

ZX Spectrum

Programación BASIC actualizada y ampliada

Queda prohibida la reproducción total o parcial de esta obra, y su distribución o comunicación pública en cualquier forma o por cualquier medio sin el permiso previo y por escrito del autor.

Esta obra no está afiliada ni respaldada oficialmente por *Sinclair Research Ltd.* ni por las empresas que en la actualidad posean los derechos sobre los productos que aquí se mencionan. Los nombres, marcas y productos citados, como *ZX Spectrum, ZX81, Z80* y otros, son propiedad de sus respectivos titulares o sucesores legales y se utilizan en el presente texto únicamente con fines históricos, educativos y de análisis tecnológico.

© dev pep, 2025. Todos los derechos reservados.

Independently published Depósito legal: M-16388-2025 ISBN: 9798291860526

Título: ZX Spectrum: Programación BASIC actualizada y ampliada

Autor: dev pep

El autor ha puesto el máximo cuidado en la elaboración de este libro, pero no se asume ninguna responsabilidad por errores o por el uso de la información contenida en el mismo.

Índice

Introducción		
Instrucciones, funciones, sentencias y comandos	15	5
Definición sintáctica	15	5
Bases numéricas	18	3
Estructura del manual	19	Э
Capítulo 1: Tipos de datos	23	3
Tipos numéricos	23	3
Números enteros	23	3
Números de punto flotante	25	5
Cadenas de caracteres (strings)	26	3
Literal string	26	3
Slicing de strings	27	7
Arrays	29	9
Variables	30	J
LET	31	1
Asignación con índices	33	3
Asignación con slicer	34	4
Asignación con índices y slicer	35	5
DIM	35	5
Arrays numéricos	36	6
Arrays de strings	.39	Э
Expresiones	42	2
Operadores	. 43	3
Conversión entre tipos	48	3
STR\$	48	3
VAL	50	J
Anidamiento de strings	52	2
VAL\$	54	4
Referencias cíclicas	55	5
LEN	55	5
Capítulo 2: Primeras instrucciones y funciones	57	7
Funciones vs. instrucciones	57	7
Funciones y paréntesis	58	3
REM		
RUN		

STOP	
CONTINUE	60
CLEAR	62
NEW	63
CHR\$	
CODÉ	
Capítulo 3: La pantalla	65
PRINT	65
Cursor de posición	66
Coordenadas de texto	68
Cambio de posición con separadores	60
Caracteres de control de posición	 70
Cantroladares de control de posicion	/ <i>2</i>
Controladores de posición	/4
LIST	<u>/</u> 5
LIST	<u>//</u>
Caracteres de control de color	77
Formas de los caracteres	79
Caracteres definidos por el usuario (UDG)	80
Visualización de los colores	81
Visualización de los colores	83
Combinaciones de teclas	83
Controladores de color	84
Colores temporales y permanentes	85
INK	87
PAPER	87
FLASH	87
BRIGHT	87
INVERSE	88
OVER	88
Ejemplos con atributos de color	88
BORDER	90
CLS	
Color y variables del sistema	91
Color permanente	92
Color temporal	9.3
Funcionamiento	
Borde	94
ATTR	94
SCREEN\$	95
Dibujo con píxeles.	90
INT	
PLOT	
Pl	
DRAW	
CIRCLE	
Ejemplo	1∪∠ 1∩ว
POINT	
Capítulo 5: El teclado	105
INPUTINPUT	
IINI U I	107

Valores asignados a las variables	109
INKEY\$Capítulo 6: Impresora, <i>microdrive</i> y canales	111
Capítulo 6: Impresora. <i>microdrive</i> v canales	113
Instrucciones sin funcionalidad	113
LPRINT	114
LLIST	
COPY	115
Los canales de E/S	115
ODEN	110
OPEN	
CLOSE	118
Capítulo 7: Bifurcaciones, bucles y saltos	121
Saltos	
GO TO	122
GO SUB / RETURN	123
Subrutinas	
¿Por qué una pila?	125
Argumentos y valor de retorno	126
IF	127
Expresiones lógicas	120
Operadores relacionales	100
Operadores relacionales	132
AND y OR en otras expresiones	133
RND	
Cálculos	
Números posibles	138
RANDOMIZÉ	
PAUSE	139
Bucles	
FOR / NEXT	140
Anidamiento de bucles	142
Fiemples	1/2
EjemplosCapítulo 8: Series de datos	140 1 17
DATA	141 117
READ	
BEEP	149
_ Altura máxima de una nota	151
Ejemplos	
RESTORE	154
Capítulo 9: Funciones	157
Aproximaciones polinomiales	157
ABS	158
EXP	
LN	
SGN	
SQR	160
DEF FN	
FN	
USR	
Trigonometría	
SIŇ	168

COS	
Cálculo del coseno	169
TAN	
ASN	170
ACS	
ATN	171
Capítulo 10: Memoria y periféricos	173
PEEK	173
POKE	176
Los puertos de entrada/salida	180
Puertos de E/S de la ULA	184
Otros puertos de E/S	186
IN	187
OUT	187
La cinta de casete	
Cabeceras y bloques	189
SAVE	191
LOAD	195
Ejemplo de bloques múltiples	198
VERIFY	199
MERGE	202
Capítulo 11: Nuevas posibilidades	205
Emuladores	205
Archivos de cinta	206
Utilidades de desarrollo	206
Ensambladores y compiladores cruzados	
Dispositivos	210
Nuevas recreaciones	210
Periféricos	213
Panorama actual	
Posibilidades educativas	214
Apéndice 1: Teclado, comandos y pantalla	217
Modos o estados de teclado	218
Teclas en emuladores	221
Comandos y programas	221
Informes de error	223
Informes de error	225
Tipos de caracteres	228
Caracteres de control	229
Otros caracteres	229
Apéndice 3: Sistema binario y hexadecimal	231
Sistema binario	231
Representación entera	233
Coma o punto flotante	240
Sistema hexadecimal	241
Correspondencia con el sistema binario	242
Conversión binario-decimal en BASIC	243
Apéndice 4: Gestión de memoria	245
Uso de la memoria	

Archivo de pantalla	Memoria ROM	248
Atributos	Archivo de pantalla	.248
Buffer de impresora. 252 Variables del sistema. 253 Mapas del microdrive. 253 Información de los canales de E/S. 253 Programa BASIC. 254 Variables. 256 Números enteros. 256 Números de punto flotante. 258 Variables en memoria. 260 Zona de edición y entrada con INPUT 268 Pilas: cálculo, sistema y GO SUB. 269 RAMTOP. 272 Gráficos de usuario. 273 Apéndice 5: Teclados. 275 Apéndice 6: Códigos de error. 277 0 CK. 277 1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279	Atributos	250
Variables del sistema. 253 Mapas del microdrive. 253 Información de los canales de E/S. 253 Programa BASIC. 254 Variables. 256 Números enteros. 256 Números de punto flotante. 258 Variables en memoria. 260 Zona de edición y entrada con INPUT 268 Pilas: cálculo, sistema y GO SUB. 269 RAMTOP. 272 Gráficos de usuario. 273 Apéndice 5: Teclados. 273 Apéndice 6: Códigos de error. 277 0: OK. 277 1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 <	Buffer de impresora	.252
Mapas del microdrive. 253 Información de los canales de E/S 253 Programa BASIC. 254 Variables. 256 Números enteros. 256 Números de punto flotante. 258 Variables en memoria. 260 Zona de edición y entrada con INPUT 268 Pilas: cálculo, sistema y GO SUB. 269 RAMTOP. 272 Gráficos de usuario. 273 Apéndice 5: Teclados. 275 Apéndice 6: Códigos de error. 277 0: OK. 277 1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: O	Variables del sistema	.253
Información de los canales de E/S. 253 Programa BASIC. 254 Variables. 256 Números enteros. 256 Números de punto flotante. 258 Variables en memoria. 260 Zona de edición y entrada con INPUT 268 Pilas: cálculo, sistema y GO SUB. 269 RAMTOP. 272 Gráficos de usuario. 273 Apéndice 5: Teclados. 275 Apéndice 6: Códigos de error. 277 0: OK. 277 1: NEXT without FOR 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280	Mapas del <i>microdrive</i>	.253
Programa BASIC. 254 Variables. 256 Números enteros. 256 Números de punto flotante. 258 Variables en memoria. 260 Zona de edición y entrada con INPUT. 268 Pilas: cálculo, sistema y GO SUB. 269 RAMTOP. 272 Gráficos de usuario. 273 Apéndice 5: Teclados. 275 Apéndice 6: Códigos de error. 277 0: OK. 277 1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280	Información de los canales de E/S	.253
Variables. 256 Números de punto flotante. 258 Variables en memoria. 260 Zona de edición y entrada con INPUT 268 Pilas: cálculo, sistema y GO SUB. 269 RAMTOP. 272 Gráficos de usuario. 273 Apéndice 5: Teclados. 275 Apéndice 6: Códigos de error. 277 0: OK. 277 1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281	Programa BASIC	.254
Números de punto flotante 258 Números de punto flotante 260 Zona de edición y entrada con INPUT 268 Pilas: cálculo, sistema y GO SUB 269 RAMTOP 272 Gráficos de usuario 273 Apéndice 5: Teclados 275 Apéndice 6: Códigos de error 277 0: OK 277 1: NEXT without FOR 277 2: Variable not found 277 3: Subscript wrong 278 4: Out of memory 278 5: Out of screen 278 6: Number too big 278 7: RETURN without GO SUB 278 8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: F	Variables	256
Números de punto flotante. 258 Variables en memoria 260 Zona de edición y entrada con INPUT 268 Pilas: cálculo, sistema y GO SUB 269 RAMTOP 272 Gráficos de usuario 273 Apéndice 5: Teclados 275 Apéndice 6: Códigos de error 277 0: OK 277 1: NEXT without FOR 277 2: Variable not found 277 3: Subscript wrong 278 4: Out of memory 278 5: Out of screen 278 6: Number too big 278 7: RETURN without GO SUB 278 8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR w		
Variables en memoria. 260 Zona de edición y entrada con INPUT 268 Pilas: cálculo, sistema y GO SUB 269 RAMTOP 272 Gráficos de usuario. 273 Apéndice 5: Teclados 275 Apéndice 6: Códigos de error 277 0: OK. 277 1: NEXT without FOR 277 2: Variable not found 277 3: Subscript wrong 278 4: Out of memory 278 5: Out of screen 278 6: Number too big 278 7: RETURN without GO SUB 278 8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid V	Números de punto flotante	258
Zona de edición y entrada con INPUT. 268 Pilas: cálculo, sistema y GO SUB. 269 RAMTOP. 272 Gráficos de usuario. 273 Apéndice 5: Teclados. 275 Apéndice 6: Códigos de error. 277 0: OK. 277 1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 H: STOP in INPUT. 281 I: FOR without NEXT. 281 J: Invalid //O device. 281 K: Invalid colour. 281 L: BREAK into program. 282 M: RAMTOP	Variables en memoria	260
Pilas: cálculo, sistema y GO SUB. 269 RAMTOP. 272 Gráficos de usuario. 275 Apéndice 5: Teclados. 275 Apéndice 6: Códigos de error. 277 0: OK. 277 1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 I: FOR without NEXT. 281 J: Invalid I/O device. 281 K: Invalid Colour. 281 L: BREAK into program. 281 M: RAMTOP no good.	Zona de edición v entrada con INPUT	.268
RAMTOP 272 Gráficos de usuario 273 Apéndice 5: Teclados 275 Apéndice 6: Códigos de error 277 0: OK 277 1: NEXT without FOR 277 2: Variable not found 277 3: Subscript wrong 278 4: Out of memory 278 5: Out of screen 278 6: Number too big 278 7: RETURN without GO SUB 278 8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 <	Pilas: cálculo, sistema v GO SUB.	.269
Gráficos de usuario. 273 Apéndice 5: Teclados. 275 Apéndice 6: Códigos de error. 277 0: OK. 277 1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 H: FOR without NEXT. 281 J: Invalid colour. 281 L: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid s	RAMTOP	272
Apéndice 5: Teclados	Gráficos de usuario	273
0: OK. 277 1: NEXT without FOR 277 2: Variable not found 277 3: Subscript wrong 278 4: Out of memory 278 5: Out of screen 278 6: Number too big 278 7: RETURN without GO SUB 278 8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 O: Invalid stream 282 O: Invalid stream 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 <	Apéndice 5: Teclados.	275
0: OK. 277 1: NEXT without FOR 277 2: Variable not found 277 3: Subscript wrong 278 4: Out of memory 278 5: Out of screen 278 6: Number too big 278 7: RETURN without GO SUB 278 8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 O: Invalid stream 282 O: Invalid stream 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 <	Apéndice 6: Códigos de error	277
1: NEXT without FOR. 277 2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 J: FOR without NEXT. 281 J: FOR without NEXT. 281 J: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid stream. 282 P: FN without DEF. 282 Q: Parameter error. 283 R: Tape loading error. 283 Apéndice 7: Las variables del sistema. 285 Apéndice 8:	0: OK	277
2: Variable not found. 277 3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 I: FOR without NEXT 281 J: Invalid I/O device. 281 K: Invalid colour. 281 L: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid stream. 282 P: FN without DEF. 282 Q: Parameter error. 283 R: Tape loading error. 283 Apéndice 7: Las variables del sistema. 285 Apéndice 8:	1: NFXT without FOR.	277
3: Subscript wrong. 278 4: Out of memory. 278 5: Out of screen. 278 6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 I: FOR without NEXT. 281 J: Invalid I/O device. 281 K: Invalid colour. 281 L: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid stream. 282 P: FN without DEF. 282 Q: Parameter error. 283 R: Tape loading error. 283 Apéndice 7: Las variables del sistema. 285 Apéndice 8: La interrupción enmascarable. 293	2: Variable not found	277
4: Out of memory 278 5: Out of screen 278 6: Number too big 278 7: RETURN without GO SUB 278 8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 O: Invalid stream 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	3: Subscript wrong.	278
5: Out of screen 278 6: Number too big 278 7: RETURN without GO SUB 278 8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	4. Out of memory	278
6: Number too big. 278 7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 I: FOR without NEXT. 281 J: Invalid I/O device. 281 K: Invalid colour. 281 L: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid stream. 282 P: FN without DEF. 282 Q: Parameter error. 283 R: Tape loading error. 283 Apéndice 7: Las variables del sistema. 285 Apéndice 8: La interrupción enmascarable. 293		
7: RETURN without GO SUB. 278 8: End of file. 278 9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 I: FOR without NEXT. 281 J: Invalid I/O device. 281 K: Invalid colour. 281 L: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid stream. 282 O: Invalid stream. 282 Q: Parameter error. 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293		
8: End of file 278 9: STOP statement 279 A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	7: RETURN without GO SUB	.278
9: STOP statement. 279 A: Invalid Argument. 279 B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 I: FOR without NEXT. 281 J: Invalid I/O device. 281 K: Invalid colour. 281 L: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid stream. 282 P: FN without DEF. 282 Q: Parameter error. 283 R: Tape loading error. 283 Apéndice 7: Las variables del sistema. 285 Apéndice 8: La interrupción enmascarable. 293	8: End of file	278
A: Invalid Argument 279 B: Integer out of range 279 C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293		
B: Integer out of range. 279 C: Nonsense in BASIC. 280 D: BREAK - CONT repeats. 280 E: Out of DATA. 280 F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 I: FOR without NEXT. 281 J: Invalid I/O device. 281 K: Invalid colour. 281 L: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid stream. 282 P: FN without DEF. 282 Q: Parameter error. 283 R: Tape loading error. 283 Apéndice 7: Las variables del sistema. 285 Apéndice 8: La interrupción enmascarable. 293	A: Invalid Argument	279
C: Nonsense in BASIC 280 D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	B: Integer out of range	279
D: BREAK - CONT repeats 280 E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	C: Nonsense in BASIC	.280
E: Out of DATA 280 F: Invalid file name 281 G: No room for line 281 H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	D: BRFAK - CONT repeats.	280
F: Invalid file name. 281 G: No room for line. 281 H: STOP in INPUT. 281 I: FOR without NEXT. 281 J: Invalid I/O device. 281 K: Invalid colour. 281 L: BREAK into program. 281 M: RAMTOP no good. 282 N: Statement lost. 282 O: Invalid stream. 282 P: FN without DEF. 282 Q: Parameter error. 283 R: Tape loading error. 283 Apéndice 7: Las variables del sistema. 285 Apéndice 8: La interrupción enmascarable. 293	F: Out of DATA	280
G: No room for line	F: Invalid file name	281
H: STOP in INPUT 281 I: FOR without NEXT 281 J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	G: No room for line	281
J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	H: STOP in INPUT	281
J: Invalid I/O device 281 K: Invalid colour 281 L: BREAK into program 281 M: RAMTOP no good 282 N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	I: FOR without NEXT	.281
K: Invalid colour	J: Invalid I/O device	281
L: BREAK into program	K: Invalid colour.	281
M: RAMTOP no good	L: BREAK into program	.281
N: Statement lost 282 O: Invalid stream 282 P: FN without DEF 282 Q: Parameter error 283 R: Tape loading error 283 Apéndice 7: Las variables del sistema 285 Apéndice 8: La interrupción enmascarable 293	M: RAMTOP no good	282
O: Invalid stream	N: Statement lost	282
P: FN without DEF		
Q: Parameter error	P: FN without DFF	282
R: Tape loading error	Q: Parameter error	.283
Apéndice 7: Las variables del sistema	R: Tape loading error	283
Apéndice 8: La interrupción enmascarable 293 Interrupción enmascarable 294	Apéndice 7: Las variables del sistema	.285
Interrupción enmascarable294	Apéndice 8: La interrupción enmascarable	.293
	Interrupción enmascarable	.294

Contador de cuadros	295
Enmascaramiento	
Apéndice 9: Índice del lenguaie BASIC	297
Bibliografía y recursos en línea	301

Introducción

Bienvenido a este libro sobre programación BASIC en el ZX Spectrum.

El objetivo de la presente obra es tratar los mismos conceptos que el manual oficial que acompañaba al *Spectrum* en los años 80. Si bien se tratan los mismos temas, el libro que tiene en sus manos tiene un enfoque distinto al del manual original.

En el desarrollo de los distintos temas se ha intentado **profundizar todo lo posible**, de tal modo que, aunque puede resultar algo abrumador para los principiantes, se han redactado las explicaciones de la forma más clara posible. De esta manera, tanto principiantes como lectores avanzados hallarán aquí todo lo que necesiten. Para un principiante no será necesario comprender cada uno de los tópicos expuestos para seguir avanzando en su lectura.

Si bien no se asume experiencia previa en programación, no estaría de más cierta familiaridad con el funcionamiento del *Spectrum*. Es altamente recomendable saber cómo introducir comandos, escribir programas, editar y eliminar líneas del programa BASIC o ejecutar el código fuente. El libro *ZX Spectrum: Introducción actualizada y ampliada* o la misma guía de introducción original proporcionan este tipo de información, aunque también existen numerosos tutoriales al respecto. Alternativamente, puede resultar útil el apéndice 1 de este libro, que ofrece una breve introducción al funcionamiento básico del teclado, la entrada de comandos y la edición y ejecución de programas.

El apéndice 5 muestra las disposiciones de teclado del *Spectrum 16K/48K* y del *Spectrum+*, de tal modo que se pueda practicar todo lo aquí expuesto, incluso en el caso de no disponer de un *Spectrum* físico. En este último caso, resultará útil la información sobre mapeo de teclas en emuladores, incluida en el apéndice 1.

Hablando del lenguaje BASIC, se puede afirmar que es idéntico en los modelos de 16 KB y 48 KB, así como en el *Spectrum+*. Las únicas diferencias estriban en la cantidad de memora RAM disponible, forzosamente distinta en el *16K* que en el *48K*.

En cuanto al *Spectrum*+, el funcionamiento es idéntico a sus dos antecesores. Por lo tanto, todo lo dicho aquí es igualmente válido para el *Spectrum 16K*, el *Spectrum 48K* y el *Spectrum*+.

La ROM original, que marca el comportamiento del ordenador, es idéntica en los tres modelos mencionados.

Tanto si se dispone de un *Spectrum* como de un *Spectrum+*, o incluso si solo se dispone de un emulador *software*, la información contenida en este manual será de gran utilidad a quien desee zambullirse en el mundo de la programación retro.

El comportamiento del *Spectrum* viene específicamente establecido por una sola aplicación, escrita «a fuego» en su memoria ROM. Esta aplicación es análoga al sistema operativo de un ordenador: contiene rutinas de bajo nivel que interactúan con los periféricos y recursos, utilidades, funcionalidad de instrucciones y funciones BASIC, y un entorno de programación que incluye un editor de comandos y programas, así como un intérprete BASIC.

La ejecución de un programa BASIC es, en realidad, el intérprete en funcionamiento, que forma parte de esta aplicación única que se está ejecutando en todo momento. Aunque es posible que deje de ejecutarse este «sistema operativo» forzando un salto hacia una zona de memoria donde exista un programa debidamente ensamblado, esto queda fuera del alcance de este manual.

Instrucciones, funciones, sentencias y comandos

Estos términos pueden dar pie a confusión si no se utilizan de forma consistente. Por lo tanto vamos a exponer cómo se usan en este manual.

En primer lugar, el lenguaje BASIC del *Spectrum* dispone de distintas **instrucciones** y **funciones**, expresadas mediante ciertas palabras o *tokens* del lenguaje, fácilmente utilizables por parte del usuario. Las instrucciones no retornan valor alguno, mientras las funciones sí lo hacen. Ejemplos de instrucciones son PRINT, INPUT O READ; algunas funciones son COS, RND O VAL.

Mediante estas instrucciones y funciones se pueden teclear **comandos**. Estos sirven para dar órdenes directas al sistema, sin necesidad de escribir un programa. Un ejemplo de comando:

```
PRINT "Buenas tardes, ";a$
```

Una misma línea puede incluir varios comandos, separados por dos puntos (:).

Por último, si en lugar de introducir un comando lo escribimos en una línea de programa, tendremos una **sentencia**. Básicamente, es lo mismo que un comando; solo cambia su ubicación y modo de introducirlo:

```
10 PRINT "Buenas tardes, ";a$
```

Una línea de programa puede incluir varias sentencias, separadas también por dos puntos.

Definición sintáctica

A la hora de definir la sintaxis correspondiente a las distintas funciones e instrucciones, utilizaremos un mismo formato a lo largo de todo el libro para indicar inequívocamente los distintos elementos que componen dicha definición:

 Los elementos entre comillas angulares (<>) indican que el texto en el interior de las comillas no es literal, sino la descripción del objeto en sí.

Introducción

- Los corchetes ([1]) indican que lo que hay en su interior es opcional. Los corchetes en sí no deben escribirse en el comando o sentencia.
- Los tres puntos (...) en una lista con elementos numerados indican una cantidad arbitraria de dichos elementos, aunque como mínimo deberá haber uno, a no ser que la lista esté entre corchetes, en cuyo caso podría no haber ninguno de ellos. El carácter separador (o posibles caracteres separadores) de los elementos se especifica justo antes de los tres puntos.
- En caso de haber múltiples opciones, se agrupa la lista de posibilidades dentro de llaves (£2) si es obligatorio incluir una, o dentro de corchetes (£1) si es opcional su inclusión. En ambos casos, las opciones quedan separadas entre sí por barras verticales (1).
- El resto sí es literal. Esto incluye paréntesis (;), comas (,) y cualquier otro signo o carácter escrito.

Los elementos sintácticos se especifican así:

Variables:

- «var» Identificador de variable de cualquier tipo.
- <v> Identificador de tipo numérico de una sola letra.

Valores (expresiones):

- «Exp» Expresión que produce un valor de cualquier tipo.
- (N) Expresión de tipo numérico (entero o punto flotante).
- (I) Expresión numérica entera.
- <s> Expresión de tipo string.

Otros elementos:

- <Eoc> Expresión o controlador (de posición o color).
- ΚΕ C V L > Expresión, controlador, variable o elemento LINE.
- <co u c > Controlador de color.
- Texto > Sucesión arbitraria de caracteres.
- <st>
 Sentencia completa.
- <stc> Slicer.

Las **listas de elementos numerados** añadirán un símbolo guión bajo (_) y un número de índice a cada elemento. Si el número de elementos es arbitrario, k será el índice del último de ellos. Por ejemplo, una lista arbitraria de expresiones empezaría por el elemento <Exp_1> y acabaría con <Exp_k>. Nos referiremos a estos elementos como <Exp_i> genéricamente.

En una lista con un número arbitrario de elementos, deberá haber como mínimo uno de ellos, a no ser que la lista esté entre corchetes

([]), en cuyo caso podría no haber ninguno.

Los parámetros pueden tener, en lugar de un número de índice, una letra que especifique mejor su uso, como por ejemplo <n_x> o <n_y>.

En el caso de que una instrucción o función espere recibir como argumento un valor numérico entero (< x >) y le pasemos un valor de punto flotante, se redondeará automáticamente al entero más próximo. Es también el caso de los índices y *slicers*.

Para pasar un número de punto flotante al entero más próximo, lo que hace el intérprete internamente es sumar 0,5 al número y seguidamente truncar al entero más bajo (aplicando la función INT). Por ejemplo: 3,5 se redondearía al número 4, mientras que -3,5 se redondearía a -3.

Por ejemplo, en el caso de LET, la sintaxis se define como:

```
LET <var>[(<Ios>)] = <Exp>
```

Aquí, LET debe escribirse tal cual, mientras que <var>, <Ios> y</Ios> y</Ios> representan una variable, un índice o *slicer*, y una expresión respectivamente, con lo que no deben entenderse de forma literal. En cuanto a I(<Ios>) indica que el fragmento (<Ios>) es opcional, pero de estar presente, se deben teclear los paréntesis y, en su interior, el índice o *slicer*.

En este ejemplo:

Podemos ver que existe un número arbitrario de elementos $\langle \mathbf{I_{i}} \rangle$ separados por comas (,), aunque como mínimo habrá un elemento ($\langle \mathbf{I_{1}} \rangle$).

Veamos una definición algo más compleja:

A simple vista puede parecer un poco confuso, pero en realidad no lo es tanto: en primer lugar, podemos comprobar que la lista de parámetros consiste básicamente en una lista numerada encerrada entre corchetes, lo cual significa que puede haber un número arbitrario de elementos o ninguno en absoluto.

En segundo lugar, vemos que todos los elementos de la lista (<Eo c $_i>$) son del tipo «expresión o controlador», y que son todos opcionales.

También podemos ver que los separadores de los elementos de la lista pueden ser un punto y coma (;), una coma (,) o un apóstrofe ().

Todo esto significa que pueden haber varios separadores consecutivos sin elemento alguno entre ellos.

Finalmente, dado que el último elemento de la definición es también opcional, la sentencia puede terminar tanto en separador como en un elemento «Eo c ».

Bases numéricas

Los números corrientes (en la acostumbrada base 10), tanto enteros como fraccionarios, se presentan tal cual, utilizándose la coma (,) como separador de decimales: 234 (entero) o 23,4 (fraccionario). También pueden presentarse con el tipo de letra del *Spectrum* cuando forman parte del código fuente (234 o 23,4).

Cuando los números no están en base 10, se les incluye un subíndice que indica la base: 11101010₂ (binario), 352₈ (octal), o EA₁₆ (hexadecimal). En otros libros o artículos se suelen indicar prefijos (del tipo 0b- o 0x-) o sufijos (como -b o -h) para indicar este tipo de números.

Estructura del manual

Los conceptos y temas se han estructurado de tal forma que la lectura de los capítulos se pueda realizar en orden. Dicho orden está altamente recomendado para quien no tenga experiencia en programación o en el lenguaje BASIC del *ZX Spectrum*. Los más expertos pueden consultar los capítulos y apéndices en el orden que precisen.

Para quien no esté familiarizado con el mundo de la programación, los apéndices proporcionan información que puede resultar muy útil. También contienen conceptos generales que no están directamente relacionados con el lenguaje BASIC. Esto incluye información de referencia sobre aspectos adicionales que no resultan estrictamente necesarios para comprender el lenguaje del *Spectrum*.

A continuación se proporciona un breve resumen de los distintos capítulos del libro:

- Capítulo 1: Tipos de datos: Se detallan los diferentes tipos de datos con los que trabaja el lenguaje BASIC del Spectrum, así como el concepto y funcionamiento de las variables, expresiones y operadores. Se explican las instrucciones ▷IM Y LET, así como las funciones LEN, STR\$, UAL Y UAL\$.
- Capítulo 2: Primeras instrucciones y funciones: Se ven algunas instrucciones (CLEAR, CONTINUE, NEW, REM, RUN y STOP) y funciones (CHR # y CODE) útiles para empezar a escribir programas.
- Capítulo 3: La pantalla: Se detalla el funcionamiento de la escritura de caracteres en pantalla, incluyendo las instrucciones
 LIST Y PRINT.
- Capítulo 4: Color y dibujo: Se explica cómo funcionan los atributos de color en el Spectrum, así como las instrucciones que tratan con el dibujo de píxeles en pantalla. Las instrucciones de este capítulo son BORDER, BRIGHT, CIRCLE, CLS, DRAW, FLASH, INK, INVERSE, OVER, PAPER Y PLOT. Las funciones explicadas son ATTR, INT, PI, POINT Y SCREEN\$.
- Capítulo 5: El teclado: Se explica cómo obtener la entrada del usuario a través del teclado. Se ve la instrucción INPUT y la función INKEY\$.
- Capítulo 6: Impresora, microdrive y canales: Explicaciones sobre la funcionalidad de los canales del Spectrum, el tratamiento de la impresora y algunos apuntes sobre los dispositivos de la unidad ZX Interface 1. Se habla sobre las instrucciones CAT, ERASE, FORMAT Y MOUE, y se detalla la funcionalidad

- de las instrucciones cLose #, COPY, LLIST, LPRINT y OPEN #.
- Capítulo 7: Bifurcaciones, bucles y saltos: Se explican las distintas maneras de modificar el flujo de ejecución de un programa, así como la forma de tomar decisiones que afectan a dicho flujo en tiempo de ejecución. Se ven las instrucciones FOR, GO SUB, GO TO, IF, NEXT, PAUSE, RANDOMIZE Y RETURN, así como la función RND.
- Capítulo 8: Series de datos: Se explica cómo tratar de forma más eficiente los volúmenes de datos de mayor tamaño. Se detallan las instrucciones BEEP, DATA, READ Y RESTORE.
- Capítulo 9: Funciones: Se detallan el resto de funciones no vistas hasta el momento, así como la técnica utilizada para definir funciones a medida. Esto incluye la instrucción DEF FN y las funciones ABS, ACS, ASN, ATN, COS, EXP, FN, LN, SGN, SIN, SQR, TAN V USR.
- Capítulo 10: Memoria y periféricos: Se explica cómo acceder a la memoria del Spectrum, así como el intercambio de información con la cinta de casete y con los puertos de entrada/salida de otros periféricos. Se ven las instrucciones LOAD, MERGE, OUT, POKE, SAVE Y VERIFY, así como las funciones IN Y PEEK.
- Capítulo 11: Nuevas posibilidades: Se consideran las distintas novedades tecnológicas que existen hoy día alrededor del mundo del Spectrum, tanto de tipo software como hardware, que no estaban disponibles en los años 80.

En cuanto a los apéndices, este es su contenido:

- Apéndice 1: Teclado, comandos y pantalla: Se trata de un breve resumen del libro ZX Spectrum: Introducción actualizada y ampliada. Son las bases para poder utilizar el Spectrum de forma básica.
- Apéndice 2: Juego de caracteres: Se detalla una lista completa del juego de caracteres del Spectrum, así como una explicación de los diversos tipos de caracteres que lo forman.
- Apéndice 3: Sistema binario y hexadecimal: Explicación detallada del funcionamiento de los métodos de numeración más utilizados en informática.
- Apéndice 4: Gestión de memoria: Se detalla el uso que hace el Spectrum de las diferentes zonas de memoria.
- Apéndice 5: Teclados: Guía visual de los teclados del Spectrum y Spectrum+.

- Apéndice 6: Códigos de error: Enumeración de los códigos de finalización de sentencias o comandos.
- Apéndice 7: Variables del sistema: Referencia de las variables del sistema, con una breve explicación de cada una de ellas.
- Apéndice 8: La interrupción enmascarable: Explicación sobre la interrupción enmascarable y su función dentro del sistema.
- Apéndice 9: Índice del lenguaje BASIC: Lista completa de palabras clave (tokens) pertenecientes al lenguaje BASIC del Spectrum.

Capítulo 1: Tipos de datos

No son muchos los tipos de datos que puede manejar el *Spectrum*. Básicamente disponemos de los tipos numéricos (enteros y coma flotante) y las cadenas de caracteres (*strings*). En este capítulo hablaremos en detalle sobre ambos tipos y cómo se almacenan en memoria.

Por otro lado, existe la posibilidad de agrupar valores dentro de una misma variable, ya sea mediante una secuencia lineal o una matriz de dos o más dimensiones. Son los llamados *arrays*.

También veremos algunas funciones utilizadas para convertir datos de un tipo a otro.

Tipos numéricos

Los números se almacenan de dos formas distintas: por un lado, en forma de enteros. Por otro lado, como números de coma flotante (también llamados de «punto flotante»).

Números enteros

Cuando el sistema identifica un número como entero, es decir, sin posiciones decimales, almacena la magnitud de dicho número en dos *bytes*. Es decir, los enteros del *Spectrum* tienen 16 *bits*.

Para una breve explicación sobre la expresión de números usando el sistema binario (y el hexadecimal), véase el apéndice 3.

Para entender cómo gestiona el *Spectrum* la memoria RAM y cómo representa los distintos tipos de datos, véase el apéndice 4.

Para que el número se almacene de este modo, no solo tiene que estar el sistema seguro de que se trata de un entero (sin punto decimal). Además, este debe estar comprendido entre -65535 y 65535, ambos inclusive.

El número 0 solo puede representarse como entero, y por ello siempre se codifica usando este formato.

Por ejemplo, como se verá en detalle más adelante, la instrucción LET asigna un valor a una variable:

```
LET a=3+4
LET b=3.5+3.5
LET c=3.0
LET d=3e4
LET e=.3e4
```

En estos ejemplos, todos los valores numéricos podrían considerarse enteros, pero no siempre serán identificados (y codificados) como tal. Aquí las variables a, c y d se almacenan como enteros, mientras que b y e lo son como números de punto flotante.

Por lo tanto, a no ser que necesitemos tener un control absoluto sobre la representación de nuestros números en memoria, lo mejor es olvidarse de ello y dejar que el *Spectrum* se encargue de codificar los números como crea conveniente.

Un **literal entero** se representa mediante una secuencia de dígitos del 0 al 9, con un carácter inicial de signo (- o +) opcional (si se omite, se entiende positivo).

Por otro lado, existe la posibilidad de indicar un valor entero mediante un **literal entero binario**. Esto se hace mediante el indicador de literal binario BIN. Este indicador debe ir seguido de cero o más dígitos Ø y 1, siempre y cuando el valor especificado no supere 65535. No es posible indicar de este modo un número negativo ni fraccionario. BIN sin un número después equivale a 0. Ejemplos: BIN 10010, BIN Ø, BIN Ejemplos no válidos: BIN 21, BIN a\$.

Números de punto flotante

Cuando un número incluye posiciones decimales o está fuera del rango [-65535, 65535], su representación en memoria se realiza siempre en formato de punto flotante.

Este tipo de número acepta valores positivos desde aproximadamente 2,94×10⁻³⁹ (2⁻¹²⁸) hasta alrededor de 1,70×10³⁸ (2¹²⁷). El rango de negativos es análogo: desde aproximadamente -2,94×10⁻³⁹ (-2⁻¹²⁸) hasta alrededor de -1,70×10³⁸ (-2¹²⁷).

El problema de la representación en punto flotante es la precisión disponible. Dado que los valores de referencia que usamos los humanos suele estar basado en el sistema decimal (base 10), en el formato de punto flotante, basado en el sistema binario, no siempre pueden representarse de forma exacta todos los números decimales. Esto, sin embargo, no representa ningún problema para el tipo de aplicaciones que se utilizan en el *Spectrum*, ya que es una precisión más que suficiente para la gran mayoría de casos.

Cualquier entero de hasta 32 bits, tanto positivo como negativo (-4294967295 a 4294967295), puede representarse con precisión absoluta mediante el formato de punto flotante. Fuera de ese rango, algunos números pueden codificarse con pequeños errores de precisión.

En cuanto a los números con posiciones decimales, el problema de representación se soluciona mediante redondeo. Por ejemplo, el número 0,8 no es representable de forma exacta: el valor almacenado es algo así como 0,799999998137355, aunque el *Spectrum* lo redondeará a 0,8 cuando toque representarlo.

Un **literal de punto flotante** es ligeramente más complejo que uno entero: empieza también con un carácter de signo opcional, y tiene una parte entera y una fraccionaria, separadas por un punto (.). Si la parte entera es cero, no es obligatorio incluirla. Son ejemplos 5.54, -8.333 o .75. También pueden representarse mediante notación científica, es decir, un número (entero o de punto flotante) seguido de un carácter e (o E) y un número entero (positivo o negativo). El valor que representa es el del primer número multiplicado por 10 elevado al segundo número. Por ejemplo, 55e -2 es igual a 56 multiplicado por 10

elevado a -2 (56×10⁻²), es decir, 0,56. Por otro lado, -45.998e4 sería igual a -469980 (46,998×10⁴).

Cadenas de caracteres (strings)

El *ZX Spectrum* utiliza una tabla de caracteres basada en el estándar ASCII y adaptada a sus necesidades específicas. Dicha tabla debe contener los símbolos, números, letras, caracteres gráficos y otros elementos necesarios para el buen funcionamiento del sistema.

Un *string* o cadena de caracteres no es más que una secuencia de caracteres agrupados de forma secuencial. Una de las formas de expresar un *string* es mediante un **literal**, consistente en una serie de caracteres entre comillas dobles.

Cada carácter tiene asociado un código entre 0 y 255, de tal modo que puede representarse con un solo *byte*.

Literal string

Para crear un literal de tipo *string*, es necesario **escribir los elementos del mismo**, **encerrados entre comillas dobles** ("). En el caso de desear que el carácter de las comillas dobles aparezca dentro del literal, **hay que indicarlas dos veces consecutivas** en su interior:

PRINT "Yo digo ""hola mundo"""

La instrucción PRINT escribe en pantalla la información que se le proporciona. Se verá con detalla más adelante.

En este caso, dado que deseamos que el fragmento hola mundo vaya entrecomillado, debemos incluir dos comillas delante y dos al final de esas dos palabras. Dado que las dos del final van justo antes de las comillas de cierre del *string* completo, escribiremos las tres comillas seguidas.

Un *string* puede contener cualquier tipo de elemento del juego de caracteres: tanto caracteres comunes como caracteres de control de color o posición, *tokens* (nombres de instrucciones y funciones BASIC) o caracteres gráficos.

Véase el apéndice 2 para una definición completa del juego de caracteres del *Spectrum*.

A la hora de definir un literal *string* a través del teclado, hay algunos caracteres que no pueden teclearse dentro de las comillas dobles. En tal caso, deberá usarse la función CHR \$.

Para saber cómo teclear caracteres de control de color, véase el capítulo 4.

El tamaño máximo que podemos darle a un literal *string* dependerá de la memoria RAM disponible. Es importante señalar que estos literales se almacenan, como mínimo, dos veces: una en el código fuente del programa, y una vez realizada la asignación a una variable (durante la ejecución), también en la zona de variables. Evidentemente, si se asigna a más de una variable, se almacenará más veces en esa zona de variables.

Slicing de strings

Dentro de un *string* es posible hacer referencia a un solo carácter o a un fragmento del mismo (llamado un *substring*). El *substring* más pequeño de un *string* cualquiera es el *string* vacío, es decir, sin caracteres. El más largo posible es el *string* original en sí.

Para «trocear» un *string* para obtener un carácter o fragmento, se debe indicar, tras el *string* (variable, literal o expresión *string*) y entre paréntesis, **un índice** (un solo valor numérico) o bien **un rango** entre dos valores numéricos. En este último caso, se utilizará un *slicer*, esto es, esos valores numéricos separados por el *token* To.

Veamos un ejemplo:

```
10 LET a$="Hasta luego, mundo."
20 PRINT a$(4)
30 PRINT a$(7 TO 11)
40 LET a$=a$(14 TO 18)
50 PRINT a$(3)
```

Esto producirá el siguiente resultado:

```
t
luego
n
```

Los índices y valores de rangos empiezan en 1, a diferencia de otros lenguajes de programación, en los que el primer carácter de un *string* tiene índice 0.

Referencias a índices o *slicers* fuera del rango correcto (posiciones inferiores a 1, o superiores al tamaño del *string*) producen un error 3 (Subscript wrong).

La primera sentencia del ejemplo asocia el string "Hasta Luego, mundo." a la variable a \$. La segunda línea escribe el cuarto (4) carácter de dicho string ("t") en pantalla. La línea 30 escribe el substring formado por los caracteres del 7 al 11, es decir, "Luego". A continuación, la variable a \$ pasa a contener un nuevo string: este es el substring del string inicial, compuesto por los caracteres del 14 al 18, es decir, "mundo". La última sentencia escribe el tercer (3) carácter de ese nuevo string ("n").

Como vemos, un índice (un solo valor numérico) hace referencia a una posición dentro del *string*, y no puede ser inferior a 1 ni superior a la longitud total del *string*.

En cuanto a un *slicer*, se define a modo de <valor1> To <valor2>. En este caso, tanto <valor1> como <valor2> deben hacer referencia también a una posición válida (entre 1 y la longitud total). En caso de que valor2 sea inferior a valor1, se obtendrá un *string* vacío (con cero caracteres). Si son iguales se obtendrá un solo carácter.

Tanto los índices como los valores de los *slicers* pueden ser variables numéricas, literales numéricos o expresiones numéricas (véase la definición de expresión, más adelante). En caso de ser valores de punto flotante, se realiza un redondeo automático al entero más cercano antes de obtener el *substring*.