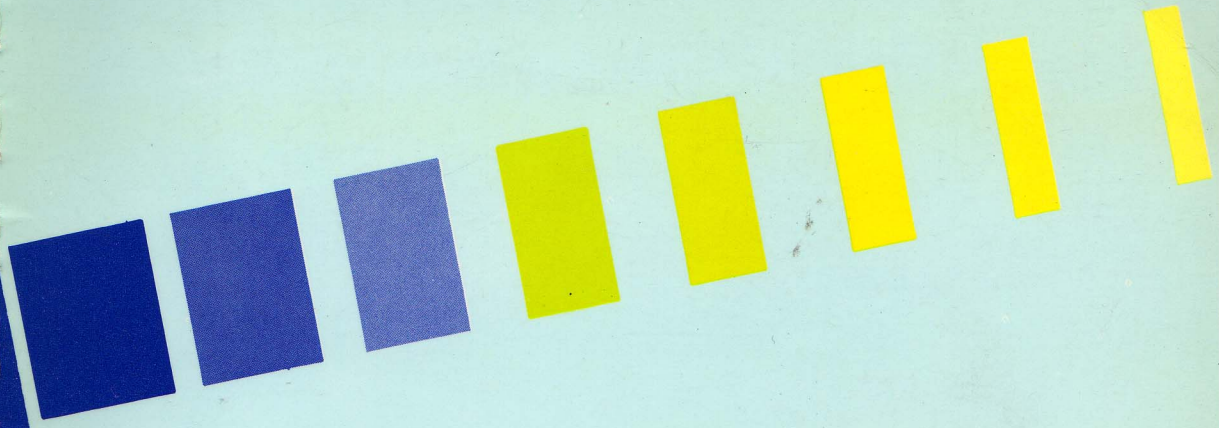
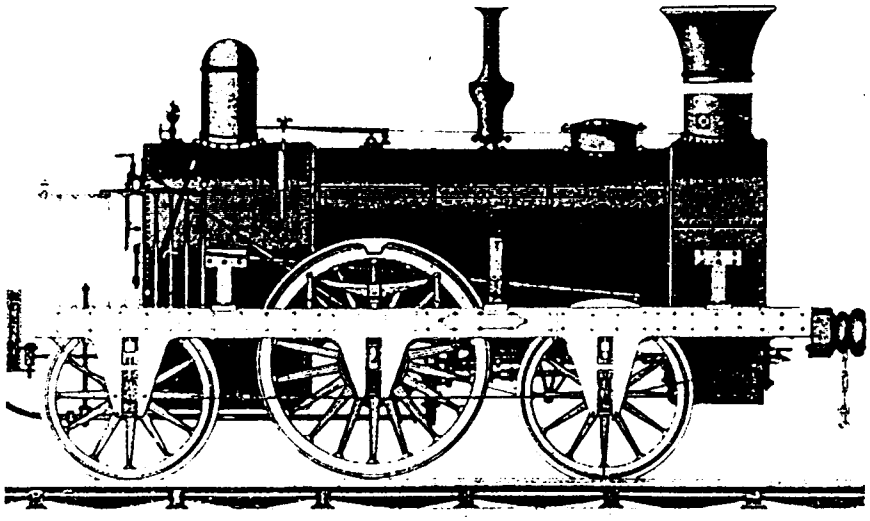
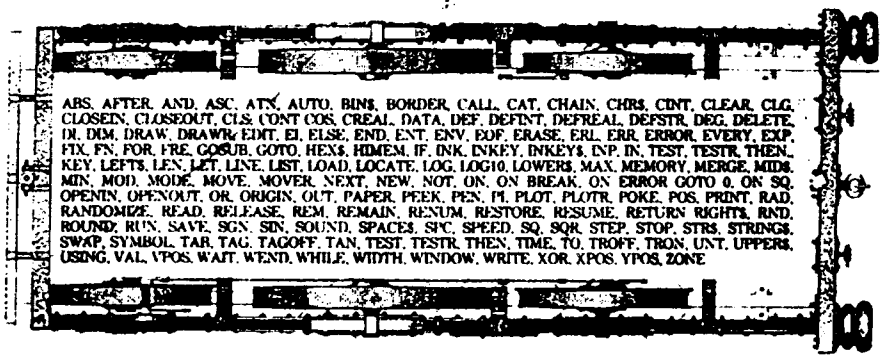


MANUAL DE REFERENCIA BASIC PARA EL PROGRAMADOR

AMSTRAD CPC·464

Manual de Referencia
Amstrad BASIC:
La Especificación Técnica Completa





Published by AMSOFT, a division of
Amstrad Consumer Electronics plc
Brentwood House
169 Kings Road
Brentwood
Essex

All rights reserved
First edition 1984
Reproduction or translation of any part of this publication without permission of the copyright owner is unlawful. Amstrad and Locomotive Software reserve the right to amend or alter the specification without notice. While every effort has been made to verify that this complex software works as described, it is not possible to test any program of this complexity under all possible conditions. Therefore Locomotive BASIC is provided "as is" without warranty of any kind either express or implied.

SOFT157 ISBN 1 85084 001 6
Copyright © 1984 Locomotive Software and Amstrad Consumer Electronics plc

Editado por **INDESCOMP, S.A.**
Avda. Mediterráneo, 9 - 28007 MADRID (ESPAÑA)

Derechos reservados en lengua española: **INDESCOMP, S.A.**

Traduce, compone e imprime: **CONORG, S.A.**

I.S.B.N.: 84-86176-12-3

Depósito Legal: M-4121-1985

PREFACIO

Esta descripción completa y concisa del interpretador BASIC de Locomotive Software's Mallard, tal y como está implementado en el AMSTRAD CPC464, está pensado para complementar la introducción hecha en la guía de usuario CPC464. No es una guía a la programación en BASIC, sino un trabajo importante de consulta que establece las reglas fundamentales y la utilización del lenguaje.

Las características del BASIC están cuidadosamente descritas usando una terminología formal, y debieras leer la descripción de este 'metalenguaje' antes de proceder a la parte principal de la descripción.

Las páginas se identifican usando la forma:

(Sección).(Número de Página)

se usan letras MAYUSCULAS, dado que muchas de las características ampliatorias están descritas en otra parte de este libro en las secciones del principio o del final, en cuyo contexto, el índice las identifica usando letras minúsculas/mayúsculas.

Recuerda que cada descripción de una palabra clave contiene una lista de palabras clave asociadas, que constituyen un añadido a las referencias crúzadas del contenido.

Observa también por favor, que se dispone de una publicación adjunta (Otoño 1984) que describe el sistema operativo y el programa pre-grabado del CPC464 (SOFT 158), y aconsejamos a los dedicados al código máquina y al BASIC avanzado que añadan esta publicación a su librería de referencia.

NOTAS SOBRE ESTILO DE LETRA

Observa por favor, que el estilo de los tipos usados en las publicaciones AMSOFT están pensados para ayudar a identificar las diferentes operaciones y secuencias usadas en la operación de la computadora:

El texto que se escribe mediante el teclado y aparece en la pantalla, se muestra con el estilo de tipos siguiente:

```
10 FOR N=1 TO 50
```

Las acciones en el teclado que instruyen una secuencia de comandos, pero que no tienen necesariamente una representación visual en la pantalla, se muestran en otro tipo de letra. Las teclas que no corresponden a caracteres visivos, se muestran encerradas entre corchetes cuadrados:

P	Cuando la letra 'P' aparece en la pantalla
[P] **[ESC]**	Como un comando, que no tiene carácter visivo correspondiente.

El texto general descriptivo se muestra en una dentro de la variedad de tipos, eg: Madeleine, Century, Palatino, Times, etc.

CONTENIDO

Introducción ——— 1.1

Los Elementos de BASIC ——— 2.1

Modos Directo y Programa ——— 3.1

La Pantalla ——— 4.1

Teclado y Joysticks ——— 5.1

Introducción de Líneas, Edición y Copiado ——— 6.1

Cauces de entrada y salida ——— 7.1

Generación de Sonido ——— 8.1

Interrupciones en BASIC ——— 9.1

Panorama de Comandos ——— 10.1

Los Comandos y las Funciones intrínsecas ——— 11.1

Palabras clave BASIC en orden alfabético ——— A.1

Números de error y mensajes de error ——— AP1.1

Rutinas y comandos externos ——— AP2.1

Palabras clave en BASIC (resumen) ——— AP3.1

Posiciones legales en pantalla de texto y caracteres de control ——— AP4.1

Números de tecla para teclado y joysticks ——— AP5.1

Tabla de colores ——— AP6.1

Notas sonoras y Períodos de tono ——— AP7.1

Lectura y Escritura con cassette ——— AP8.1

GESTION DE MEMORIA, SIMBOLOS DEFINIBLES Y 'BUFFERS' PARA CASSETTE ——— AP9.1

Indice de Contenidos

Las palabras clave se indican por letras MAYUSCULAS

ABS A.1
AFTER A.2
ASC A.3
ATN A.4
AUTO A.5
Atrape de errores 10.12

BIN$ B.1
BORDER B.2
Buffer en memoria AP9.2

CALL C.1
CAT C.2
CHAIN C.3
CHAIN MERGE C.3
CHR$ C.5
CINT C.6
CLEAR C.7
CLG C.8
CLOSEIN C.9
CLOSEOUT C.10
CLS C.11
CONT C.12
COS C.13
CREAL C.14
Caracteres (definibles por el usuario) .. 4.2
Caracteres (definidos por el usuario) .. 10.9
Caracteres de control AP4.1
Carga de programas 10.2
Cassette 10.5
Cauce (expresión de) 11.3
Cauce hacia cassette 10.5
Cauces de entrada/salida .. 7.1
Cauces hacia pantalla 10.5
Citación ('Llamada') AP2.2
Claves BASIC (detalle) A.1
Claves BASIC (resumen) .. AP3.1
Coma flotante binaria 2.11
Comandos 11.1
Comandos (externos) 2.12
Comandos para gráficos 10.7
Compatibilidad de datos ... 2.5
Contenedores de sitios reservados 1.1
Conversión de clases de datos 10.11
Copiado de líneas 6.1

DATA D.1
DEC$ D.2
DEF FN D.3
DEFINT D.4
DEFREAL D.4
DEFSTR D.4
DEG D.5
DELETE D.6
DI D.7
DIM D.8
DRAW D.9
DRAWR D.10
Datos 2.3
Datos (constantes) 10.7
Datos en memoria AP2.1
Designadores de clase 11.3
Dígito 1.2
Directo (Modo) 3.1

EDIT E.1
EI E.2
END E.3
ENT E.4
ENV E.6
EOF E.9
ERASE E.10
ERL E.11
ERR E.11
ERROR E.12
EVERY E.13
EXP E.14
Editado de programas 6.1
Elementos de BASIC 2.1
Enteras (expresiones) 11.1
Entero (sin signo) 2.6
Errores AP1.1
Escritura en el cassette AP8.1
Espacio en blanco 2.1
Expresión (literal-'string') 2.8
Expresión (logical) 2.9
Expresión (numérica) 2.7
Expresión (relacional) 2.8
Expresión 2.6, 11.1
Expresión Direccional 11.1
Externas (Rutinas) AP2.1
Externos (Comandos) AP2.2

FIX F.1
FOR F.2

FRE F.4
Ficheros en cassette 7.2
Fracciones Decimales 2.11
Frecuencia AP7.1
Función 2.10
Funciones Aritméticas 10.10

GOSUB G.1
GOTO G.2

HEX$ H.1
HIMEM H.2

IF I.1
INK I.2
INKEY I.3
INKEY$ I.4
INP I.5
INPUT I.6
INSTR I.9
INT I.10
Interrupciones BASIC 9.1
Interrupciones (comandos) 9.3
Interrupciones (temporizadas) 10.8
Introducción por cassette 10.6
Interaciones y Selecciones 10.3

JOY J.1
Joystick 5.1
Joystick (teclas/mandos) 10.9
Joystick, números de teclas AP5.1

KEY K.1
KEY DEF K.2

LEFT$ L.1
LEN L.2
LET L.3
LINE INPUT L.4
LIST L.6
LOAD L.7
LOCATE L.8
LOG L.9
LOG10 ..... L.10
LOWER$ L.11
Lectura del cassette AP8.1
Líneas (introducción) 1.1
Línea de programa 6.1
Literal -string- (constante) 2.3
Literal (expresión) 11.3
Literal (función) 10.11
Literal (variable) 11.3
Literal entrecomillado 11.2
Logical (expresión) 11.2

MAX M.1
MEMORY M.2
MERGE M.3
MID$ M.4
MIN M.6
MODE M.7
MOVE M.8
MOVER M.9
Máquina (operación a nivel de) 10.12
Memoria AP9.1
Metalenguaje 1.1
Modo 4.1

NEW N.1
NEXT N.2
Nombres 2.1
Nombres de Variables Múltiples 11.1
Notación aritmética 2.2
Notación científica 2.2
Notas y Tonos AP7.1
Numérica (expresión) 1.1
Numérica (variables) 11.2
Número de línea 11.1
Números 2.1
Número sistemado 2.2
Números de tecla AP5.1

ON BREAK GOSUB O.2
ON BREAK STOP O.3
ON ERROR GOTO O.4
ON SQ GOSUB O.6
ON <exp> GOSUB O.1
ON <exp> GOTO O.1
OPENIN O.7
OPENOUT O.8
ORIGIN O.9
OUT O.11
Origen 4.4

PAPER P.1
PEEK P.2
PEN P.3

PI P.4
PLOT P.5
PLOTR P.6
POKE P.7
POS P.8
PRINT P.9
Panorama de comandos 10.1
Pantalla (atributos de)..... 10.4
Pantalla (comandos de)..... 4.5
Pantalla (funciones de) 4.5
Pantalla de gráficos 4.4
Pantalla de texto 4.2
Programa (creación) 10.1
Programa (desarrollo)..... 10.12
Programa (modo)..... 3.2
Programa (ejecución) 10.2
Programa (terminación) 10.2
Programa gestionador del cassette AP8.1

RAD R.1
RANDOMIZE R.2
READ R.3
RELEASE R.4
REM R.5
REMAIN R.6
RENUM R.7
RESTORE R.9
RESUME R.10
RETURN R.11
RIGHT$ R.12
RND R.13
ROUND R.14
RUN <programa> R.15
RUN <línea> R.16
Rebase AP1.1
Redondeo 2.5
Reducida (tinta) 11.2
Repertorio de caracteres... 2.1
Resolución 4.1

SAVE 3.1
SGN S.3
SIN S.4
SOUND S.5
SPACE$ S.8
SPEED INK S.9
SPEED KEY S.10
SPEED WRITE S.11
SQ S.12
SQR S.13
STOP S.14
STR$ S.15
STRING$ S.16
SYMBOL S.18
SYMBOL AFTER S.17
Símbolos 'después de' AP9.2
Simple (variable) 11.2
Sintáctico (error)..... AP1.1
Sintáxis 11.1
Sistemas de coordenadas 4.4
Sonido..... 8.1
Sonido (comandos de) 8.3, 10.8
Sonido (generación de)..... 10.8
Subrutina AP2.2

TAG T.1
TAGOFF T.2
TAN..... T.3
TEST T.4
TESTR T.5
TIME T.6
TROFF T.7
TRON T.7
Tablas ('Arrays') 2.5
Tabla de colores AP6.1
Teclado 5.1
Teclado (comandos) 10.9
Teclado y Joystick 5.1
Temporizadores 9.1
Texto (exposición de) 10.4
Texto (introducción de)..... 10.6
Texto (posición en pantalla)..... AP4.1
Tinta reducida 11.2
Tintas y tintes 4.1
Tono (período del) AP7.1

UNT I.1
UPPER$ U.2

VAL V.1
VPOS V.2
Variables 10.3
Variables (nombre de) 11.3
Variables (clase de)..... 2.4
Ventanas 4.3
Viñetas para texto..... 4.3

WAIT W.1
WEND W.2
WHILE W.3
WIDTH W.4
WINDOW W.5
WINDOW SWAP W.6
WRITE W.7

XPOS X.1
YPOS Y.1
ZONE Z.1

1. Introducción

El interpretador BASIC es capaz de ejecutar los comandos descritos en la Sección 11. Cada comando está identificado por una o más palabras **clave** situadas al principio de cada frase, y puede tener un cierto número de **parámetros.** En general, cada parámetro puede ser una **expresión,** en la que intervienen constantes, variables y funciones como operandos, y signos de puntuación y operación. El BASIC admite datos literales ('strings'), y diversas formas de datos numéricos, así como el tratamiento secuencial de **ficheros.**

Los comandos se presentan al BASIC en **líneas.** Una línea puede constar de varios comandos, separados por dos-puntos, y con la única limitación en la longitud total de la línea. En el modo Directo las líneas se introducen mediante el teclado. En el modo Programa las líneas las lee del programa que en ese momento tenga depositado en memoria.

En el Modo Directo es posible añadir y suprimir líneas del programa en curso y enmendar las líneas existentes.

1.1 Introducción de Líneas

El BASIC acepta líneas procedentes de la consola con un máximo de 255 caracteres, y terminadas por un **retorno de carro.** Durante la introducción de líneas es posible 'editar' la línea en curso, y usar las facilidades de 'Copiado' mediante el cursor para insertar caracteres en dicha línea tomándolos de los expuestos en cualquier parte de la pantalla.

1.2 Metalenguaje

Con el fin de describir los comandos y sus parámetros, se utiliza un **metalenguaje** simple. La forma de cada comando se describe tal y como aparece cuando debe teclearse, con cualquier variable o cláusula opcional mostrada por 'contenedores de sitios reservados', que hacen referencia a un elemento conceptual definido en alguna otra parte.

Un elemento conceptual viene representado por su nombre encerrado entre corchetes angulados. Por ejemplo, en diversos comandos se requiere una expresión que produzca un valor numérico; Y eso se representa mediante:

```
<expresión numérica>
```

Todo lo que no está encerrado entre corchetes angulados se requiere tal y como se escribe. Por ejemplo, el comando para que **pare** tiene la forma:

```
STOP
```

Cuando hay una parte opcional en una definición, dicha parte se encierra entre corchetes cuadrados. Por ejemplo, si la expresión numérica fuera opcional en un comando, aparecería como:

[<expresión numérica>]

Si una parte opcional de un comando puede repetirse (de forma que aparezca tantas veces como se quiera, o ninguna en absoluto), se incluye un asterisco después del corchete cuadrado de cierre. Por ejemplo, una serie de dígitos, que requiere por lo menos uno de ellos, aparecería como:

<dígito>[<dígito>]*

En muchos comandos se usa una lista de parámetros con los elementos separados por comas. Se utiliza para describirlas una forma abreviada, que se comprende mejor mediante un ejemplo. Así:

<lista de: <expresión> es igual que <expresión>[,<expresión>]*
o bien: <lista de: [#]<número> es igual que [#]<número>[,[#]<número>]*

Observa que la lista puede tener un solo elemento. Si la lista contiene más de un elemento, cada elemento adicional debe estar precedido por una coma.

2. Los Elementos de BASIC

2.1 Repertorio de caracteres

Los caracteres en la máquina tienen valores asociados en la banda 0..255 (de modo que cada carácter ocupa en memoria un solo octeto). Los caracteres que intervienen en los literales pueden tener cualquier valor. En general, sin embargo, el BASIC supone que está utilizando el repertorio de caracteres ASCII.

Los caracteres con valores inferiores a 32 (&20, espacio en blanco) se tratan como caracteres no-visivos, y varios de ellos tienen un significado especial para el BASIC. Los caracteres con valores superiores a 126 (&7E, tilde) generalmente se ignoran.

Cuando el BASIC está procesando comandos y líneas de programa, no distingue entre letras minúsculas y mayúsculas, excepto cuando pertenecen a literales ('strings').

2.2 Espacios en blanco

Los espacios en blanco y todos los caracteres no-visivos distintos del 'retorno de carro' (o cualquier combinación en que intervenga) cuentan como <espacio blanco>. Dicho <espacio blanco> no tiene significado en las líneas de comando, excepto cuando tiene como efecto el delimitar un elemento.

2.3 Nombres

Las Variables y las palabras Clave tienen <nombre>s. El primer carácter de un <nombre> debe ser alfabético -una **letra**- y los demás caracteres pueden ser alfabéticos, numéricos o el **punto.** Los nombres pueden tener como máximo 40 caracteres, y todos ellos son **significativos.** Un <nombre> queda terminado cuando aparece cualquier carácter que no puede ser parte de un <nombre>. El BASIC no distingue entre caracteres en minúscula y mayúscula cuando trata con <nombre>s.

Las palabras **clave** del lenguaje también tienen la forma de <nombre>s, pero pueden incluir otros caracteres. Algunas claves se expresan por más de un <nombre> separados por espacios en blanco.

Las variables no pueden tener el mismo <nombre> que las palabras clave del lenguaje.

2.4 Números

Un <número> puede ser cualquiera de lo siguiente:

<número aritmético> o 'sin escala'
<número científico> o 'con escala'
<número sistemado> o 'con base'
<número línea>

Un <número aritmético> puede adoptar cualquiera de las siguientes formas:

<dígitos>
<dígitos>.[<dígitos>]
[<dígitos>].<dígitos>

siendo <dígitos> uno o una serie de dígitos decimales. <Espacio blanco> está permitido dentro de <número aritmético>, y puede usarse a voluntad como ayuda para mejor introducción de los números.

Un <número científico> adopta la forma:

<número aritmético>E[<signo>]<dígitos>

siendo <signo> el signo más o el menos, y <dígitos> uno o más dígitos decimales. Se llama **exponente** a la parte que comprende el <signo> y los <dígitos> que le siguen. El <espacio blanco> está permitido alrededor de los caracteres no-dígitos, pero no dentro de la porción de dígitos. El valor de tales números es el valor de <número aritmético> multiplicado por el resultado de elevar 10 al exponente dado. (Observa que el valor absoluto del exponente no debe exceder de 99).

Un <número sistemado> puede tener cualquiera de las siguientes formas:

&<cifras hexadecimales>
&H<cifras hexadecimales>
&X<cifras binarias>

siendo <cifras binarias> los símbolos empleados en el sistema de base 2 (el 0 ó el 1); y siendo <cifras hexadecimales> los símbolos empleados en el sistema de base 16 (los dígitos 0..9 y las letras A..F, o bien a..f). El equivalente en el sistema de base de 10 de todo <número sistemado> no puede ser mayor de 65535. El <espacio blanco> se permite alrededor de los caracteres que no son cifras, pero no dentro de la serie de cifras que lo componen. (Todo <número sistemado> se trata como ENTERO. Véase 2.6 posteriormente).

Un <número de línea> adopta la forma :

<dígitos>

siendo <dígitos> una serie de dígitos decimales con 1, al menos. Los números de línea están limitados a la banda 1..65535.

Un <número> es interpretado por el BASIC como terminado cuando encuentra cualquier carácter que no puede aparecer en un <número>.

2.5 Constantes literales (Strings)

Una <constante literal> es una **serie** discrecional de caracteres encerrada entre **comillas;** o que comienza con comillas y se termina por 'final de línea' (en cuyo caso los blancos posteriores no forman parte de la <constante literal>). Los caracteres que no pertenecen a la gama normal ASCII, pueden aparecer en toda <constante literal>, con la excepción del carácter Nulo (código de carácter 0). Una <constante literal> puede tener 255 caracteres como máximo. (Se suelen llamar cadenas de caracteres).

2.6 Clases de datos

El BASIC opera sobre dos amplias clases de datos, numéricos y alfanuméricos, (o literales).

Los literales (strings) son series de caracteres que pueden tener de longitud entre 0 (vacío) y 255. Los caracteres de una serie están representados por códigos o valores en la banda 0..255 (i.e., cada uno ocupa un byte). Se supone que se está usando el repertorio de caracteres ASCII, pero la mayoría del tratamiento de literales (strings) es independiente de los caracteres que componen dicho literal.

Los numéricos pueden ser de una de dos clases: Enteros o Reales.

Los datos numéricos Enteros ocupan en memoria dos bytes y están representados en el sistema de **complemento a doses,** de forma que pueden tener valores en la banda -32768..+32767. En algunas circunstancias, los **dieciséis** bits del sistema de complemento a doses se trata como un valor **sin-signo,** estando por tanto en la banda 0..65535. (Véase sección 2.10 posteriormente).

Los datos numéricos Reales ocupan en memoria cinco bytes y corresponden al sistema de **coma binaria flotante;** con cuatro bytes para la **mantisa** y un byte para el **exponente.** El valor absoluto máximo de los que pueden representarse es aproximadamente 1.7E+38; y el más pequeño (distinto de cero) es aproximadamente 2.9E-39. La mantisa de cuatro bytes permite una precisión ligeramente superior a **nueve cifras.**

La clase de un 'número' está implícita por su forma:

Un <número sistemado> es siempre Entero; y su valor sin-signo está en correspondencia biunívoca con su equivalente en complemento a doses (los valores sin-signo mayores de 32767 se corresponden exactamente al resultado de restar a 65536 el valor sin-signo).

Un <número aritmético> se trata como Entero si no existe **punto decimal** (que nosotros llamamos coma), y si su valor está incluido dentro de la banda de los Enteros. En caso contrario dicho número se trata como Real.

Todo <número científico> se trata como Real.

2.7 Variables

Las Variables tienen tres atributos: **nombre** de la variable, **clase** de la variable, y **organización.** Un <nombre variable> tiene la forma <nombre>[<designador clase>]. La clase de dato a la que hacemos referencia mediante un <nombre variable> viene especificada por su <designador clase>, o bien se supone que corresponde a la clase **prescrita para omisiones.** Todo <designador clase> es uno de los siguientes:

% para numerales Enteros
! para numerales Reales
$ para Literales

La clase prescrita para omisiones depende de la **inicial,** o primer carácter del <nombre variable>, y puede estipularse dinámicamente mediante los comandos DEFINT, DEFREAL y DEFSTR. La clase prescrita para omisiones de designación de clase, es siempre la de numerales Reales.

Las variables pueden ser en cuanto a organización: <variable simple> que corresponde a un solo dato, o <variable múltiple> que es una colección de variables que tienen todas exactamente la misma clase. Se suelen llamar también **tablas** ('arrays') y se describen más adelante.

Observa que el mismo <nombre> puede usarse para variables de diferentes clases y organizaciones, y que constituyen variables totalmente diferentes.

Las variables no necesitan ser declaradas en un programa de BASIC, sino que el espacio que ocupan en memoria queda reservado en cuanto aparece el <nombre> por primera vez, y con un valor de cero o nulo (literal vacío) según corresponda.

2.8 Tablas (Arrays)

El BASIC admite tablas de todas las clases de datos. Una tabla es una colección de variables de la misma clase, en la que puede señalarse cada elemento mediante el nombre colectivo de la tabla seguido del adecuado número de **sufijos,** en la forma:

<nombre variable>(<sufijos>)

siendo <sufijos>, también llamados subíndices, una <lista de:<expresión entera> y cada <expresión entera> da como resultado un valor dentro de la banda correcta.
(Observa que pueden usarse corchetes cuadrados en lugar de los paréntesis).

La clase de variables que componen la tabla está determinada por la clase del <nombre variable>.

Una tabla puede tener cualquier número de dimensiones. Las tablas pueden declararse explícitamente mediante el comando `DIM`, o implícitamente utilizando uno de sus elementos. Cuando se declara una tabla, se especifica el número de dimensiones y la cota superior de cada dimensión (en una declaración implícita sólo hay una dimensión y la cota superior es 10). Posteriormente en el programa, cualquier referencia a un elemento de la tabla debe usar el mismo número de subíndices, y cada uno debe pertenecer a la banda correspondiente. No es posible cambiar las dimensiones de una tabla, si previamente no se ha dejado **libre** el espacio que ocupa mediante el comando `ERASE`.

La cota inferior de todos los subíndices de una tabla es 0.

2.9 Compatibilidad y Conversión de Clases

El BASIC trata generalmente las diversas clases de numerales como compatibles entre sí. Los literales solamente son compatibles consigo mismo.

Cuando opera sobre numerales de diferentes clases, el BASIC los convertirá automáticamente a la clase que proceda. Cuando no hay otras restricciones, los numerales se 'amplían' para adaptarse a la 'mayor' clase involucrada, de manera que los Enteros se amplían a Reales. Dicha ampliación nunca puede fracasar.

Los números también son forzados a una determinada notación. Al convertir Reales a Enteros se **redondea** el número a un Entero, y el valor resultante debe pertenecer a la banda de los Enteros, -32768..+32767, o se generará un Error 6 (`Overflow` = rebase).

2.9.1 Redondeo

Hay diversas formas en que un número en una representación puede redondearse a otra representación más restrictiva. Un número dado X, en su representación mayor, cuando no puede ser representado en la menor, será acotado entre dos valores X1 y X2 (lo más cercanos posible y que puedan ser exactamente representados), de manera que X1 < X < X2. Los esquemas comúnes de redondeo escogen uno de esos dos valores de acuerdo con lo siguiente:

a. Truncamiento, o redondeo hacia cero

Los dígitos sobrantes simplemente se desprecian:

positivo X → X1,
negativo X → X2.

b. Redondeo hacia -Infinito

X → X1

c. Redondeo hacia +Infinito

X → X2

d. Redondeo al más próximo

X → X1 ó X2, dependiendo de cuál está más cerca de X.

Existen diversos esquemas para escoger entre X1 y X2, cuando X está exactamente a medio camino entre los dos. El BASIC redondea siempre en este caso alejándose de cero.

A menos que se estipule lo contrario, el término **redondeo** hace referencia a Redondeo al más próximo, tal y como hemos definido anteriormente.

2.10 Entero sin-signo

En algunas circunstancias el BASIC requiere un Entero en la banda completa sin-signo de 16 bits, que es de 0..65535; cuando por ejemplo, se examina una dirección de memoria mediante el comando PEEK. Los números en coma flotante se redondean a enteros, que han de estar en la banda -32768..+65535. El equivalente sin-signo de los números negativos en notación de complemento a doses es la que se usa en esos casos; es decir, los números negativos se tratan como 65536+número.

2.11 Expresiones

Hay cuatro clases de expresiones: numéricas, literales, relacionales y logicales. La precedencia o prioridad de operadores está designada de manera que las expresiones se evalúen como pudiera esperarse.

Cuando el orden de evaluación no está forzado por las reglas prescritas de prioridad de operadores, o por paréntesis, la evaluación procede de izquierda a derecha.

2.11.1 Expresiones Numéricas

La sintaxis de las expresiones numéricas es:

<expresión numérica> es: <numeral>[<operador><numeral>]*
<numeral> es: <menos monádico><numeral>
o bien: <constante numérica>
o bien: <variable numérica>
o bien: <función numérica>
o bien: (<expresión numérica>)
o bien: (<expresión relacional>)

Siendo <variable numérica> una referencia a una variable simple o a un elemento de una tabla de variables numéricas. Una <función numérica> es cualquiera de las que en el sistema, o definida por el usuario, da como resultado de la evaluación un valor numérico.

Los <operador>es, en orden de precedencia son:

↑	Exponenciación	Eleva el valor a la izquierda al valor especificado a su derecha. Fuerza a que ambos valores sean Reales antes de la operación, y produce un resultado Real.
-	Signo monádico	No afectando a la sintaxis, el signo menos Monádico tiene una precedencia inferior que la Exponenciación.
*	Multiplicar	Multiplicación Real o Entera.
/	Dividir	División Real. Si algún valor es Entero se fuerza a ser Real antes de la operación.
\	División Entera	Fuerza ambos valores a Enteros. El resultado se trunca a Entero.
MOD	Resto División (Módulo)	Fuerza ambos valores a Enteros. El resultado es el resto después de División Entera.
+	Adición	Suma Real o Entera.
-	Sustracción	Resta Real o Entera.

Observa que la Exponenciación es la de máxima prioridad y que la Multiplicación y División tienen igual prioridad, al igual que la Adición y la Sustracción. Excepto cuando el operador exige condiciones particulares, los valores situados a ambos lados del signo de operación son **forzados** a pertenecer a la misma clase por 'ampliación' de clase.

La división por cero genera un Error 11. Si se ha estipulado un ON ERROR GOTO nn, la subrutina se invoca de la manera habitual. Si no hay estipulada ninguna ON ERROR GOTO nn, la división entera por cero provoca el cese de la ejecución del programa; pero la división real por cero no termina dicha ejecución, ya que el cero puede ser el resultado de un 'rebase por defecto' en una operación anterior. En lugar de ello, se genera el mensaje 'Division by zero', y se entrega como valor el máximo número representable (aprox. 1.7E+38) con el signo apropiado, y la ejecución del programa continúa.

La división entera por cero genera un Error 11, en la forma habitual.

En el caso de un 'rebase por exceso' cuando los operandos son Enteros, los valores son ampliados a Reales y se repite la operación.

En el caso de rebase por exceso, cuando los operandos son Reales, se genera un Error 6 ('Overflow' = rebase). Si se ha estipulado una ON ERROR GOTO nn, la subrutina se activa de la manera habitual. Si no está estipulada ninguna ON ERROR GOTO nn, se genera el mensaje 'Overflow', entregándose el mayor número representable (aprox. 1.7E+38) con el signo apropiado, y continuando la ejecución.

2.11.2 Expresiones Literales (Strings)

La sintaxis de las expresiones literales es:

<expresión literal> es: <literal>[+<literal>]*
<literal> es: <variable literal>
o bien: <constante literal>
o bien: <función literal>
o bien: (<expresión literal>)

Siendo <variable literal> una referencia a una variable simple o a un elemento de una tabla de variables literales. Una <función literal> es cualquiera de las del sistema, o definida por el usuario, que al ser evaluada da como resultado una constante literal.

El efecto del signo de operación + sobre los literales es **concatenarlos**, o empalmarlos. Se produce un nuevo literal constituido por el primer operando literal inmediatamente seguido del segundo operando. Si el literal resultante fuera mayor de 255 caracteres, se genera el Error 15.

2.11.3 Expresiones Relacionales

Las expresiones relacionales comparan dos numerales o dos literales. Por lo tanto:

<expresión relacional>
es: <expresión numérica><operador relación><expresión numérica>
o bien: <expresión literal><operador relación><expresión literal>

Observa que todo <operador relación> tiene una precedencia o prioridad inferior que cualquiera de los operadores que aparezcan en las expresiones numéricas o literales, y mayor precedencia que cualquiera de los operadores logicales que pueda haber.

El <operador relación> es uno de los siguientes:

<	Menor que
<= =<	Menor que o Igual
=	Igual
>= =>	Mayor que o Igual
>	Mayor que
<>	Distinto de (no igual a)

Cuando los operandos son numéricos, se fuerza a que tengan la misma clase por el proceso de 'ampliación' descrito en la sección 2.9 anterior. El significado de la **relación** es el que puede esperarse.

Cuando los argumentos son literales, el significado de las **relaciones** requiere alguna explicación adicional:

Dos literales son iguales cuando son de la misma longitud, y los caracteres **homólogos** son los mismos.

Un literal es menor que otro si:

son iguales hasta el final del primero y el segundo es más largo.

o bien: el primer carácter que sea distinto en los dos, es más pequeño (y se considera más pequeño cuando el valor asociado al carácter **-el código-** es de menor valor).

El BASIC no admite datos de la clase **Booleanos.** Toda <expresión relacional> produce un valor Entero, que es -1 para **cierto** y 0 para **falso.** Observa que el comando condicional `IF` y el bucle condicional `WHILE` consideran un valor de 0 como falso, y cualquier otro valor como cierto.

2.11.4 Expresiones Logicales

El BASIC no admite la clase Booleana. Toda expresión logical efectúa en realidad operaciones booleanas sobre Enteros, teniendo en cuenta su **equivalente binario.** Toda <expresión lógica> es de la forma:

<operando>[<operador lógico><operando>]*

<operando> es: `NOT` <operando>
o bien: <expresión numérica>
o bien: <expresión relacional>
o bien: (<expresión lógica>)

Los operandos sobre los que trabajan los operadores lógicos son forzados a Enteros; y se generará un Error 6 si cualquiera de esos operandos no pertenece a la banda de Enteros. Luego se efectúa la operación indicada para cada uno de los 16 bits que representan dichos Enteros usando el sistema de complemento a doses.

El operador monádico NOT invierte simplemente cada bit del operando (0 pasa a 1, y viceversa; es la operación lógica 'negación').

Los operadores diádicos, en orden de prioridad y su efecto sobre cada bit es el siguiente:

AND	El resultado es 0 a no ser que ambos bits operandos sean 1.
OR	El resultado es 1 a no ser que ambos bits operandos sean 0.
XOR	El resultado es 1 a no ser que ambos bits operandos sean iguales.

El resultado de toda <expresión relacional> es el valor -1 o el valor 0. La representación para -1, la 'certeza', es en el **equivalente binario** todos los bits a 1; la representación para 0, la 'falsedad' es un equivalente binario con todos los bits a 0. El resultado de toda <expresión lógica> es uno de esos dos valores.

2.12 Funciones

Las funciones en BASIC son **subrutinas** internas del programa interpretador, que 'absorben' un determinado número de **argumentos** que son elaborados según un determinado método, y que entregan un valor que se denomina **resultado** de la función. Cuando las funciones son especificadas como operandos de una expresión, el valor de ese operando es el valor entregado por la función como resultado. Hay dos clases de funciones en BASIC: las **intrínsecas** del lenguaje y las 'definidas' por el usuario en el programa.

2.12.1 Funciones intrínsecas

Son parte inherente del lenguaje BASIC, y se describen en la sección 11. A diferencia de las funciones de usuario, la clase o índole del resultado no viene necesariamente dictado por el nombre que tenga la función. Por ejemplo, la función ABS siempre da un resultado de la misma clase que tenga el argumento. Algunas funciones intrínsecas tienen parámetros opcionales, y algunas permiten diferentes clases de argumentos; y ninguna de estas posibilidades están admitidas con las funciones de usuario.

2.12.2 Funciones de usuario

El usuario puede definir sus propias funciones mediante el comando DEF FN. Este comando adscribe o asocia un nombre con una lista de argumentos **formales** y con una expresión numérica o literal.

El nombre que identifica la función tiene la forma FN<nombre>[<designador clase>], en donde las siglas FN sirven sólo para distinguir el nombre de las funciones de los nombres de las variables; pero no interviene en la cuenta de los cuarenta posibles caracteres de todo <nombre>. Al igual que con las variables, funciones que tengan el mismo nombre pero tengan distinto designador de clase, son consideradas funciones diferentes.

La lista de argumentos **formales** enuncia un cierto número de variables que son 'locales' de esa función. Cuando posteriormente se cita, o 'invoca' esa función, debe hacerse dándole argumentos **actuales** que deben corresponder en lugar y en clase con los argumentos formales descritos en la definición - observando que el BASIC automáticamente cambiará los argumentos numéricos a la clase que correspondan en la definición. Los argumentos actuales son expresiones, que se evaluarán en el momento que se cita, o 'llama' la función, y cuyos valores son transferidos a la **fórmula** que define la función, reemplazando a los argumentos formales que intervienen en ella.

El cuerpo de la función, la **fórmula**, es una expresión simple. No es posible tener ninguna forma de estructura de control dentro de una función. Dicha expresión puede incluir referencias a otras funciones y a variables 'globales' así como a los argumentos formales de la función - aunque desde luego, un argumento formal que tenga el mismo nombre que una variable global, hace que el valor de la variable global sea inaccesible o esté alterado.

Cuando se evalúa una función, el resultado de la expresión que la define se convierte a un dato de la misma clase que el designado por el nombre de la función, y se entrega como resultado dicho valor convertido.

2.13 Fracciones Decimales y Coma Flotante Binaria

Es un triste hecho que sólo algunas fracciones decimales tienen una representación exacta como números binarios en coma flotante. Los números en coma flotante se redondean antes de ser exhibidos en pantalla de manera que puede quedar oscurecida una representación inexacta.

El efecto puede verse al comparar valores que son resultado de operaciones aritméticas separadas. Los dos valores aparentemente son iguales al ser expuestos por pantalla, pero sus representaciones binarias pueden ser diferentes. El mismo efecto puede aparecer cuando se restan dos de tales números y el resultado no es cero, aunque lo parezca.

A menos que se eviten totalmente las fracciones decimales, multiplicando todos los números por las potencias adecuadas de 10 (tratando todos los valores monetarios como céntimos, por ejemplo) es imposible evitar esos efectos. Sin embargo, pueden reducirse estipulando explícitamente la función intrínseca ROUND, que **redondea** números que en pantalla parecen iguales a exactamente el mismo valor.

Alternativamente se pueden retener ligeras diferencias, y tener en cuenta su existencia, cuando se comparan o restan dichos números.

2.14 Comandos Externos

Es posible 'ceder' la ejecución de algunos comandos a otros programas interpretadores de la máquina -bien estén en memoria o en cartuchos ROM de memoria de ampliaciones. Estos comandos se distinguirán de los comandos BASIC, prefijando el <nombre> con una barra vertical (|). Cuando el BASIC encuentra un comando externo, intenta primero buscar el programa externo al que 'reclamar' la ejecución. Si el comando es aceptado, el BASIC interpreta el resto de dicho comando externo como expresiones, separadas por comas; evalúa esas expresiones y luego traspasa esos valores como argumentos para la subrutina que ejecutará el comando externo.

En lo que a BASIC concierne, los <nombre>s de los comandos externos adoptan la forma de un 'nombre' en BASIC. Los caracteres en el nombre identificativo del comando externo se convierten a mayúsculas antes de que empiecen a escrutarse las tablas de comandos externos. La barra vertical es simplemente una **marca** para indicar al BASIC que inmediatamente detrás viene un comando externo - pero no forma parte del <nombre> identificativo de dicho comando externo.

Los comandos **externos** pueden ser de dos clases: de 'retaguardia' y de 'vanguardia'. Un comando de 'retaguardia' vuelve a entregar el control al BASIC una vez que ha terminado su tarea - su propósito es permitir el acceso a tales dispositivos extra como 'lápices luminosos', unidades de disco, impresoras inteligentes, y demás. Un comando de 'vanguardia' no devuelve el control al BASIC, sino que se 'apodera' completamente de la máquina - puede ser un juego autocontenido u otro interpretador de lenguaje (tal como Forth).

Nota.- Los términos ingleses son 'Background' y 'Foreground' para designar lo aquí llamado de retaguardia y vanguardia, respectivamente.

3. Modos Directo y Programa

El BASIC trabaja en uno de los dos modos principales: Directo y Programa. En Modo Directo, el BASIC toma las líneas tecleadas en la consola y ejecuta los comandos cuando se pulsa 'retorno de carro' que termina el comando. En Modo Programa, el BASIC toma las líneas del programa actualmente depositado en memoria y ejecuta los comandos que hay en cada una de las líneas. Con muy pocas excepciones todos los comandos pueden usarse en uno u otro modo.

3.1 Modo Directo

El Modo Directo puede subdividirse en:

— Comandos Directos
Los comandos se introducen e inmediatamente son ejecutados.

— Introducción de Programas
Se introducen nuevas o sustitutivas líneas de programa o se quitan algunas existentes.

— Enmienda de un Programa
Se alteran o modifican líneas de programa existentes.

El modo Comando Directo es inmediato. Es posible usar variables en el modo Comandos Directos. Ciertos comandos hacen que todas las variables se anulen, y entre ellos se incluyen los comandos RUN, NEW y LOAD.

La Introducción de Programas puede hacerse simplemente tecleando una línea que comience con un número dado. Cuando se termina la línea (por **[ENTER]**) queda incluida en el programa actual, sustituyendo cualquier línea que en él exista con el mismo número. (Observa que una línea que solamente consta del número de línea, suprime la línea existente que hubiera con ese número). El comando AUTO hace que el BASIC ayude a la introducción de programas generando los números de línea automáticamente.

La Enmienda del Programa está apoyado por el comando EDIT; véase sección 6.

Observa que el BASIC no efectúa ninguna comprobación de las líneas de programa cuando se están introduciendo o enmendando.

3.2 Modo Programa

En el Modo Programa, el BASIC automáticamente recorre paso a paso las líneas del programa presente en memoria, obedeciendo los comandos que allí encuentra. Las líneas se toman según el orden de números de línea, excepto cuando de manera explícita, los comandos obligan a hacer otra cosa. Diversos comandos hacen que el BASIC deje el Modo Programa y vuelva al Modo Directo, y se llaman a veces comandos 'conclusivos'. El comando END señala el punto esperado de **finalización** de un programa. Siempre se considera que hay un comando END implícito, después de la última línea del programa.

Las variables persisten con sus valores cuando el BASIC vuelve del Modo Programa al Modo Directo, lo que es muy utilizado al **depurar** un programa.

El modo Tratamiento de Error es un sub-modo del Modo Programa, y el BASIC entra en dicho sub-modo cuando ocurre un error y además se ha dado un comando ON ERROR GOTO nn.

Pulsando **ESC** en el teclado, mientras se está ejecutando un programa, hará que el BASIC **escape** de la obligación de ejecutar el programa actual, lo pare por tanto, y espere a la siguiente pulsación en el teclado. La acción que efectuará a continuación depende de la segunda tecla que se pulse:

ESC	El BASIC volverá al Modo Directo con un mensaje de interrupción (Break). Siempre que no se altere el programa de ninguna manera, puede hacerse que siga mediante el comando CONT, como si no hubiera sucedido ninguna interrupción en la ejecución.
SPACE	El BASIC reanuda la ejecución del programa actual. El propio carácter espacio se descarta.

Cualquier otra tecla hace que el BASIC reanude la ejecución del programa actual, pero la tecla pulsada no se descarta, y puede procesarse posteriormente.

4. La Pantalla

La pantalla es similar a una **gradilla** o 'retícula' de malla cuadrada. Es decir, cada punto de imagen (pixel) puede especificarse individualmente. Cada **punto** es la proyección de un cierto número de bits en un área de la memoria principal. El número de bits por punto, y por tanto el número de elementos pictóricos distintos en la pantalla, depende del modo en que se gestione la pantalla.

La pantalla se presenta en dos formas: una pantalla de Textos y una pantalla de Gráficos. El programa para la gestión de la pantalla de texto divide ésta en un cierto número de 'células de un carácter' y se preocupa de mostrar caracteres y símbolos. El programa para la gestión de pantalla de gráficos admite que se especifique individualmente cada punto de imagen, e incluye facilidades para dibujar líneas y para mostrar puntos en determinados colores.

4.1 Modos, Resolución, Colores y Tintas

La pantalla opera en uno de los tres **modos** posibles. Cada modo es capaz de mostrar un número diferente de puntos de imagen de distinto tamaño, con un número diferente de colores usados simultáneamente para dichas motas -y cuanto más puntos de imagen haya, menor es el número de colores. La imagen en pantalla es proyección del contenido de una zona de memoria de 16K bytes, con 4, 2 ó 1 bit para determinar el color de cada punto, dependiendo del modo.

La pantalla es capaz de dibujar en 27 colores diferentes (incluyendo blanco y negro). El valor fijado para cada punto de imagen, se traduce en uno de los 27 colores mediante el circuito denominado 'paleta'. El valor asignado a cada punto, se denomina el número de **tinta.** El color asociado con cada Tinta puede cambiarse en cualquier momento, afectando instantáneamente a todos los puntos de imagen que tienen indicado dicho número de tinta. Además pueden asociarse dos colores a un número de Tinta, con lo que el color con que aparece en pantalla el punto, alterna entre los dos, dando un efecto de parpadeo.

Los caracteres aparecen en la pantalla ocupando una célula -o **cuadratín**- formada por ocho puntos horizontales y ocho puntos verticales.

Los modos de pantalla son los siguientes:

Modo 0: 25 líneas de 20 caracteres cada una, o bien 160 puntos de anchura por 200 puntos de altura. 16 tintas posibles (4 bits por punto).

Modo 1: 25 líneas de 40 caracteres cada una, o bien 320 puntos de anchura por 200 puntos de altura. 4 tintas posibles (2 bits por punto).

Modo 2: 25 líneas de 80 caracteres cada una, o bien 640 puntos de anchura por 200 puntos de altura, 2 tintas posibles (1 bit por punto).

Hay un área que rebordea la porción usable de la pantalla. Dicho borde tiene su especificación separada de Tinta en la Paleta, de manera que puede especificarse para ella un color diferente (o una pareja alternante de colores) al del resto de la pantalla.

Los colores están numerados de 0 a 26, siendo el color 0 el negro y el color 26 el blanco. Véase el Apéndice VI para la lista de los otros colores. La numeración está dispuesta de manera que los colores están ordenados según la escala de grises, lo que presenta interés para los que usan monitores monocolor.

4.2 La Pantalla de Texto y los Caraceres Definibles por el Usuario

Cuando se usa la pantalla para exponer texto se divide en 'células de caracteres', o **cuadratines,** formado por 8 puntos a lo ancho y 8 puntos a lo alto. Eso permite 25 líneas de caracteres en todos los modos de pantalla, pero un número diferente de caracteres por cada línea, como hemos comentado. Las **líneas** (la posición 'y') se numeran a partir de 1 que es la parte superior de la pantalla, hasta la línea más inferior que es la línea 25. Los caracteres se numeran a lo largo de cada línea (la posición 'x') a partir de 1 en el extremo izquierdo de la pantalla, de manera que el situado en el extremo derecho de la pantalla es el carácter 20, 40 ú 80, dependiendo del modo.

El repertorio de caracteres comprende 256 caracteres diferentes. Los caracteres con valores asociados **-códigos-** entre 32 y 126 son los caracteres habituales ASCII. Los otros caracteres corresponden a diversos símbolos y 'cuadratines gráficos'. El aspecto visual de cada carácter queda definido por una **matriz** de 8 octetos, en que un bit corresponde a cada uno de los ocho por ocho puntos de imagen en la 'célula' del carácter. Los últimos 16 caracteres son implícitamente **definibles por el usuario,** es decir, su 'matriz de forma' puede cambiarse a discreción -aunque los que ya estuvieran expuestos en pantalla no se ven afectados. El número de caracteres definibles por el usuario puede cambiarse usando el comando SYMBOL AFTER.

En general se usan **dos tintas** cuando se va a escribir en pantalla un carácter. Si el bit en la matriz de forma está puesto a uno, el punto de imagen correspondiente en la célula de carácter aparece en el color designado por la Tinta de la Pluma. Cuando el bit en la matriz de forma del carácter está puesto a cero, el punto de imagen correspondiente queda afectado de la forma siguiente:

en modo Normal o Escritura: el punto aparece en el color correspondiente al Tinte del Papel, de manera que todo el fondo del carácter queda relleno en ese color.

en el modo Transparente: el punto permanece en el color que tuviera, de manera que el color con que aparece corresponde al fondo previo y se muestra a través -de ahí el nombre 'transparente'.

4.3 Cauces hacia Pantalla y Ventanas de Texto

Los primeros ocho cauces de salida de información, identificados como cauces 0..7, envían caracteres hacia la pantalla de texto. Cada cauce posee un conjunto separado de atributos:

- Ventana de proyección
- Tinte del Papel
- Tinta de la Pluma
- Posición del Cursor
- Opción de Transparencia
- Opción de Texto en Cursor de Gráficos

El área de la pantalla -el **recuadro**- por donde se proyecta información a través de un cauce dado, se conoce como Ventana de dicho cauce. El comando WINDOW puede usarse para recuadrar una cierta fracción de la pantalla, para servir de 'lienzo' donde proyectar información. Este recuadramiento puede usarse para separar cauces independientes para proyección de texto, o para separar textos y gráficos.

Excepto cuando sus ventanas de proyección se solapan, los cauces de envío hacia pantalla pueden considerarse como pantallitas de texto enteramente separadas. Cuando las ventanas de proyección de dos cauces se solapan, el contenido proyectado en el área de solape refleja la última actividad a través del cauce más recientemente usado.

El cursor de texto puede desplazarse fuera de la ventana corriente, o vigente, pero antes de que pueda exponerse un carácter o el símbolo del cursor, y antes de que el BASIC obedezca diversos códigos de control, se obliga automáticamente al cursor para que ocupe una posición legal en la ventana, lo que puede provocar que la ventana se desplace hacia arriba o hacia abajo. Véase el Apéndice IV para más detalles.

Los caracteres enviados hacia la pantalla de texto se interpretan como sigue:

0..31 Caracteres de Control. Tienen un significado especial para la pantalla de texto, tal como se describe en el Apéndice IV. Los caracteres de Control pueden emplearse para desplazar el cursor, limpiar la 'ventana', fijar las tintas, y otras gestiones.

	Para exponer en pantalla el símbolo asociado a un carácter, con código en la banda 0..31, el carácter debe estar precedido por CHR$(1).
32..127 Caracteres ASCII.	La forma del carácter que aparece en pantalla puede alterarse usando el mecanismo de símbolos definibles por el usuario.
128..255 Símbolos surtidos	Los últimos 16 símbolos son implícitamente definibles por el usuario, aunque esta cantidad de símbolos puede cambiarse (subirla o bajarla) mediante el comando SYMBOL AFTER.

4.4 Pantalla de Gráficos, Sistema de Coordenadas, Ventana y Origen

Sólo hay una pantalla de gráficos. La pantalla de gráficos tiene su propio sistema de coordenadas y su ventana de proyección.

El sistema de coordenadas para la pantalla de gráficos divide ésta en una gradilla de malla cuadrada con 640 **puntos** de anchura y 400 **puntos** de altura. Al proyectar estas coordenadas sobre la pantalla, producen puntos de imagen de color de un determinado tamaño, y de manera que los gráficos resultantes son independientes del modo en que se gestione la pantalla, excepto por la variación que tienen en resolución.

Se puede considerar que la proyección según el eje horizontal -a lo ancho de la pantalla- depende del modo de pantalla, en la forma siguiente:

Modo 0: 1 punto en pantalla ocupa 4 **puntos** (baja resolución)
Modo 1: 1 punto en pantalla ocupa 2 **puntos**
Modo 2: 1 punto en pantalla ocupa 1 **punto** (alta resolución)

La proyección en el sentido vertical -a lo alto- no se ve afectada por el modo de pantalla; en todos los modos un punto en pantalla ocupa 2 puntos. (Esta proyección vertical hace que un punto verticalmente corresponda a groso modo con un punto horizontalmente, lo que da una 'relación de aspecto' de uno a uno).

Las coordenadas usadas en la pantalla de gráficos crecen hacia arriba y hacia la derecha, y parten del valor 0. Por lo tanto, la esquina inferior izquierda de la pantalla tiene como posición en coordenadas absolutas la (0,0); y la esquina superior izquierda de la pantalla tiene la posición (639,399) como coordenadas absolutas. Las coordenadas gráficas pueden tener cualquier valor Entero, incluyendo valores negativos.

El origen de todo el sistema de coordenadas puede desplazarse de su posición base, en la esquina inferior izquierda de la pantalla, mediante el comando ORIGIN.

Cuando se proyecta una posición gráfica para conseguir el punto de color en pantalla, las coordenadas se 'truncan' hacia el origen. Un cambio en el origen de coordenadas es por tanto casi equivalente -pero no exacto- a añadir un 'delta', o desplazamiento compensador, a las coordenadas.

La ventana de gráficos puede ocupar la pantalla completa, o puede estar restringida a una fracción de ella. La ventana de gráficos se fija mediante parámetros opcionales del comando ORIGIN. El cursor de gráficos -el **pincel**- puede desplazarse fuera de la pantalla, pero cualquier intento de dibujar fuera de la pantalla será descartado e ignorado.

Se puede combinar gráficos y textos en una cierta medida, mediante la posibilidad de Texto en la posición del Cursor de gráficos. Un cauce de exposición de textos puede dirigirse para que muestre caracteres en la posición señalada por el Cursor de Gráficos. Esta facilidad permite que se coloquen los caracteres en cualquier parte de la pantalla, en lugar de tener que hacerlo únicamente en las posiciones correspondientes a células de caracteres o cuadratines. Los caracteres escritos según el cursor de gráficos no se ven afectados por la opción transparente (véase anteriormente), pero sí son afectados por el Modo de Tinta para Gráficos (véase ulteriormente).

El color con que aparecen los puntos de imagen trazadas por los comandos PLOT, DRAW o TAG (Texto en Cursor de Gráficos), depende del Modo de Tinta para Gráficos, que puede ser uno de los siguientes:

- Normal El punto tiene el color de la tinta nueva
- XOR El punto toma el color O-lógico exclusivo entre las tintas vieja y nueva.
- AND El punto toma el color Y-lógico entre las tintas vieja y nueva.
- OR El punto toma el color O-lógico entre las tintas vieja y nueva.

4.5 Resumen de Comandos y Funciones de Pantalla

BORDER	-fija la tinta del 'borde'
CLG	-limpia la pantalla de gráficos
CLS	-limpia la ventana de texto del cauce dado
DRAW	-traza recta en pantalla de gráficos -hasta una posición absoluta
DRAWR	-traza recta en pantalla de gráficos -hasta una posición relativa
INK	-fija una tinta dada
LOCATE	-sitúa el cursor de texto de un cauce dado
MODE	-cambia a un nuevo modo de pantalla
MOVE	-mueve el cursor de gráficos -hasta una posición absoluta
MOVER	-mueve el cursor de gráficos -hasta una posición relativa
ORIGIN	-fija el origen de coordenadas para gráficos y la ventana

PAPER	-fija el Tinte del Papel para el cauce dado
PEN	-fija la Tinta de la Pluma para el cauce dado
PLOT	-pinta un punto en pantalla de gráficos -en posición absoluta
PLOTR	-pinta un punto en pantalla de gráficos -en posición relativa
POS	-posición horizontal del cursor de textos para el cauce dado
SPEED INK	-fija la rapidez de parpadeo de Tintas alternantes
SYMBOL	-fija la matriz de Forma de los caracteres definidos por el usuario
SYMBOL AFTER	-fija como caracteres definibles por el usuario los de 'después' del dado
TAG	-fija el modo Texto en Cursor de Gráficos para el cauce dado
TAGOFF	-quita el modo Texto en Cursor de Gráficos para el cauce dado
TEST	-número de tinta usada para un punto en gráficos -coordenadas absolutas
TESTR	-número de tinta usada para un punto en gráficos -coordenadas relativas
VPOS	-posición vertical del cursor de textos para el cauce dado
WINDOW	-fija la ventana para proyección de texto a través del cauce dado
WINDOW SWAP	-intercambia, 'canjea' las ventanas correspondientes a los dos cauces dados
XPOS	-posición horizontal (coordenada X) del cursor de gráficos
YPOS	-posición vertical (coordenada Y) del cursor de gráficos.

5. Teclado y Joysticks

El teclado tiene tres grupos de teclas: el teclado principal completo con diversas teclas de 'turno' (SHIFT); el pequeño teclado numérico separado; y las teclas de control del cursor. Se admiten dos joysticks de juego, que se tratan como parte del teclado -la palanca 0 como una parte completamente separada, y la palanca 1 solapándose con algunas teclas del teclado principal. Hasta 32 teclas pueden designarse como Teclas Funcionales, cada una de las cuales puede programarse para producir toda una constante literal, o serie de caracteres, con sólo una pulsación.

Cada tecla tiene tres **valores** asociados con ella, cada uno dentro de la banda 0..255. El valor generado cuando se pulsa la tecla depende del estado de 'turno' (SHIFT). Los tres estados de 'turno' se denominan: Normal, Shift ('turno') y Control, y se producen mediante:

Normal — cuando ninguno de los otros estados de 'turno' está activo.

Shift — cuando se mantiene pulsada una de las teclas **Shift;** o cuando está 'enclavado el turno' y no se ha pulsado **[CTRL].**

Para lograr el 'turno enclavado' han de pulsarse simultáneamente las teclas **[CTRL]** y **[SHIFT],** y se 'desenclava' pulsando de nuevo esa misma combinación.

Control — cuando además se mantiene pulsada la tecla **[CTRL],** e independientemente de cualquier otro estado de 'turno' que pudiera haber.

La tecla **[CAPS LOCK]** bascula -pone y quita- el estado de 'enclavamiento de mayúsculas', en el que cualquier valor de tecla dentro de la gama 'a'..'z' se hace corresponder al valor homólogo en la gama 'A'..'Z'. (De modo que el estado 'enclavamiento de mayúsculas' no es equivalente a pulsar la tecla de 'turno' y las teclas alfabéticas; lo que constituye una facilidad interesante al cambiar el teclado).

La tecla **[ESC]** se usa para 'escapar' de la ejecución de un programa. Cuando el BASIC no está ejecutando un programa, pero está activo de cualquier otra manera, por ejemplo, durante el EDITado de un programa -pulsando **[ESC]** se abandona la operación y se retorna al Modo Directo. Cuando el BASIC está ejecutando un programa hay una acción en dos fases: cuando se pulsa **[ESC],** el BASIC suspende la ejecución del programa y espera la siguiente pulsación de tecla, cuyo efecto es:

[ESC] — detiene definitivamente el programa y el BASIC regresa al Modo Directo.

[SPACE] — continúa ejecutando el programa. El carácter 'espacio en blanco' queda descartado.

otra — continúa ejecutando el programa. El carácter pulsado queda retenido para su proceso posterior.

La tecla [**ESC**] en combinación con las teclas [**SHIFT**] y [CTRL], simultáneamente, **restauran** las condiciones iniciales de la máquina, dejándola en el mismo estado en que estaba cuando se encendió al principio (siempre que [**ESC**], [**SHIFT**] y [CTRL] sean las únicas teclas que se pulsan).

Puede hacerse que las teclas autorrepitan la pulsación cuando se mantienen pulsadas un cierto tiempo. Cuando se pulsa por primera vez, genera el valor asociado. Si la tecla ha sido prefijada para que se autorrepita después de haberla mantenido pulsada durante un período dado (llamado la 'demora al arranque'), se vuelve a generar de nuevo el valor de dicha tecla. Si la tecla permanece pulsada, la autorrepetición se hará a un ritmo determinado por el período de repetición, que generalmente es más pequeño que la demora al arranque. Los valores iniciales son de 0,6 segundos para la demora al arranque y de 0,04 segundos para el período de repetición (i.e. 25 pulsaciones por segundo).

El teclado se escruta, o explora 50 veces por segundo. En cada exploración se comparan el estado presente y el previo para mantener el seguimiento de los cambios habidos en la tecla. Se mantiene además un 'buffer' con las teclas que hayan sido pulsadas.

Los valores de autorrepetición y los tres valores -**códigos de entrada**- asociados a cada tecla, pueden cambiarse en cualquier momento. El teclado puede por tanto adaptarse a cualquier aplicación concreta. (No es sin embargo efectivo intentar cambiar las teclas [**SHIFT**] ni [**ESC**]). El **valor** generado por la pulsación de una tecla puede ser:

0..31	Estos valores intentan tener efectos especiales cuando se envían a pantalla, pero se tratan como caracteres ordinarios cuando proceden del teclado, excepto para &ØD(13) que es el 'retorno de carro'.
32..127	Caracteres ordinarios, habitualmente interpretados como ASCII.
128..159	Caracteres especiales que pueden ser 'ampliados' de acuerdo con la serie de caracteres fijados mediante el comando KEY. (Véase posteriormente).
160..223	Caracteres ordinarios.
224..254	Valores especiales usados para editado y copiado con teclas de cursor.
255	Ignorado.

Los treinta y dos caracteres especiales (llamados 'dichos', y comandos mono-tecla) en la banda 128..159, pueden usarse para implementar Teclas Funcionales -teclas que generan una serie de caracteres con sólo una pulsación.

El programa gestor del teclado mantiene un literal (string) asociado con cada una de estas teclas de carácter especial. Cuando uno de esos valores de entrada es tomado del 'buffer' del teclado, queda sustituido automáticamente por el literal que tenga asociado. Los literales pueden cambiarse en cualquier momento. El estado inicial del teclado tiene el teclado numérico separado fijado a los valores de entrada 128..139, y los literales asociados de acuerdo con las leyendas, o carátula de las teclas. La pulsación de [**CTRL**] y [**ENTER**] está prefijada al valor de entrada 140 y tiene asociado el comando RUN" <retorno de carro>. Los otros diecinueve valores especiales de comandos mono-tecla no están usados en el estado inicial del teclado.

Resumen de Comandos y Funciones del Teclado:

INKEY	-Estado actual de la tecla dada por su número.
INKEY$	-Tecla pendiente de tratar, si hay, en el 'buffer' del teclado
JOY	-Estado del mando de juegos dado por su número.
KEY	-Asocia un nuevo 'literal' a la tecla dada por su número.
KEY DEF	-Redefine el valor generado por la pulsación de la tecla dada.
SPEED KEY	-Fija la demora al arranque y el período de repetición de la tecla dada.

6. Introducción de Líneas, Edición y Copiado

La máquina incorpora un programa **editor de línea** y la habitilidad para copiar texto mostrado en la ventana vigente incluyéndolo en la línea **en curso.** Estas facultados están disponibles siempre que el BASIC está operando le sea introducida una línea desde el teclado.

Las facilidades para **edición de línea** permiten que el cursor se desplace dentro de la línea vigente para suprimir o insertar caracteres en la posición señalada por el cursor. Cuando se concluye la línea pulsando **[ENTER],** es introducida por el BASIC incluyendo cualquier texto a la derecha del cursor (si lo has retrocedido con respecto al extremo final de la línea). Pulsando **[ESC]** durante la introducción o la edición de líneas, se abandona todo lo efectuado sobre dicha línea. El comando EDIT toma la línea especificada, la considera como la **línea en curso** sobre la que operar, mostrándola en pantalla con el cursor situado en el primer carácter. La línea puede entonces **editarse.** Cuando se concluye la edición, pulsando **[ENTER],** el BASIC la vuelve a poner dentro del programa. Observa que cambiando el número de línea (o sustituyes la que hubiera ya así numerada).

Las facilidades para **copiado de líneas** funcionan mediante un cursor copiante, que se separa del cursor de entrada, y puede moverse por toda la pantalla. Cada vez que se pulsa la tecla **[COPY],** el carácter señalado por este nuevo cursor es copiado e introducido en la posición señalada por el cursor de entrada; y ambos cursores avanzan una posición.

Las teclas de cursor tienen tres significados, dependiendo del **'turno'** en que se encuentren; como sigue:

'Normal' mueve el cursor de **entrada** dentro de la línea en curso. Observa que no está permitido sacarlo de dicha línea por ningún de los extremos.
Sólo mientras no existe línea en curso, mientras está vacía, se puede con estas teclas mover el cursor por toda la pantalla.

con SHIFT 'Control' mueve el cursor de **copiado** por toda la pantalla
[CTRL] y **arriba** -lo mueve hasta el comienzo de la línea logical actual.
[CTRL] y **abajo** -lo mueve hasta el término de la línea logical actual.
[CTRL] e **izquierda** -lo mueve hasta el comienzo de la línea fisical (el 'renglón' en pantalla).
[CTRL] y **derecha** -lo mueve hasta el término de la línea fisical (el 'renglón' en pantalla).

Las otras teclas de importancia especial en la **edición de líneas,** son:

[DEL]	Suprime el carácter situado a la izquierda del cursor de **entrada.** Mueve el cursor y el resto de la línea una posición a la izquierda.
[CLR]	Suprime el carácter situado en la posición del cursor de **entrada.** Mueve el cursor y el resto de la línea una posición a la izquierda.
[ESC]	Abandona la línea en curso (i.e. como si no se hubiera hecho nada con ella).
[CTRL]-[TAB]	Bascula el modo Inserción/Sustitución (i.e. cambia alternativamente de modo).
[ENTER]	Concluye la edición de una línea y la pone adento para ser tratada por el BASIC.

El modo de Inserción se establece al comienzo de la introducción o el editado de una línea de programa. Se puede insertar nuevo texto en la línea actual simplemente tecleándolo; el resto de dicha línea se desplaza una posición a la derecha para hacer sitio a los caracteres que se insertan. La pulsación simultánea de [CTRL] y [TAB] bascula -pone y quita- el modo entre Inserción y Susstitución. En Sustitución o 'escritura encima', el nuevo texto reemplaza los caracteres existentes en la línea en curso.

7. Cauces de Entrada y Salida

Toda la entrada y salida de datos hacia el BASIC, distinto de los gráficos, usa **cauces** (en inglés 'streams'). Hay diez cauces de salida y diez cauces de entrada. Los cauces de salida son:

0..7	Cauces hacia pantalla de texto
8	Cauce hacia impresora
9	Cauce hacia un fichero de salida en cassette

Los cauces de entrada son:

0..7	Del teclado, con avisos enviados hacia el cauce correspondiente de pantalla.
8	Del teclado, con avisos enviados hacia el cauce de impresora.
9	De un fichero de entrada en cassette.

7.1 Cauces de salida

Los comandos para salida de textos, tales como PRINT, incluyen un parámetro opcional para designar el cauce de envío (si se omite dicho parámetro se adopta el cauce #0). Así se consigue que los comandos sean coherentes para todos los diversos equipos y dispositivos. Dado que el número de cauces dado en el parámetro puede ser una expresión, es posible escribir un programa que saque al final la información hacia la impresora o el cassette, pero que las pruebas se efectúen utilizando como salida de datos la pantalla.

Los cauces hacia pantalla tienen sus propios atributos de Ventana, Pluma, Papel, etc. Los comandos apropiados para fijar estos atributos, no tienen obviamente equivalentes al enviarse hacia Impresora o Cassette.

Antes de escribir a través del cauce de salida hacia cassette, debe darse un comando OPENOUT para abrir y establecer el nombre del nuevo fichero. Cuando ya se han escrito todos los datos requeridos, debe darse el comando CLOSEOUT para 'despachar' cualquier dato que hubiera en el 'buffer' de salida hacia el fichero, y marcar en la cinta la terminación del fichero.

7.2 Cauces de entrada

Sólo hay dos fuentes concretas para introducción de datos hacia el BASIC: el teclado y la cassette. Cuando el BASIC ejecuta un comando INPUT, envía un **aviso** -en forma de signo de interrogación- para indicar que está esperando recibir datos que 'impondrá' como valores de las variables. Los cauces de teclado, todos recogen los datos de entrada procedentes del teclado, pero el aviso se dirige hacia el correspondiente cauce de salida, pantalla o impresora. En el caso de los procedentes del cassette, los avisos que pudiera haber son desechados.

Antes de la lectura de datos procedentes del cassette, debe darse un comando OPENIN para 'abrir' y establecer el nombre del fichero de entrada. Cuando ya se ha hecho la lectura de todos los datos requeridos, se debe dar el comando CLOSEIN para comunicar al BASIC el hecho producido. La función EOF puede usarse para probar si se ha alcanzado la condición de 'fin de fichero' en la entrada procedente del cassette.

7.3 Ficheros en cassette

Los cauces de entrada y salida hacia el cassette se mencionan como 'cauces de fichero'. Es posible hacer la lectura de datos de un cassette y escribir en otro 'simultáneamente'. Dado que sólo hay una lecto-grabadora de cassette en el sistema, eso requiere que las cintas en cassette sean cambiadas en los momentos oportunos.

El BASIC hace la lectura y escritura de ficheros, subdividiéndolos en bloques de 2K bytes. Cuando se abre un fichero como entrada (usando OPENIN) se busca en la cinta el fichero dado, y se hace la lectura del primer bloque traspasándolo a la memoria, y preparándose para su tratamiento. Cuando se abre un fichero de salida (usando OPENOUT) no se efectúa ninguna escritura en la cinta hasta que se haya preparado en el 'buffer', o zona reservada de la memoria, el byte número 2K + 1, y en ese momento el BASIC se ve forzado a vaciar dicho 'buffer' y transferir los datos a la cinta.

Los programas que efectúen lectura y escritura de ficheros en cassette deben cerrarlos siempre (usando CLOSEIN y CLOSEOUT) cuando han acabado de tratarlos. La instrucción END, o la finalización del programa, implícitamente cierran cualquier fichero que hubiera en cassette.

Cuando se cierra un fichero de salida, cualquier información que hubiera en el 'buffer' pertinente es transcrita a la cinta, y luego se marca el final de fichero.

Los ficheros en cassette quedan 'abandonados', sin que se garantice un cierre correcto, mediante los comandos NEW, SAVE, MERGE, LOAD, RUN, CHAIN, CHAIN MERGE y CLEAR. Observa que abandonar un fichero de salida no hace que cualquier información que hubiera en el 'buffer' pertinente, sea transcrita a la cinta antes de cerrarlo; y por lo tanto, si la hubiera se perdería.

8. Generación de Sonido

Los circuitos de generación de sonidos disponen de tres canales independientes (A, B y C). Cada canal es capaz de producir ondas cuadradas en la banda aproximada de 30..120 Kilociclos y simultánea o separadamente ruido pseudoaleatorio. El volumen de cada canal puede fijarse a uno de dieciséis niveles (incluyendo el 'silencio'). Los comandos disponibles permiten producir simples sonidos de volumen y tono constante, y ruido en las mismas condiciones; y además permiten generar sonidos más complicados en que el volumen y/o el tono varía mientras se produce el sonido, bajo el control de las correspondientes **envolventes.** También existen posibilidades para garantizar que los sonidos emitidos por canales separados se convierten en 'acordes', o se emiten simultáneamente, cuando se precisa.

El circuito de generación de sonido trabaja dándole valores no de frecuencias, sino de períodos. Por tanto, para fijar un **tono** se expresa como 'período del tono', siendo un período P el que produce un tono de frecuencia F, tal que:

F = 125000/P

(e inversamente el período P que se requiere para una frecuencia F es P = 125000/F). La envolvente de tono permite que el período varíe durante la generación de una nota. Véase Apéndice VII para la lista de los períodos de tono recomendados de acuerdo con la escala habitual armónica.

El generador de ruido produce un tono cuya frecuencia varía aleatoriamente alrededor de un valor dado. La especificación del ruido también se expresa como un período, con la frecuencia y período manteniendo la misma relación que para la frecuencia y período de la nota musical.

Aunque cada canal puede hacerse que emita tono y/o ruido, los circuitos sólo tienen una especificación posible para el período de ruido, por lo que no es factible tener ruido de diferentes frecuencias en los diferentes canales al mismo tiempo.

La duración de una **nota** puede fijarse explícitamente, o puede estar gobernada por la longitud de una 'envolvente de volumen'. Si se fija explícitamente, el sonido cesa después de un tiempo determinado, sin tener en cuenta cualquier envolvente de volumen o tono que haya.

Si la duración está fijada según la longitud de la envolvente de volumen, entonces la emisión de la nota cesa cuando se extingue la envolvente de volumen, sin tener en cuenta la envolvente de tono. Es posible al determinar una nota, hacer que se repita la envolvente de volumen un número dado de veces -el sonido durará hasta el final de la última repetición.

También puede usarse una envolvente de volumen para variar el volumen de una nota. Cada envolvente puede tener hasta cinco **secciones**, siendo cada sección un cambio absoluto o relativo en el volumen, o excepcionalmente, una fijación de envolvente estándar en el circuito:

Una sección incremental se especifica como un número de pasos, con un cierto tamaño de paso (que puede ser negativo) y una pausa después de cada paso (medida en centésimas de segundo).

Una sección absoluta se especifica como un nuevo nivel de volumen y una pausa antes de la siguiente sección (medida en centésimas de segundo).

La fijación de envolvente por el propio circuito puede usarse para aprovechar la ventaja de las características inherentes para envolventes que posee la 'pastilla', cuando sea aconsejable.

Puede usarse una 'envolvente de tono' para variar el período de tono de una nota musical. Cada envolvente puede tener hasta cinco secciones, siendo cada sección un cambio incremental o absoluto al período de tono:

Una sección incremental se especifica como un cierto número de pasos, con un tamaño para cada paso (que puede ser negativo) y una pausa después de cada paso (medida en centésimas de segundo).

Una sección absoluta se especifica como un nuevo período de tono y una pausa antes de la siguiente sección (medida en centésimas de segundo).

Puede hacerse también que se repita la envolvente de tono.

Las posibilidades de fijar envolventes pueden emplearse para producir notas musicales que semejen los diversos instrumentos, particularmente usando un canal para el tono básico y los otros dos para añadir los armónicos:

Cada canal puede tener una nota emitiéndose -activa- y hasta cuatro notas en secuencia, esperando ser procesadas. Cuando termina la nota activa, el sistema intenta ejecutar el siguiente sonido que haya en la secuencia de notas del canal en cuestión. La nota comenzará a emitirse a no ser que esté Retenida, o esperando un acorde:

Cuando una nota está Retenida, permanece esperando a ser ejecutada, y el canal no produce ningún sonido, hasta que dicha nota sea Liberada.

El mecanismo de Retención/Liberación permite que las notas de la secuencia sean 'marcadas', estipulando que la primera nota de cada canal está Retenida y añadiendo más notas a las secuencias, y luego Liberando los canales.

El mecanismo de Acordes o 'Rendezvous', puede usarse para forzar que las notas emitidas por dos o por todos los canales, comiencen conjuntamente.

Cuando se coloca en la secuencia a emitir por un canal una nota dada, puede establecerse que debe ser Acorde con otra nota en uno o en ambos de los otros canales. Cuando comienza a ejecutarse la nota, será retardada hasta que los otros canales tengan notas preparadas para ser ejecutadas, y que tengan estipulado el recíproco Acorde.

Puede emitirse una nota con la marca de Oclusión (o 'tajante') puesta. Eso hace que se termine inmediatamente cualquier sonido activo y que se desechen los que hubiera en la secuencia de espera, dejándola vacía y el canal inactivo, u 'ocluido' y dispuesto para un nuevo sonido. Por ejemplo, el sonido 'pitido' producido por el carácter de código 07, efectúa siempre una Oclusión del canal antes de que sea emitido.

El estado de la secuencia de notas, contenidas en un canal dado, puede ser comprobado. El sistema informará de cuántas notas es posible enviar hacia ese canal, si una nota es activa o está sonando, si el canal está Retenido o si está esperando un Acorde.

Para facilitar la generación de sonidos 'de fondo' puede hacerse que cada canal invoque una subrutina determinada del programa en BASIC cuando no tiene completa la secuencia de notas a emitir. Esta subrutina puede hacer que envíe un nuevo sonido por ese canal. Véase la Sección 9 para más comentarios sobre las subrutinas suscitadas por 'interrupciones'.

La generación de sonido continúa después de la instrucción END o del fin del programa, hasta que deja vacía toda la secuencia de sonidos pendientes, pero las posibles interrupciones para añadir sonidos quedan impedidas.

La suspensión o detención temporal de un programa mientras hay notas activas, provoca que se suspendan también la emisión de sonidos. Cuando se reanuda la ejecución del programa, los sonidos también son reanudados a partir de donde se quedaron. Si se emite un sonido mientras se está en el Modo Directo, también se reanudará cualquier sonido que estuviera en espera de ser emitido.

El uso del cassette termina todos los sonidos corrientes y elimina todos los que hubiera en la cola.

Resumen de comandos de sonido y funciones:

END	-Fija la envolvente de tono.
ENV	-Fija la envolvente de volumen.
ON SQ	-Establece la subrutina para interrupciones en la secuencia de notas.
RELEASE	-Libera la emisión de notas retenidas por el canal dado.
SOUND	-Emite una nota hasta la secuencia del canal dado.
SQ	-El estado de la secuencia de notas en el canal dado en la función.

9. Interrupciones en BASIC

Los sucesos externos al BASIC pueden usarse para **suscitar** la ejecución de una subrutina del programa. Después de ejecutar dada instrucción, el BASIC examina si cualquiera de los sucesos designados ha ocurrido. Si detecta que se ha producido, la subrutina asociada con dicho suceso, es 'citada' o puesta en marcha, de igual manera que si se hubiera incluido una instrucción GOSUB inmediatamente detrás de la instrucción que acaba de ejecutar. Los sucesos que pueden causar estas Interrupciones y desvíos en el programa son los siguientes:

- Un grupo de cuatro temporizadores o cronómetros.
- Los tres canales de sonido, con sus secuencias incompletas.
- Una ruptura en la ejecución (**ESC/ESC**).

Si hay más de un suceso pendiente de ser atendido, el que tenga la mayor prioridad determina la subrutina a ejecutar. Mientras se está ejecutando la subrutina señalada para un suceso, no puede ser interrumpida por otro suceso, excepto cuando éste último posee mayor prioridad. Las prioridades en orden descendente, son las siguientes:

Ruptura o 'corte' en la ejecución (**ESC/ESC**)
Temporizador 3
Temporizador 2 y los tres canales de sonido (todos de igual prioridad)
Temporizador 1
Temporizador 0

Debe ejercerse con cuidado la utilización de Interrupciones. Dado que una interrupción puede suceder en cualquier momento entre las ejecuciones de las instrucciones, el programador debe garantizar cuidadosamente que la subrutina de interrupción no ve afectada la 'tarea' que realiza por quedar las variables en estados intermedios. Al contrario, la subrutina para interrupciones debe redactarse de manera que evite interacciones no deseadas con el resto del programa, provocando por ejemplo, el uso de una variable por segunda vez, o desplazar el cursor de pantalla a otras posiciones.

El mecanismo de prioridades permite un método para 'exclusiones mutuas'. También es posible **impedir** explícitamente todas las posibles interrupciones (excepto la de corte o ruptura), y **permitirlas** de nuevo, mediante las instrucciones DI y EI. Cuando las subrutinas para interrupciones interactúan unas con otras, o con el programa principal, pueden 'desactivarse' las señales de interrupción para asegurar que cualquier serie de instrucciones se ejecuta ininterrumpidamente en una sola vez.

9.1 Los Temporizadores

Los cuatro 'medidores de lapsos de tiempo' son independientes y tienen prioridades diferentes. Los 'cronómetros' cuentan hacia atrás una cierta cantidad de 50-avos (1/50) de segundo, y luego provocan la interrupción. Es posible especificar que el temporizador interrumpa sólo una vez, o que vuelva a reiniciar la cuenta después de cada interrupción, dando así una repetición regular de interrupciones, y consiguiendo por tanto la ejecución de una cierta tarea cada cierto tiempo.

Dado que las posibles interrupciones no se detectan excepto hasta el final de las instrucciones, la subrutina puede que no entre en acción durante un breve instante de retardo después de que la cuenta del temporizador ha concluido. La cuenta del temporizador se coloca al valor inicial inmediatamente después de que haya terminado con la cuenta previa, de manera que las señales de interrupción sí se generan regularmente, incluso cuando hay largas demoras antes de que pueda entrar en acción la subrutina suscitada por la interrupción. El sistema mantiene el registro de hasta 127 interrupciones, de manera que es totalmente improbable que ninguna se pierda.

Los temporizadores continúan contando cuando se suspende la ejecución del programa (por **[ESC]**), y cuando se detiene el programa por cualquier razón. Si la ejecución del programa se continúa, cualquier interrupción que haya pendiente, será procesada tan pronto como sea posible.

La función REMAIN permite saber la cuenta que le 'queda' a un temporizador dado, y permite desactivarlo. Cuando se especifica un temporizador para interrupciones 'sencillas', él mismo automáticamente queda desactivado al provocar la interrupción.

9.2 Las interrupciones debidas a los canales de sonido

Los tres canales de sonido provocan interrupciones independientes, pero todos tienen la misma prioridad. Una vez que se ha 'citado' -puesto en marca- una subrutina para tratar las interrupciones por sonidos, no se verá por tanto interrumpida por la suscitada en otro canal de sonido -de manera que las subrutinas para interrupciones de sonido pueden compartir las mismas variables.

Cuando está **permitida** la interrupción por un canal de sonido, provocará dicha interrupción cuando la secuencia de notas para ese canal no está completo; y cuando lo está provocará la interrupción inmediatamente que arranque la siguiente nota y haya por tanto, sitio para una nota más en la secuencia. La acción de suscitar la interrupción ya **impide** la producción de las siguientes, de manera que la subrutina debe re-permitir las interrupciones cuando en el programa se requiera que las sucesivas también sean atendidas. El intentar enviar una nota hacia un canal, o comprobar el estado del mismo, también 'desactiva' las posibles indicaciones de interrupción que hubiera 'activadas' para el canal o canales involucrados.

9.3 La interrupción por 'corte' ([ESC][ESC])

Esta interrupción, cuando está explícitamente **permitida,** se provoca al pulsar el par de teclas (**[ESC][ESC]**) que por otro lado detiene la ejecución del programa). Esta posibilidad puede usarse para evitar que se detenga por completo la ejecución de un programa mediante **[ESC][ESC],** o para efectuar alguna tarea de limpieza antes de detener la ejecución, o para implementar alguna característica especial.

9.4 Resumen de comandos y funciones para interupciones

AFTER	-Fija el intervalo de interrupción para el temporizador dado.
DI	-Desactiva (**impide**) interrupciones (excepto BREAK o 'corte').
EI	-Permite o 'faculta' interrupciones.
EVERY	-Fija temporizador para que interrumpa 'cada' cierto tiempo.
ON BREAK GOSUB	-Señala la subrutina para interrupciones por 'corte'.
ON BREAK STOP	-Impide que se susciten interrupciones por 'corte'.
ON SQ GOSUB	-Señala la subrutina para interrupciones por secuencia de sonidos.
REMAIN	-El tiempo que 'resta' o queda en el temporizador dado.

10. Panorama de Comandos

Esta sección no pretende describir en detalle los comandos, lo que sí se hace en la Sección 11. En ésta, los comandos se describen en grupos coherentes, con el fin de presentar algunas ideas sobre el propósito de los mismos, y hacerlo dentro del contexto en que intervienen.

10.1 Creación de programas

AUTO, DELETE, EDIT, LIST, MERGE, NEW, RENUM, SAVE

Los comandos de este grupo son utilizados cuando se está preparando un nuevo programa o se está enmendando el existente en memoria.

AUTO es una ayuda para teclear una nueva sección de un programa. Genera los números de líneas de programas a medida que se van introduciendo nuevas líneas.

DELETE suprime tramos de programa entre las líneas dadas.

EDIT permite enmendar o corregir ('editar') las líneas existentes de un programa.

LIST produce la enumeración de todo o parte del programa en pantalla, por la impresora o hacia el cassette (i.e. hacia un cauce dado).

MERGE recoge un programa de un fichero y lo 'congrega' con el programa que hay en memoria. Puede usarse por ejemplo, para añadir subrutinas útiles guardadas previamente en cassette, al programa en curso. El comando MERGE aceptará un fichero con un programa en BASIC, o un fichero con texto en ASCII -no aceptará ninguna otra clase de fichero, y en particular no aceptará ningún fichero con programa protegido.

NEW hace que el BASIC olvide el programa existente en memoria; limpia el área de programa, anula todas las variables y generalmente efectúa una limpieza para preparar la introducción de un 'nuevo' programa.

RENUM permite cambiar la numeración de parte o todo el programa. Las referencias a números de línea incluidos dentro de comandos GOTO, GOSUB, etc., quedan renumeradas.

SAVE hace que el BASIC 'guarde' el programa existente en memoria, o el contenido de un área de memoria, dejándolo grabado en un fichero en cinta cassette. El programa puede guardarse en la forma interna del BASIC, en la forma interna protegida, o como un fichero en ASCII. La grabación en cinta del contenido de un área de memoria crea un fichero Binario, que puede por ejemplo, ser un programa en código máquina o una copia de la imagen en pantalla.

10.2 Carga y Ejecución de Programas

CLEAR, CHAIN, CHAIN MERGE, LOAD, MEMORY, HIMEM, RUN, FRE

CLEAR anula todos los valores de las variables, cierra los ficheros en cassette y en general restaura las condiciones iniciales del BASIC, excepto que sigue manteniéndose el programa presente en memoria.

MEMORY permite que se cambie la memoria disponible para el BASIC, fijando la dirección del máximo byte de la memoria que puede ser usado por el BASIC.

La función HIMEM entrega la máxima dirección en memoria que puede en ese momento ser usada por el BASIC.

La función FRE computa y entrega la cantidad de memoria, dentro de los límites del BASIC, que en ese momento queda sin usar.

CHAIN y CHAIN MERGE permite 'encadenar' el programa actual con otro que se carga del periférico de almacenamiento. Con el comando CHAIN MERGE puede retenerse en memoria, parte o todo el programa presente en ella. Los valores de las variables siguen conservándose después de CHAIN y de CHAIN MERGE.

LOAD desecha el programa y las variables presentes en memoria, restaura todos los otros programas, y trae un nuevo programa del cassette dejándolo cargado en la memoria. Tanto ficheros con programas en BASIC como ficheros ASCII pueden ser cargados con este comando. La carga de un programa en BASIC protegido es futil, dado que inmediatamente será suprimido! Se puede usar también LOAD para traspasar a la memoria un fichero Binario.

Una de las formas del comando RUN comienza la ejecución del programa actual en memoria, al comienzo o bien a partir de un número de línea determinado. La otra forma del comando, obliga a que se cargue del cassette un nuevo programa y que la ejecución comience inmediatamente después a partir del principio -y pueden ser ficheros con programas en BASIC o ficheros ASCII. También puede usarse RUN para cargar y ejecutar inmediatamente programas en 'código máquina'.

10.3 Finalización de Ejecución

END

El programa se detiene y el BASIC vuelve al Modo Directo después de haber ejecutado la última línea de un programa. Se puede usar el comando END para terminar la ejecución en cualquier otro punto o puntos de un programa.

10.4 Estructuras de Iteración y Selección

FOR, GOSUB, GOTO, IF, ON x GOSUB, ON x GOTO, RETURN, WHILE

El BASIC admite dos formas de **bucles:** bucles preconfinados o FOR, y bucles condicionados o WHILE. Un bucle FOR tiene asociado con él una variable de control, que va pasando sucesivamente a través de una escala de valores -uso por iteración o ronda- hasta que se alcanza un determinado valor final. Puede especificarse el tamaño del incremento sufrido en cada ronda, incluyendo pasos negativos. El bucle FOR es saltado totalmente por el BASIC si el valor inicial de la variable de control sobrepasa el valor final. El comando NEXT marca el final de un bucle FOR.

Un bucle WHILE establece la repetición de una serie de instrucciones hasta que una determinada condición adopta el valor falso. El bucle WHILE es saltado completamente por el BASIC si la condición ya es falsa la primera vez. El comando WEND marca el final de un bucle WHILE. Tanto los bucles FOR como los WHILE pueden 'anidarse'.

Los comandos IF pueden usarse para establecer 'cruces', o elegir acciones alternativas u opcionales. La partícula IF va seguida de una expresión. Si la expresión produce un valor no-cero se elige la parte THEN, en caso contrario y si está presente, se elige la parte ELSE. Tanto la parte THEN como la ELSE pueden contener más de un comando, pero todos ellos deben estar en la línea encabezada por el comando IF. Los comandos IF también pueden 'anidarse' dentro de una misma línea.

Están admitidos los desvíos hacia subrutinas simples. El comando GOSUB llama o 'cita' la subrutina que comienza en una línea determinada. RETURN marca el final de la subrutina, y devuelve el control al comando que sigue al GOSUB que puso en marcha la subrutina. Todas las variables en BASIC son globales. Las subrutinas pueden ser 'recursivas'.

GOTO provoca un salto incondicional a la línea dada.

ON x GOSUB y ON x GOTO son comandos que evalúan la expresión 'x', y según el resultado obtenido, escogen una de entre una lista de líneas para desviarse o saltar hacia ella.

10.5 Designación de variables

DEFINT, DEFREAL, DEFSTR, DIM, ERASE

Cuando se da un nombre de variable o de función sin una marca explícita de clase, el BASIC debe suponer una determinada clase. La clase supuesta depende del primer carácter del nombre (que es una letra o carácter alfabético). Los comandos DEFINT, DEFREAL y DEFSTR pueden usarse para definir la clase prescrita para omisiones, con cada letra.

Observa que estas prescripciones para omisiones pueden cambiarse a voluntad, pero que la posibilidad de confusión es enorme.

A menos que esté explícitamente declarada en un comando DIM, el máximo valor de cada subíndice de una tabla se fija al valor 10 en la primera utilización de un elemento de la tabla (declaración implícita). El mínimo valor de cada subíndice es siempre cero.

Las tablas pueden ocupar grandes áreas de memoria. El comando ERASE se usa para dejar libre la memoria ocupada por las tablas, cuando ya no van a ser utilizadas o sus atributos van a cambiar.

10.6 Atributos de pantalla

MODE, BORDER, INK, SPEED INK

El comando MODE adopta para la pantalla uno de los tres modos posibles (véase Sección 4). Al cambiar el modo de pantalla se limpia la imagen y se restauran las condiciones de todas las ventanas de texto, cursores de texto, y la ventana de gráficos, el origen de la misma y el cursor.

BORDER fija el color o colores que han de usarse para el área que bordea el recuadro activo de la imagen en pantalla.

INK fija el color o colores que han de usarse para una determinada Tinta (véase Sección 4).

Cuando se fija una Tinta a dos colores, el color con que aparece cada punto de imagen en pantalla, alterna entre los dos especificados. SPEED INK fija el ritmo de alternancia en este caso.

10.7 Salida de Texto

PRINT, WRITE, ZONE, POS, VPOS

PRINT expone números y constantes literales hacia un cauce de salida dado. Los números se sacan en 'formato libre'. PRINT divide la salida en 'zonas de exposición' que obligan a una determinada ordenación. El comando ZONE puede usarse para cambiar el tamaño de dichas zonas. En un comando PRINT se puede usar la palabra clave USING para introducir una 'plantilla de forma', que controla la salida de todos los números y literales posteriores, sin tener en cuenta las zonas de exposición.

WRITE saca números y constantes literales hacia un cauce dado, insertando comas entre cada elemento y encerrando los literales entre comillas. Los números se sacan en 'formato libre'. WRITE es más utilizada para enviar datos hacia el cassette, de manera que puedan ser tomados de nuevo usando el comando INPUT para imponerlos como valores de variables.

Las funciones POS y VPOS entregan respectivamente la posición horizontal vertical del cursor característico del cauce especificado en la función.

10.7.1 Cauces de envío a pantalla

CLS, LOCATE, PAPER, PEN, TAG, TAGOFF, WINDOW, WINDOW SWAP

Todos estos comandos aceptan un parámetro de cauce, con el valor restringido a uno de los ocho cauces de pantalla -dado que fijan atributos específicos de pantalla.

CLS limpia la pantalla de un cauce dado, dejando el color corriente de Papel.

LOCATE sitúa el cursor del cauce en una posición dada, dentro de la ventana actual.

PAPER fija la Tinta que ha de usarse para el fondo de la imagen, para todos los caracteres subsecuentes enviados hacia el cauce dado.

PEN fija la Tinta que ha de usarse para el frente de la imagen, para todos los caracteres subsiguientes enviados hacia el cauce especificado.

TAG dirige todos los caracteres posteriores enviados por el cauce especificado hacia la posición que ocupa el cursor de gráficos. Esto permite exponer caracteres en posiciones arbitrarias de la pantalla.

TAGOFF cancela la opción establecida por TAG.

WINDOW fija la posición en pantalla de la ventana asociada a un cauce dado.

WINDOW SWAP intercambia ('canjea') todos los atributos de dos cauces de pantalla especificados. Cuando uno de los cauces es el cauce #0 (el cauce prescrito para omisiones) esto tiene el efecto de enviar las posteriores salidas del sistema a una nueva ventana de la pantalla.

10.7.2 El cauce hacia el cassette

CLOSEOUT, OPENOUT

Antes de que pueda escribirse texto en la cinta cassette, debe darse un comando OPENOUT para establecer el cauce hacia el cassette y especificar el nombre del fichero en que va a escribirse. Cuando se ha transferido todo el texto hacia el fichero de cassette, debe cerrarse el cauce de salida usando el comando CLOSEOUT.

10.8 Introducción de Texto

INPUT, LINE INPUT

Estos comandos aceptan un parámetro que especifique el cauce por donde se 'imponen' los datos. Aparte del cauce de entrada del cassette, toda la entrada de datos procede del teclado, y el cauce, de hecho, especifica por dónde se enviará la salida cuando en los comandos se requiere un aviso para la introducción de datos.

INPUT recaba una línea del 'buffer' asociado al cauce dado, y puede interpretar partes de ella como datos numéricos y partes como simple texto, dependiendo de los parámetros dados en el comando INPUT.

LINE INPUT recaba toda una línea del 'buffer' asociado a un cauce dado sin ninguna interpretación (pero con la misma imagen de línea) e 'impone la línea' como valor de una variable literal.

10.8.1 Introducción de datos por cassette

CLOSEIN, EOF, OPENIN

Antes de que puedan leerse datos de un fichero en cassette, dicho fichero debe abrirse usando un comando OPENIN, y especificando el nombre del fichero requerido. Cuando no se va a emplear más el fichero, debe ser cerrado usando un comando CLOSEIN.

Es un error intentar tomar datos de un fichero cuando se ha sobrepasado el final del mismo. La función EOF puede usarse para probar si hay registros posteriores al leído, y así detectar el final de fichero.

10.9 El Cassette en General

CAT, SPEED WRITE

CAT comienza a leer la cinta del cassette y reporta los ficheros que encuentra a medida que va haciendo las operaciones de lectura. CAT verifica si los ficheros que encuentra son legibles.

SPEED WRITE cambia la rapidez en que se escriben los datos en el cassette; cuanto más baja sea, menor será la probabilidad de errores, particularmente cuando se graban ficheros en cinta que serán leídos en otras máquinas. La velocidad de escritura no tiene ningún efecto sobre la lectura de las cintas, que se ajusta por sí misma a la que se haya usado en la grabación.

10.10 Series de Constantes

DATA, READ, RESTORE

Todos los comandos DATA en un programa, tomados según el orden de número de líneas, constituyen una 'serie de constantes' accesible en el programa. Esta 'serie' puede usarse mediante los comandos READ para apuntar las constantes como valores de variables. La posición dentro de esta 'serie', señalada internamente mediante un 'puntero', puede fijarse mediante el comando RESTORE.

10.11 Trazado de Gráficos

CLG, DRAW, DRAWR, MOVE, MOVER, ORIGIN, PLOT, PLOTR, TAG, TAGOFF, TEST, TESTR, XPOS, YPOS

CLG limpia la pantalla de gráficos.

ORIGIN establece la posición en pantalla ocupada por el origen de las coordenadas gráficas. Mediante parámetros opcionales también puede especificarse la ventana de gráficos, restringiendo el efecto de los comandos gráficos a una porción determinada de la pantalla.

DRAW y DRAWR trazan rectas en la pantalla de gráficos, desde la posición corriente del cursor de gráficos hasta, e incluyendo, un punto determinado. DRAW especifica el punto final en coordenadas absolutas. DRAWR especifica el punto final en coordenadas relativas a la posición corriente del cursor de gráficos.

MOVE y MOVER mueven el cursor de gráficos situándolo en una nueva posición. MOVE especifica la nueva posición en coordenadas absolutas. MOVER especifica la nueva posición en coordenadas relativas a la posición corriente del cursor de gráficos.

PLOT y PLOTR desplazan el cursor de gráficos hasta el punto dado y lo pintan. PLOT especifica el punto en coordenadas absolutas. PLOTR especifica el punto en coordenadas relativas a la posición corriente del cursor de gráficos.

Las funciones TEST y TESTR examinan un punto determinado de la pantalla de gráficos, y entregan el valor de Tinta usado en ese punto. TEST lo especifica en coordenadas absolutas. TESTR lo especifica en coordenadas relativas a la posición corriente del cursor de gráficos.

TAG hace que el texto enviado por un cauce de texto determinado, sea expuesto en la posición del cursor de gráficos, y no como habitualmente en la posición del cursor de texto. TAGOFF revoca el efecto provocado por TAG.

Las funciones XPOS e YPOS entregan respectivamente las coordenadas de la posición corriente 'x' e 'y' del cursor de gráficos.

10.12 Generación de Sonidos

ENT, ENV, ON SQ, RELEASE, SOUND, SQ

ENT y ENV fijan nuevos valores para las envolventes de Tono y de Volumen, respectivamente. Se dispone de quince envolventes de Tono y quince envolventes de Volumen.

SOUND intenta enviar un nuevo sonido e incluirlo en la 'cola de notas' embalsada en el canal especificado en el comando. SOUND esperará hasta que haya sitio para esa nota dentro de la secuencia o secuencias que se han especificado. SOUND desactiva cualquiera de los 'cepos' que se hayan preparado mediante el comando ON SQ para detectar sitios vacíos en los canales de sonido.

La función SQ entrega un valor que refleja el estado de la secuencia de notas del canal de sonido dado en la función -el valor indica si hay o no un sonido activo, cuántos sitios libres hay todavía en la secuencia, y si está estipulado Retención o Acorde para el siguiente sonido que ha de emitirse. La función SQ desactiva cualquier 'cepo' establecido mediante ON SQ para ese canal.

Los sonidos pueden 'retenerse' en espera de que sean emitidos. El comando RELEASE libera esa retención y hace que se emita el siguiente sonido que haya esperando en el canal especificado.

El comando ON SQ establece un 'cepo' que al ser activado pondrá en marcha una subrutina para las interrupciones por sitios vacíos en el canal especificado. La subrutina de interrupción se invoca inmediatamente en que la secuencia de notas no está llena; en caso contrario, se pondrá en marcha cuando se disponga de un sitio vacío. El 'cepo' establecido por ON SQ se desactiva cuando se produce una interrupción, o mediante la utilización de la función SQ.

10.13 Interrupciones por Temporizador

AFTER, EVERY, REMAIN

AFTER permite que el temporizador de demora mencionado en el comando envíe una señal de interrupción **después** de que ha transcurrido el período especificado. El temporizador queda desactivado cuando envía la interrupción.

EVERY permite que el temporizador de demora dado envíe una señal de interrupción **cada** vez que transcurre un período determinado. El temporizador continúa en activo hasta que sea explícitamente desactivado.

La función REMAIN entrega la cuantía de tiempo que **queda** todavía para que un temporizado determinado señale una interrupción; y el uso de la función desactiva dicho temporizador.

10.14 Admisión e Inhibición de las Interrupciones

EI, DI

DI desactiva la recepción de interrupciones. EI admite o faculta las interrupciones. DI y EI debieran usarse cuando se necesita ejecutar una secuencia de instrucciones en forma continua e ininterrumpida.

10.15 Teclado y Joystick

kEY, KEY DEF,INKEY$,JOY,SPEED KEY

Un comando KEY DEF cambia el estado de autorrepetición y el valor generado por la pulsación de una tecla, o a la situación de un conmutador del mando de juegos, en cada uno de los posibles 'turnos' del teclado. Se puede reconfigurar completamente el teclado usando los comandos KEY DEF.

Un comando KEY cambia la 'constante literal' asociada con una de las teclas ampliables del teclado. Hay treinta y dos teclas con posibilidad de ampliación, que están pensadas para usarse como 'teclas de función'. El tablero numérico autónomo está preparado como las primeras doce teclas con posible ampliación. Los comandos KEY pueden, por tanto, emplearse para programar la función ejercida por dichas teclas.

Si hay un carácter pendiente de recoger en el 'buffer' del teclado, la función INKEY$ entrega dicho valor; en caso contrario,INKEY$ entrega el literal 'vacío'.

La función INKEY examina directamente el estado de una determinada tecla.

La función JOY entrega directamente el estado de un determinado mando de juegos.

El comando SPEED KEY fija la demora al arranque y el período de repetición para las teclas autorrepetitivas del teclado.

10.16 Caracteres Definidos por el Usuario

SYMBOL, SYMBOL AFTER

La representación visiva de los caracteres en la pantalla (la 'matriz de forma') puede cambiarse a voluntad usando el comando SYMBOL. El comando SYMBOL sólo puede emplearse para caracteres que han sido declarados como modificables mediante el comando SYMBOL AFTER.

Inicialmente, los últimos dieciséis caracteres son implícitamente modificables.

Las matrices que dan forma a cada uno de los 256 caracteres usables se conservan grabadas en la ROM. Para cambiar la matriz de un carácter debe primero ser copiada en la memoria accesible para lectura y escritura (RAM). El comando SYMBOL AFTER declara que las matrices para todos los caracteres con valor superior a uno dado, han de ser copiadas en la memoria accesible. El comando SYMBOL AFTER reserva un área de memoria, cambiando automáticamente HIMEM, y copia en ella las matrices de los caracteres señalados.

10.17 Funciones Aritméticas

ATS, ATN, COS, DEG, EXP, FIX, INT, LOG, LOG10, MAX, MIN, PI, RAD, RANDOMIZE, RND, ROUND, SGN, SIN, SQR, TAN

ATN, COS, SIN y TAN son las funciones trigonométricas habituales (ATN es Arcotangente). Por omisión de especificación, están prescritas para aceptar los ángulos expresados en radianes. Después del comando DEG las funciones pueden trabajar en grados sexagesimales, de manera que COS, SIN y TAN aceptan los ángulos en grados, y ATN produce el resultado en grados. El comando RAD establece que las funciones vuelvan a trabajar en radianes. PI es una 'función' que da como resultado la representación más próxima posible de la máquina del número trascendente π.

EXP entrega el valor del número trascendente e elevado a la potencia señalada por el argumento de la función. LOG da el logaritmo natural (base e) del argumento especificado. LOG10 da el logaritmo en base 10. SQR calcula la raíz cuadrada del argumento dado.

RND entrega el siguiente número de la serie psudoaleatoria interna. RANDOMIZE fija un nuevo 'germen' para comenzar la extracción de números aleatorios -RANDOMIZE TIME es un método interesante de establecer un comienzo verdaderamente aleatorio.

MIN acepta una cantidad variable de argumentos y entrega el valor del **mínimo** argumento que se le haya especificado. MAX opera de igual manera, pero entrega el valor **máximo.**

FIX e INT dan valores **enteros.** FIX redondea el argumento hacia cero. INT redondea el argumento hacia -Infinito. ROUND redondea el argumento para obtener un cierto número de posiciones fraccionarias, redondeando al número más próximo (véase sección 2.9.1 para más comentarios sobre redondeo de números).

ABS entrega el valor absoluto, la 'magnitud', del argumento. SGN entrega el signo del argumento.

10.18 Funciones Literales (String)

BIN$, DEC$, HEX$, INSTR, LEFT$, LEN, LOWER$, MID$, RIGHT$, SPACE$, STR$, STRING$, UPPER$, VAL

Las funciones LEFT$, RIGHT$ y MID$ separan una determinada parte del argumento literal que se especifique. LEFT$ extrae un cierto número de caracteres empezando por el extremo-izquierdo del literal. RIGHT$ los toma de la parte derecha. MID$ los separa de la parte 'media', o mejor dicho, a partir de una posición dada del literal.

MID$ puede usarse también como comando, de manera que parte del literal que se mencione como parámetro, queda sustituido por el otro literal asignado en el comando.

INSTR examina si un literal está incluido o pertenece a otro literal dado.

LEN calcula la longitud, número de caracteres, de un literal.

LOWER$ entrega una copia del argumento especificado, con todas las letras minúsculas sustituidas por sus equivalentes mayúsculas. UPPER$ de forma similar, sustituye minúsculas por mayúsculas.

STRING$ entrega una constante literal formada por el carácter dado repetido un determinado número de veces. SPACE$ forma un literal que consta de un determinado número de espacios en blanco.

BIN$, DEC$, HEX$ y STR$ producen una representación literal del valor de su argumento numérico. BIN$ produce una constante literal formada por símbolos binarios; HEX$ por símbolos hexadecimales. La función STR$ produce el literal correspondiente a un numeral en formato libre. DEC$ produce un literal 'conformado' de acuerdo con una 'plantilla de formato' similar a la que puede emplearse en el comando PRINT USING. La función VAL se aplica a un literal e interpretándolo como número de la misma manera que el comando INPUT, entrega el valor resultante de la interpretación.

10.19 Funciones para Conversión de Clases

ASC, CHR$, CINT, CREAL, UNT

ASC toma el primer carácter de un literal y entrega el valor entero -código- que le corresponde.

CHR$ toma un código y entrega como resultado un literal de un carácter.

CINT se aplica a un valor numérico y lo redondea a entero, redondeando al más próximo. El resultado debe estar en la banda -32768..32767.

CREAL se aplica a un valor numérico y lo convierte a Real.

UNT acepta un valor numérico y lo redondea al Entero Sin-signo más próximo.

10.20 Operaciones a Nivel de Máquina

CALL, INP, OUT, PEEK, POKE, WAIT

CALL apela (llama) a una rutina en código máquina que comienza en una determinada dirección, traspasándole un cierto número de parámetros (por valor). El signo de operación '@' puede usarse para ceder a la rutina la dirección donde se encuentra el valor de la variable. Véase Apéndice II para una descripción más detallada.

La función PEEK y el comando POKE respectivamente 'miran' y 'meten' bytes en una determinada celdilla de la memoria de la máquina.

INP, OUT y WAIT permiten tener acceso al espacio de entrada/salida de la máquina.

10.21 Cepos para Atrapar Errores

ERL, ERR, ERROR, ON ERROR GOTO, RESUME, ON BREAK

Cuando BASIC detecta un error, la acción que tiene prescrita es emitir un mensaje de error, y volver al Modo Directo. Esta acción prescrita puede revocarse mediante un comando ON ERROR GOTO que señala el número de línea de la rutina que ha de ponerse en marcha para el resarcimiento del error producido. La rutina de tratamiento del error puede usar la función ERR para conocer cuál ha sido el error producido y la función ERL para determinar la línea donde se produjo. El comando RESUME debe emplearse para que se 'reanude' la ejecución normal del programa.

ON BREAK permite que el programa establezca de manera análoga un 'cepo para atrapar' cuando se ha pulsado la secuencia de teclas **ESC/ESC** que habitualmente detiene la ejecución del programa, y tomar en su lugar la acción alternativa que se desee.

10.22 Desarrollo del Programa

CONT, STOP, TRON, TROFF

El comando STOP hace que se pare la ejecución de un programa de manera que pueda reanudarse posteriormente, mediante el comando CONT (siempre que el programa no sea alterado durante la parada). Se pueden insertar instrucciones STOP en un programa para hacer que se pare en determinados puntos, y poder examinar el estado del programa y los valores de las variables en ese momento. Luego hacer que siga.

Los comandos TRON y TROFF ponen y quitan un mecanismo de 'rastreo' de la ruta seguida en la ejecución del programa. Cuando está puesto el rastreo, mediante TRON, antes de que se ejecute cada línea el BASIC expondrá el número en la pantalla del monitor, encerrándolo entre corchetes cuadrados. De esta manera se consigue un provechoso y simple seguimiento de la ejecución de los programas.

11. Los Comandos y las Funciones Intrínsecas

Esta sección contiene una descripción detallada de todos los Comandos y Funciones incorporadas en el BASIC, y están colocadas según orden alfabético de la palabra **clave.** A lo largo de esta sección se usan los siguientes términos del metalenguaje:

<expresión direccional>

Una <expresión numérica> que cuando redondeada a entero produce un valor en la banda -32768..65535. Los valores negativos pueden convertirse luego a su equivalente 'sin-signo' simplemente añadiéndole 65536.

<nombre variable múltiple>

El nombre de una tabla que puede incluir su 'marca de clase'.

<color>

Una <expresión entera> que produce un valor en la banda 0..26, que especifica uno de los 27 colores posibles (véase Apéndice VI).

<expresión>

Cualquier serie de operadores y operandos permitidos que produce un valor de cualquier clase.

<tinta>

Una <expresión entera> que produce un valor en la banda 0..15, que especifica una de las 16 posibles tintas de la 'paleta'.

<expresión entera>

Una <expresión numérica> que cuando se redondea a entero, produce un valor en la banda -32768..32767.

<número de línea>

Un número decimal en la banda 1..65535.

<banda de números de línea>

Una <banda de números de línea> hace referencia a todas aquellas líneas cuyos números están dentro de las 'cotas' que definen la banda. Puede tomar una de las siguientes formas:

<número de línea>	Sólo hay una línea dentro de la banda.
<número de línea>–	Las líneas desde la cota inferior dada hasta el final del programa.

—<número de línea> Las líneas desde el comienzo del programa hasta la cota superior dada.

<número de línea>—<número de línea>

Las líneas cuyos números son mayor o igual que la cota inferior dada y menor o igual que la cota superior dada.

<expresión logical>

El BASIC no admite una clase separada de datos **booleanos.** El valor entero 0 es considerado siempre como falso, y cualquier otro valor entero como cierto. Una <expresión logical> es por lo tanto, una <expresión entera> simplemente conocida con nombre.

<tinta reducida>

Una <expresión entera> que produzca un valor en la banda 0..15, que especifica una de las 16 tintas posibles simultáneamente. El valor es a continuación 'cribado' para forzarlo en la banda que corresponde al modo corriente de pantalla; por lo tanto:

Modo 0:	Criba = 15, banda final = 0..15
Modo 1:	Criba = 3, banda final = 0..3
Modo 2:	Criba = 1, banda final = 0..1

<constante numérica>

Véase Sección 2.4.

<expresión numérica>

Una serie de operadores y operandos que al evaluarse produce un valor Real o Entero, incluyendo expresiones que previamente entraban en las categorías denominadas relacional y logical.

<variable numérica>

El nombre que sirve de referencia a una variable, que puede ser un elemento de una tabla, pero cuya clase sea Real o Entera.

<literal entrecomillado>

Un <literal entrecomillado> es una cadena de caracteres que contiene entre 0 y 255 caracteres, y que está delimitada por el signo comillas al principio y por el signo comillas al final de la misma; o bien puede empezar con comillas y estar terminada por la marca de 'retorno de carro'.

<variable simple>

El nombre que sirve de referencia para una variable pero excluyendo las variables múltiples o tablas. La referencia puede incluir una 'marca de clase'.

<expresión de cauce>

Una <expresión entera> que produce un valor en la banda 0..9. Toda la entrada y salida de datos se dirigen a su destino a través de dicho cauce.

Los cauces de salida de información son:

0..7	Los ocho cauces posibles hacia la pantalla del monitor.
8	La Impresora.
9	Los ficheros de salida en Cassette.

Los cauces de entrada de información son:

0..7	El Teclado, con cualquier 'aviso' enviado al cauce de pantalla correspondiente.
8	Teclado, con cualquier aviso enviado por el cauce de impresora.
9	El fichero de entrada en Cassette, con cualquier 'aviso' ignorado.

<expresión literal>

Una serie de operadores y operandos que produce un valor de índole o clase literal.

<variable literal>

El nombre que sirve de referencia para una variable, que puede ser un elemento de una tabla, cuya clase o índole es **literal** (string).

<marca de clase>

Uno de los caracteres siguientes:

%	indica numeral Entero
!	indica numeral Real
$	indica Literal

<variable>

Un nombre que sirve de referencia a cualquier clase de variable. Incluye variables que son elementos de una tabla, en cuyo caso deberán incluirse al final del nombre los subíndices apropiados.

<nombre de variable>

El nombre de una <variable simple>, que puede incluir su <marca de clase>.

ABS

Valor absoluto. **Función**

Uso

Determinar el valor absoluto de una expresión dada.

Forma

ABS(<expresión numérica>)

Notas

ABS de un valor negativo da ese valor cambiado de signo. ABS de un valor positivo da ese valor sin cambio.

Claves Asociadas

SGN

AFTER

Suscita una subrutina después de que ha transcurrido un período dado.

Comando

Uso

El sistema mantiene un reloj en tiempo real. El comando AFTER permite que en un programa BASIC se disponga de subrutinas que van a ser suscitadas en algún momento en el futuro. Se dispone de cuatro temporizadores de demora, cada uno de los cuales puede tener una subrutina asociada con él.

Forma

AFTER<lapso de tiempo>[,<número temporizador>]GOSUB<número línea>

El <lapso de tiempo> es una <expresión entera> que especifica cuánto tiempo ha de transcurrir antes de que la subrutina sea llamada. Este tiempo se mide en 1/50-avos de segundo.

El <número temporizador> es una <expresión entera> que especifica cuál de los cuatro posibles temporizadores de demora va a usarse. La expresión debe producir un valor en la banda 0..3. Si se omite la expresión, se supone el 0.

Notas

Cuando ha transcurrido el tiempo especificado, la subrutina es suscitada automáticamente, igual que si se hubiera dado un GOSUB en la posición corriente del programa. Cuando la subrutina termina, usando un comando normal RETURN, el programa continúa ejecutándose donde se interrumpió.

Los temporizadores tienen diferentes prioridades de interrupción. El temporizador 3 tiene la máxima prioridad y el temporizador 0 la mínima. Véase Sección 9 para el comentario del efecto de las prioridades.

Los comandos AFTER pueden incluirse en cualquier momento, restaurando la subrutina y el lapso de tiempo asociado con el temporizador dado en el comando. Los temporizadores de demora son los mismos que los empleados en el comando EVERY, de manera que un AFTER revoca cualquier previo EVERY para el mismo temporizador, y viceversa.

Claves Asociadas

EVERY, REMAIN

ASC

Dice el valor ASCII del carácter. **Función**

Uso

Para conseguir el valor numérico del primer carácter de un literal, dado que la codificación de caracteres ASCII es la empleada.

La inversa de CHR$.

Forma

ASC(<expresión literal>)

Donde la <expresión literal> contiene un literal con una longitud mínima de un carácter.

Claves Asociadas

CHR$

ATN

Arcotangente. **Función**

Uso

Para calcular el arcotangente de un valor dado. El resultado obtenido es un ángulo bien en radianes (en la banda -PI/2..+PI/2) o en grados (en la banda -90..+90), dependiendo del modo vigente.

Forma

`ATN(<expresión numérica>)`

Notas

`DEG` y `RAD`, respectivamente, cambiar el BASIC al modo grados o radianes.

Claves Asociadas

`SIN, COS, TAN, DEG, RAD`

AUTO

Numeración automática de líneas. **Comando**

Uso

Ayudar a la introducción del programa. Hace que BASIC sea el que genera los números de línea.

Forma

AUTO[<número de línea>][,<incremento>]

Si se omite el <número de línea>, se supone 10.

El <incremento> adopta la forma de un <número de línea>. Si se omite el <incremento>, se supone 10.

Notas

AUTO genera el primer número de línea especificado, y permite el tecleado del texto del programa. Al final de la línea, automáticamente se genera el siguiente número de línea, añadiéndole incremento al número previo. Pulsando **ESC** se termina la acción AUTO, se desecha la línea corriente y el BASIC regresa al Modo Directo.

Si se genera un número de línea que ya está en uso, se expone un asterisco de aviso después del número. El contenido de la línea será sustituido a no ser que no se teclee nada antes del retorno de carro, o se termina la acción AUTO.

BIN$

Literal binario. **Función**

Uso

Para producir una serie de símbolos binarios representando el valor de la expresión dada.

Forma

BIN$(<expresión entera sin-signo>[,<anchura campo>])

La <expresión entera sin-signo> debe producir un valor en la banda -32768..65535. Este valor se trata como un valor sin signo de 16 bits, para ser convertido en una serie de cifras binarias.

La <anchura campo> opcional es una <expresión entera> dando el mínimo tamaño de la serie literal (string) que se produce. La expresión debe resultar en un valor en la banda 0..16.

Notas

BIN$ siempre genera tantos caracteres como se requieren para representar el número (cero genera como mínimo una cifra). Si se especifica una <anchura campo> y el literal es demasiado corto, entonces el literal se rellena hasta la longitud especificada con ceros delanteros. El literal no se trunca si es demasiado largo.

Claves Asociadas

HEX$, DEC$, STR$

BORDER

Fija el color del área que bordea la imagen. **Comando**

Uso

Cambiar el color del 'borde' de la pantalla.

Forma

BORDER <color>[,<color>]

Si se especifican dos colores, el 'borde' alterna entre los dos.

Claves Asociadas

INK

CALL

Cede el control a una subrutina externa. **Comando**

Uso

Permite que desde el BASIC sean 'citadas' (o llamadas) subrutinas desarrolladas externamente.

Forma

CALL <expresión direccional>[,<lista de: <parámetro>]

La <expresión direccional> indica la dirección de comienzo de la subrutina.

Los <parámetro>s puede ser bien una <expresión numérica> o bien: @ <variable>

Notas

Los parámetros se ceden a la subrutina por valor.

Un parámetro que sea una <expresión numérica> y produzca un valor Real, es automáticamente forzado a ser un entero sin-signo.

Un parámetro de la forma @ <variable> permite ceder a la subrutina la dirección de la variable.

Véase Apéndice II para un comentario detallado de la manera de efectuar la cesión, y la forma de los datos que se transpasan, etc.

Claves Asociadas

UNT

CAT

Muestra el catálogo de los ficheros grabados en una cinta. **Comando**

Uso

El comando CAT hace que el BASIC comience la lectura del cassette y vaya mostrando los nombres de todos los ficheros que encuentra.

Forma

```
CAT
```

Notas

Cualquier fichero abierto es abandonado, y la salida alojada en el 'buffer' se pierde.

No se efectúa lectura ni escritura de información, y el programa en curso no se ve afectado.

A medida que encuentra ficheros en cinta, muestra el siguiente mensaje:

```
FILENAME block9?Ok
```

Dando además del nombre del fichero, el número de bloque (el 9 en este caso) del más recientemente leído para el fichero dado. Las siglas 'Ok' indican que dicho bloque se leyó correctamente. El 'banderín' (el testigo) indica la clase de fichero (en este caso hemos puesto una interrogación); y puede ser uno de los siguientes caracteres:

$	un fichero con un programa en BASIC.
%	un fichero con un programa 'protegido' en BASIC.
*****	un fichero con texto en caracteres ASCII.
&	un fichero Binario (contenido de una zona de memoria).
'	un fichero Binario 'protegido'.

Pueden aparecer otros caracteres como 'banderín' si el fichero no fue producido por el BASIC.

CAT puede usarse para comprobar que los ficheros en la cinta son legibles, por ejemplo después de haberlos guardado mediante SAVE.

Claves Asociadas

SAVE, LOAD

CHAIN
CHAIN MERGE

Encadenar un programa en cinta al programa actual. **Comando**

Uso

Permite que desde el programa actual se 'traiga' de la cinta y se cargue en memoria otro programa, reteniendo las variables vigentes y parte o todo el programa ya existente en la memoria.

Formas

```
CHAIN<nombre fichero>[,<expresión de número de línea>]
CHAIN MERGE<nombre fichero>[,[<expresión de número de línea>]
                                [,DELETE<gama de número de línea>]]
```

Siendo:

El <nombre de fichero> una <expresión literal> que indica el nombre del fichero donde está contenido el programa que va a ser 'engarzado' al existente en memoria.

La <expresión de número de línea> es una <expresión numérica>, que cuando se redondea a Entero produce un valor en la banda -32768..65535. Los valores negativos son luego convertidos a sus equivalentes sin-signo (añadiendo 65536), y deben quedar por tanto en la banda 1..65535. Así se fija el número de línea en que comenzará la ejecución una vez que el nuevo programa haya sido cargado; y si dicha línea no existe, da como resultado un Error 8. Si no se da ningún número de línea, la ejecución comienza a partir de la línea de número menor.

La opción DELETE hace que se supriman del programa corriente las líneas comprendidas en la gama dada, antes de que sea traído de la cinta y cargado en memoria el nuevo programa.

Notas

Si el nombre del fichero indicado por la <expresión literal> produce una constante literal 'vacía', entonces el BASIC intenta traer de la cinta y cargar en memoria el primer fichero que encuentre en la cinta.

Si el primer carácter del nombre del fichero dado por la <expresión literal> es la admiración (!) dicho signo se quita del nombre del fichero y tiene como efecto la supresión de los mensajes habitualmente generados por el BASIC al hacer la lectura del cassette.

Ambas formas del comando CHAIN y CHAIN MERGE, tienen los siguientes efectos sobre la situación actual:

-Se olvidan todas las Funciones de Usuario
-Se desactivan los cepos ON ERROR GOTO.
-Todos los ficheros abiertos son abandonados (y se pierde cualquier dato en el 'buffer' de salida).
-Se efectúa la acción de RESTORE.
-Se olvidan todos los comandos FOR, WHILE y GOSUB que hubiera pendientes.

Durante la operación CHAIN MERGE, que 'ensambla' al programa actual el nuevo procedente del cassette, cualquier línea en el programa nuevo con el mismo número que el de una línea del programa existente, sustituirá a la línea del existente.

No es posible hacer la operación CHAIN MERGE con un programa protegido en el cassette.

La renumeración mediante el comando RENUM de un programa que contenga estas instrucciones, sólo renumerará los números de línea dentro de la <gama de números de línea> de la opción DELETE, pero no afectará a la <expresión de número de línea>. En CHAIN esta expresión se refiere a una línea en un programa separado, así que no puede ser renumerada. En CHAIN MERGE el sistema no puede decir si la <expresión de número de línea> se refiere a una línea que no será afectada por el 'ensamblamiento' de los dos programas, de manera que no la renumera para ser coherente con el caso de CHAIN.

Claves Asociadas

LOAD, MERGE, RUN, SAVE

CHR$

Convierte a carácter. **Función**

Uso

Para convertir un valor numérico a su equivalente 'literal' o carácter; dado que se usa la codificación ASCII para los caracteres.

Es la inversa de ASC.

Forma

CHR$(<expresión entera>)

Siendo <expresión entera> una que produzca un valor en la banda 0..255.

Claves Asociadas

ASC

CINT

Convierte a entero. **Función**

Uso

Para convertir el valor dado a su representación Entera, si es posible.

Forma

`CINT(`<expresión numérica>)

La <expresión numérica> es convertida por redondeo a Entero, que debe estar en la banda -32768..32767.

Claves Asociadas

`CREAL, INT, FIX, ROUND, UNT`

CLEAR

Anula todas las variables y ficheros. **Comando**

Uso

Para hacer que todas las variables sean cero si son numéricas, o 'vacías' si son variables literales. Elimina todas las tablas. Olvida todas las funciones de usuario. Abandona todos los ficheros.

Forma

```
CLEAR
```

Notas

`CLEAR` hace que el BASIC olvide cualquier comando `FOR`, `WHILE` o `GOSUB`, que hubiera pendiente.

Cualquier información en el 'buffer' de salida hacia el cassette se pierde.

CLG

Limpia la pantalla de gráficos. **Comando**

Uso

Para dejar en 'claro' la pantalla de gráficos.

Forma

CLG [<tinta cribada>]

La pantalla de gráficos se rellena con la tinta mencionada, 'filtrada' según el modo de pantalla vigente. Si no se especifica ninguna tinta, se usará la especificada en el comando CLG más reciente, o bien la tinta cero si no se hubiera ejecutado ningún comando CLG anteriormente.

Claves Asociadas

CLS, ORIGIN

CLOSEIN

Cierra los ficheros de entrada desde cassette. Comando

Uso

Para terminar de usar un fichero de entrada en cassette.

Forma

CLOSEIN

Claves Asociadas

OPENIN, CLOSEOUT

CLOSEOUT

Cierra un fichero de salida hasta el cassette. **Comando**

Uso

Para terminar de usar un fichero de salida en cassette.

Forma

```
CLOSEOUT
```

Notas

Cualquier dato que hubiera en el 'buffer' de memoria, para ser transferido a la cinta, es escrito en el fichero antes de ser cerrado.

Claves Asociadas

```
OPENOUT, CLOSEIN
```

CLS

Limpia una ventana de texto. **Comando**

Uso

Para dejar en 'claro' (Tinte de Papel) la ventana de texto correspondiente a un cauce dado.

Forma

`CLS[#<expresión de cauce>]`

La <expresión de cauce> debe producir un valor entero en la banda 0..7, que corresponde al cauce de envío de texto hacia la pantalla. Si se omite, el sistema adopta el cauce #0.

Notas

La posición del cursor se repone a la esquina superior izquierda de la ventana asociada al cauce dado en el comando.

La exposición del carácter 'salto de página', que es `CHR$(12)` tiene el mismo efecto.

CONT

Seguir la ejecución, continuar después de **[ESC][ESC]** STOP o bien END.

Comando

Uso

Cuando se ha parado un programa mediante la pulsación de **[ESC][ESC]**, o por una instrucción STOP o una END, el comando CONT hace que siga la ejecución del programa en el punto en que se cortó.

Forma

CONT

Notas

CONT será rechazado si se ha alterado el programa actual de alguna manera mientras estuvo parada su ejecución.

COS

Coseno de un ángulo. **Función**

Uso

Para calcular el coseno de un ángulo dado. En el modo Radianes, el ángulo viene expresado en radianes. En el modo Grados, el ángulo viene expresado en grados.

Forma

`COS(<expresión numérica>)`

La <expresión numérica> indica el ángulo en radianes o en grados. El valor de dicho ángulo cuando está expresado en radianes, debe estar en la banda aproximada de -200.000..+200.000.

Notas

Con ángulos mucho mayores que 2*PI (360 grados) la exactitud de esta función se ve crecientemente afectada hacia mayor error, de acuerdo con el paso a la escala angular dentro de la banda -PI..+PI (-180..+180 grados). En lugar de dar un valor muy inexacto, el BASIC no evaluará el coseno para ángulos mucho mayores de la banda citada anteriormente.

Claves Asociadas

`SIN, TAN, ATN, DEG, RAD`

CREAL

Convierte un valor a Real. **Función**

Uso

Para convertir el valor de una expresión dada a su representación Real.

Forma

CREAL (<expresión numérica>)

Claves Asociadas

CINT

DATA.

Declara datos constantes a usar en un programa. **Comando**

Forma

DATA<lista de: <constante>

siendo <constante> cualquier clase de <constante numérica>, una <constante literal entrecomillada> o una <constante literal sin entrecomillar>:

una <constante numérica> puede estar rodeada de un <espacio blanco>, que será ignorado.

una <constante literal entrecomillada> puede estar precedida y seguida de <espacio blanco> que será ignorado. Las comillas posteriores pueden omitirse si el literal es el último elemento de la lista.

una <constante literal sin entrecomillar> es una serie de caracteres, entre 0 y 255, que son tomados como un solo elemento. Una coma, un dos-puntos o un 'retorno de carro' termina un literal sin comillas, dentro de este contexto. Una <constante literal sin entrecomillar> puede estar precedida y seguida de <espacio blanco> que será ignorado.

Notas

Todas las instrucciones DATA, ordenadas según sus números de línea, dentro de un programa forman una 'serie' de constantes que pueden ser apuntadas como valores de variables mediante el comando READ. A medida que se va apuntando cada constante, el BASIC avanza el 'puntero' para señalar a la siguiente constante. La instrucción RESTORE hace que dicho 'puntero', repunte para señalar un determinado número de línea dentro de la serie contenida en el programa.

Los comandos DATA no son ejecutados. Las constantes no se comprueban en cuanto a validez, hasta que aparecen en una READ, en cuyo momento las constantes numéricas solamente pueden corresponder a variables numéricas de READ, y las constantes literales (de ambos tipos) sólo pueden ser apuntadas como valores de variables literales -teniendo bien en cuenta que las constantes numéricas pueden ser tratadas como <constantes literales sin entrecomillar>.

Claves Asociadas

READ, RESTORE

DEC$

Genera una representación literal conformada de un numeral. **Función**

Uso

Para crear una representación literal del valor de la expresión dada en la función, de acuerdo con una determinada 'plantilla de conformación'.

Forma

DEC$(<expresión numérica>,<plantilla de conformación>)

El valor de la <expresión numérica> es el convertido a constante literal.

La <plantilla de conformación> es una <expresión literal> que se usa a modo de 'horma' para dar forma específica a la serie de dígitos resultantes de aplicar la función.

Notas

El valor de la <expresión numérica> es convertido a una serie de símbolos decimales, de acuerdo con la forma expresada por la <plantilla de conformación>, que se interpreta análogamente a lo explicado en la instrucción PRINT USING.

La <plantilla de conformación> no es totalmente completa como en la instrucción PRINT USING, sino que sólo puede contener los siguientes caracteres:

+ - $ * # , . ↑

Claves Asociadas

STR$, HEX$, BIN$, PRINT USING

DEF FN

Notificar una Función de Usuario.

Comando
Ilegal en el Modo Directo

Uso

El BASIC permite que en el programa se definan y usen funciones que den como resultado un valor simple. La instrucción DEF FN es la parte que corresponde a la definición.

Forma

DEF FN<nombre función>[(<argumentos formales>)]=<expresión>

El <nombre función> tiene la misma forma que un <nombre variable> (incluyendo el opcional <designador clase>), siendo el nombre completo de la función el FN<nombre función>. El valor dado por la función es de la misma índole que el <nombre función>.

Los <argumentos formales> adoptan la forma de <lista de: <nombre variable simple>.

La <expresión> puede incluir no sólo los argumentos formales de la función, sino también cualquier otra constante, variable o función. El resultado de evaluar la expresión debe ser compatible con la clase designada para la función.

Notas

Una función de usuario se aplica cuando se incluye como operando de una función. Dicha aplicación toma la forma similar:

FN<nombre>[(<argumentos actuales>)]

Los <argumentos actuales> adoptan la forma de <lista de: <expresión>. Deben darse la misma cantidad de argumentos actuales como argumentos formales aparecen en la definición. La clase de cada argumento actual debe ser compatible con la del argumento formal correspondiente (que es el que ocupa la misma posición en la lista).

Los argumentos formales son locales a la función. Las variables con igual nombre al de los argumentos formales no se ven afectadas al aplicar la función, ni son accesibles desde dentro de la función.

Si la clase de la función no está explícitamente designada, cuando se aplica se supone el valor prescrito para omisiones. Si la clase de un argumento formal no está explícitamente designada, también se aplica la clase prescrita para omisiones cuando se utiliza la función.

DEFINT
DEFREAL
DEFSTR

Prescribir la clase para omisiones de los nombres. **Comando**

Uso

Todos los nombres de variables y de funciones tienen asociada una cierta clase con ellos. Esta clase puede estar explícita o implícitamente designada en el nombre, dependiendo de lo que se haya prescrito para los nombres mediante los comandos de esta sección.

Formas

```
DEFINT    <lista de:<gama de letras>
DEFREAL   <lista de:<gama de letras>
DEFSTR    <lista de:<gama de letras>
```

siendo <gama de letras> bien <letra> o bien <letra>-<letra>
y con la notación <letra>-<letra> se define una gama de letras inclusiva.

Notas

Cuando se encuentra un nombre de variable sin un <designador de clase> explícito, se supone la clase prescrita para omisiones. Dicha clase depende del primer carácter del nombre, de la inicial. La prescripción por omisión está fijada para cada letra mediante el más reciente de estos comandos en cuya <lista de:<gama de letras> estaba incluida dicha letra.

```
DEFINT    establece para omisiones que sean Enteras
DEFREAL   establece para omisiones que sean Reales
DEFSTR    establece para omisiones que sean Literales
```

El estado inicial prescrito para omisiones, en todas las letras, es el de Real. Dicho estado se restaura a las condiciones iniciales si se usan los comandos:

```
LOAD, RUN, CHAIN, NEW y CLEAR
```

DEG

Establece el modo grados para las funciones trigonométricas. **Comando**

Uso

Las funciones trigonométricas SIN, COS, TAN y ATN operan sobre radianes cuando no se especifica lo contrario. Después de un comando DEG operan en grados sexagesimales.

Forma

DEG

Notas

El modo Grados persiste hasta que se ejecuta un comando RAD o CLEAR, o se carga o ejecuta un nuevo programa mediante LOAD o RUN.

Claves Asociadas

RAD, ATN, COS, SIN, TAN

DELETE

Suprime líneas de programa. Comando

Uso

Para quitar parte del programa actual.

Forma

DELETE<banda de números de línea>

Notas

Elimina las líneas incluidas en la gama dada en el comando. El BASIC regresa el Modo Directo después de ejecutar el comando DELETE.

Claves Asociadas

NEW

DI

Inhibe las posibles interrupciones. Comando

Uso

Impide que se atiendan las señales de interrupción hasta que explícitamente sean 'facultadas' mediante el comando EI, o implícitamente al final de la subrutina que haya sido puesta en marcha como consecuencia de una interrupción.

Forma

DI

Notas

La inhibición de las señales de interrupción asegura que no se atienden las interrupciones que haya posteriormente, exceptuando las interrupciones de 'corte' (Break), hasta que no vuelva a 'habilitarse' la aceptación de interrupciones.

Si las interrupciones se inhiben en una subrutina suscitada precisamente por interrupciones, quedan implícitamente facultadas de nuevo al ejecutar la instrucción RETURN que marca el final de dicha subrutina.

Claves Asociadas

EI

DIM

Reserva espacio para una tabla de las dimensiones dadas. **Comando**

Uso

Con el fin de separar en la memoria el espacio correspondiente a una tabla, el BASIC debe saber cuántas dimensiones tiene, y cuál es el valor máximo permitido para cada una de ellas. El comando DIM permite establecer explícitamente dichas dimensiones.

Forma

DIM<lista de: <variable subindicada>

Siendo <variable subindicada> la formada por <nombre variable>(<lista dimensiones>) y la <lista dimensiones> viene dada por <lista de: <expresión entera>

Notas

Cada <expresión entera> en la <lista dimensiones> fija el valor máximo que está permitido para el subíndice, el subscripto correspondiente. El valor mínimo de cada subíndice es 0.

Las dimensiones de la tabla pueden fijarse explícitamente mediante el comando DIM, o implícitamente cuando se encuentra en el programa un elemento de ella. Cuando se declara implícitamente, el máximo valor de cada subíndice es de 10.

Es erróneo intentar cambiar las dimensiones que hayan sido previamente declaradas (explícita o implícitamente). Pero puede dejarse libre el espacio que ocupa una tabla, mediante el comando ERASE.

No hay ninguna restricción sobre el número de dimensiones que puede tener una tabla, siempre que no lo impida la longitud de la línea o el tamaño de la memoria!

Claves Asociadas

ERASE

DRAW

Traza una recta en la Pantalla de Gráficos. **Comando**

Uso

Para dibujar una línea en la pantalla a partir de la posición corriente hasta una posición absoluta.

Forma

DRAW<coordenada x>,<coordenada y>[,<tinta cribada>]

La <coordenada x> y la <coordenada y> deben ser <expresiones enteras>.

Si no se especifica ninguna <tinta cribada> se usará la tinta más recientemente empleada en un comando DRAW, DRAWR, PLOT o PLOTR. Si éste es el primero de tales comandos, se usa la Tinta 1.

Notas

Esta instrucción traza una línea desde la posición corriente del cursor de gráficos hasta, e incluyendo, el punto especificado en el comando. El cursor de gráficos se queda entonces en la posición de destino.

Mientras que los dos extremos de la recta pueden caer fuera de la ventana corriente de gráficos, sólo la porción interna a dicha ventana es trazada en pantalla.

Claves Asociadas

DRAWR, PLOT, PLOTR, MOVE, MOVER, TEST, TESTR

DRAWR

Traza una recta en la Pantalla de Gráficos. **Comando**

Uso

Para dibujar una línea recta en la pantalla a partir de la posición corriente hasta una posición relativa a la actual.

Forma

DRAWR<incremento x>,<incremento y>[,<tinta cribada>]

El <incremento x> y el <incremento y> deben ser <expresiones enteras>.

Si no se especifica ninguna <tinta cribada> se usa la Tinta más recientemente especificada en un comando DRAW, DRAWR, PLOT o PLOTR. Si éste es el primero de tales comandos, se usará la Tinta 1.

Notas

Esta instrucción traza una recta desde la posición corriente del cursor de gráficos hasta, e incluyendo, el punto cuya posición viene dada añadiendo el <incremento x> a la coordenada x actual, y añadiendo el <incremento y> a la coordenada y actual. El cursor de gráficos se queda en el punto de destino.

Mientras que los dos extremos de la línea pueden caer fuera de la ventana corriente de gráficos, sólo la porción interna a dicha ventana es trazada en pantalla.

Claves Asociadas

DRAW, PLOT, PLOTR, MOVE, MOVER, TEST, TESTR

EDIT

Revisar una línea de programa. **Comando**

Uso

Para enmendar líneas de programa de manera individual.

Forma

EDIT<número línea>

Notas

Con este comando el BASIC entra en el Modo de Edición con la línea dada en el comando como línea a revisar (editar).

Véase la Sección 6 para una descripción completa del Modo de Edición.

EI

Faculta el tratamiento de las interrupciones. **Comando**

Uso

Para volver a habilitar las interrupciones inhibidas mediante un comando DI.

Forma

EI

Notas

Las interrupciones quedan implícitamente facultadas mediante el comando RUN.

Si las interrupciones fueron inhibidas en una subrutina suscitada por interrupciones, quedan implícitamente facultadas de nuevo al ejecutar la instrucción RETURN que marca el final de dicha subrutina.

Claves Asociadas

DI

END

Finaliza la ejecución del programa. Comando

Uso

Para marcar el fin de un programa y detener la ejecución.

Forma

END

Notas

Pueden aparecer en cualquier parte del programa cualquier cantidad de instrucciones END, aunque está implícitamente supuesto un END después de que el BASIC haya obedecido la última línea de un programa.

END o fin de programa cierra todos los ficheros en cassette y hace que el BASIC regrese al Modo Directo.

La generación de sonido continúa después del END hasta que todas las colas de notas queden vacías.

Claves Asociadas

STOP

ENT

Fija la envolvente de Tono para las notas. **Comando**

Uso

Mientras se está produciendo un sonido es posible hacer que varíe su tono. La envolvente de tono define cómo ha de ser dicha variación.

Forma

ENT <número de envolvente>[,<lista de:<secciones de envolvente>]

El <número de envolvente> es una <expresión entera> cuyo valor absoluto debe estar en la banda 1..15, y especifica cuál de las envolventes de tono va a usarse. Si el <número de envolvente> es negativo, entonces la envolvente es repetitiva, haciendo que la nota se repita continuamente hasta su conclusión.

Se pueden dar hasta cinco <secciones de envolvente>. Cada una puede adoptar una de las dos formas siguientes:

<cuenta de pasos>,<altura del paso>,<tiempo del paso>
o bien: <período del tono>,<tiempo del paso>

La primera de las formas especifica un cambio incremental en relación con el valor corriente para el período de tono. Con la segunda forma se especifica un valor absoluto para el período de tono. Siendo:

<cuenta de pasos> da el número de pasos en la sección -una <expresión entera> en la banda 0..239.

<altura del paso> da la cuantía por la que ha de variarse el período de tono en cada paso de la sección -una <expresión entera> que produzca un valor en la banda -128..+127.

<tiempo del paso> da el tiempo que dura cada paso -una <expresión entera> especificando el tiempo en centésimas de segundo. La expresión debe producir un valor en la banda 0..255 (donde 0 se trata como 256!).

<período del tono> da un nuevo valor absoluto para el período del tono usado en la sección -una <expresión entera> que produce un valor en la banda 0..4095.

Notas

El comando SOUND fija el período de tono inicial, y puede especificar como parámetro una de las quince envolventes de tono. Si se especifica una envolvente que todavía no ha sido establecida mediante el comando ENT, o no se especifica ninguna envolvente, entonces el tono de la nota permanece constante mientras se produce el sonido.

Cuando se produce una nota con una cierta envolvente de tono, el primer paso de la primera sección se ejecuta inmediatamente. Si la primera sección es un cambio incremental, entonces el período de tono cuando el sonido comienza, corresponderá al período de tono inicial establecido en el comando SOUND **más** la <altura del tono>. Si la primera sección de la envolvente corresponde a un valor absoluto, entonces el período inicial de tono especificado en el comando SOUND es descartado.

Una envolvente de tono no tiene ningún efecto sobre la duración del sonido. Si todavía quedan pasos en la envolvente de tono cuando el sonido termina, simplemente se abandonan dichos pasos. Si la envolvente de tono acaba antes que el sonido emitido, entonces el período de tono retiene su valor final hasta que concluye la emisión del sonido.

Una envolvente de tono repetitiva (identificada por número negativo) volverá a comenzar cada vez que termina hasta que concluya la emisión del sonido.

Las expresiones en la envolvente de tono se evalúan cuando se ejecuta el comando y los resultados se conservan aparte para su uso futuro. Usar la envolvente de tono no hace que vuelva a re-ejecutarse el comando.

Cada vez que se establece una envolvente de tono dada, el valor previo que tuviera se pierde. Cambiar una envolvente mientras una nota que la usa está sonando, o está pendiente de emitir, producirá efectos indeterminados, pero posiblemente interesantes.

Especificar una envolvente con ninguna sección en ella, cancela cualquier valor previamente estipulado. Cualquier uso posterior de la envolvente será ignorado, y el tono no variará durante la emisión de la nota.

Claves Asociadas

ENV, SOUND

ENV

Fijar la envolvente de volumen para una nota. Comando

Uso

Mientras se está produciendo un sonido es posible hacer que varíe su volumen. Una envolvente de volumen define cómo ha de ser dicha variación.

Forma

ENV <número de envolvente>[,<lista de:<sección de envolvente>]

El <número de envolvente> es una <expresión entera> que produce un valor en la banda 1..15, y que identifica la envolvente de volumen que va a usarse.

Se pueden dar hasta cinco <secciones de envolvente>. Cada una puede tomar una de las dos formas siguientes:

<cuenta de pasos>,<altura del paso>,<tiempo del paso>
o bien: <envolvente prescrita>,<período de envolvente>

La primera de las formas especifica una sección en que el volumen varía incremental o absolutamente bajo control del programa, siendo los parámetros:

<cuenta de pasos> da el número de pasos en la sección -una <expresión entera> en la banda 0..127.

Una <cuenta de pasos> de cero hace que la sección sea una absoluta (valor absoiuto del volumen), mientras que si es distinto de 0 representa una sección incremental (volumen relativo al paso anterior).

<altura del paso> si la <cuenta de pasos> no es cero, la <altura del paso> da la cuantía en la que variará el volumen en cada paso sucesivo de la sección -una <expresión entera> en la banda -128..+127.

o bien: si la <cuenta de pasos> es cero, entonces la <altura del paso> representa el volumen fijado para dicho paso -es por tanto un valor absoluto, no relativo.

<tiempo del paso> especifica el tiempo que dura cada paso de la sección -una <expresión entera> que señala el tiempo en centésimas de segundo. Dicha expresión debe producir un valor en la banda 0..255 (siendo el 0 tratado como 256!).

La segunda de las formas de la instrucción especifica una sección de envolvente que va a ejecutarse bajo control directo de los circuitos procesadores de sonido, siendo:

<envolvente prescrita> es el valor a depositar dentro del registro de perfil de envolvente (registro 13).

<período de envolvente> es el valor a depositar dentro de los registros de período de envolvente (registros 11 y 12).

Las secciones con envolvente prescrita por los circuitos, no tienen asociada una duración determinada, de manera que se ejecuta inmediatamente después la sección de envolvente que venga detrás. Se sugiere que dicha sección siguiente tenga una duración de un tiempo apropiado. Si no existe la sección siguiente, se adopta un tiempo de 2 segundos.

Notas

El comando SOUND es el que establece el volumen inicial de una nota, y puede especificar como parámetros una de las quince posibles envolventes de volumen. Si se especifica una envolvente que todavía no ha sido definida en el programa, o no se especifica ninguna envolvente, entonces el volumen permanece constante a lo largo de toda la duración de la nota.

Cuando comienza un sonido que usa una cierta envolvente de volumen, se ejecuta inmediatamente el primer paso de la primera sección de dicha envolvente. Si la primera sección es un cambio incremental, entonces el volumen cuando el sonido arranca, será el volumen inicial fijado en el comando SOUND **más** la <altura del paso>. Si la primera sección es una sección de volumen absoluto, o prescrita por los circuitos, entonces el volumen inicial estipulado en el comando SOUND será descartado.

Fijando una <altura del paso> de cero, con una <cuenta de pasos> distinta de cero, se consigue mantener el valor actual para el volumen, durante toda la duración de la sección.

Las expresiones en las envolventes de volumen se evalúan cuando se ejecuta el comando ENV, y los resultados se conservan aparte para uso posterior. Usar la envolvente de volumen no hace que el comando vuelva a re-ejecutarse.

Cada vez que se establece una envolvente de volumen determinada, su valor previo se pierde. Cambiar una envolvente mientras el sonido que la usa está activo, o pendiente, producirá efectos indeterminados (pero posiblemente interesantes).

Especificar una envolvente con ninguna de sus secciones, cancela cualquier valor previo. Cualquier uso posterior de la envolvente será ignorado, y el volumen no variará.

Claves Asociadas

ENT, SOUND

EOF

Comprueba el Final de Fichero. **Función**

Uso

Para saber si al hacer la lectura de un fichero de entrada en cassette se ha alcanzado la posición de final de fichero. La función da el valor -1 (cierto) si se está en el final del fichero. Da el valor 0 (falso) en todas las otras situaciones.

Forma

```
EOF
```

Notas

Si no está abierto ningún fichero de entrada procedente del cassette, entonces la función da el valor cierto.

La función EOF puede exigir que se haga la lectura del siguiente bloque desde el cassette. Si dicha lectura se interrumpe pulsando **ESC**, el resultado de esta función será el valor cierto.

Claves Asociadas

OPENIN

ERASE

Elimina las tablas definidas en el programa. **Comando**

Uso

Cuando ya no se va a volver a requerir una tabla, puede usarse el comando ERASE para quitarla y dejar libre el espacio que ocupa en memoria para otros propósitos.

Forma

ERASE<lista de: <nombre variable>

siendo cada <nombre variable> el nombre de una tabla definida implícita o explícitamente, pero sin incluir el sufijo correspondiente a los subíndices.

Notas

Es un error intentar dejar libre una tabla mediante el comando ERASE, cuando no ha sido declarada (explícita o implícitamente) en alguna instrucción anterior.

Una vez que se ha dado el comando ERASE para eliminar una tabla, es posible volver a usar el mismo nombre, declararla de nuevo, con diferentes dimensiones.

Claves Asociadas

DIM

ERR
ERL

Código de Error y Línea de Error. **Funciones**

Uso

Estas funciones que pueden verse como variables del sistema y no usan argumentos, pueden usarse en las subrutinas para resarcimiento de errores, como medio de descubrir el código de error (ERR) y el número de línea (ERL) que se estaba ejecutando cuando se detectó o produjo el error.

Aviso

Si se usa ERL en expresiones relacionales, debe escribirse en la parte izquierda de dicha expresión, para que el BASIC pueda reconocer la parte de la derecha como un número de línea. Si se escribiera la expresión relacional al contrario, al efectuar una renumeración mediante RENUM no podría encontrar el número de línea, y por tanto lo ignoraría con las subsecuentes probables incoherencias.

Claves Asociadas

ON ERROR, ERROR

ERROR

Provoca un Error con la acción subsecuente. **Comando**

Uso

Simular un error con un código de error determinado. Dicho código de error puede ser uno de los ya usados y reconocidos por el BASIC, en cuyo caso la acción tomada es la misma que se tomaría si realmente dicho error hubiera sido detectado por el BASIC. Los códigos de error superiores a aquéllos reconocidos por el BASIC, pueden usarse en el programa para señalar y tratar los errores que el usuario considere.

Forma

ERROR<expresión entera>

Siendo la <expresión entera> una que al evaluarse produzca un número en la banda 1..255.

Notas

En el caso de que el BASIC intente mostrar un mensaje de error correspondiente a un código de error de los que él no reconoce, presentará el mensaje 'Unknown error', para indicar que es un 'error desconocido'.

Claves Asociadas

ON ERROR, ERR, ERL

EVERY

Suscitar la ejecución de una subrutina **cada** cierto tiempo. **Comando**

Uso

El sistema mantiene un reloj en tiempo real. El comando EVERY permite que en un programa BASIC se señalen subrutinas cuya ejecución ha de suscitarse a intervalos regulares de tiempo. Se dispone de cuatro temporizadores de demora, cada uno de los cuales puede tener una subrutina asociada con él.

Forma

EVERY<intervalo tiempo>[,<número temporizador>]GOSUB<número línea>

El <intervalo de tiempo> es una <expresión entera> que especifica cuánto tiempo tiene que transcurrir entre cada ejecución de la subrutina señalada. Dicho tiempo se mide en 1/50-avos de segundo.

El <número de temporizador> es una <expresión entera> que identifica cuál de los cuatro posibles temporizadores de demora va a usarse. La expresión debe producir un valor en la banda 0..3. Si se omite, se supone el temporizador 0.

Notas

Cuando ha transcurrido el tiempo señalado, se suscita automáticamente la ejecución de la subrutina, igual que si se hubiera incluido un comando GOSUB precisamente en ese punto del programa. Cuando termina la subrutina, usando una instrucción normal RETURN, el programa continúa la ejecución en el mismo punto en que se vió interrumpido.

Los temporizadores tienen diferentes prioridades de interrupción. El temporizador 3 tiene la máxima prioridad y el temporizador 0 la mínima. Véase la Sección 9 para mayor detalle del efecto de las prioridades.

Inmediatamente que el temporizador termina su cuenta, es repuesto al valor estipulado y la cuenta atrás para medir el siguiente intervalo de tiempo vuelve a comenzar.

Se pueden dar los comandos EVERY en cualquier momento, con lo que se prefija al valor inicial el temporizador asociado con el comando. Los temporizadores de demora son los mismos usados en el comando AFTER, de manera que un comando EVERY revoca cualquier comando previo AFTER para un mismo temporizador, y viceversa.

Claves Asociadas

AFTER, REMAIN

EXP

Exponencial. **Función**

Uso

Calcular el resultado de elevar el número e a una potencia dada. El número e es el número cuyo logaritmo natural es 1, y aproximadamente es 2.7182818. Esta función también se conoce como antilogaritmo.

Forma

EXP(<expresión numérica>)

Notas

EXP con números mayores que 88 causa un error de 'rebasamiento'.

EXP con números mucho menores que -88.7 hace que se alcance el límite por defecto, y por tanto produce el valor cero.

Claves Asociadas

LOG

FIX

Da la parte 'fija' de un número, haciéndolo entero. **Función**

Uso

Para truncar el valor dado y obtener un entero. Observa que con esto no se convierten números en coma flotante a la representación entera, sino que simplemente se quita cualquier parte fraccionaria que pudiera haber, redondeando el valor hacia cero para conseguir la parte entera.

Forma

`FIX(<expresión numérica>)`

Claves Asociadas

`CINT, INT, ROUND`

FOR

Bucle preconfinado. **Comando**

Uso

Ejecutar una serie de instrucciones un determinado número de veces, incrementando una variable de control entre un valor de comienzo y un valor de finalización.

Forma

FOR<variable simple>=<comienzo> TO <final>[STEP<incremento>]

La <variable simple> es la que controla el bucle FOR, y debe ser Entera o Real, y no puede ser un elemento de una tabla.

El <comienzo> es una <expresión numérica> que establece el valor inicial de la variable de control.

El <final> es una <expresión numérica> que da el valor límite para la variable de control.

El <incremento> es una <expresión numérica> que da el valor que se añade a la variable de control después de cada iteración o ronda. Si se omite, se supone un valor de 1. Observa que el <incremento> puede ser negativo.

Los resultados de los parámetros <comienzo>, <final> e <incremento> son forzados automáticamente a ser de la misma clase que la variable de control.

Notas

El comando FOR inicia un bucle preconfinado, que se emplea cuando el número de veces que ha de repetirse está previamente determinado. El final del bucle FOR está señalado por un comando NEXT.

Cuando se ejecuta el comando FOR, se evalúan los parámetros <comienzo>, <final> e <incremento>, y luego se asigna a la variable de control el valor de <comienzo>. Si dicho valor inicial es superior (véase posteriormente) al valor final, el bucle FOR se salta completamente; en caso contrario la ejecución entra en el bucle. Cuando se encuentra la correspondiente NEXT, se añade al valor de la variable de control el <incremento>, y la ejecución regresa al comienzo del bucle FOR a no ser que la variable de control esté ahora por encima del valor final.

Si el incremento es positivo (o se omite) el bucle FOR termina cuando la variable de control es mayor que el valor de <final>. Si el incremento es negativo, el bucle FOR termina cuando la variable de control está por debajo del valor de <final>.

El comando NEXT que se empareja a un FOR determinado, se establece estáticamente cuando se ejecuta por primera vez el FOR. Eso quiere decir que el comando NEXT que corresponde al FOR, depende del orden de las instrucciones en el programa, y es totalmente independiente del orden de ejecución. No es posible, por lo tanto, tener más de un NEXT asociado con un FOR determinado.

Los bucles FOR pueden ser anidados (incluidos completamente uno dentro de otro).

El valor de la variable de control queda definido una vez que ha terminado el bucle. Por lo tanto está permitido terminar un bucle FOR evitando la instrucción NEXT; y el valor de la variable de control queda sin cambiar.

Debe tenerse cuidado cuando se usan <incrementos> fraccionarios, dado que el efecto de redondeo puede significar que la variable de control nunca iguale exactamente al valor de <final>.

Claves Asociadas

NEXT, WHILE

FRE

Mide el espacio libre. **Función**

Uso

Examinar cuánta memoria libre permanece sin ser usada por el BASIC. Existen dos formas de la función, y una de ellas hace que el BASIC efectúe una 'recogida de basura' antes de decir el espacio libre.

Forma

FRE(<expresión numérica>)
FRE(<expresión literal>)

Notas

El valor del argumento en una función FRE no tiene ningún significado, por lo que FRE(0) y FRE("") son las formas recomendadas.

El BASIC mantiene ocupa parte de la memoria libre como zona de maniobra en que almacena las constantes literales. A medida que los valores literales cambian, el BASIC emplea nuevo espacio de memoria en lugar de aprovechar el espacio que ocupaba el literal anterior. Dichas áreas que van quedando libres dentro del área literal, no pueden ser reclamadas por el BASIC hasta que se efectúe una 'limpieza de basura'. En general, el BASIC no efectúa dicha operación de limpieza hasta que haya alcanzado el límite de la memoria disponible.

La función FRE(0) dice la cantidad de memoria que queda todavía por usar.

La función FRE("") obliga al BASIC a efectuar una 'recogida de basura' de manera que cualquier espacio desaprovechado dentro del área literal sea transferido al área disponible. El resultado de la función es posterior a la limpieza, y dice el máximo espacio libre disponible. Observa que la 'recogida de basura' puede emplear un tiempo apreciable.

GOSUB

Desvío de la ejecución hacia una subrutina. **Comando**

Uso

Para poner en marcha una subrutina determinada.

Forma

GOSUB<número línea>

Notas

Cuando se ejecuta una instrucción GOSUB el programa se 'desvía' hacia el número de línea citado en la instrucción, pero el BASIC recuerda el punto en que se produjo el desvío para que inmediatamente después de terminar la ejecución de la subrutina, al encontrarse la instrucción RETURN vuelva al punto inmediatamente posterior a donde se desvió.

Las subrutinas se terminan mediante un comando RETURN. Una subrutina puede contener más de un comando RETURN.

Las subrutinas pueden anidarse (incluirse unas dentro de otras) y pueden ser recursivas (citarse a ellas mismas), hasta un nivel de profundidad limitado únicamente por el tamaño del 'stack' (donde apila las direcciones de vuelta pendientes) que es suficientemente grande como para que se aniden subrutinas por encima de cincuenta veces. (Observa sin embargo, que también otros comandos y funciones BASIC requieren espacio de 'stack' para anotar direcciones de instrucciones pendientes, lo que puede afectar al límite de anidamiento en cualquier programa concreto).

Claves Asociadas

RETURN

GOTO

Saltar o bifurcar a un número de línea. Comando

Uso

Para efectuar un salto en la ejecución de un programa hasta un número de línea dado, y continuar la ejecución a partir de dicha línea.

Forma

GOTO<número línea>

HEX$

Dar la representación hexadecimal de un numeral. **Función**

Uso

Producir una constante de índole literal formada por cifras hexadecimales y representando el valor de la expresión dada.

Forma

HEX$(<expresión entera sin-signo>[,<anchura campo>])

La <expresión entera sin-signo> debe producir un valor en la banda -32768..65535. Dicho valor se trata como un número binario sin-signo de 16 bits, y es convertido a la representación hexadecimal equivalente.

La <anchura campo> opcional es una <expresión entera> que concreta la longitud mínima del literal hexadecimal que se produce, y debe ser un entero en la banda 0..16.

Notas

HEX$ siempre genera tantos caracteres como sean necesarios para representar el número (un 0 genera como mínimo una cifra hexadecimal). Si se especifica una <anchura campo> y la constante literal hexadecimal resultante es demasiado corta, se rellena hasta la longitud especificada incluyendo ceros delanteros. La constante literal resultante no se trunca si es demasiado larga.

Claves Asociadas

BIN$, DEC$, STR$

HIMEM

Máxima dirección de memoria usable por el BASIC. **Función**

Uso

La cantidad de memoria usable por el BASIC puede alterarse mediante el comando MEMORY. La función intrínseca HIMEM dice cuál es la máxima dirección que se usa en un programa concreto.

Forma

```
HIMEM
```

Notas

HIMEM da como resultado un valor Real en al banda 0..65535.

Claves Asociadas

MEMORY

IF

Establecer una premisa condicional. **Comando**

Uso

Para hacer que la ejecución de las instrucciones esté supeditada al resultado de una expresión.

Formas

IF <expresión logical> THEN <alternativa-1>[ELSE<alternativa-2>]
IF <expresión logical> GOTO <número línea>[ELSE<alternativa>]

Siendo las <alternativa>s bien: <serie de comandos>
o bien: <número línea>
y: <serie comandos>
es uno o más comandos consecutivos dentro de la misma línea que la partícula IF, y separados por dos-puntos.

Notas

La <expresión logical> se evalúa en primer lugar. Si el resultado es no-cero ('cierto'), se ejecuta la cláusula que sigue al THEN o al GOTO; en caso contrario se salta la ejecución de la cláusula THEN o GOTO y se llega hasta el final de la línea o hasta que se encuentre una cláusula precedida de ELSE que concuerde. En este último caso se ejecuta dicha cláusula que sigue a ELSE.

Las instrucciones condicionadas por IF pueden anidarse hasta cualquier nivel, limitado únicamente por la longitud de la línea de programa. Cada cláusula ELSE está asociada con la condición más interna del IF que no tenga todavía una cláusula ELSE.

THEN<número línea> y ELSE<número línea> son equivalentes a THEN GOTO<número línea> y ELSE GOTO<número línea>, respectivamente.

El comando IF se interpreta como terminado al llegar al final de la línea. No es posible tener más comandos independientes de la condición IF, dentro de la misma línea.

Claves Asociadas

WHILE

INK

Fijar el color que corresponde a una Tinta dada. Comando

Uso

Dependiendo del modo de pantalla vigente, se dispone de un cierto número de Tintas. El Color o la pareja de colores, asociada a una Tinta puede cambiarse mediante este comando INK.

Forma

INK <tinta>,<color>[,<color>]

Si se especifica el segundo color, entonces la tinta correspondiente alterna entre los dos colores dados.

Claves Asociadas

BORDER

INKEY

Comprobar el estado de una determinada tecla. **Función**

Uso

INKEY puede usarse para probar si una tecla dada está pulsada o no, y si lo está, si además están pulsadas las teclas **[SHIFT]** y **[CTRL].**

Forma

INKEY(<número tecla>)

El <número tecla> es una <expresión entera> que produce un valor en la banda 0..79, e indica cuál de las 80 teclas es la que ha de examinarse el estado.

Notas

La función da como resultado un valor entero con el significado siguiente:

	Tecla	[SHIFT]	[CTRL]
-1	Quitada	Desconocida	Desconocida
0	Pulsada	Quitada	Quitada
32	Pulsada	Pulsada	Quitada
128	Pulsada	Quitada	Pulsada
160	Pulsada	Pulsada	Pulsada

La función comprueba el estado de la tecla dada, tal y como quedó establecido en la exploración más reciente del teclado. La exploración del teclado se efectúa cada 1/50-avo de segundo.

El valor resultante es el más reciente estado conocido del interruptor correspondiente a esa tecla. Y es independiente de cualquier definición hecha para la tecla.

INKEY no afecta la generación normal de caracteres procedentes del teclado, ni al 'buffer' del teclado.

Claves Asociadas

INPUT, INKEY$

INKEY$

Recoger la tecla últimamente pulsada. **Función**

Uso

Indaga si hay alguna tecla pulsada en ese momento, o que haya sido pulsada en un momento anterior sin haber sido tratada. Observa que cualquiera que sea el carácter recogido del 'buffer' de entrada no se ve reflejado en la pantalla.

Forma

```
INKEY$
```

Notas

Si hay un carácter pendiente, entonces INKEY$ lo da como resultado (como constante literal de un solo carácter). Si no hay ningún carácter pendiente, el resultado de INKEY$ es la constante literal vacía.

Excepto para la tecla **ESC** (que detiene la ejecución del programa), todos los caracteres son pasados al programa tal y como se han recogido del teclado, sin reflejarlos en la consola.

INKEY$ no inspecciona el estado presente del teclado, sino que toma el último carácter, si lo hay, que esté pendiente de tratar en el 'buffer' de interrupciones del teclado.

Claves Asociadas

INPUT, INKEY

INP

Introduce el valor depositado en un portal de entrada a la máquina. **Función**

Uso

Recoger el valor que un periférico hubiera depositado en el portal de entrada dado.

Forma

`INP(` <número portal>)

El <número portal> es una <expresión direccional>.

Notas

`INP` hace la lectura de un byte procedente de un portal de entrada dado.

Claves Asociadas

`OUT, WAIT`

INPUT

Impone la información tomada de un cauce como valores de variables.

Comando

Uso

Para leer datos procedentes de un cauce de entrada dado. Produce un 'aviso' en la pantalla cuando el cauce de entrada es el teclado.

Formas

INPUT[#<expresión cauce>,][;][<aviso entrecomillado>;]<lista de:<variable>
INPUT[#<expresión cauce>,][;][<aviso entrecomillado>,]<lista de:<variable>

Si se omite la <expresión cauce> se usa el prescrito para omisiones que es el cauce #0, procedente del teclado.

El <aviso entrecomillado> es una constante literal entrecomillada.

Notas

1. Datos procedentes del teclado - Cauces #0..#8

Para omisiones del <aviso> está prescrito que se exponga un signo de interrogación. Si se da explícitamente un <aviso> se añade un signo de interrogación al final de él cuando se usa la primera de las formas dadas (con un punto-y-coma después del aviso), pero no cuando se utiliza la segunda de las formas (con una coma después del aviso). El aviso se envía por el cauce de salida cuyo número coincide con el cauce de entrada mencionado en la instrucción.

Cuando se ejecuta un INPUT, se lanza el <aviso> si lo hay, y se recoge la línea tecleada en la consola. La línea tecleada se trata como una lista de sucesivos elementos, y el BASIC intenta ir **imponiendo** cada uno de los valores tecleados a cada una de las variables dadas en la lista, comprobando que el dato tecleado es compatible con la variable que figura en la instrucción. La línea tecleada adopta la forma:

<lista de:<datos>

donde cada <dato> puede ser uno de los siguientes:

un <valor numérico>, que puede estar precedido y seguido de <espacio blanco>, que se descarta. Observa que un <valor numérico> puede también ser considerado como un <literal sin entrecomillar>.

un <literal entrecomillado>, que puede estar precedido y seguido de <espacio blanco>, que se ignora cuando precede o sigue a las comillas. Las comillas de cierre pueden omitirse si dicho literal es el último elemento tecleado en la línea.

un <literal sin entrecomillar>, que puede estar precedido y seguido de <espacio blanco>, que se descarta. Un <literal sin entrecomillar> puede tener una longitud entre 0 y 255 caracteres y estar terminado por una coma o por una marca de retorno de carro.

Si se teclea el número correcto de elementos en la línea, y sus clases son completamente compatibles con las variables correspondientes, entonces los datos tecleados en la línea quedan **impuestos** como valores de las variables.

Si hay pocos o demasiados datos en la línea, o si un dato no es compatible con la variable correspondiente, entonces se lanza el mensaje '?Redo from start', para preguntar si 'vuelve a hacerlo desde el comienzo', y se vuelve a ejecutar el comando INPUT desde el principio (de manera que el aviso, si lo hay, se repite).

Se puede incluir opcionalmente un punto-y-coma para impedir que el BASIC refleje el 'retorno de carro' en pantalla, que marca el final de la línea tecleada. Eso significa que el cursor permanece situado al final de la lista de datos que se ha ingresado por teclado.

2. Datos procedentes del cassette - Cauce #9.

No se genera ningún <aviso> para un INPUT procedente del cauce de entrada del cassette. Se puede especificar un <aviso> en la instrucción, pero se ignora totalmente -de manera que el mismo programa pueda leerse de una u otra clase de cauce. Igualmente se ignora el punto-y-coma opcional.

El BASIC intenta leer un dato del fichero para cada variable que aparece en la lista de la instrucción. Cada dato debe ser compatible con la variable sobre la que se va a imponer como valor. Cada dato del fichero puede ser uno de los siguientes:

un <valor numérico> que puede estar precedido por <espacio blanco> que se descarta. Un dato numérico está terminado al encontrarse con <espacio blanco>, coma, retorno de carro o fin de fichero. El <espacio blanco> posterior se descarta. Cualquier coma o retorno de carro posterior se ignora. Observa que un <valor numérico> puede también tratarse como un <literal sin entrecomillar>.

un <literal entrecomillado>, puede que esté precedido por <espacio blanco> que se ignora. Todos los caracteres después de las comillas delanteras, e incluyendo <espacio blanco>, 'retornos de carro' y 'avances de línea', hasta encontrarse las comillas posteriores son los que forman la 'retahíla de caracteres'. Estos literales se terminan mediante las comillas de cierre, o final de fichero. Después de las comillas posteriores, el posible <espacio blanco> se ignora. Cualquier coma o retorno de carro posterior se ignora.

un <literal sin entrecomillar> que puede estar precedido de <espacio blanco> que se ignora. Un dato que sea un <literal sin entrecomillar> está terminado por una coma, un 'retorno de carro' o un final de fichero.

(Un retorno de carro seguido de un avance de línea, y viceversa, se tratan como 'retorno de carro'. Las constantes literales están limitadas a 255 caracteres y terminan automáticamente después del 255-ésimo).

Claves Asociadas

LINE INPUT, READ, INKEY$

INSTR

Buscar si un literal dado está incluido en otro. **Función**

Uso

Para examinar a partir de un punto determinado una serie de caracteres dada, en busca de la primera aparición de otra serie de caracteres dada.

Forma

INSTR([<posición comienzo>,]<donde busca>,<lo que busca>)

La <posición comienzo> es una <expresión entera> que fija la posición dentro del literal donde busca, a partir de la cual empieza a comprobar si aparece el literal que busca. La expresión debe producir un valor en la banda 1..255. Si se omite, la búsqueda comienza a partir del primer carácter del literal donde busca.

El literal <donde busca> es una <expresión literal> que señala la serie de caracteres que van a ser examinados.

El literal <lo que busca> es una <expresión literal> que señala la serie de caracteres cuya pertenencia al otro literal se trata de comprobar.

Notas

Si se encuentra <lo que busca> la función da como resultado la posición en que aparece por primera vez <lo que busca> dentro del literal <donde busca>. Si no se encuentra la coincidencia, la función da el valor cero.

Si no hay ningún carácter en el literal <donde busca> a partir de la <posición comienzo> dada (o el valor tomado por omisión), entonces INSTR siempre entrega como resultado el valor cero.

Si <lo que busca> es el literal vacío, es como si lo encontrara inmediatamente a no ser que se aplique la regla anterior.

INT

Obtener el entero más próximo. **Función**

Uso

Para redondear un valor dado obteniendo el entero inferior más próximo a él (redondeando hacia menos infinito). Observa que esta función no convierte números Reales a representación Entera, sino que simplemente prescinde de la parte fraccionaria.

(Para números positivos es totalmente equivalente a FIX. Para números negativos el resultado de la función es una unidad menos que el resultado de FIX, a no ser que el argumento fuera ya un entero).

Forma

INT(<expresión numérica>)

Claves Asociadas

CINT, FIX, ROUND

JOY

Inspecciona el estado de un mando para juegos dado. **Función**

Uso

Se admiten en el sistema dos mandos para juegos. Esta función comprueba directamente el estado actual de un determinado mando para juegos.

Forma

JOY(<número del mando>)

El <número del mando> es una <expresión entera> que debe producir un valor en la banda 0..1, que señala cuál de los dos mandos para juegos posibles ha de ser inspeccionado.

Notas

La función JOY da como resultado un valor Entero que está 'calibrado en binario' con cada uno de los bits reflejando el estado presente de uno de los interruptores correspondientes del mando para juegos, en la forma siguiente:

Bit 0: **Arriba**
1: **Abajo**
2: **Izquierda**
3: **Derecha**
4: **Botón 2 de disparo**
5: **Botón 1 de disparo**

Ambos mandos para juegos se tratan en esencia como si fueran parte del teclado. El mando número 0 está completamente separado del teclado, mientras que el mando número 1 se 'proyecta' sobre un área del teclado principal. La función JOY es similar a la función INKEY excepto que con aquélla se sabe el estado de diversas 'teclas' con sólo una inspección.

La función comprueba el estado del mando para juegos tal y como ha quedado establecida por la última exploración del teclado. El teclado se explora cada 1/50-avo de segundo.

El resultado refleja el último estado conocido del mando para juegos. Es independiente de cualquier tecla y de las definiciones hechas para las teclas funcionales.

JOY no afecta la generación normal de caracteres procedente del teclado.

Claves Asociadas

INKEY

KEY

Adscribir un nuevo mensaje de entrada a una tecla funcional. **Comando**

Uso

Se admiten hasta treinta y dos 'códigos de entrada desarrollables' en el teclado. Cuando se pulsa una de estas teclas, la serie de caracteres **-el dicho-** asignado a ella, es la que se comunica en lugar del código de entrada correspondiente. El comando KEY asocia una determinada serie de caracteres con uno de estos códigos clave desarrollables.

Forma

KEY<código de entrada desarrollable>,<expresión literal>

El <código de entrada desarrollable> es una <expresión entera> que debe producir un valor en la banda 0..31, y que señala cuál de los códigos de entrada es el que se va a emplear.

La <expresión literal> se evalúa previamente y el resultado obtenido se deposita para reemplazar al código de entrada cuando se pulse la tecla o teclas correspondientes.

Notas

Los treinta y dos 'códigos desarrollables' corresponden a aquellos caracteres cuyos valores están en la banda 128..139. Cuando se pone en marcha la máquina, el teclado numérico separado está prescrito a los caracteres 128..139, que a su vez provocan el desarrollo en dígitos, punto y retorno de carro. La pulsación simultánea de **CTRL-ENTER** genera el carácter 140, que está prescrito para desarrollarse en la serie de caracteres RUN"<retorno de carro>. Los otros códigos desarrollables están sin utilizar, y tienen literales vacíos asociados con ellos. Está preparado el sistema para aceptar un total de 120 caracteres en todos los literales asociados con los códigos desarrollables.

Usando el comando KEY DEF, se puede convertir la pulsación de otras teclas en el equivalente a 'teclas funcionales' o programables.

Claves Asociadas

KEY DEF

KEY DEF

Definir los valores correspondientes a una tecla dada. **Comando**

Uso

Los valores generados cuando se pulsa una tecla, solamente ella o simultáneamente con otra, pueden redefinirse usando este comando. También permite establecer si la tecla es autorrepetitiva o no.

Forma

KEY DEF <número tecla>,<repite>[,<normal>[,<turnada>[,<control>]]]

El <número tecla> es una <expresión entera> que produce un valor en la banda 0..79, y que especifica la tecla cuyos valores van a redefinirse.

El parámetro <repite> es una <expresión entera> que produce un valor en la banda 0..1, y especifica si la tecla es autorrepetitiva (1) o si no lo es (0).

Los parámetros <normal>, <turnada> y <control>, cuando se especifican, son todos <expresiones enteras> que producen valores en la banda 0..255. Estos parámetros definen el valor que será generado cuando se pulse la tecla en las condiciones:

<normal> es el valor cuando no se han pulsado además ni **SHIFT** ni **CTRL.**

<turnada> es el valor cuando también está pulsada **SHIFT**, pero no **CTRL.**

<control> es el valor cuando también está pulsada CTRL.

Cuando no se especifica un valor nuevo para un estado de la tecla, se retiene el valor previo.

Notas

Los valores generados al pulsar las teclas, se interpretan como sigue:

0..31 Estos valores suelen producir efectos especiales al ser enviados al monitor para su exposición, pero se tratan como caracteres ordinarios cuando provienen del teclado, exceptuando a &0D(13) que es Retorno de Carro.

32..127 Caracteres ordinarios, habitualmente interpretados como ASCII.

128..159 Los 'códigos desarrollables' según una determinada serie de caracteres adscrita a ellos mediante el comando KEY.

160..223 Caracteres ordinarios.

224..254 Valores especiales usados para **ESC** y teclas de cursor en edición y copiado.

255 Ignorado.

No es aconsejable redefinir los valores de Retorno de Carro, **ESC** y las teclas de cursor, dado que puede afectar adversamente al comportamiento del BASIC.

Redefinir las teclas **SHIFT** y **CTRL** no tiene efectividad.

Claves Asociadas

KEY

LEFT$

Extraer la parte situada a la izquierda de un literal. **Función**

Uso

Para extraer un cierto número de caracteres tomándolos de la parte izquierda de un literal dado.

Forma

LEFT$(<expresión literal>,<longitud requerida>)

La <expresión literal> constituye el literal del que se extraen los caracteres pedidos.

La <longitud requerida> es una <expresión entera> que especifica cuántos caracteres han de extraerse, y debe producir un valor en la banda 0..255.

Notas

Si el literal dado es más corto que la <longitud requerida>, el resultado devuelto es todo el literal mencionado. En caso contrario, el resultado es la parte izquierda del literal, hasta el número de caracteres determinado por la <longitud requerida>.

Claves Asociadas

MID$, RIGHT$

LEN

Determina la longitud de un literal dado. **Función**

Uso

Para determinar el número de caracteres que componen un literal dado.

Forma

LEN(<expresión literal>)

Notas

LEN da como resultado un entero en la banda 0..255. El resultado 0 indica que el literal está vacío. Todos los caracteres del literal intervienen en la longitud, incluyendo cualquiera que pueda haber no-visivo.

LET

Asigna un valor a una variable. **Comando**

Uso

La palabra LET no se suele utilizar, y es realmente un recuerdo de las primeras versiones de BASIC.

Forma

[LET]<variable>=<expresión>

Notas

Se evalúa primeramente la expresión y el valor resultante queda asignado a la variable, a no ser que no sean totalmente compatibles.

La palabra clave LET es, de hecho, completamente redundante.

LINE INPUT

Introducir una línea completa procedente de un cauce de entrada. **Comando**

Uso

Para recoger una línea completa de texto introducida por un determinado cauce, y asignarla como valor de una variable literal. Produce un aviso en la pantalla cuando el cauce de entrada es el teclado.

Formas

```
LINE INPUT[#<expresión cauce>,][;][<aviso>;]<variable literal>
LINE INPUT[#<expresión cauce>,][;][<aviso>,]<variable literal>
```

Si se omite la <expresión cauce> se presupone que la entrada es por el cauce del teclado (cauce #0).

El <aviso> es opcional y debe ser una constante literal entrecomillada.

La <variable literal> especifica el nombre de la variable a la que se asignará la línea introducida.

Notas

1. Cauces de Teclado - Cauces #0..#8

Si no se especifica aviso, está prescrito el exponer un signo de interrogación. Si se especifica explícitamente un aviso, el signo de interrogación se añade al final de él cuando se usa la primera de las formas de la instrucción (con un punto-y-coma después del aviso); pero no si se utiliza la segunda de las formas (con una coma después del aviso). El aviso se expone a través del cauce de salida cuyo número coincide con el cauce de entrada por teclado especificado.

Cuando se ejecuta el comando LINE INPUT, se expone el aviso y se espera hasta que se haya introducido una línea por teclado, que luego queda asignada como valor de la variable. La línea se termina pulsando Retorno de Carro (o después de los 255 caracteres, lo que ocurra antes).

Si está presente en el comando el punto-y-coma opcional, entonces el BASIC no reflejará el retorno de carro pulsado al final de la línea introducida. Eso significa que el cursor permanece en el extremo del texto recién impuesto por el teclado.

2. Cauce de Entrada del Cassette - Cauce #9

No se genera ningún aviso cuando se usa LINE INPUT a partir del cauce de entrada del cassette.

Puede especificarse un aviso, pero se ignora -por lo tanto, el mismo programa puede imponer datos procedentes de una u otra clase de cauce. El punto-y-coma opcional se ignora.

Cuando se ejecuta el comando LINE INPUT, se recoge la línea procedente del cauce dado y se asigna como valor a la variable literal especificada. La línea se termina mediante retorno de carro (o después de 255 caracteres, lo que ocurra antes).

Claves Asociadas

READ, INPUT, INKEY$, INPUT$

LIST

Mostrar las líneas de un programa. **Comando**

Uso

Para presentar el programa en curso a través del cauce dado. Se puede listar parte del programa o todo él.

Forma

LIST[<banda números de línea>[,#<expresión cauce>]

La <banda números de línea> define las líneas de programa que van a ser listadas, incluyendo las cotas inferior y superior. Si se omite, se mostrarán todas las líneas de programa.

La <expresión cauce> define la vía por donde se enviará el listado producido. Si se omite, el listado se envía a la pantalla, cauce #0.

Notas

El BASIC regresa el Modo Directo después de obedecer un comando LIST.

Pulsando [**ESC**] se suspende temporalmente el listado, que puede reanudarse pulsando [**SPACE**]. La pulsación sucesiva [**ESC**][**ESC**] hace que el BASIC abandone el listado y regrese inmediatamente al Modo Directo.

LOAD

Cargar un programa en la memoria. **Comando**

Uso

Para traer un programa en BASIC desde el cassette y depositarlo en la memoria, reemplazando cualquier programa existente en ella; o para cargar un fichero binario en la memoria.

Forma

LOAD<nombre fichero>[,<expresión direccional>]

El <nombre fichero> es una <expresión literal> que da el nombre del fichero a partir del cual se efectúa la carga.

Si el fichero es un fichero binario, será cargado en memoria a partir de la dirección en que se escribió, a no ser que se dé el segundo parámetro que especifica la dirección a partir de la cual debe cargarse.

Notas

Si la <expresión literal> produce un literal vacío, entonces el BASIC intenta cargar el primer fichero que encuentre en la cinta.

Si el primer carácter de la <expresión literal> es '!', dicho carácter es quitado del nombre y tiene como efecto suprimir los mensajes habitualmente generados por la parte del BASIC encargada de hacer la lectura de cassette.

Si un programa en BASIC es cargado en memoria, cualquier programa existente en ella, así como las variables y funciones de usuario son suprimidas de la memoria. Los valores estipulados en instrucciones DEFINT, DEFREAL y DEFSTR, también quedan restaurados a sus condiciones originales.

Solamente pueden cargarse en memoria ficheros binarios cuando se depositan fuera de la zona de memoria activa del BASIC -por ejemplo, por encima de HIMEM.

Cualquier fichero en cassette se abandona -y cualquier salida que hubiera en el 'buffer' de un fichero se pierde.

Claves Asociadas

SAVE, RUN, CHAIN, CHAIN MERGE, MERGE

LOCATE

Sitúa el cursor en una posición dada. **Comando**

Uso

Para desplazar el cursor a una nueva posición.

Forma

LOCATE[#<expresión cauce>,]<coordenada x>,<coordenada y>

Si está presenta la <expresión cauce> debe producir un valor correspondiente a un cauce de pantalla. Si se omite, se supone cauce #0.

La <coordenada x> y la <coordenada y> son <expresiones enteras> que deben producir valores en la banda 1..255. La <coordenada x> da la nueva columna que corresponde al cursor; la <coordenada y> da el nuevo número de línea que corresponde a la posición final del cursor.

Notas

La nueva posición viene dada en forma relativa con respecto al origen de la ventana actual. La esquina superior izquierda de dicha ventana es la posición de coordenadas 1,1.

Si se desplaza el cursor fuera de la ventana actual, permanece ahí hasta que vuelve a moverse o hasta que se expone un carácter, para lo que se fuerza automáticamente a situarse dentro de la ventana. El cursor está quitado a no ser que el BASIC esté esperando la introducción de datos por el teclado, momento en el que es visible a no ser que lo haya impedido el usuario. El Apéndice IV contiene una descripción más completa de la colocación del cursor.

Claves Asociadas

WINDOW

LOG

Logaritmo natural. **Función**

Uso

Para calcular el logaritmo en base e de un número.

Forma

LOG(<expresión numérica>)

siendo el resultado de la <expresión numérica> un número siempre mayor que cero.

Claves Asociadas

EXP, LOG10

LOG10

Logaritmo decimal. **Función**

Uso

Para calcular logaritmos en base 10 de un número.

Forma

LOG10(<expresión numérica>)

siendo el resultado de la <expresión numérica> un número siempre mayor que cero.

Claves Asociadas

EXP, LOG

LOWER$

Convierte un literal a letras minúsculas ('caja baja'). **Función**

Uso

Para crear un nuevo literal que es copia de otro, con todos los caracteres alfabéticos mayúsculas convertidos a sus equivalentes en minúsculas.

Forma

LOWER$(<expresión literal>)

Notas

El resultado de la función es la misma <expresión literal> con todos los caracteres en la banda 'A'..'Z' convertidos al carácter equivalente en la banda 'a'..'z'.

Claves Asociadas

UPPER$

MAX

Determina el valor máximo de una serie de ellos. **Función**

Uso

Para conocer el máximo valor de una serie de datos.

Forma

MAX(<lista de:<expresión numérica>)

Notas

MAX devuelve el valor mayor de todos los reflejados como argumentos de la función.

Claves Asociadas

MIN

MEMORY

Restaura los parámetros de memoria para el BASIC. **Comando**

Uso

Para cambiar la cantidad de memoria utilizable por el BASIC.

Forma

MEMORY <expresión direccional>

La <expresión direccional> fija la dirección del byte más alto en la memoria que puede ser usado por el BASIC.

Notas

La función HIMEM devuelve el byte más alto presente en la memoria utilizable.

El comando MEMORY interactúa con el SYMBOL AFTER y con los ficheros en cassette, dado que éstos automáticamente reservan espacio para ellos en la parte superior de la memoria (véase Apéndice IX para una descripción más completa de esta interacción).

Claves Asociadas

HIMEM

MERGE

Congrega un programa procedente del cassette con el presente en memoria. **Comando**

Uso

Agregar el contenido de un fichero al programa existente en la memoria.

Forma

MERGE<nombre fichero>

El <nombre fichero> es una <expresión literal> que da el nombre del fichero que va a ser recogido del cassette y agregado al que hay en memoria.

Notas

Si la <expresión literal> produce un literal vacío, entonces el BASIC intenta traer del cassette el primer fichero que encuentre en la cinta.

Si el primer carácter de la <expresión literal> es la '!' será quitado del nombre del fichero y tendrá como efecto suprimir los mensajes habitualmente generados por la parte del BASIC que se ocupa de la lectura del cassette.

MERGE tiene los siguientes efectos sobre el entorno actual:

- Se descartan todas las variables y tablas.
- Se olvidan Todas las Funciones de Usuario.
- Se desactiva el posible ON ERROR GOTO.
- Se abandonan todos los ficheros abiertos -perdiéndose cualquier salida que hubiera en el 'buffer'.
- Se adopta la acción RESTORE.
- Se restaura lo establecido mediante DEFINT, DEFREAL y DEFSTR.

Durante una operación de MERGE, cualquier línea en el nuevo programa cuyo número coincida con una línea del programa existente en memoria, sustituirá a la línea en memoria.

El BASIC regresa al Modo Directo después de ejecutar un comando MERGE.

No es posible 'congregar' un programa protegido procedente del cassette.

Claves Asociadas

LOAD, CHAIN, CHAIN MERGE

MID$

Sustituye parte de un literal por otro. **Comando**
Devuelve la parte del medio de un literal. **Función**

Uso

MID$ especifica parte de un literal (un sub-literal) que puede usarse tanto como destino de una asignación (MID$ como un Comando), o como un argumento en una expresión literal (MID$ como una Función).

Forma

MID$(<literal>,<posición comienzo>[,<longitud sub-literal>])

Para MID$ como Comando, el <literal> debe ser un nombre de variable literal, parte de cuyo contenido es el que va a ser alterado.

Para MID$ como Función, el <literal> es una <expresión literal>, que será evaluada previamente y parte de la cual será devuelta como resultado de la función.

La <posición de comienzo> es una <expresión entera> que especifica la posición del carácter dentro del <literal> que va a ser el primer carácter del sub-literal. La expresión debe producir un valor en la banda 1..255.

La <longitud sub-literal> es una <expresión entera> que especifica la longitud que ha de tener el sub-literal segregado. Si se omite, el sub-literal se extiende hasta el final del <literal> original. La expresión debe producir un valor en la banda 1..255.

Notas

El sub-literal especificado en MID$ está definido por una posición de carácter para el comienzo y una longitud en caracteres. El primer carácter del literal original ocupa la posición 1. La longitud está prescrita para omisiones que corresponda a todos los caracteres situados después de la posición de comienzo del sub-literal. Si la posición de comienzo especificada está situada más allá del extremo del literal. El sub-literal correspondiente es el nulo. Si la longitud especificada no exige que el sub-literal acabe antes, terminará al alcanzar el final del literal original.

Cuando se usa MID$ como un Comando, aparece en la parte izquierda del signo igual de una asignación. En la parte derecha debe haber una <expresión literal>. El sub-literal especificado por MID$, queda sustituido por el valor de la <expresión literal>.

Si la <expresión literal> produce un valor con menos caracteres que los especificados para el sub-literal, entonces los caracteres extra en el sub-literal no se ven afectados. Si la <expresión literal> produce un valor que es más largo que el sub-literal, entonces se descartan los caracteres en exceso. El sub-literal que sufre el cambio no puede ser de longitud cero.

Claves Asociadas

LEFT$, RIGHT$

MIN

Determinar el valor mínimo de una serie de ellos. **Función**

Uso

Para saber el mínimo de una cierta serie de valores.

Forma

MIN(<lista de:<expresión numérica>)

Notas

MIN devuelve el valor más pequeño de las <expresiones numéricas> dadas.

Claves Asociadas

MAX

MODE

Fija el modo de gestión para la pantalla. **Comando**

Uso

Para cambiar la manera de operar con la pantalla del monitor.

Forma

MODE <expresión entera>

La expresión debe producir un valor entero en la banda 0..2.

Notas

Los modos de pantalla son como sigue:

0:	16 Colores.	25 líneas por 20 caracteres.	200 por 160 puntos.
1:	4 Colores.	25 líneas por 40 caracteres.	200 por 320 puntos.
2:	2 Colores.	25 líneas por 80 caracteres.	200 por 640 puntos.

La pantalla se deja en limpio con Tinta 0 (que no es necesariamente la Tinta de Papel actual). Todas las ventanas se restauran a pantalla entera; todos los cursores se reponen en la posición 1,1 (esquina superior izquierda). La ventana de gráficos se fija a pantalla entera, y el origen de gráficos y el cursor de gráficos se repone a la posición 0,0 (esquina inferior izquierda).

Claves Asociadas

WINDOW, ORIGIN

MOVE

Mover el Cursor de Gráficos - Coordenadas Absolutas. **Comando**

Uso

Desplazar el cursor de gráficos hasta la posición señalada por coordenadas absolutas.

Forma

FORMA<coordenada x>,<coordenada y>

La <coordenada x> y la <coordenada y> deben ser <expresiones enteras>.

Claves Asociadas

MOVER, PLOT, PLOTR, DRAW, DRAWR, TEST, TESTR

MOVER

Mover el Cursor de Gráficos - Coordenadas Relativas. **Comando**

Uso

Desplazar el cursor de gráficos hasta una posición especificada por coordenadas relativas con respecto a la posición actual.

Forma

MOVER<desplazamiento x>,<desplazamiento y>

El <desplazamiento x> y el <desplazamiento y> deben ser <expresiones enteras>.

Notas

Se desplaza el cursor de gráficos hasta la posición cuyas coordenadas absolutas son el resultado de sumar el <desplazamiento x> a la coordenada x actual, y el <desplazamiento y> a la coordenada y actual.

Claves Asociadas

MOVE, PLOT, PLOTR, DRAW, DRAWR, TEST, TESTR

NEW

Prepararse para un nuevo programa. **Comando**

Uso

Antes de comenzar un nuevo programa, se puede usar NEW para quitar completamente el contenido presente de la memoria.

Forma

NEW

Notas

Además de suprimir el programa existente en memoria, se toman las siguientes acciones:

- Se descartan todas las variables y tablas.
- Se olvidan Todas las Funciones de Usuario.
- Se desactiva el posible ON ERROR GOTO.
- Se abandonan todos los ficheros -perdiéndose cualquier salida en el 'buffer'.
- Se restaura lo estipulado por DEFINT, DEFREAL y DEFSTR.
- Se desactiva el rastreo puesto por TRON.

El BASIC siempre regresa al Modo Directo después de haber ejecutado un NEW.

NEXT

Incrementa la variable de control FOR al valor siguiente. **Comando**

Uso

Delimita el extremo de un bucle FOR. El comando NEXT puede ser anónimo, o puede mencionar al FOR correspondiente.

Forma

NEXT[<lista de:<variable>]

siendo: NEXT<variable>,<lista de:<variable>
equivalente a: NEXT<variable>:NEXT<lista de:<variable>

Notas

El comando NEXT marca el final de un bucle FOR. La manera en que FOR y NEXT están vinculadas conjuntamente, se describe en el comando FOR. Cuando se encuentra un comando NEXT, el BASIC sabe que debe haber asociado con él, y previamente un FOR. Si el NEXT va seguido de un nombre de variable, dicho nombre debe ser el mismo de la variable de control en el comando FOR correspondiente.

Claves Asociadas

FOR

ON ‹expression› GOSUB
ON ‹expression› GOTO

GOSUB y GOTO dependiente de un valor computado. **Comando**

Uso

Para elegir una entre varias subrutinas a las que desviarse, o entre varias líneas a las que saltar; dependiendo del resultado de una expresión.

Formas

ON<selector>GOSUB<lista de:<número línea>
ON<selector>GOTO<lista de:<número línea>

El <selector> es una <expresión entera> que debe producir un valor en la banda 0..255.

Notas

El valor del <selector> determina un número de línea de la lista dada. El valor 1 corresponde al primer número, el 2 al segundo, y así sucesivamente. La subrutina que comienza a partir del número de línea elegido es la que pasa a ser ejecutada cuando se usa GOSUB; o se salta al <número línea> elegido cuando se usa GOTO. Un <selector> de valor 0, o cualquier resultado mayor que la cantidad de <números de línea> que haya en la lista, hará que no se ejecute esta instrucción.

Sí es legal reflejar números de línea nulos en la lista. Elegir un elemento nulo en la lista no es un (Error 2).

Claves Asociadas

GOTO, GOSUB

ON BREAK GOSUB

Permitir las subrutinas suscitadas por interrupción de corte. **Comando**

Uso

Cuando se pulsa sucesivamente **ESC/ESC**, mientras el BASIC está ejecutando un programa, se suspende la ejecución, y el BASIC regresa la Modo Directo, indicando que se ha producido un corte ('Break') en la ejecución. El ON BREAK GOSUB permite que en lugar del mensaje correspondiente, el BASIC pase a ejecutar la subrutina citada en el comando, cuando se produce una interrupción de esta clase.

Forma

ON BREAK GOSUB<número línea>

Notas

La interrupción ON BREAK tiene la máxima prioridad de todas las interrupciones. Véase la Sección 9 para una descripción de la estructura de prioridades.

El comando RETURN puede usarse para terminar la subrutina de la manera habitual.

Si la subrutina de interrupción desea suspender la ejecución del programa, puede usar el comando STOP.

La subrutina de interrupción permanece 'facultada para actuar' hasta que se ejecuta un comando ON BREAK STOP.

Cada comando ON BREAK GOSUB revoca lo estipulado previamente mediante ON BREAK GOSUB o mediante ON BREAK STOP, y producirá su efecto al ocurrir la siguiente interrupción **ESC/ESC**.

Claves Asociadas

ON BREAK STOP

ON BREAK STOP

Cancela la detección de interrupciones en la ejecución. **Comando**

Uso

Quita cualquier 'cepo' establecido para detectar un corte en la ejecución producido al pulsar **ESC/ESC,** y establecido mediante el comando ON BREAK GOSUB.

Forma

ON BREAK STOP

Notas

Se puede dar el comando ON BREAK STOP dentro de la subrutina para resarcimiento de interrupciones señalada en el comando ON BREAK GOSUB. Con eso se desactiva el 'cepo detector', pero no tiene ningún otro efecto inmediato. Debe usarse el comando STOP si se desea que en la subrutina se pare el programa.

Claves Asociadas

ON BREAK GOSUB

ON ERROR GOTO

Establecer un Cepo para detectar errores. Comando

Uso

Cuando el BASIC detecta un error mientras está siguiendo un programa, puede pasar a ejercer la acción prescrita, que es generar un mensaje de error y regresar al Modo Directo; o puede hacerse que pase a efectuar una subrutina para resarcimiento de errores, incorporada en el propio programa mediante el comando ON ERROR GOTO.

Formas

```
ON ERROR GOTO<número línea>
ON ERROR GOTO 0
```

El <número línea> especifica la línea a la que será transferido el control en el caso de detectarse un error.

Notas

ON ERROR GOTO<número línea> faculta, habilita la detección de errores, y señala la línea a ejecutar en dicho suceso. ON ERROR GOTO 0 cancela la detección de errores -observa el efecto especial de esto durante el procesamiento de la rutina para resarcirse de errores, tal y como se explica a continuación.

Si ocurre un error en el Modo Programa estando facultada la detección de errores, la ejecución prosigue a partir del <número línea> señalado en el comando ON ERROR GOTO. El BASIC está entonces en el llamado Modo de Resarcimiento de errores. Las funciones ERR y ERL indican qué clase de error se ha producido, y en qué línea de programa ha ocurrido, respectivamente.

ON ERROR GOTO 0 estando en el Modo de Resarcimiento de errores, no sólo desactiva la detección posterior de errores, sino que también provoca que el BASIC adopte la acción prescrita para el error que se ha producido. Las rutinas para resarcimiento de errores pueden, por tanto, tratar sólo aquellos errores para las que están previstas y dejar que para el resto de ellos el BASIC tome la acción prescrita.

Si se detecta un error cuando ya se ha pasado al Modo de Resarcimiento de errores, el BASIC adopta inmediatamente la acción prescrita, como si no hubiera ningún 'cepo detector' establecido.

El comando RESUME sólo es legal durante el Modo de Resarcimiento de errores, y permite que el programa 'reanude' la ejecución en una de las tres formas siguientes:

- en la instrucción en la que se produjo el error
- en la instrucción siguiente a aquélla en la que se produjo el error
- en la instrucción señalada por un número de línea específico.

Claves Asociadas

ERR, ERL, RESUME

ON SQ GOSUB

Establece el Cepo para detectar vacíos en la secuencia de notas a emitir. Comando

Uso

Atender las señales de interrupción enviadas por los circuitos generadores de sonido cuando detectan un sitio libre en una determinada secuencia de notas que está esperando ser emitida.

Forma

ON SQ(<canal>) GOSUB<número línea>

El <canal> es una <expresión entera> que produce uno de los valores siguientes:

1: para canal A
2: para canal B
4: para canal C

Notas

Cada uno de los tres canales de sonido tiene circuitos separados para enviar señales de interrupción, aunque los tres tienen la misma prioridad -véase la Sección 9 para una explicación de prioridades.

La subrutina señalada para interrupciones ON SQ, pasa a ser ejecutada cuando la secuencia de notas del canal dado no está completa -la interrupción puede ocurrir inmediatamente si está facultada la detección y si la secuencia no está completa en ese momento.

La recepción de interrupciones queda cancelada cuando se produce una interrupción, por lo que la subrutina que trata estas interrupciones debe incluir dentro de sí misma un comando ON SQ, si se quiere que el canal continúe viendo atendidas las interrupciones que provoque posteriormente.

El comando SOUND y la función SQ también cancelan el 'cepo detector' de interrupciones establecido mediante el comando ON SQ.

Una subrutina ON SQ termina con un comando RETURN en la manera habitual.

Claves Asociadas

SOUND, SQ

OPENIN

Abre un fichero de entrada en cassette. **Comando**

Uso

Para prepararse a recoger datos de un fichero en cinta.

Forma

`OPENIN<nombre fichero>`

siendo <nombre fichero> una <expresión literal> que da el nombre con que está registrado el fichero en el cassette, y del cual sólo los dieciséis caracteres primeros son los significativos.

Notas

Si el primer carácter de la <expresión literal> es '!', dicho carácter será quitado del nombre del fichero a abrir en el cassette, y tiene el efecto de suprimir los mensajes que habitualmente genera la parte del BASIC encargada de la lectura del cassette.

Cuando se obedece el comando `OPENIN` se busca en la cinta el fichero mencionado, y el primer bloque es transferido de la cinta a la memoria y preparado para su tratamiento.

Claves Asociadas

`CLOSEIN, OPENOUT`

OPENOUT

Abrir un fichero de salida en cassette. Comando

Uso

Para preparar la salida de datos hacia un fichero en cinta.

Forma

OPENOUT<nombre fichero>

siendo <nombre fichero> una <expresión literal> que da el nombre con que será registrado el fichero en la cinta, y del cual sólo los dieciséis primeros caracteres son significativos.

Notas

Si el primer carácter de la <expresión literal> es '!', dicho carácter será quitado del nombre del fichero a abrir en el cassette, y tiene el efecto de suprimir los mensajes que habitualmente genera la parte del BASIC encargada de la lectura del cassette.

La salida hacia el cassette crea un nuevo fichero en la cinta con el nombre dado en el comando. Los ficheros en cassette están organizados en bloques de 2K bytes cada uno. Nada se escribe en la cinta hasta que no esté preparado un bloque completo en el 'buffer' de salida preparado en la memoria, o hasta que el fichero se cierre.

Claves Asociadas

CLOSEOUT, OPENIN

ORIGIN

Estipula el origen y la ventana de la pantalla de gráficos. **Comando**

Uso

Se puede desplazar la posición base 0,0 en el sistema de coordenadas para gráficos. Puede asimismo restringirse la zona de pantalla -la ventana- usada para gráficos.

Forma

ORIGIN <x0>,<y0>[,<izquierda>,<derecha>,<arriba>,<abajo>]

Las coordenadas <x0> e <y0> del nuevo origen, deben ser <expresiones enteras>.

Los parámetros <izquierda>, <derecha>, <arriba> y <abajo> especifican los bordes de la ventana para gráficos. Si se omiten, la ventana actual permanece sin verse afectada.

Notas

Las coordenadas del nuevo origen vienen dadas como posiciones absolutas de pantalla, donde la esquina inferior izquierda es la posición de coordenadas 0,0 y las coordenadas crecen hacia la derecha y hacia arriba.

Cada parámetro de ventana, especifica la última posición de la ventana en la dirección dada que queda incluida dentro de la ventana. Estas posiciones han de darse también en coordenadas absolutas.

Se puede invertir el orden de los parámetros <izquierda> y <derecha> (dado que siempre <derecha> debe ser mayor que, o igual a <izquierda>).

Se puede invertir el orden de los parámetros <arriba> y <abajo> (dado que <arriba> debe ser mayor, o igual a <abajo>).

Se cualquiera de los parámetros de la ventana indican posiciones fuera de la pantalla, se toman entonces los valores correspondientes a la posición extrema en la dirección dada.

El parámetro <izquierda> se trunca para que sea un múltiplo de 8. El parámetro <derecha> se fuerza para que sea el múltiplo más cercano de 8, menos uno. (Esto hace que la pantalla de gráficos tenga un número completo de bytes a lo ancho, lo que permite diversas optimizaciones en el programa que maneja la pantalla).

El parámetro <abajo> se trunca para que sea múltiplo de 2. El parámetro <arriba> se fuerza para que sea el múltiplo más próximo de 2, menos uno. (Con eso se logra que la ventana de gráficos tenga un número entero de puntos de imagen a lo alto).

Claves Asociadas

WINDOW

OUT

Saca un valor hacia el portal de salida de la máquina. **Comando**

Uso

Para enviar un byte a un determinado portal de salida.

Forma

OUT<número portal>,<expresión entera>

El <número portal> es una <expresión direccional> que identifica una de las posibles vías de salida de datos hacia el exterior.

La <expresión entera> da el byte que va a ser depositado en el portal de salida dado, y ha de producir un valor en la banda 0..255.

Claves Asociadas

INP, WAIT

PAPER

Estipula la tinta a usar para el fondo de los textos e imágenes. **Comando**

Uso

Cuando se exponen caracteres en pantalla, la posición que ocupa el carácter se rellena con el Tinte del Papel antes de mostrar el carácter -a no ser que se haya seleccionado el modo Transparente. El comando PAPER especifica el Tinte que va a usarse.

Forma

PAPER[#<expresión cauce>,]<tinta cribada>

Si está presenta la <expresión cauce> debe producir el valor correspondiente a un cauce de pantalla. Si se omite, se supone el cauce #0.

Notas

Sólo en el Modo 0 se admiten dieciséis tintas. En los otros modos, el valor dado para el parámetro <tinta cribada> es tratado interna y automáticamente para que produzca un número de Tinta apropiado y más pequeño.

Claves Asociadas

PEN

PEEK

Examina y dice lo que hay en una celdilla de memoria. **Función**

Uso

Para observar el contenido de un byte determinado de la memoria de la máquina.

Forma

PEEK(<expresión direccional>)

Notas

La función PEEK entrega como resultado un valor en la banda 0..255, en base 10.

PEEK hace que el BASIC examine la memoria de escritura/lectura de la máquina, por lo que puede observarse el contenido de la memoria, situada por debajo de la 'ROM de la parte alta' y la memoria por encima de la 'ROM de la parte baja'.

Claves Asociadas

POKE

PEN

Estipular la Tinta usada para el frente de los caracteres e imágenes.

Comando

Uso

Identificar la Tinta que va a usarse al exponer caracteres o dibujar figuras en la pantalla.

Forma

PEN[#<expresión cauce>,]<tinta cribada>

Si está presente la <expresión cauce> debe producir un valor correspondiente a un cauce de pantalla. Si se omite, se supone el cauce #0.

Notas

Sólo en el Modo 0 se admiten dieciséis Tintas. En los otros modos, el valor dado para <tinta cribada> se trata interna y automáticamente para producir un número de tinta más pequeño y apropiado.

Claves Asociads

PAPER

PI

Obtener el valor del número trascendente π. **Función**

Uso

Conseguir la representación en la máquina del valor de pi.

Forma

```
PI
```

Notas

Pi es aproximadamente 3.1415926535898. La representación interna en máquina más cercana es aproximadamente 3.1415926534683.

Claves Asociadas

SIN, COS, TAN, ATN, DEG, RAD

PLOT

Pinta un punto en la Pantalla de Gráficos. Comando

Uso

Para mostrar un punto de color en una determinada posición de la pantalla dada en coordenadas absolutas, y con un tamaño en puntos dependiente del modo de pantalla en que se opere.

Forma

PLOT <coordenada x>,<coordenada y>[,<tinta cribada>]

La <coordenada x> y la <coordenada y> deben ser <expresiones enteras>.

Si no se especifica ninguna tinta se usará la más recientemente usada en un comando DRAW, DRAWR, PLOT o PLOTR. Si este es el primero de tales comandos se usará la tinta 1.

Notas

Se ilumina el punto situada en la posición absoluta especificada. El cursor de gráficos se queda en esa posición.

Claves Asociadas

DRAW, DRAWR, PLOTR, MOVE, MOVER, TEST, TESTR

PLOTR

Para pintar un punto en la Pantalla de Gráficos. **Comando**

Uso

Para mostrar un punto de color en la posición especificada relativa a la posición corriente del cursor de gráficos, y del tamaño en puntos dependiente del modo de gráficos empleado.

Forma

PLOTR<desplazamiento x>,<desplazamiento y>[,<tinta cribada>]

El <desplazamiento x> y el <desplazamiento y> deben ser <expresiones enteras>.

Si no se especifica ninguna tinta, se usará la más recientemente empleada en un comando DRAW, DRAWR, PLOT o PLOTR. Si es el primero de tales comandos, se usará la tinta 1.

Notas

El punto se ilumina en la posición obtenida añadiendo el <desplazamiento x> a la coordenada actual x, y añadiendo el <desplazamiento y> a la coordenada y actual. El cursor de gráficos se queda en la posición de destino.

Claves Asociadas

DRAW, DRAWR, PLOT, MOVE, MOVER, TEST, TESTR

POKE

Meter un valor en la memoria de la máquina. **Comando**

Uso

Permite el acceso directo a la memoria de la máquina para escribir un byte.

Forma

POKE<expresión direccional>,<expresión entera>

La <expresión direccional> da la dirección de la celdilla de memoria en que se va a escribir.

La <expresión entera> da el valor que va a escribirse en la dirección dada, y debe producir un valor en la banda 0..255.

Claves Asociadas

PEEK

POS

Posición horizontal del cursor en un cauce dado. **Función**

Uso

Para observar la posición horizontal que ocupa en ese momento el cursor correspondiente a un cauce de salida dado.

Forma

POS(#<expresión cauce>)

Notas

1. Cauces de Pantalla - Cauces #0-#7

POS devuelve la coordenada 'x' de la posición actual del cursor. Observa que es un valor relativo al presente origen de ventana. La esquina superior izquierda de una ventana es la posición 1,1.

2. Cauce de Impresora - Cauce #8

Devuelve la posición corriente del cabezal de la impresora. El borde de la parte izquierda de la impresora es la posición 1. El BASIC cuenta todos los caracteres cuyos valores son mayores de &1F(31) para dar el valor de POS.

3. Cauce de Salida hasta el Cassette - Cauce #9

El BASIC mantiene la posición de un cursor lógico en los cauces de fichero, contando todos los caracteres visivos enviados a través de dicho cauce desde el envío del último retorno de carro. Los caracteres visivos son aquéllos cuyos valores son mayores de &1F(31). La posición inmediatamente detrás del envío de un retorno de carro es la posición 1.

Claves Asociadas

VPOS

PRINT

Enviar datos a través de un cauce de salida. **Comando**

Uso

Para sacar datos, numéricos o literales, a través de un cauce dado.

Forma

PRINT[#<expresión cauce>,][<lista a sacar>][<cláusula usando>][<separador>]

La <expresión cauce> da el cauce por el que se van a exponer o sacar los datos. Si se omite, se supone el cauce #0.

<lista a sacar> es: <dato a sacar>[<separador><dato a sacar>]*
siendo <dato a sacar> : <expresión>
o bien: SPC(<expresión entera>)
o bien: TAB(<expresión entera>)

la <cláusula usando> es: USING<literal conformador>;<lista a conformar>
siendo <lista a conformar>: <expresión>[<separador><expresión>]*

y donde <separador> es: coma o punto-y-coma.

(Observa que la construcción metalingüista [..]* significa que dicho objeto es opcional, pero puede repetirse cualquier número de veces).

1. Efecto:

Los <datos a sacar>, si los hay dentro de la <lista a sacar>, se evalúan y sacan en 'formato libre'.

Si está presente una <cláusula usando>, entonces el <literal conformador> define una 'horma' o 'plantilla' que controla el formato en que aparecerán los valores de las expresiones incluidas en la <lista a conformar>.

Cuando se hayan procesado todos los argumentos, se enviará un retorno de carro, a no ser que el comando PRINT termine en un <separador>, TAB o SPC.

2. Exposición de datos en Formato Libre:

Los <datos a sacar> se evalúan sucesivamente y se van sacando los resultados obtenidos por el cauce especificado. Los resultados numéricos se exponen como se describe a continuación. Los resultados literales se envían a través del cauce carácter por carácter. Las opciones SPC y TAB se describen a continuación.

Una coma que siga a un <dato a sacar> hace que el BASIC avance con espacios en blanco hasta el comienzo de la siguiente Zona de exposición, antes de procesar la siguiente expresión. Un punto-y-coma simplemente separa las expresiones, sin incluir ningún espacio en blanco entre ellas.

2.1 Zonas de Exposición y Anchura del Cauce

El BASIC divide cada línea de la pantalla en Zonas, cuya anchura está fijada por el comando ZONE, y para omisiones están prescritas que sean de 13 caracteres de anchura. Cuando PRINT avanza con espacios en blanco hasta la siguiente zona de exposición, el BASIC comenzará a partir de una nueva línea si quedan menos caracteres de la 'anchura de zona' hasta el borde de la pantalla (o del cauce correspondiente).

Antes de exponer el resultado de una expresión, el BASIC comprueba que hay suficiente sitio para ella entre la posición actual y el borde del cauce, a no ser que la posición actual sea el comienzo de una línea. Si no hay suficiente sitio, se comenzará con una nueva línea, antes de sacar el resultado a la pantalla, o al cauce correspondiente.

El 'borde de un cauce' depende de la clase de cauce de salida:

Cauces de Pantalla:	El borde es el borde de la ventana actual
Cauce de Impresora:	El borde está definido por el comando WIDTH. La posición en la impresora se puede saber mediante la función POS.
Cauce de Cassette:	No existe ningún borde que lo limite.

2.2 Exposición de Números en Formato Libre

Los números positivos van precedidos de un espacio en blanco; los negativos por un signo menos. Todos los números van seguidos de un espacio en blanco.

Todos los números se exponen con el mínimo de caracteres. No se saca ningún punto decimal si no hay ningún dígito fraccionario significativo. Como mínimo, se saca un dígito antes de cualquier punto decimal -así que los números fraccionarios pueden incluir un cero delante del punto.

Los números reales se exponen según 'formato escalado' (con un exponente) si no pueden representarse exactamente mediante un número con nuevo o menos dígitos (excluyendo un posible cero delantero). El formato escalado que se emplea da sólo seis dígitos significativos, y si se requiere una mayor precisión debe usarse la cláusula USING.

2.3 Función SPC para el comando PRINT

Uso

SPC avanza un determinado número de espacios, antes de exponer el siguiente dato.

Forma

SPC(<expresión entera>)

Si la <expresión entera> produce un valor negativo, se supone un cero. Si el valor es mayor que la anchura del cauce, es reducido a la banda 1..<anchura cauce> restando un número adecuado de veces dicha anchura de cauce. El valor resultante da el número de espacios en blanco que ha de avanzarse el cursor, cabezal de impresión o de lectura, según el cauce especificado.

Notas

Se saca el número de espacios dado, comenzando incondicionalmente a partir de la posición actual. SPACE$ no es exactamente equivalente, dado que el BASIC comprobaría que la 'serie de blancos' encaja en la línea actual, y posiblemente enviaría un retorno de carro antes de los espacios en blanco señalados por SPACE$.

SPC no necesita ir seguida de una coma o de un punto-y-coma, ya que siempre se supone que hay un punto-y-coma detrás (incluyendo el caso en que SPC es el último dato del comando PRINT).

2.4 Función TAB en el comando PRINT

Uso

TAB saca los espacios en blanco suficientes para pasar a una determinada posición del cursor, cabezal de impresión o cabezal de lectura, según el cauce especificado.

Forma

TAB(<expresión entera>)

Si la <expresión entera> produce un valor menor que uno, se supone el uno. Si el valor es mayor que la anchura de cauce, se reduce a la banda 1..<anchura cauce> restando las veces que sean necesarias la anchura de cauce. El resultado da la posición del cursor o cabezal a la que tiene que desplazarse.

Notas

Si la posición de exposición requerida es mayor que, o igual a la posición actual, se exponen espacios en blanco hasta que se alcanza la posición requerida (y eso puede hacer que no se exponga nada en absoluto).

Si la posición requerida para exponer el dato, es menor que la posición actual del cursor o cabezal, el BASIC lanza un retorno de carro seguido de los espacios necesarios para alcanzar la posición requerida en una nueva línea.

TAB no necesita ir seguida de una coma o de un punto-y-coma, ya que se supone siempre un punto-y-coma (incluyendo cuando TAB es el último elemento del comando PRINT).

3. Usando Exposición de datos con Formato

Las expresiones en la <lista a conformar> se evalúan una a una sucesivamente, y se exponen a través del cauce de acuerdo con la 'horma' o 'plantilla' de conformación. Después de que se ha evaluado cada expresión, se procesa la 'plantilla' hasta encontrar una especificación adecuada de formato para dicho resultado. Si el <literal conformador> se ha quedado vacío mientras se examina en busca de una especificación de formato, la búsqueda vuelve a comenzar de nuevo a partir del inicio del <literal conformador>. Sólo si no se encuentra un formato adecuado, se genera un error (Error 5).

Observa que de distinta manera que en la exposición en 'formato libre', la exposición de datos USING formato, no efectúa separación de zonas de exposición ni comprobación de bordes de cauce.

3.1 Plantilla conformadora (también llamada 'máscara de formato')

La 'horma' o plantilla conformadora es un literal que el BASIC interpreta carácter a carácter para controlar la manera en que se expone el resultado de cada expresión, por el cauce de pantalla, impresora o grabadora especificado. El significado de los caracteres en la plantilla se describen más adelante. Se aceptan y reconocen los siguientes caracteres en las especificaciones de la plantilla conformadora:

! \ & # . + - * $ ↑ , _

Cualquier otro carácter que el BASIC encuentre al examinar la plantilla conformadora, será sacado literalmente. Si cualquiera de los caracteres arriba mencionados, aparece 'fuera de contexto' también será sacado por el cauce. El carácter '_' no se expondrá, sino que hace que el carácter que le sigue aparezca en pantalla 'tal cual es'.

3.2 Especificaciones de Formato para datos Literales

'!' Sólo el primer carácter del literal es el que se saca.

'\<espacios>\'
Se sacarán los primeros 'n' caracteres del dato literal, siendo 'n' la longitud en caracteres de \<espacios>\ (habiendo por tanto 'n-2' espacios en blanco entre las barras inclinadas invertidas.

'&' El dato literal completo se saca 'tal cual es'.

3.3 Especificaciones de Formato para datos Numerales

Los números pueden exponerse por pantalla o impresora, o grabarse en cassette, sin la parte exponencial ('sin-escala') o con notación científica ('escalada', con mantisa y exponente).

La plantilla conformadora para un número no puede sobrepasar de 20 caracteres, sin contar el exponente ni tampoco las opciones de signos en la parte posterior.

Los números se redondean para que aparezca la cantidad de dígitos que se especifique en la plantilla.

Para especificar los dígitos significativos de un dato numérico:

'#' Cada carácter '#' en la plantilla especifica una posición de un dígito del dato a exponer.

'.' Especifica la posición que ha de ocupar el punto decimal. Puede haber como máximo un carácter '.' dentro de la plantilla conformadora de cada dato a exponer.

',' Puede aparecer antes del '.' anteriormente mencionado. Especifica una posición de dígito para el dato a exponer, y exige también que los dígitos que haya antes del punto decimal sean divididos en grupos de tres y separados por comas. (Recuerde que en inglés la notación numérica tiene el punto y la coma invertida con respecto a otros países europeos).

Dólar delantero y Asterisco:

Las siguientes son opciones mutuamente exclusivas. Estas opciones deben incluirse en la plantilla, inmediatamente antes de los caracteres que fijan las posiciones de dígito:

'$$' Especifica dos posiciones de dígito. Indica que ha de sacarse un signo '$' inmediatamente antes del primer dígito significativo o punto decimal (después de cualquier signo delantero). Observa que el '$' ocupará una de las posiciones de dígito del dato a exponer.

'**' Especifica dos posiciones de dígito. Indica que cualquier espacio blanco delantero que pudiera haber en el dato a exponer, sea sustituido por '*'s.

'**$' Especifica tres posiciones de dígito. Actúa como combinación de las dos opciones previamente mencionadas.

Opciones de Signo:

Lo prescrito para omisiones es que se saque el signo '-' inmediatamente antes del número (y de cualquier dólar delantero) cuando el número es negativo; y que no se saque ningún signo en absoluto cuando es positivo. El '-' en la plantilla, ocupará una de las posiciones de dígito del dato a exponer, de las situadas antes del punto decimal.

Es posible especificar que se saque el carácter '+' para números positivos, y que dicho signo vaya detrás del número.

'+' Especifica que se saque el '+' o el '-' según corresponda.

Si el '+' aparece al comienzo de la plantilla conformadora, el signo se saca inmediatamente antes del número (y de cualquier dólar delantero).

Si el '+' aparece al final de la plantilla conformadora, el signo se sacará después del número (y de cualquier exponente que hubiera).

'-' Unicamente puede aparecer al final de una especificación para conformar datos numéricos. Hace que se saque el '-' para datos numéricos negativos, y un espacio en blanco para los positivos. Esta indicación de signo aparece inmediatamente después del número (y de cualquier parte de exponente que hubiera).

Opción de Notación Científica, con Exponente:

'↑↑↑↑' seguido de la parte principal del número (la 'mantisa') y precediendo cualquier indicación posterior de signo, permite la notación con exponente. Los cuatro caracteres reservan espacio precisamente para los cuatro caracteres que forman la parte del exponente.

El 'cuerpo' del número se saca con la máxima cantidad de dígitos posible antes del punto decimal, reservando una posición de dígito para el signo (incluso cuando el número es positivo), si no se ha hecho ninguna otra especificación para ese dato.

Rebasamiento de Campo:

Si no se puede exponer un número que cumpla con la plantilla conformadora dada, el BASIC intenta acercarse lo más posible al formato exigido, pero hace que el resultado aparezca precedido de un '%' para señalar que se ha 'rebasado el campo'.

Ningún número cuya magnitud, cuyo valor absoluto sea mayor que o igual a 1E9 puede sacarse en 'formato sin escala'. Cualquier intento de hacerlo, hace que el número aparezca en 'formato libre' precedido del '%' para indicar el fallo producido.

4. Continuación de la Exposición de datos

Terminando el comando PRINT con un <separador>, TAB o SPC, impide que el BASIC lance una nueva línea, y que por tanto, el siguiente comando PRINT sea tratado como continuación del anterior en cuanto a posiciones.

Claves Asociadas

WIDTH, ZONE

RAD

Considera los radianes como medidas de los ángulos. **Comando**

Uso

Las funciones trigonométricas SIN, COS, TAN y ATN, pueden operar con los ángulos tanto en grados como en radianes. Está prescrito que trabajen en radianes, pero puede cambiarse esta prescripción mediante el comando DEG. RAD hace que las funciones vuelvan de nuevo a operar sobre ángulos en radianes.

Forma

RAD

Notas

La operación en grados sexagesimales persiste después de un comando DEG y hasta que se ejecute un comando RAD o uno CLEAR, o se cargue o ejecute un nuevo programa mediante LOAD y RUN respectivamente.

Claves Asociadas

DEG, SIN, COS, TAN, ATN

RANDOMIZE

Hace que sea aleatorio el germen actual de la secuencia de números de azar. **Comando**

Uso

El generador de números aleatorios del BASIC produce una secuencia pseudo-aleatoria, en que cada número depende completamente del número previamente producido. A partir de un valor dado como germen, la secuencia es siempre la misma. El comando RANDOMIZE fija un nuevo valor inicial para el generador de números aleatorios, bien sea con un valor determinado, o bien con un valor introducido por el operador.

Forma

RANDOMIZE[<expresión numérica>]

Si se omite la expresión, el operador es avisado con el mensaje:

```
Random Number Seed ?
```

que le pide que teclee un 'germen de números aleatorios' para ser usado como valor inicial de la secuencia a generar.

Notas

Fijando el número aleatorio inicial siempre al mismo número, da como resultado que se genere siempre la misma secuencia de números aleatorios.

RANDOMIZE TIME produce una secuencia que será dificilísimo que se repita en alguna otra ocasión.

Claves Asociadas

RND

READ

Leer las constantes incluidas en instrucciones DATA. **Comando**

Uso

READ recoge las constantes numéricas o literales de las instrucciones DATA y las 'apunta' como valores de variables.

Forma

READ<lista de:<variable>

Notas

Todas las instrucciones DATA que haya en un programa, tomadas en orden, forman una colección, un fichero secuencial de constantes. Dichas constantes pueden ser asignadas a variables mediante el comando READ. READ hace que el 'puntero' interno del BASIC avance a la siguiente constante de la lista, y la tome para apuntarla como valor de la siguiente variable dentro de la lista de variables de la instrucción READ. La constante y la variable a la que se asigna como valor, deben ser compatibles. Intentar hacer una lectura mediante el comando READ, una vez que se ha sobrepasado la última constante de las instrucciones DATA, genera un error (Error 4).

Claves Asociadas

DATA, RESTORE

RELEASE

Deja que sean emitidas las notas almacenadas en los canales de sonido.

Comando

Uso

Cuando se envía una nota sonora para que sea colocada en la secuencia de sonidos que van a emitirse por un determinado canal, puede incluirse una marca que fije a esa nota como 'retenida'. Si cualquiera de los canales especificados en este comando está en dicho estado de 'retención', queda suelto, liberado para que puedan ser emitidas sus notas sonoras.

Forma

RELEASE <canales sonido>

siendo <canales sonido> una <expresión entera> que produce un valor en la banda 1..7 y que especifica los canales que van a quedar liberados. Dicho valor es 'significativo según sus bits' con:

Bit 0 (el bit menos significativo) especifica el canal A
Bit 1 especifica el canal B
y **Bit 2** especifica el canal C

Notas

El comando RELEASE no tiene ningún efecto sobre los canales que no están 'retenidos'.

Claves Asociadas

SOUND

REM

Incluir comentarios en un programa. **Comando**

Uso

Para reflejar comentarios y explicaciones en los programas en BASIC.

Forma

REM<resto de línea>

El BASIC ignora lo que queda de la línea después de la palabra REM. Observa que también queda ignorado el signo separador de comandos, el dos-puntos.

Notas

Está permitido que se transfiera el control a un comando REM, mediante GOTO o por GOSUB.

El apóstrofe (la comilla simple) dentro de una línea -pero que no forme parte de un literal- es equivalente al comando :REM, y puede también servir para marcar como prefijo un texto que sirva de comentario.
La única excepción a esta situación es dentro del comando DATA, donde un apóstrofe se considera como parte de un literal sin entrecomillar.

REMAIN

Obtiene la cuenta que queda en el temporizador de demora. **Función**

Uso

Para conocer el tiempo que queda en cada uno de los 'cronómetros' posibles, y al mismo tiempo desactivarlo.

Forma

`REMAIN(<número del temporizador>)`

El <número del temporizador> es una <expresión entera> que especifica cuál de los cuatro temporizadores de demora es el que va a examinarse. La expresión debe producir un valor en la banda 0..3.

Notas

Esta función devuelve como resultado la cuenta que le queda a dicho temporizador. El resultado es cero si el temporizador especificado no había sido habilitado.

Claves Asociadas

`AFTER, EVERY`

RENUM

Renumera las líneas del programa en curso. **Comando**

Uso

Para cambiar la numeración de todo el programa o parte de él.

Forma

RENUM[<nuevo número línea>][,<viejo número línea>][,<incremento>]]

El <nuevo número línea> es un <número línea> que da el nuevo número que se quiere para la primera línea de programa que se vaya a renumerar. Si se omite, se supone 10.

El <viejo número línea> es un <número línea> que da la primera línea del programa que va a ser renumerada. Si se omite, se supone la primera línea que haya en dicho programa.

El <incremento> es un <número línea> que da el salto que va a usarse en la nueva numeración de líneas. Si se omite, se supone 10.

Observa que se pueden omitir incluso todos los parámetros, y que ello sería equivalente al comando RENUM 10,,10.

Notas

RENUM ajusta las referencias que en el programa hay a los números de línea para que queden coherentes con la nueva numeración del programa. Las referencias a números de línea mencionadas en los siguientes comandos son las que se ven afectadas:

DELETE, el DELETE en CHAIN MERGE, ELSE, GOSUB, GOTO, LIST, RESTORE, RESUME, RUN, THEN y las diversas ON...GOTO... y ON...GOSUB...
(véase también la función ERL que interactúa con RENUM).

Cuando se cambia la numeración de parte del programa, RENUM desautoriza y no permite ningún intento de cambiar el orden de las líneas, o de usar números de línea que ya estén siendo empleados en la parte del programa que no se está renumerando.

Si RENUM descubre referencias a números de línea que corresponden a líneas que no existen en el programa actual, el BASIC produce el mensaje:

```
Undefined Line 99999 in 88888
```

para indicar que hay una 'línea sin definir',

siendo: 99999 el número de línea al que se hace referencia en el programa, pero que no existe; y dicha referencia se deja sin cambiar de numeración.

y: 88888 es el nuevo número de la línea en que aparece la referencia incorrecta.

RESTORE

Hacer que el puntero repunte a una instrucción DATA determinada.

Comando

Uso

Para desplazar el 'puntero' interno que lleva la cuenta de las constantes ya usadas de las instrucciones DATA, se puede usar el comando RESTORE haciendo que señale a un determinado número de línea del programa (típicamente, para que vuelva a señalar a la primera instrucción DATA del programa).

Notas

RESTORE[<número línea>]

El <número línea> fija la línea hacia la que debe señalar el puntero interno de DATA. Si se omite, dicho puntero señala de nuevo a la línea inicial del programa.

Notas

Los comandos READ mueven un puntero interno sucesivamente a través de las constantes que figuran en las instrucciones DATA que haya en el programa. El comando RESTORE desplaza dicho puntero para que señale hacia una determinada línea del programa. El siguiente comando READ que se ejecute comenzará a tomar las constantes que apuntan las variables a partir de la constante inicial que aparezca en la primera instrucción DATA posterior a la mencionada en el comando RESTORE.

Claves Asociadas

READ, DATA

RESUME

Reanuda la ejecución después de tratar un error.

Comando
Sólo es Legal en el Modo
de Resarcimiento de Errores

Uso

Cuando se ha detectado, se ha 'atrapado' un error y subsecuentemente se ha pasado a ejecutar la subrutina señalada mediante el comando ON ERROR GOTO, para resarcirse del error, el comando RESUME permite que continúe la ejecución normal del programa a partir de diversos puntos del mismo.

Formas

```
        RESUME
o bien  RESUME<número línea>
o bien  RESUME NEXT
```

Notas

RESUME sólo es legal durante el Modo de Resarcimiento de Errores (i.e., dentro de una rutina ON ERROR GOTO, que haya sido activada por la aparición de un error).

RESUME sin ningún parámetro, devuelve el control de la ejecución al principio de la instrucción en que se detectó primeramente el error.

RESUME<número línea> devuelve el control a la línea que se especifique por su número.

RESUME NEXT devuelve el control de la ejecución a la instrucción que va inmediatamente detrás de aquélla en que se detectó el error en un principio.

Claves Asociadas

ON ERROR GOTO

RETURN

Volver de una subrutina. Comando

Uso

Marca el final de una subrutina. El BASIC continúa la ejecución del programa a partir del punto inmediatamente detrás de aquél en que se produjo el 'desvío' mediante el comando GOSUB.

Forma

RETURN

Notas

RETURN se usa al final de las subrutinas normales y también de las subrutinas cuya ejecución es suscitada por 'interrupciones'.

Claves Asociadas

GOSUB, ON x GOSUB, ON SQ GOSUB, AFTER n GOSUB, EVERY n GOSUB, ON BREAK GOSUB

RIGHT$

Extrae la parte de un literal situada a la derecha. **Función**

Uso

Para extraer una determinada cantidad de caracteres de un literal, tomándolos a partir del extremo derecho del mismo.

Forma

RIGHT$(<expresión literal>,<longitud requerida>)

La <expresión literal> determina el literal del que se van a estraer los caracteres para formar un nuevo sub-literal.

La <longitud requerida> es una <expresión entera> que da la cantidad de caracteres que van a ser extraídos del literal, y debe producir un valor en la banda 0..255.

Notas

Si el literal dado es más corto que la <longitud requerida>, entonces todo el literal dado es entregado como resultado de la función. En caso contrario, sólo se entrega como resultado la parte derecha del literal dado, que forma un sub-literal con la <longitud requerida>.

Claves Asociadas

MID$, LEFT$

RND

Número Aleatorio. **Función**

Uso

Para conseguir un número aleatorio. Puede ser el siguiente dentro de la secuencia actual de números aleatorios, el más reciente número aleatorio tratado o el primero de una nueva secuencia.

Forma

RND[(<expresión numérica>)]

Notas

RND devuelve como resultado un valor Real, siendo $0 \leq$ valor < 1. El generador de números pseudo-aleatorios produce cada número operando sobre el previamente tratado, de manera que a partir del mismo valor inicial, se produce siempre la misma secuencia de números aleatorios.

RND habiendo omitido su argumento, o con una expresión como argumento que produzca un valor mayor que cero, devuelve como resultado el siguiente número aleatorio que le corresponda en la secuencia actual.

RND con una expresión como argumento que produzca el valor cero, devuelve como resultado una copia del número aleatorio más recientemente generado.

RND con una expresión como argumento que produzca un valor menor que cero, comienza una nueva secuencia libremente, pero predecible, basada en dicho valor. El primer número en la nueva secuencia es el resultado de la función.

Claves Asociadas

RANDOMIZE

ROUND

Redondeo de valores numéricos. **Función**

Uso

Para conseguir un dato numérico con un determinado número de posiciones fraccionarias, o elevado a una determinada potencia de diez.

Forma

ROUND(<expresión numérica>[,<decimales>])

La <expresión numérica> constituye el valor que va a ser redondeado, y puede ser de cualquier clase numérica.

El parámetro opcional <decimales> especifica la cantidad de posiciones decimales que han de considerarse en el redondeo. Si está presente dicho parámetro <decimales>, ha de ser una <expresión entera> que produzca un valor en la banda -39..+39. Si se omite, se supone cero.

Notas

El valor de la <expresión numérica> se redondea hasta el número de posiciones fraccionarias especificadas por el parámetro <decimales>, en la forma siguiente:

<decimales> mayor que cero
: el valor se redondea hasta el número dado de dígitos después del punto decimal.

<decimales> cero o ausente
: el valor se redondea hasta un entero

<decimales> menor que cero
: el valor se redondea hasta dar ABS(<decimales>) ceros antes del punto decimal.

('Redondeo' significa redondear al más próximo, tal y como se describe en 2.9.1).

Claves Asociadas

INT, FIX, CINT

RUN ‹filename›

Carga y comienza a ejecutar un programa. **Comando**

Uso

Para cargar un programa desde el cassette y comenzar automáticamente a ejecutarlo.

Forma

RUN<nombre fichero>

El <nombre fichero> es una <expresión literal> que da el nombre que tiene el fichero en el cassette del que va a ser traído a la memoria.

Notas

Cualquier programa existente, función de usuario, variables y tablas, será eliminado de la memoria. Lo estipulado mediante DEFINT, DEFREAL y DEFSTR queda cancelado. Todos los ficheros en cassette quedan abandonados, perdiéndose cualquier posible información de salida que hubiera en el 'buffer'.

RUN<nombre fichero> es equivalente a LOAD<nombre fichero> seguido del comando RUN (excepto en el caso de los programas en BASIC protegidos, en que LOAD seguido de RUN no funciona, pero sí lo hace RUN<nombre fichero>).

Si el nombre del fichero dado por la <expresión literal> produce el literal vacío, entonces el BASIC intenta cargar el primer fichero que encuentre en la cinta.

Si el primer carácter del nombre del fichero dado por la <expresión literal> es '!', entonces dicho carácter se quita del nombre del fichero, y tiene el efecto de suprimir los mensajes que habitualmente genera el programa que efectúa la lectura del cassette.

Claves Asociadas

LOAD, SAVE, CHAIN

RUN [‹line number›]

Comenzar la ejecución del programa existente en memoria. **Comando**

Uso

Para iniciar la ejecución del programa en curso, ya sea al principio del mismo, o bien a partir de una línea dada.

Forma

RUN[<número línea>]

Si se omite el <número línea>, la ejecución comienza con la primera línea que haya en el programa.

Notas

Cualquier función de usuario existente, las variables y las tablas, se eliminan de la memoria. Lo estipulado mediante DEF INT, DEFREAL y DEFSTR queda cancelado. Todos los ficheros en cassette quedan abandonados, y cualquier posible información que hubiera en el 'buffer' de salida se pierde.

SAVE

Guardar en cassette un programa o el contenido de la memoria. **Comando**

Uso

Para escribir en el cassette el programa actualmente alojado en la memoria, o para escribir un área determinada de memoria en un fichero en cassette.

Forma

SAVE <nombre fichero>[,<clase fichero>[,<parámetros binarios>]]

El <nombre fichero> es una <expresión literal> que da el nombre con que quedará registrado en el cassette el fichero.

La <clase fichero>, si está presente, define la modalidad de fichero que va a ser creada. Si no se da <clase fichero> el programa actual en BASIC queda grabado en la cinta según la forma interna del BASIC. La <clase fichero> se especifica mediante una sola letra, que puede ser una de las siguientes:

A - el programa actual en BASIC se guarda como un fichero de texto en ASCII.
P - el programa actual en BASIC se guarda en la forma protegida interna.
B - se guarda un cierto área de la memoria como un fichero Binario.

Los <parámetro binarios> se exigen para la <clase fichero> **B,** pero en los demás casos es ilegal. Véase más adelante un comentario sobre el SAVE de ficheros Binarios.

Notas

Si el primer carácter del nombre de fichero dado por la <expresión literal> es '!', entonces dicho carácter se quita del nombre y tiene el efecto de suprimir los mensajes generados habitualmente por el programa que efectúa la lectura del cassette.

Después de una operación SAVE se puede usar el comando CAT para verificar que el fichero escrito en el cassette el legible (y además CAT proporciona el 'catálogo' de los ficheros que hay en la cinta).

SAVE sin una <clase fichero> crea un fichero con un programa en BASIC sin estar protegido.

SAVE,A crea un fichero con textos ASCII, de forma igual a lo que haría LIST#9.

SAVE,P crea un fichero de programa en BASIC protegido. Los programas protegidos sólo pueden ser ejecutados mediante RUN o encadenados mediante CHAIN. Si el BASIC regresa al Modo Directo por cualquier razón, mientras el programa alojado en memoria está protegido, se efectúa automáticamente una limpieza como si fuera NEW.

SAVE,B crea un fichero Binario que contiene una copia del recinto de memoria delimitado por los <parámetros binarios> especificados en el comando, y que tienen la siguiente forma:

<dirección comienzo>,<longitud>[,<punto de entrada>]

donde:

<dirección comienzo> es una <expresión direccional> que da la dirección del primer byte que va a escribirse.

<longitud> es una <expresión direccional> que da la cantidad de bytes que van a escribirse.

<punto de entrada> es una <expresión direccional> que da la dirección a partir de la cual comenzará la ejecución si el fichero corresponde a un programa, mediante RUN. Si se omite el parámetro <punto de entrada>, se supone que es cero, con lo que se restaura el sistema cuando se ejecuta el fichero mediante RUN.

La dirección de comienzo, la longitud del recinto de memoria y el punto de entrada al ejecutarse, quedan registrados en el cassette junto con los datos y son usados posteriormente por RUN y LOAD con Binario. El comando SAVE Binario transfiere el contenido de la memoria de escritura/lectura de la máquina, por lo que puede conservarse la memoria correspondiente a la pantalla situada por debajo de la 'ROM de la parte alta' y la memoria situada por debajo de la 'ROM de la parte baja'.

Claves Asociadas

LOAD, RUN, MERGE, CHAIN, CHAIN MERGE

SGN

Signo de un valor numérico. **Función**

Uso

Para determinar el signo de un valor numérico dado.

Forma

SGN(<expresión numérica>)

Notas

SGN da como resultado un valor Entero:

-1 si la <expresión numérica> es < 0
0 si la <expresión numérica> es = 0
+1 si la <expresión numérica> es > 0

Claves Asociadas

ABS

SIN

Seno de un ángulo. **Función**

Uso

Para calcular el seno de un ángulo dado. En el modo Radianes, el ángulo se expresa en radianes. En el modo grados el ángulo debe expresarse en grados sexagesimales.

Forma

`SIN(` <expresión numérica>)

La <expresión numérica> da el ángulo en radianes o grados cuyo seno se va a calcular. El valor de dicho argumento, al ser convertido a radianes, debe producir un valor en la banda aproximada -200.000..200.000.

Notas

Con ángulos extremadamente más grandes que 2*PI (360 grados) la exactitud de esta función se ve crecientemente deteriorada al transferir el ángulo al valor equivalente en la banda -PI..+PI (-180..+180 grados). Por tanto, en lugar de dar como resultado una cifra muy inexacta, el BASIC no evaluará la función `SIN` para valores que estén situados más allá de la banda anteriormente mencionada.

Claves Asociadas

`COS, TAN, ATN`

SOUND

Incluir una nota sonora dentro de la secuencia de sonidos a emitir.

Comando

Uso

El sistema mantiene una secuencia separada de comandos sonoros para cada uno de los tres canales de sonido existentes. El comando SOUND especifica que ha de generarse un sonido determinado, una nota sonora, e insertarla en la posición que le corresponda dentro de la secuencia o secuencias requeridas en el comando.

Forma

SOUND<estado canal>,<período tono>[,<duración>[,<volumen>[,<envolvente volumen>[,<envolvente tono>[,<período ruido>]]]]]

Los primeros dos parámetros son obligatorios, el resto son todos opcionales. Los parámetros tienen los siguientes significados:

<estado canal> es una <expresión entera> en la banda 1..255.

El valor dado está 'calibrado en binario' en la forma siguiente:

Bit 0: envía la nota al canal A
Bit 1: envía la nota al canal B
Bit 2: envía la nota al canal C

Bit 3: hay 'acorde' con el canal A
Bit 4: hay 'acorde' con el canal B
Bit 5: hay 'acorde' con el canal C

Bit 6: se marca como 'retenida'

Bit 7: hace que se 'evacúe' (vacíe) la secuencia de notas sonoras.

El <período tono> es una <expresión entera> en la banda 0..4095.

Fija el tono de la nota a emitir. Un <período tono> P produce un tono de frecuencia F, donde F=125000/P.

Un <período tono> de 0 no produce ningún tono en absoluto, lo que es útil cuando el comando SOUND se usa para producir únicamente ruido.

La <duración> es una <expresión entera> en la banda -32768..+32767.

El efecto del parámetro <duración> depende del signo y del valor:
> 0 especifica la duración del sonido en centésimas de segundo.
= 0 especifica que el sonido ha de durar hasta que termine la envolvente de volumen.

< 0 especifica que la envolvente de volumen debe repetirse ABS (<duración>) veces.

Si no se especifica ninguna <duración>, se supone 20, con lo que el sonido durará un quinto de segundo (20 centésimas de segundo).

El <volumen> es una <expresión entera> en la banda 0..15.

Especifica el volumen inicial de la nota. Si se especifica una envolvente de volumen, entonces se podrá variar el nivel de volumen mientras se está emitiendo el sonido.

Si no se especifica ningún <volumen> se supone el valor 12.

La <envolvente volumen> es una <expresión entera> en la banda 0..15.

Especifica cuál de las envolventes de volumen, si hay alguna, se va a usar para generar la nota sonora.

Si no se especifica ninguna <envolvente volumen>, o si la envolvente especificada no ha sido previamente estipulada e identificada en el programa, entonces se emplea la envolvente 0.

La envolvente 0 es una constante, y no puede por tanto ser cambiada mediante el comando ENV. La envolvente 0 mantiene el volumen dado por el parámetro <volumen> durante 2 segundos.

La <envolvente tono> es una <expresión entera> en la banda 0..15.

Especifica cuál de las envolventes de tono, si hay alguna, va a ser usada al generar la nota sonora.

Si se especifica envolvente 0 o no se especifica ninguna <envolvente tono>, o si la envolvente especificada no está previamente estipulada e identificada en el programa, entonces no se usa ninguna y el tono permanecerá constante mientras dure el sonido (o no habrá ningún tono si el período de tono especificado fue el cero).

El <período ruido> es una <expresión entera> en la banda 0..31.

Especifica cuál de los ruidos, si hay alguno, se va a añadir a la nota generada. Si se omite el parámetro o se da el valor cero, significa que no se añade ninguna clase de ruido.

Notas

El comando SOUND comprueba que hay sitio en la secuencia de notas a emitir por cada uno de los canales mencionados, antes de incluir la nota generada por el comando SOUND dentro de cualquiera de las secuencias. El comando SOUND esperará por tanto indefinidamente hasta que se consiga sitio en la secuencia, que va quedando libre al terminar de emitirse una nota sonora activa.

SOUND cancela cualquiera de las posibles interrupciones que haya sido facultada mediante el comando ON SQ, para cada canal mencionado en el comando.

El bit de 'evacuación' (que vacía y ocluye el canal sonoro) que es el bit más significativo del parámetro <estado canal>, hace que todos los comandos mencionados en el comando queden anulados. La nota activa en ese momento, si la hubiera, se termina; y la secuencia de notas se vacía. A continuación se inserta el nuevo sonido en la secuencia especificada y comienza a emitirse, a no ser que la emisión esté impedida por especificaciones de Acorde o de Retención.

El bit de Retención, que es el bit número seis del parámetro <estado canal> se añade cuando se incluye el sonido generado en la secuencia de notas del canal especificado. Cuando una nota con Retención estipulada alcanza la posición de cabeza de su secuencia, ese canal cesará de emitir sonido hasta que reciba el comando RELEASE, que libere la retención. Este mecanismo está pensado para permitir que las secuencias de notas sean marcadas con la primera nota sonora en estado Retenido.

Los Acordes, o 'encuentros entre canales' es un mecanismo para forzar el acompasamiento de diversos canales. Un sonido en el canal A marcado como Acorde con el canal B, no comenzará a emitirse hasta que en el canal B haya otro sonido marcado como Acorde con el canal B que esté preparado para comenzar a sonar -y viceversa. Todos los tres canales disponibles pueden marcarse como Acordes con cada uno de los otros.

Si se especifica más de un canal en el parámetro <estado canal>, se establece implícitamente que las notas correspondientes queden marcadas como Acordes.

Los circuitos de generación de sonido tienen una sola posibilidad de <período ruido> para los tres canales, de manera que cada vez que se estipule un <período ruido> revoca el previamente estipulado.

Claves Asociadas

ENV, ENT, RELEASE, ON SQ, SQ

SPACE$

Serie literal de espacios en blanco. **Función**

Uso

Para crear una constante literal formada por espacios en blanco y de una longitud dada.

Forma

SPACE$(<longitud>)

La <longitud> es una <expresión entera> que especifica la longitud requerida, o cantidad de espacios en blanco que contiene el literal. La expresión debe producir un valor en la banda 0..255. (Una <longitud> de cero produce el literal 'vacío', o nulo -que no contiene ningún carácter).

Claves Asociadas

SPC, TAB, STRING$

SPEED INK

Gradúa la rapidez con que parpadea la Tinta. **Comando**

Uso

Los comandos INK y BORDER permiten asociar dos colores con cada número identificativo de Tinta, en cuyo caso la Pluma, Papel o 'Reborde' alternará entre los dos colores especificados. Este comando establece los períodos, las duraciones que tienen los cambios entre colores.

Forma

SPEED INK<período>,<período>

Los dos <período>s son <expresiones enteras> que deben producir valores en la banda 1..255. El primero especifica el lapso de tiempo en que estará presente el primer color asociado a la Tinta; el segundo especifica el lapso de tiempo para el segundo color.

Notas

Para evitar efectos visuales desagradables, los 'colores de la paleta de tintas' se cambia durante el retroceso de los cuadros de imagen. Los tiempos que se manejan vienen dados por tanto, de acuerdo con los períodos de exploración de cuadros -que son 1/50-avos o 1/60-avos de segundo, dependiendo de las normas de TV que se usen. En el Reino Unido y en España de usan 1/50-avos. Esa es la única temporización que varía entre las diversas versiones de la máquina.

Claves Asociadas

INK, BORDER

SPEED KEY

Gradúa la rapidez de autorrepetición de las teclas. Comando

Uso

Al pulsar una tecla se consigue generar el valor asociado a ella; pero si la tecla se mantiene pulsada automáticamente, después de una cierta demora de tiempo, repetirá la generación de dicho valor. Este comando establece el tiempo de demora antes de que empiece a repetirse el envío del valor, y además la rapidez con que se efectuará dicho envío a partir de ese momento.

Forma

SPEED KEY<demora comienzo>,<período repetición>

La <demora comienzo> es una <expresión entera> en la banda 1..255, y especifica en 1/50-avos de segundo el retardo que ha de haber entre la primera pulsación de la tecla y la repetición automática del primer valor.

El <período repetición> es una <expresión entera> en la banda 1..255, y especifica en 1/50-avos de segundo el tiempo que ha de transcurrir entre cada repetición automática del valor, una vez que haya comenzado a actuar la repetición automática.

Notas

Si se fija una <demora comienzo> muy pequeña, puede ser que sea difícil de usar el teclado, porque la acción autorrepetitiva comienza a actuar dentro de la misma exploración del teclado.

No todas las teclas son de repetición automática, el comando KEY DEF permite facultar y cancelar la posibilidad de autorrepetición para cada tecla.

Claves Asociadas

KEY DEF

SPEED WRITE

Gradúa la rapidez con que se transfieren datos al cassette. **Comando**

Uso

Se pueden transferir datos al cassette en una de dos 'cadencias'. Este comando elige la que va a usarse.

Forma

SPEED WRITE<expresión entera>

La <expresión entera> debe producir un valor en la banda 0..1, especificando cuál de las cadencias de transferencia va a usarse, en la forma siguiente:

0: Nominal 1000 bits por segundo
1: Nominal 2000 bits por segundo

Notas

La rapidez de transferencia que se ha mencionado, son nominales porque se presupone que por término medio se escribe una cantidad igual de ceros y de unos; y porque no se tiene en cuenta la existencia de 'vanos' o 'cinta roja' alrededor de cada bloque de datos que se transfiere.

Están admitidas dos cadencias de transferencia hacia el cassette, porque la cinta en cassette no es el más fiable de los soportes informáticos. Mientras que con una mayor rapidez se obtiene un rendimiento adecuado, con menor rapidez se tiene un mayor margen de seguridad.

La máquina tiene prescrito para omisiones que use la rapidez de transferencia menor.

Al efectuar la lectura de datos en cinta, la máquina utiliza los bits de sincronización que hay en la parte delantera de cada bloque para acomodarse automáticamente a la 'cadencia' con que se haya grabado la información.

SQ

Comprueba el estado de la secuencia de notas sonoras. **Función**

Uso

La función SQ puede usarse para observar cuántos 'sitios libres' hay todavía en la secuencia de notas a emitir por un canal dado, para comprobar si el canal está activo, y si no lo está porque la nota situada en la cabeza de la secuencia (si hay alguna) está esperando a ser emitida.

Forma

SQ(<canal>)

El <canal> es una <expresión entera> que produce uno de los siguientes valores:

1: para el canal A
2: para el canal B
4: para el canal C

Notas

La función SQ entrega como resultado un valor Entero que está 'calibrado en binario', de acuerdo con los siguientes campos:

Bits 0..2	el número de sitios libres en la secuencia, en la banda 0..4	
Bits 3..5	el estado de Acorde en la cabeza de la secuencia (si lo hay)	mutuamente excluyentes
Bit 6	está 'alzado' si la cabeza de la secuencia está Retenida	
Bit 7	está activo en ese momento	

Los estados Acorde y Retenido se describen en el comando SOUND.

La función SQ cancela cualquier detección de interrupciones para el canal mencionado en la función, que haya sido facultada mediante el comando ON SQ.

Claves Asociadas

SOUND, ON SQ GOSUB

SQR

Raíz Cuadrada. **Función**

Uso

Evaluar la raíz cuadrada de un valor numérico dado.

Forma

SQR(<expresión numérica>)

Donde la <expresión numérica> debe producir un valor positivo.

STOP

Para la ejecución de un programa. **Comando**

Uso

Para detener la ejecución de un programa, pero dejando al BASIC en un estado desde el que se puede continuar la ejecución del programa mediante el comando CONT. Puede usarse este comando para hacer una pausa en la ejecución de un programa en un determinado punto, cuando se está intentando 'depurar'.

Forma

STOP

Notas

Cuando obedece el comando STOP, el BASIC suspende la ejecución del programa y expone el mensaje de 'Break' para indicar que se ha producido una ruptura en la ejecución.

Se puede hacer que continúe la ejecución interrumpida mediante un STOP, si se usa el comando CONT, siempre y cuando no se haya alterado el programa en ninguna manera mientras ha estado interrumpido.

Claves Asociadas

CONT, END

STR$

Representación literal de un valor numérico. **Función**

Uso

Para convertir un valor numérico dado en su representación literal -alfanumérica.

Forma

STR$(<expresión numérica>)

Notas

El valor de la <expresión numérica> se convierte a un 'literal' usando los mismos símbolos y de la misma manera que se usa en el comando PRINT. Observa que los valores positivos producirán una 'serie de caracteres' con un espacio en blanco delantero, mientras que los valores negativos tendrán un signo menos delante.

Claves Asociadas

VAL, PRINT, DEC$, HEX$, BIN$

STRING$

Literales con una serie de caracteres repetidos. **Función**

Uso

Para construir una constante literal formada por un determinado carácter que está repetido una cierta cantidad de veces. El carácter puede especificarse por su valor numérico.

Forma

STRING$(<longitud>,<especificador carácter>)

La <longitud> es una <expresión entera> que da la longitud requerida del literal resultante de la función. La expresión debe producir un valor en la banda 0..255.

El <especificador carácter> puede ser uno de los dos siguientes:

<expresión entera> que indica CHR$(<expresión entera>)
<expresión literal> que indica el primer carácter, el inicial, del literal dado.

Claves Asociadas

SPACE$

SYMBOL

Define la matriz de los Símbolos Definibles por el Usuario. **Comando**

Uso

El comando SYMBOL redefine la forma o presentación de un determinado carácter.

Forma

SYMBOL <número carácter>,<lista de:<filas>

El <número carácter> es una <expresión entera> en la banda n..255, siendo n el primero de los Símbolos Definibles por el Usuario (véase comando SYMBOL AFTER).

La <lista de:<filas> es una lista con uno y ocho elementos posibles, en que cada uno de ellos son <expresiones enteras> que han de producir un valor en la banda 0..255. Si se especifican menos de ocho filas para la matriz que determina la forma, el 'tipo' del carácter, las filas no especificadas se colocan al valor cero.

Notas

La matriz que fija la forma de un carácter es un área de ocho bytes, cada bit de los cuales corresponde a un 'punto de imagen' en la célula de ocho por ocho que el carácter ocupa en pantalla. Cuando se muestra el carácter en la pantalla, un bit con valor cero en su matriz de forma, establece que el correspondiente punto (pixel) aparezca con el color actual para el tinte del Papel, a no ser que se haya activado el modo Transparente en cuyo caso, el punto de imagen correspondiente no se vería afectado. De manera similar, un bit con valor uno en la matriz de forma del carácter produce un punto en el color correspondiente a la tinta de la Pluma.

Las filas especificadas en el comando SYMBOL se escriben en la matriz del carácter en el orden en que se dan. El bit más significativo de la primera fila corresponde a la esquina superior izquierda de la celdilla que ocupa el carácter en pantalla. El bit menos singificativo de la última fila corresponde a la esquina inferior izquierda de la celdilla que ocupa el carácter en la pantalla.

Cambiar la matriz que da forma a un carácter, no tiene ningún efecto sobre los caracteres que ya hayan sido mostrados en la pantalla.

Los Símbolos Definidos por el Usuario quedan restaurados a sus valores iniciales cada vez que se lanza un comando SYMBOL AFTER.

Claves Asociadas

SYMBOL AFTER

SYMBOL AFTER

Establece una nueva cantidad de Símbolos Definibles por el Usuario.

Comando

Uso

El número de caracteres cuya representación en pantalla puede alterarse mediante el comando SYMBOL se establece mediante el comando SYMBOL AFTER, que indica que los 'símbolos detrás' de uno dado pasan a ser considerados como definibles por el usuario.

Forma

SYMBOL AFTER<expresión entera>

La <expresión entera> debe producir un valor en la banda 0..256, y especifica el número del carácter a partir del cual todos son definibles por el usuario.

Notas

Cuando se pone en marcha por primera vez la máquina, automática e implícitamente obedece un comando SYMBOL AFTER 240, con lo que se obtienen 16 caracteres cuya forma es definible por el usuario.

SYMBOL AFTER n establece que todos los caracteres desde n hasta el 255 inclusive, pasan a ser definibles por el usuario. SYMBOL AFTER 256 establece que ningún carácter es definible por el usuario. SYMBOL AFTER 0 establece que todos los caracteres son definibles por el usuario.

Siempre que el BASIC obedece un comando SYMBOL AFTER todos los caracteres ya definidos por el usuario quedan restaurados a sus formas primitivas.

Los comandos SYMBOL AFTER no están permitidos si se ha alterado el tope de memoria mediante HIMEM, desde el comando más reciente SYMBOL AFTER (a no ser que haya sido precisamente SYMBOL AFTER 256). El tope de memoria HIMEM se altera mediante comandos MEMORY, o mediante el reparto automático de las zonas de memoria para 'buffers' de entrada/salida hacia cassette. Véase Apéndice IX para una descripción más completa de estas interacciones.

Claves Asociadas

SYMBOL

TAG

Texto en la posición del cursor de gráficos. **Comando**

Uso

El texto enviado hasta un cauce dado puede conseguirse que aparezca en la posición que actualmente ocupa el cursor de gráficos de dicho cauce. Así se permite que se combinen textos y símbolos con imágenes gráficas.

Forma

TAG[#<expresión cauce>]

La <expresión cauce> debe producir un valor coherente con un cauce de pantalla. Si se omite, se supone el cauce #0.

Notas

El carácter que se escribe en la posición ocupada por el cursor de gráficos, aparece de manera que la esquina superior izquierda de la celdilla que ocupa es la que se sitúa exactamente en la posición actual del cursor de gráficos.

El cursor de gráficos se avanza por tanto, ocho puntos de imagen después de que se ha mostrado cualquier carácter de esta manera. Observa que el número de incrementos horizontales conseguidos realmente según la pantalla de gráficos, y que viene representado por ocho puntos, varía según el modo en que se esté operando con la pantalla (multiplicado por 1, 2 ó 4).

Los caracteres de control no son reconocidos por el BASIC como tales, cuando se envían a la pantalla de gráficos de la manera mencionada aquí, y sólo tienen el efecto de mostrar el símbolo que corresponde a dicho valor de carácter. Observa en concreto, que los retornos de carro y los avances de línea, incluyendo aquéllos producidos automáticamente por el BASIC, no desplazarán el cursor de gráficos hacia abajo y hacia la izquierda, sino que harán que aparezca en lugar de eso, el símbolo o los dos símbolos que le corresponden.

El cauce #0 se ve forzado por el BASIC a dejar el estado TAG, cuando el BASIC regresa al Modo Directo.

Claves Asociadas

TAGOFF, SYMBOL

TAGOFF

Quitar el estado de Texto a Gráficos. **Comando**

Uso

Cancela la posibilidad de mostrar texto en la posición ocupada por el cursor de gráficos, para un cauce dado; que ha sido facultada mediante el comando TAG.

Forma

TAGOFF [#<expresión cauce>]

La <expresión cauce> debe producir un valor coherente con un cauce de pantalla. Si se omite, se supone el cauce #0.

Notas

El texto enviado hacia un cauce dado, se vuelve a dirigir a partir de este comando a la posición que ocupa actualmente el cursor de textos. El cursor de texto no se ha desplazado por los caracteres que se hayan enviado hacia el cauce dado, mientras el estado TAG estaba en activo.

Claves Asociadas

TAG

TAN

Tangente de un ángulo. **Función**

Uso

Para calcular la tangente de un ángulo dado. En el modo Radianes el valor del ángulo se expresa en radianes. En el modo Grados el valor del ángulo se expresa en ángulos sexagesimales.

Forma

TAN(<expresión numérica>)

La <expresión numérica> da el ángulo en radianes, o en grados. El valor del argumento, cuando se convierte a radianes debe estar en la banda aproximada -200.000..200.000.

Notas

Con valores extremadamente mayores que 2*PI (360 grados) la exactitud de esta función se ve crecientemente deteriorada al cambiar la escala del ángulo a la banda -PI..+PI (-180..+180 grados). En lugar de entregar como resultado una cifra muy inexacta, el BASIC no evaluará TAN para valores situados mucho más allá de la banda anteriormente mencionada.

Claves Asociadas

COS, SIN, ATN

TEST

Comprueba la Tinta usada en una posición de gráficos dada. **Función**

Uso

TEST puede usarse para examinar la tinta que se ha empleado en una determinada posición de la pantalla.

Forma

TEST (<coordenada x>,<coordenada y>)

La <coordenada x> y la <coordenada y> deben ser <expresiones enteras>.

Notas

La función TEST entrega como resultado el número identificativo de la tinta que se ha usado en la posición de la pantalla de gráficos especificada por las coordenadas dadas como argumentos. Si la posición especificada cae fuera de la ventana actual de gráficos, entonces el resultado corresponde a la tinta usada en el comando CLG más recientemente dado, y si no se ha ejecutado ningún comando CLG, el resultado que se obtiene es el valor cero.

TEST desplaza el cursor de gráficos hasta la posición dada en la función.

Claves Asociadas

TESTR, MOVE, MOVER, PLOT, PLOTR, DRAW, DRAWR

TESTR

Comprueba la tinta usada en una determinada posición de gráficos. **Función**

Uso

TESTR puede emplearse para saber el número de tinta usado en una determinada posición de la pantalla, que viene señalada por sus coordenadas relativas a la posición actual del cursor de gráficos.

Forma

TESTR(<desplazamiento x>,<desplazamiento y>)

El <desplazamiento x> y el <desplazamiento y> deben ser <expresiones enteras>.

Notas

La función TESTR entrega como resultado la tinta que está en uso en una determinada posición de gráficos de la pantalla, obtenida añadiendo el <desplazamiento x> a la coordenada x actual, y el <desplazamiento y> a la coordenada y actual. Si la posición especificada cae fuera de la ventana actual de gráficos, se entrega como resultado la tinta más recientemente usada en un comando CLG, y si no se ha ejecutado ningún comando CLG el resultado obtenido es el valor cero.

TESTR desplaza el cursor de gráficos hasta la posición dada en la función.

Claves Asociadas

TEST, MOVE, MOVER, PLOT, PLOTR, DRAW, DRAWR

TIME

Tiempo transcurrido. **Función**

Uso

El sistema mantiene la cuenta del tiempo transcurrido, el tiempo 'elapsado', desde que se puso en marcha la máquina. La función TIME entrega como resultado dicho 'tiempo'.

Forma

```
TIME
```

Notas

El tiempo se controla en unidades de 1/300-avos de segundo.

No se mantiene actualizado el tiempo mientras se están efectuando operaciones de lectura y escritura con el cassette.

TRON
TROFF

Para poner y quitar el mecanismo de 'rastreo' de la ejecución. **Comando**

Uso

El BASIC incluye la facilidad de poder rastrear la ejecución de un programa, presentando en pantalla el número de cada línea del programa, justamente antes de que pase a ser ejecutada.

Forma

```
TRON
TROFF
```

Notas

TRON faculta la posibilidad de rastreo. TROFF la cancela.

Mientras está habilitado el mecanismo de rastreo, va apareciendo en el monitor (encerrado entre corchetes cuadrados) el número de cada línea del programa, inmediatamente antes de que sea ejecutada.

TRON queda automáticamente cancelado siempre que se carga un nuevo programa, por lo que se quita el rastreo con los comandos NEW, LOAD, CHAIN y RUN<nombre fichero>.

UNT

Convierte un valor a entero sin-signo. **Función**

Uso

Para hallar el valor entero equivalente a un valor numérico sin-signo dado.

Forma

UNT (<expresión direccional>)

Notas

UNT da como resultado un valor Entero en la gama -32768..32767 (i.e. el equivalente en 'complemento a doses' del valor sin-signo de la <expresión direccional>). Véase Sección 2.10.

Claves Asociadas

INT, FIX, CINT, ROUND

UPPER$

Convierte a mayúsculas los caracteres de un literal. **Función**

Uso

Para crear un nuevo literal que es copia de otro literal con todos los caracteres alfabéticos en minúsculas convertidos a sus equivalentes en mayúsculas. ('Upper case' = 'caja alta').

Forma

UPPER$(<expresión literal>)

Notas

Se entrega como resultado la misma <expresión literal>, con todos los caracteres que tenga en la banda 'a'..'z' convertidos al carácter equivalente en la banda 'A'..'Z'.

Claves Asociadas

LOWER$

VAL

Convierte un literal a su valor numérico. **Función**

Uso

Para tomar la representación literal de un dato numérico y convertirlo a su valor de 'índole' numérica igual.

Forma

VAL(<expresión literal>)

Notas

VAL convierte el literal dado como argumento de la función, a un dato dato numérico de la misma manera que con el comando INPUT.

Claves Asociadas

STR$

VPOS

Halla la Posición Vertical actual. **Función**

Uso

Para establecer la posición vertical que corresponde actualmente al cursor de un cauce dado.

Forma

VPOS(#<expresión cauce>)

La <expresión cauce> debe producir un valor coherente con un cauce de pantalla.

Notas

La función VPOS entrega como resultado la posición vertical que ocupa en ese momento el cursor dentro de la ventana correspondiente al cauce dado como argumento de la función. La esquina superior izquierda de la ventana es la posición vertical 1.

Claves Asociadas

POS

WAIT

Vigila el estado de un portal de entrada/salida. **Comando**

Uso

Esperar hasta que en un portal de entrada/salida dado, haya un determinado valor.

Forma

WAIT<número portal>,<máscara>[,<inversión>]

El <número portal> es una <expresión direccional>.

Tanto el parámetro <máscara> como el <inversión> han de ser <expresiones enteras> que produzcan valores dentro de la banda 0..255.

Notas

El BASIC entra en un bucle al ejecutar este comando: hace una lectura del portal entrada/salida, efectúa la operación OR exclusiva entre el valor leído y el parámetro <inversión>, y con el resultado obtenido y el parámetro <máscara> efectúa a continuación una operación AND; y continúa repitiendo ese ciclo de operaciones hasta que el resultado final sea un valor distinto de cero. (Si no se da ningún parámetro <inversión>, entonces se omitirá la operación OR exclusivo).

Observa que no hay ninguna manera de interrumpir este bucle, de modo que el BASIC permanecerá en él indefinidamente si no se consigue nunca que aparezca la condición necesaria para salir del bucle.

Claves Asociadas

INP, OUT

WEND

Marca el límite final de un bucle WHILE. **Comando**

Uso

Un bucle WHILE hace que se ejecute repetidamente una serie de líneas de programa hasta que una determinada condición sea 'cierta'. El comando WEND define el extremo de dicho bucle.

Forma

WEND

Notas

El comando WEND señala la línea de programa que marca el final de un bucle WHILE. La manera en que están relacionados los comandos WHILE y WEND se describe en la sección correspondiente al comando WHILE. Cuando el BASIC encuentra un comando WEND sabe cuál es -si hay alguno- el comando WHILE con el que está asociado.

Claves Asociadas

WHILE

WHILE

Iniciar un bucle WHILE. **Comando**

Uso

Un bucle WHILE hace que se ejecute repetidamente una serie de líneas consecutivas de programa 'mientras' sea cierta una condición dada. El comando WHILE define la línea inicial de dicho bucle y establece la condición que ha de ser cierta para que el bucle sea ejecutado.

Forma

WHILE<expresión logical>

Notas

Cuando un comando WHILE es obedecido, se evalúa la <expresión logical>. Si el resultado es cero, el BASIC salta las líneas de programa siguientes hasta situarse por detrás del correspondiente comando WEND. Si el resultado de la evaluación, si la <expresión logical> no es cero, la ejecución continúa con las líneas sucesivas de programa y hasta que se encuentre el comando WEND correspondiente, momento en que el BASIC saltará hacia atrás hasta el comando WHILE que vuelve a ser examinado para ver si se cumple la condición, repitiéndose el proceso descrito.

El comando WEND que concuerda con un WHILE determinado, se establece estáticamente cuando se ejecuta la primera 'ronda' del comando WHILE. Esto quiere decir que el WEND que concuerda con el WHILE depende del orden de instrucciones en el programa, completamente independiente del orden de ejecución. No es posible, por lo tanto, tener más de un comando WEND asociado con un solo WHILE determinado.

Los bucles WHILE pueden ser 'anidados' (incluyendose completamente unos dentro de otros).

También está permitido terminar un bucle WHILE esquivando al comando WEND mediante un salto.

Claves Asociadas

WEND, FOR

WIDTH

Fija la anchura de línea en la impresora. **Comando**

Uso

Para decirle al BASIC cuántos caracteres pueden imprimirse a lo ancho del papel de la impresora. Esta información se necesita al imprimir para que el BASIC pueda insertar retornos de carro en los momentos apropiados.

Forma

`WIDTH<expresión entera>` .

La <expresión entera> da la anchura de la impresora y debe producir un valor en la banda 1..255 (aunque los valores pequeños pueden tener efectos curiosos).

Notas

El valor inicial prescrito para la anchura de la impresora es 132.

Estipulando que la anchura es de 255 tiene un significado especial, ya que el BASIC trata la impresora como si fuera de anchura infinita, por lo que nunca inserta retornos de carro.

El BASIC mantiene una cuenta de los caracteres imprimibles enviados hasta la impresora desde el más reciente retorno de carro, con lo que sabe la posición teórica a lo ancho de la impresora. Cuando dicha posición teórica alcanza el valor 255, el contador mencionado ya no puede incrementarse más, con lo que todas las posiciones mayores de 255 se consideran como si fueran 255. (Los caracteres imprimibles son aquéllos cuyos valores son mayores de 31 (&1F).).

Claves Asociadas

`PRINT, POS`

WINDOW

Fija una ventana de texto en la pantalla. **Comando**

Uso

Para estipular la pantalla de texto que corresponde a un cauce dado de pantalla.

Forma

WINDOW[#<expresión cauce>,]<izquierda>,<derecha>,<arriba>,<abajo>

La <expresión cauce> identifica al cauce cuyas dimensiones de ventana se están estipulando. Si se omite, se supone el cauce #0.

Todos los otros parámetros deben ser <expresiones enteras> que produzcan valores en la banda 1..255, y especifican las posiciones de los cuatro bordes de la ventana de forma absoluta según las filas y columnas de la ventana, siendo la esquina superior izquierda de la pantalla la posición 1,1:

<izquierda>	La columna extremo-izquierda incluida en la ventana
<derecha>	La columna extremo-derecha incluida en la ventana
<arriba>	La fila más superior (con mínimo número) incluida en la ventana
<abajo>	La fila más inferior (con máximo número) incluida en la ventana

Notas

El cursor correspondiente al cauce dado se desplaza a la esquina superior izquierda de la nueva ventana, posición 1,1.

El orden de los parámetros <izquierda> y <derecha> puede invertirse, dado que el <derecha> debe ser siempre mayor que, o igual al parámetro <izquierda>.

El orden de los parámetros <arriba> y <abajo> puede invertirse, dado que el <abajo> debe ser siempre mayor que, o igual al parámetro <arriba>.

Claves Asociadas

ORIGIN

WINDOW SWAP

Canjea -intercambia- ventanas de texto. **Comando**

Uso

Si no se especifica ningún parámetro de cauce, en los comandos que lo aceptan, está prescrito que se use para omisiones el cauce #0. Los mensajes producidos por el BASIC se envían hasta el cauce #0. Este comando permite, entre otros efectos, que se intercambie un cauce cualquiera con el cauce #0 para 'redirigir' el texto enviado hasta el cauce #0.

Forma

```
WINDOW SWAP<expresión cauce>,<expresión cauce>
```

Donde ambas <expresión cauce> deben producir un valor coherente con un cauce de pantalla.

Notas

Todos los atributos de los cauces de texto que intervienen en el comando también son intercambiados, incluyendo la Posición del Cursor, la Pluma, el Papel, y el estado de TAG (Texto A Gráficos).

Claves Asociadas

WINDOW, PEN, PAPER, TAG

WRITE

Escribir datos a través de un cauce de salida. **Comando**

Uso

Para sacar los valores de un cierto número de expresiones hasta un cauce dado, separándolos por comas y encerrándolos entre dobles comillas.

Forma

WRITE [#<expresión cauce>,][<lista a escribir>]

La <expresión cauce> especifica el cauce a través del que va a escribirse la información. Si se omite, se supone el cauce #0.

La <lista a escribir> es: <expresión>[<separador><expresión>]*

donde <separador> puede ser una coma o un punto-y-coma, intercambiablemente.

Notas

WRITE es similar a PRINT, exceptuando que:

- las zonas de exposición se ignoran
- los literales se escriben encerrados entre dobles comillas
- se añaden comas entre los elementos de la lista a escribir
- WRITE no permite la opción de separadores posteriores.

WRITE está pensado para ser usado con el cauce de cassette. La forma en que se escriben los datos permite que puedan ser recuperados directamente usando un comando INPUT con una lista de variables homóloga a la dada en el comando WRITE.

Claves Asociadas

PRINT

XPOS

Posición horizontal del cursor de gráficos. **Función**

Uso

Para conocer la posición X que ocupa actualmente el cursor de gráficos.

Forma

XPOS

Notas

XPOS da como resultado un valor Entero que corresponde a la posición actual según el eje X del cursor de gráficos.

Claves Asociadas

YPOS, MOVE, MOVER, ORIGIN

YPOS

Posición vertical del cursor de gráficos. **Función**

Uso

Para conocer la posición Y que ocupa actualmente el cursor de gráficos.

Forma

YPOS

Notas

YPOS da como resultado un valor Entero que corresponde a la posición actual según el eje Y del cursor de gráficos.

Claves Asociadas

XPOS, MOVE, MOVER, ORIGIN

ZONE

Fija el tamaño de la zona de exposición. **Comando**

Uso

Para cambiar la anchura de la Zona de Exposición empleada con el comando PRINT.

Forma

ZONE<expresión entera>

La <expresión entera> fija la nueva anchura de las zonas de exposición en pantalla, y debe producir un valor en la banda 1..255.

Notas

El valor prescrito para omisiones de la Zona de Exposición es de 13 caracteres. La anchura queda repuesta a dicho valor prescrito, siempre que se carga un nuevo programa, por lo que se hará así con los comandos NEW, LOAD, CHAIN y RUN<nombre fichero>.

Claves Asociadas

PRINT, WIDTH

APENDICE I

NUMEROS DE ERROR Y MENSAJES DE ERROR

Todos los errores detectados por el BASIC se enumeran aquí, según el orden de números de error que tienen asignados. Los mensajes producidos por el BASIC se presentan tal y como aparecen, junto con una breve descripción de las posibles causas.

1 Unexpected NEXT

Se ha encontrado un comando NEXT 'inesperado' porque no se estaba dentro de un bucle FOR, o porque la variable de control en el comando NEXT no concuerda con la de FOR.

2 Syntax Error

El BASIC considera 'Error sintáctico' dentro de la línea dada porque detecta una construcción gramatical que no es legal.

3 Unexpected RETURN

Se ha encontrado un comando RETURN 'inesperado' por no encontrarse dentro de una subrutina.

4 DATA exhausted

Se ha intentado un comando READ cuando ya se han quedado 'exhaustas' las constantes de las instrucciones DATA.

5 Improper argument

Este es un error de propósito general que indica que determinado 'argumento es inapropiado'. De alguna manera, el valor del argumento de una función, o de un parámetro de un comando no es válido.

6 Overflow

El resultado de una operación aritmética ha producido un 'rebase', o desbordamiento del valor permitido. Puede ser con notación en coma flotante, en cuyo caso es que alguna operación ha producido un valor mayor de 1.7E+38 (aproximadamente). Por el contrario, puede que sea el resultado de un intento fallido de cambiar un número en coma flotante a un entero con signo de 16 bits.

7 Memory full

El programa actual o sus variables puede que sean demasiado grandes, o que la estructura de control está anidada en demasiados niveles (GOSUBs, WHILEs, FORs), por lo que está la 'memoria llena'.

Un comando MEMORY dará este error si se hace un intento de establecer demasiado baja la 'cima' de la memoria del BASIC, o un valor extremada e imposiblemente alto. Observa que un fichero abierto en cassette tiene una cantidad de memoria reservada -un 'buffer'- para su uso exclusivo, y que eso restringe la cantidad de memoria que puede usarse. Véase Apéndice IX.

8 Line does not exist

Se ha mencionado una 'línea no existente'.

9 Subscript out of range

Uno de los subíndices empleados para señalar un elemento de una tabla es demasiado grande o demasiado pequeño, por lo que el mensaje dice que está 'subscrito fuera de gama'.

10 Array already dimensioned

Una de las tablas (array) mencionada en una instrucción DIM 'ya está dimensionada'.

11 Division by zero

Puede ocurrir en una división Real, en una división Entera, al calcular el módulo entero, o en una exponenciación.

12 Invalid direct command

El último comando que se ha pedido ejecutar no es 'válido como comando directo', i.e. en Modo Directo.

13 Type mismatch

Se ha presentado un valor numérico cuando se requería un valor literal, o viceversa; o se ha encontrado en un READ o en un INPUT un dato que no corresponde a la variable correspondiente, por lo que el mensaje dice que hay 'discordancia de clase'.

14 String space full

Se han creado tantos literales (strings) que no se dispone de más sitio, incluso después de la 'recogida de basura' de los literales previamente empleados.

15 String too long

El literal sobrepasa los 255 caracteres en longitud, por lo que es 'demasiado largo'. Puede generarse al empalmar o concatenar datos literales.

16 String expression too complex

La 'expresión literal demasiado compleja' puede generarse a través de valores literales intermedios. Cuando el número de estos valores sobrepasa un límite razonable, el BASIC ceja en su empeño y produce este mensaje de error.

17 Cannot CONTinue

Por una razón u otra, el programa en curso 'no puede continuar' su ejecución mediante el comando CONT. Observa que CONT está pensado para que siga la ejecución que se ha detenido mediante un comando STOP, la pulsación de [ESC][ESC], o un error, pero cualquier alteración que sufra el programa mientras está detenido, hace que la continuación de la ejecución sea imposible.

18 Unknown user function

Es 'desconocida la función de usuario' que se cita en una instrucción usando el comando FN, sin que previamente se haya dado nombre a dicha función mediante el comando DEF FN.

19 RESUME missing

Se ha encontrado el final del programa mientras estaba en el Modo de Resarcimiento de Errores (i.e. dentro de una rutina ON ERROR GOTO), y el mensaje comunica que 'falta RESUME', que es el comando usado para reanudar la ejecución después del tratamiento del error.

20 Unexpected RESUME

Se encuentra un RESUME 'inesperado', ya que este comando sólo es válido mientras está en el Modo de Resarcimiento de Error (i.e. dentro de una rutina ON ERROR GOTO).

21 Direct command found

Cuando se carga un programa desde el cassette se ha 'encontrado un comando directo', lo que indica que se grabó una línea de programa sin estar precedida de número de línea.

22 Operand missing

El BASIC considera que le 'falta un operando' en una expresión.

23 Line too long

Una 'línea demasiado larga' ha sido detectada por el BASIC cuando la ha convertido a su forma interna.

24 EOF met

Se ha hecho un intento de leer un fichero a partir del cauce de entrada por cassette, cuando ya se había 'tropezado con el final de fichero'.

25 File type error

Hay un 'error en la clase de fichero' que se está leyendo. OPENIN sólo está preparado para abrir ficheros con textos en ASCII. LOAD, RUN y otros comandos sólo están preparados para tratar con las clases de ficheros producidos mediante SAVE.

26 NEXT missing

Le 'falta el NEXT' que necesita para hacerlo concordar con un comando FOR.

27 File already open

Se ha ejecutado un comando OPENIN o un OPENOUT cuando el 'fichero ya estaba abierto'.

28 Unknown command

Es un 'comando desconocido' para el BASIC, y corresponde a un comando Externo.

29 WEND missing

Le 'falta el WEND' que necesita para que concuerde con un comando WHILE.

30 Unexpected WEND

El BASIC se ha encontrado un 'WEND inesperado' porque no estaba dentro de un bucle WHILE, o dicho WEND no corresponde con el bucle WHILE que está en activo.

APENDICE II

RUTINAS EXTERNAS Y COMANDOS EXTERNOS

El comando CALL se usa para 'ceder' la ejecución a rutinas externas en lenguaje máquina -para 'citarlas' o llamarlas-; y dichas rutinas deben cargarse en la memoria de la máquina por encima de la máxima dirección que emplee el BASIC. El formato de los parámetros y el mecanismo de cesión de parámetros son los que se describen en este apéndice.

Los Comandos Externos se tratan como CALL exceptuando que la rutina puede ya estar en la memoria de la máquina, o en circuito ROM externo; y que la dirección de la rutina a la que se cede el control se establece dinámicamente buscando el nombre dado al comando.

1. FORMA DE LOS DATOS EN MEMORIA

Esta es una breve descripción de la forma en que se depositan las tres clases de datos en la memoria. Todos se conservan con el byte más significativo en la parte delantera (en la dirección inferior).

1.1 Enteros

Valores de 16 bits en notación de complemento a doses.

1.2 Literales (Strings)

Los literales se conservan en dos partes: el Descriptor y el Cuerpo del Literal. El Descriptor del Literal es un vector de tres bytes. El byte menos significativo determina la longitud del literal, con el cero indicando que es el literal vacío y que el resto del Descriptor se ignora en ese caso. Los siguientes dos bytes dan la dirección de memoria donde comienza el Cuerpo del Literal.

1.3 Reales

Una mantisa de cuatro bytes, seguida de un exponente de un byte. Los números se depositan siempre en notación normalizada, y el bit más significativo de la mantisa es reemplazado por el signo. El exponente está 'polarizado' por 128. Un exponente cero significa que el número es cero, y que la mantisa deberá ignorarse. La mantisa está en la forma de signo y magnitud, con el punto binario a la izquierda del bit más significativo implicado.

2. SUBRUTINAS MEDIANTE CALL Y COMANDOS EXTERNOS

El comando CALL especifica la dirección de la subrutina, y puede opcionalmente ir seguida de la lista de parámetros que se ceden para la ejecución de la subrutina.

Un Comando Externo se identifica mediante una barra vertical (|) seguido del nombre del comando, y opcionalmente seguido de la lista de parámetros. El nombre del comando adopta la misma forma que un nombre de variable en BASIC, pero sin ningún designador de clase. Observa que la barra vertical no forma propiamente parte del nombre, sino que es simplemente la manera de distinguir entre comandos internos y externos. El BASIC fuerza a que todos los caracteres en el nombre del comando estén en mayúsculas, antes de que comience la búsqueda de la rutina a la que se cede la ejecución del comando externo.

Los parámetros se pasan a la subrutina por valores, por lo que cada parámetro puede ser una expresión numérica cuyo resultado es realmente el que se pasa, o una referencia a una dirección en memoria. El número y clase de parámetros debe concordar entre lo que interpreta el programa en BASIC y la subrutina a la que se cede la ejecución -el BASIC no efectúa ninguna comprobación.

Cada parámetro que se cede para la ejecución, es un número de dos bytes, cuya interpretación depende de su clase:

Expresión entera:	Da como resultado un valor entero en complemento a doses.
Expresión real:	Valor del resultado Real forzado a ser un Entero sin-signo.
Referencia a dirección:	Dirección del valor asignado a la variable. (Observa que el valor de un literal es su Descriptor).

Las condiciones de entrada son las siguientes:

El Registro A contiene el número de parámetros que se pasan a la subrutina.

El Registro Indice X contiene la dirección de los parámetros. Si hay 'n' parámetros, entonces el parámetro n-ésimo es un desplazamiento con (n-1)*2 a partir de la dirección señalada por el registro índice -de manera que el primer parámetro tiene el valor de desplazamiento mayor, y el último parámetro es el señalado directamente por el registro índice X.

Los contenidos de los otros registros del microprocesador no están definidos.

Las direcciones de salida son:

Se conservan los valores de los registros solidarios AF' BC'.
Todos los otros registros y los testigos (flags) quedan con información corrupta.

Las variables literales pueden alterarse siempre que el Descriptor de Literal no sea alterado **DE NINGUNA MANERA.**

APENDICE III

PALABRAS CLAVE EN BASIC

Las siguientes son palabras clave en BASIC, y están reservadas y no pueden ser usadas como nombres de variables:

ABS, AFTER, AND, ASC, ATN, AUTO

BIN$, BORDER

CALL, CAT, CHAIN, CHR$, CINT, CLEAR, CLG,
CLOSEIN, CLOSEOUT, CLS, CONT, COS, CREAL

DATA, DEC$, DEF, DEFINT, DEFREAL, DEFSTR, DEG,
DELETE, DI, DIM, DRAW, DRAWR

EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE,
ERL, ERR, ERROR, EVERY, EXP

FIX, FN, FOR, FRE

GOSUB, GOTO

HEX$, HIMEM

IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT

JOY

KEY

LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE,
LOG, LOG10, LOWER$

MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE, MOVER

NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO Ø, ON SQ, OPENIN, OPENOUT,
OR, ORIGIN, OUT

PAPER, PEEK, PEN, PI, PLOT, PLOTR, POKE, POS, PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM,
RESTORE, RESUME, RETURN, RIGHT$, RND, ROUND, RUN

SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED, SQ, SQR,
STEP, STOP, STR$, STRING$, SWAP, SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, THIME, TO,
TROFF, TRON

UNT, UPPER$, USING

VAL, VPOS

WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE

XOR, XPOS

YPOS

ZONE

APENDICE IV

POSICIONES LEGALES EN LA PANTALLA DE TEXTOS Y CARACTERES DE CONTROL

El cursor de textos puede ubicarse fuera de la ventana activa en ese momento. Diversas operaciones fuerzan a que el cursor esté situado en una posición legal antes de que puedan ser efectuadas las siguientes operaciones:

- exposición en pantalla de un carácter
- mostrar el 'cuadradito' indicativo del cursor
- obedecer los códigos de control marcados con un asterisco en la lista más adelante mencionada.

El procedimiento para forzar a que el cursor ocupe una posición legal en pantalla es el siguiente:

a. Si el cursor está más a la derecha del borde derecho de la ventana, entonces se mueve a la columna más a la izquierda de la fila inmediatamente debajo de la que está.

b. Si el cursor está más a la izquierda del borde izquierdo de la ventana, entonces se mueve a la columna más a la derecha de la fila inmediatamente encima de la que está.

c. Si el cursor está ahora por encima del borde superior de la ventana, entonces toda la ventana se desliza hacia abajo una fila, y el cursor se coloca en la fila superior de dicha ventana.

d. Si el cursor está ahora por debajo del borde inferior, entonces la ventana se desliza hacia arriba una fila, y el cursor se coloca en la línea inferior de dicha ventana.

Los textos y operaciones se efectúan en el orden mencionado. Las posiciones ilegales del cursor pueden ser cero o negativas, que indican que está fuera por la izquierda o por encima de la ventana.

Los caracteres con valores en la banda 0..31 enviados hasta la pantalla de texto no producen normalmente un carácter visivo en la pantalla, sino que son interpretados como códigos de control correspondientes a una determinada operación. Algunos de los códigos son del tipo llamado 'de escape' porque afectan al significado de uno o más de los caracteres que van detrás de él, y que corresponden a los parámetros de la acción que representan. Los códigos marcados con un '*' en la siguiente lista, fuerzan a que el cursor se sitúe en una posición legal dentro de la ventana actual, antes de que pueda obedecerse el control que representan -pero pueden dejar después de efectuar la acción, el propio cursor en una posición ilegal.

Los códigos y sus significados son los siguientes:

Valor		Nombre	Parámetro	Significado
&ØØ	0	NULO		Ningún efecto. Se ignora.
&Ø1	1	SOH	Ø.255	Expone el símbolo dado por el parámetro. Con ello se permite exponer en pantalla los símbolos cuyo valor está dentro de la banda 0..31.
&Ø2	2	STX		Oculta el cursor de textos.
&Ø3	3	ETX		Muestra el cursor de textos. Observa que el BASIC usa primordialmente la invisibilidad del cursor, y cancela esa función haciéndolo visible cuando está esperando introducción de datos por teclado.
&Ø4	4	EOT	Ø..2.	Fija el modo de gestión de la pantalla. El parámetro se reduce MOD 4. Es equivalente a un comando MODE.
&Ø5	5	ENQ	Ø..255	Envía el carácter especificado por el parámetro al cursor de gráficos.
&Ø6	6	ACK		Faculta la pantalla de textos (véase &15,NAK).
&Ø7	7	BEL		Suena el zumbador. Observa que con esto se evacúa las secuencias de notas pendientes de emitir.
&Ø8	8 *	BS		Retrocede el cursor una posición.
&Ø9	9 *	TAB		Avanza el cursor una posición.
&ØA	10 *	LF		Baja el cursor una fila.
&ØB	11 *	VT		Sube el cursor una fila.
&ØC	12	FF		Borra la ventana de texto y mueve el cursor a la esquina superior izquierda. Equivalente a un comando CLS.
&ØD	13 *	CR		Mueve el cursor al borde izquierdo de la pantalla dentro de la fila actual.
&ØE	14	SO	Ø..15	Fija la Tinta del Papel. El parámetro se reduce MOD 16. Equivalente a un comando PAPER.

&ØF	15	SI	Ø..15	Fija la tinta de la Pluma. El parámetro se reduce MOD 16. Es equivalente al comando PEN.
&1Ø	16	* DLE		Suprime el carácter actual. Rellena el cuadradito ocupado por el carácter con el Tinte del Papel actual.
&11	17	* DC1		Limpia desde el borde izquierdo de la ventana, hasta, e incluyendo la posición del carácter actual. Rellena los cuadraditos afectados con el Tinte del Papel actual.
&12	18	* DC2		Limpia desde, e incluyendo, la posición del carácter actual hasta el borde derecho de la pantalla. Rellena los cuadraditos afectados con el Tinte del Papel actual.
&13	19	* DC3		Limpia desde el principio de la ventana, hasta, e incluyendo, la posición del carácter actual. Rellena los cuadraditos afectados con el Tinte del Papel actual.
&14	20	* DC4		Limpia desde, e incluyendo, la posición del carácter corriente hasta el final de la ventana. Rellena los cuadraditos afectados con el Tinte del Papel actual.
&15	21	NAK		Cancela la pantalla de texto. La pantalla ya no reaccionará a nada que se le envíe hacia ella, hasta que reciba de nuevo un ACK(&06).
&16	22	SYN	Ø..1	Parámetro reducido MOD 2: 0 cancela la opción transparente 1 faculta la opción transparente.
&17	23	ETB	Ø..3	Parámetro reducido MOD 4: 0 fija Normal Gráficos Tinta Modo 1 " XOR " " " 2 " AND " " " 3 " OR " " "
&18	24	CAN		Intercambia tintas de Pluma y Papel.
&19	25	EM	Ø..255 Ø..255 Ø..255	Matriz de forma para los caracteres definibles por el usuario. Es equivalente al comando SYMBOL.
			Ø..255 Ø..255 Ø..255	Acepta nueve parámetros. El primer parámetro especifica cuál es la matriz del carácter.

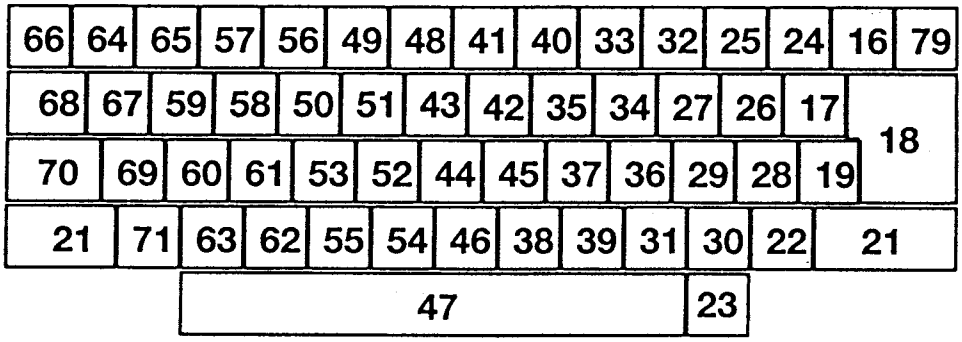
			Ø..255 Ø..255 Ø..255	Los siguientes ocho especifican la matriz de forma: el bit más significativo del primer byte corresponde al punto de imagen de la esquina inferior izquierda del cuadradito ocupado por el carácter; el bit menos significativo del último byte corresponde al punto de imagen de la esquina inferior derecha del cuadradito ocupado por el carácter.
&1A	26	SUB	1..8Ø 1..8Ø 1..25 1..25	Fija una Ventana. Es equivalente al comando WINDOW. Los primeros dos parámetros especifican los bordes izquierdo y derecho de la ventana -el valor inferior se toma como borde izquierdo, el superior como derecho. Los dos últimos parámetros especifican los bordes superior e inferior de la ventana -el valor inferior se toma como borde de arriba, el mayor como borde de abajo.
&1B	27	ESC		Ningún efecto. Se ignora.
&1C	28	FS	Ø..15 Ø..31 Ø..31	Asocia la Tinta a un par de colores alternantes. Es equivalente a un comando INK. El primer parámetro, reducido MOD 16 especifica el número de Tinta; los dos siguientes, reducidos MOD 32, los colores requeridos.
&1D	29	GS	Ø..31 Ø..31	Fija el reborde a un par de colores alternantes. Es equivalente a un comando BORDER. Los dos parámetros, reducidos MOD 32, especifican los dos colores requeridos.
&1E	30	RS		Desplaza el cursor a la esquina superior izquierda de la ventana, que es la posición 'base'.
&1F	31	US	1..80 1..25	Desplaza el cursor a la posición mencionada dentro de la ventana actual. Es equivalente a un comando LOCATE. El primer parámetro indica la columna, y el segundo la fila, de la nueva posición del cursor.

APENDICE V

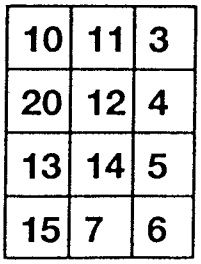
NUMEROS DE TECLA PARA TECLADO Y JOYSTICK

El comando KEY DEF permite que los valores producidos al pulsar una tecla puedan ser cambiados con respecto a los prescritos en el sistema; la función INKEY permite examinar el estado de una determinada tecla. A cada tecla se hace referencia mediante su **número de tecla.** Dichos números de tecla se expresan en el sistema de base 10 y están dados en los siguientes diagramas:

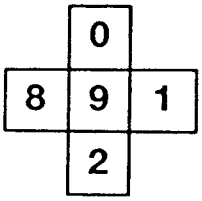
Teclado Principal



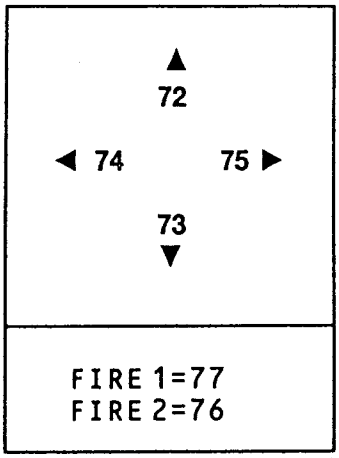
Tablero Funciones/Numérico



Teclas de Cursor

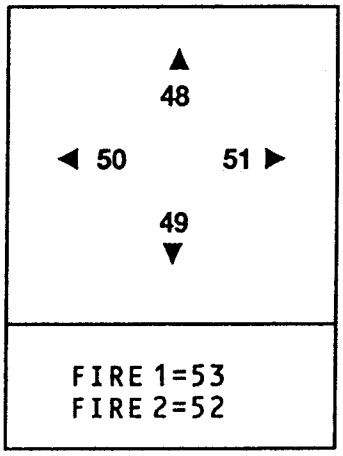


Joystick 0



el segundo joystick se solapa con algunas de las teclas que hay en el teclado principal del sistema, de la manera siguiente:

Joystick 1



La palabra FIRE, que indica 'fuego' se refiere al botón de acción -de disparo- que hay en los joysticks.

APENDICE VI

COLORES

Los números de color están dispuestos de manera que en pantallas monocromáticas, cuanto más alto sea el número, más brillo aparezca. Los números de color y los colores producidos (en una pantalla adecuada) son los siguientes:

0	NEGRO
1	AZUL
2	AZUL BRILLANTE
3	ROJO
4	MAGENTA
5	MALVA
6	ROJO BRILLANTE
7	PURPURA
8	MAGENGA BRILLANTE
9	VERDE
10	CIANO
11	AZUL CIELO
12	AMARILLO
13	BLANCO
14	AZUL PASTEL
15	NARANJA
16	ROSA
17	MAGENTA PASTEL
18	VERDE BRILLANTE
19	VERDE MAR
20	CIANO BRILLANTE
21	LIMA
22	VERDE PASTEL
23	CIANO PASTEL
24	AMARILLO BRILLANTE
25	AMARILLO PASTEL
26	BLANCO BRILLANTE

APENDICE VII

NOTAS SONORAS Y PERIODOS DE TONO

La tabla que sigue son los valores recomendados para los Períodos de Tono, para las notas en la escala habitual armónica par para la gama completa de ocho octavas.

La frecuencia producida no es exactamente la frecuencia requerida porque el valor dado para el período ha de ser un entero. El error relativo es la relación entre la diferencia de las frecuencias requeridas y reales, y la frecuencia requerida, i.e.: (REQUERIDA-REAL)/REQUERIDA).

NOTA	FRECUENCIA	PERIODO	ERROR RELATIVO	Octava-3
C	32.703	3822	-0.007%	
C#	34.648	3608	+0.007%	
D	36.708	3405	-0.007%	
D#	38.891	3214	-0.004%	
E	41.203	3034	+0.009%	
F	43.654	2863	-0.016%	
F#	46.249	2703	+0.009%	
G	48.999	2551	-0.002%	
G#	51.913	2408	+0.005%	
A	55.000	2273	+0.012%	
A#	58.270	2145	-0.008%	
B	61.735	2025	+0.011%	

NOTA	FRECUENCIA	PERIODO	ERROR RELATIVO	Octava-2
C	65.406	1911	-0.007%	
C#	69.296	1804	+0.007%	
D	73.416	1703	+0.022%	
D#	77.782	1607	-0.004%	
E	82.407	1517	+0.009%	
F	87.307	1432	+0.019%	
F#	92.499	1351	-0.028%	
G	97.999	1276	+0.037%	
G#	103.826	1204	+0.005%	
A	110.000	1136	-0.032%	
A#	116.541	1073	+0.039%	
B	123.471	1012	-0.038%	

NOTA	FRECUENCIA	PERIODO	ERROR RELATIVO	Octava-1
C	130.813	956	+0.046%	
C#	138.591	902	+0.007%	
D	146.832	851	-0.037%	
D#	155.564	804	+0.058%	
E	164.814	758	-0.057%	
F	174.614	716	+0.019%	
F#	184.997	676	+0.046%	
G	195.998	638	+0.037%	
G#	207.652	602	+0.005%	
A	220.000	568	-0.032%	
A#	233.082	536	-0.055%	
B	246.942	506	-0.038%	

NOTA	FRECUENCIA	PERIODO	ERROR RELATIVO	Octava 0
C	261.626	478	+0.046%	
C#	277.183	451	+0.007%	
D	293.665	426	+0.081%	
D#	311.127	402	+0.058%	
E	329.628	379	-0.057%	
F	349.228	358	+0.019%	
F#	369.994	338	+0.046%	
G	391.995	319	+0.037%	
G#	415.305	301	+0.005%	
A	440.000	284	-0.032%	
A#	466.164	268	-0.055%	
B	493.883	253	-0.038%	

NOTA	FRECUENCIA	PERIODO	ERROR RELATIVO	Octava 1
C	523.251	239	+0.046%	
C#	554.365	225	-0.215%	
D	587.330	213	+0.081%	
D#	622.254	201	+0.058%	
E	659.255	190	+0.206%	
F	698.457	179	+0.019%	
F#	739.989	169	+0.046%	
G	783.991	159	-0.277%	
G#	830.609	150	-0.328%	
A	880.000	142	-0.032%	
A#	932.328	134	-0.055%	
B	987.767	127	+0.356%	

NOTA	FRECUENCIA	PERIODO	ERROR RELATIVO	Octava 2
C	1046.502	119	-0.374%	
C#	1108.731	113	+0.229%	
D	1174.659	106	-0.390%	
D#	1244.508	100	-0.441%	
E	1318.510	95	+0.206%	
F	1396.913	89	-0.543%	
F#	1479.978	84	-0.548%	
G	1567.982	80	+0.350%	
G#	1661.219	75	-0.328%	
A	1760.000	71	-0.032%	
A#	1864.655	67	-0.055%	
B	1975.533	63	-0.435%	

NOTA	FRECUENCIA	PERIODO	ERROR RELATIVO	Octava 3
C	2093.004	60	+0.462%	
C#	2217.461	56	-0.662%	
D	2349.318	53	-0.390%	
D#	2489.016	50	-0.441%	
E	2637.021	47	-0.855%	
F	2793.826	45	+0.574%	
F#	2959.955	42	-0.548%	
G	3135.963	40	+0.350%	
G#	3322.438	38	+0.992%	
A	3520.000	36	+1.357%	
A#	3729.310	34	+1.417%	
B	3951.066	32	+1.134%	

NOTA	FRECUENCIA	PERIODO	ERROR RELATIVO	Octava 4
C	4186.009	30	+0.462%	
C#	4434.922	28	-0.662%	
D	4698.636	27	+1.469%	
D#	4978.032	25	-0.441%	
E	5274.041	24	+1.246%	
F	5587.652	22	-1.685%	
F#	5919.911	21	-0.548%	
G	6271.927	20	+0.350%	
G#	6644.875	19	+0.992%	
A	7040.000	18	+1.357%	
A#	7458.621	17	+1.417%	
B	7902.133	16	+1.134%	

Estos valores están todos calculados a partir de la Internacional 'A', como sigue:

```
FREQUENCY = 440*(2↑(OCTAVE+((N-10)/12)))
PERIOD = ROUND(125000/FREQUENCY)
```

donde N es 1 para C, 2 para C#, 3 para D, etc.

APENDICE VIII

LECTURA Y ESCRITURA CON CASSETTE

1. Ficheros BASIC en Cassette

El BASIC efectúa la lectura y escritura de datos agrupándolos en bloques de 2K bytes. Cada bloque consta de un registro de cabecera y hasta ocho registros de datos. Cada registro tiene su propia 'comprobación por suma'. La cabecera de cada uno de los bloques contiene la siguiente información:

Nombre del fichero
Clase del fichero
Número del bloque e indicación de primer/último bloque
Longitud del fichero
De dónde fue escrito (para ficheros Binarios)
Punto de entrada (para ficheros Binarios en Código Máquina)

Las clases de fichero reconocidas por el BASIC son:

- Fichero con programa en BASIC
- Fichero protegido con programa en BASIC
- Fichero con texto ASCII
- Fichero Binario

Los programas en BASIC pueden ser cargados en memoria a partir de cualquiera de las tres primeras clases de ficheros, aunque hay algunas restricciones cuando son ficheros protegidos. El contenido de los ficheros Binarios puede ser depositado de nuevo en la memoria, y puede ser ejecutado.

El cauce establecido para un fichero en cassette sólo permitirá lectura y escritura de ficheros con textos ASCII.

2. Mensajes producidos por el Programa Gestor del Cassette

A menos que explícitamente se comunique la cancelación al programa gestor del cassette, éste producirá un cierto número de mensajes al efectuar la lectura o escritura de la cinta, tal y como siguen:

```
Press PLAY then any key:
```

Este mensaje 'oprima PLAY y luego cualquier tecla', se lanza cuando el programa comienza a efectuar la lectura de la cinta (como resultado de un comando `LOAD`, `CAT`, `OPENIN`, o de la lectura de un fichero en cassette).

Lo de 'cualquier tecla' no es exacto en la totalidad. Si se pulsa **[SHIFT]**, **[CTRL]** o **[CAPS LOCK]**, no tiene ningún efecto. Si se pulsa **[ESC]** se abandonará la operación de lectura.

`Found FILENAME block 9`

Este mensaje: 'encontrado NOMBREFICHERO bloque 9', se produce al ir buscando el bloque requerido del fichero requerido y encontrar algún otro fichero o bloque. Este mensaje sólo es para información, pero puede indicar que se ha colocado en el cassette una cinta errónea, o se ha bobinado más allá del sitio donde se encuentra el bloque requerido.

`Rewind tape`

Mientras se está buscando el bloque requerido del fichero requerido, se ha encontrado un bloque para ese fichero con numeración mayor, por lo que se pide 'rebobinar la cinta'.

`Loading FILENAME block 9`

Se ha encontrado el bloque requerido del fichero requerido, y se comunica que se está 'cargando NOMBREFICHERO bloque 9' en la memoria de la máquina.

`Press REC and PLAY then any key:`

Este mensaje: 'oprima REC y PLAY y luego pulse cualquier tecla', se lanza cuando el programa comienza a grabar en la cinta (como resultado del comando `SAVE` o una operación de escritura en un fichero de cinta).

Lo de 'cualquier tecla' no es exacto en la totalidad. Si se pulsa **[SHIFT]**, **[CTRL]** o **[CAPS LOCK]**, no tiene ningún efecto. Si se pulsa **[ESC]** se abandonará la operación de escritura.

`Saving FILENAME block 9`

Comunica que el bloque requerido del fichero requerido se está 'guardando' -salvando-.

`Read error x`

La 'x' representa una única letra que indica la clase de 'error en lectura' que se ha producido, de acuerdo con lo siguiente:

a = bit a destiempo -datos que son 'basura'.
b = error CRC (al comprobar código por redundancia cíclica)
d = bloque demasiado largo para el 'buffer' previsto en memoria.

`Write error a`

El programa ha fallado mientras 'escribía'. Nunca debe suceder.

APENDICE IX

GESTION DE MEMORIA, SIMBOLOS DEFINIBLES Y 'BUFFERS' PARA CASSETTE

1. Introducción al Uso de la Memoria del BASIC

El BASIC tiene una forma muy simple de gestionar la memoria. El programa de usuario, las tablas y las variables se conservan en la parte inferior del área de memoria, a partir de la dirección más pequeña disponible para el BASIC, y hasta la máxima dirección disponible. La máxima dirección disponible para el BASIC -la 'cima'- se conoce como HIMEM, y es el resultado obtenido al aplicar dicha función.

El espacio inmediatamente por debajo de HIMEM está ocupado por la 'percha' en que se van apilando, o mejor, de la que se van 'colgando' los nuevos datos literales que van apareciendo. A medida que se cambian o añaden valores literales son tratados y depositados en dicha 'percha', que va creciendo hacia la parte de la memoria ocupada por el programa de usuario, las tablas, etc., engullendo el espacio de memoria libre. Durante el tratamiento de los literales, algunos de ellos son descartados o cambiados de valor, pero el espacio que ocupaban en memoria no queda liberado hasta que se haya extinguido todo el espacio libre, momento en que se 'recoge la basura' que se ha ido colgando en la 'percha'.

La dirección más pequeña disponible para el BASIC, se estipula cuando se implanta el BASIC por primera vez. La 'cima' de la memoria -HIMEM- puede cambiarse usando el comando MEMORY y también se ve afectada tanto por los comandos SYMBOL AFTER como por los 'buffers' necesarios para los ficheros en cassette. Cuando se cambia el valor de HIMEM, se efectúa habitualmente una recogida de la basura que haya en la percha, para asegurar que ésta ocupa el mínimo espacio posible, y luego se desplaza la 'cima' de la memoria. Si ha habido un montón de literales activos, dicha recogida de basura puede tardar un tiempo apreciable.

2. MEMORY y HIMEM

La función HIMEM da la dirección de la máxima celdilla en memoria que el BASIC puede usar para tramitar el programa del usuario en BASIC, con sus variables, tablas y 'percha' para literales. El comando MEMORY puede usarse para desplazar hacia arriba y hacia abajo la 'cima' de la memoria usable -HIMEM-; extendiendo o restringiendo el espacio libre. El BASIC rehusará un comando MEMORY que él no pueda obedecer, por no disponer de espacio, y generará un Error 7 (Memory full = Memoria llena).

Si se mueve HIMEM hacia abajo se reserva un área de memoria, justamente por encima de la nueva dirección que señala HIMEM; espacio que el BASIC ya no intentará nunca usar. Este área desocupada, puede usarse de la manera que al usuario le apetezca, pero habitualmente se emplea para subrutinas en 'código máquina'. La siguiente subrutina crea una subrutina en código máquina que estipula el papel de gráficos:

```
10 MEMORY HIMEM-5                          :REM Reserva 5 bytes
20 gra.set.paper =HIMEM+1                  :REM Dirección del área reservada
30 POKE HIMEM+1,&DD:POKE HIMEM+2.&7E       :REM LD A,(IX+0)
40 POKE HIMEM+3,&C3:POKE HIMEM+4,&E4
                  :POKE HIMEM+5,&BB        :REM JP GRA SET PAPER
```

y ahora puede hacerse que el BASIC **ceda** la ejecución de esa tarea a la subrutina en lenguaje máquina, incluida en dicha área mediante el comando:

```
50 CALL gra.set.paper,3:   REM Fija el papel de gráficos a tinta 3.
```

Reservando áreas de memoria situadas fuera del control del BASIC, interactúa con la propia reserva de áreas que el BASIC efectúa para los caracteres definibles por el usuario, y para los 'buffers' de entrada/salida de los ficheros en cassette, tal y como se describe a continuación.

3. Ficheros en Cassette - Reparto Dinámico

En cuanto se abre un fichero en cassette, el BASIC exige 4K (4096 bytes) que va a manejar como **'buffers'**, o memorias intermedias, en donde y de donde depositar los registros que posteriormente transferirá desde y hasta el cassette. En lugar de reservar permanentemente dicho espacio, inutilizándolo cuando no hay ningún fichero abierto, el BASIC sólo lo adquiere cuando lo necesita y para ello desplaza hacia abajo HIMEM. Cuando el área ocupada por ese 'buffer' ya no se necesita más, el BASIC deja libre ese espacio, y si puede vuelve a desplazar hacia arriba HIMEM. El espacio del 'buffer' sólo puede quedar disponible cuando está inmediatamente situado por encima de HIMEM. El siguiente programa reserva permanente y efectivamente dichos 'buffers':

```
10 OPENOUT "postizo"       :REM Hace que el BASIC reserve el 'buffer'
20 como.estaba=HIMEM       :REM Recuerda la posición actual
30 MEMORY HIMEM-1          :REM baja HIMEM un byte
40 CLOSEOUT                :REM Abandona el fichero
```

Observa que todavía no se ha escrito ningún dato en la cinta, ya que BASIC no se molesta en crear ficheros vacíos. Los 'buffers' reservados no pueden quedan libres cuando se ejecuta el comando CLOSEOUT, porque no están inmediatamente debajo de HIMEM. La memoria reservada por desplazamientos de HIMEM desde que se reservaron los 'buffers', no puede moverse; por tanto HIMEM no puede ahora moverse, y no puede liberarse el espacio de los 'buffers'. Si se efectúan posteriormente operaciones con el cassette, los 'buffers' existentes se vuelven a utilizar.

El siguiente comando liberaría subsecuentemente dichos 'buffers':

```
5Ø MEMORY como.estaba
```

siempre que no se hubiera abierto ningún fichero en cassette. Devolviendo la memoria reservada mediante movimiento de HIMEM, quita el bloque de datos al liberar los 'buffers'.

4. Caracteres Definibles por el Usuario - SYMBOL AFTER

La ROM contiene **matrices de forma** para todos los 256 caracteres disponibles en la máquina. El usuario puede declarar cualquier cantidad de caracteres como 'definibles por el usuario', simplemente estipulando el primer carácter que desea sea definible por el usuario. El comando SYMBOL AFTER es el mecanismo del BASIC para declarar aquellos caracteres que van a ser definibles por el usuario, por ejemplo

```
SYMBOL AFTER 192
```

hace que todos los caracteres cuyos números estén comprendidos entre 192 y 255, ambos inclusive, sean considerados como definibles por el usuario. Los casos extremos serán pues:

```
SYMBOL AFTER Ø   :REM hace que todos los caracteres sean definibles por el usuario
SYMBOL AFTER 256 :REM hace que ningún carácter sea definible por el usuario
```

Las matrices correspondientes a los caracteres definibles por el usuario, deben ser transferidas desde la ROM hasta la memoria de escritura/lectura de manera que pueda dárseles la forma que al usuario le convenga. El BASIC reserva pues espacio para esas matrices, cambiando el valor de HIMEM. Si ya existen algunos caracteres definibles por el usuario, un comando SYMBOL AFTER puede aumentar o disminuir la memoria que se exige para las matrices. El BASIC sólo puede hacer esto cuando las matrices corrientes definibles por el usuario están inmediatamente por debajo de HIMEM -no puede reorganizar la memoria situada entre HIMEM y las matrices.

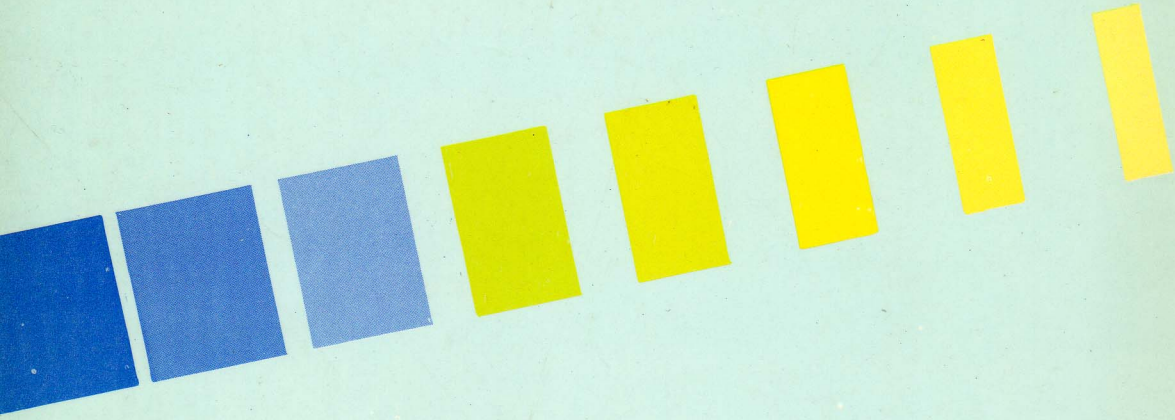
Cuando se implanta el BASIC al poner en marcha, los últimos 16 caracteres (desde 240 hasta 255) se declaran implícitamente como definibles por el usuario. Si el usuario requiere más de estos caracteres y además quiere reservar algo de memoria por encima de HIMEM, puede lograrlo bien mediante:

```
1Ø SYMBOL AFTER 16Ø    :REM declara 92 caracteres definibles por el usuario
2Ø MEMORY HIMEM-4Ø96   :REM Reserva 4K bytes por encima de HIMEM
```

o bien mediante:

```
1Ø SYMBOL AFTER 256    :REM quita todos los caracteres definibles por el usuario
2Ø MEMORY HIMEM-4Ø96   :REM Reserva 4K bytes por encima de HIMEM
3Ø SYMBOL AFTER 192    :REM Declara 64 caracteres definibles por el usuario
```

El segundo de estos esquemas es más flexible, dado que permite al programa cambiar posteriormente el número de caracteres definibles por el usuario.



AMSTRAD

Avda. del Mediterráneo, 9. 28007 MADRID