESTORE (RES.)

Formato: RESTORE aexp
Ejemplo: 100 RESTORE

El Sistema de Ordenadores Personales ATARI contiene un indicador interno que mantiene orden, o sea, lleva control de la declaración DATA que se debe leer próximamente. Utilizada sin el aexp opcional, la declaración RESTORE refija el indicador al primer artículo DATA en el programa. Utilizada con el aexp opcional, la declaración RESTORE fija al indicador al primer primer artículo DATA especificado por el valor de aexp. Esta declaración permite la utilización repetitiva de los mismos datos.

```

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

```

Figura 4-12. Listado de programa RESTORE.

En la primera ejecución a través del lazo, A será 30 y B será 30, por lo tanto, el total de la línea 50 imprimirá EL TOTAL IGUAL A 60, pero en la segunda ejecución, A será igual a 15 y B será igual a 30, por lo de la declaración RESTORE, aún será igual a 30. Por lo tanto, la declaración PRINT en la línea 50 expondrá EL TOTAL IGUAL A 45.

TRAP (T.)

Formato: TRAP aexp

Ejemplo: 100 TRAP 120

La orden TRAP es utilizada para dirigir el programa a un número de línea especificado si un error es detectado. Sin una declaración TRAP el programa cesa de ejecutar cuando se encuentra con un error y expone un mensaje de error en la pantalla.

La declaración TRAP trabaja sobre cualquier error que pueda ocurrir después que haya sido ejecutada, pero una vez que un error ha sido detectado y atrapado, es necesario refijar la trampa con otra orden TRAP. Esta orden TRAP puede ser situada al principio de la sección del código que maneja las entradas del teclado a fin de que el TRAP sea refijado después de cada error. El PEEK (195) le dará un mensaje de error (ver Apéndice B). $256 * \text{PEEK}(187) + \text{PEEK}(186)$ le dará el número de la línea donde ocurrió el error. El TRAP se puede despejar al ejecutar una declaración TRAP con un aexp cuyo valor es de 32676 a 65535, por ejemplo, 40000.

5 ORDENES Y DISPOSITIVOS DE ENTRADA/SALIDA (I/O)

Esta sección explica los dispositivos de entrada y salida y cómo se mueven los datos entre ellos. Las órdenes que se explican en esta sección son aquellas que permiten el acceso a los dispositivos de entrada y salida. Las órdenes de entrada son aquellas asociadas con conseguir la entrada de datos en el RAM y los dispositivos preparados para aceptar entradas. Las órdenes de salida son aquellas asociadas con recuperar datos del RAM y los dispositivos preparados para generar salidas.

Las órdenes descritas en esta sección son:

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

DISPOSITIVOS DE ENTRADA/SALIDA

La configuración de Hardware de cada uno de los siguientes dispositivos es ilustrada en los manuales individuales que viene con cada uno de ellos. El subsistema central de entrada y salida (CIO) provisiona a quien lo utiliza con un interface único de acceso a todos los dispositivos periféricos del sistema de una manera (ampliamente) independiente. Esto significa que hay un punto de entrada único y una secuencia de llamadas independientes de dispositivos. Cada dispositivo tiene un nombre de dispositivo simbólico utilizado para identificarse; ej., **K**: para el teclado. Cada dispositivo debe ser abierto antes del acceso y cada uno debe ser asignado a un Bloc de Control de Input/Output (Entrada/Salida) (IOCB). De ahí en adelante, se refiere al dispositivo por su número de IOCB.

El BASIC ATARI contiene ocho blocs en RAM que identifican al Sistema Operativo la información que necesita para desempeñar una operación de Entrada/Salida. Esta información incluye la orden, longitud de buffer, dirección de buffer, y dos variables auxiliares de control. El BASIC ATARI pone a punto los IOCB, pero el que lo utilice debe especificar cuál IOCB se ha de utilizar. El BASIC se reserva el IOCB0 para Entrada/Salida al Editor de Pantalla, y por lo tanto, el que la utiliza no puede solicitar el IOCB0. La declaración GRAPHICS

(Ver la Sección 9) abre al IOCB6 para entrada y salida de pantalla. (Esto es la ventana Gráfica S:.) El IOCB7 es utilizado por el BASIC para las órdenes LPRINT CLOAD y CSAVE. Se puede referir al número de IOCB también como el número de dispositivo (o fichero). El 1 al 5 del IOCB es utilizado en la apertura de otros dispositivos para las operaciones de Entrada/Salida. Si el IOCB7 está siendo utilizado, éste impedirá al LPRINT o algunas de las otras declaraciones del BASIC de Entrada/Salida de ser llevadas a cabo.

Teclado (K:): Es un dispositivo de entrada únicamente. El teclado permite, a quien lo utilice, leer los datos del teclado convertidos (ATASCII) al pulsar cada tecla.

Impresora (P:): Es un dispositivo de salida únicamente. La impresora imprime los caracteres ATASCII, una línea a la vez. No reconoce ningún carácter de control.

Grabadora de Programas (C:): Es un dispositivo de Entrada/Salida. La grabadora es un dispositivo de lectura/escritura, la cual se puede utilizar como cualquiera de los dos, pero nunca en conjunto simultáneamente. El cassette tiene dos canales para propósitos de sonido y grabación del programa. El canal de audio no puede ser grabado del Sistema ATARI, pero se puede escuchar a través del altavoz de la televisión.

Unidad de Disco (D1:, D2:, D3:, D4:): Son dispositivos de Entrada/Salida. Es necesario tener más de 16K RAM en el ATARI, éste puede utilizar de una a cuatro unidades de disco. Si únicamente una unidad de disco es acoplada, no hay ninguna necesidad de agregar un número después del código simbólico del dispositivo, D.

Editor de Pantalla (E:): Es un dispositivo de Entrada/Salida. Este dispositivo utiliza el teclado y la visualización (Ver *Monitor de Televisión*) para simular un terminal de editado de pantalla. Escribiendo a este dispositivo resulta en la aparición de datos en la visualización comenzando en la posición actual del cursor. Leyendo desde este dispositivo activa el proceso de editado de pantalla y permite a quien lo utilice, entrar y corregir datos. Siempre que se oprima la tecla **RETURN**, la línea lógica entera dentro de la cual reside el cursor es seleccionada como el registro actual que ha de ser transferido por el CIO al programa de quien lo esté utilizando (Ver la Sección 9).

Monitor de Televisión (S:): Es un dispositivo de Entrada/Salida. Este dispositivo permite a quien lo utilice leer caracteres y escribirlos desde y hasta la visualización, utilizando al cursor como el dispositivo de dirección de pantalla. Ambas operaciones de texto y gráficos son sostenidas. Ver en la Sección 9 para tener una descripción completa de los modos gráficos.

Interface, RS-232 (R:): El dispositivo RS-232 le permite al Sistema ATARI ha interconectarse con los dispositivos compatibles de RS-232 tales como impresoras, terminales y trazadores. Contiene un puerto paralelo a los cuales la impresora de 80-Columnas ATARI 825 puede ser acoplada.

CLOAD (CLOA.)

Formato: CLOAD

Ejemplos: CLOAD
100 CLOAD

Esta orden se puede utilizar en el Modo Directo o Modo Diferido para cargar un programa desde la cinta del cassette al RAM para su ejecución. Al teclear CLOAD, un zumbido será oído, indicando que hay que pulsar la tecla PLAY seguido por la tecla RETURN. Sin embargo, no oprima la tecla PLAY hasta después de que haya colocado la cinta. Las instrucciones específicas para cargar un programa están contenidas en el *Manual de Grabación de Programas ATARI*

Los pasos a seguir para cargar programas de tamaño extenso están incluidos en los párrafos bajo el título de **ENCADENANDO PROGRAMAS** al final de esta sección.

CSAVE (CS.)

Formato: CSAVE

Ejemplos: CSAVE
100 CSAVE
100 CS.

Esta orden se utiliza generalmente en el Modo Directo para guardar un programa residente en RAM en una cinta. El CSAVE guarda la versión señalizada del programa. Al teclear CSAVE se oirán dos zumbidos indicando que las teclas PLAY y RECORD deben ser oprimidas seguido por la pulsación de la tecla RETURN. Sin embargo, no oprima estas teclas hasta que haya sido colocada la cinta. Es más rápido guardar un programa utilizando esta orden en vez de SAVE "C" (Ver SAVE) porque se utilizan espacios cortos de inter-grabación.

Notas: Las cintas que se guardan por medio de dos órdenes, SAVE y CSAVE, no son compatibles.

Quizás sea necesario teclear LPRINT (Ver LPRINT) antes de utilizar CSAVE. De otra forma, CSAVE no funcionará correctamente.

Para obtener instrucciones específicas acerca de cómo conectar y operar el hardware, indicar la cinta, etc., ver el *Manual de Grabación de Programas de ATARI*

DOS (DO.)

Formato: DOS

Ejemplo: DOS

La orden DOS es utilizada para ir desde el BASIC al Sistema Operativo de Disco (DOS). Si el Sistema no ha sido puesto en la memoria, el ordenador entrará al Autoensayo y el que lo utilice debe oprimir la tecla RESET para volver al Modo Directo. Si el DOS ha sido puesto en la memoria, el DOS Menu es visualizado. Para despejar el Menu DOS de la pantalla, oprima la tecla RESET. El control entonces pasa al BASIC. El control también se puede devolver al BASIC seleccionando B en el Menu DOS.

La orden DOS es utilizada generalmente en el Modo Directo; sin embargo, puede ser utilizado en un programa. Para obtener más detalles acerca de esto, ver el *Manual ATARI DOS*.

ENTER (E.)

Formato: ENTER nombre de fichero

Ejemplos: ENTER "C
ENTER "D:EJEMPLO.DOS"

Esta declaración hace que la cinta cargue un programa originalmente grabado utilizando el LIST (ver Sección 2, LIST). El programa es entrado en una forma no procesada (no señalizada), y es interpretada al recibir los datos. Cuando se termina de cargar, se puede llevar en la memoria normal. La orden ENTER también puede ser utilizada con la unidad de disco. Observe que tanto LOAD como CLOAD despejan el programa anterior de la memoria antes de cargar uno nuevo. El ENTER fusiona los viejos y nuevos programas. La declaración ENTER es utilizada en el Modo Directo.

INPUT (I.)

Formato: $\left[\begin{array}{c} \#aexp \quad \{ , \} \\ \{ ; \} \end{array} \right] \left\{ \begin{array}{c} avar \\ svar \end{array} \right\} \left[\begin{array}{c} \{ avar \} \\ \{ svar \} \end{array} \right] \dots$

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

Esta declaración requiere datos del teclado de quien lo utiliza. En la ejecución, el ordenador imprime una ? cuando el programa se encuentra con una declaración INPUT. Esta es generalmente precedida por una declaración PRINT, que informa a quien lo utiliza sobre qué tipo de información está siendo requerida.

Las variables en cadena únicamente se permiten sino no están suscritas. Las variables de matriz no se permiten.

El #aexp es opcional y es utilizado para especificar el número de fichero o dispositivo de donde se han de entrar los datos (Ver Dispositivos de Entrada/Salida). Si ningún aexp es especificado, entonces la entrada es desde el editado de pantalla (E:).

Si varias cadenas han de ser entradas desde el editado de pantalla, teclear una cadena, oprima la tecla RETURN, teclee la próxima cadena, oprima la tecla RETURN, etc. Los números aritméticos pueden ser tecleados en la misma línea separándolos con comas.

```
10 PRINT: "TECLEE 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 LOS NUMEROS ES";C  
70 END
```

Figura 5-1. Listado de programa INPUT.

LOAD (LO.)

Formato: LOAD nombre de fichero

Ejemplo: LOAD "D1:JANINE, BRY"

Esta orden es imilar al CLOAD, excepto que se puede utilizar el sistema de nombres enteros de ficheros. El LOAD utiliza espacios largos de inter-grabación en la cinta (Ver CLOAD), y utiliza la versión señalizada del programa. Cuando esté utilizando únicamente una unidad de disco, no es necesario especificar un número después de la "D" porque la ausencia es la unidad de disco 1.

LPRINT (LP.)

Formato: LPRINT [exp] [{ , } exp...]
[{ ; }]

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

Esta declaración hace que el ordenador imprima datos en la impresora en vez de en la pantalla. Puede ser utilizada en el Modo Directo o en el Modo Diferido. No requiere ningún especificador de dispositivo ni una declaración OPEN ni CLOSE. (El BASIC utiliza el IOCB7).

La lista del programa de arriba ilustra un programa que sumará cinco números tecleados por quien lo utiliza. Para imprimir un listado del programa en la impresora, ver LIST.

NOTE (N.)

Formato: NOTE # aexp, avar, avar

Ejemplo: 100 NOTE #1, X, Y

Esta orden es utilizada para almacenar el número actual del sector del disco en el primer avar y el número actual de byte dentro del sector en el segundo avar. Esta es la actual posición de lectura o escritura en el fichero especificado en donde el próximo byte que se leerá o escribirá será ubicado. La orden NOTE es utilizada cuando está escribiendo datos a un fichero de disco (Ver POINT). La información en la orden NOTE es escrita en un segundo fichero, la cual es entonces utilizada como un índice en el primer fichero.

OPEN (O.); CLOSE (CL.)

Formatos: OPEN # aexp, aexp1, aexp2, nombre de fichero
CLOSE # aexp

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

Antes que se pueda acceder al dispositivo, éste debe ser abierto. Este proceso de "apertura" une a un IOCB específico al 'manejador' apropiado del dispositivo, marca con iniciales cualquier variable de control relacionada-CIO y pasa cualquier

opción específica del dispositivo al 'manejador del mismo'. Los parámetros de la orden OPEN están definidos a continuación:

Es el carácter obligatorio que debe ser tecleado por el que lo utilice.

aexp Es el número del fichero o de referencia del IOCB a los mismos parámetros para la futura utilización (como la orden CLOSE). El número puede ser de 1 a 7.

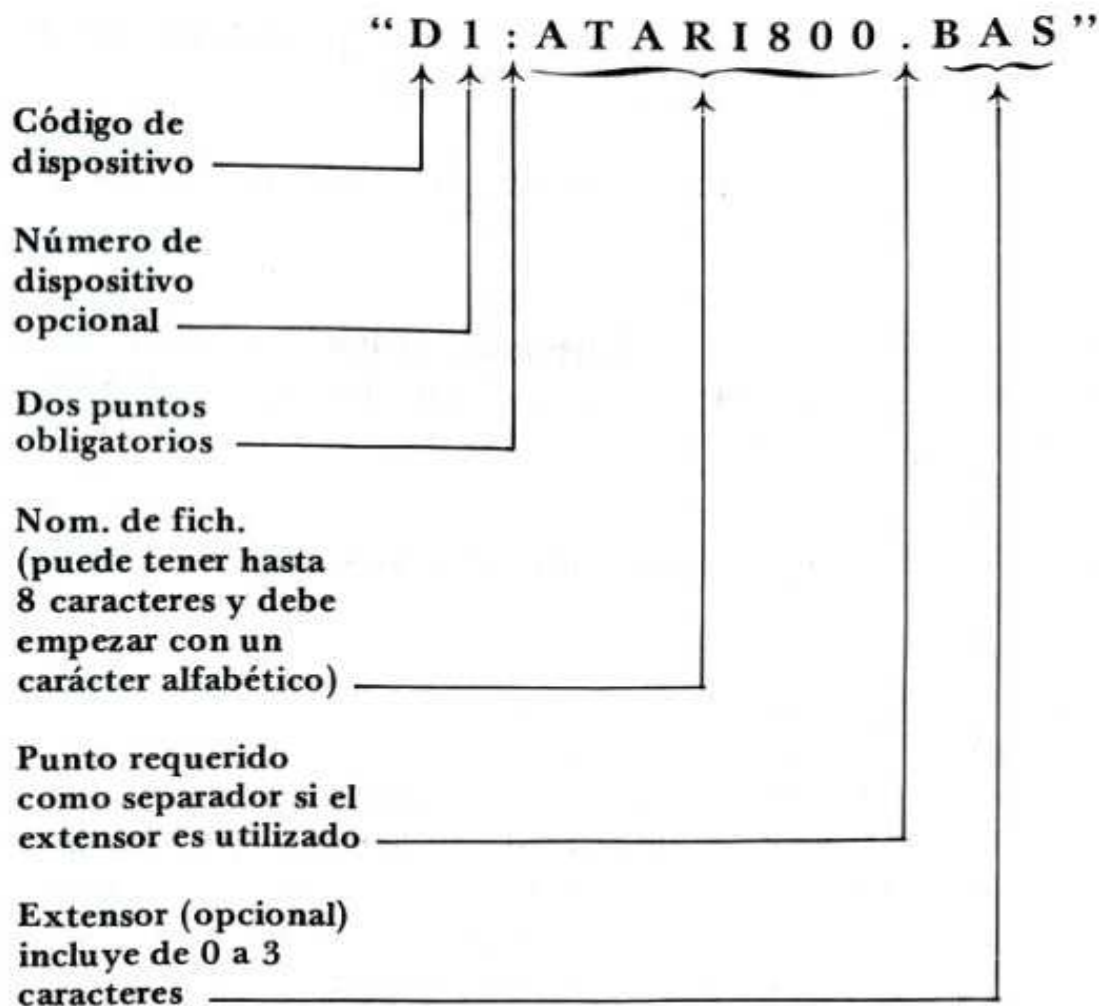
aexpl Es el número de código para determinar la operación de entrada o salida.

Código 4 = Operación de Entrada.
" 8 = Operación de Salida.
" 12 = Operación de Entrada/Salida.
" 6 = Operación de entrada del Directorio del disco.
(En este caso el nombre de fichero es la especificación de la búsqueda.)
" 9 = Operación al final del fichero (de Salida). La operación adjunta es también utilizada para un modo especial de entrada del editado de pantalla. Este modo permite que un programa entre en la próxima línea desde E: sin esperar que el que la utilice oprima la tecla RETURN.

aexp2 Es el código auxiliar dependiente del dispositivo.

nom. de fich. Es la designación específica del fichero encerrado entre comillas. El formato para el parámetro del nombre de fichero es demostrado en la Figura 5-2.

La orden CLOSE simplemente cierra los ficheros que han sido previamente abiertos con una orden OPEN. Observe en el ejemplo que en el aexp que sigue al carácter obligatorio, el # debe ser el mismo que el número de referencia del aexp en la declaración OPEN.



Nota: Los nombre de los ficheros no son utilizados con la grabadora de programas.

Figura 5-2. Descomposición del nom. de fich.

POINT (P.)

Formato: POINT # aexp, avar, avar

Ejemplo: 100 POINT # 2, A, B

IT

Esta orden se utiliza cuando se está leyendo un fichero del disco.

El primer avar especifica el número de sector y el segundo avar especifica el byte de ese sector, que debe ser leído o escrito. Esencialmente se mueve un puntero controlador, que especifica la localización del archivo. Esto se usa para el acceso aleatorio de los datos almacenados en un archivo en disco.

Las órdenes POINT y NOTE se discuten con más detalle en el manual del DOS.

PRINT (PR. ó ?)

Formato: PRINT [# aexp] {,} [exp] 1,exp...]
 {;}

Ejemplos: PRINT X, Y, Z, A\$
 100 PRINT "EL VALOR DE X ES ";X
 100 PRINT "COMAS", "CAUSAN", "ESPACIAMIENTO", "DE",
 "COLUMNA"
 100 PRINT # 3, A\$

Una orden de PRINT puede ser utilizada en el Modo Directo o Modo Diferido. En el Modo Directo, esta orden imprime cualquier información que esté contenida dentro de las comillas exactamente como aparece. En el primer ejemplo, PRINT X, Y, Z, A\$, la pantalla visualizará los valores actuales de X, Y, Z, y A\$ como aparezcan en el programa residente-RAM. En el último ejemplo, PRINT # 3, A\$, el # 3 es el especificador del fichero (puede ser cualquier número entre el 1 y el 7) que controla en cuál dispositivo será imprimido el valor de A\$. (Ver **Dispositivos de Entrada/Salida.**)

Una coma hace que se mueva a la próxima posición de sangrado. Varias comas seguidas harán que salte a varios sitios de sangrado. Un punto y coma hace que el próximo aexp o el sexp sea ubicado inmediatamente después de la expresión que la precede sin ningún espacio. Por lo tanto, en el segundo ejemplo se pone un espacio antes de las comillas finales para que el valor de X no se ponga inmediatamente después de la palabra "ES". Si ninguna coma ni punto y coma es utilizado al final de una declaración PRINT, entonces un RETURN es sacado y el próximo PRINT comenzará en la siguiente línea.

PUT (PU.)/GET (GE.)

Formato: PUT # aexp, aexp
 GET # aexp

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

Las órdenes PUT y GET son contrarias. La orden PUT saca un único byte desde el 0 al 255 al fichero especificado por el # aexp. (El # es un carácter obligatorio en ambas órdenes). La orden GET lee un byte del 0 al 255 (utilizando el # aexp para designar el fichero, etc. en el diskette o en otro sitio) y luego almacena el byte en la variable avar.

READ (REA.) DATA (D.)

Formato: READ var [, var...]
DATA adata [, adata...]

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

Estas dos órdenes son siempre utilizadas en conjunto y la declaración DATA siempre es utilizada en Modo Diferido (*). La declaración DATA se puede ubicar en cualquier parte del programa, pero debe contener tantas piezas de datos como estén definidos en la declaración READ. Si no, un ERROR de "error por falta de datos" se visualizará en la pantalla.

Las variables de cadenas utilizadas en declaraciones READ deben ser dimensionadas y no pueden ser suscritas (Ver la Sección CADENAS). Tampoco se pueden utilizar las variables en colección en una declaración READ.

La declaración DATA tiene un número de datos en cadena para que sean accesibles a la declaración READ. No puede incluir operaciones aritméticas, funciones, etc. Además, el tipo de datos en la declaración DATA debe ser igual al tipo de la variable definida en la declaración READ correspondiente.

El siguiente programa suma una lista de números:

```
100 FOR N=1 TO 5
200 READ D
300 M=M+D
400 NEXT N
500 PRINT "SUMA TOTAL IGUAL A ";M
600 END
700 DATA 30, 15, 106, 17, 87
```

Figura 5-3. Listado de programa READ/DATA.

El programa de la Figura 5-3, al ejecutarse, imprimirá la declaración:

SUMA TOTAL IGUAL A 255

(*) Un READ en Modo Directo únicamente leerá datos si una declaración DATA fue ejecutada en el programa.

SAVE (S.)

Formato: SAVE nombre de fichero

Ejemplo: SAVE "D1:FICHERO.ESP"

La orden SAVE es similar a la orden CSAVE excepto que el tema de nombres enteros de los ficheros pueden ser utilizados. El número de código de dispositivo es opcional cuando se esté utilizando solamente una unidad de disco. El valor de omisión es a la unidad de disco #1. Como el LOAD, utiliza espacios largos de inter-grabación en el cassette (ver CSAVE) y la forma señalizada del programa.

STATUS (ST.)

Formato: STATUS # aexp,avar

Ejemplo: 350 STATUS #1,Z

La orden STATUS llama a la rutina STATUS para el dispositivo especificado (aexp). El status de la orden STATUS (ver MENSAJES DE ERROR, Apéndice B) es almacenado en la variable especificada (avar). Esto puede ser útil para dispositivos futuros tales como el interface RS-232.

XIO (X.)

Formato: XIO cmdno, # aexp, aexp1, aexp2, nom. de fichero.

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

La orden XIO es una declaración general de Entrada/Salida utilizado para operaciones especiales. Un ejemplo es su capacidad de rellenar un área en la pantalla entre puntos y líneas trazados, con un color (Ver la Sección 9). Los parámetros para esta orden se definen a continuación:

cmdno	OPERACION	EJEMPLO
3	OPEN	Igual que el OPEN del BASIC.
5	GET REGISTRO	Estas cuatro órdenes son similares a BASIC INPUT, GET, PRINT y PUT.
7	GET CARACTERES	
9	PUT REGISTRO	
11	PUT CARACTERES	
12	CLOSE	Respectivamente.
13	STATUS PEDIDO	Igual que el CLOSE del BASIC.
		Igual que el STATUS del BASIC

17	DIBUJAR LINEA	Igual que el DRAWTO del BASIC.
18	RELLENAR	Ver la Sección 9.
32	RE-NOMBRAR	XIO 32, #1,0,0, "D:TEMP.CAROL"
33	SUPRIMIR	XIO 33, #1,0,0, "D:TEMP.BAS"
35	TRABAR FICHERO	XIO 35, #1,0,0, "D:TEMP.BAS"
36	DESTRABAR FICHERO	XIO 36, #1,0,0, "D:TEMP.BAS"
37	POINT	Igual que el POINT del BASIC.
38	NOTE	Igual que el NOTE del BASIC.
254	FORMATO	XIO 254, #1,0,0, "D2:"

- aexp Es un número de dispositivo (igual que en el OPEN). La mayoría de las veces se ignora, pero debe ser precedido por el #.
- aexp1
aexp2 Son dos bytes de control auxiliar. Su utilización depende de la orden y la declaración particular. En la mayoría de los casos no son utilizados y son fijados a 0.
- nom. de fich. Es una expresión de cadena que especifica el dispositivo. Debe encerrarse entre comillas. Aunque algunos comandos, con el relleno (X10 18) no lo tienen en cuenta, ha de estar siempre presente en la declaración.

ENCADENANDO PROGRAMAS

Si un programa necesita más memoria de la que hay disponible, actúe de la siguiente forma, para encadenar programas:

1. Teclee la primera parte del programa de la forma habitual.
2. La última línea de la primera parte contiene únicamente el número de línea y el comando RUN "C:".
3. Posicione el grabador en un sector limpio. Anote el número de contador, para su ejecución posterior. Apriete los botones PLAY y RECORD hasta que ambos queden apretados.
4. Escriba "SAVE "C:" y presione RETURN.
5. Cuando suene el sonido "beep", presione otra vez RETURN.
6. Cuando en la pantalla se escriba READY, y el grabador se pare, escriba NEW RETURN.
7. Repita las mismas instrucciones para la segunda parte del programa.
8. Como la segunda parte del programa es esencialmente un programa nuevo, es posible usar los mismos números de línea que en la primera parte del programa.
9. Si hubiera una tercera parte del programa, asegúrese que la última línea de la segunda parte del programa, es un comando RUN "C:".

Para ejecutar un programa encadenado, utilice los siguientes pasos:

1. Posicione el grabador en el principio de la parte 1 del programa.
2. Apriete el botón PLAY.
3. Escriba RUN "C:" RETURN.
4. Cuando suene un zumbido, pulse RETURN otra vez.

El ordenador automáticamente carga y ejecuta la primera parte del programa, un sonido "beep" indica cuando debe pulsarse RETURN para conectar el motor para cargar/ejecutar la segunda parte del programa. La carga llevará unos pocos segundos.

Nota: Un programa de una sola parte puede ser grabado en la cinta y cargado en la memoria de esta misma forma, o se pueden utilizar los comandos CSAVE y CLOAD.

Nota: Recuerde cargar el DOS antes de escribir su programa si lo va a almacenar en disco.

MODIFICANDO UN PROGRAMA EN UN DISCO

El procedimiento para modificar un programa BASIC existente almacenado en un disco, es demostrado con los siguientes pasos:

1. Conecte la unidad de disco y enciéndala —sin ningún diskette dentro.
2. Espere a que se apague la luz de BUSY y que el motor de la unidad se pare. Abra entonces el seguro de la unidad.
3. Inserte el diskette en la unidad y cierre la puerta.
4. Encienda la consola. El DOS deberá cargarse y el mensaje de READY aparecerá en la pantalla.
5. Para cargar el programa desde el disco, teclee:
LOAD "D:nombre de fichero.ext"
6. Modifique el programa (o escriba uno nuevo).
7. Para salvar el programa en el disco, teclee:
SAVE "D:nombre de fichero.ext"
8. Siempre espere que se apague la luz de BUSY antes de quitar el diskette.
9. Para obtener un listado del directorio, no quite el diskette y teclee DOS.

- Al pulsar RETURN, el Menu DOS será visualizado. Elija la letra de selección, A, y pulse la tecla RETURN dos veces para ver el listado del directorio en la pantalla; o teclee A seguido por la pulsación de la tecla RETURN y seguido por la letra P: RETURN para listar el directorio en la impresora.
10. Para volver al BASIC, teclee B RETURN o pulse RESET.

6 BIBLIOTECA DE FUNCIONES

Esta sección describe las funciones aritméticas, trigonométricas y de propósito especial que estén incorporadas en el BASIC ATARI. Una función realiza una computación y devuelve el resultado (generalmente un número) para que se lea o para uso adicional de computación. Incluido en las funciones trigonométricas hay dos declaraciones, radianes (RAD) y grados (DEG), que se utilizan frecuentemente con funciones trigonométricas. Cada función descrita en esta sección puede ser utilizada en el Modo Directo o en el Modo Diferido. Las funciones múltiples son totalmente iguales.

Las siguientes funciones y declaraciones son descritas en esta sección:

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

FUNCIONES ARITMETICAS

ABS **Formato:** ABS (aexp)
Ejemplo: 100 AB = ABS (-190)

Este devuelve el valor absoluto del número sin molestarse con que sea positivo o negativo. El valor devuelto siempre es positivo.

CLOG **Formato:** CLOG (aexp)
Ejemplo: 100 C = CLOG (1)

Este devuelve el logaritmo en la base de 10 de la variable o expresión entre paréntesis. CLOG (0) debe dar un error y CLOG (1) debe ser 0.

EXP **Formato:** EXP (aexp)
Ejemplo: 100 PRINT EXP (3)

Este devuelve el valor de e (aprox. 2.71828283), elevado a la potencia especificada por la expresión entre paréntesis. En el ejemplo dado arriba el número devuelto es 20.0855365. En algunos casos, el EXP es únicamente preciso a seis dígitos significativos.

INT **Formato:** INT (aexp)
Ejemplo: 100 I = INT (3.455)
 100 X = INT (-14.66789)

Este devuelve el mayor número entero menor que o igual al valor de la expresión. Esto es cierto tanto si la expresión se evalúa a un número positivo como negativo. Por lo tanto, en el primer ejemplo de arriba, la I es utilizada para almacenar el número 3. En el segundo ejemplo, la X es utilizada para almacenar el número -15 (el primer número entero que es menor que o igual a -14.66789). Esta función INT no debe ser confundida con la función utilizada en las calculadoras que simplemente truncan (cortan) todos los lugares decimales.

LOG **Formato:** LOG (aexp)
Ejemplo: 100 L = LOG (67.89/2.57)

Este devuelve el logaritmo natural de número o la expresión entre paréntesis. El LOG (0) debe dar un error y el LOG (1) debe ser 0.

RND **Formato:** RND (aexp)
Ejemplo: 10 A = RND (0)

Este devuelve un número al azar generado entre el 0 y el 1, pero nunca devuelve el número 1. La variable o la expresión entre paréntesis siguiendo al RND es "postiza" y no tiene ningún efecto sobre los números devueltos. Generalmente la función RND es utilizada en combinación con otras declaraciones o funciones del BASIC para devolver un número para juegos, el haber decisiones y cosas similares. Aquí hay una rutina sencilla que devuelve un número al azar entre el 0 y el 999:

```
10 X=RND(0)
20 RX=INT(1000*X)
30 PRINT RX
```

SGN **Formato:** SGN (aexp)
Ejemplo: 100 X = SGN (-199)

Devuelve un -1 si un aexp se evalúa a un número negativo; un 0 si el aexp se evalúa a un 0; o un 1 si el aexp se evalúa a un número positivo.

SQR **Formato:** SQR (aexp)
Ejemplo: 100 PRINT SQR (100)

Este devuelve la raíz cuadrada del aexp, el cual debe ser positivo.

FUNCIONES TRIGONOMETRICAS

ATN **Formato:** ATN (aexp)
Ejemplo: 100 X = ATN (65)

Este devuelve el arco tangente de la variable o expresión entre paréntesis.

COS **Formato:** COS (aexp)
Ejemplo: 100 X = COS (X + Y + Z)

Nota: Se supone que la X, Y y Z son previamente definidas. Este devuelve el coseno trigonométrico de la expresión entre paréntesis.

SIN **Formato:** SIN (aexp)
Ejemplo: 100 X = SIN (Y)

Nota: Se supone que la Y haya sido previamente definida. Este devuelve el seno trigonométrico de la expresión entre paréntesis.

DEG/RAD **Formato:** DEG
 RAD
Ejemplo: 100 DEG
 100 RAG

Estas dos declaraciones permiten al programador especificar si en las operaciones trigonométricas posteriores se van a utilizar

grados o radianes, respectivamente. El ordenador adopta por defecto radianes si el DEG no es especificado. Una vez que se ha ejecutado la declaración DEG, el RAD se debe utilizar para volver a radianes.

Ver el Apéndice E para funciones trigonométricas adicionales que se pueden derivar.

FUNCIONES DE PROPOSITOS ESPECIALES

ADR Formato: ADR (svar)
Ejemplo: ADR (A\$)

Este devuelve la dirección decimal de la memoria de la cadena especificada por la expresión entre paréntesis. Conociendo la dirección permite que el programador pueda pasar la información a la rutina **USR**, etc. (Ver **USR** y Apéndice D).

FRE Formato: FRE (aexp)
Ejemplo: PRINT FRE (0)
 100 IF FRE (0) < 100 THEN PRINT "QUEDA
 POCA MEMORIA"

Esta función devuelve el número de bytes libres de RAM. Su utilización primaria es en el Modo Directo con una variable postiza (0) para informar al programador cuánto espacio de memoria queda para la completación de un programa. Por supuesto el **FRE** también puede ser utilizado dentro de un programa **BASIC** en el Modo Diferido.

PEEK Formato: PEEK (aexp)
Ejemplo: 100 IF PEEK (4000) = 255 THEN PRINT "255"
 100 PRINT "MARGEN IZQUIERDO ="; PEEK (82)

Devuelve el contenido de una posición especificada de memoria (aexp). La dirección especificada debe ser un número entero o una expresión aritmética que se evalúa a un número entre el 0 y el 65535 y representa la dirección de la memoria en anotación decimal (no hexadecimal). El número devuelto también será un número entero decimal con una extensión de 0 a 255. Esta función permite a quien lo utilice examinar las posiciones RAM ó ROM. En el primer ejemplo de arriba, el **PEEK** es utilizado para determinar si la posición 4000 (decimal) contiene al número 255. En el segundo ejemplo, la función del **PEEK** es utilizada para examinar el margen izquierdo.

POKE

Formato: POKE aexp1, aexp2
Ejemplo: POKE 82, 10
100 POKE 82, 20

Aunque esto no es una función, está incluida en esta sección porque está estrechamente asociada con la función PEEK. Esta orden POKE inserta datos en la posición de memoria o modifica los datos ya almacenados allí. En el formato de arriba, aexp1 es la dirección decimal de la posición en la que se ha de escribir, y aexp2 es el número que se ha de introducir. Observe que este número es un número decimal entre el 0 y el 255. El POKE no puede ser utilizado para alterar las posiciones de ROM. Al adquirir familiaridad con esta orden se aconseja mirar la posición de memoria con un PEEK y anotar el contenido de la ubicación. Entonces, si el POKE no funciona como se deseaba, el contenido original puede ser introducido de nuevo en dicha posición.

El ejemplo del Modo Directo de arriba cambia el margen de la izquierda de la pantalla desde su posición de ausencia de 2 a una posición de 10. En otras palabras, el nuevo margen será de 8 espacios hacia la derecha. Para reponer el margen a su posición normal de ausencia, oprima la tecla RESET.

USR

Formato: USR (aexp1 [, aexp2] [, aexp3...])
Ejemplo: 100 RESULTADO = USR (ADD1, A*2)

Esta función devuelve los resultados de una subrutina de lenguaje de máquina. La primera expresión, aexp1, debe ser un número entero o una expresión aritmética que se evalúa a un número entero que representa la dirección de la memoria decimal de la rutina del lenguaje de máquina que ha de ser desempeñada. Los argumentos de entrada aexp2, aexp3, etc. son opcionales. Estos deben ser expresiones aritméticas dentro de un alcance decimal de 0 a 65535. Un valor que no sea un número entero puede ser utilizado; sin embargo, será redondeado al número entero más cercano.

Estos valores serán convertidos de formato de número de coma flotante del BCD (Binary Coded Decimal <Código Decimal Binario>) del BASIC a un número binario de dos-byte, y luego almacenado sobre la pila hardware, compuesto de un grupo de ubicaciones de memoria RAM bajo el control directo del chip micro

procesador 65 02. La figura 6-1 ilustra la estructura de la pila hardware:

N	Número de argumentos en la pila — puede ser 0
X ₁	Alto byte del argumento X
X ₂	Bajo byte del argumento X
Y ₁	Alto byte del argumento Y
Y ₂	Bajo byte del argumento Y
Z ₁	Alto byte del argumento Z
Z ₂	Bajo byte del argumento Z
:	
:	
R ₁	Bajo byte de la dirección de retorno
R ₂	Alto byte de la dirección de retorno

Figura 6.1. Definición de la pila hardware.

Nota: X es el argumento que sigue la declaración de la rutina, Y es la segunda, Z es la tercera, etc. Hay N pares de bytes.

Ver la Sección 11 para la descripción de la función USR en la programación de lenguaje de máquina. El Apéndice D define los bytes en RAM disponibles para la programación de lenguaje de máquina.

7 CADENAS

Esta sección describe las cadenas y las funciones asociadas con el manejo de cadenas. Cada cadena debe ser dimensionada (Ver la declaración **DIM**, Sección 8) y cada variable de cadena debe finalizar con un \$. Una cadena por sí misma es un grupo de caracteres "encadenados" juntos. Los caracteres individuales pueden ser letras, números o símbolos (incluyendo los símbolos especiales del teclado ATARI). Una subcadena es una parte de una cadena más larga y cualquier subcadena es accesible en el BASIC ATARI si la cadena ha sido correctamente dimensionada (Ver el final de la Sección). Los caracteres en una cadena están clasificados del 1 a la longitud actual de la cadena, la cual es menor que o igual a la longitud dimensionada de la cadena.

Las funciones de cadena descritas en esta sección son:

ASC	STR\$
CHR\$	VAL
LEN	

ASC **Formato:** ASC (sexp)
Ejemplo: 100A=ASC(A\$)

La función devuelve el número-código del ATASCII para el primer carácter de la expresión en cadena (sexp). Esta función se puede utilizar en el Modo Directo o en el Modo Diferido. La Figura 7-1 es un programa corto ilustrando 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, el cual es un código de ATASCII para la letra "E". Observe que cuando la cadena en sí es utilizada, se debe encerrar entre comillas.

CHR\$

Formato: CHR\$ (aexp)
Ejemplo: 10 PRINT CHR\$ (65)
10 A\$ = CHR\$ (65)

Esta función de cadena devuelve el carácter, en formato de cadena, correspondiente al número de código ATASCII entre paréntesis. Únicamente un carácter es devuelto. En los ejemplos de arriba, la letra A es devuelta. Utilizando las funciones ASC y CHR\$, el programa siguiente imprime las letras mayúsculas y minúsculas del abecedario.

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

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

Nota: Únicamente puede haber un STR\$ y únicamente un CHR\$ en una comparación lógica.

LEN

Formato: LEN (aexp)
Ejemplo: 100 PRINT LEN (A\$)

Esta función devuelve la longitud en bytes de la cadena asignada. Esta información puede entonces ser imprimida o utilizada posteriormente en el programa. La longitud de una variable en cadena es simplemente el índice para el carácter que está actualmente al final de la cadena. Las cadenas tienen una longitud de cero hasta que los caracteres han sido almacenados en ellos. Es posible almacenar en el medio de una cadena utilizando suscritos. Sin embargo, el principio de la cadena contendrá residuos si previamente no se almacenó nada allí.

La siguiente rutina ilustra uno de los usos de LEN.

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

Figura 7-3. Ejemplo de la función LEN.

El resultado de realizar al programa de arriba sería 5.

STR\$ **Formato:** STR\$ (aexp)
 Ejemplo: A\$ = STR\$ (65)

Esta función de cadena devuelve una cadena en la que está almacenado el número entre paréntesis. El ejemplo de arriba devolverá el número real 65, pero sería reconocido por el ordenador como una cadena.

Nota: Únicamente puede haber un STR\$ y únicamente un CHR\$ en una comparación lógica. Por ejemplo, A = STR\$(1) > STR\$(2) no es válido y no funcionará correctamente.

VAL **Formato:** VAL (sexp)
 Ejemplo: 100 A = VAL (A\$)

Esta función devuelve un número con el mismo valor que el número almacenado como una cadena. Esto es lo contrario de una función STR\$. Utilizando esta función, el ordenador puede desempeñar operaciones aritméticas en cadenas como está demostrado en el siguiente ejemplo de programa:

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

Figura 7-4. Función de VAL.

Al ejecutarse, la pantalla visualiza:

LA RAIZ CUADRADA DE 10000 ES 100

No es posible utilizar la función VAL con una cadena que no empiece con un número, o que no puede ser interpretada por el ordenador como un número. Puede, sin embargo, interpretar números de puntos flotantes; ej., VAL ("1E9") devolvería el número 1.000.000.000.

MANIPULACIONES DE CADENAS

Las cadenas se pueden manipular de varias formas. Pueden ser partidas,

concatenadas, reacomodadas y clasificadas. Los siguientes párrafos describen las distintas manipulaciones.

Concatenación de Cadenas

Concatenación significa unir a dos o más cadenas juntas para formar una cadena larga. Cada cadena que ha de ser incluida en una cadena mayor es llamada una "subcadena". Cada subcadena debe ser dimensionada (Ver DIM). En el BASIC ATARI, una subcadena puede contener hasta 99 caracteres (incluyendo los espacios). Después de la concatenación, las subcadenas se pueden almacenar en otra variable de cadena, imprimir, o utilizar en las secciones posteriores del programa. La Figura 7-5 es una muestra de un programa que demuestra la concatenación de la cadena. En este programa, A\$, B\$ y C\$ son concatenados y ubicados en A\$.

```
10 DIM A$(100), B$(100), C$(100)
20 A$="VAMOS A CONCATENAR"
30 B$="TRES SUBCADENAS Y"
40 C$="ALMACENARLAS EN UNA SOLA CADENA: A$"
50 A$(LEN(A$)+1)=B$
60 A$(LEN(A$)+1)=C$
70 PRINT A$
```

Figura 7-5. Ejemplo de concatenación de cadena.

Seccionando Cadenas

El formato de una variable suscrita en cadena quedará como a continuación:

svarnombre (aexp1 [,aexp2])

El svarnombre es utilizado para indicar el nombre de la variable en cadena no suscrita (con \$). El aexp1 indica la ubicación del comienzo de la subcadena y el aexp2 (si es utilizado) indica la ubicación de la terminación de la subcadena. Si ningún aexp2 es especificado, entonces el final de la subcadena es el final actual de la cadena. La ubicación del comienzo no puede ser mayor que la longitud actual de la cadena. Los dos problemas del ejemplo en la Figura 7-6 ilustran una cadena seccionada que no tiene indicada ningún final y una cadena seccionada con una ubicación final indicada.

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

El resultado es BCD
(sin ubicación final)

```
10 DIM S$(20)
20 S$="ATARI 800 BASIC"
30 PRINT S$(7,9)
40 END
```

El resultado es 800
(con ubicación final)

Figura 7-6. Ejemplos de cadenas seccionadas.

Comparaciones de cadena y Clasificadores

En comparaciones de cadena, los operadores lógicos son utilizados exactamente de la misma manera que lo son los números. El segundo programa en el Apéndice H es un ejemplo típico de los clasificadores de burbuja.

Al utilizar los operadores lógicos, recuerde que cada letra, número y símbolo es asignado a un número de código ATASCII. Unas pocas reglas generales son aplicables a estos códigos:

1. Los códigos ATASCII para los números están medidos en el orden del valor real del número y siempre son menores que los códigos para las letras. (Ver Apéndice C.)
2. Las letras en mayúsculas tienen menor valor numérico que las letras minúsculas. Para obtener el código ATASCII para una letra en minúscula si conoce el valor de la letra mayúscula, agregue 32 al código de la letra mayúscula.

Nota: El sistema de dirección de la memoria del BASIC ATARI mueve las cadenas en la memoria para hacer lugar para las nuevas declaraciones. Esto hace que la dirección de la cadena varíe si un programa es modificado o si el Modo Directo es utilizado.

8 TABLAS Y MATRICES

Una tabla es una lista uni-dimensional de números asignados a variables suscritas; ej., $A(0)$, $A(1)$, $A(2)$. Los números de los suscritos oscilan del 0 al valor dimensionado. La Figura 8-1 ilustra una tabla de 7 elementos.

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

Figura 8-1. Ejemplo de una tabla.

Una matriz, en este contexto, es una tabla bi-dimensional que contiene filas y columnas. Las filas corren horizontalmente y las columnas corren verticalmente. Los elementos de la matriz son almacenados por el BASIC en un orden de fila mayor. Esto quiere decir que todos los elementos de la primera fila son almacenados primero, seguido por todos los elementos de la segunda fila, etc. La Figura 8-2 ilustra una matriz de 7 por 4.

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

Figura 8-2. Ejemplo de matriz.

Esta sección describe las dos órdenes asociadas con las tablas, matrices y cadenas, y cómo rellenar las tablas y las matrices. Las órdenes de esta sección son:

	DIM	CLR
DIM (DL.)	Formato: DIM { svar (aexp) mvar (aexp [,aexp]) }	$\left[\left\{ \begin{array}{l} \text{, svar (aexp)} \\ \text{, mvar (aexp [,aexp])} \end{array} \right\} \right] \dots$
	Ejemplos: DIM A (100) DIM M (6,3) DIM B\$ (20)	Utilizado con cadenas

Una declaración DIM es utilizada para reservar una cierta cantidad de ubicaciones en la memoria para una cadena, tabla o matriz. Un carácter en una cadena toma un byte en la memoria y un número en una tabla toma 6 bytes. El primer ejemplo reserva 101 ubicaciones para una tabla designada A. El segundo ejemplo reserva siete filas por cuatro columnas para una tabla bi-dimensional (matriz) designada M. El tercer ejemplo reserva 20 bytes designados B\$. Todas las cadenas, tablas y matrices deben ser dimensionadas. Es un buen hábito poner todas las declaraciones DIM al principio del programa. Observe en la Figura 8-1 que aunque la tabla es dimensionada como DIM A(6), hay realmente siete elementos en la tabla por culpa del elemento 0. Aunque la Figura 8-2 es dimensionada como DIM M(6,3), 28 ubicaciones son reservadas.

Nota: El Ordenador Personal ATARI no marca con iniciales a las variables de tablas o matriz a 0 al principio de la ejecución del programa. Para inicializar los elementos de una tabla o de una matriz a 0, utilice los siguientes pasos de programa:

```

10 DIM A(100)
20 FOR E=0 TO 100
30 A(E) = 0
40 NEXT E

```

Las tablas y matrices son "rellenadas" con datos al utilizar las declaraciones FOR/NEXT, las declaraciones READ/DATA y las órdenes de INPUT. La Figura 8-3 ilustra la "construcción" de parte de una tabla utilizando el lazo FOR/NEXT, y la Figura 8-4 ilustra la construcción de una tabla utilizando las declaraciones READ/DATA.

```

10 DIM A(100)
20 X=10
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. Construcción de tabla con FOR/NEXT.

```

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. Construcción de tabla con READ/DATA.

La Figura 8-5 demuestra un ejemplo para construir una matriz de 6 por 3.

```

10 DIM M(6,3)
20 FOR FILA=0 TO 6
30 FOR COL=1 TO 3
40 M(FILA,COL)=INT(RND(0)*1000)
50 NEXT COL:NEXT FILA
60 FOR FILA=0 TO 6
70 FOR COL=1 TO 3
80 PRINT M(FILA,COL)
90 NEXT COL:PRINT :NEXT FILA

```

Figura 8-5. Construyendo una matriz.

Observe que las palabras FILA y COLUMNA no son órdenes, ni declaraciones, ni funciones, ni palabras claves del BASIC. Son simplemente nombres de variables utilizadas aquí para designar

que función de lazo es la primera. El programa se podría haber escrito sencillamente con X e Y como los nombres de las variables.

CLR

Formato: CLR

Ejemplo: 200 CLR

Esta orden despeja la memoria de todas las tablas, matrices y cadenas previamente dimensionadas para que la memoria y los nombres de la variable se puedan utilizar para otros propósitos. También despeja los valores almacenados en variables no dimensionadas. Si una matriz, cadena o tabla es necesitada después de una orden CLR, debe ser re-dimensionada con una orden DIM.

9 COMANDOS Y MODOS GRAFICOS

Esta sección explica las órdenes del BASIC ATARI y los distintos modos gráficos del Ordenador Personal de ATARI. Utilizando estas órdenes, es posible crear gráficos para juegos, gráficos y patrones.

Las órdenes que serán descritas en esta sección:

GRAPHICS	LOCATE	PUT/GET
COLOR	PLOT	SET/COLOR
DRAWTO	POSITION	XIO

Las órdenes PUT/GET y XIO explicadas en esta sección son aplicaciones especiales de las mismas órdenes descritas en la Sección 5.

GRAPHICS Formato: GRAPHICS aexp
(GR.) Ejemplo: GRAPHICS 2

Esta orden es utilizada para seleccionar uno de los 16 modos gráficos. La Tabla 9-1 resume los dieciséis modos y las características de cada uno. La orden GRAPHICS automáticamente abre la pantalla, S: (la ventana gráfica), como dispositivo #6, por lo que para escribir texto en la ventana de texto, no es necesario el código del dispositivo. El aexp tiene que ser positivo, redondeado al número entero más cercano. El modo gráfico 0 es una visualización de pantalla entera mientras que los modos 1 al 8 son visualizaciones de las pantallas partidas. Para anular la pantalla partida, agregue los caracteres +16 al número de modo (aexp) en la orden GRAPHICS. Agregando 32 impide que la orden GRAPHICS despeje la pantalla.

Para volver al modo gráfico 0 en el Modo Directo, pulse **SYSTEM RESET** o teclee GR.0 y pulse **RETURN**.

TABLA 9.1 – TABLA DE MODOS Y DE FORMATOS DE PANTALLA

FORMATO DE PANTALLA							
Modo de Gráficas	Tipo de Modo	Colum-nas	Filas-Pantalla Dividida	Filas-Pantalla Entera	Núm. de Colores	RAM Requerida (Bytes)	
						Dividida	Entera
0	TEXT	40	—	24	1-1/2	—	992
1	TEXT	20	20	24	5	674	672
2	TEXT	20	10	12	5	424	420
3	GRAPHICS	40	20	24	4	434	432
4	GRAPHICS	80	40	48	2	694	696
5	GRAPHICS	80	40	48	4	1174	1176
6	GRAPHICS	160	80	96	2	2174	2184
7	GRAPHICS	160	80	96	4	4190	4200
8	GRAPHICS	320	160	192	1-1/2	8112	8138
9	GRAPHICS	80	—	192	1	—	8138
10	GRAPHICS	80	—	192	9	—	8138
11	GRAPHICS	80	—	192	16	—	8138
12	GRAPHICS	40	20	24	5	1154	1152
13	GRAPHICS	40	10	12	5	664	660
14	GRAPHICS	160	160	192	2	4270	4296
15	GRAPHICS	160	160	192	4	8112	8138

Los siguientes párrafos describen los dieciséis modos gráficos.

**MODO
GRAFICO 0**

Por defecto, éste es el modo de 1-color, 2-luminosidad para los Ordenadores Personales ATARI. Contiene una matriz de pantalla de 24 por 40. Los valores de ausencia de los márgenes que están fijados en 2 y en 39 permiten 38 caracteres por línea. Los márgenes pueden ser cambiados al “empujar” el margen izquierdo y el margen derecho en las posiciones (82 y 83). Ver el Apéndice 1. Algunos sistemas tienen diferentes valores de ausencia para los márgenes. El color de los caracteres es determinado por el color de fondo.

Únicamente la luminosidad de los caracteres pueden ser diferentes. La visualización de la pantalla entera tiene un área de visualización azul bordeada en negro (si el borde no es especificado para que sea de otro color). Para visualizar los caracteres en posiciones especificadas, utilice uno de los siguientes métodos:

Método 1.

lineno POSITION aexp1, aexp2

Pone al cursor en la ubicación especificada por aexp1 y aexp2.

lineno PRINT sexp.

Método 2.

lineno GR.0	<i>Especifica el Modo Gráfico.</i>
lineno POKE 752,1	<i>Suprime el cursor.</i>
lineno COLOR ASC (sexp)	<i>Especifica carácter a ser imprimido.</i>
lineno PLOT aexp1, aexp2	<i>Especifica dónde se ha de imprimir el carácter.</i>
lineno GOTO lineno	<i>Empieza el lazo para prevenir que el READY se imprima (GOTO al mismo número de línea).</i>
	<i>Oprima la tecla BREAK para finalizar el lazo.</i>

El GRAFICO 0 también es utilizado como una orden de despeje de la pantalla en el Modo Directo o en el Modo Diferido. Finaliza cualquier modo gráfico previamente elegido y devuelve la pantalla al modo de ausencia (GRAPHICS 0).

MODO GRAFICO 1 y 2

Como es definido en la Tabla 9.1, estos dos modos de 5-color son modos de texto. Sin embargo, ambos son modos de pantalla partida (Ver la Figura 9-1). Los caracteres imprimidos en el Modo Gráfico 1 son el doble de ancho que aquellos imprimidos en el Modo Gráfico 0, pero son de la misma altura. Los caracteres imprimidos en el Modo Gráfico 2, son el doble de ancho y altura de aquellos en el Modo Gráfico 0. En el modo de pantalla partida una orden PRINT es utilizada para visualizar los caracteres en la ventana del texto o la del gráfico. Para imprimir caracteres en la ventana gráfica, especifique el dispositivo #6 después de la orden PRINT.

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

Los colores de ausencia dependen del tipo de carácter entrado. La Tabla 9-2 define los colores de ausencia y el registro de color para cada uno utilizado.

Si no es especificado de otra manera, todos los caracteres serán visualizados en mayúsculas de forma no-inversa. Para imprimir letras minúsculas y caracteres de gráficos, utilice un POKE 756,226. Para volver a mayúsculas, utilice POKE 756,224.

TABLA 9-2. COLORES DE AUSENCIA PARA TIPOS DE ENTRADA ESPECIFICOS

Tipo de carácter	Registro de color	Color de ausencia
Mayúscula alfabética	0	Naranja
Minúscula alfabética	1	Verde claro
Mayúscula alfabética inversa	2	Azul oscuro
Minúscula alfabética inversa	3	Rojo
Números	0	Naranja
Números inversos	2	Azul oscuro

Nota: Ver SETCOLOR para cambiar los colores de los caracteres.

En los modos gráficos 1 y 2, no hay ningún video-inverso, pero es posible obtener todos los demás caracteres en cuatro colores distintos. (Ver el final de esta sección).

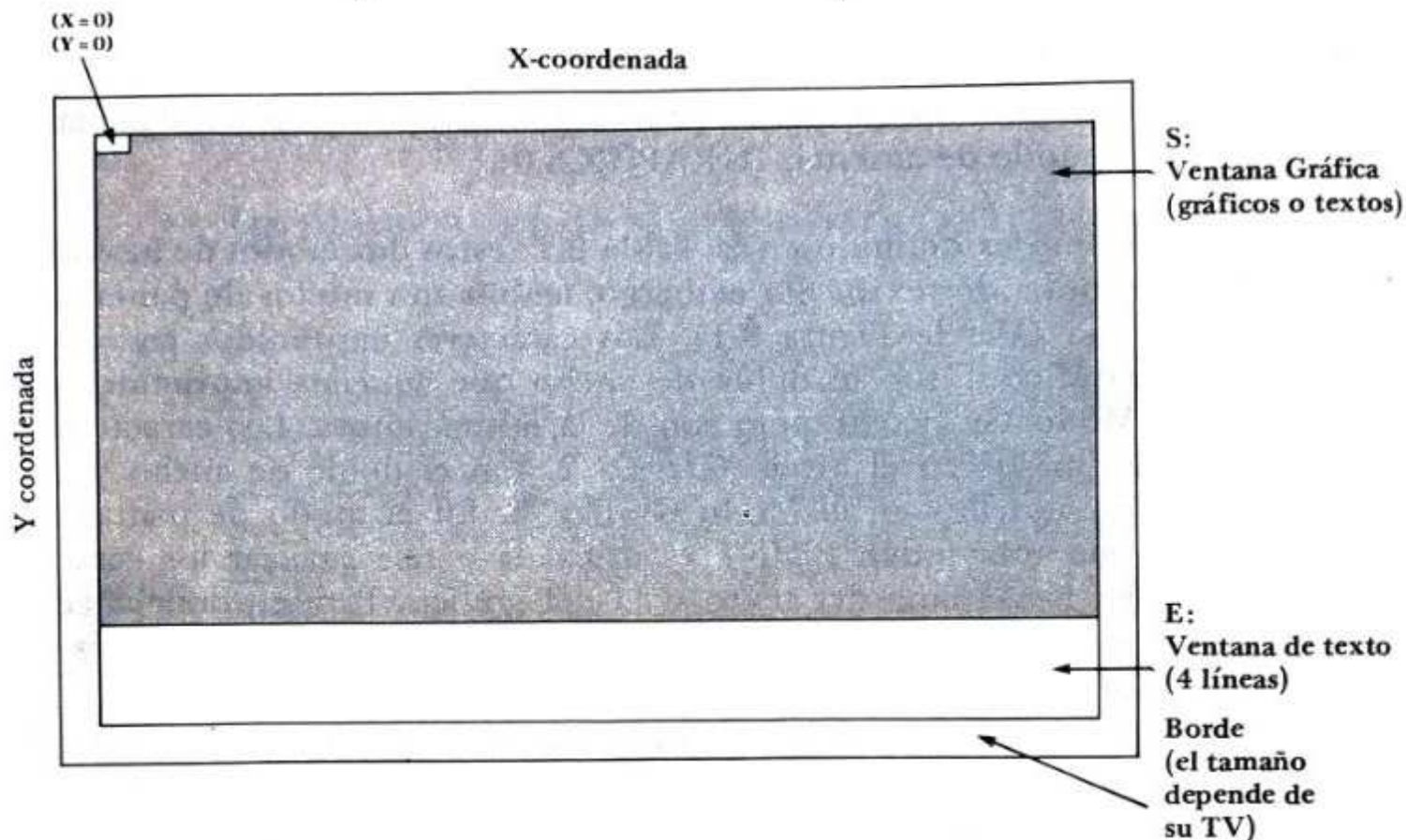


Figura 9.1. Visualización de pantalla partida para los Modos 1 y 2.

Las coordenadas X e Y comienzan en el punto 0 (parte superior izquierda). Los valores máximos son los números de las filas y columnas menos 1 (Ver Tabla 9-1). Esta configuración de pantalla partida puede ser cambiada a una visualización de pantalla entera al agregar los caracteres + 16 al número de modo.

Ejemplo: GRAPHICS 1 + 16

**MODOS
GRAFICOS
3, 5 Y 7**

Estos tres modos gráficos de 4-colores también son visualizaciones de pantalla partida en su estado de ausencia, pero también pueden ser cambiados a visualizaciones de pantalla entera agregando + 16 al número del modo. Los modos 3, 5 y 7 son parecidos, excepto que los modos 5 y 6 utilizan más puntos (pixels) para dibujar. Los puntos son más pequeños, por lo que la resolución será mucho más alta.

**MODOS
GRAFICOS
4 Y 6**

Estos dos modos gráficos de 2-colores son visualizaciones de pantalla partida y pueden visualizar únicamente en dos colores mientras que los otros pueden visualizar 4 y 5 colores. La ventaja de un modo de dos colores es que requiere menos espacio de RAM (Ver la Tabla 9-1). Por lo tanto, es utilizada cuando únicamente se necesitan dos colores y el RAM se está llenando. Estos dos modos también tienen una resolución más alta, lo que significa puntos más pequeños que el Modo Gráfico 3.

**MODO
GRAFICO 8**

Este modo gráfico da la más alta resolución que todos los otros modos gráficos. Como ocupa mucha RAM para obtener este tipo de resolución, únicamente puede tener cabida para un color y para dos luminosidades diferentes.

**MODOS
GRAFICOS
9, 10, 11**

Estos modos gráficos son visualizados en pantalla entera, los tres tienen la misma resolución gráfica. La diferencia entre ellos está en el número de colores que pueden visualizar: 1, 9, 16, respectivamente.

**MODOS
GRAFICOS
12, 13**

Estos dos modos trabajan en pantalla partidas y puede visualizar hasta 5 colores. La diferencia entre estos dos modos estriba en el número de pixels que pueden representar.

**MODOS
GRAFICOS
14, 15**

Son modos de alta resolución, trabajan en pantalla partida, la diferencia entre ellos está en el número de colores que pueden visualizar: 2 y 4.

COLOR (C.)

Formato: COLOR aexp
Ejemplo: 110 COLOR ASC ("A")
110 COLOR 3

El valor de la expresión en la declaración COLOR determina los datos que han de ser almacenados en la memoria de la visualización para todas las órdenes PLOT y DRAWTO subsecuentes hasta que la próxima declaración COLOR es ejecutada. El valor tiene que ser positivo y generalmente es un número entero del 0 al 255.

Los números no enteros son redondeados al número entero más cercano. El hardware de la visualización gráfica interpreta estos datos de distintas maneras en los diferentes modos gráficos. En los modos de texto del 0 a 2, el número puede ser de 0 a 255 (8 bits) y determina el carácter que ha de ser visualizado y su color. (Los dos bits más significantes determinan el color. Eso es porque hay únicamente 64 caracteres distintos disponibles en estos modos en vez de los 256 caracteres del conjunto.)

Las tablas 9-6 y 9-7 al final de esta sección ilustran el conjunto de caracteres internos y la asignación de carácter y color. La Tabla 9-2 es una tabla simplificada que permite la fácil generación de algunos de los colores. Por ejemplo, **COLOR ASC ("A")**; **PLOT 5,5** visualizará un carácter anaranjado A en el modo gráfico 1 ó 2 en la posición 5,5.

Los modos gráficos de 3 a 15 no son modos de texto, así que los datos almacenados en el RAM de visualización simplemente determinan el color de cada pixel. Los modos de 2-colores o 2-luminosidades requieren modo de 0 ó 1 (1 bit) y los modos de 4-colores requieren modos de 0, 1, 2 ó 3. La expresión en la declaración **COLOR** puede tener un valor mayor que 3, pero únicamente se utilizarán uno o dos bits. El color que es realmente visualizado depende del valor en el registro de los colores que corresponden a los datos de 0, 1, 2 ó 3 en el particular modo gráfico que está siendo utilizado. Esto se puede determinar mirando la Tabla 9-5, que le da los colores de ausencia y los números de registros correspondientes. Los colores pueden ser cambiados utilizando **SETCOLOR**.

Observe que cuando el BASIC es impulsado por primera vez, el dato de color es 0, y cuando se ejecuta la orden **GRAPHICS** sin el + 32, todos los puntos son fijados en 0. Por lo tanto, nada parece ocurrirle al **PLOT** y al **DRAWTO** en el **GRAPHICS 3** al 7 cuando ninguna declaración de **COLOR** haya sido ejecutada. Corrija esto tecleando un **COLOR 1** primero.

DRAWTO
(DR.)

Formato: DRAWTO aexp1, aexp2
Ejemplo: 100 DRAWTO 10, 8

Esta declaración hace que una línea sea dibujada desde el último punto expuesto por un **PLOT** (Ver **PLOT**) a la ubicación especificada por el aexp1 y el aexp2. La primera expresión representa la coordenada X y la segunda expresión representa la coordenada Y (Ver la Figura 9-1). El color de la línea es del mismo color que el punto expuesto por el **PLOT**.

**LOCATE
(LOC.)**

Formato: LOCATE aexp1, aexp2, var
Ejemplo: 150 LOCATE 12, 15, X

Esta orden posiciona el cursor gráfico invisible en la posición especificada en la ventana gráfica, recoge los datos en ese punto y lo almacena en la variable aritmética especificada. Esta da un número de 0 a 255 para los modos gráficos de 0 a 2; y de 0 ó 1 para los modos gráficos de 2-color; y 0, 1, 2 ó 3 para los modos de 4-color. Las dos expresiones aritméticas especifican las coordenadas X e Y del punto. El LOCATE es equivalente a:

POSITION aexp1, aexp2:GET #6,avar

Si se hace un PRINT después de un LOCATE o un GET (desde la pantalla) puede ocurrir que se modifiquen los datos que se acaban de examinar. Este problema puede resolverse posicionando de nuevo el cursor y colocando de nuevo el dato que se leyó, antes de ejecutar el PRINT. El siguiente programa ilustra el uso de la orden LOCATE.

```
10 GRAPHICS 3+16
20 COLOR 1
30 SETCOLOR 2,10,8
40 PLOT 10,15
50 DRAWTO 15,15
60 LOCATE 12,15,X
70 PRINT X
```

Figura 9-2. Programa de ejemplo usando LOCATE.

Al ejecutarse, el programa imprime los datos(1) determinados por la declaración COLOR que fue almacenada en el punto 12, 15.

PLOT (PL.)

Formato: PLOT aexp1, aexp2
Ejemplo: 100 PLOT 5,5

La orden PLOT es utilizada en los modos gráficos del 3 al 15 para exponer un punto en la ventana gráfica. El aexp1 especifica la coordenada X y el aexp2 la coordenada Y. El color del punto trazado es determinado por el tono y la luminosidad en el registro de color de la última declaración COLOR ejecutada. Para cambiar este registro de color y el color del punto trazado, utilice SETCOLOR. Los puntos que pueden ser trazados en la pantalla son dependientes del modo gráfico que está siendo utilizado.

La extensión de puntos comienza en el 1 y se extiende a uno menos que el número total de filas (coordenadas X) o columnas (coordenadas Y) expuestas en la Tabla 9-1.

**POSITION
(POS.)**

Formato: POSITION aexp1, aexp2

Ejemplo: 100 POS. 8, 12

La declaración POSITION es utilizada para ubicar el cursor invisible de la ventana gráfica en una ubicación especificada en la pantalla (generalmente precede a una declaración PRINT). Esta declaración puede ser utilizada en todos los modos. Observe que el cursor no se mueve realmente hasta que una orden I/O, que incumbe a la pantalla, es ordenada.

**PUT/GET
(PU./GE.)**

Formato: PUT # aexp, aexp

GET # aexp, avar

Ejemplo: 100 PUT # 6, ASC ("A")

200 GET # 1, X

En los trabajos gráficos, el PUT es utilizado para sacar datos a la visualización de la pantalla. Esta declaración trabaja estrechamente relacionada con la declaración POSITION. Después de un PUT ó GET, el cursor es trasladado a la próxima posición en la pantalla. Tecleando PUT a un dispositivo # 6 hace que la entrada de un-byte (el segundo aexp) sea visualizado en la posición del cursor. El byte es un código ATASCII de byte para un carácter particular (modos 0 a 2) o un dato de color (modos 3 a 15).

El GET se utiliza para almacenar el código del carácter visualizado en la posición del cursor en la variable aritmética especificada. Los valores utilizados en el PUT y GET corresponden a los valores en la declaración de COLOR. (El PRINT y el INPUT también pueden ser utilizados.)

Nota: Tecleando un PRINT después de un GET o un LOCATE desde la pantalla puede hacer que los datos en el punto, que fueron examinados, sean modificados. Para evitar este problema, reposiciones el cursor y ponga los datos que fueron leídos nuevamente en el punto antes de teclear el PRINT.

**SETCOLOR
(SE.)**

Formato: SETCOLOR aexp1, aexp2, aexp3

Ejemplo: 100 SETCOLOR 0, 1, 4

Esta declaración es utilizada para elegir el tono y la luminosidad

particular que ha de ser almacenada en el registro de color especificado. Los parámetros de la declaración SETCOLOR están definidos seguidamente:

- aexp1 = Registro de color (de 0 a 4 dependiendo del modo gráfico).
- aexp2 = Número de tono del color (de 0 a 15. Ver la Tabla 9-3).
- aexp3 = Luminosidad del color. (Debe ser un número entre 0 y 15; cuanto más alto sea el número, más brillante será el color.)

El hardware de visualización de ATARI contiene cinco registros de color, enumerados del 0 al 4. El Sistema Operativo (OS) tiene cinco posiciones RAM (desde COLOR0 hasta COLOR4, ver el Apéndice I: Ubicaciones de Memoria), donde lleva el control de los colores actuales. La declaración SETCOLOR es utilizada para cambiar los valores de estas ubicaciones de RAM. El OS transfiere estos valores a los registros hardware con cada marco de la televisión. La declaración COLOR utiliza distintos números porque especifican los datos que única e *indirectamente* corresponden a un registro de colores. Esto se puede confundir fácilmente, así que se aconseja una experimentación y estudio cuidadoso de las distintas tablas en esta sección.

TABLA 9.3. LOS NUMEROS DE TONO Y COLORES (ORDEN 'SETCOLOR')

COLORES	NUMEROS (aexp) SETCOLOR
GRIS	0
NARANJA CLARO (ORO)	1
NARANJA	2
ROJO - NARANJA	3
ROSA	4
MORADO	5
MORADO-AZUL	6
AZUL	7
AZUL	8
AZUL CLARO	9
TURQUESA	10
VERDE - AZUL	11
VERDE	12
AMARILLO - VERDE	13
VERDE - NARANJA	14
NARANJA CLARO	15

Nota: Los colores varían con el tipo y reajuste del televisor o monitor utilizado.

Ninguna de las órdenes SETCOLOR son necesarias si el conjunto de ausencia de cinco colores es utilizado. Aunque sean posibles 256 combinaciones distintas de color-luminosidad, no más de cinco colores pueden ser expuestos a la vez. El propósito del registro de color y de la declaración SETCOLOR es la de especificar estos cinco colores.

TABLA 9.4. TABLA DE LOS COLORES "DE AUSENCIA" DE LA ORDEN SETCOLOR (*)

SETCOLOR (Registro de color)	Ausencias al color	Luminosidad	Color actual
0	2	8	NARANJA
1	12	10	VERDE
2	9	4	AZUL OSCURO
3	4	6	ROSA O ROJO
4	0	0	NEGRO

(*) La "AUSENCIA" ocurre si ninguna declaración SETCOLOR es utilizada.
Nota: Los colores pueden variar dependiendo del tipo, ajuste y condición del monitor o televisor utilizado.

Un programa ilustrando el Modo Gráfico 3 y las órdenes explicadas hasta ahora en esta sección son demostradas abajo:

```

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 " ORDENADORES ATARI"
90 GOTO 90

```

Las declaraciones COLOR y SETCOLOR fijan los colores de los puntos que han de ser trazados (Ver la Tabla 9.5). La orden SETCOLOR carga el registro de color 0 con el tono 2 (naranja) y con una luminosidad de 8 ("normal"). Las próximas cuatro líneas trazan los puntos que han de ser expuestos. La línea 70

suprime el cursor y la línea 80 imprime la expresión en cadena ORDENADORES ATARI en la ventana de texto.

Observe que el color de fondo nunca fue fijado porque la ausencia es el color deseado (negro).

Si el programa es ejecutado, imprimirá el logo ATARI en la ventana gráfica y la expresión en cadena en la ventana de texto como en la Figura 9-3.

XIO (X.) **Formato:** XIO 18, #aexp, aexp1, aexp2, nombre de fichero.
Ejemplo: 100 XIO 18, #6, 0, 0, "S:".

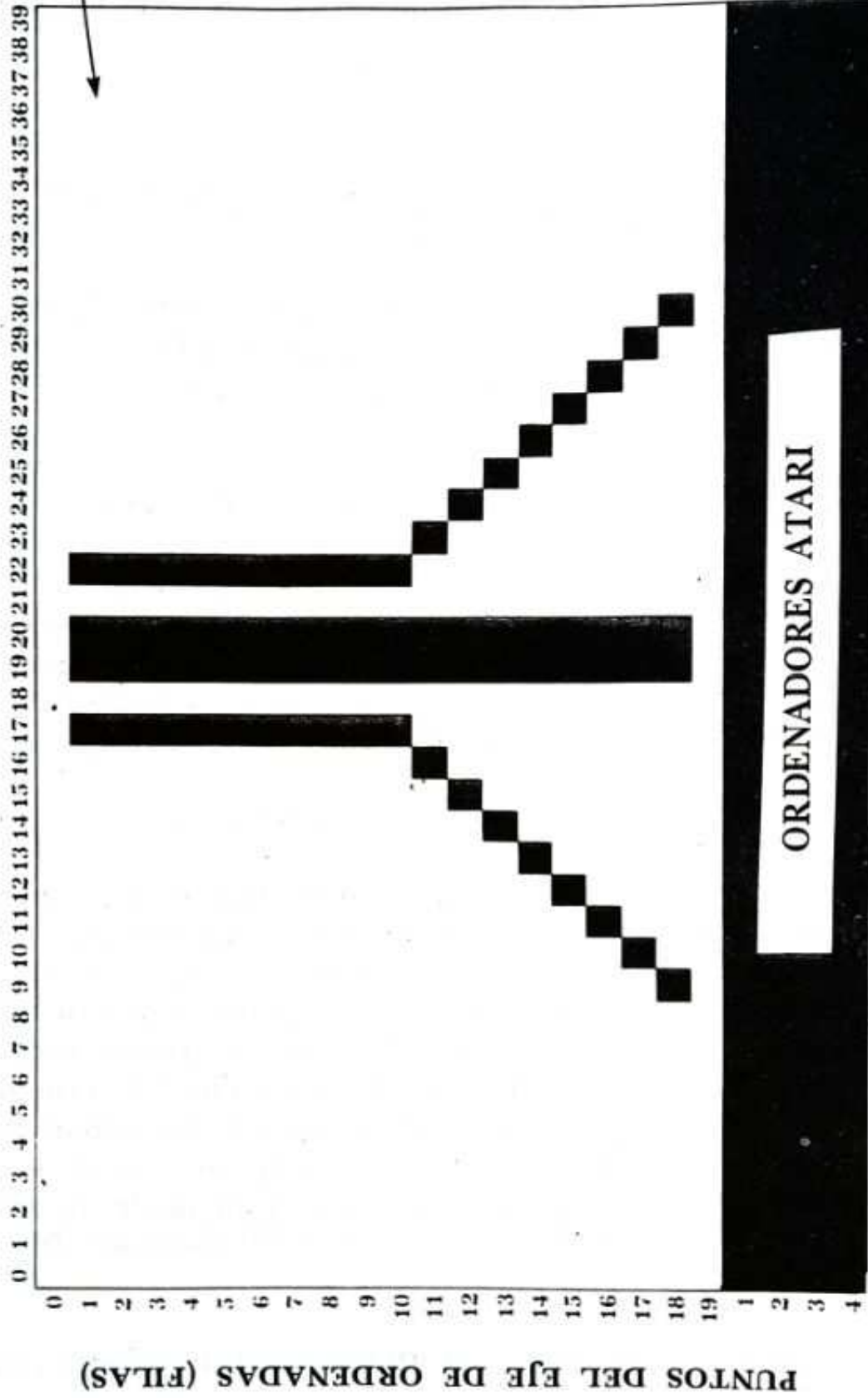
La aplicación especial de la declaración XIO rellena un área en la pantalla entre los puntos y las líneas trazadas con un valor de color no-cero. Se utiliza un cero para aexp1 y aexp2.

Los siguientes pasos ilustran el proceso de relleno:

1. PLOT la esquina derecha inferior (punto 1).
2. DRAWTO (dibuja) la esquina derecha superior (punto 2). Este traza el borde derecho que ha de ser relleno.
3. DRAWTO la esquina izquierda superior (punto 3).
4. POSITION (posiciona) el cursor en la esquina izquierda inferior (punto 4).
5. POKE la dirección 765 con los datos de relleno en color (1, 2 ó 3).
6. Este método es utilizado para rellenar cada línea horizontal, de arriba hacia abajo del área especificado. El relleno comienza en la izquierda y procede a través de la línea hacia la derecha hasta que llegue a un punto que contenga datos no-cero. Esto significa que el relleno no puede ser utilizado para cambiar un área que haya sido relleno con un valor no-cero, puesto que el relleno cesaría. La orden de relleno entrará en un lazo infinito si un relleno con datos cero es intentado en una línea que no tiene puntos de no-cero. Las teclas **BREAK** o **RESET** pueden ser utilizadas para detener el relleno si esto ocurre.

El siguiente programa crea una figura y la rellena con un dato de

PUNTOS DEL EJE DE ABCISAS (COLUMNAS)



PARA LA FIGURA 9.3

color 3. Observe que la orden XIO dibuja las líneas de la parte izquierda e inferior 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, #6,0,0, "S:"
90 GOTO 90
```

Figura 9.4. Programa de "Relleno".

Asignando colores a los caracteres en Modos de Texto 1 y 2

Este procedimiento describe el método de asignar colores al conjunto de caracteres Atari. Primero, busque el número del carácter en la Tabla 9-6. Luego, vea la Tabla 9-7 para obtener la conversión de ese número requerido para asignar un registro de color al mismo.

Ejemplo: Asigne al SETCOLOR 0 la minúscula "r" en el Modo 2, cuyo color es determinado por el registro 0.

1. En la Tabla 9-6, busque la columna y número para "r" (114-columna 4).
2. Utilizando la Tabla 9-7, localice la columna 4. La conversión se realiza restando 32 al número del carácter ($114 - 32 = 82$).
3. POKE la dirección de la Base del carácter (CHBAS) con el 226 para especificar las letras minúsculas o los caracteres gráficos especiales; ej.,

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

Para volver a las letras mayúsculas, números y signos de puntuación, POKE CHBAS con 224.

4. Una declaración PRINT utilizando el número convertido (82) asigna la minúscula "r" al SETCOLOR en el Modo 2 (Ver la Tabla 9-5).

TABLA 9.5. TABLA DE MODOS, SETCOLOR Y DE COLORES

Colores de ausencia	Modo o condición	SETCOLOR (aexp1) N.º de Registro	COLOR (aexp)	Descripción y comentarios
AZUL CLARO AZUL OSCURO NEGRO	MODO 0 Y TODAS LAS VENTANAS DE TEXTO	0 1 2 3 4	Los datos COLOR realmente determinan el carácter que ha de ser trazado	- Luminosidad de carácter. El mismo color que el fondo. - Borde.
NARANJA VERDE CLARO AZUL OSCURO ROJO NEGRO	MODOS 1 y 2 Modos de texto	0 1 2 3 4	Los datos COLOR realmente determinan el carácter que ha de ser trazado	Carácter. Carácter. Carácter. Carácter. Carácter de fondo, borde.
NARANJA VERDE CLARO AZUL OSCURO NEGRO	MODOS 3, 5 y 7 (Modos de 4-colores)	0 1 2 3 4	1 2 3 - 0	Punto gráfico. Punto gráfico. Punto gráfico. - Punto gráfico (ausencia de fondo), borde.
NARANJA NEGRO	MODOS 4 y 6 Modos de 2-color 4	0 1 2 3 0	1 - - -	Punto gráfico. - - - Punto gráfico (ausencia de fondo), borde.
VERDE CLARO AZUL OSCURO NEGRO	MODO 8 1 color 2-luminosidad	0 1 2 3 4	- 1 0 - -	- Punto gráfico de luminosidad (mismo color que fondo) Punto gráfico (ausencia de fondo). - Borde

Caracteres de Control Gráfico

Estos caracteres son producidos cuando la tecla CTRL es oprimida con las teclas alfabéticas que se muestran en la siguiente tabla. Estos caracteres pueden ser utilizados para trazar diseños, dibujos, etc., en el Modo 0 y en los Modos 1 y 2 si el CHBAS es cambiado.

TABLA 9.6. CONJUNTO DE CARACTERES INTERNOS.

Columna 1		Columna 2		Columna 3		Columna 4									
#	CHR	#	CHR	#	CHR	#	CHR								
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	
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 ser precedidos por un escape, CHR\$(27), para ser imprimidos.

TABLA 9.7. ASIGNACIONES DE COLOR/CARACTER

		Conversión 1	Conversión 2	Conversión 3	Conversión 4
MODO 0	SETCOLOR 2	# + 32	# + 32	# - 32	NINGUNA
		POKE 756,224		POKE 756,226	
MODO 1	SETCOLOR 0	# - 32	# + 32	# - 32	# - 32
	SETCOLOR 1	NINGUNO	# + 64	# - 64	NINGUNO
MODO 2	SETCOLOR 2	# + 160	# + 160	# + 96	# + 96
	SETCOLOR 3	# + 128	# + 192	# + 64	# + 128

10 SONIDOS Y CONTROLADORES DE JUEGOS

Esta sección describe la declaración utilizada para generar notas musicales y sonidos por medio del sistema de audio del monitor de la televisión. Pueden ser "tocados" simultáneamente hasta cuatro sonidos diferentes creando así una armonía. Esta declaración de SOUND también puede ser utilizada para simular explosiones, silbidos y otros interesantes efectos de sonido. Las demás órdenes descritas en esta sección tratan con las funciones utilizadas para manipular el teclado, "joystick" (control de palanca), y los controles de raqueta, "paddle". Estas funciones permiten que estos controladores puedan ser enchufados y utilizados en los programas BASIC para juegos, etc.

Las órdenes y las funciones descritas en esta sección son:

SOUND

PADDLE
PTRIG

STICK
STRIG

SOUND
(SO.)

Formato: SOUND aexp1, aexp2, aexp3, aexp4.
Ejemplo: 100 SOUND 2, 204, 10, 12.

La declaración SOUND hace que la nota especificada comience a tocar en cuanto la declaración es ejecutada. La nota continuará sonando hasta que se encuentre con otra declaración SOUND con el mismo aexp1 o una declaración END. Esta orden se puede utilizar en el Modo Directo o en el Modo Diferido.

Los parámetros del SOUND están descritos a continuación:

aexp1 = *Canal*. Puede ser de 0 a 3, pero cada canal requiere una declaración SOUND por separado.

aexp2 = *Timbre*. Puede ser cualquier número entre el 0 y el 255. Cuanto mayor sea el número, más bajo es el timbre. La Tabla 10-1 define los números de los timbres para las distintas notas musicales oscilando desde dos octavas arriba del DO central a una octava debajo del DO central.

aexp3 = *Distorsión*. Pueden ser números pares entre el 0 y el 14. Es utilizado para crear efectos de sonido. Un 10 es utilizado para crear un tono "puro"; en cambio, un 12 da un sonido de zumbido muy interesante. Un sonido de zumbido puede ser producido utilizando dos órdenes SOUND separadas con el valor de distorsión aexp3 alternando entre el 0 y el 1. Un valor de 1 es utilizado para forzar la salida al altavoz utilizando un volumen especificado (ver aexp4). El resto de los números son utilizados para otros efectos especiales, generación de ruidos y uso experimental.

aexp4 = *Control de Volumen*. Puede ser entre 1 y 15. Utilizando un 1 crea un sonido que es casi inaudible, en cambio un 15 es muy fuerte. Un valor de 8 es considerado como normal. Si más de una declaración de sonido está siendo utilizada, el volumen total no debe exceder de 32. Esto crearía un desagradable sonido "cortante". Con los valores de las notas en la Tabla 10-1 se ha teclado el siguiente programa:

```
10 READ A
20 IF A=256 THEN END
30 SOUND 0,A,10,10
40 FOR W=1 TO 400:NEXT W
50 PRINT A
60 GOTO 10
70 END
80 DATA 29,31,35,40,45,47,53,60,64,72,81
90 DATA 91,96,108,121,128,144,162,182
100 DATA 193,217,243,256
```

Figura 10-1. Programa de escala musical.

Observe que la declaración DATA en la línea 100 termina con un 256, lo que está fuera del alcance designado. El 256 es utilizado como un marcador de fin de datos.

TABLA 10-1. TABLA DE LOS VALORES DE TIMBRE PARA LAS NOTAS MUSICALES

NOTAS ALTAS		C	29	
		B	31	
		A# Bb	33	
		A	35	
		G# Ab	37	
		G	40	
		F# Gb	42	
		F	45	
		E	47	
		D# E	50	
		D	53	
		C# Db	57	
		C	60	
		B	64	
		A# B	68	
		A	72	
		G# Ab	76	
NOTAS MEDIAS		G	81	
		F# Gb	85	
		F	91	
		E	96	
		D# Eb	102	
		D	108	
		C# Db	114	
		C	121	
		B	128	
		A# Bb	136	
		A	144	
		G# Ab	153	
		G	162	
		F# Gb	173	
		F	182	
	NOTAS BAJAS		D	193
			D# Eb	204
		D	217	
b		C# Db	230	
		C	243	

FUNCIONES DE CONTROLADORES DE JUEGO

La Figura 10-2 es una ilustración de los tres controladores utilizados con los Ordenadores Personales ATARI. Dichos controladores pueden ser acoplados directamente a los Ordenadores Personales ATARI.

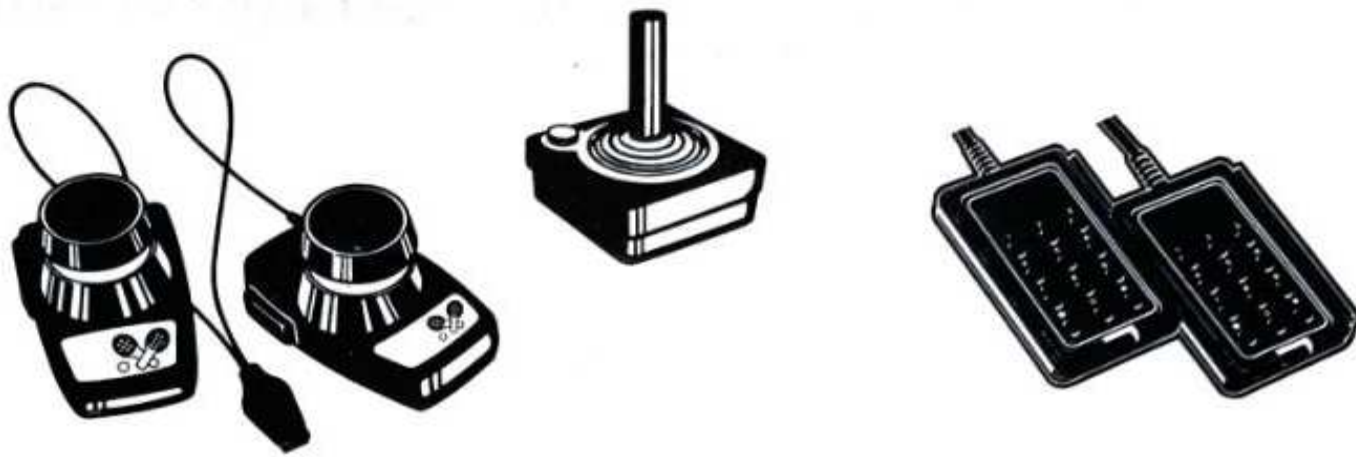


Figura 10-2. Controladores de juego.

PADDLE

Formato: PADDLE (aexp)

Ejemplo: PRINT PADDLE (3)

Esta función devuelve el status de un controlador enumerado particular. Los controladores de paleta son enumerados del 0 al 7 y de izquierda a derecha. Esta función puede ser utilizada con otras funciones u órdenes para “causar” más acciones como sonido, controles gráficos, etc.

Por ejemplo:

IF PADDLE (3) = 14 THEN PRINT “PALETA ACTIVA”

Observe que la función PADDLE devuelve un número entre el 1 y el 228, con el número aumentado en tamaño a la vez que el botón en el controlador es rotado en el sentido inverso de las agujas del reloj (rotando hacia la izquierda).

PTRIG

Formato: PTRIG (aexp)

Ejemplo: 100 IF PTRIG (4) = 0 THEN PRINT “S”

La función PTRIG devuelve un status de 0 si el botón rojo (gatillo) del controlador designado es oprimido. Si no, devuelve un valor de 1. El aexp tiene que ser un número entre el 0 y el 7 puesto que éste designa al controlador.

STICK

Formato: STICK (aexp)

Ejemplo: 100 PRINT STICK (3)

Esta función trabaja exactamente de la misma manera que la orden PADDLE, pero puede ser utilizada con el controlador de palanca de mando. Los controladores de palanca están numerados del 0 al 3 y de izquierda a derecha.

Controlador 1 = STICK (0)

Controlador 2 = STICK (1)

Controlador 3 = STICK (2)

Controlador 4 = STICK (3)

La Figura 10-3 demuestra los números que serán devueltos cuando el controlador de palanca es movido en cualquier dirección.

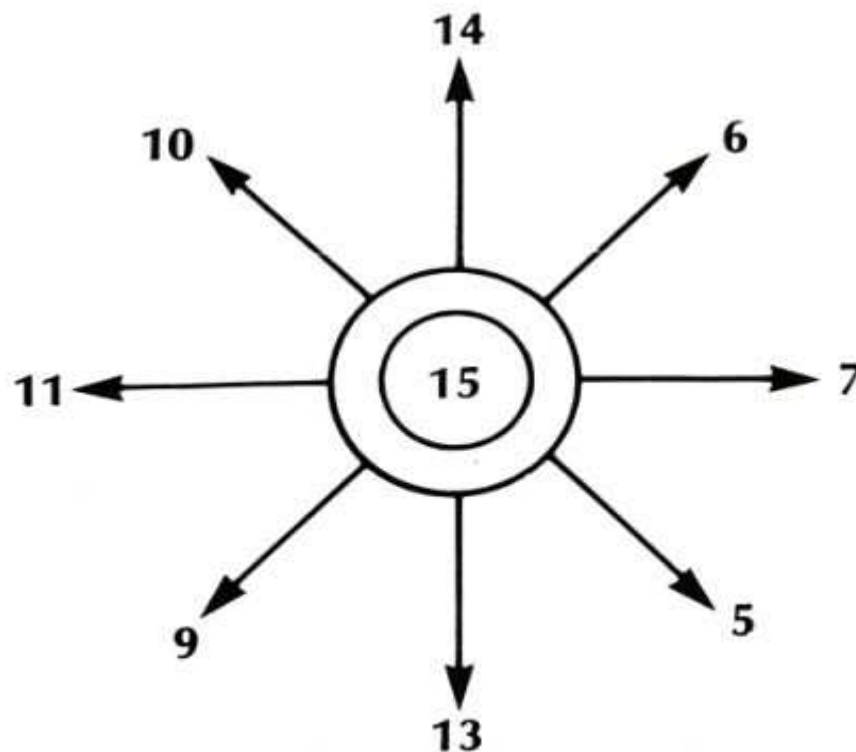


Figura 10-3. Movimiento de palanca de control.

STRIG

Formato: STRIG (aexp)

Ejemplo: 100 IF STRIG (\emptyset) = \emptyset THEN PRINT "S"

La función STRIG trabaja de la misma manera que la función PTRIG, excepto que STRIG se utiliza con la palanca de juegos.

11 TECNICAS DE PROGRAMACION AVANZADAS

Esta sección incluye consejos acerca de cómo incrementar la eficiencia de la programación, conservación de la memoria y la combinación de los programas en lenguaje de máquina con los programas del BASIC ATARI. Esta sección no contiene un conjunto de instrucciones para el chip microprocesador 6502, ni tampoco de instrucciones acerca de la programación en el lenguaje de máquina. Una compra adicional del CARTUCHO ATARI DEL ASSEMBLER EDITOR y un estudio cuidadoso del Manual del ASSEMBLER EDITOR de ATARI es altamente recomendado.

CONSERVACION DE MEMORIA

Estos consejos ofrecen maneras de conservar la memoria. Algunos de estos métodos hacen que los programas sean menos legibles y más difíciles de modificar, pero hay casos donde esto es necesario debido a las limitaciones de memoria.

1. En muchos ordenadores pequeños, eliminando los espacios en blanco entre palabras y caracteres como están escritos en el teclado ahorrará memoria. Esto no es verdad del Sistema de Ordenadores Personales ATARI, el cual quita los espacios sobrantes. Las declaraciones siempre están expuestas de la misma manera, indiferente a cuantos espacios fueron utilizados en la entrada del programa. Los espacios deben ser utilizados (igual que cuando está escribiendo en una máquina de escribir) entre palabras claves sucesivas y entre palabras claves y los nombres de las variables. Aquí hay un ejemplo:

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

Observe el espacio entre el IF y el A y entre el THEN y el PRINT. En la mayoría de los casos, una declaración será interpretada correctamente por el ordenador por más que todos los espacios sean omitidos, pero esto no siempre es verdad. Utilice el espaciado convencional.

2. Cada nuevo número de línea representa el principio de lo que es llamado una nueva línea lógica. Cada línea lógica toma 6 bytes de por encima (overhead), por más que se utiliza a capacidad plena o no. Agregando una declaración BASIC adicional y utilizando dos puntos (:) para separar cada par de declaraciones en la misma línea únicamente ocupa 3 bytes. Si necesita ahorrar memoria, evite programas como éste:

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

Y consolide líneas de esta manera:

```
10 X=Y+1 :Y=Y+1 :Z=X+Y :PRINT Z :GOTO 50
```

Esta consolidación ahorra 12 bytes.

3. También las variables y las constantes deben ser “administradas” para ahorrar. Cada vez que una constante (4, 5, 16, 3.1416, etc.) es utilizada, ocupa 7 bytes. Definir una nueva variable requiere 8 bytes más la longitud del nombre de la variable (en caracteres). Pero cada vez que es utilizada después de haber sido definida, ocupa únicamente 1 byte, indiferente de su longitud. Por tanto, si una constante (como 3.1416) es utilizada más de una vez en un programa, debe ser definida como una variable, el nombre de la variable debe ser utilizado a lo largo de todo el programa, por ejemplo.

```
10 PI=3.14159
20 PRINT "AREA DEL CIRCULO=RADIO CUADRADO POR";PI
```

4. Las cadenas literales (entre comillas) requieren 2 bytes más 1 byte por cada carácter (incluyendo todos los espacios) en la cadena.
5. Las variables en cadena ocupan 9 bytes cada una, más la longitud del nombre de la variable (incluyendo los espacios), más el espacio ocupado por la declaración DIM, más el tamaño de la cadena de por sí (1 byte por carácter) cuando es definido. Obviamente, la utilización de variables en cadena es muy costosa con respecto al RAM.
6. La definición de una nueva matriz requiere 15 bytes más la longitud del nombre de la variable de la matriz, más el espacio necesario para la decla-

ración DIM, más 6 veces el tamaño de la matriz (el producto del número de filas y el número de columnas). Por lo tanto, una matriz de 25 filas por 4 columnas requerirá 15, más aproximadamente 3 (por el nombre de la variable), más aproximadamente 10 (por la declaración DIM), más 6 veces 100 (por el tamaño de la matriz), o alrededor de 630 bytes.

7. Cada carácter después del REM ocupa un byte de memoria. Los comentarios ayudan a los programadores que están tratando de entender un programa, pero a veces es necesario quitar las declaraciones de comentarios para ahorrar memoria.
8. Las sub-rutinas pueden ahorrar memoria porque una sub-rutina y varias llamadas cortas ocupan menos espacio de memoria que duplicar el código varias veces. Pero por otro lado, una sub-rutina que es llamada únicamente una vez ocupa extra bytes para las declaraciones GOSUB y RETURN.
9. Los paréntesis ocupan un byte cada uno. Los paréntesis de más son una buena idea en algunos casos, pues hacen que una expresión sea más comprensible al programador. Sin embargo, quitando los paréntesis innecesarios y fiándose en la precedencia del operador se ahorrará varios bytes.

PROGRAMANDO EN EL LENGUAJE DE MÁQUINA

El lenguaje de máquina está escrito enteramente en un código binario. El Ordenador Personal ATARI contiene un microprocesador 6502 y es posible llamar a subrutinas en código máquina desde el BASIC utilizando la función USR. Las rutinas cortas pueden entonces ser tecleadas en un programa manualmente (si es necesario).

Antes de que vuelva al BASIC, la rutina en lenguaje ensamblador debe sacar el número de argumentos (N) de la pila mediante la instrucción PLA. Si el número no es 0, entonces todos los argumentos de entrada deben sacarse de la pila usando también PLA (Ver Figura 6-1).

La subrutina debe finalizar colocando el byte de menor peso del resultado en la posición 212 (decimal), y luego volver al BASIC utilizando una instrucción RTS (Regreso de la Sub-rutina). El interpretador BASIC convertirá el número binario de 2-byte almacenado en las ubicaciones 212 y 213 en un número entero entre el 0 y el 65535 en un formato de coma flotante para obtener el valor devuelto por la función USR.

La función ADR puede ser utilizada para pasar los datos que están almacenados en tablas o en cadenas a una sub-rutina en el lenguaje de máquina. Utilice la función ADR para obtener la dirección de una tabla o de una cadena, y luego utilice esta dirección como una de los argumentos de entrada de la función USR.

El siguiente programa, Cargador de Código Hexadecimal, provee los medios para entrar códigos hexadecimales, convirtiendo cada número hexadecimal en decimal, y almacenando al número decimal en una tabla. La tabla es entonces ejecutada como una subrutina en lenguaje ensamblador. Se utiliza un vector (array) como almacenamiento de la rutina en memoria. (*Nota:* Lenguaje de Montaje es el Assembler.)

1. Para utilizar este programa, primero tecléelo. Después, almacénelo en disco o cassette para su posterior utilización.

```
10 GR.0:PRINT "PROGRAMA DE CARGADOR-HEXA":
  PRINT
20 REM ALMACENA EQUIVALENTE DECIMALES EN LA
  TABLA A, LO
30 REM SACAR EN LAS 'DECLARACIONES DATA' IMPRI-
  MIDAS EN
40 REM EL NUMERO DE LINEA 1500.
50 REM USUARIO ENTONCES UBICA EL CURSOR EN LA
  LINEA DE
60 REM SALIDA IMPRESA, PULSA RETURN, Y TECLEA
  EL RESTO
70 REM DEL PROGRAMA BASIC INCLUYENDO LA DE-
  CLARACION USR
80 DIM A(50), HEX$(5)
90 REM ENTRADA, CONVERSION, Y ALMACENAMIENTO
  DE DATOS
100 N=0:PRINT "TECLE UN CODIGO-HEXA. SI EL ULTI-
  MO ESTA
110 PRINT "DENTRO, TECLEE: 'HECHO.'";
120 INPUT HEX$
130 IF HEX$="HECHO" THEN N=999:GOTO 180
140 FOR I=1 TO LEN(HEX$)
150 IF HEX$(I,I) <="9" THEN N=N*16 + VAL(HEX$(I,I)):
  GOTO 170
160 N=N*16 + ASC(HEX(I,I)) - ASC("A") + 10
170 NEXT I
180 PRINT N:C=C+1:A(C)=N
190 IF N<>999 THEN 100
200 REM IMPRIMA LINEA DE DATOS EN LINEA 1500
210 GRAPHICS 0: PRINT "1500 DATA";
215 C=0
220 C=C+1: IF A(C) = 999 THEN PRINT "999":STOP
```

```

230 PRINT A(C);",,";
240 A(C)=0
250 GOTO 220
260 PRINT "PONGA NO. CORRECTO DE BYTES HEXA.
      EN LA LINEA 1000":STOP:REM LINEA TRAP
999 REM ** MODULO DE EJECUCION **
1000 CLR :BYTES=0
1010 TRAP 260:DIM E$(1), E(INT(BYTES/6)+1)
1020 FOR I=1 TO BYTES
1030 READ A:IF A>255 THEN GOTO 1050
1040 POKE ADR(E$)+I,A
1050 NEXT I
1060 REM PARTE DEL PROGRAMA DEL USUARIO CON-
      TINUA

```

Figura 11.1. Programa de entrada de cargador HD.

2. Ahora agregue la parte del lenguaje BASIC de su programa comenzando en la línea 1070 incluyendo la función **USR** que llama a la sub-rutina de lenguaje de máquina (Ver el ejemplo siguiente).
3. Cuente el número total de códigos hexadecimales que han de ser entradas y entre este número en la línea 1000 cuando sea requerido. Si otro número está ya entrado, simplemente reemplácelo.
4. Ejecute el programa y entre los códigos hexadecimales de la sub-rutina del nivel de máquina oprimiendo la tecla **RETURN** después de cada entrada. Después de la última entrada, teclee **HECHO** y oprima la tecla **RETURN**.
5. Ahora la línea de datos 1500 se visualiza en la pantalla. No será entrada al programa hasta que el cursor sea movido a la línea **DATA** y la tecla **RETURN** sea oprimida.
6. Agregue una línea de programa **5 GOTO 1000** para evitar el cargador de código hexadecimal (o suprima el cargador código hexadecimal hasta la línea 250). Ahora guarde el programa completo utilizando el **CSAVE** o el **SAVE**. Es importante hacer esto antes de ejecutar la parte del programa que contiene la llamada **USR**. Un error en una rutina del lenguaje de máquina puede bloquear el sistema. Si el sistema se cuelga, oprima

la tecla **RESET**. Si no responde el sistema, encienda y apague el ordenador, y cargue de nuevo el programa, y corríjalo.

Nota: Este método únicamente trabaja rutinas de lenguaje de máquina reubicables.

Los dos ejemplos siguientes del programa pueden ser tecleados en el programa Cargador de Códigos Hexadecimales. El primer programa imprime NADA SE ESTA MOVIENDO mientras que el programa de máquina cambia de colores. El segundo programa visualiza un diseño gráfico BASIC y luego cambia de colores.

```
1070 GRAPHICS 1+16
1080 FOR I=1 TO 6
1090 PRINT #6;"NADA SE ESTA MOVIENDO"
1100 PRINT #6;"NADA SE ESTA MOVIENDO"
1110 PRINT #6;"NADA SE ESTA MOVIENDO"
1120 PRINT #6;"NADA SE ESTA MOVIENDO"
1130 NEXT I
1140 Q=USR(ADR(E$)+1)
1150 FOR I=1 TO 25:NEXT I:GOTO 1140
```

Después de teclear este programa, compruebe que la línea 1000 lee:

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

Teclee RUN y pulse la tecla **RETURN**.

Ahora teclee los códigos hexadecimales como están demostrados 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
```

Cuando esté completado, teclee HECHO y oprima la tecla **RETURN**. Ahora posicione el cursor después de la última entrada (999) en la línea DATA y oprima la tecla **RETURN**.

Ahora realice el programa tecleando GOTO 1000 y oprimiendo la tecla **RETURN**, o si se ha agregado la línea 5, teclee RUN y oprima la tecla **RETURN**. Oprima la tecla **BREAK** para detener el programa y suprimir la línea 5. En el segundo programa que sigue debería ser entrado en el lugar del programa de NADA SE ESTA MOVIENDO. Asegúrese de comprobar la cuenta de los BYTES = ---- en la línea 1000. Siga los pasos desde el 2 al 6.

```
1070 GRAPHICS 7+16
1080 SETCOLOR 0,9,4
1090 SETCOLOR 1,9,8
1100 SETCOLOR 2,9,4
1110 CR=1
1120 FOR X=0 TO 159
1130 COLOR INT(CR)
1140 PLOT 80,0
1150 DRAWTO X,95
1160 CR=CR+0.125
1170 NEXT X
1180 X=USR(ADR(E$)+1)
1190 FOR I=1 TO 15:NEXT I
1200 GOTO 1200
```

Teclee RUN y pulse la tecla **RETURN**.

Teclee 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

Cuando esté completo, teclee HECHO y oprima la tecla **RETURN**. Ahora ubique el cursor después de la última entrada (999) en la línea DATA y oprima

la tecla **RETURN**. Ahora realice el programa tecleando **GOTO 1000** y pulsando la tecla **RETURN**, o agregue la línea 5 **GOTO 1000** y teclee **RUN** y oprima **RETURN**. Oprima la tecla **BREAK** para parar el programa y suprimir la línea 5. La Figura 11.2 ilustra una sub-rutina de montaje utilizada para rotar los colores que pueden llegar a serle útil. Está incluido aquí para información de quien lo utilice:

SUBRUTINA DE MONTAJE PARA ROTAR COLORES					
Dirección	Código de objeto	N.º de línea	Etiqueta	Mnem.	Datos
		0100			Rutina para rotar los datos de
		0110			COLOR de un registro a otro.
		0120			4 colores son rotados.
		0130			
		0140			Sistema Operativo de Dirección.
02C4		0150			COLOR 0 = \$02C4
02C5		0160			COLOR 1 = \$02C5
02C6		0170			COLOR 2 = \$02C6
02C7		0175			COLOR 3 = \$02C7
		0180			
		0190		* =	\$6000 Dirección del comienzo de máquina*.
6000	6B	0200		PLA	Salto de Pila.
6001	A200	0210		LDX	#0 Cero al registro X.
6003	ACC402	0220		LDY	COLOR0 Salvar COLOR 0.
6006	BDC502	0230	LAZO	LDA	COLOR1,X
6009	9DC402	0240		STA	COLOR0,X
600C	E8	0250		INX	Incrementa Reg. X (Suma uno).
600D	E002	0260		CPX	#3 Compara contenidos de Reg. X con 2.
600F	90F5	0270		BCC	LAZO Lazo si los contenidos del Reg. X son menos que 2.
6011	8CC602	0280		STY	COLOR3 Salve COLOR0 en COLOR3.
6014	60	0290		RTS	Vuelva de la sub-rutina del nivel de máquina.
Assembler Imprime esto		Esta porción es la información de origen que el programador entra utilizando un Cartucho Assembler de ATARI			
# = Indica datos de origen; * = rutina reubicada; \$ = N.º HD.					

Figura 11-2. Subrutina Assembler para rotar colores.

APENDICE A

DIRECTORIO ALFABETICO DE PALABRAS RESERVADAS DEL BASIC

Nota: El espacio es obligatorio después de teclear todas las abreviaciones.

PALABRA RESERVADA	ABREVIACION	
ABS		Esta función devuelve el valor (sin signo) de una variable o una expresión.
ADR		Esta función devuelve la dirección de memoria, de inicio de una cadena de caracteres.
AND		Operador Lógico: La expresión es cierta sólo si las subexpresiones relacionadas con AND son ciertas.
ASC		Esta función de cadena devuelve el valor numérico de una cadena de carácter único.
ATN		Esta función devuelve el arco tangente en radianes o degradianes, de un número o una expresión.
BYE	B.	Sale del BASIC, y pone al ordenador en modo "Autoensayo".
CLOAD	CLOA.	Carga en la memoria RAM un programa grabado en cinta.
CHR\$		Esta función de cadena devuelve una cadena de carácter único, equivalente a un número valuado entre 0 y 255 en el Código ATACIL.
CLOG		Esta función devuelve el logaritmo en base 10 de una expresión.
CLOSE	CL.	Esta es una declaración de Entrada y Salida

PALABRA RESERVADA	ABREVIACION	
----------------------	-------------	--

		(I/O) utilizada para cerrar un fichero al final de las operaciones de I/O.
CLR		Es lo contrario de DIM. Borra todas las variables (numéricas, alfanuméricas y matriciales).
COLOR	C.	Este elige el registro de colores que ha de ser utilizado en los trabajos gráficos en color.
COM		Igual que el DIM.
CONT	CON.	Continuar. Esto hace que un programa recommence la ejecución en la próxima línea siguiendo la utilización de la tecla BREAK o al encontrarse con un STOP .
COS		Esta función devuelve el coseno de la variable o expresión (en grados o radianes).
CSAVE		Almacena el programa que está en memoria en una cinta de cassette.
DATA	D.	Este forma parte de la combinación READ/DATA . Es utilizada para identificar los artículos que lo siguen (los cuales deben ser separados por comas) como artículos de los datos individuales.
DEG	DE.	La declaración DEG informa al ordenador que ha de desempeñar las funciones trigonométricas en grados en vez de en radianes. (Ausenta a radianes.)
DIM	DI.	Este reserva la cantidad especificada de memoria para la matriz, tabla o cadena. Todas las variables en cadena, tablas y matrices deben ser dimensionadas con una declaración DIM .
DOS	DO.	Son siglas reservadas para operadores de discos. Hace que el Menu se visualice.

PALABRA RESERVADA	ABREVIACION	
DRAWTO	DR.	Esta dibuja una línea recta entre un punto trazado y un punto especificado.
END		Esta detiene la ejecución del programa; cierra los ficheros; y apaga los sonidos. Se puede recomenzar el programa utilizando CONT . <i>Nota:</i> El END puede ser utilizado más de una vez en un programa.
ENTER	E.	Esta es una orden de I/O utilizada para almacenar datos o programas de una forma no señalizada.
EXP		Esta función devuelve $e(2.7182818)$ elevado a la potencia especificada.
FOR	F.	Esta es utilizada con el NEXT para establecer los lazos FOR/NEXT . Introduce la extensión en donde ha de operar la variable del lazo durante la ejecución del mismo.
FRE		Esta función devuelve la cantidad restante de la memoria utilizada en bytes.
GET	GE.	Es utilizado mayormente con operaciones de discos para entrar un único byte de datos.
GOSUB	GOS.	Es para ramificarse a una subrutina comenzando en un número de línea especificado.
GOTO	G.	Es una ramificación incondicional a un número de línea especificado.
GRAPHICS	GR.	Especifica cuál de los ocho gráficos va a ser utilizado. El GR.0 puede ser utilizado para despejar la pantalla.
IF		Esta es utilizada para ocasionar una ramificación condicional o para ejecutar otra declaración que se encuentre en la misma línea (solamente si la primera expresión es verdadera).

PALABRA RESERVADA	ABREVIACION	
INPUT	I.	Esta hace que el ordenador pida entradas del teclado. La ejecución continúa únicamente cuando la tecla RETURN es oprimida después de entrar datos.
INT		Esta función devuelve al próximo número entero debajo del valor especificado. El redondeo es siempre hacia abajo, aun cuando el número sea negativo.
LEN		Es una función en cadena que devuelve la longitud de la cadena especificada en bytes o en caracteres (un byte contiene un carácter).
LET	LE.	Asigna un valor a un nombre de variable especificada. El LET es opcional en el BASIC ATARI , y puede ser simplemente suprimido.
LIST	L.	Expone, o si no visualiza un listado del programa.
LOAD	LO.	Entrada desde el disco, etc. al ordenador.
LOCATE	LOC.	Gráficos; almacena, en una variable especificada, el valor que controla un punto gráfico especificado.
LOG		Esta función devuelve el logaritmo natural de un número.
LPRINT	LP.	Es una orden para la impresora para que imprima el mensaje especificado.
NEW		Borra todo el contenido del RAM utilizable.
NEXT	N.	Hace que un lazo FOR/NEXT finalice o continúe dependiendo de las variables o expresiones particulares. Todos los lazos son ejecutados por lo menos una vez.
NOT		Un "1" es devuelto únicamente si la expresión

**PALABRA
RESERVADA**

ABREVIACION

		NO es verdadera. Si es verdadera un "0" es devuelto.
NOTE	NO.	Ver el Manual DOS/FMS... utilizado únicamente en operaciones de discos.
ON		Utilizado con el GOTO o el GOSUB para propósitos de ramificaciones. Ramificaciones múltiples a distintos números de línea son posibles dependiendo en el valor de la variable o expresión ON.
OPEN	O.	Abre el fichero especificado para las operaciones de entrada y salida.
OR		Es un operador lógico, utilizado entre dos expresiones. Si cualquiera de las dos es verdadera, se devuelve un "1". Un "0" resulta únicamente si las dos son falsas.
PADDLE		Esta función devuelve la posición de la raqueta de control.
PEEK		Esta función devuelve la forma decimal del contenido de la posición especificada de memoria (RAM ó ROM).
PLOT	PL.	Hace que un único punto sea trazado en la posición X,Y especificada.
POINT	P.	Es utilizada únicamente con operaciones de discos.
POKE	POK.	Introduce al byte especificado en la posición especificada de memoria. Puede ser utilizada únicamente con el RAM. No intente POKE ROM u obtendrá un error.
POP		Quita la variable del lazo de la pila de GOSUB. Es utilizada cuando la salida del lazo está he-

PALABRA RESERVADA	ABREVIACION	
		cha de una manera que no sea la manera normal.
POSITION	POS.	Fija el cursor en la posición especificada de la pantalla.
PRINT	Pr. o?	Es una orden de I/O que ocasiona salidas del ordenador al dispositivo de salida especificada.
PTRIG		Esta función devuelve el status del botón-gatillo en los controladores de juego.
PUT	PU.	Esta ocasiona la salida de un único byte de datos del ordenador al dispositivo especificado.
RAD		Esta especifica que la información está en radianes en vez de en grados cuando está utilizando las funciones trigonométricas. La ausencia es a RAD (Ver DEG).
READ	REA.	Esta lee los artículos próximos en la lista DATA y asigna a las variables especificadas.
REM	R.	Comentarios. Esta declaración no hace nada, pero los comentarios pueden ser imprimidos dentro del listado del programa para una futura referencia por el programador. Las declaraciones que se encuentran en una línea que comienza con REM no son ejecutadas.
RESTORE	RES.	Esta permite que el DATA sea leído más de una vez.
RETURN	RET.	Vuelve de la sub-rutina a la declaración que inmediatamente sigue a la declaración en donde apareció el GOSUB.
RND		Esta función devuelve un número al azar entre el 0 y 1, pero nunca 1.
RUN	RU.	Esta realiza el programa. Fija las variables normales en 0, des-dimensiona tablas y cadenas.

PALABRA RESERVADA	ABREVIACION	
SAVE	S.	Esta declaración de I/O hace que los datos o programas sean grabados en el disco bajo el nombre de fichero proveído con el SAVE .
SETCOLOR	SE.	Esta almacena los datos de luminosidad y tono de color en un particular registro de color.
SGN		Esta función devuelve un +1 si el valor es positivo, un 0 si es 0, un -1 si es negativo.
SIN		Esta función devuelve el seno trigonométrico del valor dado en DEG o en RAD .
SOUND	SO.	Esta controla el registro, timbre del sonido, distorsión y volumen de un tono o nota.
SQR		Esta función devuelve la raíz cuadrada del valor especificado.
STATUS	ST.	Esta llama a la rutina status para un dispositivo especificado.
STEP		Esta es utilizada con el FOR/NEXT . Determina la cantidad que ha de ser saltada entre cada par de valores de las variables de los lazos.
STICK		Esta función devuelve la posición del STICK controlador de juegos.
STRIG		Esta función devuelve un 1 si el botón del STICK no está presionado, 0 si está presionado.
STOP	STO.	Hace que se detenga la ejecución del programa. Pero no cierra ficheros ni apaga sonidos.
STR\$		Esta función devuelve un carácter cadena igual al valor numérico dado. Por ejemplo, STR\$(65) devuelve al 65 como una cadena.
THEN		Es utilizado con el IF : Si la expresión es verdadera, las declaraciones THEN son ejecuta-

**PALABRA
RESERVADA**

ABREVIACION

das. Si la expresión es falsa, el control pasa a la próxima línea.

TO

Esta es utilizada con el **FOR** como en "FOR X = 1 to 10". Separa las expresiones del alcance del lazo.

TRAP

T.

Esta toma el control del programa por si hay un error de **INPUT** y dirige la ejecución a un número de línea especificada.

USR

Esta función devuelve los resultados de una sub-rutina de lenguaje de máquina.

VAL

Esta función devuelve el valor numérico equivalente de una cadena.

XIO

X

Esta es una declaración general de I/O utilizada con operaciones de discos (Ver el *Manual DOS/FMS*) y en trabajos gráficos (rellenar).

APENDICE B

MENSAJES DE ERROR

NUMERO DE CODIGO DE ERROR	MENSAJE DEL CODIGO DE ERROR
2	Memoria insuficiente: No hay memoria suficiente para poder almacenar la declaración, el nuevo nombre de la variable, o para DIMensionar una nueva cadena de nombre de variable.
3	Error de valor: Un valor que se espera que sea positivo es negativo, o un valor que se espera que esté dentro de una extensión especificada no lo está.
4	Demasiadas variables: Un máximo de 128 nombres distintos de variables está permitido.
5	Error de la longitud de la cadena: Intentó almacenar más allá de la longitud dimensionada de la cadena.
6	Error por falta de datos: La declaración READ requiere más artículos de datos que aquellos suministrados por las declaraciones DATA.
7	Número mayor que 32767: El valor no es un número entero positivo o es mayor que 32767.
8	Error en la declaración de entrada (Input): Intentó entrar un valor no numérico en una variable numérica.
9	Error de DIM de la colección o de cadena: El tamaño DIM es mayor que 32767 o una referencia de tabla (colección)/matriz está fuera del alcance del tamaño dimensionado, o la tabla/matriz o cadena ya ha sido DIMensionada, o una referencia ha sido hecha a una tabla o cadena no dimensionada.
10	Desborde de la pila de argumentos: Hay demasiados GOSUB o hay una expresión demasiado larga.

**NUMERO
DE CODIGO
DE ERROR**

MENSAJE DEL CODIGO DE ERROR

- 11 **Error de desborde o insuficiencia de la coma flotante:** Intentó dividir por cero o se refirió a un número mayor que $1 * 10^{(93)}$ o más pequeño que $1 * 10^{(-99)}$.
- 12 **Línea no hallada:** Un GOSUB, GOTO ó THEN es referido a un número de línea no existente.
- 13 **Declaración FOR no correspondida:** Un NEXT fue encontrado sin un FOR previo, o las declaraciones FOR/NEXT anidadas no corresponden correctamente. (El error es avisado en la declaración NEXT, no en la FOR.)
- 14 **Error de una línea demasiado extensa:** La declaración es demasiado larga o compleja para que lo maneje el BASIC.
- 15 **Error de una línea GOSUB o FOR suprimido:** Fue encontrada una declaración de NEXT o RETURN y el FOR o el GOSUB correspondiente ha sido suprimido desde el último RUN.
- 16 **Error de RETURN:** Un RETURN fue encontrado sin su correspondiente GOSUB.
- 17 **Error de Garbage (Basura):** La ejecución de residuo fue intentada. El código de este error puede indicar un problema de Hardware, pero también puede ser el resultado del uso fallido del POKE. Intente teclear NEW o apagar y encender el ordenador, luego entre de nuevo el programa sin ninguna orden POKE.
- 18 **Carácter no válido de la cadena:** La cadena no comienza con un carácter válido, o la cadena en la declaración VAL no es una cadena numérica.
- Nota:* Los que siguen a continuación son errores de I/O que resultan durante la utilización de unidades de disco, impresoras u otros dispositivos o accesorios. Más información es dada con el hardware auxiliar.
- 19 **El programa LOAD es demasiado extenso:** No queda suficiente memoria para completar el LOAD.

**NUMERO
DE CODIGO
DE ERROR**

MENSAJE DEL CODIGO DE ERROR

- 11 Error de desborde o insuficiencia de la coma flotante: Intentó dividir por cero o se refirió a un número mayor que $1 * 10^{93}$ o más pequeño que $1 * 10^{-99}$.
- 12 Línea no hallada: Un GOSUB, GOTO ó THEN es referido a un número de línea no existente.
- 13 Declaración FOR no correspondida: Un NEXT fue encontrado sin un FOR previo, o las declaraciones FOR/NEXT anidadas no corresponden correctamente. (El error es avisado en la declaración NEXT, no en la FOR.)
- 14 Error de una línea demasiado extensa: La declaración es demasiado larga o compleja para que lo maneje el BASIC.
- 15 Error de una línea GOSUB o FOR suprimido: Fue encontrada una declaración de NEXT o RETURN y el FOR o el GOSUB correspondiente ha sido suprimido desde el último RUN.
- 16 Error de RETURN: Un RETURN fue encontrado sin su correspondiente GOSUB.
- 17 Error de Garbage (Basura): La ejecución de residuo fue intentada. El código de este error puede indicar un problema de Hardware, pero también puede ser el resultado del uso fallido del POKE. Intente teclear NEW o apagar y encender el ordenador, luego entre de nuevo el programa sin ninguna orden POKE.
- 18 Carácter no válido de la cadena: La cadena no comienza con un carácter válido, o la cadena en la declaración VAL no es una cadena numérica.
- Nota:* Los que siguen a continuación son errores de I/O que resultan durante la utilización de unidades de disco, impresoras u otros dispositivos o accesorios. Más información es dada con el hardware auxiliar.
- 19 El programa LOAD es demasiado extenso: No queda suficiente memoria para completar el LOAD.

**NUMERO
DE CODIGO
DE ERROR**

MENSAJE DEL CODIGO DE ERROR

- 20 El número de dispositivo es mayor: Que 7 o igual a 0.
- 21 Error en la carga del fichero: Intentaron cargar un fichero no cargable.
- 128 Aborto de la tecla **BREAK**: El que utilizó el ordenador oprimió la tecla **BREAK** durante una operación de I/O.
- 129 IOCB: Está ya abierto.
- 130 Dispositivo especificado no existente.
- 131 IOCB escritura únicamente: Orden **READ** a un dispositivo que solamente escribe (impresora).
- 132 Orden no válida: La orden no es válida para este dispositivo.
- 133 Dispositivo fichero no abierto: Ningún **OPEN** fue especificado para el dispositivo.
- 134 Mal número de IOCB: El número del dispositivo es ilegal.
- 135 Error de lectura de IOCB únicamente: Orden **WRITE** a un dispositivo de lectura únicamente.
- 136 EOF (final de fichero): El final de la lectura del fichero ha sido alcanzado. (*Nota: Este mensaje puede ocurrir cuando se está utilizando ficheros en cassette.*)
- 137 Registro truncado: Intentó leer un registro más largo de 256 caracteres.
- 138 Dispositivo fuera de tiempo: El dispositivo no responde.
- 139 Dispositivo **NAK**: El residuo está en el puerto en serie o una unidad de disco mala.
- 140 Bus en serie: Error en el enmarcado de la entrada **BUS** en serie.
- 141 Cursor fuera de alcance: Para un modo particular.

**NUMERO
DE CODIGO
DE ERROR**

MENSAJE DEL CODIGO DE ERROR






- 142 **Datos bus en serie exceden el marco.**
- 143 **Error de la comprobación de suma de los datos BUS en serie, en el marco.**
- 144 **Error del dispositivo DONE (byte done no válido): Intentó escribir en un disco con protección-de-escritura.**
- 145 **Error de la comprobación de lectura después de escribir (controlador de disco) o mal manejo del modo de pantalla.**
- 146 **Función no implementada: en el controlador.**
- 147 **Insuficiente RAM: para operar los modos gráficos elegidos.**
- 160 **Error del número de la unidad.**
- 161 **Demasiados ficheros abiertos (ningún sector de buffer (memoria) disponible).**
- 162 **Disco completo (ningún sector libre).**
- 163 **Error del sistema de datos I/O irrecuperable.**
- 164 **Número de fichero no correspondiente: Los ensambles de las unidades están desordenados.**
- 165 **Error de nombre de fichero.**
- 166 **Error de longitud del dato POINT.**
- 167 **Fichero bloqueado.**
- 168 **Orden no válida: Código especial de operaciones.**
- 169 **Guía de archivos llena: 64 ficheros.**
- 170 **Fichero no hallado.**
- 171 **POINT no válido.**

APENDICE C

CONJUNTO DE CARACTERES ATASCII

CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER INTERNACIONAL	CARACTER	CODIGO DECIMAL	CODIGO HEXADECIMAL	CARACTER INTERNACIONAL	CARACTER	CODIGO DECIMAL	CARACTER INTERNACIONAL	CODIGO HEXADECIMAL	CARACTER
0	0	á		13	D	ú		26	À	1A	
1	1	ù		14	E	ó		27		1B	
2	2	Ñ		15	F	Ö		28		1C	
3	3	É		16	10	Ü		29		1D	
4	4	ç		17	11	â		30		1E	
5	5	ô		18	12	û		31		1F	
6	6	ò		19	13	î		32		20	Space
7	7	ï		20	14	é		33		21	!
8	8	£		21	15	è		34		22	”
9	9	ï		22	16	ñ		35		23	#
10	A	ü		23	17	ê		36		24	\$
11	B	ä		24	18	â		37		25	%
12	C	ö		25	19	à		38		26	&

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	
94	5E	^	110	6E	n	126	7E	
95	5F	_	111	6F	o	127	7F	
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**




183 B7
184 B8
185 B9
186 BA
187 BB
188 BC
189 BD
190 BE
191 BF
192 C0
193 C1
194 C2
195 C3
196 C4
197 C5
198 C6

**CODIGO
DECIMAL** **CODIGO
HEXADECIMAL** **CARACTER**

199 C7
200 C8
201 C9
202 CA
203 CB
204 CC
205 CD
206 CE
207 CF
208 D0
209 D1
210 D2
211 D3
212 D4
213 D5
214 D6

**CODIGO
DECIMAL** **CODIGO
HEXADECIMAL** **CARACTER**

215 D7
216 D8
217 D9
218 DA
219 DB
220 DC
221 DD
222 DE
223 DF
224 E0
225 E1
226 E2
227 E3
228 E4
229 E5
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	 (Buzzer)
236	EC		245	F5		254	FE	 (Delete character)
237	ED		246	F6		255	FF	 (Insert character)
238	EE		247	F7				
239	EF		248	F8				

Ver el Apéndice H para ver el programa que realiza la conversión decimal/hexadecimal.

Notas:

1. ATASCII representa "ATARI ASCII". Las letras y números tienen el mismo valor como aquellos en el ASCII, pero algunos de los caracteres especiales son diferentes.
2. Excepto en donde esté demostrado, los caracteres del 128 al 255 son los colores inversos del 1 al 127.
3. Agregue 32 al código de mayúsculas para obtener el código de minúsculas para esa misma letra.

4. Para obtener el código ATASCII, dígame al ordenador (en Modo Directo) que PRINT ASC ("_____"). Rellene el espacio en blanco con una letra, carácter o número de código. Debe utilizar comillas.
5. En las páginas 117 y 119 la exposición normal de las tapas de las teclas son demostradas como símbolos blancos en un fondo blanco; en las páginas 120 y 122 los símbolos inversos de la parte superior de las teclas son demostrados con negros en un fondo blanco.
6. Para habilitar los caracteres internacionales hay que hacer un POKE 756,204.

APENDICE D

MAPA DE LA MEMORIA DEL ATARI

DIRECCION		CONTENIDOS
Decimal	Hexadecimal	
65535	FFFF	ROM DEL SISTEMA OPERATIVO
57344	E000	ROM DEL SISTEMA OPERATIVO
57343	DFFF	COMA FLOTANTE DEL ROM
55296	D800	COMA FLOTANTE DEL ROM
55295	D7FF	REGISTRO DE HARDWARE
53248	D000	REGISTRO DE HARDWARE
53247	CFFF	NO UTILIZADO
49152	C000	NO UTILIZADO
49151	BFFF	RANURA DEL CARTUCHO
40960	A000	
40959	9FFF	RANURA DEL CARTUCHO Es cartucho B
32768	8000	← <i>RAMTOP (MSB)</i>
32767	7FFF	(7FFF si es sistema 32K) VISUALIZACION DATOS (tamaño varía)
31755	7C1F	VISUALIZACION DE LISTA (tamaño varía) (7C1F si sistema 32K (GRAPHICS 0))
		← <i>OS MEMTOP</i>

		RAM LIBRE Tamaño varía	← BASIC MEMTOP
		Programas, buffers, tablas, pila de tiempo de realización BASIC. (2A80 si DOS, puede variar)	
10880	2A80		← OS MEMLO BASIC LOMEM
10879	2A7F	SISTEMA OPERATIVO DE DISCO (2A7F-700)	
9856	2680	BUFFERS DE DISCO DE I/O (DOS actual)	
9855	267F	SISTEMA OPERATIVO DE DISCO RAM	
4864	1300	(DOS actual)	
4863	12FF	SISTEMA RAM DEL MANEJO DEL FICHERO	
1792	700	(DOS actual)	
1791	6FF	RAM LIBRE	
1536	600	RAM LIBRE	
1535	5FF	COMA FLOTANTE	
1406	57E	(utilizado por BASIC)	
1405	57D	BASIC	
1152	480	BASIC	
1151	47F	SISTEMA OPERATIVO RAM (47F-200)	
1021	3FD	BUFFER DE CASSETTE	
1020	3FC	RESERVADO	
1000	3E8	RESERVADO	
999	3E7	BUFFER DE IMPRESORA	
960	3C0	BUFFER DE IMPRESORA	

959	3BF	IOCB	
832	340	IOCB	
831	33F	VARIABLES MISCELANEAS OS	
512	200	VARIABLES MISCELANEAS OS	
511	1FF	PILA DE HARDWARE	
256	100	PILAR DE HARDWARE	
255	FF	PAGINA CERO	
212	D4	COMA FLOTANTE (utilizado por BASIC)	
211	D3	PROGRAMA	
210	D2	PROGRAMA	
209	D1	RAM BASIC LIBRE	
208	D0	RAM BASIC LIBRE	
207	CF	BASIC Y ASSEMBLER RAM LIBRE	
203	CB	BASIC Y ASSEMBLER RAM LIBRE	
202	CA		
176	B0	ASSEMBLER RAM LIBRE	
128	80	PAGINA CERO DEL ASSEMBLER	PAG. CERO DEL BASIC
127	7F	SISTEMA OPERATIVO RAM	
0	0	SISTEMA OPERATIVO RAM	

Como las direcciones para la parte superior del RAM, OS y BASIC, y las puntas del OS y BASIC varían de acuerdo con la cantidad de memoria, éstas direcciones están indicadas con indicadores. Las direcciones de los indicadores están definidas en el Apéndice I.

APENDICE E

FUNCIONES DERIVADAS

Funciones derivadas

Secante
 Cosecante
 Arco Seno
 Arco Coseno
 Arco Secante
 Arco Cosecante
 Arco Cotangente
 Seno Hiperbólico
 Coseno Hiperbólico
 Tangente Hiperbólica
 Secante Hiperbólica
 Cosecante Hiperbólica
 Cotangente Hiperbólica
 Arco Seno Hiperbólico
 Arco Coseno Hiperbólico
 Arco Tangente Hiperbólico
 Arco Secante Hiperbólico
 Arco Cosecante Hiperbólico
 Arco Cotangente Hiperbólico

Funciones derivadas con respecto a las de Atari

$SEC(X) = 1/COS(X)$
 $CSC(X) = 1/SIN(X)$
 $ARCSIN(X) = ATN(X/SQR(-X * X + 1))$
 $ARCCOS(X) = -ATN(X/SQR(-X * X + 1)) + CONSTANTE$
 $ARSEC(X) = ATN(SQR(X * X - 1)) + SGN(X - 1) * CONS.$
 $ARCCSC(X) = ATN(1/SQR(X * X - 1)) + (SGN(X - 1) * CONS.$
 $ARCCOT(X) = ATN(X) + CONS.$
 $SINH(X) = (EXP(X) - EXP(-X))/2$
 $COSH(X) = (EXP(X) + EXP(-X))/2$
 $TANH(X) = -EXP(-X)/(EXP(X) + EXP(-X)) * 2 + 1$
 $SECH(X) = 2/(EXP(X) + (EXP(-X)))$
 $CSCH(X) = 2/(EXP(X) - EXP(-X))$
 $COTH(X) = EXP(-X)/(EXP(X) - EXP(-X)) * 2 + 1$
 $ARCSINH(X) = LOG(X + SQR(X * X + 1))$
 $ARCCOSH(X) = LOG(X + SQR(X * X - 1))$
 $ARCTANH(X) = LOG((1 + X)/(1 - X))/2$
 $ARCSECH(X) = LOG((SQR(-X * X + 1) + 1)/X)$
 $ARCCSCH(X) = LOG((SGN(X) * SQR(X * X + 1) + 1)/X)$
 $ARCCOTH(X) = LOG((X + 1)/(X - 1))/2$

Notas:

1. Si está en el modo (por defecto) RAD, la constante = 1.57079633.
 Si está en el modo DEG, la constante = 90.
2. En esta tabla, la variable X entre paréntesis representa el valor o la expresión que ha de ser evaluada por la función derivada. Obviamente, se permite cualquier nombre de variable, siempre y cuando represente al número o expresión que ha de ser evaluado.

APENDICE F

VERSIONES IMPRIMIDAS DE LOS CARACTERES DE CONTROL

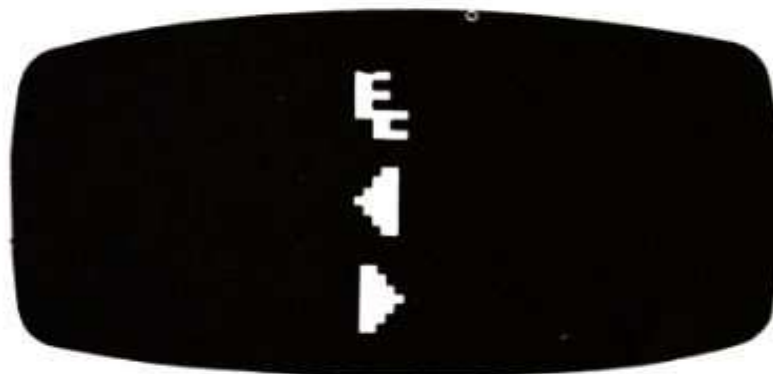
Los caracteres de control de pantalla y el cursor pueden ser ubicados en una cadena en un programa o utilizados en una declaración en el Modo Directo al oprimir la tecla ESC antes de entrar el carácter del teclado. Esto hace que los símbolos especiales, que mostramos seguidamente, sean visualizados. (Referirse a la Sección 1: Tecla ESC.)

VERA ESTO

PULSE



PULSE



PULSE

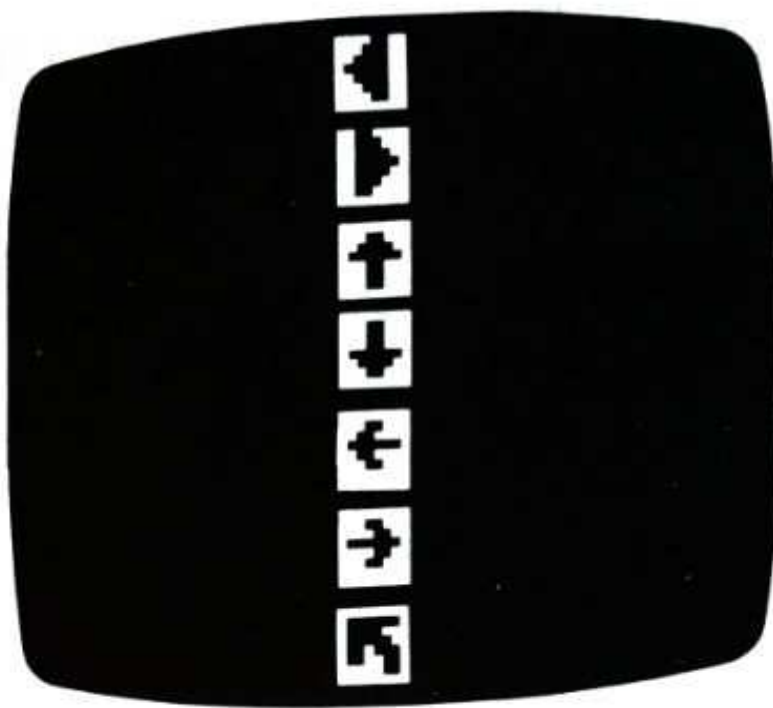
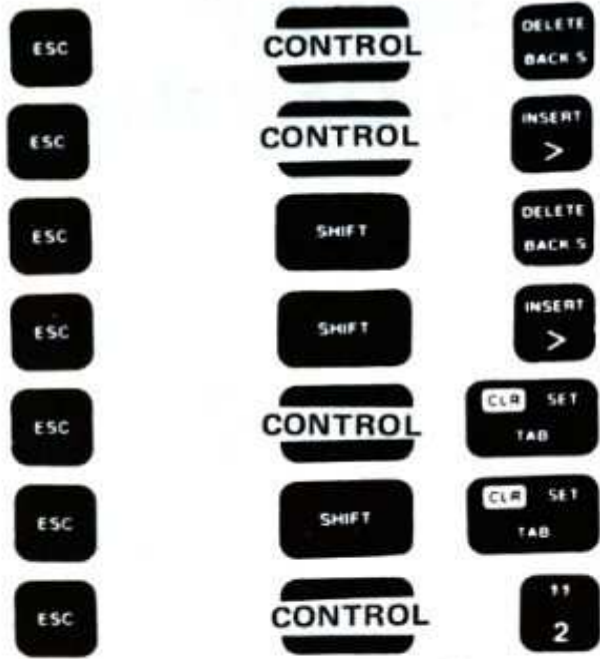


PULSE Y SOSTENGA PULSE



OR





APENDICE G

GLOSARIO

- Alfanumérico:** Las letras alfabéticas de la A a la Z, los números 0 al 9, y algunos símbolos. (Ningún signo de puntuación o símbolo gráfico.)
- ATASCII:** Significa Código Standard Americano de Atari para el Intercambio de Información.
- Ausencia (por defecto):** Esto es una condición o modo "asumida" por el ordenador hasta que le sea indicado hacer otra cosa. Por ejemplo, se "ausentará" o "por defecto" tomará la pantalla o teclado si no se le indica utilizar otros dispositivos de I/O.
- Aviso:** Es un símbolo que aparece en la pantalla del monitor que indica que el ordenador está preparado para aceptar el Input del teclado. En el BASIC ATARI, esto toma la forma de la palabra READY. Un "?" es también utilizado para incitar a quien lo utilice para entrar información o para tomar otra acción apropiada.
- BASIC:** Significa el Código de Instrucciones de Símbolos de Todo Propósito para Principiantes. Es un lenguaje de programación de alto nivel. La palabra BASIC siempre es escrita con mayúsculas. Fue desarrollado por los Mssrs. Kemey y Kurtz en la Facultad de Dartmouth en 1963.
- Binario:** Es un sistema numérico que utiliza la base dos. Por lo tanto, los únicos dígitos posibles son el 0 y el 1, que pueden ser utilizados en el ordenador para representar verdadero y falso, encendido y apagado, etc.
- Bit:** Es una forma corta para decir Dígito Binario. Un bit puede considerarse como representante de verdadero o falso, si un circuito está encendido o apagado, o cualquier otro tipo de concepto que contenga dos posibilidades. Un bit es la unidad más pequeña de datos con la cual puede funcionar un ordenador.

Bug:	Es un error o equivocación del programa o en el software.
Byte:	Generalmente son ocho bits. Un byte de datos puede ser utilizado para representar un carácter ATASCII o un número en la extensión del 0 al 255.
Cadena:	Es una secuencia de números, letras y otros caracteres. Puede ser almacenado en una variable en cadena. El nombre de la variable en cadena debe finalizar con un \$.
Cadena nula:	Es una cadena no absolutamente compuesta por ningún carácter.
Caracteres de control:	Son los caracteres que se producen al sostener la tecla CTRL mientras que simultáneamente está oprimiendo otra tecla.
Carácter Especial:	Es un carácter que puede ser expuesto por un ordenador que no es ni una letra ni un número. Los símbolos gráficos de ATARI son caracteres especiales. Como también lo son los signos de puntuación, etc.
Código:	Son instrucciones que son escritas en un lenguaje que es entendido por el ordenador.
Concatenación:	Es el proceso que une dos o más cadenas juntas para formar una cadena más larga.
CRT:	Es la abreviatura de "Tubo de Rayos Catódicos" (el tubo que es utilizado en aparatos de televisión). En la práctica, esto es utilizado a menudo para describir al receptor de televisión utilizado para visualizar el output del ordenador. También llamado monitor.
Cursor:	Este es un recuadro visualizado en la pantalla que le dice a usted dónde será expuesto el próximo carácter escrito.
Datos (Data):	Es información de cualquier tipo.
Declaración:	Es una instrucción al ordenador. Ver también Orden. Mientras que todas las órdenes pueden ser consideradas como declaraciones, todas las declaraciones no son de ninguna manera órdenes. Una declaración contiene un número de línea (en el Modo Diferido), una palabra clave, el valor que ha de operar y la orden RETURN .

Debug:	Esto es el proceso de ubicar y corregir las equivocaciones y errores en un programa. También el proceso de "purificar" un programa.
Digital:	Es información que puede ser representada por un grupo de bits. Virtualmente todos los ordenadores de hoy en día, especialmente los micro-ordenadores, utilizan el acceso digital.
Diskette (Disco):	Es un medio de grabar/reproducir como una cinta, pero construido con la forma de un disco chato que está puesto dentro de un sobre para protegerlo. La ventaja del disco con respecto al cassette en cuanto al almacenamiento de memoria, es que el acceso a cualquier parte del disco es casi inmediata. El Sistema de Ordenadores Personales ATARI 800 puede controlar hasta cuatro periféricos de unidades de disco simultáneamente. En este manual, las palabras disco y diskette son utilizadas intercambiamente.
DOS:	Es la abreviatura de "Sistema Operativo de Disco", o sea, el software o programas que facilitan la utilización de un sistema de unidades de disco.
Editado:	Es el proceso de hacer correcciones o cambios en un programa o en los datos.
Ejecutar:	Es hacer lo que especifica un programa u orden. O de RUN un programa o porción del mismo.
En serie:	Es lo contrario de paralelo. Son cosas que ocurren únicamente una a la vez en una secuencia. Por ejemplo, un interface en serie.
Expresión:	Es una combinación de variable, números y operadores (como el +, -, etc.) que pueden ser evaluados a una cantidad única. Esta cantidad puede ser una cadena o un número.
Formato:	Es utilizado para especificar la forma en la cual algo ha de aparecer.
Generador de números al azar:	Este puede ser de hardware (como es el de ATARI) o un programa que provee un número cuyo valor es difícil de

	predecir. Es utilizado primordialmente para formular decisiones en los programas de juegos, etc.
Hard Copy:	Es el output imprimido al contrario de lo expuesto temporalmente en el monitor del televisor.
Hardware:	Son los aparatos físicos y electrónicos que componen el ordenador.
Incrementar:	Esto significa incrementar en valor (generalmente) agregando 1. Es utilizado muchísimo para contar.
Inicializar:	Es fijar una inicial o valor que comienza. En el BASIC ATARI, todas las variables que no sean puestas en una tabla, están marcadas con iniciales al cero cuando la orden RUN es dada. Las tablas y los elementos de una cadena no son marcados con iniciales.
Input:	Es la información que se le da al ordenador. Output es la salida de información por parte del ordenador (I/O). En este manual, el Output y el Input están siempre relacionados con el ordenador.
Interactivo:	Es un sistema que responde rápidamente a quien lo utilice, generalmente entre un segundo o dos. Todos los sistemas de los ordenadores personales son interactivos.
Interface:	Son los sistemas electrónicos utilizados para permitir la comunicación entre dos dispositivos.
IOCB:	Significa Bloque de Control de I/O. Es un bloque de datos en el RAM que le indica al Sistema Operativo la información que necesita conocer para una operación I/O.
I/O:	Es lo corto para Input/Output (Entrada/Salida), los dispositivos I/O incluyen el teclado, el monitor de la televisión; la grabadora de programas, la impresora y las unidades de disco.
K:	Significa "kilo", queriendo decir "multiplicado por 1000". Por lo tanto 1KByte es (aproximadamente) 1000 bytes. (Realmente son 1024 bytes.) Además, es el código de tipo de dispositivo para el teclado.

Lenguaje:	Es un conjunto de convenciones especificando cómo indicarle al ordenador lo que debe hacer.
Menu:	Es una lista de opciones de donde puede elegir el que lo utilice.
Micro-ordenador:	Es un ordenador que está basado en el chip micro-procesador; en el caso de ATARI, es el 6502.
Monitor:	Es el receptor de televisión utilizado para visualizar el output del ordenador.
Orden:	Es cualquier instrucción dada al ordenador, la cual es inmediatamente ejecutada. Un buen ejemplo es la orden RUN.
Ordenador:	Es cualquier dispositivo que puede recibir y luego seguir instrucciones para manipular información. Tanto las instrucciones como la información pueden ser variadas de un momento a otro. La distinción entre un ordenador y una calculadora programable está en la habilidad del ordenador de manipular textos además de números. La mayoría de las calculadoras pueden manejar únicamente números.
OS:	Es la abreviatura utilizada para el Sistema Operativo. Esto es realmente una colección de programas para ayudar a quien lo utilice para controlar el el ordenador.
Output:	Ver I/O.
Palabra clave:	Es una palabra que tiene significado como una orden o una instrucción en el lenguaje de los ordenadores, y por lo tanto no debe ser utilizada como el nombre de una variable o al principio del nombre de una variable.
Palabra reservada:	Ver Palabra clave.
Pantalla:	La pantalla del televisor. En el BASIC ATARI, un dispositivo particular de I/O que da un código de "S:".
Paralelo:	Son dos o más cosas que ocurren simultáneamente. Un interface paralelo, por ejemplo, controla un número de señales electrónicas inconfundibles al mismo tiempo. Es lo contrario de "en serie".

Periférico:	Un dispositivo de I/O. Ver I/O.
Pixel:	Un elemento de dibujo. Es un punto en la pantalla. El tamaño depende del Modo Gráfico que se está utilizando.
Precedencia:	Son las reglas que determinan la prioridad en la cual se conducen las operaciones, especialmente con respecto a los operadores aritméticos/lógicos.
Programa:	Es una secuencia de instrucciones que describen un proceso. Un programa tiene que estar en el lenguaje que el ordenador en particular puede entender.
RAM	Acceso a la Memoria al Azar. Es la memoria principal en la mayoría de los ordenadores. El RAM es utilizado para almacenar tanto programas como datos.
Rama:	El BASIC ATARI ejecuta un programa siguiendo el orden de numérico de las líneas. Esta secuencia de ejecución puede ser alterada por el programador, y el programa puede ser instruido a saltar un cierto número de líneas o volver a una línea anterior. Este cambio inventado de la secuencia de ejecución es llamado "ramificación".
ROM:	Memoria de Lectura Únicamente. En este tipo de memoria electrónica de estado sólido, la información es almacenada por el fabricante y no puede ser cambiada por el que lo utilice. Programas tales como el intérprete BASIC y otros cartuchos utilizados con los Sistemas ATARI utiliza el ROM.
Save:	Es utilizado para copiar un programa o datos a una posición que no sea el RAM. (Por ejemplo, en diskette o cassette.)
Software:	Es lo contrario de hardware. Se refiere a los programas y datos.
Subrutina:	Es una parte de un programa que puede ser ejecutado por una declaración especial (GOSUB) en el BASIC: Este eficazmente da una única declaración al poder de un programa entero.

Tabla:	Es una lista de valores numéricos almacenados en una serie de posiciones de memoria precedidas por una declaración DIM. Se pueden referir a ella por medio de la utilización de una variable de tabla. Se refieren a sus elementos individuales por medio de nombres de variables suscritas.
Unidad Procesadora Central (CPU):	En los micro-ordenadores tal como los sistemas ATARI, éstos son también llamados microprocesadores o MPU. En una época, el CPU era aquella porción de cualquier ordenador que controlaba la memoria y los periféricos. Ahora, el CPU o MPU es generalmente hallado en un circuito integrado único o CHIP (en el caso de ATARI es un CHIP Microprocesador 6502).
Variable:	Se puede pensar en una variable como fuese una caja en la cual un valor puede ser almacenado. Tales valores son típicamente números o cadenas.
Ventana:	Es una porción de la visualización de la televisión dedicada a un propósito específico tal como para gráficos o texto.

APENDICE H

PROGRAMAS PARA EL USUARIO

Este apéndice contiene programas y rutinas que demuestran las capacidades diversas del Sistema de Ordenadores Personales ATARI. Incluido en este apéndice hay un programa Decimal/Hexadecimal para aquellas personas que lo utilicen para escribir programas que requieren este tipo de conversión.

SALDO DE TALONARIOS

Este es uno de los programas "tradicionales" que cada principiante escribe. Este permite la entrada de cheques sobresalientes e ingresos no acreditados como también los cheques comprobados y los ingresos acreditados.

```
10 DIM A$(30), MSG$(40), MSG1$(30), MSG2$(30), MSG3$(30),  
    MSG4$(30), MSG5$(30), MSG6$(30)  
20 PENDIENTE = 0  
30 GRAPHICS 0:?:?:" SALDO DEL TALONARIO":?  
40 ? "Puede hacer correcciones en cualquier momento tecleando  
    un valor negativo de pesetas"  
50 MSG1$ = "CHEQUE ANTIGUO -- AUN PENDIENTE"  
60 MSG2$ = "INGRESO ANTIGUO -- NO ACREDITADO"  
70 MSG3$ = "CHEQUE ANTIGUO -- ACABA DE SER LIQUI-  
    DADO"  
80 MSG4$ = "INGRESO ANTIGUO - ACABA DE SER ACRE-  
    DITADO"  
90 MSG5$ = "CHEQUE NUEVO (O GASTO BANCARIO)"  
100 MSG6$ = "INGRESO NUEVO (O INTERES)"  
150 TRAP 150:?"Teclee el primer saldo de su talonario";:INPUT  
    TUSALDO  
160 TRAP 160:?"Teclee su primer saldo de su extracto de banco";  
    :INPUT SALBANC  
165 TRAP 40000  
170 GOTO 190  
180 CLOSE #1:?"LA IMPRESORA NO RESPONDE"
```

```

185 ? "POR FAVOR COMPRUEBE LOS CONECTORES"
190 PERM=0
200 ? "Desearía un registro permanente en la impresora?"; :INPUT
    A$
210 IF LEN(A$)=0 THEN 200
220 IF A$(1,1)="N" THEN 400
230 IF A$(1,1)<>"S" THEN 200
240 TRAP 180
250 LPRINT :REM TEST A LA IMPRESORA
260 PERM=1
280 LPRINT "SU SALDO DE COMIENZO ES $";TUSALDO
290 LPRINT "EXTRACTO DE BANCO COMENZANDO EL
    SALDO ES $";SALBANC:LPRINT
400 TRAP 400:?:? "Elija uno de los siguientes:"
410 ?"(1) ";MSG1$
415 ?"(2) ";MSG2$
420 ?"(3) ";MSG3$
425 ?"(4) ";MSG4$
430 ?"(5) ";MSG5$
435 ?"(6) ";MSG6$
440 ?"(7) ACABADO"
490 ?
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 MSG$="SALDO NUEVO DEL TALONARIO ES": CANTI-
    DAD=TUSAL: GOSUB 8000
530 MSG$="SALDO NUEVO DEL EXTRACTO DE BANCO ES":
    CANTIDAD=SALBANC:GOSUB 8000
540 MSG$="TALONARIOS-INGRESOS PENDIENTES=":CAN-
    TIDAD=PENDIENTE:GOSUB 8000
545 IF PERM THEN LPRINT
550 GOTO 400
1000 REM CHEQUE ANTIGUO -- AUN PENDIENTE
1010 MSG$=MSG1$:GOSUB 8100
1020 PENDIENTE=PENDIENTE+CANTIDAD
1030 RETURN
2000 REM INGRESO ANTIGUO -- AUN NO ACREDITADO
2010 MSG$=MSG2$:GOSUB 8100
2020 PENDIENTE=PENDIENTE-CANTIDAD
2030 RETURN

```



```

3000 REM CHEQUE ANTIGUO -- ACABA DE SER LIQUIDADO
3010 MSG$=MSG3$:GOSUB 8100
3020 SALBANC=SALBANC-CANTIDAD
3030 RETURN
4000 REM INGRESO ANTIGUO -- ACABA DE SER ACREDI-
TADO
4010 MSG$=MSG4$:GOSUB 8100
4020 SALBANC=SALBANC+CANTIDAD
4030 RETURN
5000 REM CHEQUE NUEVO (O GASTO BANCARIO) - ACABA
DE SER LIQUIDADO
5010 MSG$=MSG5$:GOSUB 8100
5020 TUSAL=TUSAL-CANTIDAD
5030 ? "ES EL CHEQUE NUEVO AUN PENDIENTE";:INPUT A$
5040 IF LEN(A$)=0 THEN 5030
5050 IF A$(1,1)<>"N" THEN 5060
5055 SALBANC=SALBANC-CANTIDAD
5057 IF PERM THEN LPRINT "EL CHEQUE SIGUE PENDIENTE
5058 RETURN
5060 IF A$(1,1)<>"S" THEN 5030
5070 PENDIENTE=PENDIENTE+CANTIDAD
5075 IF PERM THEN LPRINT "EL CHEQUE SIGUE PENDIENTE."
5080 RETURN
6000 REM INGRESO NUEVO (O INTERES) - ACABA DE SER
ACREDITADO
6010 MSG$=MSG6$:GOSUB 8100
6020 TUSAL=TUSAL+CANTIDAD
6030 ?"HA SIDO ACREDITADO SU NUEVO SALDO";:INPUT A$
6040 IF LENT(A$)=0 THEN 6030
6050 IF A$(1,1)<>"Y" THEN 6060
6052 SALBANC=SALBANC+CANTIDAD
6053 IF PERM THEN LPRINT "EL INGRESO HA SIDO ACREDI-
TADO."
6055 RETURN
6060 IF A$(1,1)<>"N" THEN 6030
6070 PENDIENTE=PENDIENTE-CANTIDAD
6075 IF PERM THEN LPRINT "EL INGRESO NO HA SIDO ACRE-
DITADO"
6080 RETURN
7000 REM ACABADO

```



```

7010 ?"EL SALDO DEL BANCO MENOS(PENDIENTE      IN-
      GRESOS DE TALON) DEBERAN AHORA IGUALAR SU
      SALDO DEL TALONARIO."
7020 DIF=TUSAL-(SALBANC-PENDIENTE)
7030 IF DIF<>0 THEN 7040
7035 ? "ES $";SALBANC;" EL SALDO FINAL DE SU EXTRAC-
      TO DE BANCO"; :INPUT A$
7036 IF LEN(A$)=0 THEN 7035
7037 IF A$(1,1)="S" THEN ? "BIEN: SU BALANCE CUADRA!"
      :END
7038 GOTO 7060
7040 IF DIF>0 THEN ?"EL TOTAL DE SU TALONARIO ES $";
      DIF;"SOBRE EL TOTAL DEL BANCO.":GOTO 7060
7050 ? "EL TOTAL DE SU TALONARIO ES $";-DIF;"POR DE-
      BAJO DEL TOTAL DEL BANCO"
7060 ? "DESEARIA HACER ALGUNA CORRECCION?"
7070 ? "RECUERDE, PUEDE TECLEAR UN VALOR NEGATIVO
      DE PESETAS PARA HACER ALGUNA CORRECCION."
7080 ?"TECLEE S O N";INPUT A$
7090 IF LEN(A$)=0 THEN END
7100 IF A$(1,1)="S" THEN RETURN
7110 END
7999 REM RUTINA DE IMPRESION DE MSG
8000 ? MSG$;" $";CANTIDAD
8010 IF PERM=1 THEN LPRINT MSG$;" $";CANTIDAD
8020 RETURN
8100 REM RUTINA DE IMPRESION Y DE ENTRADA DE MSG
8110 TRAP 8110:? "TECLEE LA CANTIDAD PARA ";MSG$;
      :INPUT CANTIDAD
8120 TRAP 40000
8130 IF PERM=1 THEN LPRINT MSG$;" $";CANTIDAD
8140 RETURN

```

BUBBLES SORT

Este programa utiliza el operador de comparación de cadenas "<=" que ordena cadenas de acuerdo con los valores ATASCII de los distintos caracteres. Puesto que el BASIC ATARI no tiene tablas o cadenas, todas las cadenas utilizadas en este programa son realmente subcadenas de una cadena más larga. El BUBBLE SORT, aunque sea relativamente lento si hay muchos artículos para

almacenar, es fácil de escribir, es bastante corto y más fácil de entender que otros clasificadores (sorts) más complejos.

```
10 DIMB$(1)
20 GRAPHICS 0:?:?"CLASIFICADOR DE CADENAS":?
30 TRAP 30:?:?"Teclee la longitud máxima de la cadena";:IN-
  PUT SLEN:SLEN1=SLEN-1
35 IF SLEN<1 OR INT(SLEN)<>SLEN THEN ?"POR FAVOR
  TECLEE UN NUMERO POSITIVO ":GOTO 30
40 TRAP 40:?:?"Teclee el número máximo de entradas."
41 ? "(Entradas que sean más cortas que la longitud máxima serán
  rellenas con espacios en blanco.)"
42 INPUT ENTRADAS
45 IF ENTRADAS<2 OR INT(ENTRADAS)<>ENTRADAS
  THEN ?"POR FAVOR, TECLEE UN NUMERO POSITIVO
  >1.":GOTO 40
47 TRAP 40000
50 DIM A$(SLEN*ENTRADAS),TEMP$(SLEN)
60 ? :?"Teclee las cadenas de una una en una."
70 ?"Teclee cadenas vacías cuando haya terminado (simplemente
  pulse RETURN)."
75 ? :?"POR FAVOR, ESPERE MIENTRAS LAS CADENAS SON
  CLASIFICADAS...";
80 FOR I=1 TO SLEN*ENTRADAS:A$(I,I)=" ":NEXT I
85 ? :?
90 I=1
100 FOR J=1 TO ENTRADAS
110 ? " ";J;" ";:INPUT TEMP$
120 IF LEN(TEMP$)=0 THEN ENTRADAS=J-1:GOTO 190
130 A$(I,I+SLEN1)=TEMP$
140 I=I+SLEN
150 NEXT J
190 ? :?:?"POR FAVOR, ESPERE MIENTRAS LAS CADENAS
  SON CLASIFICADAS...";
200 GOSUB 1000:REM LLAMAR A LA RUTINA DE CLASIFICA-
  CION
202 ? :?
205 I=1
210 FOR K=1 TO ENTRADAS
220 ?"#";K;" ";A$(I,I+SLEN1)
225 I=I+SLEN
```

```

230 NEXT K
240 TRAP 300:?:?"DESEA UNA COPIA IMPRESA";:INPUT B$
250 IF B$(1,1)="S" THEN 400
300 END
400 I=1:LPRINT:FOR K=1 TO ENTRADAS
420 LPRINT " ";K;" ";A$(I,I+SLEN1)
430 I=I+SLEN:NEXT K:END
1000 REM RUTINA DE CADENA DE BUBBLE SORT
1010 REM INPUT:A$,SLEN,ENTRADAS
1015 REM TEMP$ DEBE TENER UNA DIMENSION SLEN
1020 SLEN1=SLEN-1:MAX=SLEN*(ENTRADAS-1)+1
1040 FOR I=1 TO MAX STEP SLEN
1050 ACABADO=1
1060 FOR K=1 TO MAX-I-SLEN1 STEP SLEN
1070 KSLEN1=K+SLEN1:KSLEN=K+SLEN:KSLENSLEN1=
KSLEN+SLEN1
1080 IF A$(K,KSLEN1)<A$(KSLEN,KSLENSLEN1) THEN GOTO
1110
1090 ACABADO=0
1100 TEMP$=A$(K,KSLEN1):A$(K,KSLEN1)=A$(KSLEN,
KSLENSLEN1):A$(KSLEN,KSLENSLEN1)=TEMP$
1110 NEXT K
1120 IF ACABADO THEN RETURN
1130 NEXT I
1140 RETURN

```

IMPRESION DE LOS CARACTERES DE LOS MODOS DE TEXTO

Este programa imprime los caracteres ATARI en sus colores de ausencia para los modos de texto 0, 1 y 2. Al entrar este programa, recuerde que el símbolo para despejar la pantalla: "␣" es imprimido como un paréntesis "}".

```

1 DIM A$(1)
5 ? "}" : REM DESPEJAR LA PANTALLA
10 ? "GRAPHICS 0, 1, Y ¿ (MODOS DE TEXTO)"
20 ? "DEMOSTRACION."
30 ? "VISUALIZA CONJUNTOS DE CARACTERES PARA CADA
MODO."
60 ESPERE=1000:REM SUBROUTINA DE NUMERO DE LINEA
70 CHBAS=756:REM DIRECCION DE LA BASE DEL CARAC-
TER

```



```

80 MAYUSCULAS=224:REM VALOR POR DEFECTO PARA
  CHBAS
90 MINUSCULAS=226:REM MINUSCULAS Y GRAFICOS
95 GOSUB ESPERE
100 FOR L=0 TO 2
112 REM UTILICE E: PARA GRAPHICS 0
115 IF L=0 THEN OPEN #1,8,0,"E:":GOTO 118
116 REM UTILICE S: PARA GRAFICOS 1 Y 2
117 OPEN#1,8,0,"S:"
118 GRAPHICS L
120 PRINT "GRAPHICS ";L
130 FOR J=0 TO 7:REM 8 LINEAS
140 FOR I=0 TO 31:REM 32 CARACTERES/LINEA
150 K=32*J+I
155 REM NO EXPONGA "DESPEJE DE PANTALLA" O 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 VISUALIZAR CARACTERES
180 NEXT I
190 PRINT#1;" ":REM FINAL DE LINEA
200 IF L< >2 OR J< >3 THEN 240
210 REM PANTALLA LLENA
220 GOSUB ESPERE
230 PRINT#1;" ":REM LIMPIAR PANTALLA
240 NEXT J
250 GOSUB ESPERE
265 PRINT "MINUSCULAS Y GRAFICOS"
270 IF L< >0 THE POKE CHBAS, MINUSCULAS:GOSUB ES-
  PERE
275 CLOSE#1
280 NEXT L
300 GRAPHICS 0:END
1000 REM ESPERE PARA "RETURN"
1010 PRINT "PULSE RETURN PARA CONTINUAR";
1020 INPUT A$
1030 RETURN

```

SHOW DE LUCES

Este programa demuestra otros aspectos de los gráficos ATARI. Utiliza el modo gráfico 7 para alta resolución y las declaraciones PLOT y DRAWTO para

dibujar las líneas. En la página 20, el título será más eficaz si es entrado en video inverso.

```
10 FOR ST=1 TO 8:GRAPHICS 7
15 POKE 752,1
20 ? :? "    SHOW DE LUCES DE ATARI":SETCOLOR 2, 0, 0
30 SETCOLOR 1, 2*ST, 8:COLOR 2
40 FOR DR=0 TO 80 STEP ST
50 PLOT 0,0:DRAWTO 100,DR
60 NEXT DR:FOR N=1 TO 800:NEXT N:NEXT ST
70 FOR N=1 TO 2000:NEXT N:GOTO 10
```

LA BANDERA DE LOS EE. UU.

Este programa incumbe en cambiar colores para formar rayas. Utiliza el Modo Gráfico 7 + 16 para que lo expuesto aparezca en la pantalla entera. Observe la concordancia de las declaraciones SETCOLOR. Para divertirse o experimentar, agregue una declaración de SOUND y utilice una combinación READ/DATA para añadir el Himno Nacional de los EE. UU. después de la línea 470. Referirse a la Sección 10.

```
10 REM DIBUJAR LA BANDERA DE LOS EE.UU.
20 REM ALTA RESOLUCION, GRAFICOS 4-COLORES, NIN-
   GUNA VENTANA DE TEXTO.
30 GRAPHICS 7+16
40 REM SETCOLOR 0 CORRESPONDE A COLOR 1
50 SETCOLOR 0,4,4:ROJO=1
60 REM SETCOLOR 1 CORRESPONDE A COLOR 2
70 SETCOLOR 1,0,14:BLANCO=2
80 REM SETCOLOR 2 CORRESPONDE A COLOR 3
90 AZUL=3:REM TOMA POR DEFECTO AZUL.
100 REM DIBUJA 13 FRANJAS ROJAS/BLANCAS
110 C=ROJO
120 FOR I=0 TO 12
130 COLOR C
140 REM CADA FRANJA TIENE VARIAS LINEAS HORIZON-
   TALES
150 FOR J=0 TO 6
160 PLOT 0,I*7+J
170 DRAWTO 159,I*7+J
180 NEXT J
```



```

190 REM INTERCAMBIA COLORES
200 C=C+1:IF C>BLANCO THEN C=ROJO
210 NEXT I
300 REM DIBUJA CUADRADO AZUL
310 COLOR AZUL
320 FOR I=0 TO 48
330 PLOT 0,I
340 DRAWTO 79,I
350 NEXT I
360 REM DIBUJA 9 FILAS DE ESTRELLAS BLANCAS
370 COLOR BLANCO
380 K=0:REM EMPIEZA CON FILA DE SEIS ESTRELLAS
390 FOR I=0 TO 8
395 Y=4+I*5
400 FOR J=0 TO 4:REM 5 ESTRELLAS POR FILA
410 X=K+5+J*14:GOSUB 1000
420 NEXT J
430 IF K<>0 THEN K=0:GOTO 470
440 REM AGREGA UNA ESTRELLA A CADA LINEA
450 X=5+5*14:GOSUB 1000
460 K=7
470 NEXT I
500 REM SI UNA TECLA ES PRESIONADA, ENTONCES STOP
510 IF PEEK(764)=255 THEN 510
515 REM ABRE LA VENTANA DE TEXTO SIN DESPEJAR LA
    PANTALLA
520 GRAPHICS 7+32
525 REM CAMBIAR LOS COLORES DE NUEVO
530 SETCOLOR 0,4,4:SETCOLOR 1,0,14
550 STOP
1000 REM DIBUJA UNA ESTRELLA EN LA POSICION X,Y
1010 PLOT X-1,Y:DRAWTO X+1,Y
1020 PLOT X,Y-1:DRAWTO X,Y+1
1030 RETURN

```

GAVIOTA SOBRE EL OCEANO

Este programa combina los sonidos y los gráficos. Los sonidos son "nítidos" ni muy realistas, pero simulan el rugir del océano y el "tweet" de las gaviotas.

Los símbolos gráficos para simular la gaviota no pueden ser imprimidos en la impresora. Teclee los siguientes caracteres en la línea 20.

```
20 PAJARO$ = "V--"
```

Para obtener estos símbolos, utilice CTRL-G, CTRL-F, CTRL-R, CTRL-R.

```
10 DIM PAJARO$(7)
20 PAJARO$ = "      "
30 BANDERA = 1: FILA = 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 océano"
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 FOR H = 1 TO 10: NEXT H
185 GOSUB 200
190 NEXT T
195 GOTO 70
200 GOSUB 300
210 POSITION COL, FILA
220 PRINT #6; PAJARO$(BANDERA, BANDERA + 1)
230 BANDERA = BANDERA + 2: IF BANDERA = 5 THEN BAN-
    DERA = 1
240 RETURN
300 IF RND(0) > 0.5 THEN RETURN
310 POSITION COL, FILA
320 PRINT #6; "      "
330 A = INT(RND(0) * 3) - 1
```



```

340 B=INT(RND(0)*3)-1
350 FILA=FILA+A
360 IF FILA=0 THEN FILA=1
370 IF FILA=20 THEN FILA=19
380 COL=COL+B
390 IF COL=0 THEN COL=1
400 IF COL>18 THEN COL=18
410 RETURN

```

VIDEO GRAFFITI

Este programa requiere un control de palanca (Joystick) para cada jugador. Cada control de palanca tiene un color asociado con él. Mediante el movimiento del control de palanca, diversos dibujos son visualizados en la pantalla. Obsérvese el uso de las órdenes STICK y STRIG

```

1 GRAPHICS 0
2 ? "VIDEO GRAFFITI"
5 REM LAS TABLAS X&Y CONTIENEN CO-ORDENADAS
6 REM PARA HASTA 4 POSICIONES PARA JUGADORES
7 REM TABLA DE COLORES CONTIENE COLORES
10 DIM A$(1),X(3),Y(3),COLR(3)
128 ? "UTILICE CONTROL DE PALANCA PARA DIBUJAR"
129 ? "PRESIONE LOS BOTONES PARA CAMBIAR DE COLOR"
130 ? "COLORES INICIALES:"
131 ? "CONTROL DE PALANCA 1 ES ROJO"
132 ? "CONTROL DE PALANCA 2 EN BLANCO"
133 ? "CONTROL DE PALANCA 3 ES AZUL"
134 ? "CONTROL DE PALANCA 4 ES NEGRO (FONDO)"
135 ? "LA UBICACION DEL COLOR NEGRO ES INDICADO POR
UN BREVE DESTELLO ROJO."
136 ? "EN GRAPHICS 8, LOS CONTROLES DE PALANCA 1 Y 3
SON EL COLOR BLANCO Y EL 4 ES EL COLOR AZUL."
138 PRINT "CUANTOS JUGADORES(1-4)";
139 INPUT A$:IF LEN(A$)=0 THEN A$="1"
140 JOYMAX=VAL(A$)-1
145 IF JOYMAX<0 OR JOYMAX>4 THEN 138
147 PRINT "GRAPHICS 3 (40*24), 5(80*48),"
150 PRINT "7(160*96) o 8(320*192)";

```

```

152 INPUT A$:IF LEN(A$)=0 THEN A$="3"
153 A=VAL(A$)
154 IF A=3 THEN XMAX=40:YMAX=24:GOTO 159
155 IF A=5 THEN XMAX=80;YMAX=48:GOTO 159
156 IF A=7 THEN XMAX=160:YMAX=96:GOTO 159
157 IF A=8 THEN XMAX=320:YMAX=192:GOTO 159
158 GOTO 147:REM A NO VALIDA
159 GRAPHICS A + 16
160 FOR I=0 TO JOYMAX:X(I)=XMAX/2+I:Y(I)=YMAX/2+I:
    NEXT I:REM EMPEZAR CERCA DEL CENTRO DE LA PAN-
    TALLA
161 IF A<>8 THEN 166
162 FOR I=0 TO 2:COLR(I)=1:NEXT I
163 SETCOLOR 1,9,14:REM LT. BLUE
165 GOTO 180
166 FOR I=0 TO 2:COLR(I)=I+1:NEXT I
167 SETCOLOR 0,4,6:REM RED
168 SETCOLOR 1,0,14:REM BLANCO
180 COLR(3)=0
295 FOR J=0 TO 3
300 FOR I=0 TO JOYMAX:REM COMPROBAR CONTROLES
    DE MANDO
305 REM COMPROBAR EL BOTON ROJO DEL CONTROL DE
    MANDO
310 IF STRIG(I) THEN 321
311 IF A<>8 THEN 320
312 COLR(I)=COLR(I)+1:IF COLR(I)=2 THEN COLR(I)=0:
    REM MODO DE 2-COLOR
313 GOTO 321
320 COLR(I)=COLR(I)+1:IF COLR(I)>=4 THEN COLR(I)=0:
    REM MODO DE 4 COLORES
321 IF J>0 THEN COLOR COLR(I):GOTO 325
322 IF COLR(I) THEN COLOR 1:GOTO 325
323 COLOR 0
325 PLOT X(I),Y(I)
330 JOYIN=STICK(I):REM LEER JOYSTICK
340 IF JOYIN=15 THEN 530:REM NINGUN MOVIMIENTO
342 COLOR COLR(I)
344 PLOT X(I),Y(I)
350 IF JOYIN>=8 THEN 390

```



```

360 X(I)=X(I)+1:REM MOVIMIENTO HACIA LA DERECHA
370 IF X(I)>=XMAX THEN X(I)=0
380 GOTO 430
390 IF JOYIN>=12 THEN 430
400 X(I)=X(I)-1: REM MOVIMIENTO HACIA LA IZQUIERDA
410 IF X(I)<0 THEN X(I)=XMAX-1
430 IF JOYIN<>5 AND JOYIN<>9 AND JOYIN<>13 THEN
470
440 Y(I)=Y(I)+1: IF Y(I)>=YMAX THEN Y(I)=0: REM MOVI-
MIENTO HACIA ABAJO.
460 GOTO 500
470 IF JOYIN<>6 AND JOYIN<>10 AND JOYIN<>14 THEN
500
480 Y(I)=Y(I)-1: IF Y(I)<0 THEN Y(I)=YMAX-1: REM MOVI-
MIENTO HACIA ARRIBA.
500 PLOT X(I),Y(I)
530 NEXT I
535 NEXT J
540 GOTO 295

```

CONTROLADOR DEL TECLADO

Este programa altera los registros en un chip llamado PIA. Para fijar éstos de nuevo a su valor de ausencia, para así hacer más I/O, pulse la tecla **RESET** o teclee **POKE PACTL,60**.

```

1 GRAPHICS 0
5 PRINT :PRINT "CONTROLADOR DEL TECLADO"
10 DIM FILA(3), I$(13), TECLA$(1)
30 GOSUB 6000
40 FOR CNT=1 TO 4
60 POS. 2,CNT*2*5:PRINT "CONTROLADOR ";CNT;":";
70 NEXT CNT
80 FOR CNT=1 TO 4:GOSUB 7000:POS. 19,CNT+CNT+5:
PRINT TECLA$;:NEXT CNT
6000 REM MONTAJE
6010 PORTA = 54016:PORTB = 54017:PACTL = 54018:PBCTL =
54019
6020 POKE PACTL,48:POKE PORTA,255:POKE PACTL,52:POKE
PORTA,221

```

```

6025 POKE PBCTL,48:POKE PORTB,255:POKE PBCTL,52:POKE
    PORTB,221
6030 FILA(0)=238:FILA(1)=221:FILA(2)=187:FILA(3)=119
6040 I$=" 123456789*0 "
6050 RETURN
7001 REM UN 1 SERA EXPUESTO SI NINGUN CONTROLADOR
    ESTAR CONECTADO
7003 PORT=PORTA:IF CNT>2 THEN PORT=PORTB
7005 P=1
7008 PAD=CNT+CNT-2
7010 FOR J=0 TO 3
7020 POKE PORT,FILA(J)
7030 FOR I=1 TO 10:NEXT I
7050 IF PADDLE(PAD+1)>10 THEN P=J+J+2:GOTO 7090
7060 IF PADDLE(PAD)>10 THEN P=J+J+3:GOTO 7090
7070 IF STRIG(CNT-1)=0 THEN P=J+J+4:GOTO 7090
7080 NEXT J
7090 TECLA$=I$(P,P)
7095 RETURN

```

TECLEE -UNA-MELODIA

Este programa asigna notas de valores musicales a las teclas de la parte superior del teclado (1, 2, 3, 4, 5, 6, 7, 8, 9, 0, <, >). Pulse una tecla a la vez.

TECLA	VALOR MUSICAL
>	SI
<	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	SI BEMOL (O RE SOSTENIDO)
3	RE
2	RE BEMOL (O DO SOSTENIDO)
1	DO


```

10 DIM CHORD(37), TUNE(12)
20 GRAPHICS 0: ? : ? "TECLEE UNA MELODIA"
25 ? : ? "PULSE TECLAS 1-9, 0, <, > PARA PRODUCIR NOTAS";
27 ? "SUELTE UNA TECLA DESPUES DE PULSAR OTRA"
28 ? "DE OTRA FORMA HABRA UNA DEMORA"
30 FOR X=1 TO 37: READ A: CHORD(X)=A: NEXT X
40 FOR X=1 TO 12: READ A: TUNE(X)=A: NEXT X
50 OPEN #1, 4, 0, "K:"
55 OLDCHR = -1
60 A = PEEK(764): IF A = 255 THEN 60
63 IF A = OLDCHR THEN 100
65 OLDCHR = A
70 FOR X=1 TO 12: IF TUNE(X)=A THEN SOUND 0, CHORD
(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: OLDCHR = -1: GOTO 60
200 DATA 243, 230, 217, 204, 193, 182, 173, 162, 153, 144, 136, 128,
121, 114, 108, 102, 96, 91, 85, 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, 29, 27, 51, 53, 48, 50, 54, 55

```

BLUES DEL ORDENADOR

Este programa genera notas musicales al azar para "escribir" unas muy interesantes melodías para el bajo programado.

```

1 GRAPHICS 0: ? : ? "BLUES DEL ORDENADOR"
2 PTR = 1
3 THNOT = 1
5 CHORD = 1
6 PRINT "BASS TEMPO (1 = RAPIDO)";
7 INPUT TEMPO
8 GRAPHICS 2 + 16: GOSUB 2000
10 DIM BAJO(3, 4)
20 DIM LOW(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:BAJO(X,Y)=A
60 NEXT Y
70 NEXT X
80 FOR X=1 TO 3:READ A:LOW(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 510
110 T=T+1
115 GOSUB 205
120 GOTO 100
205 IF RND(0)<0.25 THEN RETURN
210 IF RND(0)<0.25 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(CHORD,NT),10,NT*2
280 RETURN
510 IF BASS=1 THEN 700
520 BDUR=BDUR+1
530 IF BDUR<>TEMPO THEN 535
531 BASS=1:BDUR=0
535 SOUND 0,LOW(CHORD),10,4
540 SOUND 1,BAJO(CHORD,THNOT),10,4
550 RETURN
700 SOUND 0,0,0,0
710 SOUND 1,0,0,0
720 BDUR=BDUR+1
730 IF BDUR<>1 THEN 800
740 BDUR=0:BASS=0
750 THNOT=THNOT+1
760 IF THNOT<>5 THEN 800
765 THNOT=1
770 PTR=PTR+1
780 IF PTR=17 THEN PTR=1

```



```

790 CHORD=LINEA(PTR)
800 RETURN
1000 DATA 162,144,136,144,121,108,102,108,96,91,96
1010 DATA 243,182,162
1020 DATA 1,1,1,1,2,2,2,2,1,1,1,1,3,2,1
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 DEL"
2006 PRINT #6
2010 PRINT #6;"  ORDENADOR"
2030 RETURN

```

PROGRAMA DE CONVERSION DECIMAL/HEXADECIMAL

Este programa puede ser escrito y utilizado para convertir los números hexadecimales a números decimales y viceversa.

```

10 DIM A$(9),AD$(1)
20 GRAPHICS 0:?" :?"CONVERSIONES DE NUMEROS HEX."
30 ? :?"Teclee 'D' para conversión DEC a HEX "?:? "Teclee 'H'
   para conversión HEX a DEC.":INPUT A$
40 IF LEN(A$)=0 THEN 30
50 IF A$="H" THEN 300
60 IF A$<>"D" THEN 30
90 TRAP 90
100 ? :? "TECLEE UN NUMERO DECIMAL DEL 0 AL 99999999
   99."
110 ? "DEC:":;INPUT N
120 IF N<0 OR N>1E+10 THEN GOTO 100
130 I=9
140 TEMP=N:N=INT(N/16)
150 TEMP=TEMP-N*16
160 IF TEMP<10 THEN A$(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 ? :? "TECLEE UN NUMERO HEX. DEL 0 AL FFFFFFFF."  
320 ? "HEX: ";:INPUT A$  
330 N=0  
340 FOR I=1 TO LEN(A$)  
350 IF A$(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 PRINT "DEC: ";N:?  
390 GOTO 320  
400 END
```


APENDICE I

POSICIONES DE MEMORIA

Nota: Muchas de estas ubicaciones son de interés primordial para programadores expertos y están incluidas aquí para su conveniencia. Las etiquetas dadas son utilizadas por los programadores ATARI para hacer que los programas sean más fáciles de leer.

ETIQUETA	POSICION DECIMAL	POSICION HEXADECIMAL	COMENTARIOS Y DESCRIPCIONES
APPMHI	14,15	DE	Posición más alta utilizada por el BASIC (LSB, MSB)
RTCLOK	18,19,20	12,13,14	Contador del marco del TV (1/seg.) (LSB,NSB,MSB)
SOUNDR	65	41	Bandera I/O ruidosa (0 = silencio)
	77		Bandera del Modo de atracción (128 = Modo de Atracción)
LMARGIN	82,83	52,53	Margen Izquierdo, Derecho (Valores por defecto 2, 39)
RMARGIN			
ROWCRS	84	54	Fila actual del cursor (ventana gráfica)
COLCRS	85,86	55,56	Columna actual del cursor (ventana gráfica)
OLDROW	90	5A	Fila previa del cursor (ventana gráfica)
OLDCDL	91,92	5B	Columna previa del cursor (ventana gráfica)

ETIQUETA	POSICION DECIMAL	POSICION HEXADECIMAL	COMENTARIOS Y DESCRIPCIONES
	93	5C	Datos bajo el cursor (ventana gráfica si no Modo 0)
NEWROW	96	60	Fila del cursor donde irá el DRAWTO
NEWCOL	97,98	61,62	Columna del cursor donde irá el DRAWTO
RAMTOP	106	6A	Parte superior actual de la memoria (n.º de páginas)
LOMEM	128,129	80,81	Punto bajo de memoria BASIC
MEMTOP	144,145	90,91	Punto alto de memoria BASIC
STOPLN	186,187	BA,BB	N.º de línea donde ocurrió el STOP ó TRAP (n.º binario de 2-byte)
ERRSAV	195	C3	Error de número
PTABW	201	C9	Imprime el ancho del Tab (toma por defecto 10)
FRO	212,213	D4,D5	Bytes altos y bajos del valor a ser devuelto al BASIC de la función USR
RADFLG	251	FB	Indicador RAD/DEG (0 = radianes, 6 = grados)
LPENH	564	234	Valor horizontal del Light Pen
LPENV	565	235	Valor vertical del Light Pen
TXTRW	656	290	Fila del cursor (ventana de texto)
TXTCOL	657,658	291,292	Columna del cursor (ventana de texto)

ETIQUETA	POSICION DECIMAL	POSICION HEXADECIMAL	COMENTARIOS Y DESCRIPCIONES
COLOR0	708	2C4	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	2E5,2E6	OS parte superior del indicador disponible de memoria (LSB, MSB)
MEMLO	743,744	2E7,2E8	OS punto bajo de memoria
CRSINH	752	2F0	Suprimir el cursor (0 = cursor puesto, 1 = cursor suprimido)
CHACT	755	2F3	Registro del Modo de Carácter (4 = reflejo vertical; 2 = normal; 1 = en blanco)
CHBAS	756	2F4	Registro de la base del carácter (por defecto 224) (224 = mayúsculas, 226 = minúsculas) (204 = carácter internacional)
ATACHR	763	2FB	Ultimo carácter ATASCII
CH	764	2FC	Ultima tecla pulsada; código interno (255 suprime el carácter)
FILDAT	765	2FD	Datos para el relleno para gráficos Relleno (XIO)
DSPLFG	766	2FE	Indicador de Visualización (1 = visualización del control del carácter)

ETIQUETA	POSICION DECIMAL	POSICION HEXADECIMAL	COMENTARIOS Y DESCRIPCIONES
SSFLAG	767	2FF	Indicador Start/Stop para el paginado 90 = listado normal) asignado por CTRL1
HATABS	794	31A	Tabla de manejo de direcciones (3 bytes/manejador)
IOCB	832	340	Bloques de control de I/O (16 bytes/IOCB)
	1664,1791	680,6FE	RAM disponible
CONSOL	53279	D01F	Interruptores de la Consola (bit 2 = Option; bit 1 = Select; bit 0 = Start. POKE 53279,0 antes de lectura. 0 = interruptor presionado)
PORTA	54016	D300	PIA Clavija del Controlador del Puerto A de puertos I/O
PORTB	54017	D301	PIA Puerto B inicializado al hexadecimal 3C
PACTL	54018	D302	Registro de Control del Puerto A (en la grabadora de programas 52 = ON; 60 = OFF)
PBCTL	54019	D303	Registro de Control del Puerto B
SKCTL	53775	D20F	Registro de Control del Puerto en Serie. Bit 2 = 0 (última tecla presionada)

INDICE

A

Abreviaturas, 19-20.
ABS, 59.
Acceso aleatorio a la unidad de disco, 52.
Adata, 20.
ADR, 62, 92.
Aexp, 20.
Anotaciones
 En coma flotante, 65.
 En el manual, 18.
Aop, 20.
Armonía, 91.
ASC, 65.
ATASCII, 20, 69, 115-119.
ATN, 61.
Audio, 46.
Avar, 19.

B

BASIC, 1.
Biblioteca de funciones, 59.
 Propósito especial:
 ADR, 60.
 FRE, 60.
 PEEK, 60.
 POKE, 61.
 URS, 61.
 Frigonométricos:
 ATN, 61.
 COS, 61.
 DEG, 61.
 RAD, 61.
 SIN, 61.
Blues del ordenador, 155.
Brillo (ver Luminosidad).
Bubbe Sort, 142.
BYE, 27.

C

Cadenas
 Clasificación, 69.
 Comparación, 69.

Concatenación, 68.
Dimensión, 65.
Funciones:
 ASC, 65.
 CHR\$, 66.
 LEN, 67.
 STR\$, 67.
 VAL, 67.
Manipulación, 67.
Seccionado, 68.
Variables, 19.
Canales de Audio, 46.
Caracteres
 Asignación de colores, 87.
 ATASCII, 115-121.
 Control gráfico, 89.
 Internos, 89.
 Obligatorio, 51.
Cargador cogido hexadecimal, 156.
Cargador de Programas (G:)
CHR\$, 66.
CIO, 45.
CLEAR, 22.
CLOAD, 47.
CLOG, 59.
CLOSE, 51.
CLR, 74.
Cmdno, 55.
Código de dispositivo, 45.
Color
 Asignación, 87.
 Ausencia, 78, 84.
 Cambio, 82.
 Registro, 83.
COM (ver DIM).
Conservación
 Constantes, 17.
 Memoria, 97.
Construcción de tablas y matrices, 72.
CONT, 27.
Control de volumen, 92.
Controlador de juegos, 93.
Conversión Decimal/Hexadecimal, 156.

COS, 61.
CSAVE, 47.
CURSOR, 27.

D

Declaraciones

FOR, 35.
GOSUB, 37.
GOTO, 38.
IF, 38.
ON GOSUB, 41.
ON GOTO, 41.
POP, 42.
RESTORE, 43.
RETURN, 37.
STEP, 35.
THEN, 38.
TO, 35.
TRAP, 44.
STEP, 35.
STOP, 30.

Defectos (Ausencia de)

Colores, 78.
Márgenes en modo d, 76.
Tabulación, 22.
Unidad de disco, 55.

DEG, 61.

DIM, 72.

Disk drive

Número en ausencia, 46.
Requerimientos (ver el manual ATARI DOS).
Modificación de un programa BASIC, 57.

Distorsión, 92.

DOS, 48.

DRAWTO, 80.

E

Editado de pantalla, 31.
Editor de pantalla, 46.
Encadenado de programas, 56.
Enganche del DOS, 48.
END, 28.
Escala musical, 93.
Espacios, 97.
Exp, 20.

Expresiones

Aritméticas (ver aexp).
Lógicas (ver Lexp).
Cadenas (ver sexp).

F

Ficheros (nombre), 52.

Filospec, 20, 52.

Final archivo, 33.

FOR/NEXT, 35.

Con step, 35.

Sin step, 35.

FRE, 62.

Funciones (ver Biblioteca de funciones).

Funciones derivadas, 127.

G

Gaviota sobre el océano, 147.

Gráficos

Modos, 76.

Comandos:

COLOR, 79.

DRAWTO, 80.

GET, 82.

GRAPHICS, 75.

LOCATE, 80.

PLOT, 81.

POSITION, 82.

PUT/GET, 82.

SETCOLOR, 82.

XIO, 83.

GOSUB/RETURN, 37.

GOTO, 38.

Con ramificación condicional, 38.

I

Indicador interno de READ, 43.

INPUT, 48.

INT, 60.

Interrogación, 53.

I/O

Comandos:

CLOAD, 47.

CLOSE, 47.

CSAVE, 47.

DATA, 54.

DOS, 48.

ENTER, 48.

GET, 53.

INPUT, 48.

LOAD, 49.

LPRINT, 50.

NOTE, 60.

OPEN, 50.

POINT, 52.

PRINT, 53.

PUT, 53.

READ, 54.
SAVE, 55.
STATUS, 55.
XIO, 55.

Dispositivos:

Disk drive (D:), 46.
Teclado (K:), 46.
Impresora (P:), 46.
Interface RS-232 (R:), 47.
Grabador de programas (C:), 46.
Editor de pantalla (E:), 46.
Monitor TV (S:), 48.

J

JOYSTICK, 93.

L

Lazos

Sin final, 38.
Con final, 35.

LEN, 66.

LET, 16, 17, 28.

Letras

Minúsculas, 18.
Mayúsculas, 18.

Lexp, 30.

Lينو, 20.

Límite de nombre de una variable, 17.

LIST, 28.

Listado, 29.

LOAD, 49.

LOCATE, 80.

LPRINT, 50.

Luces, 145.

Luminosidad, 83.

M

Mapa de memoria, 123.

Márgenes en modo d, 76.

Matrices, 71.

Mensajes de error, 111.

Modificación de un programa BASIC, 57.

Modos

Gráficos, 75.

Operación

Directo, 21.
Indirecto, 21.
Ejecución, 21.

Texto, 71.

Monitor de TV (S:), 46.

N

NEW, 29.

O

ON/GOSUB, 41.

ON/GOTO, 41.

Operadores, 16.

Aritméticos, 20, 23.

Binarios, 23.

Lógicos, 23.

Relacionales, 24.

Uranios, 23.

Ordenes en cadena, 15.

Ordenes en encabezamiento, 19.

Ordenes

BYE, 27.

CONT, 27.

END, 28.

LET, 28.

LIST, 28.

NEW, 29.

REM, 29.

RUN, 29.

STOP, 30.

P

PADDLE, 94.

Paréntesis, 99.

PEEK, 60.

Periféricos (ver I/O).

Pila

Hardware, 64.

GOSUB, 37.

Pixel, 79.

PLA, 94.

POKE, 61.

POP, 42.

POSITION, 82.

Precedencia, 25.

PRINT, 18, 21, 35, 53.

Programas

Usando código máquina, 99.

Para el usuario, 139.

Punto y coma, 53.

PUT, 53.

R

RAD, 61.

RAM, 45.

Ramificación

Condicional, 40.

Incondicional, 38.

READ, 54.

- En modo directo, 54.
- Registro, 83.
- REM, 29.
- Repetición del teclado, 25.
- RESTORE, 43.
- RETURN, 37.
- RND, 60.
- RS-232 (R:), 47.
- RTS, 99.
- RUN, 29.

S

- Saldo de talonarios, 139, 142.
- Save, 55.
- SET COLOR, 82.
- Sexp, 20.
- SGN, 61.
- SIN, 61.
- Símbolo exponencial, 23.
- SQR, 61.
- SOUND, 91.
 - Terminación, 91.
- Supresión de líneas, 31.

T

- Tabulación, 22.
- Teclas
 - Funciones especiales:
 - Video inverso, 21.
 - BACK SPACE, 21.
 - BREAK, 22.
 - CAPS, 21.
 - DELETE, 22.
 - ESC, 21.
 - INSERT, 22.
 - RETURN, 23.
 - RESET, 22.
 - TAB, 22.
 - SHIFT, 32.
 - Control del cursor, 32.
- Teclado "K:", 40.
- Teclee una melodía, 152.
- Terminología, 15.
- Timbre
 - Definición, 93.
 - Valores, 93.
- TRAP, 44.

V

- Variables, 19.
- Ventana

Gráfica, 78.

Texto, 78.

Video Graffiti, 149.

Visualización en pantalla, 78.

Voces, 91.

X

X-Coordenada, 78.

XIO, 55.

XIO (Relleno), 85.

Y

Y-Coordenada, 78.

Z

Zumbador, 33.



ATARI[®]

DISTRIBUIDOR EXCLUSIVO

Unimport Ibérica, S.A.

C/. Dos Amigos 3 - 28015 Madrid