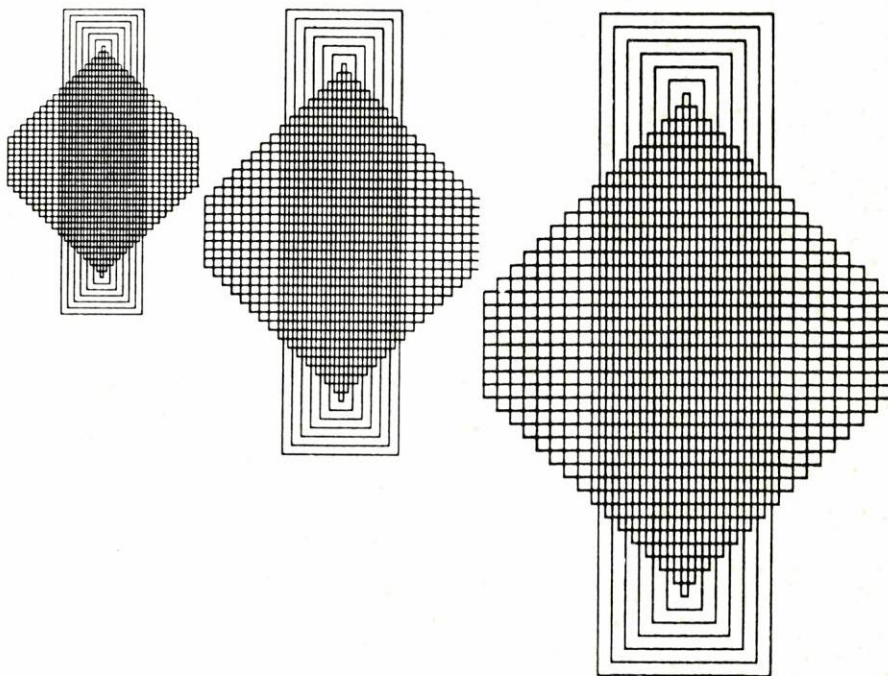


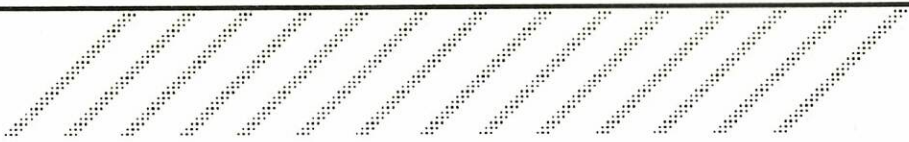
ATARI®

01857

ST BASIC™




GUIA DE REFERENCIA RAPIDA



Si está interesado en disponer de una documentación más detallada para la programación con ST BASIC, puede solicitar a ORDENADORES ATARI el MANUAL DEL USUARIO DE ST BASIC.

Este manual incluye una introducción completa para el programador que se inicia en Basic y, además, una sección exhaustiva para el programador adelantado. Con el MANUAL DEL USUARIO DE ST BASIC usted dispone de todo lo necesario para obtener los mejores resultados en sus trabajos de programación.

Para adquirir esta publicación, contacte con su distribuidor ATARI o dirijase a ORDENADORES ATARI, que se lo remitirá a su domicilio.






**ATARI
ST BASIC™**

GUIA DE REFERENCIA RAPIDA





Se ha realizado todo tipo de esfuerzo para asegurar la exactitud de la documentación del producto descrito en este libro. No obstante, dado que continuamente estamos mejorando y actualizando, tanto el hardware como el software de nuestros ordenadores, Ordenadores Atari S.A. no puede garantizar la exactitud el material impreso despues de la fecha de su publicación, por cuyo motivo no se hace responsable de los cambios, errores u omisiones.

ATARI, ST, ST BASIC, TOS son marcas registradas de ATARI Corp.


GEM es una marca registrada de Digital Research Inc.

ATARI

Copyright 1987, Atari Corporation

Sunnyvale, CA 94086


Todos los derechos reservados.





CONTENIDOS

| | |
|--|----|
| ST BASIC ampliado | 1 |
| Bienvenido al BASIC ampliado | 1 |
| Como utilizar esta guía | 1 |
| Que hay en el disco de lenguaje del ST . | 2 |
| Manual de usuario de ST BASIC | 2 |
| Comenzando | 3 |
| Carga del lenguaje ST BASIC | 3 |
| Convertir programas a ST BASIC | 5 |
| Leyendo un programa BASIC | 5 |
| Diferencias | 5 |
| ST BASIC | 5 |
| Otras versiones de BASIC | 7 |
| Códigos y mensajes de error | 11 |
| Guía rápida | 17 |
| Caracteres de declaración | 17 |
| Delimitadores | 17 |
| Comandos de edición | 17 |
| Operadores | 18 |
| Modo de dibujo (Draw Mode) | 18 |
| Tipo de relleno | 19 |
| Tipo de líneas | 19 |
| Puertas | 19 |
| Ventanas | 19 |
| Sonido | 20 |
| Lista de comandos | 21 |
| Lista de sentencias | 21 |
| Lista de funciones | 22 |
| Lista de variables del sistema | 22 |
| Comandos, sentencias, funciones y variables del sistema | 23 |





ST BASIC AMPLIADO

Bienvenido al BASIC ampliado.

La versión original de ST BASIC recibida con todos los ordenadores ST, ha sido reemplazada por esta nueva versión ampliada del lenguaje. Ambas son similares al BASIC standard, pero estas permiten la utilización de ventanas, barras de menus, y los elementos gráficos del entorno GEM. Es evidente, que tanto la versión original como esta nueva ampliada del lenguaje, utilizan los recursos de velocidad en el tratamiento de la información y capacidades gráficas que proporciona el sistema ST.

La nueva versión ampliada de ST BASIC se ejecuta unas tres veces más rápido que la original, y permite a la vez la utilización de un mayor número de funciones (Existen 33 nuevas palabras reservadas , permite un rango mayor para la utilización de variables enteras y se ha perfeccionado la sintaxis haciendola mucho más eficiente). Ha aumentado la lista de mensajes de error, siendo estos más claros en las explicaciones reflejadas para cada uno de los errores detectados.

El lenguaje ST BASIC ampliado, es compatible con la versión original del mismo, de esta manera Vd. podrá utilizar programas escritos en la versión anterior y ejecutarlos en esta nueva ampliada del lenguaje. Encontrará una descripción más detallada, en la sección "CONVERTIR PROGRAMAS A ST BASIC", para usar los programas creados anteriormente con esta nueva versión ampliada de ST BASIC.

COMO UTILIZAR ESTA GUIA

Esta guía está escrita, para programadores con experiencia que conocen el lenguaje BASIC, y estan familiarizados con los procedimientos standard utilizados dentro del entorno GEM.

Está preparada para que los programadores puedan analizar las diferencias entre la versión ampliada del lenguaje y la anterior. Tambien se especifican las características especiales de BASIC sobre el ordenador ST, y se proporcionan ejemplos para facilitar la carga y ejecución de programas escritos en otras versiones de BASIC, sobre la nueva versión ampliada del ST.



QUE HAY EN DISCO DE LENGUAJE DEL ST.

El disco de lenguaje del ST , que se proporciona junto con su ordenador, contiene los ficheros necesarios para rodar la versión ampliada de ST BASIC.

BASIC.PRG es el programa BASIC
BASIC.RSC es el fichero de recursos para el lenguaje.

Nota: Olvidese de cualquier otro fichero que pueda estar en el disco; pueden ser ficheros auxiliares o programas de aplicaciones. Sólo son necesarios los ficheros mencionados anteriormente para utilizar la versión ampliada de ST BASIC.

MANUAL DE USUARIO y TUTORIAL DE ST BASIC

El manual de usuario y el tutorial forman una guía completa de ST BASIC ampliado. El nuevo manual le permite un fácil acceso a todos los niveles de información sobre el uso del lenguaje. Aquellos de Vds. que comienzan a utilizarlo podrán encontrar en el un extenso manual de aprendizaje, mientras que los programadores con experiencia no carecerán del acceso a una amplia y rigurosa información técnica.

Si Vd. esta interesado en programar con ST BASIC ampliado, y desea disponer de una información más amplia a la reflejada en esta guía, contacte con su distribuidor ATARI quien se la podrá facilitar en las condiciones establecidas.

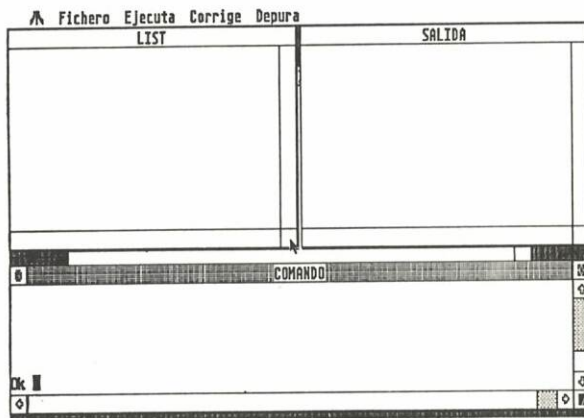
COMENZANDO

Antes de empezar a trabajar con ST BASIC, efectue una copia de seguridad del disco de lenguaje ST. Esto le protegerá contra la destrucción o deterioro accidental de su disco original. (Vea el manual de usuario de su ordenador ST, donde se explica como realizar una copia de seguridad).


Una vez realizada la copia de su disco original, está listo para comenzar a trabajar con ST BASIC. Comience por cargar el programa de lenguaje en su ordenador ST, para ello siga Vd. las siguientes instrucciones.

CARGA DEL LENGUAJE ST BASIC.

1. Con su ordenador conectado, y presentando la pantalla principal, efectue una doble pulsación en la tecla izquierda del ratón, sobre el icono que representa la unidad de disco A.
2. Cuando el directorio del disco aparezca en pantalla, pulse dos veces sobre BASIC.PRG. La pantalla correspondiente al lenguaje ST BASIC aparecerá, mostrando el siguiente aspecto:



Esta pantalla, es su entorno de programación ST BASIC



Nota: ST BASIC utiliza las técnicas standard de operación del entorno GEM para el acceso a los distintos elementos de los menus ,selección de opciones, manipulación de ventanas y carga de programas y aplicaciones. Estas técnicas se explican detalladamente en el manual de usuario entregado junto con su ordenador.



CONVERTIR PROGRAMAS A ST BASIC

INTRODUCCION

En esta sección se describen, las nuevas mejoras incluidas en el ST BASIC ampliado respecto a la anterior versión, y las diferencias principales entre ST BASIC y otras versiones del lenguaje. Utilice pues, la información que encontrará más adelante, para convertir programas escritos en otras versiones de BASIC de modo que puedan ejecutarse sobre el ST.

LEYENDO UN PROGRAMA BASIC

ST BASIC sólo podrá leer aquellos programas, que hayan sido almacenados en formato de ficheros de texto ASCII. Asegúrese de que los programas que desea transferir se encuentran en este formato.

Una línea de sentencias ST BASIC, debe empezar con un número de línea, terminar con un line feed (Salto de línea) y no contener más de 255 caracteres. El rango válido para números de líneas va desde 1 a 65529 (0 a 65529 en otras versiones de BASIC). ST BASIC no reconoce los caracteres de continuación de línea (line feed en algunas versiones del lenguaje), tenga en cuenta que en este caso la segunda parte o continuación de la línea se perderá.

Con las excepciones mencionadas, ST BASIC podrá guardar el texto del programa aunque incluya errores de sintaxis, posteriormente podrá utilizar el editor ST para modificar las sentencias necesarias. El camino más sencillo para convertir un programa BASIC es cargarlo en ST BASIC, ver el listado de errores y modificar las líneas correspondientes.

DIFERENCIAS

Las diferencias entre al BASIC ampliado y otras versiones del lenguaje, se especifican en los siguientes párrafos.

ST BASIC

ST BASIC ampliado es compatible con la versión anterior del lenguaje. Los programas escritos en ella pueden ser usados con la nueva versión, si se tienen en cuenta las diferencias que a continuación se especifican.

la instrucción DEF SEG ha sido eliminada. Sin embargo se han creado unas versiones especiales de PEEK y POKE palabras, bytes y largo . Estas son PEEK_W, PEEK_B, PEEK_L y POKE_W, POKE_B y POKE_L. Los programas escritos en la versión anterior que incorporen la instrucción DEF SEG deben ser modificados.

Los direccionamientos para PEEK y POKE usan enteros siendo su precisión suficiente.

Los números enteros tienen ahora 32 bits. Esto aumenta su rango de validez desde -2.147.483.648 hasta 2.147.483.647.

Ha sido introducida una nueva sintaxis para GEMSYS y VDISYS que trabaja de forma bastante más eficiente que la anterior. La sintaxis anterior podrá funcionar todavía con una adición: El número entre parentesis debe ser colocado en GEM_CONTRL(0) o GEMSYS o CONTRL(0) para VDISYS. Así los programas que utilicen VDISYS y GEMSYS deben ser modificados de acuerdo con la nueva sintaxis.

El siguiente listado presenta el conjunto de nuevas palabras reservadas que han sido añadidas a la nueva versión ampliada:

| | | |
|------------|--------------|---------|
| AREA | GEM_ADDRROUT | PATTERN |
| ASK MOUSE | GEM_CONTRL | PEEK_B |
| ASK RGB | GEM_GLOBAL | PEEK_L |
| BIOS | GEM_INTIN | PEEK_W |
| BOX | GEM_INTOUT | POKE_B |
| CLEAR | GEMDOS | POKE_L |
| DRAW | GSHAPE | POKE_W |
| DRAWMODE | LINEPAT | RGB |
| ED | MAT AREA | SSHAPE |
| ERR\$ | MAT DRAW | STATUS |
| GEM_ADDRIN | MAT SOUND | XBIOS |

Aquellos programas escritos en la versión anterior de ST BASIC que utilicen alguna de estas palabras, sin considerarlas palabra reservadas deben ser modificados.

Cualquier sintaxis que tenga que acceder a una lista de pares x,y esta documentada de forma que haya que poner ";" entre los pares. El punto y coma hace más fácil su lectura a pesar de que la sintaxis antigua siga siendo válida.

SYSTAB es ahora un array de 2 bytes enteros. El acceso a SYSTAB se realiza como elementos de un array cuyo indice es la mitad del desplazamiento previo.: EJEM. SYSTAB+6 se convierte en SYSTAB(3).

INP usado con -1 no devolverá nunca un número negativo. Sin embargo podrá devolver un número distinto de cero.

Ya no se permite la utilización del punto "." en los nombres de variables y en las palabras reservadas. Este debe ser sustituido por el carácter subrayado "_".

OTRAS VERSIONES DE BASIC

Identificadores

los nombres de variables y palabras reservadas en ST BASIC deben comenzar con una letra, pueden contener (Desde A-Z o a-z) y dígitos (0-9) y el carácter "_". Otros BASICS permiten el carácter "." pero no el "_" y en algunos casos existen diferencias significantivas entre mayúsculas y minúsculas.

Constantes cadena

ST BASIC le permite a Vd. insertar comillas en una constante de cadena de caracteres, siempre y cuando existan otras comillas que la compensen.

EJEMPLO:

```
PRINT ""ESTAS PALABRAS ESTAN ENTRECORNILLADAS.", PERO ESTAS NO."
```

IMPRIMIRA:

```
"ESTAS PALABRAS ESTAN ENTRECORNILLADAS.", PERO ESTAS NO.
```

Otras versiones de BASIC tratan las comillas como finalización de una constante cadena de caracteres y comienzo de otra. Teniendo en cuenta que los separadores entre elementos de una lista de PRINT no son obligatorios, se podrán encontrar sentencias tales como:

```
PRINT "A""B"
```

Otras versiones de BASIC no trabajan de la misma forma que ST BASIC, en el PRINT "A""B" se imprime como A"B mientras que en otras versiones se imprime como AB.

Variables aritmeticas

ST BASIC reconoce tres tipos de caracteres numéricos:

- ENTEROS
- COMA FLOTANTE EN SIMPLE PRECISION
- COMA FLOTANTE EN DOBLE PRECISION

Los enteros ocupan 4 bytes y abarcan un rango que va desde -2.147.483.648 hasta 2.147.483.647 , en otras versiones de BASIC ocupan 2 Bytes y abarcan un rango que va desde -32.768 hasta 32.767 . Esto afecta a las funciones MKI\$ y CVI y afectará al formato de los registros que contengan enteros.

ST BASIC evalúa las expresiones compuestas exclusivamente por terminos enteros, como un entero, lo cual puede significar que se pueda producir desbordamiento en los valores intermedios del calculo. Por ejemplo:

```
a%=b*c%-170000
```

es evaluado como entero.

Los valores en coma flotante se consideran en formato IEEE. Este tiene una pequeña diferencia en cuanto al rango dinámico y precisión con respecto a otros formatos. ST BASIC no reconoce números no normalizados, NANS o infinitos. Por ejemplo:

```
PRINT 1E-38
```

Imprimirá 0, ya que 1E-38 es menor, que el número más pequeño normalizado dentro del formato standard IEEE.

Cuando se efectuan operaciones que combinan diferentes tipos de números, el resultado es calculado en doble precisión, ya que la precisión de los enteros (31 bits más el signo) es mayor que la de los operandos en simple precisión coma flotante (24 bits más signo).

TRADUCCION E INTERPRETACION

ST BASIC traduce un programa en un conjunto de de códigos internos, cuando Vd. lo teclea o lo carga. Cuando un programa se hace correr (RUN), este código interno es ejecutado. Esto difiere de otras versiones de BASIC sobre otros microordenadores, los cuales traducen y ejecutan según avanza el programa, y causan diferencias en el efecto de las sentencias de declaración.

Las sentencias tipo DEF (Las cuales definen por defecto tipos de variables) alteran la acción del traductor. Ellas actuan según son tecleadas, en cualquier parte en el programa donde puedan ocurrir. Por esta razón el ejemplo que les mostramos a continuación no es posible en ST BASIC.

Ejemplo:

```
100 input "pulse un número de opción " ; a%
110 if a% = 1 then gosub 2000: goto 500
130 a = 5.0 : go to 600
500 a = "si"
600 print a
700 end
2000 defstr a
2010 return
```

DEF FN define una función de usuario en cualquier parte del programa donde se encuentre. La función DEF FN no tiene que ser ejecutada para definir la función. Vd. no puede tener dos sentencias DEF FN que definan la misma función. Por esta razón el siguiente ejemplo tampoco es posible en ST BASIC.

```
100 input "¿Que función desea <1> o <2>? " , a%
110 if a% = 1 then gosub 2000 else gosub 2100
120 print FNR(1,2)
130 end
2000 def fnr(x, y) = x/y
2050 return
2100 def fnr(x, y) = y/x
2150 return
```

FOR/NEXT Restricciones

ST BASIC precisa que cada sentencia FOR tenga emparejada exactamente su sentencia NEXT correspondiente, y que cada sentencia WHILE este análogamente emparejada con su correspondiente sentencia WEND. Este emparejamiento es analizado antes de que el programa sea ejecutado. Algunas versiones de BASIC analizan el emparejamiento según se ejecuta el programa, permitiendo construcciones de programa tales como:

```
100 FOR i% = 1 TO 1000
300 IF i% > 500 THEN NEXT
500 NEXT
```

ST BASIC comunicará la existencia de un error cuando va a ejecutar el programa. Otras versiones del lenguaje pueden ejecutarlo y generar un error tal como:

```
"NEXT without FOR" at line 300
"NEXT sin FOR" en línea 300
```



ENTORNO DE LOS COMANDOS

ST BASIC efectúa una distinción entre las sentencias (Por ejemplo PRINT) la cual forma parte de un programa pero no puede alterar el flujo del mismo, y aquellos comandos que se utilizan para analizar o alterar un programa, los cuales no pueden formar parte del programa mismo. Algunas versiones de BASIC permiten a estos comandos que formen parte del programa en el que operan. ST BASIC le comunicará la existencia de un error si encuentra esto en una línea de programa (línea numerada). Los comandos de ST BASIC se listan más adelante.



CODIGOS Y MENSAJES DE ERROR

Introducción

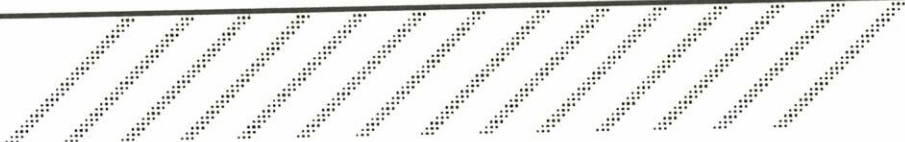
Los mensajes de error aparecen en la ventana COMANDO, tanto durante la introducción de un programa, como al teclear una línea de forma incorrecta, durante la ejecución de un programa o al detectar la inexistencia de una línea necesaria para el desarrollo del flujo del programa.


Descripción

Los siguientes códigos y mensajes de error se utilizan con ERL, ERR, ERR\$, y ERROR. (Para una explicación detallada de estas palabras reservadas, dirijase a la sección COMANDOS, SENTENCIAS, FUNCIONES y VARIABLES DEL SISTEMA).


CODIGO/MENSAJE


- 0-1 Undefined error**
(Error indefinido)
- 2 Syntax error**
(Error de sintaxis)
- 3 RETURN without GOSUB**
(RETURN sin GOSUB)
- 4 Out of data**
(Faltan Datos)
- 5 Illegal function call**
(Función de llamada ilegal)
- 6 Overflow**
(Desbordamiento)
- 7 Out of memory**
(Sin memoria)
- 8 Undefined line number**
(Número de línea indefinido)
- 9 Subscript out of range**
(Subíndice fuera de rango)
- 10 Duplicate Definition**
(Definición duplicada)
- 11 Division by zero**
(División por cero)

- 
- 12 **Illegal in immediate mode**
(Ilegal en modo inmediato)
 - 13 **Type mismatch**
(Distinto tipo)
 - 14 **Undefined error**
(Error indefinido)
 - 15 **String too long**
(Cadena demasiado larga)
 - 16 **Expression too complex**
(Expresión muy compleja)
 - 17 **CONT valid only in BREAK mode**
(CONT válido solo en modo BREAK)
 - 18 **Undefined user function**
(función de usuario indefinida)
 - 19 **Undefined error**
(Error indefinido)
 - 20 **RESUME without error**
(RESUME sin error)
 - 21 **Undefined error**
(Error indefinido)
 - 22 **Missing operand**
(Falta operando)
 - 23 **Program line too long**
(Línea de programa demasiado larga)
 - 24 **Undefined error**
(Error indefinido)
 - 49 **Undefined error**
(Error indefinido)
 - 50 **Field overflow**
(Desbordamiento del campo)
 - 51 **Invalid record length**
(Longitud de registro inválida)
 - 52 **Invalid file number**
(Número de fichero inválido)
 - 53 **File not found**
(Fichero no encontrado)

- 
- 54 **Invalid file mode**
(Modo de fichero inválido)
 - 55 **File already open**
(Fichero abierto)
 - 56 **Undefined error**
(Error indefinido)
 - 57 **Device I/O error**
(Error I/O en dispositivo)
 - 58 **File exist**
(Fichero existente)
 - 59 **Unable to create a file**
(Imposible crear fichero)
 - 60 **Undefined error**
(Error indefinido)
 - 61 **Disk full**
(Disco lleno)
 - 62 **End of file**
(Fin de fichero)
 - 63 **Invalid record number**
(Número de registro inválido)
 - 64 **Invalid filename**
(Nombre de fichero inválido)
 - 65 **Invalid character in program file**
(Carácter inválido en fichero de programa)
 - 66 **Direct statement in file**
(Sentencia directa en fichero)
 - 67 **KILL failed**
(Fallo en KILL)
 - 68 **Device unavailable**
(Dispositivo no disponible)
 - 69 **Undefined error**
(Error indefinido)
 - 92 **Undefined error**
(Error indefinido)
 - 93 **Undefined segment**
(Segmento indefinido)

- 94 **Protected file**
(Fichero protegido)
- 95 **Not a BASIC program**
(No es un programa BASIC)
- 96 **Undefined error**
98 (Error indefinido)
- 99 **—BREAK—**
- 100 **Undefined error**
(Error indefinido)
- 101 **Program too large**
(Programa demasiado largo)
- 102 **Undefined error**
(Error indefinido)
- 103 **Invalid line number**
(Número de línea incorrecto)
- 104 **Missing line number**
(Falta número de línea)
- 105 **Undefined error**
(Error indefinido)
- 106 **Statement not found**
(Sentencia no encontrada)
- 107 **Integer overflow**
(Desbordamiento de entero)
- 108 **Redo from start**
(Empezar de nuevo)
- 109 **Stop**
- 110 **GOSUBs nested too deep**
(Demasiados GOSUBs anidados)
- 111 **Invalid BLOAD file**
(Fichero inválido para BLOAD)
- 112 **Undefined error**
200 (Error indefinido)

- 
- 201 **Invalid option**
(Opción incorrecta)
 - 202 **Command not allowed here**
(Comando no permitido aquí)
 - 203 **Line number required**
(Se necesita número de línea)
 - 204 **FOR without NEXT**
(FOR sin NEXT)
 - 205 **NEXT Without FOR**
(NEXT sin FOR)
 - 206 **Comma missing**
(Falta una coma)
 - 207 **Parenthesis missing**
(Falta parntesis)
 - 208 **Option base must be 0 o 1**
Option base debe ser 0 o 1
 - 209 **Undefined error**
(Error indefinido)
 - 210 **WHILE without WEND**
(WHILE sin WEND)
 - 211 **WHEN without WHILE**
(WHEN sin WHILE)
 - 212 **Undefined error**
(Error indefinido)
 - 213 **Duplicate DEF FN**
(DEF FN duplicado)
 - 214 **Invalid jump into loop**
(Salto incorrecto dentro de un bucle)
 - 215 **Duplicate line number**
(Número de línea duplicado)
 - 216 **Duplicate label**
(Etiqueta duplicada)
 - 217 **Undefined error**
220 (Error indefinido)

- 
- 221 **System error #&u**
(Error de sistema #&u)
 - 222 **Program not run**
(Programa sin ejecutar)
 - 223 **Too many FOR loops**
(Demasiados bucles FOR)
 - 224 **Undefined error**
230 (Error indefinido)

GUIA RAPIDA

CARACTERES DE DECLARACION

| Carácter | Función |
|----------|--|
| % | Carácter de declaración de tipo entero. |
| \$ | Carácter de declaración de tipo cadena. |
| ! | Carácter de declaración de simple precisión. |
| # | Carácter de declaración de doble precisión. |

DELIMITADORES

| Carácter | Objeto |
|----------|----------------------|
| ' | Delimita comentarios |
| " | Delimita cadenas |
| ; | Delimita caracteres |
| , | Delimita argumentos |
| : | Delimita sentencias |

COMANDOS DE EDICION

| Tecla de FUNCION | Opción |
|------------------|-----------------------------|
| [F1] | Inserta espacio |
| [F2] | Borra caracter |
| [F3] | Inserta línea |
| [F4] | Borra línea |
| [F5] | Página arriba |
| [F6] | Página abajo |
| [F7] | Carga texto |
| [F8] | Guarda texto |
| [F9] | Nuevo Buffer |
| [F10] | Abandona ventana de edición |
| EDIT [Return] | Comienza edición. |

OPERADORES

Operadores aritméticos

Objeto

| | |
|------|---------------------------------|
| + | Sumas; Concatenación de cadenas |
| - | Restas; negaciones |
| * | Multiplicaciones |
| / | Divisiones |
| \ | Covierte a enteros y divide |
| o ** | Exponenciaciones |

Operadores relacionales

Significado

| | |
|----|-----------------------------|
| = | Igual a |
| < | Menor que |
| > | Mayor que |
| <= | Menor o igual que |
| >= | Mayor o igual que |
| <> | No igual que o distinto que |

Operadores lógicos

Objeto

| | |
|-----|-------------------------|
| AND | Realiza un AND lógico |
| EQV | Prueba de equivalencia |
| IMP | Prueba de implicación |
| NOT | Expresión negada |
| OR | Realiza un OR lógico |
| XOR | Realiza un OR EXCLUSIVO |

MODO DRAW (MODO DE DIBUJO)

Número

Modo

| | |
|---|----------------------|
| 1 | Sustitución |
| 2 | Transparente |
| 3 | OR EXCLUSIVO |
| 4 | Transparente inverso |



FILL PATTERN STYLE (TIPO DE RELLENADO DEL DIBUJO)

| Número | Tipo |
|---------------|-------------|
| 0 | Vacio |
| 1 | Sólido |
| 2 | Dibujo |
| 3 | Sombreado |

LINE PATTERN STYLE (TIPO DE LAS LINEAS DEL DIBUJO)

| Número | Tipo |
|---------------|--------------------------------|
| 1 | Continua |
| 2 | Guión largo |
| 3 | Punto |
| 4 | Guión punto |
| 5 | Guión |
| 6 | Guión punto punto |
| 7 | Patrón definido por el usuario |

PUERTAS

| Número | Puerta |
|---------------|---------------|
| 0 | Impresora |
| 1 | Modem |
| 2 | Consola |
| 3 | MIDI |

VENTANAS

| Número | Ventana |
|---------------|----------------|
| 0 | Edición |
| 1 | Listado |
| 2 | Salida |
| 3 | Comandos |


SONIDO

| Parámetros | Descripción | Rango |
|-------------------|---|---------------------|
| Duración | Tiempo en unidades 1/50 seg. antes del comienzo del siguiente sonido. | |
| Nota | Controla el tono: Posición de la nota en la escala. | 1 a 12 |
| Octava | Controla el tono: Número de octava | 1 a 8 |
| Voz | Número del canal de sonido | 1 a 3 |
| Volumen | Controla el volumen del sonido | (0=off a 15 máximo) |

LISTA DE COMANDOS

| | | | | |
|--------|--------|---------|-------|----------|
| AUTO | EDIT | LOAD | RUN | TRON |
| BREAK | ERA | MERGE | SAVE | UNBREAK |
| CONT | FOLLOW | NEW | STEP | UNFOLLOW |
| DELETE | LIST | RENUM | TRACE | UNTRACE |
| DIR | LLIST | REPLACE | TROFF | |

LISTA DE SENTENCIAS

| | | | |
|---------------|-----------------|---------------|------------|
| AREA | END | MAT SOUND | SOUND |
| ASK MOUSE | ERASE | NAME | SSHAPE |
| ASK RGB | ERROR | NEXT | STOP |
| BLOAD | FIELD | ON | SWAP |
| BOX | FILL | ON ERROR GOTO | SYSTEM |
| BSAVE | FOR | OPEN | WAVE |
| CALL | FULLW | OPENW | WEND |
| CHAIN (MERGE) | GEMDOS | OPTION BASE | WHILE |
| CIRCLE | GET | OUT | WIDTH |
| CLEAR | GOSUB | PATTERN | WRITE[#] |
| CLEARW | GOTO | PCIRCLE | XBIOS |
| CLOSE | GOTOXY | PELLIPSE | |
| CLOSEW | GSHAPE | PRINT[#] | |
| COLOR | IF | PRINT USING | |
| COMMON | INPUT[#] | PUT | |
| DATA | KILL | QUIT | |
| DEF FN | LET | RANDOMIZE | |
| DEFDBL | LINE INPUT[#] | READ | |
| DEFINT | LINEF | REM | |
| DEFSNG | LINEPAT | RESET | |
| DEFSTR | LPRINT | RESTORE | |
| DIM | LSET | RESUME | |
| DRAW | MAT AREA | RETURN | |
| DRAWMODE | MAT DRAW | RGB | |
| ELLIPSE | MAT LINEF | RSET | |



LISTA DE FUNCIONES

| | | | |
|-------|---------|--------|----------|
| ABS | FIX | LPOS | RIGHT\$ |
| ASC | FLOAT | MID\$ | RND |
| ATN | FRE | MKD\$ | SGN |
| BIOS | GEMSYS | MKI\$ | SIN |
| CDBL | HEX\$ | MKS\$ | SPACE\$ |
| CHR\$ | INP | OCT\$ | SPC |
| CINT | INPUT\$ | PEEK | SQR |
| COS | INSTR | PEEK_B | STR\$ |
| CSNG | INT | PEEK_L | STRING\$ |
| CVD | LEFT\$ | PEEK_W | TAB |
| CVI | LEN | POKE | TAN |
| CVS | LOC | POKE_B | VAL |
| EOF | LOF | POKE_L | VARPTR |
| ERR\$ | LOG | POKE_W | VDISYS |
| EPX | LOG10 | POS | |

LISTA DE VARIABLES DEL SISTEMA

| | | | |
|------------|--------------|------------|--------|
| CONTRL | GEM_ADDRROUT | GEM_INTOUT | PTSIN |
| ERL | GEM_CONTRL | INTIN | PTSOUT |
| ERR | GEM_GLOBAL | INTOUT | STATUS |
| GEM_ADDRIN | GEM_INTIN | PI | SYSTAB |

COMANDOS, SENTENCIAS, FUNCIONES Y VARIABLES DEL SISTEMA

| NOMBRE | FORMATO/DESCRIPCION |
|-----------|--|
| = | <variable> = <expresión numérica> "cadena" Asigna valores a una variable (ver LET) |
| ABS | ABS (<expresión numérica>) Devuelve el valor absoluto del argumento |
| AREA | AREA <lista de puntos> Dibuja un polígono relleno. |
| ASC | ASC (<expresión de cadena>) Devuelve el código ASCII correspondiente al primer carácter de la cadena. |
| ASK MOUSE | ASK MOUSE <x>,<y>, Devuelve las coordenadas de la posición del ratón y el estado de la tecla asignándolas a las variables x,y,b. |
| ASK RGB | ASK RGB <reg>,<r>,<q>, Asigna el valor actual de valores del rojo, verde y azul de la paleta especificada, a las variables listadas. |
| ATN | ATN (<expresión numérica>) Devuelve el arco tangente del argumento. |
| AUTO | AUTO[<línea de comienzo>][,<incremento>] Numera las líneas automáticamente. |
| BIOS | BIOS <expresión numérica>,<listarg> Genera una llamada del sistema operativo al BIOS. |
| BLOAD | BLOAD <nomfichero>,<dirección> Carga un fichero binario en memoria en la dirección especificada. |
| BOX | BOX[FILL] <x1,y1>;<x2,y2> La sentencia BOX dibuja un caja. |

| | |
|--------------------|--|
| BREAK | BREAK [<lista de números de línea>] Provoca detenciones en el programa, en las líneas especificadas en la lista. |
| BSAVE | BSAVE <espfile.>,<dirección>,<longitud> Graba la parte de memoria especificada por dirección y longitud sobre el fichero especificado. |
| CALL | CALL <expresión numérica> [[<lista de parámetros>]] Genera una llamada a una subrutina en lenguaje máquina. |
| CDBL | CDBL (<expresión numérica>) Convierte el argumento a un número en doble precisión. |
| CHAIN | CHAIN <espfile.>[,<descriptor de línea>] [,ALL] Reemplaza el programa en curso por el programa especificado ejecutándolo automáticamente. |
| CHAIN MERGE | CHAIN MERGE <espfile.> [,<descriptor de línea>] [,DELETE <lista de descriptores de línea>] Mezcla el programa en curso, con el especificado y ejecuta el resultado. |
| CHR\$ | CHR\$ (<expresión numérica>) Convierte un entero a una cadena de un carácter de acuerdo con el código ASCII. |
| CINT | CINT (<expresión numérica>) Convierte el argumento a un entero que a su vez es redondeado. |
| CIRCLE | CIRCLE <x>,<y>,<radio> [,<ángulo inicial,ángulo final>] Dibuja círculos y arcos. |
| CLEAR | CLEAR Pone todas las variables numéricas a 0 y asigna a las variables de cadena el carácter nulo. Este comando deja sin definir las matrices o arrays. |

| | |
|---------------|--|
| CLEARW | CLEARW <expresión numérica> Borra las ventanas de ST BASIC. |
| CLOSE | CLOSE[#] <número de fichero> Cierra el fichero de datos especificado finalizando el acceso I/O al fichero. |
| CLOSEW | CLOSEW <número de ventana> Cierra las ventanas BASIC especificadas. |
| COLOR | COLOR [<color del texto,color de relleno ,color de línea, índice, estilo>] Asigna el color del texto, relleno , trazado de dibujos y patrones. |
| COMMON | COMMON <variable> [, <variable> ...] Permite el traspaso de variables a otro programa encadenado (CHAIN). |
| CONT | CONT Continúa con la ejecución de un programa que ha sido detenido por la instrucción BREAK. |
| CONTRL | CONTRL (<desplazamiento>)=<expresión> Una variable del sistema relacionada con VDI que puede actuar como una matriz o array. |
| COS | COS (<expresión numérica>) Devuelve el coseno del argumento. |
| CSNG | CSNG (<expresión numérica>) Convierte el argumento a un número en simple precisión. |
| CVD | CVD (<cadena de 8 bytes>) Convierte una cadena de 8 bytes a un número en doble precisión. |
| CVI | CVI (<cadena de 4 Bytes>) Convierte una cadena de 4 bytes en un número entero. |
| CVS | CVS (<cadena de 4 bytes>) Convierte una cadena de 4 bytes a un número en simple precisión. |

DATA DATA <constante> [, <constante>...]
 Provee al programa con datos, que serán leídos a través de la sentencia READ.

DEF FN DEF FN <nombre de función>
 [(<lista de variables>)] = <definición>
 Define una función de usuario.

DEFDBL DEFDBL <letra>[-<letra>]
 Define un rango de iniciales (letras) como números en doble precisión.

DEFINT DEFINT <letra>[-<letra>]
 Define un rango de iniciales (letras) como números enteros.

DEFSNG DEFSNG <letra>[-<letra>]
 Define un rango de iniciales (letras) como números en simple precisión.

DEFSTR DEFSTR <letra>[-<letra>]
 Define un rango de iniciales (letras) como cadenas.

DELETE DELETE <número de línea>
 [-<número de línea>]
 Borra de la memoria el rango especificado de líneas del programa.

DIM DIM <nombre de matriz>
 (<subíndice>[, <subíndice>]...)
 [, <nombre de matriz> (<subíndice>, <...>)]
 Asocia el nombre de una variable con una matriz de dimensiones especificadas.

DIR DIR [(<unidad de disco> :)]
 [(<nombre fich.>) . [(<extensión>)]]
 Genera una lista total o parcial de los ficheros del directorio especificado.

DRAW DRAW <lista de puntos>
 Dibuja una línea, siendo sus extremos los especificados en la lista de argumentos.

DRAWMODE DRAWMODE <variable entera>
 Establece el modo en curso para el dibujo.

EDIT EDIT [(<número de línea>)]
 Llama al editor del ST, permitiendo la modificación del programa en curso.

ELLIPSE ELLIPSE <x>,<y>,<radio horizontal>,
<radio vertical> [,<ángulo inicial>,
<ángulo final>]
Dibuja una elipse.

END END
Detiene la ejecución del programa, cierra los ficheros y vuelve al nivel de comandos.

EOF EOF (<número de fichero>)
Detecta el fin de fichero.

ERA ERA [<unidad de disco:>][<nombre fich.>]
Borra un fichero del disco.

ERASE ERASE <nombre de matriz>
[,<nombre de matriz>].....
Borra una matriz.

ERL ERL = <línea de error>
Contiene el número de línea donde ha ocurrido el error.

ERR ERR = <código de error>
Contiene un código de error.

ERR\$ ERR\$(<n>)
Devuelve el mensaje de error del código especificado.

ERROR ERROR <expresión numerica>
Simula un error.

EXP EXP (<expresión numerica>)
Devuelve la base de logaritmos neperianos e elevada al exponente.

FIELD FIELD #<número de fichero>
<anchura del campo> AS <variable cadena>
[,<anchura del campo> AS <variable cadena>]
.....
Reserva espacio para los valores de las variables en el buffer de ficheros aleatorios.

FILL FILL <x>,<y>
Rellena figuras con colores o patrones definidos.

FIX FIX (<expresión numerica>
Trunca el arqumento convirtiendolo en un número entero.

FLOAT FLOAT (<expresión entera>)
Convierte un entero en un número en simple precisión.

FOLLOW FOLLOW <variable>[, <variable>...]
Realiza un seguimiento de los valores de las variables especificadas durante la ejecución del programa.

FOR ... TO FOR <variable contador> = <valor inicial>
TO <valor límite> [STEP <incremento>]
Crea un bucle en el programa.
Los valores pueden ser expresiones numericas o variables.

FRE FRE [(<expresión numerica>)]
Devuelve el número de bytes libres en memoria (No usados por ST BASIC)


FULLW FULLW <número de ventana>
Coloca el tamaño de las ventanas de ST BASIC igual a la pantalla completa.

GEM_ADDRIN GEM_ADDRIN (<desplazamiento>) =
<expresión>
Una variable del sistema relacionada con el GEM que puede actuar como una matriz o array.


GEM_ADDRROUT GEM_ADDRROUT (<desplazamiento>) =
<expresión>
Una variable del sistema relacionada con el GEM que puede actuar como una matriz o array.

GEM_CONTRL GEM_CONTRL (<desplazamiento>) =
<expresión>
Una variable del sistema relacionada con el GEM que puede actuar como una matriz o array

GEM_GLOBAL GEM_GLOBAL (<desplazamiento>) =
<expresión>
Análogo a los anteriores.



| | |
|-------------------|---|
| GEM_INTIN | GEM_INTIN (<desplazamiento>) = <expresión> Análogo a los anteriores. |
| GEM_INTOUT | GEM_INTOUT (<desplazamiento>) = <expresión> Análogo a las anteriores. |
| GEMDOS | GEMDOS <expresión numerica>, <listarg.> Genera una llamada del sistema operativo al GEMDOS. |
| GEMSYS | GEMSYS <código de operación AES> Accede al interface AES del sistema operativo. |
| GET | GET[#]<número de fichero> [, <número de registro>] Lee un registro de un fichero aleatorio que se encuentra en disco, en el buffer. |
| GOSUB | GOSUB <descriptor de línea> Transfiere el control del programa a una subrutina. |
| GOTO | GOTO <descriptor de línea> Efectúa un salto incondicional, haciendo que el programa continúe en la línea especificada. |
| GOTOXY | GOTOXY <posición de columna>, <posición fila> Coloca el cursor en el punto de la pantalla especificado por la columna y la fila. |
| GSHAPE | GSHAPE <x1>, <y1>, <matriz> Presenta en pantalla la trama almacenada en la matriz especificada. |
| HEX& | HEX& (<expresión numerica>) Devuelve como cadena el valor en forma hexadecimal del argumento. |



| | |
|---------------|---|
| IF | <p>IF <expresión lógica> THEN <número línea> <etiqueta></p> <p>IF <expresión lógica> GOTO <número línea> <etiqueta></p> <p>IF <expresión lógica> THEN <número línea> [ELSE <número línea> <etiqueta>]</p> <p>IF <expresión lógica> THEN <sentencia> [ELSE <sentencia>]</p> <p>Ejecuta condiciones y decide la dirección del flujo del programa de acuerdo con los resultados obtenidos.</p> |
| INP | <p>INP (<número de puerta>)</p> <p>Devuelve un valor de un Byte de la puerta de entrada seleccionada.</p> |
| INPUT | <p>INPUT; [<cadena de caracteres><< ,>] <variable>[,<variable>]...</p> <p>Permite la introducción de datos desde el teclado durante la ejecución del programa.</p> |
| INPUT# | <p>INPUT#<número del fichero>, <variable>[,<variable>]...</p> <p>Asigna datos de un fichero secuencial a las variables especificadas.</p> |
| INPUT% | <p>INPUT%(<número de caracteres>[,<#> <número de fichero>])</p> <p>Devuelve una cadena de longitud especificada desde el teclado o desde un fichero de datos.</p> |
| INSTR | <p>INSTR([<comienzo>],<cadena>, <cadena de referencia>)</p> <p>Busca una cadena dentro de otra y devuelve su posición.</p> |
| INT | <p>INT (<expresión numerica>)</p> <p>Redondea el argumento por defecto.</p> |
| INTIN | <p>INTIN(<desplazamiento>)=<expresión></p> <p>Una variable del sistema relacionada con VDI que puede actuar como una matriz.</p> |
| INTOUT | <p>INTOUT(<desplazamiento>)=<expresión></p> <p>Una variable del sistema relacionada con VDI que puede actuar como una matriz.</p> |

| | |
|--------------------|--|
| KILL | KILL <expresión de cadena> Borra un fichero. |
| LEFT\$ | LEFT\$(<cadena>,<número de caracteres> Devuelve la subcadena formada por el número de caracteres que se especifican empezando desde la parte izq. de la cadena. |
| LEN | LEN (<expresión de cadena> Devuelve la longitud de una cadena especificada. |
| LET | LET (variable) = <expresión numerica> "cadena" Asigna valores a una variable. |
| LINE INPUT | LINE INPUT [[;]"mensaje" [;]"mensaje"] <variable de cadena> Recoge información que proviene del teclado y la asigna a la variable cadena especificada. |
| LINE INPUT# | LINE INPUT# <número de fichero>, <variable de cadena> Lee datos desde un fichero secuencial y los asigna a la variable cadena. |
| LINEF | LINEF <par de puntos, par de puntos> Dibuja una línea desde los puntos que se indican en la lista. |
| LINEPAT | LINEPAT <estilo>[,<tipo>] Establece el estilo de las líneas. |
| LIST | LIST (<lista de descriptores de línea>] Presenta las líneas especificadas del programa en curso en la ventana de listado. |
| LLIST | LLIST (<lista de descriptores de línea>] Imprime las líneas especificadas en la impresora. |
| LOAD | LOAD <nombre de fichero> Carga el programa especificado. |
| LOC | LOC (<número fich.> o <número registro> Devuelve el número de registro o el número de caracteres leídos o escritos. |

LOF LOF(<número de fichero>
Devuelve la longitud del fichero.

LOG LOG(<expresión numerica>
Calcula y devuelve el logaritmo neperiano o natural del argumento.

LOG10 LOG10(<expresión numerica>
Calcula y devuelve el logaritmo en base 10 del argumento.

LPOS LPOS[<argumento simulado>]
Devuelve la posición de la cabeza de la impresora matricial.

LPRINT LPRINT[<lista de expresiones>]
Imprime datos a través de la impresora.

LPRINT USING <formato expresión de cadena>;
[<lista de expresiones>]
Imprime datos a través de la impresora justificandolos a la izquierda.

LSET LSET <variable de cadena> =
<variable de cadena>
Mueve datos a una cadena justificandolos a la izquierda.

MAT AREA MAT AREA <cantidad>, <matriz>
Dibuja un polígono relleno, tomando los vertices de la matriz especificada.

MAT DRAW MAT DRAW <cantidad>, <matriz>
Dibuja una línea, tomando los vertices de la matriz especificada.

MAT LINEF MAT LINEF <cantidad>, <matriz>
Dibuja una línea, tomando los vertices de la matriz especificada.

MAT SOUND MAT SOUND <matriz>
Somete la matriz a la acción del chip de sonido.

MERGE MERGE <nombre de fichero>
Mezcla el programa especificado con el residente.

MID\$ MID\$ ("cadena de caracteres", <comienzo> [, <longitud>])
Devuelve la subcadena especificada por comienzo y longitud desde la cadena de caracteres.

MID\$ MID\$ (<variable de cadena>, <comienzo> [, <longitud>])="cadena"
Sustituye una subcadena por otra en la cadena existente.

MKD\$ MKD\$ (<expresión numerica>)
Convierte números en doble precisión a cadenas.

MKI\$ MKI\$ (<entero>)
Convierte enteros a cadenas.

MKS\$ MKS\$ (<expresión numerica>)
Convierte números en simple precisión a cadenas.

NAME NAME <nombre anterior del fichero> AS <nuevo nombre del fichero>
Cambia el nombre de un fichero (renombrar)

NEW NEW [<nombre del fichero>]
Borra el programa existente en memoria y opcionalmente asigna un nombre al nuevo.


NEXT NEXT [<contador>][, <contador>]]...
Define el final de un bucle.

OCT\$ OCT\$(<expresión numerica>)
Devuelve como cadena el valor en forma octal del argumento.

ON ON <expresión> GOSUB <descriptor de línea> [, <descriptor de línea>].....
Define múltiples saltos a subrutinas.

ON <expresión> GOTO <descriptor de línea> [, <descriptor de línea>].....
Define múltiples saltos en el flujo del programa.

| | |
|--------------------------|---|
| ON ERROR GOTO | ON ERROR GOTO 0 <descriptor de línea> Define la línea de comienzo de una rutina de error. |
| OPEN | OPEN "modo", # <número de fichero>, "nombre fichero" [, <longitud del registro>] Abre el fichero de datos especificado. |
| OPENW | OPENW <número de ventana> Abre ventanas ST BASIC. |
| OPTION BASE | OPTION BASE <0 1> Fija la base para dimensionar matrices. |
| OUT | OUT <número de puerta>, <byte> Pone un valor de un byte en la puerta de salida seleccionada. |
| PATTERN | PATTERN <plano>, <matriz> Selecciona el tipo de relleno. |
| PCIRCLE | PCIRCLE <x>, <y>, <radio> [, <ángulo inicial>, <ángulo final>] Dibuja círculos sólidos y sectores circulares. |
| PEEK | PEEK_B(<dirección>) Devuelve un valor de 8 bits contenido en la dirección de memoria especificada. PEEK_W(<dirección>) Devuelve un valor de 16 bits contenido en dirección de memoria especificada. PEEK_L(<dirección>) Devuelve un valor de 32 bits contenido en dirección de memoria especificada. |
| PELLIPSE | PELLIPSE <x>, <y>, <radio horizontal>, <radio vertical>, <ángulo inicial>, <ángulo final> Dibuja una elipse sólida o sectores elípticos. |
| PI | PI = <variable> Contiene el valor de pi. |



POKE POKE_B(<dirección>, <datos>
Escribe un valor de 8 bits en la posición de memoria especificada.

 POKE_B(<dirección>, <datos>
Escribe un valor de 16 bits en la posición de memoria especificada.

 POKE_B(<dirección>, <datos>
Escribe un valor de 32 bits en la posición de memoria especificada.

POS POS(<número del fichero>
Devuelve el valor del número de caracteres que han sido escritos en la línea actual para el fichero seleccionado, si el número del fichero es 0 devolverá el número de caracteres escritos en la línea actual de la pantalla.

PRINT PRINT [<elemento a imprimir>]<;|,>
[<elemento a imprimir>[<;|,>]...]
? [<elemento a imprimir>]<;|,>
[<elemento a imprimir>[<;|,>]...]
Presenta datos en la pantalla.

PRINT USING PRINT USING <"formato de cadena">;
<lista de variables>
Imprime datos en pantalla usando el formato especificado.

 PRINT# <número de fichero>, USING
<"formato de cadena">;<expresión>
[,<expresión>...]
Imprime datos en un fichero usando el formato especificado.

PRINT# PRINT#<número de fichero>,
<elemento a imprimir>
[<elemento a imprimir>...]
Escribe datos sobre un fichero.

PTSIN PTSIN (<desplazamiento>)=<expresión>
Una variable del sistema relacionada con VDI que puede actuar como una matriz.

PTSOUT PTSOUT (<desplazamiento>)=<expresión>
Una variable del sistema relacionada con VDI que puede actuar como una matriz.

| | |
|------------------|---|
| PUT | PUT[#]<número de fichero> [,<número de registro>] Escribe registros en ficheros de acceso aleatorio. |
| QUIT | QUIT Abandona ST BASIC y vuelve a la pantalla principal. |
| RANDOMIZE | RANDOMIZE [<expresión numérica>] Activa el generador de números aleatorios. |
| READ | READ <variable>,<variable>,... Asigna elemento(s) contenido(s) en la sentencia DATA a las variables indicadas. |
| REM | REM <comentario> ' <comentario> Inserta comentarios en el programa. |
| RENUM | RENUM [<primer número de línea nuevo>] [,<incremento>][,<línea de cominzo>] [,<último número de línea>] Renumeras las líneas del programa. |
| REPLACE | REPLACE[<nombre fich.>] [,<lista de números de líneas>] Reemplaza la versión existente del programa por una nueva. |
| RESET | RESET Coloca el contenido de la ventana de salida en el buffer de gráficos. |
| RESTORE | RESTORE [<descriptor de línea>] Resetea el puntero de las sentencias DATA especificadas. |
| RESUME | RESUME [NEXT 0 <descriptor de línea>] Define los puntos de vuelta del flujo del programa desde la rutina de error. |
| RETURN | RETURN Marca el final de una subrutina. |
| RGB | RGB <reg>,<r>,<g>, Pone la paleta de color con las proporciones especificadas de rojo, verde y azul. |

RIGHT\$ **RIGHT\$(<cadena>, <entero>)**
 Devuelve una subcadena con el número de caracteres especificado desde el final de la cadena.

RND **RND(<expresión numerica>)**
 Devuelve un número aleatorio.

RSET **RSET <variable de cadena> = <expresión de cadena de caracteres>**
 Mueve datos a una cadena justificandolos a la derecha.

RUN **RUN [<nombre fich.>] [, <descriptor de línea>]**
 Ejecuta un programa.

SAVE **SAVE [<nombre fich.>] [, <descriptor de línea>]**
 Almacena el fuente del programa en curso.

SGN **SGN [<expresión numerica>]**
 Devuelve el signo de un número.

SIN **SIN (<expresión numerica>)**
 Devuelve el seno del argumento.

SOUND **SOUND <voz>, <volumen>, <nota>, <octava>, <duración>**
 Genera notas musicales.

SPACE\$ **SPACE\$(<expresión numerica>)**
 Devuelve una cadena rellena de espacios con la longitud especificada.

SPC **PRINT SPC (<expresión numerica>)**
 Inserta el número especificado de espacios en blanco en una cadena PRINT.

SQR **SQR (<expresión numerica>)**
 Devuelve la raíz cuadrada del argumento.

SSHAPE **SSHAPE <x1,y1>;<x2,y2>, <matriz>**
 Guarda una trama en la matriz especificada

| | |
|-----------------|---|
| STATUS | STATUS = <variable> Contiene el valor devuelto desde cada llamada al TOSTM, GEM, VDI o AES. |
| STEP | STEP [<nombre fich.>] [, <descriptor de línea>] ejecuta un programa línea a línea. |
| STOP | STOP Detiene la ejecución del programa. |
| STR\$ | STR\$ (<expresión numerica>) Convierte argumentos numericos en cadenas. |
| STRING\$ | STRING\$ (<expresión numerica>, <expresión numerica> <cadena>) Devuelve una cadena de longitud determinada rellena con los caracteres especificados. |
| SWAP | SWAP < lª variable>,< 2ª variable> Intercambia los valores de las variables especificadas. |
| SYSTAB | SYSTAB (<desplazamiento>)=<expresión> Genera una tabla de punteros e indicadores del sistema. |
| SYSTEM | SYSTEM Abandona ST BASIC y vuelve a la pantalla principal. |
| TAB | PRINT TAB (<posición de tabulación>) Inserta tabuladores en sentencias PRINT. |
| TAN | TAN (<angulo en radianes>) Devuelve la tangente del argumento. |
| TRACE | TRACE [<número línea>-<número línea>] Analiza la ejecución del programa, imprimiendo el número de línea en curso (<permite seguir el flujo del programa>) |
| TROFF | TROFF [<número línea>-<número línea>] Desactiva TRON. |
| TRON | TRON [<número línea>-<número línea>] Analiza la ejecución del programa, imprimiendo el número de línea en curso y los valores de las variables. Al igual que TRACE permite seguir el flujo del programa. |

| | |
|-----------------|---|
| UNBREAK | UNBREAK [<número línea> - <número línea>] Desactiva BREAK. |
| UNFOLLOW | UNFOLLOW [<variable> [, <variable>]...] Desactiva FOLLOW. |
| UNTRACE | UNTRACE [<número línea> - <número línea>] Desactiva TRACE. |
| VAL | VAL (<cadena de caracteres>) Devuelve el valor numerico de la cadena especificada. |
| VARPTR | VARPTR (<variavble> # <número de fichero>) Devuelve la dirección de una variable. |
| VDISYS | VDISYS [<argumento simulado>] Permite al usuario acceder al interface VDI del sistema operativo. |
| WAVE | WAVE <control> , <envolvente> , <forma> <periodo> , <retraso> Controla la forma de onda utilizada en la sentencia SOUND. |
| WEND | WEND Define el final de un WHILE. |
| WHILE | WHILE <expresión lógica> Define el comienzo y condiciones de un bucle indefinido. |
| WIDTH | WIDTH [#<número de fichero> ,] <anchura> Establece la anchura de la salida por pantalla. WIDTH LPRINT <anchura> Establece la anchura de la salida por impresora. |
| WRITE | WRITE [<expresión>][, <expresión>] Envia los datos a la pantalla. |
| WRITE# | WRITE# <número de fichero> , <expresión> [, <expresión>]... Envia los datos a un fichero secuencial. |



XBIOS

XBIOS <función>[,<listaarg.>]
Genera una llamada del sistema operativo al
XBIOS.

Nota: <expresión lógica> es una expresión numérica la cual se evalua asumiendo un valor entero. Si este valor es cero se toma por falsa, mientras que si es distinta de cero se toma por verdadera.





ATARI®

Copyright © 1987 Atari Corporation
Sunnyvale, CA 94086
Reservados todos los derechos



C100536-007
Printed in Taiwan
K. I. 2. 1988

