6

Este é um livro fundamental na bibliografia sobre o microcomputador Spectrum. Nas suas páginas, o Dr. Ian Logan e o Dr. Frank O'Hara explicam o que faz funcionar o Spectrum, analisando exaustivamente as diferentes rotinas do 16K ROM. Entre estas, salientamos a rotina *restart* e tabelas, as rotinas do teclado, do altifalante, do tratamento de cassetes, do visor e impressora, de execução, aritméticas, de avaliação e outras. Em Apêndice, são fornecidos alguns programas em Basic, os algoritmos DRAW e CIRCLE e ainda uma nota sobre os inteiros pequenos e – 65536, assim como um índice de rotinas.

SISTEMA OPERATIVO DO SPECTRUM
Ian Logan
e Frank O'Hara

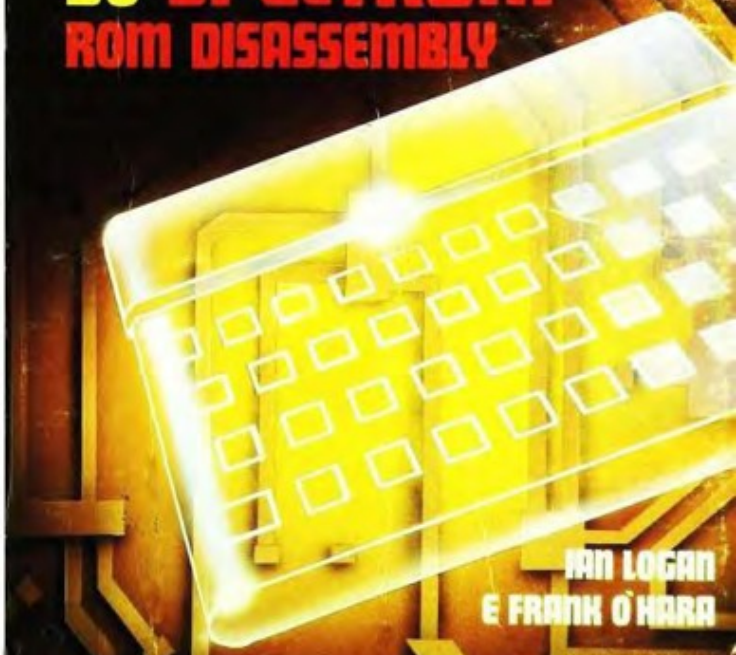
COLECÇÃO **SISTEMAS**

1. A INFORMÁTICA NA ESCOLA
   Manual de Utilização do ZX Spectrum
   (e tc 2068), *Luís de Campos*
2. GUIA DOS MICROPROCESSADORES
   E. A. Parr
3. INICIAÇÃO À BASE DE DADOS
   François Fargette
4. PROGRAMAÇÃO DE COMPUTADORES
   EM PASCAL *David Lightfoot*
5. OS SISTEMAS OPERATIVOS
   A M. Lister
6. O SISTEMA OPERATIVO DO SPECTRUM
   ROM DISASSEMBLY, *Ian Logan*
   e Frank O' Hara

EDITORIAL PRESENÇA

COLECÇÃO SISTEMAS

O SISTEMA OPERATIVO DO SPECTRUM
ROM DISASSEMBLY



IAN LOGAN
E FRANK O'HARA

PRESENÇA

COLECÇÃO SISTEMAS

O SISTEMA OPERATIVO DO SPECTRUM

Rom Disassembly

Ian Logan e Frank O'Hara

O SISTEMA OPERATIVO DO SPECTRUM
Rom Disassembly

EDITORIAL PRESENÇA

O Sinclair ZX Spectrum é um importante sucessor do ZX 81, que por sua vez, substituiu o ZX 80.

O Spectrum possui um programa monitor ocupando 16 K em ROM. Este programa foi desenvolvido directamente a partir do programa de 4 K do ZX 80, se bem que contenha tantas características novas que as diferenças se sobrepõem às semelhanças.

Ambos tivemos bastante prazer na preparação deste livro. Aprendemos sobre as técnicas de programação em código-máquina Z 80, e pensamos ter descoberto os «segredos do Spectrum».

Gostaríamos de agradecer:

— Às nossas famílias.

— A Alfred Milgrom, o nosso editor, que nos foi extremamente útil.

— A Philip Mitchell, cujas notas sobre o formato para cassete foram muito ilustrativas.

— A Clive Sinclair e à sua equipa da Sinclair Research Ltd., que produziram uma máquina tão útil.

Janeiro de 1983

Ian Logan — Lincoln, Reino Unido
Frank O'Hara — Londres, Reino Unido

FICHA TÉCNICA

Título original: *THE COMPLETE SPECTRUM ROM DISASSEMBLY*
Autores: *Dr. Ian Logan e Dr. Frank O'Hara*

Edição publicada por acordo com Melbourne House (Publishers) Ltd., London
Tradução: *Eduardo Nogueira*
Capa: *António Marques*
Fotocomposição, paginação e fotolitos: *Textype - Artes Gráficas — Lisboa*
Impressão e acabamento: *Tipografia Guerra — Viseu*
1.ª Edição, Lisboa, 1986

O programa monitor de 16 K do Spectrum é um complexo programa em código-máquina Z 80. A sua estrutura global é bastante clara, dividindo-se em três partes principais:

a) Rotinas de entrada/saída.
b) Interpretador BASIC.
c) Tratamento de expressões.

No entanto, estes blocos são demasiado grandes para poderem ser tratados facilmente, e, de facto, neste livro optámos por discutir o monitor em dez partes.

Vamos esboçar agora cada uma delas.

Rotinas de «Restart» e tabelas

No início do programa monitor encontram-se as várias rotinas de «restart», que são invocadas por instruções «RST» ocupando um único byte. Todos estes «restarts» são usados. Por exemplo, usa-se «Restart 0008» para indicação de erros de sintaxe ou execução.

As tabelas contidas nesta parte do programa monitor contêm as formas por extenso das palavras-chave e os «códigos de teclas».

A rotina de teclado

O teclado é lido cinquenta vezes por segundo (modelo europeu), servindo esta rotina para aceitar o código de caracter apropriado. Todas as teclas do teclado «repetem» se forem premidas continuamente, e a rotina de teclado toma este facto em consideração.

As rotinas do altifalante

O Spectrum possui um único altifalante interno, sendo produzida uma nota usando repetidamente a instrução «OUT» apropriada. Na rotina de controlo foram tomados bastantes cuidados no sentido de garantir que a nota mantenha a mesma tonalidade durante toda a sua duração.

As rotinas de tratamento de cassete

Uma característica bastante infeliz do ZX 81 era o facto de ter sido dedicada uma secção tão reduzida do seu programa monitor ao tratamento de cassetes.

No entanto, no caso do Spectrum, é-lhe dedicada uma extensa codificação, e, de facto, a elevada qualidade do tratamento de cassetes é agora uma das características de maior êxito nesta máquina.

Os programas ou blocos de dados BASIC são tratados da mesma maneira, dispondo de um bloco de «cabeçalho» *(header)* contendo dezassete bytes que é gravado primeiro. Este cabeçalho descreve o bloco de dados que se encontra gravado a seguir.

Uma desvantagem deste sistema é que não é possível produzir programas com qualquer «protecção».

As rotinas de tratamento do visor e da impressora

Todas as rotinas de entrada/saída restantes do Spectrum são passadas pelas «áreas de informação de canal e 'stream'».

No Spectrum *standard* só é possível a entrada por teclado, mas a saída pode ser dirigida para a impressora, a parte superior do visor de uma televisão ou a parte inferior deste.

A principal rotina de entrada desta parte do programa monitor é o EDITOR, que permite ao utilizador dar entrada a caracteres directamente para a parte inferior do visor.

A rotina PRINT-OUT é bastante lenta, dado que tem em conta todas as possibilidades. Por exemplo, o acrescento de um único byte à «área de imagem» envolve a consideração do estado actual de OVER e INVERSE em todos os casos.

As rotinas de execução

Nesta parte do programa monitor encontram-se o procedimento de INICIALIZAÇÃO e o «ciclo principal de execução» do interpretador BASIC.

No Spectrum a linha Basic produzida pelo EDITOR é verificada em termos sintácticos e gravada na área de programa, se se trata de uma linha iniciada por um número de linha; no caso contrário, será executada automaticamente.

Esta execução pode conduzir por sua vez à consideração de novas instruções (o que se observa mais claramente no caso de RUN).

Interpretação de linhas e comandos BASIC

Esta parte do programa monitor considera uma linha Basic como um conjunto de instruções, e por sua vez, cada uma destas, como iniciando-se por um determinado comando. Para cada um destes comandos (palavras-chave) existe uma «rotina de comando», sendo a execução do código-máquina da «rotina de comando» apropriada que efectua a interpretação.

Avaliação de expressões

O Spectrum possui um avaliador de expressões muito completo, permitindo uma vasta gama de tipos de variáveis, funções e operações. Esta parte do monitor é também bastante lenta dada a quantidade de alternativas que devem ser consideradas.

O tratamento de cadeias é particularmente bem conseguido. Todas as cadeias simples são tratadas «dinamicamente», sendo as cópias antigas «reclamadas» depois de se tornarem redundantes. Isto significa que não é necessário «deitar fora o lixo».

As rotinas aritméticas

O Spectrum possui duas formas de representar números. Os valores inteiros na gama -65535 a +65535 utilizam uma forma «inteira» ou «curta», enquanto todos os outros são representados em vírgula flutuante, usando cinco bytes.

A versão aqui estudada do monitor contém, no entanto, dois erros nesta secção.

1. Existe um erro na «divisão», devido ao qual é perdido o 34.º bit de qualquer divisão.
2. O valor -65536 é por vezes usado em forma «curta» e outras em vírgula flutuante, o que conduz a problemas.

O calculador de vírgula flutuante

O calculador do Spectrum trata números e cadeias, sendo as suas operações especificadas por «literais». Pode, portanto, considerar-se que existe uma linguagem interna ao calculador.

Esta parte do programa monitor contém rotinas para todas as funções matemáticas. As aproximações das funções SIN X, EXP X, LN X e ATN X são obtidas através do desenvolvimento de polinómios de Chebyshev, sendo fornecidos no apêndice incluído no final do livro todos os pormenores relativos a estas funções.

Em termos globais, o programa monitor de 16 K incluído no Spectrum oferece uma gama bastante vasta de comandos e funções Basic diferentes. No entanto, os programadores não dispuseram de muito espaço, e preocuparam-se mais em escrever um programa compacto, à custa da rapidez.

A «start»

A interrupção mascarável é inibida e o par de registos DE recebe o valor «topo superior da RAM».

```
0000 START      DI
                XOR   A          Inibe a «interrupção do teclado».
                                 +00 para «arranque» (mas +FF para
                                 «NEW»).
                LD    DE,+FFFF   Topo superior da RAM.
                JP    11CB,START/NEW  Salto para a frente.
```

O restart «erro»

O indicador de erro é levado a apontar para a posição do erro.

```
0008 ERROR-1    LD    HL,(CH-ADD)  O endereço atingido pelo interpreta-
                LD    (X-PTR),HL   dor é copiado para o indicador
                JR    0053,ERROR-2 de erro antes de continuar.
```

O restart «imprimir caracter»

O registo A guarda o código do caracter que se pretende imprimir.

```
0010 PRINT-A-1  JP    15F2,PRINT-A-2  Salto imediato para a frente.
                DEFB  +FF,+FF,+FF,+FF,+FF Posições não utilizadas.
```

O restart «recuperar caracter»

É recuperado o conteúdo da posição actualmente endereçada por CH-ADD. É feito um retorno no caso de o valor representar um caracter susceptível de ser impresso, senão incrementa-se CH-ADD e repetem-se as comparações.

```
0018 GET-CHAR   LD    HL,(CH-ADD)  Recuperar o valor endereçado
                LD    A,(HL)       por CH-ADD.
001C TEST-CHAR  CALL  007D,SKIP-OVER Descobrir se o caracter pode ser
                RET   NC           impresso. Retorno se assim for.
```

O restart «recuperar caracter seguinte»

Quando é interpretada uma linha Basic, esta rotina é invocada repetidamente para prosseguir ao longo da linha.

```
0020 NEXT-CHAR   CALL  0074,CH-ADD+1    CH-ADD deve ser incrementado.
                 JR    001C,TEST-CHAR   Salto atrás para verificar o novo valor.
                 DEFB  +FF,+FF,+FF      Posições não usadas.
```

O restart «calculador»

Entra-se no calculador de vírgula flutuante por 335B.

```
0028 FP-CALC     JP    335B,CALCULATE   Salto imediato para a frente.
                 DEFB  +FF,+FF,+FF,+FF,+FF Posições não usadas.
```

O restart «fazer BC espaços»

Esta rotina cria posições livres no espaço de trabalho. O número de posições é determinado pelo conteúdo do par de registos BC.

```
0030 BC-SPACES   PUSH  BC               Salvaguardar o «número».
                 LD    HL,(WORKSP)      Recuperar o endereço actual do
                 PUSH  HL               início do espaço de trabalho e salva-
                 JP    169E,RESERVE     guardá-lo também antes de continuar.
```

A rotina «interrupção mascarável»

É incrementado o relógio de tempo real, lendo-se o teclado sempre que ocorre a interrupção.

```
0038 MASK-INT    PUSH  AF               Salvaguardar os valores actual-
                 PUSH  HL               mente nestes registos.
                 LD    HL,(FRAMES)      Os dois bytes inferiores do contador
                 INC   HL               de imagens são incrementados em
                 LD    (FRAMES),HL      cada 20 ms (Europa). O byte
                 LD    A,H              superior do contador de imagens só
                 OR    L                é incrementado quando o valor dos
                 JR    NZ,0048,KEY-INT  dois bytes inferiores é
                 INC   (FRAMES-3)       igual a zero.
0048 KEY-INT     PUSH  BC               Salvaguardar os valores actuais
                 PUSH  DE               destes registos.
                 CALL  02BF,KEYBOARD    Ler o teclado.
                 POP   DE               Recuperar os valores.
                 POP   BC
                 POP   HL
                 POP   AF
                 EI                     A interrupção mascarável é
                 RET                    novamente permitida.
```

A rotina «error-2»

O endereço de retorno ao interpretador aponta para o «DEFB» que indica o erro ocorrido. Este «DEFB» é recuperado e transferido para ERR-NR. O «stack» da máquina é limpo antes de saltar para diante a fim de limpar o «stack» do calculador.

```
0053 ERROR-2     POP   HL               O endereço no «stack» aponta
                 LD    L,(HL)           para o código de erro.
0055 ERROR-3     LD    (ERR-NR),L       É transferido para ERR-NR.
                 LD    SP,(ERR-SP)      O «stack»-máquina é limpo antes
                 JP    16C5,SET-STK     de sair por SET-STK.
                 DEFB  +FF,+FF,+FF,+FF  Posições não usadas.
                 DEFB  +FF,+FF,+FF
```

A rotina «interrupção não mascarável»

Esta rotina não é usada no Spectrum, permitindo apenas a execução de um «reset» de sistema quando é activada a linha NMI (Non-Maskable Interrupt). A variável de sistema em 5CB0, aqui designada NMIADD, deve conter o valor zero para que este «reset» seja executado.

```
0066 RESET       PUSH  AF               Salvaguardar os valores actuais
                 PUSH  HL               destes registos.
                 LD    HL,(NMIADD)      Os dois bytes de NMIADD devem
                 LD    A,H              ser zero para que ocorra
                 OR    L                o «reset».
                 JR    NZ,0070,NO-RESET Nota: Esta instrução deveria ser
                                        «JR Z».
                 JP    (HL)             Salto para START.
0070 NO-RESET    POP   HL               Recuperar os valores destes
                 POP   AF               registos, e retorno.
                 RETN
```

A subrotina «CH-ADD+1»

É recuperado o endereço guardado em CH-ADD, em seguida, incrementado e colocado de novo em CH-ADD. Recupera-se em seguida o conteúdo da posição agora endereçada por CH-ADD. São usados os pontos de entrada TEMP-PTR1 e TEMP-PTR2 para accionar CH-ADD temporariamente.

```
0074 CH-ADD+1    LD    HL,(CH-ADD)      Recuperar o endereço.
0077 TEMP-PTR1   INC   HL               Incrementar o indicador.
0078 TEMP-PTR2   LD    (CH-ADD),HL      Accionar CH-ADD.
                 LD    A,(HL)           Recuperar o valor endereçado e
                 RET                    retorno.
```

A subrotina «skip-over»

O valor passado à subrotina no registo A é comparado a fim de verificar se pode ser impresso. Diversos códigos especiais levam HL a ser incrementado uma vez ou duas, sendo CH-ADD corrigido em função disso.

```
007D SKIP-OVER   CP    +21              Retorno com a flag «carry» a zero
                 RET   NC               se se trata de um código normal de
                 CP    +0D              caracter. Retorno se foi atingido
                 RET   Z                o final da linha.
                 CP    +10              Retorno para códigos +00 a +0F
                 RET   C                mas com a flag «carry» a um.
                 CP    +18              Retorno para os códigos +18 a +20
                 CCF                    também com «carry» a um.
                 RET   C
```

```
                INC   HL          Incrementar uma vez.
                CP    +16         Saltar para a frente para os códigos
                JR    C,0090,SKIPS +10 a +15 (INK a OVER).
                INC   HL          Incrementar de novo (AT e
                                  TAB).
0090 SKIPS      SCF               Retorno com flag «carry» a um
                LD    (CH-ADD),HL e CH-ADD contendo o endereço
                RET               apropriado.
```

Tabela de palavras-chave («tokens»)

Todas as palavras usadas pelo Spectrum são construídas usando esta tabela. O último código de cada uma delas é «invertido» passando a um o bit 7.

0095	BF	52	4E	C4	49	4E	4B	45	'?'	R	N	'D'	I	N	K	E
009D	59	A4	50	C9	46	CE	50	4F	Y	'$'	P	'I'	F	'N'	P	O
00A5	49	4E	D4	53	43	52	45	45	I	N	'T'	S	C	R	E	E
00AD	4E	A4	41	54	54	D2	41	D4	N	'$'	A	T	T	'R'	A	'T'
00B5	54	41	C2	56	41	4C	A4	43	T	A	'B'	V	A	L	'$'	C
00BD	4F	44	C5	56	41	CC	4C	45	O	D	'E'	V	A	'L'	L	E
00C5	CE	53	49	CE	43	4F	D3	54	'N'	S	I	'N'	C	O	'S'	T
00CD	41	CE	41	53	CE	41	43	D3	A	'N'	A	S	'N'	A	C	'S'
00D5	41	54	CE	4C	CE	45	58	D0	A	T	'N'	L	'N'	E	X	'P'
00DD	49	4E	D4	53	51	D2	53	47	I	N	'T'	S	Q	'R'	S	G
00E5	CE	41	42	D3	50	45	45	CB	'N'	A	B	'S'	P	E	E	'K'
00ED	49	CE	55	53	D2	53	54	52	I	'N'	U	S	'R'	S	T	R
00F5	A4	43	48	52	A4	4E	4F	D4	'$'	C	H	R	'$'	N	O	'T'
00FD	42	49	CE	4F	D2	41	4E	C4	B	I	'N'	O	'R'	A	N	'D'
0105	3C	BD	3E	BD	3C	BE	4C	49	'<'	'='	'>'	'='	'<'	'>'	L	I
010D	4E	C5	54	48	45	CE	54	CF	N	'E'	T	H	E	'N'	T	'O'
0115	53	54	45	D0	44	45	46	20	S	T	E	'P'	D	E	F	
011D	46	CE	43	41	D4	46	4F	52	F	'N'	C	A	'T'	F	O	R
0125	4D	41	D4	4D	4F	56	C5	45	M	A	'T'	M	O	V	'E'	E
012D	52	41	53	C5	4F	50	45	4E	R	A	S	'E'	O	P	E	N
0135	20	A3	43	4C	4F	53	45	20		'#'	C	L	O	S	E	
013D	A3	4D	45	52	47	C5	56	45	'#'	M	E	R	G	'E'	V	E
0145	52	49	46	D9	42	45	45	D0	R	I	F	'Y'	B	E	E	'P'
014D	43	49	52	43	4C	C5	49	4E	C	I	R	C	L	'E'	I	N
0155	CB	50	41	50	45	D2	46	4C	'K'	P	A	P	E	'R'	F	L
015D	41	53	C8	42	52	49	47	48	A	S	'H'	B	R	I	G	H
0165	D4	49	4E	56	45	52	53	C5	'T'	I	N	V	E	R	S	'E'
016D	4F	56	45	D2	4F	55	D4	4C	O	V	E	'R'	O	U	'T'	L
0175	50	52	49	4E	D4	4C	4C	49	P	R	I	N	'T'	L	L	I
017D	53	D4	53	54	4F	D0	52	45	S	'T'	S	T	O	'P'	R	E
0185	41	C4	44	41	54	C1	52	45	A	'D'	D	A	T	'A'	R	E
018D	53	54	4F	52	C5	4E	45	D7	S	T	O	R	'E'	N	E	'W'
0195	42	4F	52	44	45	D2	43	4F	B	O	R	D	E	'R'	C	O
019D	4E	54	49	4E	55	C5	44	49	N	T	I	N	U	'E'	D	I
01A5	CD	52	45	CD	46	4F	D2	47	'M'	R	E	'M'	F	O	'R'	G
01AD	4F	20	54	CF	47	4F	20	53	O		T	'O'	G	O		S
01B5	55	C2	49	4E	50	55	D4	4C	U	'B'	I	N	P	U	'T'	L
01BD	4F	41	C4	4C	49	53	D4	4C	O	A	'D'	L	I	S	'T'	L
01C5	45	D4	50	41	55	53	C5	4E	E	'T'	P	A	U	S	'E'	N
01CD	45	58	D4	50	4F	4B	C5	50	E	X	'T'	P	O	K	'E'	P
01D5	52	49	4E	D4	50	4C	4F	D4	R	I	N	'T'	P	L	O	'T'
01DD	52	55	CE	53	41	56	C5	52	R	U	'N'	S	A	V	'E'	R
01E5	41	4E	44	4F	4D	49	5A	C5	A	N	D	O	M	I	Z	'E'
01ED	49	C6	43	4C	D3	44	52	41	I	'F'	C	L	'S'	D	R	A
01F5	D7	43	4C	45	41	D2	52	45	'W'	C	L	E	A	'R'	R	E
01FD	54	55	52	CE	43	4F	50	D9	T	U	R	'N'	C	O	P	'Y'

Tabelas das teclas

Existem seis tabelas diferentes para as teclas. O código de caracter obtido no final depende da tecla premida e do «modo» em que o foi.

a) Tabela principal — Modo L e CAPS SHIFT.

0205	42	48	59	36	35	54	47	56	B	H	Y	6	5	T	G	V
020D	4E	4A	55	37	34	52	46	43	N	J	U	7	4	R	F	C
0215	4D	48	49	38	33	45	44	58	M	K	I	8	3	E	D	K
021D	0E	4C	4F	39	32	57	53	5A	SYMBOL SHIFT	L	O	9	2	W	S	Z
0225	20	0D	50	30	31	51	41		SPACE	ENTER	P	0	1	Q	A	

b) Modo «extenso» (*extended*). Teclas de letras e sem «shift».

022C	E3	C4	E0	E4	READ	BIN	LPRINT	DATA
0230	B4	BC	BD	BB	TAN	SGN	ABS	SQR
0234	AF	B0	B1	C0	CODE	VAL	LEN	USR
0238	A7	A6	BE	AD	PI	INKEY$	PEEK	TAB
023C	B2	BA	E5	A5	SIN	INT	RESTORE	RND
0240	C2	E1	B3	B9	CHR$	LLIST	COS	EXP
0244	C1	B8			STR$	LN		

c) Modo «extenso» (*extended*). Teclas de letras e qualquer dos «shifts».

0246	7E	DC	DA	5C	~	BRIGHT	PAPER	\
024A	B7	7B	7D	D8	ATN	{	}	CIRCLE
024E	BF	AE	AA	AB	IN	VAL$	SCREEN$	ATTR
0252	DD	DE	DF	7F	INVERSE	OVER	OUT	©
0256	B5	D6	7C	D5	ASN	VERIFY	|	MERGE
025A	5D	DB	B6	D9	]	FLASH	ACS	INK
025E	5B	D7			[	BEEP		

d) Códigos de comando. Teclas de números e CAPS SHIFT.

0260	0C	07	06	04	DELETE	EDIT	CAPS LOCK	TRUE VIDEO
0264	05	08	0A	0B	INV VIDEO	Cursor left	Cursor down	Cursor up
0268	09	0F			Cursor right	GRAPHICS		

e) Códigos de símbolos. Teclas de letras e SYMBOL SHIFT.

026A	E2	2A	3F	CD	STOP	*	?	STEP
026E	C8	CC	C8	5E	>=	TO	THEN	↑
0272	AC	2D	2B	3D	AT	-	+	=
0276	2E	2C	3B	22	.	,	;	"
027A	C7	3C	C3	3E	<=	<	NOT	>
027E	C5	2F	C9	60	OR	/	<>	£
0282	C6	3A			AND	:		

f) Modo «extenso» (*extended*). Teclas de números e SYMBOL SHIFT.

0284	D0	CE	A8	CA	FORMAT	DEF FN	FN	LINE
0288	D3	D4	D1	D2	OPEN	CLOSE	MOVE	ERASE
028C	A9	CF			POINT	CAT		

2
ROTINAS DO TECLADO

A subrotina «leitura do teclado»

Esta importantíssima subrotina é invocada tanto pela rotina principal do teclado como pela rotina INKEY$ (em SCANNING).

Em todos os casos o registo E recebe um valor na gama +00 a +27, que será diferente para cada uma das quarenta teclas do Spectrum, ou o valor +FF no caso de não ter sido premida qualquer tecla.

O registo D recebe um valor que indica qual a tecla «shift» em que se carregou. Se ambas tiverem sido premidas, os registos D e E recebem respectivamente os valores de CAPS SHIFT e SYMBOL SHIFT. Se, por outro lado, não estiver a ser premida qualquer tecla, o par de registos DE recebe o valor +FFFF.

A flag «zero» é deixada pela rotina com o valor zero se estiverem a sèr premidas mais de duas teclas, ou ainda no caso de nenhuma das eventuais duas teclas premidas ser uma das teclas de «shift».

```
028E KEY-SCAN   LD    L,+2F          O valor inicial da tecla para cada
                                     linha será +2F, +2E, . . . , +28
                                     (Oito linhas).
                LD    DE,+FFFF       Inicializar DE para «nenhuma tecla».
                LD    BC,+FEFE       C = endereço do porto, B = contador.
```

A execução entra agora num ciclo. São, com efeito, realizadas oito passagens, cada uma delas a partir de um valor de tecla inicial diferente, e controlando uma linha também diferente com cinco teclas (a primeira linha, ou mais propriamente meia-linha, é CAPS SHIFT, Z, X, C, V).

```
0296 KEY-LINE   IN    A,(C)          Ler do porto especificado.
                CPL                  Uma tecla premida na linha passará
                AND   +1F            a um o bit respectivo (bit 0-tecla
                                     exterior, a bit 4-tecla interior).
                JR    Z,02AB,KEY-DONE Saltar para a frente se nenhuma
                                     das cinco teclas da linha estiver
                                     a ser premida.
                LD    H,A            Os bits das teclas são passados
                LD    A,L            para o registo H enquanto é re-
                                     cuperado o valor da tecla inicial.
029F KEY-3KEYS  INC   D              Se são premidas três teclas do
                RET   NZ             teclado o registo D
                                     deixará de conter +FF,
                                     provocando o retorno.
```



```
02A1 KEY-BITS   SUB   +08            Subtrai repetidamente «8» ao valor
                SRL   H              da tecla actual até ser encontrado
                JR    NC,02A1,KEY-BITS um bit-tecla.
                LD    D,E            Copiar qualquer valor de tecla inicial
                                     para o registo D.
                LD    E,A            Passar o novo valor de tecla para
                                     o registo E.
                JR    NZ,029F,KEY-3KEYS Se existir uma segunda ou terceira
                                     tecla premida na linha, saltar
                                     para trás.
02AB KEY-DONE   DEC   L              A linha foi «varrida», pelo que é
                                     reduzido o valor da tecla inicial
                                     para a passagem seguinte.
                RLC   B              O contador é rodado, e executado
                JR    C,0296,KEY-LINE o salto, se ainda restarem linhas a
                                     considerar.
```

São agora realizadas quatro comparações.

```
                LD    A,D            Aceita qualquer valor de tecla que
                INC   A              ainda mantenha o registo D em +FF,
                RET   Z              ou seja, uma única tecla premida
                                     ou nenhuma.
                CP    +28            Aceita o valor de tecla de um par
                RET   Z              de teclas se a tecla em «D» for
                                     CAPS SHIFT.
                CP    +19            Aceita o valor de tecla de um par
                RET   Z              de teclas se a tecla em «D» for
                                     SYMBOL SHIFT.
                LD    A,E            É no entanto possível que a tecla
                LD    E,D            «E» de um par de teclas seja
                LD    D,A            SYMBOL SHIFT — o que deve
                CP    +18            portanto ser considerado.
                RET                  Retorno com a flag «zero» a um se
                                     se tratava de SYMBOL SHIFT e
                                     «outra tecla»; senão, a flag é zero.
```

A subrotina «keyboard»

Esta subrotina é invocada em todas as ocasiões em que ocorre uma interrupção mascarável. Em funcionamento normal isto acontecerá 50 vezes por segundo. O objectivo desta subrotina é ler o teclado e descodificar o valor da tecla. O código produzido, no caso de o estado de «repetição» da tecla o permitir, será passado para a variável de sistema LAST-K. Quando é colocado um código nesta variável de sistema o bit 5 da FLAGS é passado a um para indicar que foi premida uma nova tecla.

```
028F KEYBOARD   CALL  028E,KEY-SCAN  Recuperar um valor de tecla no
                RET   NZ             par de registos DE, com retorno
                                     imediato se a flag «zero» for zero.
```

Daqui em diante é usado um sistema duplo de «variáveis de sistema KSTATE» (KSTATE0 - KSTATE3, e KSTATE4 - KSTATE7).

Os dois conjuntos permitem a detecção de uma nova tecla premida (usando um deles) enquanto ainda persiste o «período de repetição» da tecla anterior (indicada no outro conjunto).

Cada conjunto só ficará livre para tratar uma nova tecla se esta for premida durante cerca de 1/10 de segundo, isto é, o correspondente a cinco chamadas a KEYBOARD.

```
                LD    HL,KSTATE0      Começar por KSTATE0.
02C6 K-ST-LOOP  BIT   7,(HL)          Saltar para a frente se o conjunto
                JR    NZ,02D1,K-CH-SET está «livre» (KSTATE0/4 com +FF).
                INC   HL              Se não está livre, diminuir o seu
                DEC   (HL)            contador de 5 chamadas, e libertar
                DEC   HL              o conjunto quando o contador atinge
                JR    NZ,02D1,K-CH-SET zero.
                LD    (HL),+FF
```

Depois de considerar o primeiro conjunto, alterar o indicador e considerar o segundo conjunto.

```
02D1 K-CH-SET   LD    A,L             Recuperar o byte inferior do
                LD    HL,+KSTATE4     endereço e saltar para trás se
                CP    L               o segundo conjunto ainda não foi
                JR    NZ,02C6,K-ST-LOOP considerado.
```

Retorno neste momento se o valor de tecla indica «nenhuma tecla» ou apenas uma tecla de «shift».

```
                CALL  031E,K-TEST     Realizar os testes necessários, com re-
                RET   NC              torno se for caso disso. Alterar o valor
                                      de tecla para um «código principal».
```

Se houver repetição da tecla, separar a tecla repetida de uma tecla nova.

```
                LD    HL,+KSTATE0     Consultar primeiro KSTATE0.
                CP    (HL)            Saltar para diante se os códigos
                JR    Z,0310,K-REPEAT são iguais — indicando repetição.
                EX    DE,HL           Salvaguardar o endereço de
                                      KSTATE0.
                LD    HL,+KSTATE4     Consultar KSTATE4.
                CP    (HL)            Saltar para diante se os códigos
                JR    Z,0310,K-REPEAT são iguais — indicando repetição.
```

Só será, no entanto, aceite uma nova tecla se um dos conjuntos de variáveis de sistema KSTATE estiver «livre».

```
                BIT   7,(HL)          Considerar o segundo conjunto.
                JR    NZ,02F1,K-NEW   Saltar para diante se estiver livre.
                EX    DE,HL           Considerar o primeiro conjunto.
                BIT   7,(HL)          Continuar se está livre, mas sair
                RET   Z               da subrotina KEYBOARD no caso
                                      contrário.
```

Aceita-se a tecla nova. Mas antes de a variável de sistema LAST-K poder ser preenchida é necessário inicializar o conjunto de variáveis de sistema KSTATE que está a ser usado, de modo a tratar quaisquer repetições, e descodificar o código da tecla.

```
02F1 K-NEW      LD    E,A             O código é passado para o registo
                LD    (HL),A          E, para KSTATE0/4.
```



```
                INC   HL              O «contador de 5 chamadas» deste
                LD    (HL),+05        conjunto é passado para «5».
                INC   HL              A terceira variável de sistema do
                LD    A,(REPDEL)      conjunto contém o valor REPDEL
                LD    (HL),A          (normalmente 0,7 segundos).
                INC   HL              Apontar para KSTATE3/7.
```

A descodificação de um «código principal» depende do estado actual de MODE, bit 3 da FLAGS e do «byte de shift».

```
                LD    C,(MODE)        Recuperar MODE.
                LD    D,(FLAGS)       Recuperar FLAGS.
                PUSH  HL              Salvaguardar o indicador enquanto
                CALL  0333,K-DECODE   o «código principal» é descodificado.
                POP   HL
                LD    (HL),A          O código final é salvaguardado em
                                      KSTATE3/7; é recolhido daí em
                                      caso de repetição.
```

As três linhas de instruções seguintes são comuns ao tratamento das teclas «novas» e «repetidas».

```
0308 K-END      LD    (LAST-K),A      Passar o código final para LAST-K
                SET   5,(FLAGS)       e sinalizar «tecla nova».
                RET                   Retorno.
```

A subrotina «repetição»

Uma tecla repetirá da primeira vez ao fim de um período de tempo guardado em REPDEL (normalmente 0,7 segundos), e depois ao fim de um período de tempo mais reduzido indicado por REPPER (normalmente 0,1 segundos).

```
0310 K-REPEAT   INC   HL              Apontar para o «contador de 5 cha-
                LD    (HL),+05        madas» do conjunto que está em
                                      uso e passá-lo para «5».
                INC   HL              Apontar para a terceira variável de
                DEC   (HL)            sistema — valor REPDEL/REPPER,
                                      e decrementá-la.
                RET   NZ              Sair da subrotina KEYBOARD
                                      se ainda não terminou o período
                                      de atraso.
                LD    A,(REPPER)      Depois de passar, seleccionar
                LD    (HL),A          REPPER como período de tempo a
                                      considerar em seguida.
                INC   HL              Foi aceite a repetição, pelo
                LD    A,(HL)          que é recuperado o código final
                                      de KSTATE3/7 e passado a
                JR    0308,K-END      K-END.
```

A subrotina «K-test»

O valor da tecla é comparado, executando-se o retorno se for «nenhuma tecla» ou «só SHIFT»; senão, é descoberto o «código principal» dessa tecla.

```
031E K-TEST     LD    B,D          Copiar o byte de shift.
                LD    D,+00        Limpar o registo D para uso ulterior.
                LD    A,E          Deslocar o número da tecla.
                CP    +27          Retorno se for apenas CAPS SHIFT
                RET   NC           ou «nenhuma tecla».
                CP    +18          Saltar para a frente a menos que
                JR    NZ,032C,K-MAIN  a tecla «E» seja SYMBOL SHIFT.
                BIT   7,B          Aceitar porém SYMBOL SHIFT e
                RET   NZ           outra tecla; retorno com
                                   SYMBOL SHIFT apenas.
```

Determina-se o «código principal» indexando à tabela das teclas.

```
032C K-MAIN     LD    HL,+0205     O endereço base da tabela.
                ADD   HL,DE        Indexar a tabela e recuperar o
                LD    A,(HL)       «código principal».
                SCF                Sinalizar «tecla válida»
                RET                antes do retorno.
```

A subrotina «descodificação do teclado»

Entra-se nesta subrotina com o «código principal» no registo E, o valor de FLAGS no registo D, o valor de MODE no registo C e o «byte de shift» no registo B.

Considerando estes quatro valores e consultando, se necessário, as seis tabelas de teclas, produz-se um «código final». Este é passado ao registo A antes da saída da subrotina.

```
0333 K-DECODE   LD    A,E          Copiar o «código principal».
                CP    +3A          Saltar para diante se está a ser
                JR    C,0367,K-DIGIT  considerada uma tecla numérica,
                                   SPACE, ENTER ou ambos os
                                   SHIFT's.
                DEC   C            Decrementa o valor MODE.
                JP    M,034F,K-KLC-LET  Saltar para diante, conforme o
                JR    Z,0341,K-E-LET    caso, para «K», «L», «C» e «E».
```

Só resta o modo «gráficos», e o «código final» das teclas de letras neste modo é calculado a partir do «código principal».

```
                ADD   A,+4F        Somar deslocamento.
                RET                Retorno com «código final».
```

Em seguida, consideram-se as teclas de letras em modo «extenso».

```
0341 K-E-LET    LD    HL,+01EB     Endereço base da tabela «b».
                INC   B            Saltar para diante para usar esta
                JR    Z,034A,K-LOOK-UP  tabela se não é premida nenhuma
                                   tecla SHIFT.
                LD    HL,+0205     Caso contrário, usar o endereço
                                   base da tabela «c».
```

As tabelas de teclas «b-í» são todas servidas pela seguinte rotina de consulta. Em todos os casos é encontrado e devolvido um «código final».


```
034A K-LOOK-UP  LD    D,+00        Limpar o registo D.
                ADD   HL,DE        Indexar a tabela requerida e
                LD    A,(HL)       recuperar o «código final».
                RET                Retorno.
```

As teclas de letras nos modos «K», «L» e «C» são consideradas em seguida. Mas primeiro, é necessário tratar os códigos especiais SYMBOL SHIFT.

```
034F K-KLC-LET  LD    HL,+0229     Endereço base da tabela «e».
                BIT   0,B          Saltar atrás se se usou SYMBOL
                JR    Z,034A,K-LOOK-UP  SHIFT e uma tecla de letras.
                BIT   3,D          Saltar para diante se o modo
                JR    Z,0364,K-TOKENS   é «K».
                BIT   3,(FLAGS2)   Se se usa CAPS LOCK, retorno
                RET   NZ           com «código principal».
                INC   B            Retorno da mesma forma se se usa
                RET   NZ           CAPS SHIFT.
                ADD   A,+20        Porém, se os códigos de minúsculas
                RET                forem requeridos, somar +20 ao
                                   «código principal» a fim de obter
                                   o «código final» correcto.
```

Os valores de «código final» no caso das palavras-chave são determinados somando +5 ao «código principal».

```
0364 K-TOKENS   ADD   A,+A5        Somar o deslocamento requerido.
                RET                Retorno.
```

Em seguida, são consideradas as teclas numéricas, de SPACE, ENTER e ambas as de «shift».

```
0367 K-DIGIT    CP    +30          Continuar apenas com as numéricas,
                RET   C            ou seja, retorno para SPACE (+20),
                                   ENTER (+0D) e ambas as «shifts»
                                   (+0E).
                DEC   C            Separar as teclas numéricas em
                                   três grupos — em função do
                                   modo.
                JP    M,039D,K-KLC-DGT  Salto nos modos «K», «L» e «C»;
                JR    NZ,0389,K-GRA-DGT e também no «G».
                                   Continuar em modo «E».
                LD    HL,+0254     Endereço base da tabela «I».
                BIT   5,B          Usar esta tabela para SYMBOL
                JR    Z,034A,K-LOOK-UP  SHIFT e um algarismo em modo
                                   extenso.
                CP    +38          Saltar para diante no caso das
                JR    NC,0382,K-8-&-9   teclas «8» e «9».
```

As teclas «8» e «9» em modo extenso servem para obter um «código de cor de papel» ou um «código de cor de tinta», conforme o uso dado a CAPS SHIFT.

```
                SUB   +20          Reduzir a gama +30 a +37
                                   obtendo +10 a +17.
                INC   B            Retorno com este «código de cor
                RET   Z            de papel» se não está a ser usada
                                   CAPS SHIFT.
```

```
        ADD   A,+08          Mas se está, a gama deverá ser
        RET                  +18 a +1F — indicando, neste caso,
                             um «código de cor de tinta».
```

As teclas «8» e «9» produzem códigos BRIGHT e FLASH.

```
0382 K-8-&-9  SUB   +36      +38 e +39 passam a +02 e +03.
              INC   B        Retorno com estes códigos se não é
              RET   Z        usado CAPS SHIFT (códigos
                             «BRIGHT»).
              ADD   A,+FE    Subtrair «2» se é usada
              RET            CAPS SHIFT, dando +00 e +01
                             (códigos «FLASH»).
```

As teclas numéricas, em modo gráfico, devem produzir os caracteres gráficos de blocos (+80 a +8F), o código GRAPHICS (+0F) e o código DELETE (+0C).

```
0389 K-GRA-DGT  LD    HL,+0230   Endereço base da tabela «d».
                CP    +39        Usar directamente esta tabela para
                JR    Z,034A,K-LOOK-UP  a tecla «9» que dará GRAPHICS, e
                CP    +30        para a tecla «0» que dará
                JR    Z,034A,K-LOOK-UP  DELETE.
                AND   +07        No caso das teclas «1» a «8»,
                ADD   A,+80      transformar para a gama +80 a +87.
                INC   B          Retorno com um valor desta gama
                RET   Z          se não se estiver a carregar
                                 em qualquer tecla «shift».
                XOR   +0F        Se se estiver, usar a gama +88
                RET              a +8F.
```

Considerar, finalmente, as teclas numéricas nos modos «K», «L» e «C».

```
039D K-KLC-DGT  INC   B          Retorno directo se não é usada
                RET   Z          qualquer tecla «shift» (códigos
                                 finais +30 a +39).
                BIT   5,B        Usar tabela «d» se é premida
                LD    HL,+0230   também a tecla CAPS
                JR    NZ,034A,K-LOOK-UP  SHIFT.
```

Os códigos das diversas teclas numéricas usadas em simultâneo com SYMBOL SHIFT podem agora ser encontrados.

```
                SUB   +10        Reduzir a gama de modo a obter
                                 +20 a +29.
                CP    +22        Separar o caracter «@» dos
                JR    Z,03B2,K-@-CHAR  outros.
                CP    +20        Deve separar-se também o
                                 caracter «=».
                RET   NZ         Retorno com os «códigos finais»
                                 +21, +23 a +29.
                LD    A,+5F      Dar ao caracter «=» um código
                RET              +5F.
0382 K-@-CHAR   LD    A,+40      Dar ao caracter «@» um código
                RET              +40.
```


As duas subrotinas desta secção são a BEEPER, que, de facto, controla o altifalante, e a rotina do comando BEEP.

O altifalante é activado colocando o bit D4 a zero numa instrução OUT que utilize o porto «254». Quando D4 é um numa situação semelhante, o altifalante é desactivado. Pode, portanto, produzir-se um BEEP alterando continuamente o nível de D4.

Consideremos agora o Dó central da escala, com uma frequência de 261,63 Hz. Para obter esta nota será necessário activar e desactivar, alternadamente, o altifalante em cada 1/523,23 de segundo. No Spectrum, o relógio de sistema funciona a 3,5 MHz e o Dó central obrigará a executar a instrução OUT tão próxima quanto possível de 6689 estados T. Este último valor, quando ligeiramente reduzido por atrasos impossíveis de evitar, representa a «duração do ciclo de temporização» na rotina BEEPER.

A subrotina «Beeper»

Entra-se nesta subrotina apresentando no par de registos DE o valor «f*t», onde *f* é a frequência pretendida para uma nota que deve durar *t* segundos, e no par de registos HL um valor igual ao número de estados T do «ciclo de temporização» dividido por 4.

Como exemplo, para que seja produzido um Dó central durante um segundo, DE conterá +0105 (INT (261.63 * 1)) e HL +066A (obtido de 6689/4 - 30.125).

```
0385 BEEPER     DI               Inibe interrupções durante o
                                 tempo de um «beep».
                LD    A,L        Salvaguarda L temporariamente.
                SRL   L          Cada «1» no registo L obriga a
                SRL   L          contar 4 estados T, mas considera-se
                                 em vez disso INT (L/4) e contam-se
                                 16 estados T.
                CPL              Volta ao valor original em L e
                AND   +03        determina o que se perdeu usando
                LD    C,A        INT (L/4).
                LD    B,+00
                LD    IX,+03D1   Endereço base do ciclo de temporização.
                ADD   IX,BC      Altera a duração do ciclo de
                                 temporização. Usa um ponto de entrada anterior por cada «1» perdido
                                 por considerar INT (L/4).
```

```
                LD      A,(BORDCR)      Recupera a actual cor
                AND     +38             de margem, e desloca-a
                RRCA                    para os bits 2, 1 e 0 do registo A.
                RRCA
                RRCA
                OR      +08             Garante que a saída MIC está «off».
```

Entra-se agora no ciclo de produção do som. São realizadas «DE» passagens completas, ou seja, uma passagem para cada ciclo da nota considerada.

O registo HL contém a «duração do ciclo de temporização», sendo usados 16 estados T por cada «1» do registo L e 1024 estados T por cada «1» no registo H.

```
03D1 BE-IX+3    NOP                     Acrescentar 4 estados T por cada
03D2 BE-IX+2    NOP                     ponto de entrada baixo que seja
03D3 BE-IX+1    NOP                     usado.
03D4 BE-IX+0    INC     B               Os valores dos registos B e C
                INC     C               virão dos registos H e L — ver
                                        abaixo.
03D6 BE-H&L-LP  DEC     C               «Ciclo de temporização»,
                JR      NZ,03D6,BE-H&L-LP  isto é, «BC» * 4 estados T
                LD      C,+3F           (notar que no ponto de meio ciclo
                DEC     B               C tornar-se-á igual a
                JP      NZ,03D6,BE-H&L-LP  «L+1»).
```

O altifalante é agora activado e desactivado alternadamente.

```
                XOR     +10             Flip bit 4.
                OUT     (+FE),A         Realizar a operação OUT,
                                        deixando a margem sem alterações.
                LD      B,H             Passar o registo B para zero.
                LD      C,A             Salvaguardar o registo A.
                BIT     4,A             Saltar no ponto de
                JR      NZ,03F2,BE-AGAIN  meio-ciclo.
```

Ao fim de um ciclo completo é verificado o par de registos.

```
                LD      A,D             Saltar para diante se já foi
                OR      E               feita completamente a última
                JR      Z,03F6,BE-END   passagem.
                LD      A,C             Recuperar o valor salvaguardado.
                LD      C,L             Passar a zero o registo C.
                DEC     DE              Diminuir o contador de execuções.
                JP      (IX)            Saltar atrás para o início
                                        apropriado do ciclo.
```

São definidos os parâmetros do segundo meio-ciclo.

```
03F2 BE-AGAIN   LD      C,L             Passa a zero o registo C.
                INC     C               Soma 16 estados T porque este
                                        trajecto é mais curto.
                JP      (IX)            Salto para trás.
```

Ao terminar o «beep», são aceites as interrupções mascaráveis.

```
03F6 BE-END     EI                      Admitir interrupções.
                RET                     Retorno.
```


A rotina do comando «beep»

Entra-se nesta subrotina com dois números no «stack» do calculador. O número mais acima representa a frequência da nota, e o inferior a sua duração.

```
03F8 BEEP       RST     0028,FP-CALC    É usado o calculador de vírgula
                                        flutuante para manipular os
                                        dois valores - t e P.
                DEFB    +31,cópia       t, P, P
                DEFB    +27,int         t, P, i (i = INT P)
                DEFB    +C0,st-mem-0    t, P, i (mem-0 contém i)
                DEFB    +03,subtrair    t, p (p = parte fraccionária
                                        de P)
                DEFB    +34,stk-dado    «Stack» o valor decimal K,
                DEFB    +EC,expoente+7C 0,0577622606 (um pouco inferior
                DEFB    +6C,+98,+1F,+F5 a 12 √2 – 1)
                DEFB    +04,multiplicar t,pK
                DEFB    +A1,stk-um      t,pK,1
                DEFB    +0F,soma        t,pK+1
                DEFB    +38,fim-calc
```

Realizam-se, em seguida, vários testes sobre *i*, a parte inteira de «frequência».

```
                LD      HL,+5C92        É «mem-0-primeiro» (MEMBOT)
                LD      A,(HL)          Recupera o expoente de i.
                AND     A               Dá erro se i não está na forma
                JR      NZ,046C,REPORT-B  inteira (curta).
                INC     HL              Copia o byte de sinal para o
                LD      C,(HL)          registo C.
                INC     HL              Copia o byte baixo para o registo
                LD      B,(HL)          B; e também para
                LD      A,B             o registo A.
                RLA                     Produz também o «report» B se
                SBC     A,A             não satisfaz o teste:
                CP      C               –128 <= i <= +127
                JR      NZ,046C,REPORT-B
                INC     HL
                CP      (HL)
                JR      NZ,046C,REPORT-B
                LD      A,B             Recupera o byte baixo e testa-o
                                        novamente.
                ADD     A,+3C
                JP      P,0425,BE-i-OK  Aceita –60 <=i <= 67.
                JP      PO,046C,REPORT-B  Rejeita –128 a –61.
```

Nota: a gama +70 a +127 será rejeitada mais adiante.
Pode determinar-se agora a frequência adequada a *i*.

```
0425 BE-i-OK    LD      B,+FA           Começar 6 oitavas abaixo do Dó
                                        central.
0427 BE-OCTAVE  INC     B               Reduzir repetidamente i a fim de
                SUB     +0C             descobrir a oitava correcta.
                JR      NC,0427,BE-OCTAVE
                ADD     A,+0C           Somar o valor da última subtracção.
                PUSH    BC              Salvaguardar o número da oitava.
```

```
LD      HL,+046E        Endereço base da «tabela de meios-
                        -tons».
CALL    3406,LOC-MEM    Considerar a tabela e passar o valor
CALL    33B4,STACK-NUM  de ordem A para o «stack» do
                        calculador (chamá-lo C).
```

Agora pode considerar-se a parte fraccionária da frequência.

```
RST     0028,FP-CALC        f,pK+1,C
DEFB    +04,multiplicar     f,C(pK+1)
DEFB    +38,fim-calc
```

A frequência final *f* é determinada modificando o «último valor» em função do número de oitava.

```
POP     AF              Recuperar o número de oitava.
ADD     A,(HL)          Multiplicar o «último valor»
                        por 2 elevado à potência do
                        número de oitava.
RST     0028,FP-CALC    f,f
DEFB    +C0,st-mem-0    A frequência é deixada por
DEFB    +02,limpar      agora em mem-0.
```

Dá-se agora atenção à «duração».

```
DEFB    +31,cópia       f,f
DEFB    +38,fim-calc
CALL    1E94,FIND INT1  O valor «INT f» deve estar
CP      +0B             na gama +00 a +0A
JR      NC,046C,REPORT-B
```

O número de ciclos completos do «beep» é dado por «f*t», pelo que se determina agora esse valor.

```
RST     0028,FP-CALC    f
DEFB    +E0,obter-mem-0 f,f
DEFB    +04,multiplicar f*f
```

O resultado é deixado no «stack» do calculador enquanto é calculada a duração requerida para «beep»;

```
DEFB    +E0,obter-mem-0     f*f,f
DEFB    +34,stk-dado        É formado o valor «3.5*10^4/8»
DEFB    +80, 4 bytes        no topo do «stack» do
DEFB    +43,expoente+93     calculador.
DEFB    +55,+9F,+80,(+00)   f*f,f,437,500 (dec.)
DEFB    +01,troca           f*f,437,500,f
DEFB    +05,divisão         f*f,437,500/f
DEFB    +34,stk-dado
DEFB    +35,expoente+85
DEFB    +71,(+00,+00,+00)   f*f,37,500/f,30.125 (dec.)
DEFB    +03,subtrair        f*f,437,500/f-30.125
DEFB    +38,fim-calc
```


Nota: O valor «437,500/f» dá a duração de meio ciclo da nota, e a sua redução de «30.125» produz «120.5» estados T durante os quais se produz a nota, ajusta os contadores, etc.

Pode agora transferir-se os valores para os registos apropriados.

```
CALL    1E99,FIND-INT2  O valor do «ciclo de temporização»
                        é comprimido no par de registos
PUSH    BC              BC; e salvaguardado.
```

Nota: Se o valor do ciclo de temporização é excessivo, ocorrerá um erro (devolvido através de ERROR-1); excluem-se, assim, valores de tonalidade de «+70» a «+127».

```
CALL    1E99,FIND-INT2  O valor «f*t» é comprimido para
                        o par de registos BC.
POP     HL              Desloca o valor do «ciclo de
                        temporização» para HL.
LD      D,B             Desloca o valor «f*t» para o
LD      E,C             par de registos DE.
```

No entanto, antes de fazer o «beep», verificar o valor «f*t».

```
LD      A,D             Retorno se «f*t» deu o
OR      E               resultado requerido de
RET     Z               «nenhum ciclo».
DEC     DE              Diminuir o número de ciclo
JP      03B5,BEEPER     e saltar para a subrotina BEEPER
                        (fazendo pelo menos uma passagem).
```

Mensagem «B — Integer out of range»

```
046C REPORT-B   RST     0008,ERROR-1    Invocar a rotina de tratamento
                DEFB    +0A             de erro.
```

A tabela de meios-tons

Esta tabela contém as frequências dos doze meios-tons de uma oitava.

```
                                    FREQUÊNCIA (Hz)     NOTA
046E  DEFB +89,+02,+D0,+12,+86      261.63              C
      DEFB +89,+0A,+97,+60,+75      277.18              C#
      DEFB +89,+12,+D5,+17,+1F      293.66              D
      DEFB +89,+18,+90,+41,+02      311.13              D#
      DEFB +89,+24,+D0,+53,+CA      329.63              E
      DEFB +89,+2E,+9D,+36,+B1      349.23              F
      DEFB +89,+38,+FF,+49,+3E      369.99              F#
      DEFB +89,+43,+FF,+6A,+73      392                 G
      DEFB +89, +4F,+A7,+00,+54     415.30              G#
      DEFB +89,+5C,+00,+00,+00      440                 A
      DEFB +89,+69,+14,+F6,+24      466.16              A#
      DEFB +89,+76,+F1,+10,+05      493.88              B
```

A subrotina «nome de programa» (ZX81)

A rotina que se segue aplica-se ao ZX81 e não foi removida quando o programa foi reescrito para o SPECTRUM.

```
04AA DEFB +CD,+FB,+24,+3A
     DEFB +3B,+5C,+87,+FA
     DEFB +8A,+1C,+E1,+D0
     DEFB +E5,+CD,+F1,+2B
     DEFB +62,+6B,+0D,+F8
     DEFB +09,+C8,+FE,+C9
```


O programa monitor de 16 K dispõe de um extenso conjunto de rotinas para tratar o interface de cassete. De facto, estas rotinas formam as rotinas de comando de SAVE, LOAD, VERIFY e MERGE.

O ponto de entrada das rotinas é em SAVE-ETC (0605). No entanto, ainda antes deste ponto encontram-se as subrotinas que realizam de facto o SAVE e LOAD (ou VERIFY) dos bytes.

Em todos os casos, os bytes a tratar por estas subrotinas são descritos pelo par de registos DE que guarda o «comprimento» do bloco, o par de registos IX que guarda o «endereço base» e o registo A que contém +00 no caso de um bloco de cabeçalho, ou +FF no de um bloco de dados/programa.

A subrotina «sa-bytes»

Esta subrotina é invocada para SAVE a informação de cabeçalho («header») (a partir de 098A), e mais tarde o bloco de dados/programa (a partir de 009E).

```
04C2 SA-BYTES   LD      HL,+053F        Carregar previamente no «stack»
                PUSH    HL              o endereço — SA/LD-RET.
                LD      HL,+1F80        Esta constante produzirá um
                                        sinal «leader» de cerca de 5 segs.
                BIT     7,A             Saltar para diante se está a
                JR      Z,04D0,SA-FLAG  SAVE um cabeçalho.
                LD      HL,+0C98        Esta constante produz um «leader»
                                        de cerca de 2 segundos para o
                                        bloco de dados/programa.
04D0 SA-FLAG    EX      AF,A'F'         Salvaguarda a flag.
                INC     DE              O «comprimento» é incrementado
                DEC     IX              e o endereço base reduzido
                                        deixando espaço para a flag.
                DI                      A interrupção mascarável é
                                        inibida durante o SAVE.
                LD      A,+02           Sinal MIC «ligado», e margem
                                        («border») vermelha.
                LD      B,A             Dá um valor a B.
```

Entra-se agora num ciclo que cria os impulsos do «leader» inicial. Tanto os impulsos que ligam como os que desligam o microfone têm um comprimento de 2.168 estados T. A cor da margem altera-se de vermelho para cião como cada rebordo do sinal.

Nota: chamamos «rebordo» a cada passagem de «ligado» a «desligado» e vice-versa.

```
04D8 SA-LEADER  DJNZ  04D8,SA-LEADER  Temporização principal.
                OUT   (+FE),A        MIC ligado/desligado, margem
                XOR   +0F            RED/CYAN, em cada passagem.
                LD    B,+A4          Constante principal de temporização.
                DEC   L              Diminuir contador baixo.
                JR    NZ,04D8,SA-LEADER Saltar atrás para novo impulso.
                DEC   B              Ter em conta trajecto maior
                                     (reduzir de 13 estados T).
                DEC   H              Diminuir contador alto.
                JP    P,04D8,SA-LEADER Saltar atrás para novo impulso
                                     até terminar o «leader».
```

É agora enviado um sinal de sincronização.

```
                LD    B,+2F
04EA SA-SYNC-1  DJNZ  04EA,SA-SYNC-1 MIC desligado durante 667 estados T
                                     entre OUT e OUT.
                OUT   (+FE),A        MIC ligado e RED.
                LD    A,+0D          Sinal «MIC desligado e CYAN».
                LD    B,+37          MIC ligado durante 735 estados T
04F2 SA-SYNC-2  DJNZ  04F2,SA-SYNC-2 entre OUT e OUT.
                OUT   (+FE),A        MIC desligado e margem cián.
```

O primeiro byte a gravar será a flag do cabeçalho ou bloco de dados/programa.

```
                LD    BC,+3B0E       +3B é uma constante de temporiza-
                                     ção; +0E sinaliza «MIC desligado e
                                     amarelo».
                EX    AF,A'F'        Recuperar a flag e passá-la
                LD    L,A            ao registo L para «envio».
                JP    0507,SA-START  Saltar para diante, para ciclo
                                     de SAVE.
```

Entra-se agora no ciclo de SAVE bytes. O primeiro byte a gravar é a flag; é seguido dos bytes de dados, terminando pelo byte de paridade, formado tendo em conta os valores de todos os bytes anteriores.

```
04FE SA-LOOP    LD    A,D            Verifica o contador «comprimento»
                OR    E              saltando quando este atinge
                JR    Z,050E,SA-PARITY zero.
                LD    L,(IX+00)      Recupera o byte que deve ser
                                     SAVEd em seguida.
0505 SA-LOOP-P  LD    A,H            Recupera a «paridade» actual.
                XOR   L              Inclui o byte presente.
0507 SA-START   LD    H,A            Corrige a «paridade». Notar
                                     que, no início, «paridade» é
                                     inicializado para o valor da flag.
                LD    A,+01          Sinal «MIC ligado e azul».
                SCF                  Passa a uma flag «carry». Esta
                                     servirá para «marcar» os 8 bits
                                     de um byte.
                JP    0525,SA-8-BITS Saltar para diante.
```

Quando chega o momento de enviar o byte de «paridade», este é transferido para o registo L a fim de ser SAVEd.


```
050E SA-PARITY  LD    L,H            Obter «paridade» final.
                JR    0505,SA-LOOP-P Saltar atrás.
```

O ciclo interior que se segue produz os impulsos. Entra-se no ciclo por SA-BIT-1, sendo o tipo de bit a SAVE indicado pela flag «carry». O ciclo é percorrido duas vezes para cada bit, produzindo assim um impulso «off» e um impulso «on». Os impulsos para um bit zero são mais curtos 855 estados T.

```
0511 SA-BIT-2   LD    A,C            Vir aqui na segunda passagem e
                                     recuperar «MIC desligado e amarelo».
                BIT   7,B            Passar a um a flag «zero» para
                                     indicar «segunda passagem».
0514 SA-BIT-1   DJNZ  0514,SA-BIT-1  Principal ciclo de temporização; sem-
                                     pre 801 estados T na 2.ª passagem.
                JR    NC,051C,SA-OUT Saltar, pelo trajecto mais curto,
                                     quando grava (SAVE) um zero.
                LD    B,+42          No entanto, se grava um 1, somar
051A SA-SET     DJNZ  051A,SA-SET    855 estados T.
051C SA-OUT     OUT   (+FE),A        Na 1.ª passagem «MIC ligado e azul»
                                     e na 2.ª passagem «MIC desligado
                                     e amarelo».
                LD    B,+3E          Definir a constante de temporização
                                     para a segunda passagem.
                JR    NZ,0511,SA-BIT-2 Saltar atrás no final da primeira
                DEC   B              passagem; senão, reclamar
                                     13 estados T.
                XOR   A              Limpar a flag «carry» e passar
                INC   A              A para +01 (MIC ligado e azul)
                                     antes de continuar para o ciclo
                                     de 8 bits.
```

O «ciclo de 8 bits» é primeiramente executado com o byte completo no registo L e a flag «carry» a um. No entanto, é retomado depois do SAVE de cada bit até se atingir um ponto em que o «marcador» passa para a flag «carry», deixando o registo L vazio.

```
0525 SA-8-BITS  RL    L              Deslocar o bit 7 para a flag
                                     «carry», e o «marcador» para a
                                     esquerda.
                JP    NZ,0514,SA-BIT-1 SAVE o bit a menos que
                                     o byte tenha terminado.
                DEC   DE             Diminuir o «contador».
                INC   IX             Avançar o «endereço-base».
                LD    B,+31          Definir a constante
                                     de temporização para
                                     o primeiro bit do byte seguinte.
                LD    A,+7F          Retorno (para SA/LD-RET) se está
                IN    A,(+FE)        a ser premida a tecla BREAK.
                RRA
                RET   NC
                LD    A,D            Se não, testar o «contador» e
                INC   A              saltar atrás mesmo que tenha
                JP    NZ,04FE,SA-LOOP atingido zero (a fim de enviar
                                     o byte de «paridade»).
                LD    B,+3B          Sair quando o «contador»
053C SA-DELAY   DJNZ  053C,SA-DELAY  atinge +FFFF. Mas antes, incluir
                RET                  um pequeno atraso.
```

Nota: um bit zero produzirá um impulso «MIC desligado» de 855 estados T seguido de um impulso «MIC ligado» de 855 estados T. Entretanto, um bit um produzirá impulsos com um comprimento exactamente duplo. Note-se, igualmente, que não existem quaisquer separações entre o impulso de sincronização e o primeiro bit de flag, ou entre os bytes.

A subrotina «SA/LD-RET»

Esta subrotina é comum a SAVE e a LOAD.
A margem é passada para a sua cor original, sendo a tecla BREAK verificada pela última vez.

```
053F SA/LD-RET   PUSH  AF              Salvaguardar a flag «carry» (é passada a zero após um erro de LOAD).
                 LD    A,(BORDCR)      Recuperar a cor original da margem
                 AND   +38             a partir da variável de sistema.
                 RRCA                  Deslocar a cor da margem para os
                 RRCA                  bits 2, 1 e 0.
                 RRCA
                 OUT   (+FE),A         Passar a margem para a sua cor original.
                 LD    A,+7F           Ler a tecla BREAK pela última
                 IN    A,(+FE)         vez.
                 RRA
                 EI                    Activar as interrupções mascaráveis.
                 JR    C,0554,SA/LD-END Saltar a menos que deva ser executado um «break».
```

Mensagem «D — BREAK-CONT repeats»

```
0552 REPORT-D    RST   0008,ERROR-1    Invocar a rotina de tratamento
                 DEFB  +0C             de erro.
```

Continuar aqui.

```
0554 SA/LD-END   POP   AF              Recuperar a flag «carry».
                 RET                   Retorno à rotina inicial.
```

A subrotina «LD-Bytes»

Esta subrotina é invocada para carregar (LOAD) a informação de cabeçalho (a partir de 076E) e depois LOAD, ou VERIFY, um bloco de dados (a partir de 0802).

```
0556 LD-BYTES    INC   D                 Passa a zero a flag «zero» (D não pode conter +FF).
                 EX    AF,A'F'           O registo A contém +00 para um cabeçalho e +FF para um bloco de dados.
                                         A flag «carry» passa a zero para VERIFY e a um para LOAD.
```



```
                 DEC   D                 Passar ao valor original de D.
                 DI                    Inibe a interrupção mascarável neste momento.
                 LD    A,+0F           A margem passa a branco.
                 OUT   (+FE),A
                 LD    HL,+053F        Carrega no «stack» o endereço SA/LD-RET.
                 PUSH  HL
                 IN    A,(+FE)         Leitura inicial do porto 254.
                 RRA                   Roda o byte obtido, mas mantém
                 AND   +20             apenas o bit EAR.
                 OR    +02             Sinaliza margem vermelha.
                 LD    C,A             Guarda o valor no registo C. (+22 para «off» e +02 para «on», estado presente de EAR)
                 CP    A               Passa a um a flag «carry».
```

A primeira fase da leitura de uma fita envolve a verificação da existência de um sinal (ou seja, saltos «on/off» ou «off/on»).

```
0568 LD-BREAK    RET   NZ              Retorno se a tecla BREAK está a ser premida.
056C LD-START    CALL  05E7,LD-EDGE-1  Retorno com a flag «carry» em
                 JR    NC,056B,LD-BREAK zero se não há orla de um sinal em cerca de 14 000 estados T. Se há, a margem passa a cião.
```

A fase seguinte envolve esperar um pouco e verificar se o sinal ainda se mantém.

```
                 LD    HL,+0415
0574 LD-WAIT     DJNZ  0574,LD-WAIT    A duração do período de espera equivalerá a quase um segundo.
                 DEC   HL
                 LD    A,H
                 OR    L
                 JR    NZ,0574,LD-WAIT
                 CALL  05E3,LD-EDGE-2  Continuar apenas se são encontradas duas orlas de sinal no tempo definido.
                 JR    NC,056B,LD-BREAK
```

Aceitar agora apenas um sinal «leader».

```
0580 LD-LEADER   LD    B,+9C           Constante de temporização.
                 CALL  05E3,LD-EDGE-2  Continuar apenas se se encontram
                 JR    NC,056B,LD-BREAK duas orlas de sinal no tempo definido.
                 LD    A,+C6           Porém, as orlas devem ter sido
                 CP    B               encontradas a cerca de 3000
                 JR    NC,056C,LD-START estados T entre si.
                 INC   H               Contar os pares de orlas de sinal
                 JR    NZ,0580,LD-LEADER no registo H, até atingir 256 pares.
```

A seguir ao «leader» surgem as partes «off» e «on» do impulso de sincronização.

```
058F LD-SYNC    LD      B,+C9          Constante de temporização.
                CALL    05E7,LD-EDGE-1 São consideradas todas as orlas
                JR      NC,056B,LD-BREAK de sinal até se encontrarem duas
                LD      A,B            suficientemente próximas — que
                CP      +D4            serão a orla inicial e final do
                JR      NC,058F,LD-SYNC impulso «off» de sincronização.
                CALL    05E7,LD-EDGE-1 Deve existir a orla final do
                RET     NC             impulso «on» (retorno com
                                       flag «carry» a zero).
```

Pode-se agora LOAD ou VERIFY os bytes do cabeçalho ou do bloco de dados/programa. Mas o primeiro byte é a flag indicadora de tipo.

```
                LD      A,C
                XOR     +03            As cores da margem serão daqui
                LD      C,A            em diante azul e amarelo.
                LD      H,+00          Inicializar o byte de teste de
                                       paridade para zero.
                LD      B,+B0          Definir a constante de tempori-
                                       zação para o byte de flag.
                JR      05C8,LD-MARKER Saltar para diante, para o ciclo
                                       de LOAD bytes.
```

O ciclo de carga de bytes é usado para recolher os bytes separadamente, um de cada vez. O primeiro é o byte de flag. Seguem-se-lhe os bytes de dados, e o último será o byte de «paridade».

```
05A9 LD-LOOP    EX      AF,A'F'        Recuperar as flags de trabalho.
                JR      NZ,05B3,LD-FLAG Saltar para diante apenas quando
                                       trata o primeiro byte.
                JR      NC,05BD,LD-VERIFY Saltar para diante se está a
                                       VERIFY uma fita.
                LD      (IX+00),L      Realizar o LOAD quando for
                                       apropriado.
                JR      05C2,LD-NEXT   Saltar para diante a fim de
                                       carregar o byte seguinte.
05B3 LD-FLAG    RL      C              Manter a flag «carry» num
                                       local seguro, temporariamente.
                XOR     L              Retorno se a flag de tipo não
                RET     NZ             concorda com o primeiro byte da
                                       fita (flag «carry» em zero).
                LD      A,C            Recuperar a flag «carry».
                RRA
                LD      C,A
                INC     DE             Aumentar o contador a fim de
                JR      05C4,LD-DEC    compensar a sua diminuição após
                                       o salto.
```

Se se está a verificar um bloco de dados, comparar o byte carregado com o original.

```
05BD LD-VERIFY  LD      A,(IX+00)      Recuperar o byte original.
                XOR     L              Compará-lo com o novo byte.
                RET     NZ             Retorno se forem diferentes
                                       (flag «carry» em zero).
```

Pode agora carregar-se um novo byte da fita.


```
05C2 LD-NEXT    INC     IX             Aumentar o endereço de destino.
05C4 LD-DEC     DEC     DE             Diminuir o contador.
                EX      AF,A'F'        Salvaguardar as flags.
                LD      B,+B2          Definir a constante de tempo.
05C8 LD-MARKER  LD      L,+01          Limpar o registo «objecto»,
                                       exceptuando um bit «marcador».
```

Usa-se o ciclo LD-8-BITS para construir um byte completo no registo L.

```
05CA LD-8-BITS  CALL    05E3,LD-EDGE-2 Determinar o comprimento dos
                                       impulsos «off» e «on» do novo bit.
                RET     NC             Retorno se é excedido o período
                                       de tempo (flag «carry» em zero).
                LD      A,+C8          Comparar a duração com cerca de
                                       2400 estados T; a flag
                CP      B              «carry» passa a zero para um 0,
                                       e a um para um 1.
                RL      L              Incluir o novo bit no registo
                                       L.
                LD      B,+B0          Definir a constante de tempo para
                                       o bit seguinte.
                JP      NC,05CA,LD-8-BITS Saltar atrás enquanto existem
                                       ainda bits para recolher.
```

É necessário actualizar o byte de «paridade» para cada novo byte recebido.

```
                LD      A,H            Recuperar o byte de «teste de
                XOR     L              paridade» e incluir o novo byte.
                LD      H,A            Salvaguardá-lo de novo.
```

O ciclo é percorrido até o «contador» atingir zero. Nesse ponto, o byte de «teste de paridade» deve guardar zero.

```
                LD      A,D            Fazer mais uma passagem se o
                OR      E              par de registos DE não contiver
                JR      NZ,05A9,LD-LOOP zero.
                LD      A,H            Recuperar o byte de «teste de
                                       paridade».
                CP      +01            Retorno com a flag «carry»
                RET                    a um se o valor for zero (flag
                                       a zero se houver erro).
```

As subrotinas «LD-EDGE-2» e «LD-EDGE-1»

Estas duas subrotinas constituem a parte mais importante da operação de LOAD/VERIFY.

Entra-se nestas subrotinas com uma constante de temporização no registo B, e a cor de margem e tipo de «orla de impulso» no registo C.

As subrotinas são abandonadas com a flag «carry» em um se foi encontrado o número requerido de «orlas» no tempo previsto; e a mudança para o valor contido no registo B indica quanto tempo foi necessário para encontrar as «orlas».

A flag «carry» terá o valor zero no caso de se ter verificado um erro. A flag «zero» sinaliza então «BREAK premida» ao passar a zero, ou «tempo cumprido» se estiver a um.

O ponto de entrada LD-EDGE-2 é usado quando é requerida a duração de um impulso completo, sendo usado LD-EDGE-1 para determinar o tempo antes da «orla» seguinte.

```
05E3 LD-EDGE-2    CALL  05E7,LD-EDGE-1   Invocar, de facto, LD-EDGE-1 duas
                  RET   NC               vezes; retorno entre ambas se
                                         houver um erro.
05E7 LD-EDGE-1    LD    A,+16            Esperar 358 estados T antes
05E9 LD-DELAY     DEC   A                de entrar no ciclo seguinte.
                  JR    NZ,05E9,LD-DELAY
                  AND   A
```

Entra-se agora no ciclo de amostragem. O valor existente no registo B é incrementado para cada passagem; é produzido «tempo cumprido» quando B atinge zero.

```
05ED LD-SAMPLE    INC   B                Contar cada passagem.
                  RET   Z                Retorno com «carry» a zero e
                                         «zero» a um se «tempo cumprido».
                  LD    A,+7F            Ler do porto +7FFE,
                  IN    A,(+FE)          isto é, BREAK e EAR.
                  RRA                    Deslocar o byte.
                  RET   NC               Retorno com «carry» a zero e
                                         «zero» a zero se BREAK premida.
                  XOR   C                Comparar o byte com «último
                  AND   +20              tipo de orla»; saltar atrás
                  JR    Z,05ED,LD-SAMPLE a menos que se tenha alterado.
```

Foi encontrada uma nova «orla» de impulso dentro do período de tempo permitido para a procura. Altera-se, portanto, a cor da margem e passa-se a um a flag «carry».

```
                  LD    A,C              Alterar o «último tipo de orla»
                  CPL                    e a cor da margem.
                  LD    C,A
                  AND   +07              Manter apenas a cor da margem.
                  OR    +08              Sinal «MIC desligado».
                  OUT   (+FE),A          Alterar a cor de margem (vermelho/
                                         /cião ou azul/amarelo).
                  SCF                    Indicar o êxito da procura
                  RET                    antes do retorno.
```

Nota: A subrotina LD-EDGE-1 utiliza 465 estados T, mais 58 estados T adicionais para cada passagem sem êxito pelo ciclo de amostragem.

Por exemplo, quando se espera o impulso de sincronização (ver LD-SYNC em 058F) são tidas em conta dez passagens adicionais pelo ciclo de amostragem. Procura-se, portanto, encontrar a «orla» seguinte em cerca de 1100 estados T (465 + 10 * 58 + atraso). Este método terá êxito no caso do impulso «off» de sincronização que ocorre após os compridos impulsos do «leader».

As rotinas de comando de «SAVE, LOAD, VERIFY, MERGE»

O ponto de entrada SAVE-ETC é usado para as quatro instruções. O valor guardado em T-ADDR distingue, no entanto, a instrução em uso. A primeira parte da rotina que se segue tem a ver com a construção da informação de «cabeçalho» na área de trabalho.

```
0605 SAVE-ETC     POP   AF               Recuperar o endereço SCAN-LOOP.
                  LD    A,(T-ADDR-lo)    Reduzir T-ADDR (byte baixo) de
                  SUB   +E0              +E0; dando +00 para SAVE, +01
                  LD    (T-ADDR-lo),A    para LOAD, +02 para VERIFY e
                                         +03 para MERGE.
                  CALL  1C8C,EXPT-EXP    Passar os parâmetros do «nome»
                                         para o «stack» do calculador.
                  CALL  2530,SYNTAX-Z    Saltar para diante se verifica
                  JR    Z,0652,SA-DATA   sintaxe.
                  LD    BC,+0011         Reservar 17 posições para o
                  LD    A,(T-ADDR-lo)    cabeçalho de um SAVE, mas
                  AND   A                trinta para as outras
                  JR    Z,0621,SA-SPACE  instruções.
                  LD    C,+22
0621 SA-SPACE     RST   0030,BC-SPACES   Reservado o espaço apropriado
                                         na área de trabalho.
                  PUSH  DE               Copiar o endereço inicial para
                  POP   IX               o par de registos IX.
                  LD    B,+0B            Um nome de programa pode ter até
                  LD    A,+20            dez caracteres, mas antes,
0629 SA-BLANK     LD    (DE),A           enviam-se onze caracteres
                  INC   DE               «espaço» para a área já
                  DJNZ  0629,SA-BLANK    reservada.
                  LD    (IX+01),+FF      Um nome nulo é apenas +FF.
                  CALL  2BF1,STK-FETCH   Os parâmetros do nome são
                                         recuperados, e o seu comprimento
                                         verificado.
                  LD    HL,+FFF6         Isto é «-10».
                  DEC   BC               De facto, saltar para diante se
                  ADD   HL,BC            o comprimento do nome não é
                  INC   BC               excessivo (ou seja, não mais de
                  JR    NC,064B,SA-NAME  dez caracteres).
                  LD    A,(T-ADDR-lo)    Mas permitir o LOAD, VERIFY e
                  AND   A                MERGE de programas com nomes
                  JR    NZ,0644,SA-NULL  «nulos» ou nomes muito
                                         extensos.
```

Mensagem «F — Invalid file name».

```
0642 REPORT-F     RST   0008,ERROR-1     Invocar a rotina de tratamento
                  DEFB  +0E              de erro.
```

Continuar a tratar o nome do programa.

```
0644 SA-NULL      LD    A,B              Saltar para diante se o nome
                  OR    C                tem comprimento nulo.
                  JR    Z,0652,SA-DATA
                  LD    BC,+000A         Mas truncar nomes excessivos.
```

O nome é agora transferido para a área de trabalho (a partir da segunda posição).

```
064B SA-NAME    PUSH IX          Copiar o endereço inicial
                POP  HL          para o par de registos HL.
                INC  HL          Passar à segunda posição.
                EX   DE,HL       Trocar os indicadores e
                LDIR             copiar o nome.
```

Os muitos parâmetros diferentes que se seguem ao nome, se existirem, são agora considerados. Começa-se por tratar 'xxx «nome» DATA'.

```
0652 SA-DATA    RST  0018,GET-CHAR      O código actual é a palavra
                CP   +E4                DATA?
                JR   NZ,06A0,SA-SCR$    Saltar, se não.
                LD   A,(T-ADDR-lo)      Porém, não pode existir
                CP   +03                'MERGE nome DATA'.
                JP   Z,1C8A,REPORT-C
                RST  0020,NEXT-CHAR     Avançar CH-ADD.
                CALL 28B2,LOOK-VARS     Procurar na área das variáveis
                                        o «array» em causa.
                SET  7,C                Passar a 1 o bit 7 do nome do
                                        array.
                JR   NC,0672,SA-V-OLD   Saltar se se está a tratar
                                        um array existente.
                LD   HL,+0000           Sinal «uso de um novo array».
                LD   A,(T-ADDR-lo)      Considerar o valor em T-ADDR
                DEC  A                  e dar erro se se tenta
                JR   Z,0685,SA-V-NEW    SAVE ou VERIFY um array novo.
```

Mensagem «2 — Variable not found»

```
0670 REPORT-2   RST  0008,ERROR-1       Invocar rotina de tratamento
                DEFB +01                de erro.
```

Continuar o tratamento de um array existente.

```
0672 SA-V-OLD   JP   NZ,1C8A,REPORT-C   Nota: não exclui cadeias
                                        («strings») simples.
                CALL 2530,SYNTAX-Z      Saltar para diante se verifica
                JR   Z,0692,SA-DATA-1   sintaxe.
                INC  HL                 Apontar para o «comprimento
                                        baixo» da variável.
                LD   A,(HL)             O byte baixo do comprimento
                LD   (IX+0B),A          passa para a área de trabalho; se-
                INC  HL                 guido pelo byte alto do comprimento.
                LD   A,(HL)
                LD   (IX+0C),A
                INC  HL                 Passar à frente dos bytes de compri-
                                        mento.
```

O trajecto que se segue é comum aos arrays «novos» e «velhos».
Nota: erro do trajecto de sintaxe.

```
0685 SA-V-NEW   LD   (IX+0E),C          Copiar o nome do array.
                LD   A,+01              Considerar que é array numérico.
                BIT  6,C                Saltar se assim for.
                JR   Z,068F,SA-V-TYPE
                INC  A                  É array de caracteres.
068F SA-V-TYPE  LD   (IX+00),A          Indicar o tipo na primeira
                                        posição da área de cabeçalho.
```

A última parte da declaração é agora examinada antes de passar aos outros trajectos.

```
0692 SA-DATA-1  EX   DE,HL              Salvaguardar o indicador em DE.
                RST  0020,NEXT-CHAR     O caracter que se segue é
                CP   +29                «)»?
                JR   NZ,0672,SA-V-OLD   Produzir mensagem C se não for.
                RST  0020,NEXT-CHAR     Avançar CH-ADD.
                CALL 1BEE,CHECK-END     Passar à declaração seguinte
                                        se se verifica sintaxe.
                EX   DE,HL              Apontar o indicador para o par
                JP   075A,SA-ALL        de registos HL antes de saltar
                                        para diante (o indicador para
                                        o início do conteúdo de um
                                        array existente).
```

Considera-se agora «SCREEN$».

```
06A0 SA-SCR$    CP   +AA                O código actual é a palavra
                                        SCREEN$?
                JR   NZ,06C3,SA-CODE    Saltar, se não.
                LD   A,(T-ADDR-lo)      Porém, não é possível usar
                CP   +03                'MERGE nome SCREEN$'.
                JP   Z,1C8A,REPORT-C
                RST  0020,NEXT-CHAR     Avança CH-ADD.
                CALL 1BEE,CHECK-END     Passar à declaração seguinte
                                        se se verifica sintaxe.
                LD   (IX+0B),+00        A área de imagem e a área de
                LD   (IX+0C),+1B        atributos ocupam +1B00
                                        posições e estas começam em
                LD   HL,+4000           +4000; estes pormenores são
                LD   (IX+0D),L          passados para a área de
                LD   (IX+0E),H          cabeçalho na zona de trabalho.
                JR   0710,SA-TYPE-3     Saltar para diante.
```

Considerar agora CODE.

```
06C3 SA-CODE    CP   +AF                O código actual é a palavra
                                        CODE?
                JR   NZ,0716,SA-LINE    Saltar, se não.
                LD   A,(T-ADDR-lo)      Porém, não é possível usar
                CP   +03                'MERGE nome CODE'.
                JP   Z,1C8A,REPORT-C
                RST  0020,NEXT-CHAR     Avança CH-ADD.
                CALL 2048,PR-ST-END     Saltar para diante se a
                JR   NZ,06E1,SA-CODE-1  declaração não terminou.
                LD   A,(T-ADDR-lo)      Porém não se pode usar
                AND  A                  'SAVE nome CODE'
                JP   Z,1C8A,REPORT-C    isoladamente.
                CALL 1CE6,USE-ZERO      Colocar um zero no «stack»
                                        do calculador — para um «início».
                JR   06F0,SA-CODE-2     Saltar para diante.
```

Procura de um «endereço inicial».

```
06E1 SA-CODE-1   CALL  1C82,EXPT-1NUM     Recupera o primeiro número.
                 RST   0018,GET-CHAR      O caracter actual é ou não
                 CP    +2C                uma «,»?
                 JR    Z,06F5,SA-CODE-3   Saltar se for — o número era
                                          um «endereço inicial».
                 LD    A,(T-ADDR-lo)      Porém, recusar 'SAVE nome
                 AND   A                  CODE' que não possua um «início»
                 JP    Z,1C8A,REPORT-C    e um «comprimento».
06F0 SA-CODE-2   CALL  1CE6,USE-ZERO      Colocar um zero no «stack» do
                                          calculador — para «comprimento».
                 JR    06F9,SA-CODE-4     Saltar para diante.
```

Recuperar o «comprimento» como especificado.

```
06F5 SA-CODE-3   RST   0020,NEXT-CHAR     Avançar CH-ADD.
                 CALL  1C82,EXPT-1NUM     Recuperar «comprimento».
```

Os parâmetros são agora guardados na área de cabeçalho, na zona de trabalho.

```
06F9 SA-CODE-4   CALL  1BEE,CHECK-END     Mas passar à declaração seguinte
                                          se se verifica sintaxe.
                 CALL  1E99,FIND-INT2     Comprimir o «comprimento» para
                 LD    (IX+0B),C          o par de registos BC e
                 LD    (IX+0C),B          guardá-lo.
                 CALL  1E99,FIND-INT2     Comprimir o «endereço inicial»
                 LD    (IX+0D),C          para o par de registos BC
                 LD    (IX+0E),B          e guardá-lo.
                 LD    H,B                Transferir o «indicador» para o par
                 LD    L,C                de registos HL como é habitual.
```

«SCREEN$» e «CODE» são ambos de tipo 3.

```
0710 SA-TYPE-3   LD    (IX+00),+03        Indicar número de «tipo».
                 JR    075A,SA-ALL        Passar aos outros trajectos.
```

Considerar agora «LINE» e «sem outros parâmetros».

```
0716 SA-LINE     CP    +CA                O código actual é a palavra
                                          LINE?
                 JR    Z,0723,SA-LINE-1   Saltar, se sim.
                 CALL  1BEE,CHECK-END     Passar à declaração seguinte
                                          se se verifica a sintaxe.
                 LD    (IX+0E),+80        Quando não existem mais parâmetros
                                          indica-se +80.
                 JR    073A,SA-TYPE-0     Saltar para diante.
```

Recuperar o «número de linha» que deve seguir-se a LINE.

```
0723 SA-LINE-1   LD    A,(T-ADDR-lo)      Só permitir porém 'SAVE
                 AND   A                  nome LINE número'
                 JP    NZ,1C8A,REPORT-C
                 RST   0020,NEXT-CHAR     Avançar CH-ADD.
                 CALL  1C82,EXPT-1NUM     Passar o número para o «stack»
                                          do calculador.
```



```
                 CALL  1BEE,CHECK-END     Passar para a declaração seguinte
                                          se se verifica a sintaxe.
                 CALL  1E99,FIND-INT2     Comprimir o número de linha
                 LD    (IX+0D),C          para o par de registos BC e
                 LD    (IX+0E),B          guardá-lo.
```

«LINE» e «sem outros parâmetros» são ambos de tipo 0.

```
073A SA-TYPE-0   LD    (IX+00),+00        Indicar o número de tipo.
```

Os parâmetros que descrevem o programa, e as suas variáveis, são agora definidos e guardados na área de cabeçalho na zona de trabalho.

```
                 LD    HL,(E-LINE)        Indicador do final da área
                                          de variáveis.
                 LD    DE,(PROG)          Indicador do início do programa
                                          Basic.
                 SCF                      Realizar agora a subtracção para
                 SBC   HL,DE              definir o comprimento de 'programa
                 LD    (IX+0B),L          + variáveis'; guardar o
                 LD    (IX+0C),H          resultado.
                 LD    HL,(VARS)          Repetir a operação, guardando
                 SBC   HL,DE              desta vez apenas o comprimento
                 LD    (IX+0F),L          do 'programa'.
                 LD    (IX+10),H
                 EX    DE,HL              Transferir o indicador para
                                          HL como habitual.
```

A informação do cabeçalho é assim preparada para todos os casos.

A posição «IX+00» guarda o número de tipo.

As posições «IX+01» a «IX+0A» guardam o nome (+FF em IX+01 se for nulo).

As posições «IX+0B» e «IX+0C» guardam o número de bytes que existem no bloco de dados.

As posições «IX+0D» a «IX+10» guardam uma variedade de parâmetros cuja interpretação exacta depende do tipo.

A rotina continua, começando por separar SAVE de LOAD, VERIFY e MERGE.

```
075A SA-ALL      LD    A,(T-ADDR-lo)      Saltar para diante quando
                 AND   A                  trata uma ordem de SAVE.
                 JP    Z,0970,SA-CONTRL
```

No caso de um LOAD, VERIFY ou MERGE os primeiros dezassete bytes da área de cabeçalho no espaço de trabalho guardam a informação preparada, como se indicou acima; é agora chegado o momento de receber um «cabeçalho» da fita.

```
                 PUSH  HL                 Salvaguardar o indicador «destino».
                 LD    BC,+0011           Formar no par de registos IX o
                 ADD   IX,BC              endereço base da segunda
                                          «área de cabeçalho».
```

Entra-se agora num ciclo, que apenas será abandonado quando o cabeçalho tiver sido completamente carregado.

```
0767 LD-LOOK-H  PUSH  IX                  Copiar o endereço base.
                LD    DE,+0011            Carregar dezassete bytes.
                XOR   A                   Sinal «cabeçalho».
                SCF                       Sinal 'LOAD'.
                CALL  0556,LD-BYTES       Procurar o cabeçalho.
                POP   IX                  Recuperar o endereço base.
                JR    NC,0767,LD-LOOK-H   Percorrer o ciclo até ter
                                          êxito.
```

É agora impresso o novo «cabeçalho» no visor, mas a rotina só prossegue se o «novo» cabeçalho concorda com o antigo.

```
                LD    A,+FE               Verificar se o canal «S»
                CALL  1601,CHAN-OPEN      está aberto.
                LD    (SCR-CT),+03        Definir o contador de «scroll».
                LD    C,+80               Sinalizar «nomes não concordantes».
                LD    A,(IX+00)           Comparar o tipo «novo» com o
                CP    (IX-11)             «antigo».
                JR    NZ,078A,LD-TYPE     Saltar se os tipos não
                                          coincidirem.
                LD    C,+F6               Se coincidem: sinal «dez
                                          caracteres a comparar».
078A LD-TYPE    CP    +04                 Obviamente, o cabeçalho está
                JR    NC,0767,LD-LOOK-H   errado se «tipo 4 ou mais».
```

É impressa a mensagem «Program:», «Number array:», «Character array:» ou «Bytes:».

```
                LD    DE,+09C0            Endereço base do bloco de
                                          mensagens.
                PUSH  BC                  Salvaguardar o registo C
                CALL  0C0A,PO-MSG         enquanto é impressa a
                POP   BC                  mensagem adequada.
```

É impresso o «novo» nome, enquanto este é comparado com o «antigo».

```
                PUSH  IX                  Levar os registos DE a
                POP   DE                  apontarem para o «nome novo»
                LD    HL,+FFF0            e o par de registos HL para
                ADD   HL,DE               o «nome antigo».
                LD    B,+0A               Devem-se considerar dez
                                          caracteres.
                LD    A,(HL)              Saltar para diante se se
                INC   A                   compararam de facto 2 nomes.
                JR    NZ,07A6,LD-NAME
                LD    A,C                 Mas se «nome antigo» for
                ADD   A,B                 nulo, sinalizar «dez caracteres
                LD    C,A                 já concordantes».
```

Entra-se num ciclo que imprime os caracteres de «nome novo». O nome será aceite se o «contador» atingir finalmente zero.

```
07A6 LD-NAME    INC   DE                  Considerar cada caracter do
                LD    A,(DE)              nome novo alternadamente.
                CP    (HL)                Compará-lo com o caracter
                INC   HL                  correspondente do antigo.
                JR    NZ,07AD,LD-CH-PR    Não contar se não
                INC   C                   concordarem.
```



```
07AD LD-CH-PR   RST   0010,PRINT-A-1      Imprimir o «novo» caracter.
                DJNZ  07A6,LD-NAME        Realizar o ciclo para 10 caracteres.
                BIT   7,C                 Só aceitar o nome se o contador
                JR    NZ,0767,LD-LOOK-H   atingiu zero.
                LD    A,+0D               Seguir o «novo nome» de um
                RST   0010,PRINT-A-1      retorno de linha.
```

Foi encontrado o cabeçalho correcto, tendo chegado o momento de considerar as três ordens LOAD, VERIFY e MERGE separadamente.

```
                POP   HL                  Recuperar o indicador.
                LD    A,(IX+00)           'SCREEN$' e 'CODE' são
                CP    +03                 tratados com VERIFY.
                JR    Z,07CB,VR-CONTRL
                LD    A,(T-ADDR-lo)       Saltar para a frente se
                DEC   A                   se usa uma ordem LOAD.
                JP    Z,0808,LD-CONTRL
                CP    +02                 Saltar para a frente se
                JP    Z,08B6,ME-CONTRL    se usa uma ordem MERGE;
                                          continuar no caso de uma
                                          ordem VERIFY.
```

A rotina de controlo «VERIFY»

O processo de verificação envolve a carga de um bloco de dados, um byte de cada vez, mas os bytes não são guardados — apenas comparados. Esta rotina é igualmente usada para carregar blocos de dados que foram descritos por «SCREEN$ e CODE».

```
07CB VR-CONTRL  PUSH  HL                  Salvaguardar o indicador.
                LD    L,(IX-06)           Recuperar o número de bytes
                LD    H,(IX-05)           descrito no cabeçalho antigo.
                LD    E,(IX+0B)           Recuperar também o número de
                LD    D,(IX+0C)           bytes do cabeçalho novo.
                LD    A,H                 Saltar para diante se o
                OR    L                   comprimento não é especificado,
                JR    Z,07E9,VR-CONT-1    p. ex., só «LOAD nome CODE».
                SBC   HL,DE               Produzir mensagem R se se
                JR    C,0806,REPORT-R     tenta carregar um bloco maior
                                          do que o indicado.
                JR    Z,07E9,VR-CONT-1    Aceitar comprimentos iguais.
                LD    A,(IX+00)           Produzir também a mensagem R
                CP    +03                 se se verificam blocos de
                JR    NZ,0806,REPORT-R    tamanho diverso (comprimento
                                          «antigo» maior do que «novo»).
```

A rotina continua considerando o «indicador de destino».

```
07E9 VR-CONT-1  POP   HL                  Recuperar o indicador, isto é,
                                          o «início.»
                LD    A,H                 Este indicador será usado a
                OR    L                   menos que seja zero, caso em
                JR    NZ,07F4,VR-CONT-2   que se preferirá o «início»
                LD    L,(IX+0D)           referido no cabeçalho
                LD    H,(IX+0E)           novo.
```

Considera-se agora a «flag» VERIFY/LOAD, e realiza-se a carga.

```
07F4 VR-CONT-2   PUSH  HL              Deslocar o «indicador» para
                 POP   IX              o par de registos IX.
                 LD    A,(T-ADDR-lo)   Saltar para diante a menos
                 CP    +02             que se use a ordem VERIFY;
                 SCF                   a flag «carry» sinaliza
                 JR    NZ,0800,VR-CONT-3 'LOAD'.
                 AND   A               Sinal «VERIFY».
0800 VR-CONT-3   LD    A,+FF           Sinal «aceitar só bloco de dados»
                                       antes de carregar o bloco.
```

A subrotina «carregar um bloco de dados»

Esta subrotina é comum a todas as rotinas de carga. No caso de LOAD e VERIFY actua como um retorno das rotinas de tratamento da cassete, mas no caso de MERGE é ainda necessário executar a «mistura» do bloco de dados.

```
0802 LD-BLOCK    CALL  0556,LD-BYTES   LOAD/VERIFY um bloco de dados.
                 RET   C               Retorno a menos que haja erro.
```

Mensagem «R — Tape Loading error»

```
0806 REPORT-R    RST   0008,ERROR-1    Invocar a rotina de tratamento
                 DEFB  +1A             de erro.
```

A rotina de controlo de «LOAD»

Esta rotina controla a carga de um programa Basic, e as suas variáveis, ou um array.

```
0808 LD-CONTRL   LD    E,(IX+08)       Recuperar o «número de bytes»
                 LD    D,(IX+0C)       indicado pelo novo cabeçalho.
                 PUSH  HL              Salvaguardar o indicador «destino».
                 LD    A,H             Saltar para diante a menos que
                 OR    L               se tente carregar um array não
                 JR    NZ,0819,LD-CONT-1 declarado anteriormente.
                 INC   DE              Somar três bytes ao comprimento
                 INC   DE              para o nome a bytes maior e
                 INC   DE              menor de uma variável
                 EX    DE,HL           nova.
                 JR    0825,LD-CONT-2  Saltar para diante.
```

Verificar se existe espaço suficiente em memória para o novo bloco de dados.

```
0819 LD-CONT-1   LD    L,(IX-06)       Recuperar o tamanho de «programa+
                 LD    H,(IX-05)       +variáveis ou array» existente.
                 EX    DE,HL
                 SCF                   Saltar para diante se não for
                 SBC   HL,DE           necessário mais espaço; tendo
                 JR    C,082E,LD-DATA  em conta que se deve reclamar
                                       a memória em uso.
```


Verificar de facto o espaço disponível.

```
0825 LD-CONT-2   LD    DE,+0005        Permitir um atraso de 5
                 ADD   HL,DE           bytes.
                 LD    B,H             Passar o resultado para o
                 LD    C,L             par de registos BC e
                 CALL  1F05,TEST-ROOM  comparar.
```

Tratar agora a carga de arrays.

```
082E LD-DATA     POP   HL              Recuperar de novo o «indicador».
                 LD    A,(IX+00)       Saltar para diante se se carrega
                 AND   A               um programa Basic.
                 JR    Z,0873,LD-PROG
                 LD    A,H             Saltar para diante se se carrega
                 OR    L               um array novo.
                 JR    Z,084C,LD-DATA-1
                 DEC   HL              Recuperar o comprimento do
                 LD    B,(HL)          array existente lendo os bytes
                 DEC   HL              de comprimento da área de
                 LD    C,(HL)          variáveis.
                 DEC   HL              Apontar para nome antigo.
                 INC   BC              Somar três bytes ao comprimento,
                 INC   BC              um para o nome e dois para
                 INC   BC              o «comprimento».
                 LD    (X-PTR),IX      Salvaguardar o par de registos
                 CALL  19E8,RECLAIM-2  IX, temporariamente, enquanto
                 LD    IX,(X-PTR)      se reclama o antigo array.
```

Reserva-se agora espaço para o novo array — no final da actual área de variáveis.

```
084C LD-DATA-1   LD    HL,(E-LINE)     Descobrir o indicador do separador
                 DEC   HL              final da área de variáveis — «byte 80».
                 LD    C,(IX+0B)       Recuperar o «comprimento»
                 LD    B,(IX+0C)       do novo array.
                 PUSH  BC              Salvaguardar este comprimento.
                 INC   BC              Somar três bytes — um para
                 INC   BC              o nome e dois para o
                 INC   BC              «comprimento».
                 LD    A,(IX-03)       «IX+0E» do cabeçalho antigo
                                       indica o nome do quadro.
                 PUSH  AF              O nome é salvaguardado
                 CALL  1655,MAKE-ROOM  enquanto se reserva a
                 INC   HL              memória necessária, «BC»
                 POP   AF              espaços antes de «novo byte 80».
                 LD    (HL),A          Indica-se o nome.
                 POP   DE              Recupera-se o comprimento,
                 INC   HL              e indicam-se igualmente os
                 LD    (HL),E          seus dois bytes.
                 INC   HL
                 LD    (HL),D
                 INC   HL              HL aponta agora para a primeira
                                       posição que será preenchida
                                       com dados da fita.
                 PUSH  HL              Este endereço é passado ao
                 POP   IX              par de registos IX; a flag
```

```
                SCF                     «carry» passa a um; sinal «bloco de
                LD      A,+FF           dados»; e carrega-se de facto
                JP      0802,LD-BLOCK   o bloco.
```

Trata-se agora a carga de um programa Basic e das respectivas variáveis.

```
0873 LD-PROG    EX      DE,HL           Salvaguardar o indicador «destino».
                LD      HL,(E-LINE)     Descobrir o separador final
                DEC     HL              da área actual de variáveis —
                                        o «byte 80».
                LD      (X-PTR),IX      Salvaguardar IX temporariamente.
                LD      C,(IX+0B)       Recuperar o «comprimento» do
                LD      B,(IX+0C)       novo bloco de dados.
                PUSH    BC              Manter uma cópia de «compri-
                CALL    19E5,RECLAIM-1  mento» enquanto se reclama as áreas
                POP     BC              de programa e variáveis actuais.
                PUSH    HL              Salvaguardar o indicador da área
                PUSH    BC              de programa e o comprimento
                                        do novo bloco de dados.
                CALL    1655,MAKE-ROOM  Reservar espaço suficiente
                                        para o novo programa e as
                                        suas variáveis.
                LD      IX,(X-PTR)      Recuperar o par de registos IX.
                INC     HL              Definir também a variável de
                LD      C,(IX+0F)       sistema VARS para o novo
                LD      B,(IX+10)       programa.
                ADD     HL,BC
                LD      (VARS),HL
                LD      H,(IX+0E)       Se foi especificado um número
                LD      A,H             de linha, é também necessário
                AND     +C0             considerá-lo.
                JR      NZ,08AD,LD-PROG-1 Saltar se «sem número»; senão,
                LD      L,(IX+0D)       definir NEWPPC e NSPPC.
                LD      (NEWPPC),HL
                LD      (NSPPC),+00
```

Pode carregar-se agora o bloco de dados.

```
08AD LD-PROG-1  POP     DE              Recuperar «comprimento».
                POP     IX              Recuperar «início».
                SCF                     Sinalizar «LOAD».
                LD      A,+FF           Sinalizar só «bloco de dados».
                JP      0802,LD-BLOCK   Carregá-lo.
```

A rotina de controlo de «MERGE».

Existem três partes principais nesta rotina.

1. Carregar o bloco de dados para a área de trabalho.
2. Misturar as linhas do novo programa com as do antigo.
3. Misturar as novas variáveis e as antigas.

Começa-se, portanto, com a carga do bloco de dados.

```
08B6 ME-CONTRL  LD      C,(IX+0B)       Recuperar o «comprimento» do
                LD      B,(IX+0C)       bloco de dados.
                PUSH    BC              Copiar «comprimento».
```



```
                INC     BC              Reservar «comprimento+1» posições
                RST     0030,BC-SPACES  na área de trabalho.
                LD      (HL),+80        Colocar um separador final
                                        na posição extra.
                EX      DE,HL           Passar o indicador «início»
                                        para o par de registos HL.
                POP     DE              Recuperar o comprimento original.
                PUSH    HL              Copiar «início».
                PUSH    HL              Definir o par de registos
                POP     IX              IX para a carga.
                SCF                     Sinalizar «LOAD».
                LD      A,+FF           Sinalizar «só bloco de dados».
                CALL    0802,LD-BLOCK   Carregar o bloco de dados.
```

As linhas do novo programa são misturadas com as linhas do programa antigo.

```
                POP     HL              Recuperar o «início» do
                                        novo programa.
                LD      DE,(PROG)       Inicializar DE para o «início»
                                        do programa antigo.
```

Entrar num ciclo que trata as linhas do programa novo.

```
08D2 ME-NEW-LP  LD      A,(HL)          Obter um número de linha
                AND     +C0             e compará-lo.
                JR      NZ,08F0,ME-VAR-LP Saltar quando terminar
                                        todas as linhas.
```

Entrar num ciclo interior que trata as linhas do programa antigo.

```
08D7 ME-OLD-LP  LD      A,(DE)          Obter o byte «alto» do número
                INC     DE              de linha e compará-lo.
                CP      (HL)            Saltar para diante se não é
                INC     HL              igual, mas, de qualquer modo,
                JR      NZ,08DF,ME-OLD-L1 avançar os dois indicadores.
                LD      A,(DE)          Repetir a comparação para os
                CP      (HL)            bytes «baixos» dos n.os de linha.
08DF ME-OLD-L1  DEC     DE              Corrigir os indicadores.
                DEC     HL
                JR      NC,08EB,ME-NEW-L2 Saltar para diante se se
                                        encontrou o local correcto de
                                        uma linha do novo programa.
                PUSH    HL              Senão, determinar o início da
                EX      DE,HL           linha antiga seguinte.
                CALL    1988,NEXT-ONE
                POP     HL
                JR      08D7,ME-OLD-LP  Percorrer o ciclo para cada
                                        uma das linhas antigas.
08EB ME-NEW-L2  CALL    092C,ME-ENTER   Passar novamente ao ciclo
                JR      08D2,ME-NEW-LP  exterior, das linhas novas.
```

De um modo semelhante, misturam-se as variáveis do programa novo com as do programa antigo.

Entra-se num ciclo que trata, alternadamente, cada uma das novas variáveis.

```
08F0 ME-VAR-LP  LD    A,(HL)        Recuperar em separado cada
                LD    C,A           nome de variável e verificá-lo.
                CP    +80           Retorno depois de considerar
                RET   Z             todas as variáveis.
                PUSH  HL            Salvaguardar o indicador actual.
                LD    HL,(VARS)     Recuperar VARS (programa
                                    antigo.)
```

Entra-se agora num ciclo interior que investiga a área de variáveis existente.

```
08F9 ME-OLD-VP  LD    A,(HL)        Obter cada nome de variável
                CP    +80           e compará-la.
                JR    Z,0923,ME-VAR-L2  Saltar para diante depois de
                                    encontrar o separador final
                                    (fazer um «acrescento»).
                CP    C             Comparar os nomes (1.os bytes).
                JR    Z,0909,ME-OLD-V2  Saltar para diante desenvolvendo
                                    o assunto; voltar aqui se a
                                    semelhança não for total.
0901 ME-OLD-V1  PUSH  BC            Salvaguardar o nome da nova
                CALL  1988,NEXT-ONE variável enquanto é localizada
                POP   BC            a variável «antiga» seguinte.
                EX    DE,HL         Restaurar o indicador do
                JR    08F9,ME-OLD-VP  par de registos DE e percorrer
                                    novamente o ciclo.
```

Os primeiros bytes da variável nova e da antiga são iguais, mas as variáveis com nomes compridos devem ser mais bem estudadas.

```
0909 ME-OLD-V2  AND   +E0           Considerar os bits 7, 6 e 5.
                CP    +A0           Aceitar todos os tipos de
                JR    NZ,0921,ME-VAR-L1  variáveis excepto as de nome com-
                                    prido.
                POP   DE            Fazer DE apontar para o pri-
                PUSH  DE            meiro caracter do «novo nome».
                PUSH  HL            Salvaguardar o indicador do
                                    «nome antigo».
```

Entrar num ciclo que compara as letras dos nomes compridos.

```
0912 ME-OLD-V3  INC   HL            Actualizar os indicadores
                INC   DE            «antigo» e «novo».
                LD    A,(DE)        Comparar as duas letras.
                CP    (HL)
                JR    NZ,091E,ME-OLD-V4  Saltar para diante se não
                                    existe igualdade.
                RLA                 Percorrer o ciclo até ser
                JR    NC,0912,ME-OLD-V3  encontrado o último caracter.
                POP   HL            Obter o indicador do início
                                    do nome «antigo» e saltar para
                JR    0921,ME-VAR-L1  diante — com êxito.
091E ME-OLD-V4  POP   HL            Obter o indicador e saltar
                JR    0901,ME-OLD-V1  para trás — sem êxito.
```

Vir aqui se existe igualdade dos nomes.

```
0921 ME-VAR-L1  LD    A,+FF         Sinal «substituir» variável.
```

E vir aqui se não existe igualdade (A contém +80 — variável a «somar»).


```
0923 ME-VAR-L2  POP   DE            Obter indicador de nome «novo».
                EX    DE,HL         Comutar os registos.
                INC   A             A flag «zero» passa a um
                                    se há «substituição»; a zero
                                    se há «acrescento».
                SCF                 Sinal «tratando variáveis».
                CALL  092C,ME-ENTER Fazer a entrada.
                JR    08F0,ME-VAR-LP  Percorrer o ciclo para considerar
                                    a variável nova seguinte.
```

A subrotina «misturar linha ou variável»

Entra-se nesta subrotina com os seguintes parâmetros:

Flag «carry»	zero	— MERGE uma linha Basic.
	um	— MERGE uma variável.
Flag «zero»	zero	— Será um «acrescento».
	um	— Será uma «substituição».
Registos HL		— Indicam o início da nova entrada.
Registos DE		— Indicam para onde é realizado o MERGE.

```
092C ME-ENTER   JR    NZ,093E,ME-ENT-1  Saltar se se faz «acrescento».
                EX    AF,A'F'       Salvaguardar as flags.
                LD    (X-PTR),HL    Salvaguardar o «novo» indicador
                EX    DE,HL         enquanto é reclamada a linha
                CALL  1988,NEXT-ONE ou variável «antiga».
                CALL  19E8,RECLAIM-2
                EX    DE,HL
                LD    HL,(X-PTR)
                EX    AF,A'F'       Restaurar as flags.
```

Pode introduzir-se a nova entrada.

```
093E ME-ENT-1   EX    AF,A'F'       Salvaguardar as flags.
                PUSH  DE            Copiar o indicador
                                    «destino».
                CALL  1988,NEXT-ONE Determinar comprimento da
                                    nova variável/linha.
                LD    (X-PTR),HL    Salvaguardar o indicador da
                                    nova variável/linha.
                LD    HL,(PROG)     Recuperar PROG — para evitar
                                    corrupção.
                EX    (SP),HL       Salvaguardar PROG e obter
                                    o novo indicador.
                PUSH  BC            Salvaguardar o comprimento.
                EX    AF,A'F'       Recuperar as flags.
                JR    C,0955,ME-ENT-2  Saltar para diante se se
                                    acrescenta uma nova variável.
                DEC   HL            Acrescenta-se uma nova linha
                                    antes da posição «destino».
                CALL  1655,MAKE-ROOM  Reservar espaço para a nova linha.
                INC   HL
                JR    0958,ME-ENT-3 Saltar para diante.
0955 ME-ENT-2   CALL  1655,MAKE-ROOM  Reservar espaço para a nova
                                    variável.
0958 ME-ENT-3   INC   HL            Apontar para a 1.ª posição nova.
                POP   BC            Recuperar o comprimento.
```

```
POP     DE          Recuperar PROG e guardá-la
LD      (PROG),DE   no local correcto.
LD      DE,(X-PTR)  Obter também o «novo» indicador.
PUSH    BC          Salvaguardar de novo o compri-
PUSH    DE          mento e o «novo» indicador.
EX      DE,HL       Comutar os indicadores e
LDIR                copiar a «nova» variável/linha
                    para o espaço reservado.
```

Deve-se agora remover a «nova» variável/linha da área de trabalho.

```
POP     HL          Recuperar o «novo» indicador.
POP     BC          Recuperar o comprimento.
PUSH    DE          Salvaguardar o indicador
                    «antigo» (aponta a posição após
                    a variável/linha acrescentada).
CALL    19E8,RECLAIM-2  Remover a variável/linha da
                        área de trabalho.
POP     DE          Retorno com o indicador «antigo»
RET                 no par de registos DE.
```

A rotina de comando de «SAVE»

O modo de gravar um programa ou bloco de dados é bastante simples.

```
0970 SA-CONTRL  PUSH    HL          Salvaguardar o «indicador».
                LD      A,+FD       Garantir que o canal «K» está
                CALL    1601,CHAN-OPEN  aberto.
                XOR     A           Sinal «primeira mensagem».
                LD      DE,+09A1    Imprimir mensagem «Start tape,
                CALL    0C0A,PO-MSG then press any key».
                SET     5,(TV-FLAG) Sinal «necessário limpar o
                                    visor».
                CALL    15D4,WAIT-KEY   Esperar que seja premida uma tecla.
```

Depois de ser premida uma tecla é enviado o «cabeçalho».

```
PUSH    IX          Salvaguardar o endereço base do
                    cabeçalho no «stack».
LD      DE,+0011    Serão SAVEd dezassete
                    bytes.
XOR     A           Sinal «é um cabeçalho».
CALL    04C2,SA-BYTES   Enviar o cabeçalho; com um
                        byte inicial de «tipo» e um
                        byte final de paridade.
```

Segue-se um curto momento de espera antes de o programa/bloco de dados ser enviado.

```
                POP     IX          Recuperar o indicador do
                                    cabeçalho.
                LD      B,+32       O atraso é de cinquenta ciclos,
0991 SA-1-SEC   HALT                isto é, um segundo.
                DJNZ    0991,SA-1-SEC
                LD      E,(IX+0B)   Obter o comprimento do bloco
                LD      D,(IX+0C)   de dados que deve ser SAVEd.
```



```
LD      A,+FF       Sinalizar «bloco de dados».
POP     IX          Obter o «indicador de início do
JP      04C2,SA-BYTES   bloco» e SAVE o bloco.
```

As mensagens do tratamento de cassetes

Cada mensagem é indicada com o último caracter invertido (+ 80 hexadecimal).

```
09A1 DEFB +80   — Ultrapassa-se o byte inicial.
09A2 DEFM       — «Start tape, then press any key».
09C1 DEFM       — Retorno de linha — «Program:»
09CB DEFM       — Retorno de linha — «Number array:»
09DA DEFM       — Retorno de linha — «Character array:»
09EC DEFM       — Retorno de linha — «Bytes:»
```

5
AS ROTINAS DE TRATAMENTO DO VISOR E DA IMPRESSORA

As rotinas de «impressão»

Toda a impressão para a parte principal do visor, a parte inferior deste e a impressora, é tratada por este conjunto de rotinas.

Entra-se na rotina «PRINT-OUT» com o código de um caracter de comando, um caracter a imprimir ou uma palavra-chave no registo A.

```
09F4 PRINT-OUT  CALL  0803,PO-FETCH   Posição actual de impressão.
                CP    +20             Se o código representa um
                JP    NC,0AD9,PO-ABLE caracter para imprimir, saltar.
                CP    +06             Imprimir um ponto de interrogação
                JR    C,0A69,PO-QUEST para os códigos +00 a +05.
                CP    +18             E também para os códigos +18 a
                                      +1F.
                JR    NC,0A69,PO-QUEST Base da tabela de «comando».
                LD    HL,+0A0B        Passar o código para o par de
                LD    E,A             registos DE.
                LD    D,+00           Indexar a tabela e obter o
                ADD   HL,DE           deslocamento.
                LD    E,(HL)          Somar o deslocamento e fazer
                ADD   HL,DE           um salto indirecto para a
                PUSH  HL              subrotina apropriada.
                JP    0803,PO-FETCH
```

A tabela «caracter de comando»

Endereço	Deslocamento	Caracter	Endereço	Deslocamento	Caracter
0A11	4E	Separador «vírgula»	0A1A	4F	Não usado
0A12	57	EDIT	0A1B	5F	Comando de INK
0A13	10	Cursor para a esquerda	0A1C	5E	Comando de PAPER
0A14	29	Cursor para a direita	0A1D	5D	Comando de FLASH
0A15	54	Cursor para baixo	0A1E	5C	Comando de BRIGHT
0A16	53	Cursor para cima	0A1F	5B	Comando de INVERSE
0A17	52	DELETE	0A20	5A	Comando de OVER
0A18	37	ENTER	0A21	54	Comando AT
0A19	50	Não usado	0A22	53	Comando TAB

A subrotina «cursor para a esquerda»

Entra-se nesta subrotina contendo no registo B o número de linha actual e no registo C o número de coluna actual.


```
0A23 PO-BACK-1  INC   C               Deslocar 1 coluna para a esquerda.
                LD    A,+22           Aceitar a alteração a menos
                CP    C               que se esteja na primeira coluna.
                JR    NZ,0A3A,PO-BACK-3
                BIT   1,(FLAGS)       Se se trata da impressora
                JR    NZ,0A38,PO-BACK-2 saltar para diante.
                INC   B               Subir uma linha.
                LD    C,+02           Definir valor da coluna.
                LD    A,+18           Verificar se é topo de linha.
                CP    B               Nota: deveria ser +19.
                JR    NZ,0A3A,PO-BACK-3 Aceitar a mudança a menos
                                      que seja o topo do visor.
                DEC   B               Inaceitável, descer uma linha.
0A38 PO-BACK-2  LD    C,+21           Passar à primeira coluna.
0A3A PO-BACK-3  JP    0DD9,CL-SET     Retorno indirecto através de
                                      CL-SET e PO-STORE.
```

A subrotina «cursor para a direita»

Esta subrotina realiza uma operação idêntica à da ordem Basic «PRINT OVER 1; CHR$ 32; —».

```
0A3D PO-RIGHT   LD    A,(P-FLAG)      Obter P-FLAG e salvaguardar
                PUSH  AF              no «stack».
                LD    (P-FLAG),+01    Passar P-FLAG para OVER 1.
                LD    A,+20           Um «espaço».
                CALL  0B65,PO-CHAR    Imprimir caracter.
                POP   AF              Obter o valor antigo de
                LD    (P-FLAG),A      P-FLAG.
                RET                   Terminado.
                                      Nota: o programador esqueceu-se
                                      de sair por PO-STORE.
```

A subrotina «retorno de linha»

Se a impressão é realizada para a impressora, o retorno de linha provoca o esvaziamento do «buffer» da impressora. Se se imprime no visor é verificado «scroll?» antes de diminuir o número de linha.

```
0A4F PO-ENTER   BIT   1,(FLAGS)       Saltar para diante se se
                JP    NZ,0ECD,COPY-BUFF trata da impressora.
                LD    C,+21           Passar à primeira coluna.
                CALL  0C55,PO-SCR     «Scroll» se necessário.
                DEC   B               Descer uma linha.
                JP    0DD9,CL-SET     Retorno indirecto através de
                                      CL-SET e PO-STORE.
```

A subrotina «separador vírgula»

É manipulado o valor da coluna actual, e o registo A passa a conter +00 (para TAB 0) ou +10 (para TAB 16).

```
0A5F PO-COMMA   CALL  0B03,PO-FETCH   Porquê outra vez?
                LD    A,C             Número de coluna actual.
```

```
                DEC   A           Deslocar duas colunas para
                DEC   A           a direita e verificar.
                AND   +10         O registo A será +00 ou
                                  +10.
                JR    0AC3,PO-FILL Sair por PO-FILL.
```

A subrotina «imprimir ponto de interrogação»

É impresso um ponto de interrogação sempre que se realiza uma tentativa de imprimir um código que não pode ser impresso.

```
0A69 PO-QUEST   LD    A,+3F       O caracter «?».
                JR    0AD9,PO-ABLE Imprimir este caracter.
```

A rotina «caracteres de comando com operandos»

Os caracteres de comando entre INK e OVER requerem um único operando, enquanto os caracteres de comando AT e TAB requerem dois operandos.

A rotina presente provoca a salvaguarda do código do caracter de comando em TVDATA (byte baixo), do primeiro operando em TVDATA (byte alto) ou no registo A se é apenas necessário um operando, e do segundo operando no registo A.

```
0A6D PO-TV-2    LD    DE,+0A87    Salvaguardar o 1.º operando
                LD    (TVDATA-hi),A em TVDATA (alto) e alterar o
                JR    0A80,PO-CHANGE endereço da rotina de «saída»
                                  para PO-CONT (+0A87).
```

Entrar aqui quando se tratam os caracteres AT e TAB.

```
0A75 PO-2-OPER  LD    DE,+0A6D    O código de caracter será
                JR    0A7D,PO-TV-1 guardado em TVDATA (baixo), e o
                                  endereço da rotina de saída
                                  mudado para PO-TV-2 (+0A6D).
```

Entrar aqui ao tratar atributos de cor — INK a OVER.

```
0A7A PO-1-OPER  LD    DE,+0A87    A rotina de saída deve ser
                                  mudada para PO-CONT (+0A87).
0A7D PO-TV-1    LD    (TVDATA-lo),A Salvaguardar o caracter de co-
                                  mando.
```

É alterado temporariamente o actual endereço da rotina de «saída».

```
0A80 PO-CHANGE  LD    HL,(CURCHL) HL apontará para o endereço
                                  da rotina de saída.
                LD    (HL),E      Introduzir o novo endereço da
                INC   HL          rotina de saída, e forçar
                LD    (HL),D      assim à aceitação do código
                RET               seguinte como operando.
```

Depois de serem recolhidos os operandos, a rotina continua.


```
0A87 PO-CONT    LD    DE,+09F4    Restaurar o endereço original
                CALL  0A80,PO-CHANGE de PRINT-OUT (+09F4).
                LD    HL,(TVDATA) Obter o código de comando
                                  e o primeiro operando se
                                  existirem dois.
                LD    D,A         Desloca-se o «último» operando
                LD    A,L         e o código de comando.
                CP    +16         Saltar para diante se se trata
                JP    C,2211,CO-TEMPS INK a OVER.
                JR    NZ,0AC2,PO-TAB Saltar para diante se se trata
                                  TAB.
```

Trata-se agora o caracter de comando AT.

```
                LD    B,H         Número da linha.
                LD    C,D         Número da coluna.
                LD    A,+1F       Inverter o número de coluna;
                SUB   C           ou seja, +00 a +1F passa a
                                  +1F a +00.
                JR    C,0AAC,PO-AT-ERR Deve respeitar a gama aceite.
                ADD   A,+02       Somar o deslocamento dando
                LD    C,A         +21 a +22 no registo C.
                BIT   1,(FLAGS)   Saltar para diante se se
                JR    NZ,0ABF,PO-AT-SET trata da impressora.
                LD    A,+16       Inverter o número de linha;
                SUB   B           ou seja, +00 a +15 passa a
                                  +16 a +01.
0AAC PO-AT-ERR  JP    C,1E9F,REPORT-B Saltar para diante se apropriado.
                INC   A           A gama +16 a +01 transforma-se em
                LD    B,A         +17 a +02.
                INC   B           E agora +18 a +03.
                BIT   0,(TV-FLAG) Se se imprime na parte inferior
                JP    NZ,0C55,PO-SCR do visor, considerar se é
                                  necessário «scrolling».
                CP    (DF-SZ)     Produzir mensagem 5 «Out
                JP    C,0C86,REPORT-5 of screen», se necessário.
0ABF PO-AT-SET  JP    0DD9,CL-SET Retorno através de CL-SET
                                  e PO-STORE.
```

E tratar o caracter de comando TAB.

```
0AC2 PO-TAB     LD    A,H         Obter o primeiro operando.
0AC3 PO-FILL    CALL  0B03,PO-FETCH Posição de impressão actual.
                ADD   A,C         Somar o valor da coluna actual.
                DEC   A           Determinar a quantidade de
                AND   +1F         espaços (módulo 32) e retorno
                RET   Z           para o resultado zero.
                LD    D,A         Usar D como contador.
                SET   0,(FLAGS)   Suprimir «espaço inicial».
0AD0 PO-SPACE   LD    A,+20       Imprimir um número «D» de
                CALL  0C3B,PO-SAVE espaços.
                DEC   D
                JR    NZ,0AD0,PO-SPACE
                RET               Terminado.
```

Códigos de caracteres a imprimir

O caracter (ou caracteres) requeridos são impressos invocando PO-ANY seguido de PO-STORE.

```
0AD9 PO-ABLE      CALL  0B24,PO-ANY      Imprimir caracter(es) e
                                        continuar por PO-STORE.
```

A subrotina «guardar posição»

Os valores de linha e coluna da nova posição e o endereço «pixel» são guardados nas variáveis de sistema apropriadas.

```
0ADC PO-STORE     BIT   1,(FLAGS)       Saltar para diante se se
                  JR    NZ,0AFC,PO-ST-PR trata da impressora.
                  BIT   0,(TV-FLAG)     Saltar para diante se se
                  JR    NZ,0AF0,PO-ST-E trata da parte inferior do visor.
                  LD    (S-POSN),BC     Salvaguardar os valores
                  LD    (DF-CC),HL      relativos à parte principal do
                  RET                   visor. Retorno em seguida.
0AF0 PO-ST-E      LD    (S-POSNL),BC    Salvaguardar os valores
                  LD    (ECHO-E),BC     relativos à parte inferior do
                  LD    (DF-CCL),HL     visor.
                  RET                   Retorno em seguida.
0AFC PO-ST-PR     LD    (P-POSN),C      Salvaguardar os valores
                  LD    (PR-CC),HL      relativos ao buffer da impressora.
                  RET                   Retorno em seguida.
```

A subrotina de «recuperação da posição»

Os parâmetros da posição actual são recuperados a partir das variáveis de sistema apropriadas.

```
0B03 PO-FETCH     BIT   1,(FLAGS)       Saltar para diante se se
                  JR    NZ,0B1D,PO-F-PR trata da impressora.
                  LD    BC,(S-POSN)     Obter os valores relativos
                  LD    HL,(DF-CC)      à parte principal do visor;
                  BIT   0,(TV-FLAG)     retorno se era este o
                  RET   Z               objectivo.
                  LD    BC,(S-POSNL)    Se não, obter os valores
                  LD    HL,(DF-CCL)     relativos à parte inferior
                  RET                   do visor.
0B1D PO-F-PR      LD    C,(P-POSN)      Obter os valores relativos
                  LD    HL,(PR-CC)      ao buffer da impressora.
                  RET
```

A subrotina «imprimir quaisquer caracteres»

Os códigos de caracter normais, códigos de palavras-chave e de gráficos definidos pelo utilizador, e ainda os códigos gráficos, são tratados separadamente.

```
0B24 PO-ANY       CP    +80             Saltar para diante no caso de
                  JR    C,0B65,PO-CHAR  caracteres normais.
                  CP    +90             Saltar para diante no caso de
                  JR    NC,0B52,PO-T&UDG palavras-chave e UDG's.
                  LD    B,A             Deslocar o código gráfico.
                  CALL  0B38,PO-GR-1    Construir a forma gráfica.
```



```
                  CALL  0B03,PO-FETCH   HL foi afectado, portanto,
                                        «recuperar» novamente.
                  LD    DE,+5C92        Levar DE a apontar para o
                                        início da forma gráfica, MEMBOT.
                  JR    0B7F,PO-ALL     Saltar para diante para imprimir
                                        o caracter gráfico.
```

Os caracteres gráficos são construídos de maneira *ad hoc* na área do calculador, isto é, MEM-0 e MEM-1.

```
0B38 PO-GR-1      LD    HL,+5C92        MEMBOT.
                  CALL  0B3E,PO-GR-2    Invocar a subrotina seguinte
                                        duas vezes.
0B3E PO-GR-2      RR    B               Determinar o bit 0 (e depois
                  SBC   A,A             o bit 2) do código gráfico.
                  AND   +0F             O registo A conterá +00 ou +0F
                                        conforme o valor do bit
                                        do código.
                  LD    C,A             Salvaguardar o resultado em C.
                  RR    B               Determinar o bit 1 (e depois
                  SBC   A,A             o bit 3) do código gráfico.
                  AND   +F0             O registo A conterá +00 ou
                                        +F0.
                  OR    C               Os dois resultados são combinados.
                  LD    C,+04           O registo A guarda metade do
0B4C PO-GR-3      LD    (HL),A          caracter e terá de ser usado
                  INC   HL              quatro vezes.
                  DEC   C               Isto é feito primeiro para a
                  JR    NZ,0B4C,PO-GR-3 metade superior do caracter e
                  RET                   depois para a inferior.
```

Separam-se agora as palavras-chave e os gráficos definidos pelo utilizador.

```
0B52 PO-T&UDG     SUB   +A5             Saltar para diante (palavras-chave).
                  JR    NC,0B5F,PO-T
                  ADD   A,+15           Os códigos UDG são +00 a +0F.
                  PUSH  BC              Salvaguardar a posição actual
                                        no «stack».
                  LD    BC,(UDG)        Recuperar o endereço base da
                  JR    0B6A,PO-CHAR-2  área UDG e saltar para diante.
0B5F PO-T         CALL  0C10,PO-TOKENS  Imprimir a palavra e voltar
                  JP    0B03,PO-FETCH   através do PO-FETCH.
```

É identificada a forma de caracter requerida.

```
0B65 PO-CHAR      PUSH  BC              Salvaguarda a posição actual.
                  LD    BC,(CHARS)      Recupera o endereço base da área
                                        de caracteres.
0B6A PO-CHAR-2    EX    DE,HL           Guarda endereço de impressão.
                  LD    HL,+5C3B        Isto é FLAGS.
                  RES   0,(HL)          Permitir um espaço inicial.
                  CP    +20             Saltar para diante se o caracter
                  JR    NZ,0B76,PO-CHAR-3 não é um espaço.
                  SET   0,(HL)          Mas «suprimir» se for.
0B76 PO-CHAR-3    LD    H,+00           Passar o código de caracter
                  LD    L,A             para o par de registos HL.
                  ADD   HL,HL           O código de caracter é de facto
                  ADD   HL,HL           multiplicado por oito.
                  ADD   HL,HL
```

```
              ADD   HL,BC      É descoberto o endereço base
                               da forma do caracter.
              POP   BC         Recupera-se a posição actual
              EX    DE,HL      e passa-se o endereço base
                               para o par de registos DE.
```

A subrotina «imprimir todos os caracteres»

Esta subrotina é usada para imprimir todos os caracteres de 8*8 bits. No início, o par de registos DE contém o endereço base da forma do caracter, o registo HL o endereço de destino e o registo BC os actuais valores de «linha e coluna».

```
0B7F PR-ALL      LD    A,C        Recuperar o número de coluna.
                 DEC   A          Mover uma coluna para a direita.
                 LD    A,+21      Saltar para diante a menos que
                 JR    NZ,0B93,PR-ALL-1   seja indicada uma nova linha.
                 DEC   B          Descer uma linha.
                 LD    C,A        O número de coluna é +21.
                 BIT   1,(FLAGS)  Saltar para diante se se trata
                 JR    Z,0B93,PR-ALL-1    o visor.
                 PUSH  DE         Salvaguardar o endereço base
                 CALL  0ECD,COPY-BUFF     enquanto o buffer da impressora
                 POP   DE         é esvaziado.
                 LD    A,C        Copiar o novo número de coluna.
0B93 PR-ALL-1    CP    C          Verificar se é usada uma linha
                 PUSH  DE         nova. Se for, verificar
                 CALL  Z,0C55,PO-SCR      se é necessário «scroll» a
                 POP   DE         imagem.
```

Considera-se agora o estado actual de INVERSE e OVER.

```
                 PUSH  BC         Salvaguardar os valores de
                 PUSH  HL         posição e o endereço de destino
                                  no «stack».
                 LD    A,(P-FLAG) Obter P-FLAG e ler o bit 0.
                 LD    B,+FF      Preparar a «máscara-OVER» no
                 RRA              registo B; OVER 0 = +00 e
                 JR    C,0BA4,PR-ALL-2    OVER 1 = +FF.
                 INC   B
0BA4 PR-ALL-2    RRA              Ler o bit 2 de P-FLAG e
                 RRA              preparar a «máscara-INVERSE»
                 SBC   A,A        no registo C; ou seja,
                 LD    C,A        INVERSE 0 = +00 e INVERSE
                                  1 = +FF.
                 LD    A,+08      Colocar no registo A o
                 AND   A          contador de linhas de pixels
                                  e limpar a flag «carry».
                 BIT   1,(FLAGS)  Saltar para diante se se trata
                 JR    Z,0BB6,PR-ALL-3    o visor.
                 SET   1,(FLAGS2) Sinal «buffer da impressora já
                                  não está vazio».
                 SCF              Passar a um a flag «carry» —
                                  — a impressora está a ser usada.
0BB6 PR-ALL-3    EX    DE,HL      Trocar o endereço de destino
                                  pelo endereço base antes de
                                  iniciar o ciclo.
```


Pode agora imprimir-se o caracter. O ciclo é percorrido oito vezes — uma para cada linha de pixels.

```
0BB7 PR-ALL-4    EX    AF,A'F'    A flag «carry» está a um quando
                                  se usa a impressora. Guardá-la em F'.
                 LD    A,(DE)     Recuperar a «linha de pixels» actual.
                 AND   B          Usar a «máscara-OVER» e depois
                 XOR   (HL)       XOR o resultado com a «linha de
                                  pixels» da forma do caracter.
                 XOR   C          Considerar finalmente a
                                  «máscara-INVERSE».
                 LD    (DE),A     Introduzir resultado.
                 EX    AF,A'F'    Recuperar a flag da impressora
                 JR    C,0BD3,PR-ALL-6    e saltar para diante se necessário.
                 INC   D          Actualizar o endereço de destino.
0BC1 PR-ALL-5    INC   HL         Actualizar a «linha de pixels»
                                  da forma de caracter.
                 DEC   A          Diminuir o contador e voltar
                 JR    NZ,0BB7,PR-ALL-4   ao ciclo a menos que seja zero.
```

Depois de o caracter ter sido impresso, define-se o byte de atributos do modo requerido.

```
                 EX    DE,HL      Colocar no registo H o byte alto
                 DEC   H          correcto do endereço da
                                  área de caracteres.
                 BIT   1,(FLAGS)  Definir o byte de atributos
                 CALL  Z,0BD8,PO-ATTR     apenas no caso do visor.
                 POP   HL         Restaurar o endereço original
                 POP   BC         de destino e os valores de
                                  posição.
                 DEC   C          Diminuir o número de coluna
                 INC   HL         e aumentar o endereço de
                 RET              destino antes do retorno.
```

Quando se está a usar a impressora, é necessário actualizar o endereço de destino em incrementos de +20.

```
0BD3 PO-ALL-6    EX    AF,A'F'    Guardar de novo a flag da impres-
                                  sora.
                 LD    A,+20      Valor do incremento requerido
                 ADD   A,E        Somar o valor e devolver
                 LD    E,A        o resultado ao registo E.
                 EX    AF,A'F'    Recuperar a flag.
                 JR    0BC1,PR-ALL-5      Saltar atrás para o ciclo.
```

A subrotina «definir byte de atributos»

É identificado e recuperado o byte de atributos apropriado. É formado o novo valor manipulando o antigo, ATTR-T, MASK-T e P-FLAG. Finalmente, copia-se o novo valor para a área de atributos.

```
0BD8 PO-ATTR     LD    A,H        O byte alto do endereço
                 RRCA             de destino é dividido por
                 RRCA             oito e sofre uma AND
                 RRCA             com +03 para determinar qual
                 AND   +03        o terço da imagem que é
                                  endereçado; 00, 01 ou 02.
```

```
                OR      +58             O byte alto da área de atributos
                LD      H,A             é formado em seguida.
                LD      DE,(ATTR-T)     D contém ATTR-T e E
                                        contém MASK-T.
                LD      A,(HL)          Antigo valor de atributos.
                XOR     E               Os valores de MASK-T e
                AND     D               ATTR-R são tomados em
                XOR     E               consideração.
                BIT     6,(P-FLAG)      Saltar para diante a menos
                JR      Z,0BFA,PO-ATTR-1 que se trate de PAPER 9.
                AND     +C7             A antiga cor de papel é
                                        ignorada, passando a negro
                BIT     2,A             (000) ou branco (111) em
                JR      NZ,0BFA,PO-ATTR-1 função da cor de tinta ser
                XOR     +38             clara ou escura.
0BFA PO-ATTR-1  BIT     4,(P-FLAG)      Saltar para diante a menos
                JR      Z,0C08,PO-ATTR-2 que se trate de INK 9.
                AND     +F8             A antiga cor de tinta é
                                        ignorada e passa a negro
                BIT     5,A             (000) ou branco (111) em
                JR      NZ,0C08,PO-ATTR-2 função de a cor de papel
                XOR     +07             ser clara ou escura.
0C08 PO-ATTR-2  LD      (HL),A          Introduzir o novo valor de
                RET                     atributos. Retorno.
```

A subrotina «impressão de mensagem»

Esta subrotina é usada para imprimir mensagens e palavras-chave. O registo A contém o «número correspondente» à mensagem ou palavra numa tabela. O par de registos DE contém o endereço base desta tabela.

```
0C0A PO-MSG     PUSH    HL              O byte alto da última entrada
                LD      H,+00           no «stack» é passado a zero
                EX      (SP),HL         a fim de suprimir espaços a
                                        mais (ver abaixo).
                JR      0C14,PO-TABLE   Saltar para diante.
```

Entrada aqui quando se ampliam os códigos de palavras-chave.

```
0C10 PO-TOKENS  LD      DE,+0095        Endereço base da tabela
                                        de palavras-chave.
                PUSH    AF              Guardar o código no «stack»
                                        (gama +00/+5A; RND-COPY).
```

Procura-se na tabela e imprime-se a entrada correcta.

```
0C14 PO-TABLE   CALL    0C41,PO-SEARCH  Localiza a entrada adequada.
                JR      C,0C22,PO-EACH  Imprime a mensagem/palavra.
                LD      A,+20           É impresso um «espaço» antes
                BIT     0,(FLAGS)       da mensagem/palavra se
                CALL    Z,0C3B,PO-SAVE  necessário.
```

São impressos um a um os caracteres da mensagem/palavra.

```
0C22 PO-EACH    LD      A,(DE)          Recolher um código.
                AND     +7F             Anular qualquer «bit invertido».
```



```
                CALL    0C3B,PO-SAVE    Imprimir o caracter.
                LD      A,(DE)          Recolher novamente o código.
                INC     DE              Avançar o indicador.
                ADD     A,A             O «bit invertido» passa à
                JR      NC,0C22,PO-EACH flag «carry» e sinaliza o
                                        final da mensagem/palavra;
                                        se não, saltar para trás.
```

Considerar se é necessário um espaço a mais no final.

```
                POP     DE              Mensagens — D contém +00;
                                        Palavras — D contém +00/+5A.
                CP      +48             Saltar para diante se o último
                JR      Z,0C35,PO-TRSP  caracter era um «$».
                CP      +82             Retorno se o último caracter
                RET     C               era qualquer outro antes de «A».
0C35 PO-TRSP    LD      A,D             Examinar o valor em D e
                CP      +03             retornar se indicar mensagem,
                RET     C               RND, INKEY$ ou PI.
                LD      A,+20           Todos os outros casos exigem
                                        um espaço a mais.
```

A subrotina «PO-SAVE»

Esta subrotina permite a impressão «recorrente» de caracteres. São salvaguardados os registos apropriados enquanto é invocada PRINT-OUT.

```
0C3B PO-SAVE    PUSH    DE              Guardar o par de registos DE.
                EXX                     Guardar HL e BC.
                RST     0010,PRINT-A-1  Imprimir o caracter isolado.
                EXX                     Restaurar HL e BC.
                POP     DE              Restaurar DE.
                RET                     Terminado.
```

A subrotina «Procura em tabela»

Esta subrotina termina com o par de registos DE apontando para o caracter inicial da entrada requerida em tabela, e a flag «carry» a zero se se deve considerar um «espaço inicial».

```
0C41 PO-SEARCH  PUSH    AF              Salvaguardar o «número da en-
                                        trada».
                EX      DE,HL           HL contém o endereço base.
                INC     A               Passar a gama a +01-?.
0C44 PO-STEP    BIT     7,(HL)          Esperar um «caracter
                INC     HL              invertido».
                JR      Z,0C44,PO-STEP
                DEC     A               Contar as entradas até achar
                JR      NZ,0C44,PO-STEP a correcta.
                EX      DE,HL           DE aponta para o caracter inicial.
                POP     AF              Recuperar o «número de entrada»
                CP      +20             e retorno com flag «carry» a um
                RET     C               para as primeiras 32 entradas.
                LD      A,(DE)          Porém, se o caracter inicial é
                SUB     +41             uma letra, pode ser necessário
                RET                     um espaço inicial.
```

A subrotina «Teste de scroll»

Esta subrotina é invocada sempre que possa haver necessidade de «rolar» a imagem. Isso acontece em três ocasiões: quando se trata um caracter «retorno de linha»; quando se usa AT numa linha de INPUT; quando a linha actual está cheia e se deve passar à linha seguinte.

Ao iniciar a rotina, o registo B guarda o número de linha que está a ser verificado.

```
0C55 PO-SCR     BIT   1,(FLAGS)       Retorno imediato se está a
                RET   NZ               ser usada a impressora.
                LD    DE,+0DD9         Carregar no «stack» o
                PUSH  DE               endereço de «CL-SET».
                LD    A,B              Transferir o número de linha.
                BIT   0,(TV-FLAG)      Saltar para diante se se
                JP    NZ,0D02,PO-SCR-4 considera «INPUT... AT...»
                CP    (DF-SZ)          Retorno, por CL-SET, se o
                JR    C,0C86,REPORT-5  número de linha é maior do que
                RET   NZ               o valor de DF-SZ; mensagem 5 se
                                       for menor; continuar de outro modo.
                BIT   4,(TV-FLAG)      Saltar para diante a menos que se
                JR    Z,0C88,PO-SCR-2  trate de «listagem automática».
                LD    E,(BREG)         Recuperar o contador de linha.
                DEC   E                Diminuir este contador.
                JR    Z,0CD2,PO-SCR-3  Saltar para diante se a listagem
                                       deve ser «scrolled».
                LD    A,+00            Senão, abrir canal «K»,
                CALL  1601,CHAN-OPEN   restaurar o indicador de «stack»,
                LD    SP,(LIST-SP)     sinalizar que a listagem
                RES   4,(TV-FLAG)      automática terminou e retorno
                RET                    através de CL-SET.
```

Mensagem «5 — Out of screen»

```
0C86 REPORT-5   RST   0008,ERROR-1     Invocar rotina de tratamento
                DEFB  +04              de erro.
```

Considerar agora se é necessária a pergunta «Scroll?»

```
0C88 PO-SCR-2   DEC   (SCR-CT)         Diminuir o contador de «scroll»
                JR    NZ,0CD2,PO-SCR-3 e continuar para fazer apenas
                                       a pergunta se atingir zero.
```

Apresentar a mensagem.

```
                LD    A,+18            Contador passado a zero.
                SUB   B
                LD    (SCR-CT),A
                LD    HL,(ATTR-T)      São guardados os valores actuais
                PUSH  HL               de ATTR-T e MASK-T.
                LD    A,(P-FLAG)       É guardado o valor de
                PUSH  AF               P-FLAG.
                LD    A,+FD            É aberto o canal «K».
                CALL  0601,CHAN-OPEN
                XOR   A                A mensagem «scroll?» é a
                LD    DE,+0CF8         mensagem «0». Esta é agora
                CALL  0C0A,PO-MSG      impressa.
```



```
                SET   5,(TV-FLAG)      Sinal «limpar a parte inferior do
                                       visor depois de premida uma tecla».
                LD    HL,+5C3B         Isto é FLAGS.
                SET   3,(HL)           Sinal «modo L».
                RES   5,(HL)           Sinal «tecla não premida».
                EXX                    Nota: DE deve também ser guardado.
                CALL  15D4,WAIT-KEY    Recuperar o código da tecla.
                EXX                    Restaurar os registos.
                CP    +20              Salto para diante para
                JR    Z,0D00,REPORT-D  mensagem «D — BREAK-CONT
                CP    +E2              repeats» se a tecla é BREAK,
                JR    Z,0D00,REPORT-D  STOP, N ou n; senão, aceitar
                OR    +20              a tecla como indicando
                CP    +6E              a necessidade de «rolar» a
                JR    Z,0D00,REPORT-D  imagem.
                LD    A,+FE            Abrir canal «S».
                CALL  1601,CHAN-OPEN
                POP   AF               Restaurar o valor de
                LD    (P-FLAG),A       P-FLAG.
                POP   HL               Restaurar os valores de
                LD    (ATTR-T),HL      ATTR-T e MASK-T.
```

A imagem é agora deslocada.

```
0CD2 PO-SCR-3   CALL  0DFE,CL-SC-ALL   Toda a imagem é rolada.
                LD    B,(DF-SZ)        Os números de linha e coluna
                INC   B                do início da linha acima da
                LD    C,+21            parte inferior do visor são
                PUSH  BC               achados e salvaguardados.
                CALL  0E9B,CL-ADDR     É então determinado o byte
                LD    A,H              de atributos correspondente
                RRCA                   a esta área de caracter. O
                RRCA                   par de registos HL guarda o
                RRCA                   endereço do byte.
                AND   +03
                OR    +58
                LD    H,A
```

A linha em questão terá valores de «atributos» equivalentes aos da parte inferior do visor, e a nova linha na parte inferior da imagem pode ter valores ATTR-P, pelo que os valores dos atributos são trocados.

```
                LD    DE,+5AE0         DE aponta para o primeiro byte
                                       de atributos da linha inferior.
                LD    A,(DE)           O valor é recuperado.
                LD    C,(HL)           Valor «parte inferior».
                LD    B,+20            Existem 32 bytes.
                EX    DE,HL            Trocar os indicadores.
0CF0 PO-SCR-3A  LD    (DE),A           Fazer a primeira troca e
                LD    (HL),C           usar os mesmos valores
                INC   DE               para os trinta e dois bytes
                INC   HL               de atributos das duas linhas
                DJNZ  0CF0,PO-SCR-3A   que estão a ser tratadas.
                POP   BC               Os números de linha e coluna
                                       da linha inferior da «parte
                RET                    superior» são recuperados
                                       antes do retorno.
```

Mensagem «scroll?»

```
0CF8            DEFB  +80              Separador inicial — ultrapassado.
                DEFB  +73,+63,+72,+6F  s-c-r-o
                DEFB  +6C,+6C,+BF      l-l-? (invertido).
```

Mensagem «D — BREAK — CONT repeats»

```
0D00 REPORT-D   RST   0008,ERROR-1     Invocar a rotina de tratamento
                DEFB  +0C              de erro.
```

A parte inferior do visor é tratada do seguinte modo:

```
0D02 PO-SCR-4   CP    +02              O erro «out of screen» é
                JR    C,0C86,REPORT-5  enviado se a parte inferior
                ADD   A,(DF-SZ)        vai ser «demasiado grande»;
                SUB   +19              retorno se o «scrolling» é
                RET   NC               desnecessário.
                NEG                    O registo A conterá o número
                                       de «scrolls» que devem
                                       ser realizados.
                PUSH  BC               Guarda os números de linha e
                                       coluna.
                LD    B,A              São salvaguardados o «número
                LD    HL,(ATTR-T)      de scroll», ATTR-T, MASK-T
                PUSH  HL               e P-FLAG.
                LD    HL,(P-FLAG)
                PUSH  HL
                CALL  0D4D,TEMPS       Serão usados os atributos de
                                       cor «permanentes».
                LD    A,B              Recupera-se o «número de scroll».
```

A parte inferior do visor é agora «scrolled» um número «A» de vezes.

```
0D1C PO-SCR-4A  PUSH  AF               Guardar o «número».
                LD    HL,+5C6B         Isto é DF-SZ.
                LD    B,(HL)           O valor em DF-SZ é
                LD    A,B              incrementado; o registo B
                INC   A                passa a conter o valor inicial
                LD    (HL),A           e o registo A o novo valor.
                LD    HL,+5C89         Isto é S-POSN (alto).
                CP    (HL)             O salto é dado se só deve
                JR    C,0D2D,PO-SCR-4B ser «scrolled» a parte inferior
                                       do visor (B=antigo DF-SZ).
                INC   (HL)             Se não, S-POSN (alto) é
                LD    B,+18            incrementado, rolando-se toda
                                       a imagem (B = +18).
0D2D PO-SCR-4B  CALL  0E00,CL-SCROLL    «Scroll» B linhas.
                POP   AF               Recuperar e decrementar
                DEC   A                o «número de scroll».
                JR    NZ,0D1C,PO-SCR-4A Saltar para trás se terminou.
                POP   HL               Restaurar o valor de
                LD    (P-FLAG),L       P-FLAG.
                POP   HL               Restaurar os valores de
                LD    (ATTR-T),HL      ATTR-T e MASK-T.
                LD    BC,(S-POSN)      Se S-POSN se alterou, chamar
                RES   0,(TV-FLAG)      CL-SET para fornecer um valor
                CALL  0DD9,CL-SET      correspondente a DF-CC.
                SET   0,(TV-FLAG)      Passar a flag a zero para
                POP   BC               indicar que se está a tratar
                RET                    a parte inferior do visor,
                                       recuperar os números de linha e
                                       coluna, e retorno.
```


A subrotina «Atributos de cor temporários»

Trata-se de uma subrotina extremamente importante. É usada sempre que é requerida a cópia dos «detalhes» permanentes para as variáveis de sistema «temporárias». Primeiro considera-se ATTR-T, e depois MASK-T.

```
0D4D TEMPS      XOR   A                A passa a conter +00.
                LD    HL,(ATTR-P)      São recuperados os valores
                BIT   0,(TV-FLAG)      actuais de ATTR-P e MASK-P.
                JR    Z,0D5B,TEMPS-1   Salto para diante se se trata
                                       a parte superior do visor.
                LD    H,A              Se não, usa-se +00 e o valor
                LD    L,(BORDCR)       em BORDCR.
0D5B TEMPS-1    LD    (ATTR-T),HL      Definir agora ATTR-T e MASK-T.
```

Em seguida, considera-se P-FLAG.

```
                LD    HL,+5C91         Isto é P-FLAG.
                JR    NZ,0D65,TEMPS-2  Saltar para diante se se trata
                                       a parte inferior do visor
                                       (A = +00).
                LD    A,(HL)           Senão, recuperar o valor de
                RRCA                   P-FLAG e passar os bits ímpares
                                       para os pares.
0D65 TEMPS-2    XOR   (HL)             Copiar os bits pares de A para
                AND   +55              P-FLAG.
                XOR   (HL)
                LD    (HL),A
                RET
```

A rotina «Comando CLS»

Primeiro, é limpo todo o visor — os pixels passam todos a zero e os bytes de atributos são passados para o valor de ATTR-P; depois é reconstituída a parte inferior do visor.

```
0D6B CLS        CALL  0DAF,CL-ALL      É limpo o conjunto do
                                       visor.
0D6E CLS-LOWER  LD    HL,+5C3C         Isto é TV-FLAG.
                RES   5,(HL)           Sinal «não limpar parte
                                       inferior após tecla premida».
                SET   0,(HL)           Sinal «parte inferior».
                CALL  0D4D,TEMPS       Usar os valores permanentes,
                                       ou seja, ATTR-T é copiada
                                       de BORDCR.
                LD    B,(DF-SZ)        A parte inferior do visor é
                CALL  0E44,CL-LINE     agora «limpa» com estes valores.
```

Exceptuando os bytes de atributos para as linhas 22 e 23, os das linhas da parte inferior do visor devem ser igualados ao valor de ATTR-P.

```
                LD    HL,+5AC0         Byte de atributos no início
                                       da linha 22.
                LD    A,(ATTR-P)       Recuperar ATTR-P.
                DEC   B                Contador de linha.
                JR    0D8E,CLS-3       Saltar para diante, para ciclo.
0D87 CLS-1      LD    C,+20            +20 caracteres por linha.
```

```
0D89 CLS-2      DEC   HL              Voltar atrás através da
                LD    (HL),A          definição dos bytes de atributos.
                DEC   C
                JR    NZ,0D89,CLS-2
0D8E CLS-3      DJNZ  0D87,CLS-1      Continuar o ciclo até terminar.
```

Pode agora fixar-se a dimensão da parte inferior do visor.

```
                LD    (DF-SZ),+02     Terá uma dimensão de duas linhas.
```

Resta realizar as seguintes tarefas de «limpeza da casa».

```
0D94 CL-CHAN    LD    A,+FD           Abrir o canal «K».
                CALL  1601,CHAN-OPEN
                LD    HL,(CURCHL)     Recuperar o endereço do
                LD    DE,+09F4        canal actual e definir o
                AND   A               endereço de saída +09F4
0DA0 CL-CHAN-A  LD    (HL),E          (=PRINT-OUT) e o de
                INC   HL              entrada +10A8
                LD    (HL),D          (=KEY-INPUT).
                INC   HL
                LD    DE,+10A8
                CCF                   Primeiro, o endereço de saída,
                JR    C,0DA0,CL-CHAN-A depois o de entrada.
                LD    BC,+1721        No tratamento da parte
                                      inferior do visor, a «linha
                                      de impressão inferior» será
                                      a linha 23.
                JR    0DD9,CL-SET     Retorno através de CL-SET.
```

A subrotina «Limpar toda a área de imagem»

Esta subrotina é invocada por: 1) rotina do comando CLS; 2) rotina executiva principal; e 3) rotina de listagem automática.

```
0DAF CL-ALL     LD    HL,+0000        A variável de sistema COORDS
                LD    (COORDS),HL     é passada a zero.
                RES   0,(FLAGS2)      Sinal «visor limpo».
                CALL  0D94,CL-CHAN    Realizar tarefas de
                                      «limpeza da casa».
                LD    A,+FE           Abrir o canal «S».
                CALL  1601,CHAN-OPEN
                CALL  0D4D,TEMPS      Usar os valores «permanentes».
                LD    B,+18           «Limpar» as 24 linhas do
                CALL  0E44,CL-LINE    visor.
                LD    HL,(CURCHL)     Assegurar que o endereço
                LD    DE,+09F4        de saída actual é +09F4
                LD    (HL),E          (PRINT-OUT).
                INC   HL
                LD    (HL),D
                LD    (SCR-CT),+01    Passar a zero o contador de «scroll».
                LD    BC,+1821        No tratamento da parte superior
                                      do visor a «linha de impressão
                                      superior» será a linha zero.
                                      Continuar para CL-SET.
```


A subrotina «CL-SET»

Entra-se nesta subrotina com os números de linha e coluna de uma área de caracter no par de registos BC, ou o número de coluna no buffer da impressora no registo C. É então determinado o endereço apropriado do primeiro bit do caracter. A subrotina termina através do PO-STORE, guardando todos os valores nas variáveis de sistema requeridas.

```
0DD9 CL-SET     LD    HL,+5B00        Início do buffer da impressora.
                BIT   1,(FLAGS)       Saltar para diante se se trata
                JR    NZ,0DF4,CL-SET-2 o buffer da impressora.
                LD    A,B             Transferir o número de linha.
                BIT   0,(TV-FLAG)     Saltar para diante se se trata
                JR    Z,0DEE,CL-SET-1 a parte principal do visor.
                ADD   A,(DF-SZ)       A linha superior da janela
                SUB   +18             inferior do visor é chamada
                                      «linha+18» e isto deve ser
                                      convertido.
0DEE CL-SET-1   PUSH  BC              Salvaguarda os números de
                                      linha e coluna.
                LD    B,A             Desloca o número de linha.
                CALL  0E98,CL-ADDR    Forma-se em HL o endereço
                                      do início da linha.
                POP   BC              Recupera os números de
                                      linha e coluna.
0DF4 CL-SET-2   LD    A,+21           O número de coluna é agora
                SUB   C               invertido e transferido
                LD    E,A             para o par de registos DE.
                LD    D,+00
                ADD   HL,DE           Forma-se o endereço requerido;
                JP    0ADC,PO-STORE   e este, junto com os números
                                      de linha e coluna, são
                                      guardados saltando para
                                      PO-STORE.
```

A subrotina «Scrolling»

O número de linhas do visor que devem ser «scrolled» é guardado, ao entrar na rotina principal, no registo B.

```
0DFE CL-SC-ALL  LD    B,+17           Ponto de entrada após «scroll?»
```

O principal ponto de entrada — de cima e quando se executa o «scroll» para INPUT...AT.

```
0E00 CL-SCROLL  CALL  0E98,CL-ADDR    Descobrir o endereço inicial
                                      da linha.
                LD    C,+08           Existem oito linhas de pixels
                                      numa linha completa.
```

Entra-se agora no ciclo principal de «scroll». O registo B contém o número da linha superior a «rolar», o par de registos HL o endereço inicial desta linha no ficheiro de imagem, e o registo C o contador de linhas de pixels.

```
0E05 CL-SCR-1   PUSH  BC          Salvaguardar ambos os contadores.
                PUSH  HL          Guardar o endereço inicial.
                LD    A,B         Saltar para diante excepto se
                AND   +07         se trata actualmente um «terço»
                LD    A,B         do visor.
                JR    NZ,0E19,CL-SCR-3
```

As linhas de pixels das linhas superiores dos «terços» da imagem devem ser deslocadas de 2K (cada terço = 2K).

```
0E0D CL-SCR-2   EX    DE,HL       O resultado desta acção é
                LD    HL,+F8E0    deixar HL na mesma e DE
                ADD   HL,DE       indicando o destino
                EX    DE,HL       requerido.
                LD    BC,+0020    Existem +20 caracteres.
                DEC   A           Diminuir o contador quando
                                  é tratada uma linha.
                LDIR              Deslocar agora os 32 bytes.
```

As linhas de pixels no interior de cada «terço» podem agora ser roladas. O registo A contém, na primeira passagem, +01 e +07, +09 e +0F ou +11 e +17.

```
0E19 CL-SCR-3   EX    DE,HL       DE é levado novamente a
                LD    HL,+FFE0    apontar para o destino requerido.
                ADD   HL,DE       Desta vez a apenas 32 posições
                EX    DE,HL       de distância.
                LD    B,A         Guardar o número de linha em B.
                AND   +07         Determinar quantos caracteres
                RRCA              restam ainda no «terço»
                RRCA              considerado.
                RRCA
                LD    C,A         Passar o «total de caracteres»
                                  para o registo C.
                LD    A,B         Recuperar o número de linha.
                LD    B,+00       BC contém o «total de caracteres»
                LDIR              e é «scrolled» uma linha de
                                  pixels de cada caracter.
                LD    B,+07       Prepara para incrementar
                                  o endereço de salto para um
                                  novo «terço».
                ADD   HL,BC       Aumentar HL de +0700.
                AND   +F8         Saltar atrás se restam ainda
                JR    NZ,0E0D,CL-SCR-2  «terços».
```

Verificar agora se o ciclo foi usado oito vezes — uma para cada linha de pixels.

```
                POP   HL          Recuperar o endereço inicial.
                INC   H           Endereçar a linha de pixels se-
                                  guinte.
                POP   BC          Recuperar os contadores.
                DEC   C           Diminuir o contador de linhas de
                JR    NZ,0E05,CL-SR-1  pixels e saltar atrás a menos que
                                  tenham sido movidas 8 linhas.
```

Em seguida, são rolados os bytes de atributos. Note-se que o registo B contém ainda o número de linhas a rolar e que o registo C contém zero.


```
                CALL  0E88,CL-ATTR  É determinado o endereço
                                    requerido do ficheiro de
                                    atributos e o número de
                                    caracteres em «B» linhas.
                LD    HL,+FFE0      O deslocamento de todos os
                ADD   HL,DE         bytes de atributos é de 32
                EX    DE,HL         posições.
                LDIR                Os atributos são agora
                                    «rolados».
```

Resta agora limpar a linha inferior do visor.

```
                LD    B,+01         Carrega-se +01 no registo B
                                    e passa-se a CL-LINE.
```

A subrotina «Limpar linhas»

Esta subrotina limpa as «B» linhas inferiores do visor.

```
0E44 CL-LINE    PUSH  BC          O número de linha é guardado
                                  enquanto dura a subrotina.
                CALL  0E9B,CL-ADDR  O endereço inicial da linha é
                                  formado em HL.
                LD    C,+08       É novamente necessário
                                  considerar 8 linhas de pixels.
```

Entrar agora num ciclo que limpe todas as linhas de pixels.

```
0E4A CL-LINE-1  PUSH  BC          Guardar o número de linha e
                                  o contador de linhas de pixels.
                PUSH  HL          Guardar o endereço.
                LD    A,B         Guardar o número de linha em A.
0E4D CL-LINE-2  AND   +07         Descobrir quantos caracteres
                RRCA              existem em «B/8» linhas.
                RRCA              Passar o resultado para o
                RRCA              registo C (conterá +00 - 256 -
                LD    C,A         - para um «terço»).
                LD    A,B         Recuperar número de linha.
                LD    B,+00       Colocar no par de registos BC
                DEC   C           «um menos» do que o número de
                                  caracteres.
                LD    D,H         Fazer DE apontar para o
                LD    E,L         primeiro caracter.
                LD    (HL),+00    Limpar o byte-pixel do
                                  primeiro caracter.
                INC   DE          Fazer DE apontar para o 2.º
                LDIR              caracter e depois limpar
                                  os bytes-pixels de todos os
                                  outros caracteres.
                LD    DE,+0701    Para cada «terço» da imagem
                ADD   HL,DE       HL deve ser aumentado de
                                  +0701.
                DEC   A           Diminuir o número de linha.
                AND   +F8         Eliminar linhas extra e passar
                LD    B,A         a contagem de «terços» para B.
                JR    NZ,0E4D,CL-LINE-2  Saltar atrás se existem ainda
                                  «terços» para tratar.
```

Verificar se o ciclo foi executado oito vezes.

```
        POP     HL              Actualizar o endereço para
        INC     H               cada linha de pixels.
        POP     BC              Recuperar contadores.
        DEC     C               Diminuir o contador de linhas
        JR      NZ,0E4A,CL-LINE-1  de pixels a saltar para trás
                                a menos que tenha terminado.
```

Em seguida são definidos os bytes de atributos do modo requerido. O valor presente em ATTR-P será utilizado quando se tratar a parte principal do visor, e o valor de BORDCR quando se tratar a parte inferior.

```
        CALL    0E88,CL-ATTR    São determinados os endereços
                                do primeiro byte de atributos
                                e o número de bytes.
        LD      H,D             HL apontará para o primeiro
        LD      L,E             byte de atributos e DE para o
        INC     DE              segundo.
        LD      A,(ATTR-P)      Recuperar o valor em ATTR-P.
        BIT     0,(TV-FLAG)     Saltar para diante se se trata
        JR      Z,0E80,CL-LINE-3  a parte principal do visor.
        LD      A,(BORDCR)      Se não, usar BORDCR.
0E80 CL-LINE-3
        LD      (HL),A          Definir o byte de atributos.
        DEC     BC              Foi alterado um byte.
        LDIR                    Copiar agora o valor para
                                todos os bytes de atributos.
        POP     BC              Restaurar o número de linha.
        LD      C,+21           Passar o número de coluna para
        RET                     a da esquerda e retorno.
```

A subrotina «CL-ATTR»

Esta subrotina cumpre duas funções separadas:

1) A partir de um dado endereço do ficheiro de imagem, determina e coloca no par de registos DE o endereço dos atributos correspondentes. Note-se que o valor no início da subrotina aponta para a «nona» linha de um caracter.

2) A partir de um dado número de linha, colocado no registo B, determina e coloca no par de registos BC o número de áreas de caracter na imagem a partir do início dessa linha.

```
0E88 CL-ATTR
        LD      A,H             Recuperar o byte alto.
        RRCA                    Multiplicar este valor por
        RRCA                    trinta e dois.
        RRCA
        DEC     A               Voltar à linha «oito».
        OR      +50             Endereçar a área de atributos.
        LD      H,A             Restaurar o byte alto e passar
        EX      DE,HL           o endereço para DE.
        LD      H,C             Isto é sempre zero.
        LD      L,B             O número de linha.
        ADD     HL,HL           Multiplicar por 32.
        ADD     HL,HL
        ADD     HL,HL
        LD      B,H             Deslocar o resultado para o
        LD      C,L             par de registos BC antes do
        RET                     retorno.
```

A subrotina «CL-ADDR»

Para um dado número de linha, no registo B, é formado o endereço correspondente no ficheiro de imagem no par de registos HL.

```
0E9B CL-ADDR
        LD      A,+18           O número de linha deve ser
        SUB     B               invertido.
        LD      D,A             O resultado é guardado em D.
        RRCA                    De facto, «(A/8)*32».
        RRCA                    Num «terço» do visor
        RRCA                    o byte baixo para:
        AND     +E0             1.ª linha = +00,
                                2.ª linha = +20, etc.
        LD      L,A             O byte baixo vai para L.
        LD      A,D             Recupera o verdadeiro n.º de linha.
        AND     +18             De facto, «64+8*INT(A/8)».
        OR      +40             Para o «terço» superior do
                                visor o byte alto = +40,
                                para o médio é +48, e para
                                o inferior é +50.
        LD      H,A             O byte alto passa para H.
        RET                     Terminado.
```

A rotina do comando «COPY»

As cento e setenta e seis linhas de pixels do visor são tratadas uma a uma.

```
0EAC COPY
        DI                      Inibe a interrupção
                                mascarável durante COPY.
        LD      B,+B0           «176» linhas.
        LD      HL,+4000        Endereço base da imagem.
```

Entra-se agora no seguinte ciclo.

```
0EB2 COPY-1
        PUSH    HL              Guardar o endereço base e
        PUSH    BC              o número da linha.
        CALL    0EF4,COPY-LINE  É invocado «176» vezes.
        POP     BC              Recupera o n.º de linha e
        POP     HL              o endereço base.
        INC     H               O endereço base é actualizado
                                de «256» posições para cada
                                linha de pixels.
        LD      A,H             Saltar para diante e depois
        AND     +07             directamente para o ciclo para
        JR      NZ,0EC9,COPY-2  cada uma das 8 linhas de pixels
                                de uma linha de caracter.
```

Para cada nova linha de caracteres é necessário actualizar o endereço base.

```
                LD      A,L             Recuperar o byte baixo.
                ADD     A,+20           Actualizá-lo de +20 bytes.
                LD      L,A             A flag «carry» passa a zero
                                        dentro dos «terços» da imagem.
                CCF                     Alterar a flag «carry».
                SBC     A,A             O registo A conterá +F8 no
                AND     +F8             interior de um «terço», e +00
                                        quando se atinge outro «terço».
                ADD     A,H             É agora actualizado o byte
                LD      H,A             alto do endereço.
0EC9 COPY-2     DJNZ    0EB2,COPY-1     Saltar atrás até terem sido
                                        impressas «176» linhas.
                JR      0EDA,COPY-END   Saltar para diante para a
                                        rotina final.
```

A subrotina «COPY-BUFF»

Esta subrotina é invocada sempre que o conteúdo do buffer da impressora deva passar para esta.

```
0ECD COPY-BUFF  DI                      Inibir as interrupções mascaráveis.
                LD      HL,+5B00        Endereço base do buffer da
                                        impressora.
                LD      B,+08           Existem 8 linhas de pixels.
0ED3 COPY-3     PUSH    BC              Guardar o número de linha.
                CALL    0EF4,COPY-LINE  É invocada «8» vezes.
                POP     BC              Recuperar o nº de linha.
                DJNZ    0ED3,COPY-3     Saltar atrás até terem sido
                                        impressas «8» linhas.
```

Continuar para a rotina COPY-END.

```
0EDA COPY-END   LD      A,+04           Parar o motor da impressora.
                OUT     (+FB),A
                EI                      Activar as interrupções mascaráveis
                                        e continuar para CLEAR-PRB.
```

A subrotina «Limpar buffer da impressora»

O buffer da impressora é esvaziado invocando esta subrotina.

```
0EDF CLEAR-PRB  LD      HL,+5B00        Endereço base do buffer da impres-
                                        sora.
                LD      (PR-CC),L       Passar a 0 a «coluna» da impres-
                                        sora.
                XOR     A               Limpar o registo A.
                LD      B,A             Limpar também o registo B
                                        (contém de facto «256»).
0EE7 PRB-BYTES  LD      (HL),A          Os 256 bytes do buffer da
                INC     HL              impressora são limpos um
                DJNZ    0EE7,PRB-BYTES  de cada vez.
                RES     1,(FLAGS2)      Sinal «buffer esvaziado».
                LD      C,+21           Definir a posição de impressão
                JP      0DD9,CL-SET     e retorno por CL-SET e PO-STORE.
```


A subrotina «COPY-LINE»

Entra-se na subrotina contendo no par de registos HL o endereço base dos trinta e dois bytes que formam a linha de pixels, e no registo B o número da linha de pixels.

```
0EF4 COPY-LINE  LD      A,B             Copiar o nº da linha de pixels.
                CP      +03             O registo A conterá +00
                SBC     A,A             até serem tratadas as
                AND     +02             últimas duas linhas.
                OUT     (+FB),A         Travar o motor apenas para
                                        as duas últimas linhas.
                LD      D,A             O registo A conterá +00
                                        ou +02.
```

É necessário executar três testes antes de realizar qualquer impressão.

```
0EFD COPY-L-1   CALL    1F54,BREAK-KEY  Saltar para diante a menos que
                JR      C,0F0C,COPY-L-2 se carregue na tecla BREAK.
                LD      A,+04           Mas se se carrega;
                OUT     (+FB),A         parar o motor,
                EI                      activar interrupções,
                CALL    0EDF,CLEAR-PRB  limpar o buffer da impressora
                RST     0008,ERROR-1    e sair pela rotina de erro
                DEFB    +0C             «BREAK-CONT repeats».
0F0C COPY-L-2   IN      A,(+FB)         Recuperar o estado da
                ADD     A,A             impressora.
                RET     M               Retorno imediato se a
                                        impressora não está presente.
                JR      NC,0EFD,COPY-L-1 Esperar pela caneta.

                LD      C,+20           Existem 32 bytes.
```

Entra-se agora num ciclo que trata estes bytes.

```
0F14 COPY-L-3   LD      E,(HL)          Recuperar um byte.
                INC     HL              Actualizar o indicador.
                LD      B,+08           Oito bits por byte.
0F18 COPY-L-4   RL      D               Deslocar bits de D.
                RL      E               Passar cada bit para a «carry».
                RR      D               Deslocar novamente DF recolhendo
                                        a «carry» de E.
0F1E COPY-L-5   IN      A,(+FB)         Recuperar de novo o estado da
                RRA                     impressora e esperar pelo
                JR      NC,0F1E,COPY-L-5 sinal do codificador.
                LD      A,D             Continuar agora, e passar
                OUT     (+FB),A         o bit para a impressora.
                                        Nota: o bit 2 (baixo) arranca o
                                        motor, e o bit 1 (alto) trava-o;
                                        o bit 7 está ao valor um para
                                        realizar a impressão.
                DJNZ    0F18,COPY-L-4   «Imprimir» cada bit.
                DEC     C               Diminuir o contador de bytes.
                JR      NZ,0F14,COPY-L-3 Saltar atrás enquanto existem
                RET                     ainda bytes; senão, retorno.
```

As rotinas «EDITOR»

O editor é invocado em duas ocasiões:

1) Pela rotina principal de execução, permitindo ao utilizador introduzir uma linha Basic no sistema.

2) Pela rotina do comando INPUT.

Primeiramente, é guardado o «indicador do 'stack' de erro», e fornecido um endereço alternativo.

```
0F2C EDITOR     LD    HL,(ERR-SP)     O valor actual é guardado no
                PUSH  HL              «stack».
0F30 ED-AGAIN   LD    HL,+107F        Trata-se de ED-ERROR.
                PUSH  HL              Qualquer acontecimento que conduza
                LD    (ERR-SP),SP     à rotina de tratamento de erro
                                      voltará à posição
                                      ED-ERROR.
```

Entra-se agora num ciclo que trata cada tecla premida.

```
0F38 ED-LOOP    CALL  15D4,WAIT-KEY   Retorno, assim que é accionada
                                      uma tecla.
                PUSH  AF              Guardar o código temporariamente.
                LD    D,+00           Obter a duração do contacto
                LD    E,(PIP)         no teclado.
                LD    HL,+00C8        E o tom.
                CALL  03B5,BEEPER     Produzir o som da tecla.
                POP   AF              Recuperar o código.
                LD    HL,+0F38        Carregar no «stack» o
                PUSH  HL              endereço de ED-LOOP.
```

Analisar o código obtido.

```
                CP    +18             Aceitar todos os códigos de
                JR    NC,0F81,ADD-CHAR caracteres, gráficos e palavras.
                CP    +07             Aceitar também «,».
                JR    C,0F81,ADD-CHAR
                CP    +10             Saltar para diante se o código
                JR    C,0F92,ED-KEYS  representa uma tecla de «edit».
```

São agora consideradas as teclas de controlo — INK a TAB.

```
                LD    BC,+0002        INK e PAPER requerem duas
                                      posições.
                LD    D,A             Copiar o código para D.
                CP    +16             Saltar para diante para INK e
                JR    C,0F6C,ED-CONTR PAPER.
```

AT e TAB serão tratadas do seguinte modo:

```
                INC   BC              São necessárias três posições.
                BIT   7,(FLAGX)       Saltar para diante se não se
                JP    Z,101E,ED-IGNORE trata de INPUT LINE...
                CALL  15D4,WAIT-KEY   Obter o segundo código,
                LD    E,A             e colocá-lo em E.
```


São agora recuperados os outros bytes dos caracteres de controlo.

```
0F6C ED-CONTR   CALL  15D4,WAIT-KEY   Obter outro código.
                PUSH  DE              Guardar códigos anteriores.
                LD    HL,(K-CUR)      Recuperar K-CUR.
                RES   0,(MODE)        Sinal «modo K».
                CALL  1655,MAKE-ROOM  Abrir dois ou três espaços.
                POP   BC              Restaurar códigos anteriores.
                INC   HL              Apontar para a 1.ª posição.
                LD    (HL),B          Introduzir primeiro código.
                INC   HL              E agora o segundo, que será
                LD    (HL),C          desprezado se houver apenas
                                      dois códigos — por exemplo,
                                      para INK e PAPER.
                JR    0F8B,ADD-CH-1   Saltar para diante.
```

A subrotina «ADD-CHAR»

Esta subrotina acrescenta um código à linha actual de EDIT ou INPUT.

```
0F81 ADD-CHAR   RES   0,(MODE)        Sinal «modo K».
                LD    HL,(K-CUR)      Obter a posição do cursor.
                CALL  1652,ONE-SPACE  Abrir um espaço.
0F8B ADD-CH-1   LD    (DE),A          Introduzir o código no espaço
                INC   DE              e sinalizar que o cursor deve
                LD    (K-CUR),DE      ocorrer na posição seguinte.
                RET                   Retorno indirecto a ED-LOOP.
```

As teclas de «editing» são tratadas do seguinte modo:

```
0F92 ED-KEYS    LD    E,A             O código é transferido para
                LD    D,+00           o par de registos DE.
                LD    HL,+0F99        O endereço base da tabela das
                                      teclas de montagem.
                ADD   HL,DE           É endereçada a entrada, depois
                LD    E,(HL)          recolhida em E.
                ADD   HL,DE           É guardado no «stack» o endereço
                PUSH  HL              da rotina usada para o
                                      tratamento.
                LD    HL,(K-CUR)      Define-se o par de registos HL
                RET                   e executa-se um salto indirecto
                                      para a rotina requerida.
```

Tabela das «Teclas de Montagem»

Endereço	Deslocamento	Caracter	Endereço	Deslocamento	Caracter
0FA0	09	EDIT	0FA5	70	DELETE
0FA1	66	Cursor para a esquerda	0FA6	7E	ENTER
0FA2	6A	Cursor para a direita	0FA7	CF	SYMBOL SHIFT
0FA3	50	Cursor para baixo	0FA8	D4	GRAPHICS
0FA4	B5	Cursor para cima			

A subrotina «Tecla EDIT»

Quando se está em «modo montagem» *(editing)*, basta carregar na tecla EDIT para trazer a «Linha Basic actual» para a janela inferior do visor. No entanto, no modo INPUT, a acção da tecla EDIT é «limpar» a resposta actual e permitir uma nova entrada.

```
0FA9 ED-EDIT    LD      HL,(E-PPC)      Obter o número de linha actual.
                BIT     5,(FLAGX)       Mas saltar para diante se o modo
                JP      NZ,1097,CLEAR-SP é «INPUT».
                CALL    196E,LINE-ADDR  Determinar o endereço do
                CALL    1695,LINE-NO    início da linha actual, e
                                        portanto o seu número.
                LD      A,D             Se o número da linha assim
                OR      E               obtido é zero, limpar apenas
                JP      Z,1097,CLEAR-SP a área do «editing».
                PUSH    HL              Guardar o endereço da linha.
                INC     HL              Recolher em seguida o
                LD      C,(HL)          comprimento da linha.
                INC     HL
                LD      B,(HL)
                LD      HL,+000A        Somar +0A ao comprimento e veri-
                ADD     HL,BC           ficar se existe espaço suficiente
                LD      B,H             para uma cópia da linha.
                LD      C,L
                CALL    1F05,TEST-ROOM
                CALL    1097,CLEAR-SP   Limpar agora a área de «editing».
                LD      HL,(CURCHL)     Obter o endereço do canal actual
                EX      (SP),HL         e substitui-lo pelo endereço
                                        da linha.
                PUSH    HL              Guardá-lo temporariamente.
                LD      A,+FF           Abrir o canal «R» de tal modo que
                CALL    1601,CHAN-OPEN  a linha seja copiada para a
                                        área de montagem.
                POP     HL              Obter o endereço da linha.
                DEC     HL              Recuar para antes da linha.
                DEC     (E-PPC-lo)      Decrementar o número da linha
                                        actual a fim de evitar a
                                        impressão do cursor.
                CALL    1855,OUT-LINE   Imprimir a linha Basic.
                INC     (E-PPC-lo)      Incrementar o número de
                                        linha actual.
                                        Nota: A decrementação do
                                        número de linha nem sempre
                                        impede a impressão do
                                        cursor.
                LD      HL,(E-LINE)     Obter o início da linha
                INC     HL              na área de montagem e passar à
                INC     HL              frente do n.º de linha e do
                INC     HL              comprimento para determinar
                INC     HL              o endereço de K-CUR.
                LD      (K-CUR),HL
                POP     HL              Recuperar o endereço original
                CALL    1615,CHAN-FLAG  do canal e definir as flags
                RET                     apropriadas antes de voltar
                                        a ED-LOOP.
```


A subrotina «Cursor para baixo»

```
0FF3 ED-DOWN    BIT     5,(FLAGX)       Saltar para diante se o modo
                JR      NZ,1001,ED-STOP é «INPUT».
                LD      HL,+5C49        Isto é E-PPC.
                CALL    190F,LN-FETCH   Descobre o número de linha
                JR      106E,ED-LIST    seguinte, sendo produzida uma
                                        nova listagem automática.
1001 ED-STOP    LD      (ERR-NR),+10    Mensagem «STOP in INPUT».
                JR      1024,ED-ENTER   Saltar para diante.
```

A subrotina «Cursor para a esquerda»

```
1007 ED-LEFT    CALL    1031,ED-EDGE    O cursor é deslocado.
                JR      1011,ED-CUR     Saltar para diante.
```

A subrotina «Cursor para a direita»

```
100C ED-RIGHT   LD      A,(HL)          É testado o caracter actual,
                CP      +0D             e se for «retorno de linha»,
                RET     Z               retorno.
                INC     HL              Senão, obrigar o cursor a
                                        ocorrer a seguir ao caracter.
1011 ED-CUR     LD      (K-CUR),HL      Definir a variável de sistema
                RET                     K-CUR.
```

A subrotina «DELETE»

```
1015 ED-DELETE  CALL    1031,ED-EDGE    Deslocar o cursor para a esquerda.
                LD      BC,+0001        Reclamar o caracter
                JP      19E8,RECLAIM-2  actual.
```

A subrotina «ED-IGNORE»

```
101E ED-IGNORE  CALL    15D4,WAIT-KEY   São ignorados os dois
                CALL    15D4,WAIT-KEY   códigos seguintes da rotina de
                                        aceitação de teclas.
```

A subrotina «ENTER»

```
1024 ED-ENTER   POP     HL              São eliminados os endereços
                POP     HL              de ED-LOOP e ED-ERROR.
1026 ED-END     POP     HL              É restaurado o antigo valor de
                LD      (ERR-SP),HL     ERR-SP.
                BIT     7,(ERR-NR)      Retorno, se não há quaisquer
                RET     NZ              erros.
                LD      SP,HL           Senão, salto directo para a
                RET                     rotina de erro.
```

A subrotina «ED-EDGE»

O endereço do cursor encontra-se no par de registos HL, e será decrementado a menos que o cursor já se encontre no início da linha. Toma-se o cuidado de não colocar o cursor entre os caracteres de comando e os respectivos parâmetros.

```
1031 ED-EDGE     SCF
                 CALL  1195,SET-DE     DE guardará E-LINE (caso de
                                       montagem) ou WORKSP (caso de
                                       INPUT).
                 SBC   HL,DE           A flag «carry» passará a um
                                       se o cursor já se encontra no
                 ADD   HL,DE           início da linha.
                 INC   HL              Corrigir a subtracção.
                 POP   BC              Obter o endereço de retorno.
                 RET   C               Retorno através de ED-LOOP se
                                       a flag «carry» está a um.
                 PUSH  BC              Restaurar o endereço de retorno.
                 LD    B,H             Passar o endereço actual do
                 LD    C,L             cursor para BC.
```

Entrar agora num ciclo que verifica se os caracteres de controlo não são separados dos respectivos parâmetros.

```
103E ED-EDGE-1   LD    H,D             HL apontará para o
                 LD    L,E             caracter da linha à frente
                 INC   HL              do endereçado por DE.
                 LD    A,(DE)          Obter um código de caracter.
                 AND   +F0             Saltar para diante se o
                 CP    +10             código não representa INK a
                 JR    NZ,1051,ED-EDGE-2  TAB.
                 INC   HL              Ter em conta um parâmetro.
                 LD    A,(DE)          Obter novamente o código.
                 SUB   +17             A flag «carry» passa a 0 para TAB.
                 ADC   A,+00           Nota: Isto divide AT e TAB, mas
                                       como estas não são implementadas
                                       nesta forma, a divisão não tem
                                       qualquer relevância.
                 JR    NZ,1051,ED-EDGE-2  Saltar para diante a menos
                 INC   HL              que se trate AT e TAB que terão
                                       dois parâmetros possíveis.
1051 ED-EDGE-2   AND   A               Preparar para subtracção.
                 SBC   HL,BC           A flag «carry» passa a zero
                 ADD   HL,BC           quando o «indicador actualizado»
                                       atinge K-CUR.
                 EX    DE,HL           No ciclo seguinte, usar o
                 JR    C,103E,ED-EDGE-1  «indicador actualizado»,
                 RET                   mas se existir, usar o «indicador
                                       presente» para K-CUR.
                                       Nota: É o caracter de
                                       controlo que é eliminado
                                       ao usar DELETE.
```

A subrotina «Cursor para cima»

```
1059 ED-UP       BIT   5,(FLAGX)       Retorno se em modo «INPUT».
                 RET   NZ
```



```
                 LD    HL,(E-PPC)      Obter o número da linha
                 CALL  196E,LINE-ADDR  actual e o seu endereço inicial.
                 EX    DE,HL           HL aponta agora para a linha
                                       anterior.
                 CALL  1695,LINE-NO    Obtém-se o número desta linha.
                 LD    HL,+5C4A        Isto é E-PPC (alto).
                 CALL  191C,LN-STORE   Guarda o número de linha.
106E ED-LIST     CALL  1795,AUTO-LIST  Produz-se uma nova listagem
                 LD    A,+00           automática, e o canal «K» é
                 JP    1601,CHAN-OPEN  reaberto antes de voltar a
                                       ED-LOOP.
```

A subrotina «ED-SYMBOL»

Se se usam códigos SYMBOL e GRAPHICS devem ser tratados do seguinte modo:

```
1076 ED-SYMBOL   BIT   7,(FLAGX)       Saltar atrás a menos que
                 JR    Z,1024,ED-ENTER  trate INPUT... LINE.
107C ED-GRAPH    JP    0F81,ADD-CHAR   Saltar atrás.
```

A subrotina «ED-ERROR»

Vir aqui quando ocorreu algum tipo de erro.

```
107F ED-ERROR    BIT   4,(FLAGS2)      Saltar atrás se se usa um
                 JR    Z,1026,ED-END   canal diferente de «K».
                 LD    (ERR-NR),+FF    Eliminar o número do erro
                 LD    D,+00           e produzir som antes de
                 LD    E,(RASP)        percorrer de novo o editor.
                 LD    HL,+1A90
                 CALL  03B5,BEEPER
                 JP    0F30,ED-AGAIN
```

A subrotina «CLEAR-SP»

A área de montagem ou o espaço de trabalho são limpos.

```
1097 CLEAR-SP    PUSH  HL              Guardar o indicador da área.
                 CALL  1190,SET-HL     DE apontará para o primeiro
                                       caracter e HL para o último.
                 DEC   HL              Reclama a quantidade
                 CALL  19E5,RECLAIM-1  apropriada.
                 LD    (K-CUR),HL      As variáveis de sistema
                 LD    (MODE),+00      K-CUR e MODE («modo K»)
                 POP   HL              são inicializados antes
                 RET                   de recuperar o indicador.
```

A subrotina «Entrada por teclado»

Esta importante subrotina produz o código da última tecla premida, mas note-se que CAPS LOCK, a mudança de modo e os parâmetros dos controlos de cor são tratados no interior de subrotina.

```
10A8 KEY-INPUT  BIT   3,(TV-FLAG)        Copiar a linha de montagem
                CALL  NZ,111D,ED-COPY    ou de INPUT para o visor se
                                         o modo se alterou.
                AND   A                  Retorno com as flags «carry»
                BIT   5,(FLAGS)          e «zero» a zero se não foi
                RET   Z                  premida qualquer tecla.
                LD    A,(LAST-K)         Senão, recolher o código
                RES   5,(FLAGS)          e sinalizar o facto.
                PUSH  AF                 Guardar o código temporariamente.
                BIT   5,(TV-FLAG)        Limpar a parte inferior do visor
                CALL  NZ,0D6E,CLS-LOWER  se necessário;
                                         por exemplo, após «scroll?».
                POP   AF                 Recuperar o código.
                CP    +20                Aceitar todos os caracteres e
                JR    NC,111B,KEY-DONE   palavras-chave.
                CP    +10                Saltar para diante com a maior
                JR    NC,10FA,KEY-CONTR  parte dos caracteres de controlo
                CP    +06                Saltar para diante com os
                JR    NC,10DB,KEY-M&CL   códigos de «modo» e de CAPS LOCK.
```

Tratar agora os códigos FLASH, BRIGHT e INVERSE.

```
                LD    B,A                Guardar o código.
                AND   +01                Manter apenas o bit zero.
                LD    C,A                C contém +00 (=OFF) ou
                                         +01 (=ON).
                LD    A,B                Recuperar o código.
                RRA                      Rodá-lo uma vez (perde bit 0).
                ADD   A,+12              Aumentá-lo de +12 dando para
                JR    1105,KEY-DATA      FLASH +12,, BRIGHT +13 e
                                         INVERSE +14.
```

Os códigos de CAPS LOCK e de modo são tratados «localmente».

```
10DB KEY-M&CL   JR    NZ,10E6,KEY-MODE   Saltar para diante (códigos «modo»).
                LD    HL,+5C6A           Isto é FLAGS2.
                LD    A,+08              Actuar no bit 3 de FLAGS2. É
                XOR   (HL)               a flag de CAPS LOCK.
                LD    (HL),A
                JR    10F4,KEY-FLAG      Saltar para diante.
10E6 KEY-MODE   CP    +0E                Verificar o limite inferior.
                RET   C
                SUB   +0D                Reduzir a gama.
                LD    HL,+5C41           Isto é MODE.
                CP    (HL)               Foi alterado?
                LD    (HL),A             Indicar o novo código «modo».
                JR    NZ,10F4,KEY-FLAG   Saltar se mudou;
                LD    (HL),+00           senão, usar «modo L».
10F4 KEY-FLAG   SET   3,(TV-FLAG)        Sinal «talvez o modo
                                         tenha sido alterado».
                CP    A                  Passar a zero a flag «carry»
                RET                      e retorno.
```

São tratados os códigos das teclas de controlo (excepto FLASH, BRIGHT e INVERSE).

```
10FA KEY-CONTR  LD    B,A                Guardar o código.
                AND   +07                Colocar no registo C o
                LD    C,A                parâmetro (+00 a +07).
```



```
                LD    A,+10              Código INK em A.
                BIT   3,B                Mas se o código é de uma
                JR    NZ,1105,KEY-DATA   tecla não «shifted», colocar
                INC   A                  em A o código de PAPER.
```

O parâmetro é guardado em K-DATA e o endereço do canal mudado de KEY-INPUT para KEY-NEXT.

```
1105 KEY-DATA   LD    (K-DATA),C         Guardar o parâmetro.
                LD    DE,+110D           Isto é KEY-NEXT.
                JR    1113,KEY-CHAN      Saltar para diante.
```

Nota: Na primeira passagem, entrando em KEY-INPUT, o registo A volta com um «código de controlo»; e na passagem seguinte, entrando em KEY--NEXT, com o parâmetro.

```
110D KEY-NEXT   LD    A,(K-DATA)         Recuperar o parâmetro.
                LD    DE,+10A8           Isto é KEY-INPUT.
```

Definir agora o endereço de entrada na primeira área do canal.

```
1113 KEY-CHAN   LD    HL,(CHANS)         Recuperar o endereço do canal.
                INC   HL
                INC   HL
                LD    (HL),E             Definir agora o endereço
                INC   HL                 de entrada.
                LD    (HL),D
```

Finalmente, sair com o código requerido no registo A.

```
111B KEY-DONE   SCF                      Mostrar que foi encontrado um
                RET                      código e retorno.
```

A subrotina «Cópia da parte inferior do visor»

Esta subrotina é invocada sempre que se pretende imprimir na parte inferior do visor a linha presente na área de «editing» ou de INPUT.

```
111D ED-COPY    CALL  0D4D,TEMPS         Usar as cores permanentes.
                RES   3,(TV-FLAG)        Sinalizar que o «modo deve ser
                RES   5,(TV-FLAG)        considerado inalterado» e a
                                         «janela inferior não necessita
                                         de ser limpa».
                LD    HL,(S-POSNL)       Guardar o valor actual de
                PUSH  HL                 S-POSNL.
                LD    HL,(ERR-SP)        Manter o valor actual de
                PUSH  HL                 ERR-SP.
                LD    HL,+1167           Isto é ED-FULL.
                PUSH  HL                 Colocar este endereço no «stack»
                LD    (ERR-SP),SP        de modo a tornar ED-FULL
                                         ponto de entrada em caso de
                                         erro.
                LD    HL,(ECHO-E)        Passar o valor de ECHO-E para
                PUSH  HL                 o «stack».
                SCF                      Levar HL a apontar para o início
                CALL  1195,SET-HL        do espaço e DE para o fim.
                EX    DE,HL
```

```
                CALL    187D,OUT-LINE2      Imprimir agora a linha.
                EX      DE,HL               Trocar os indicadores e
                CALL    18E1,OUT-CURS       imprimir o cursor.
                LD      HL,(S-POSNL)        Depois obter o valor actual
                EX      (SP),HL             de S-POSNL e trocá-lo com
                                            ECHO-E.
                EX      DE,HL               Passar ECHO-E a DE.
                CALL    0D4D,TEMPS          Obter novamente as cores
                                            permanentes.
```

O resto de qualquer linha que tenha sido iniciada é agora terminado com espaços impressos com a cor de PAPER «permanente».

```
1150 ED-BLANK   LD      A,(S-POSNL-hi)      Recuperar o n.º de linha actual
                SUB     D                   e subtrair o n.º de linha antigo.
                JR      C,117C,ED-C-DONE    Saltar para diante se não é
                                            necessário «limpar» linhas.
                JR      NZ,115E,ED-SPACES   Saltar para diante se não se
                                            está na mesma linha.
                LD      A,E                 Obter o número de coluna
                SUB     (S-POSNL-lo)        antigo e subtrair o novo
                                            número de coluna.
                JR      NC,117C,ED-C-DONE   Saltar se não precisa de espaços.
115E ED-SPACES  LD      A,+20               Um «espaço».
                PUSH    DE                  Guardar os valores antigos.
                CALL    09F4,PRINT-OUT      Imprimir.
                POP     DE                  Recuperar valores antigos.
                JR      1150,ED-BLANK       Para trás de novo.
```

Tratar agora quaisquer erros.

```
1167 ED-FULL    LD      D,+00               Produzir um som (-rasp-).
                LD      E,(RASP)
                LD      HL,+1A90
                CALL    03B5,BEEPER
                LD      (ERR-NR),+FF        Anular o número do erro.
                LD      DE,(S-POSNL)        Recuperar o valor actual de
                JR      117E,ED-C-END       S-POSNL e saltar para diante.
```

Saída normal após execução da cópia da linha de «edit» ou INPUT.

```
117C ED-C-DONE  POP     DE                  O novo valor de posição.
                POP     HL                  O «endereço de erro».
```

Mas vir aqui se houver um erro.

```
117E ED-C-END   POP     HL                  Restaurar o valor antigo
                LD      (ERR-SP),HL         de ERR-SP.
                POP     BC                  Obter o valor antigo de
                                            S-POSNL.
                PUSH    DE                  Guardar os novos valores de
                                            posição.
                CALL    0DD9,CL-SET         Definir as variáveis de sistema.
                POP     HL                  O valor antigo de S-POSNL
                LD      (ECHO-E),HL         passa para ECHO-E.
                LD      (X-PTR-hi),+00      X-PTR é limpa de modo
                RET                         adequado, e retorno.
```


As subrotinas «SET-HL» e «SET-DE»

Estas subrotinas terminam com HL apontando para a primeira posição e DE para a «última» posição, tanto da área de montagem como da área de trabalho.

```
1190 SET-HL     LD      HL,(WORKSP)         Apontar para última posição
                DEC     HL                  da área de montagem.
                AND     A                   Limpar flag «carry».
1195 SET-DE     LD      DE,(E-LINE)         Apontar para o início da
                BIT     5,(FLAGX)           área de montagem, e retorno
                RET     Z                   se o modo é «editing».
                LD      DE,(WORKSP)         Senão, alterar DE.
                RET     C                   Retorno se for pretendido.
                LD      HL,(STKBOT)         Recuperar STKBOT e retorno
                RET                         em seguida.
```

A subrotina «REMOVE-FP»

Esta subrotina elimina os formatos de vírgula flutuante escondidos nas linhas Basic.

```
11A7 REMOVE-FP  LD      A,(HL)              Cada caracter é examinado
                                            em separado.
                CP      +0E                 É um separador de número?
                LD      BC,+0006            Ocupará seis posições.
                CALL    Z,19E8,RECLAIM-2    Reclamar o número F.P.
                LD      A,(HL)              Recuperar novamente o código.
                INC     HL                  Actualizar o indicador.
                CP      +0D                 Retorno de linha?
                JR      NZ,11A7,REMOVE-FP   Para trás, se não. Retorno
                RET                         simples, se sim.
```

6
AS ROTINAS DE EXECUÇÃO

A rotina «Inicialização»

O principal ponto de entrada desta rotina é START/NEW (11CB). Quando se entra por START (0000), como acontece quando se liga o sistema, o registo A contém zero e o par de registos DE o valor +FFFF. No entanto, o principal ponto de entrada pode também ser atingido após a execução da rotina de comando NEW.

A rotina de comando «NEW»

```
11B7 NEW        DI                      Inibe interrupções mascaráveis.
                LD      A,+FF           Flag NEW.
                LD      DE,(RAMTOP)     O valor existente de RAMTOP é
                                        preservado.
                EXX                     Carregar os registos alternativos
                LD      BC,(P-RAMT)     com as seguintes variáveis de
                LD      DE,(RASP/PIP)   sistema. Todas serão assim
                LD      HL,(UDG)        preservadas.
                EXX
```

Principal ponto de entrada.

```
11CB START/NEW  LD      B,A             Salvaguardar a flag.
                LD      A,+07           Passar a margem (BORDER)
                OUT     (+FE),A         a branco.
                LD      A,+3F           Colocar no registo I o
                LD      I,A             valor +3F.
                DEFB    +00,+00,+00     Esperar 24 estados T.
                DEFB    +00,+00,+00
```

Verificar agora a memória.

```
11DA RAM-CHECK  LD      H,D             Transferir o valor em DE
                LD      L,E             (START = +FFFF, NEW =
                                        RAMTOP).
11DC RAM-FILL   LD      (HL),+02        Introduzir +02 em todas as
                DEC     HL              posições acima de +3FFF.
                CP      H
                JR      NZ,11DC,RAM-FILL
11E2 RAM-READ   AND     A               Preparar para subtracção.
                SBC     HL,DE           A flag «carry» ficará a zero
                ADD     HL,DE           quando for atingido o topo.
```



```
                INC     HL              Actualiza o indicador.
                JR      NC,11EF,RAM-DONE Saltar quando atinge o topo.
                DEC     (HL)            +02 passa a +01.
                JR      Z,11EF,RAM-DONE Mas se zero, RAM deficiente.
                                        Usar HL actual como topo.
                DEC     (HL)            +01 passa a +00.
                JR      Z,11E2,RAM-READ Passar ao teste seguinte a
                                        menos que fracasse.
11EF RAM-DONE   DEC     HL              HL aponta para a última posição
                                        em bom estado.
```

Em seguida, restaurar as variáveis de sistema «preservadas» (sem sentido quando se entrou por START).

```
                EXX                     Comutar os registos.
                LD      (P-RAMT),BC     Restaurar P-RAMT, RASP/PIP e
                LD      (RASP/PIP),DE,  UDG.
                LD      (UDG),HL
                EXX
                INC     B               Verificar a flag START/NEW.
                JR      Z,1219,RAM-SET  Saltar para diante se se vem
                                        da rotina de comando NEW.
```

Reescrever as variáveis de sistema quando se vem de START e inicializar a área de gráficos definidos pelo utilizador.

```
                LD      (P-RAMT),HL     Topo da RAM física.
                LD      DE,+3EAF        Último byte do «U» do conjunto
                                        de caracteres.
                LD      BC,+00A8        Existe este número de bytes em
                                        vinte e uma letras.
                EX      DE,HL           Comutar os indicadores.
                LDDR                    Copiar as formas de caracter
                                        das letras «A» a «U».
                EX      DE,HL           Comutar de novo os indicadores.
                INC     HL              Apontar para o primeiro byte.
                LD      (UDG),HL        Definir os UDG.
                DEC     HL              Diminuir uma posição.
                LD      BC,+0040        Definir as variáveis de sistema
                LD      (RASP/PIP),BC   RASP e PIP.
```

O resto da rotina é comum às operações START e NEW.

```
1219 RAM-SET    LD      (RAMTOP),HL     Definir RAMTOP.
                LD      HL,+3C00        Inicializar a variável de
                LD      (CHARS),HL      sistema CHARS.
```

Em seguida, define-se o «stack»-máquina.

```
                LD      HL,(RAMTOP)     A posição de topo recebe o
                LD      (HL),+3E        valor +3E.
                DEC     HL              A posição seguinte é deixada
                                        em zero.
                LD      SP,HL           Estas duas posições representam
                                        a «última entrada».
```

```
DEC   HL          Descer duas posições para
DEC   HL          descobrir o valor correcto
LD    (ERR-SP),HL de ERR-SP.
```

A rotina de inicialização continua assim:

```
IM    1           Usa-se o modo de interrupção 1.
LD    IY,+5C3A    IY contém sempre +ERR-NR.
EI                Pode activar-se agora as
                  interrupções. O relógio de
                  tempo real é actualizado e o
                  teclado será varrido em cada
                  1/50 de segundo.
LD    HL,+5CB6    Endereço base da área de
LD    (CHANS),HL  informação de canal.
LD    DE,15AF     Os dados de canal iniciais
LD    BC,+0015    são deslocados da tabela
EX    DE,HL       (15AF) para a área de
LDIR              informação de canal.
EX    DE,HL       A variável de sistema DATADD
DEC   HL          é feita apontar para a última
LD    (DATADD),HL posição dos dados de canal.
INC   HL          E PROG e VARS para a
LD    (PROG),HL   posição seguinte.
LD    (VARS),HL
LD    (HL),+80    Separador da área de
                  variáveis.
INC   HL          Passar uma posição para definir
LD    (E-LINE),HL o valor de E-LINE.
LF    (HL),+0D    Transformar a «edit-line» num
INC   HL          único caracter de «retorno de linha».
LD    (HL),+80    Inserir agora um separador final.
INC   HL          Passar uma posição para definir
LD    (WORKSP),HL o valor de WORKSP, STKBOT
LD    (STKBOT),HL e STKEND.
LD    (STKEND),HL
LD    A,+38       Inicializar as variáveis de
LD    (ATTR-P),A  sistema de cor para: FLASH 0,
LD    (ATTR-T),A  BRIGHT 0, PAPER 7 e INK 0.
LD    (BORDCR),A
LD    HL,+0523    Inicializar nas variáveis de
LD    (REPDEL),HL sistema REPDEL e REPPER.
DEC   (KSTATE-0)  Colocar +FF em KSTATE-0.
DEC   (KSTATE-4)  Colocar +FF em KSTATE-4.
LD    HL,+15C6    Deslocar os dados iniciais de
LD    DE,+5C10    «stream» da sua tabela para a
LD    BC,+000E    área de «streams».
LDIR
SET   1,(FLAGS)   Sinal «impressora em uso», e
CALL  0EDF,CLEAR-PRB limpeza do buffer da impressora.
LD    (DF-SZ),+02 Definir o tamanho da janela
CALL  0D6B,CLS    inferior da imagem, e limpar
                  toda esta.
XOR   A           Imprimir agora a mensagem
LD    DE,+1538    «© Sinclair Research Ltd»
CALL  0C0A,PO-MSG na linha inferior.
SET   5,(TV-FLAG) Sinalizar «a parte inferior
                  deve ser limpa».
JR    12A9,MAIN-1 Saltar para diante, para o
                  ciclo executivo principal.
```


O ciclo «executivo principal»

Este ciclo principal estende-se da posição 12A2 até à posição 15AE e controla o modo «editing», a execução de ordens directas e a produção de mensagens.

```
12A2 MAIN-EXEC LD    (DF-SZ),+02      A parte inferior do visor
                                      terá uma dimensão de 2 linhas.
               CALL  1795,AUTO-LIST   Produz uma listagem automática.
12A9 MAIN-1    CALL  16B0,SET-MIN     Todas as áreas após E-LINE
                                      recebem as suas configurações
                                      mínimas.
12AC MAIN-2    LD    A,+00            O canal «K» é aberto antes de
               CALL  1601,CHAN-OPEN   invocar o EDITOR.
               CALL  0F2C,EDITOR      O EDITOR é invocado para deixar
                                      o utilizador construir a linha Basic.
               CALL  1B17,LINE-SCAN   Verifica a sintaxe da linha
                                      actual.
               BIT   7,(ERR-NR)       Salta para diante se a sintaxe
               JR    NZ,12CF,MAIN-3   está correcta.
               BIT   4,(FLAGS2)       Salta para diante se está a ser
               JR    Z,1303,MAIN-4    usado um canal diferente de «K».
               LD    HL,(E-LINE)      Aponta para o início da linha
                                      com o erro.
               CALL  11A7,REMOVE-FP   Elimina as formas em vírgula
                                      flutuante desta linha.
               LD    (ERR-NR),+FF     Passa ERR-NR a zero e salta atrás
               JR    12AC,MAIN-2      para MAIN-2 deixando a listagem
                                      sem alterações.
```

A «edit-line» não acusou erros de sintaxe, sendo agora necessário distinguir entre os três tipos de linha possíveis.

```
12CF MAIN-3    LD    HL,(E-LINE)      Apontar para o início da linha.
               LD    (CH-ADD),HL      Definir também CH-ADD para o início.
               CALL  19FB,E-LINE-NO   Recuperar o número de linha para
                                      BC.
               LD    A,B              O número de linha é
               OR    C                válido?
               JR    NZ,155D,MAIN-ADD Saltar, se sim, e acrescentar
                                      a nova linha ao programa.
               RST   0018             Obter o primeiro caracter da
               CP    +0D              linha e verificar se esta é
                                      «apenas retorno de linha».
               JR    Z,12A2,MAIN-EXEC Se for, saltar atrás.
```

A «edit-line» deve começar por uma ordem Basic directa, sendo esta a primeira a ser interpretada.

```
BIT   0,(FLAGS2)   Limpar toda a imagem a menos
CALL  NZ,0DAF,CL-ALL que a flag afirme ser desnecessário.
CALL  0D6E,CLS-LOWER Limpar sempre a parte inferior.
LD    A,+19        Definir o valor apropriado
SUB   (S-POSN-hi)  do contador de «scroll».
LD    (SCR-CT),A
SET   7,(FLAGS)    Sinalizar «execução de linha».
LD    (ERR-NR),+FF Assegurar que ERR-NR está
                   correcto.
```

```
                LD      (NSPPC),+01     Tratar a primeira instrução da
                                        linha.
                CALL    1B8A,PROG-RUN   Agora interpretar a linha.
                                        Nota: O endereço 1303 passa para
                                        o «stack» da máquina e é endereça-
                                        do por ERR-SP.
```

Depois de a linha ter sido interpretada e de terem sido realizadas todas as acções dela decorrentes, é feito um retorno a MAIN-4, permitindo a impressão de uma mensagem.

```
1303 MAIN-4     HALT                    Deve activar-se a interrupção
                                        mascarável.
                RES     5,(FLAGS)       Sinal «pronto para nova tecla».
                BIT     1,(FLAGS2)      Esvaziar o buffer da impressora
                CALL    NZ,0ECD,COPY-BUFF se foi usado.
                LD      A,(ERR-NR)      Recuperar o número de erro e
                INC     A               incrementá-lo.
1313 MAIN-G     PUSH    AF              Guardar o novo valor.
                LD      HL,+0000        As variáveis de sistema FLAGX,
                LD      (FLAGX),H       X-PTR (alto) e DEFADD são
                LD      (X-PTR-hi),H    passadas a zero.
                LD      (DEFADD),HL
                LD      HL,+0001        Garantir que o «stream» +00
                LD      (STRMS-6),HL    aponta para o canal «K».
                CALL    16B0,SET-MIN    Limpar as áreas de trabalho e
                                        o «stack» do calculador.
                RES     5,(FLAGX)       Sinal «modo editing».
                CALL    0D6E,CLS-LOWER  Limpar janela inferior.
                SET     5,(TV-FLAG)     Sinal «janela inferior exige
                                        ser limpa».
                POP     AF              Obter o valor da mensagem.
                LD      B,A             Fazer uma cópia em B.
                CP      +0A             Saltar para diante com n.os de
                JR      C,133C,MAIN-5   mensagem «0 a 9».
                ADD     A,+07           Somar o deslocamento da letra
                                        ASCII.
133C MAIN-5     CALL    15EF,OUT-CODE   Imprimir o código da mensagem
                LD      A,+20           e um espaço a seguir.
                RST     0010,PRINT-A-1
                LD      A,B             Recuperar o código e usá-lo
                LD      DE,+1391        para identificar a mensagem
                CALL    0C0A,PO-MSG     requerida.
                XOR     A               Imprimir a mensagem, uma
                LD      DE,+1536        vírgula e um espaço.
                CALL    0C0A,PO-MSG
                LD      BC,(PPC)        Obter agora o n.º de linha
                CALL    1A1B,OUT-NUM1   actual e imprimi-lo também.
                LD      A,+3A           Imprimir «:» em seguida.
                RST     0010,PRINT-A-1
                LD      C,(SUBPPC)      Obter o n.º da instrução
                LD      B,+00           actual para o registo BC e
                CALL    1A1B,OUT-NUM1   imprimi-lo.
                CALL    1097,CLEAR-SP   Limpar a área de «editing».
                LD      A,(ERR-NR)      Obter de novo o n.º de erro.
                INC     A               Incrementá-lo como habitual.
                JR      Z,1386,MAIN-9   Se o programa foi terminado
                                        com êxito não pode haver
                                        «CONTinuação», portanto saltar.
```



```
                CP      +09             Se o programa parou com «STOP
                JR      Z,1373,MAIN-6   statement» ou «BREAK into
                CP      +15             program», a CONTinuaçao será a
                JR      NZ,1376,MAIN-7  partir da instrução seguinte;
1373 MAIN-6     INC     (SUBPPC)        se não, SUBPPC não se altera.
1376 MAIN-7     LD      BC,+0003        As variáveis de sistema OLDPPC
                LD      DE,+5C70        e OSPCC devem agora conter
                                        os n.os de linha e de instrução
                                        para CONTinuação.
                LD      HL,+5C44        Os valores usados serão os de
                BIT     7,(NSPPC)       PPC e SUBPPC a menos que
                JR      Z,1384,MAIN-8   NSPPC indique que ocorreu «break»
                ADD     HL,BC           antes de um salto
1384 MAIN-8     LDDR                    (isto é, depois de uma
                                        instrução GO TO, etc.).
1386 MAIN-9     LD      (NSPPC),+FF     NSPPC passa a zero para
                                        indicar «sem salto».
                RES     3,(FLAGS)       É seleccionado o modo «K».
                JP      12AC,MAIN-2     Finalmente, é feito o salto
                                        atrás, mas só aparecerá a
                                        listagem do programa se for
                                        pedido.
```

As mensagens de erro

Cada mensagem é dada com o último caracter invertido (+80 hex.).

```
1391  DEFB +80          — é passado o byte inicial.
1392  Mensagem 0        — «OK»
1394  Mensagem 1        — «NEXT without FOR»
13A4  Mensagem 2        — «Variable not found»
13B6  Mensagem 3        — «Subscript wrong»
13C6  Mensagem 4        — «Out of memory»
13D2  Mensagem 5        — «Out of screen»
13DF  Mensagem 6        — «Number too big»
13ED  Mensagem 7        — «RETURN without GOSUB»
1401  Mensagem 8        — «End of file»
140C  Mensagem 9        — «STOP statement»
141A  Mensagem A        — «Invalid argument»
142A  Mensagem B        — «Integer out of range»
143E  Mensagem C        — «Nonsense in Basic»
144F  Mensagem D        — «BREAK - CONT repeats»
1463  Mensagem E        — «Out of DATA»
146E  Mensagem F        — «Invalid file name»
147F  Mensagem G        — «No room for line»
148F  Mensagem H        — «STOP in INPUT»
149C  Mensagem I        — «FOR without NEXT»
14AC  Mensagem J        — «Invalid I/O device»
14BE  Mensagem K        — «Invalid colour»
14CC  Mensagem L        — «BREAK into program»
14DE  Mensagem M        — «RAMTOP no good»
14EC  Mensagem N        — «Statement lost»
14FA  Mensagem O        — «Invalid stream»
1508  Mensagem P        — «FN without DEF»
```

1516 Mensagem Q — «Parameter error»
1525 Mensagem R — «Tape loading error»

Existem ainda as duas seguintes mensagens:
1537 — «, » Uma vírgula e um espaço
1539 — « © 1982 Sinclair Research Ltd»

Mensagem «G — No room for line»

```
1555 REPORT-G   LD    A,+10          «G» possui o código «10+07+30».
                LD    BC,+0000       Limpar BC.
                JP    1313,MAIN-G    Saltar atrás para produzir mensagem.
```

A subrotina «MAIN-ADD»

Esta subrotina permite o acrescento de uma nova linha Basic ao programa Basic existente na área de programas. Se uma linha possui simultaneamente uma versão antiga e uma versão nova, é «reclamada» e substituída a antiga. Uma linha nova que consiste apenas num número de linha não passa para a área de programa.

```
155D MAIN-ADD   LD    (E-PPC),BC     Passar o novo número de linha
                                     a linha actual.
                LD    HL,(CH-ADD)    Obter CH-ADD e guardar o
                EX    DE,HL          endereço em DE.
                LD    HL,+1555       Passar o endereço de REPORT-G
                PUSH  HL             para o «stack» da máquina.
                                     ERR-SP apontará
                                     para REPORT-G.
                LD    HL,(WORKSP)    Recuperar WORKSP.
                SCF                  Determinar o comprimento de
                SBC,  HL,DE          linha desde o byte seguinte
                                     ao n.º de linha até ao caracter
                                     «mudança de linha» inclusive.
                PUSH  HL             Guardar o comprimento.
                LD    H,B            Passar o número de linha para
                LD    L,C            o par de registos HL.
                CALL  196E,LINE-ADDR Existe alguma linha com este
                                     número?
                JR    NZ,157D,MAIN-ADD1 Saltar, se não.
                CALL  1988,NEXT-ONE  Descobrir o comprimento da
                CALL  19E8,RECLAIM-2 linha «antiga» e reclamá-la.
157D MAIN-ADD1  POP   BC             Obter o comprimento da linha
                LD    A,C            «nova» e saltar para diante
                DEC   A              se for apenas um «número de
                OR    B              linha e um retorno de caracter».
                JR    15AB,MAIN-ADD2
                PUSH  BC             Guardar o comprimento.
                INC   BC             Serão necessárias quatro
                INC   BC             posições extra, isto é,
                INC   BC             duas para os números e duas
                INC   BC             para o comprimento.
                DEC   HL             Levar HL a apontar para a
                                     posição anterior a «destino».
                LD    DE,(PROG)      Guardar o valor actual de
                PUSH  DE             PROG de modo a evitar corrupção
                                     ao acrescentar uma 1.ª linha.
```



```
                CALL  1655,MAKE-ROOM Cria espaço para nova linha.
                POP   HL             O valor antigo de PROG é
                LD    (PROG),HL      aqui obtido e restaurado.
                POP   BC             É feita uma cópia do comprimento
                PUSH  BC             da linha (sem parâmetros).
                INC   DE             Faz-se DE apontar para a
                                     última posição da nova área
                LD    HL,(WORKSP)    e HL para o «retorno de linha»
                DEC   HL             da nova linha na área de
                DEC   HL             «editing».
                LDDR                 Copia a linha.
                LD    HL,(E-PPC)     Obtém o número de linha.
                EX    DE,HL          «Destino» para HL e
                                     número para DE.
                POP   BC             Obtém o comprimento de nova
                                     linha.
                LD    (HL),B         Byte alto do comprimento.
                DEC   HL
                LD    (HL),C         Byte baixo do comprimento.
                DEC   HL
                LD    (HL),E         Byte baixo do n.º de linha.
                DEC   HL
                LD    (HL),D         Byte alto do n.º de linha.
15AB MAIN-ADD2  POP   AF             Endereço de REPORT-G.
                JP    12A2,MAIN-EXEC Salta atrás, desta vez para
                                     produzir uma listagem automática.
```

A «Informação inicial de canal»

Inicialmente, existem quatro canais — «K», «S», «R» e «P» — para comunicação com o teclado, o visor, a área de trabalho e a impressora.

Para cada canal, o endereço da rotina de saída ocorre antes do endereço da rotina de entrada e do código de canal.

```
15AF  DEFB  F4 09  —  PRINT-OUT
      DEFB  A8 10  —  KEY-INPUT
      DEFB  4B     —  «K»
15B4  DEFB  F4 09  —  PRINT-OUT
      DEFB  C4 15  —  REPORT-J
      DEFB  53     —  «S»
15B9  DEFB  81 0F  —  ADD-CHAR
      DEFB  C4 15  —  REPORT-J
      DEFB  52     —  «R»
15BE  DEFB  F4 09  —  PRINT-OUT
      DEFB  C4 15  —  REPORT-J
      DEFB  50     —  «P»
15C3  DEFB  80     —  Separador final.
```

Mensagem «J — Invalid I/O device»

```
15C4 REPORT-J   RST   0008,ERROR-1  Invoca a rotina de tratamento
                DEFB  +12           de erro.
```

Os «Dados iniciais de STREAM»

Inicialmente existem sete streams: +FD a +03.

```
15C6 DEFB  01 00 —  stream +FD conduz a canal «K»
15C8 DEFB  06 00 —  stream +FE conduz a canal «S»
15CA DEFB  0B 00 —  stream +FF conduz a canal «R»
15CC DEFB  01 00 —  stream +00 conduz a canal «K»
15CE DEFB  01 00 —  stream +01 conduz a canal «K»
15D0 DEFB  06 00 —  stream +02 conduz a canal «S»
15D2 DEFB  10 00 —  stream +03 conduz a canal «P»
```

A subrotina «WAIT-KEY»

Esta subrotina é a subrotina de controlo que invoca a subrotina de entrada.

```
15D4 WAIT-KEY   BIT  5,(TV-FLAG)          Saltar para diante se a flag
                JR   NZ,15DE,WAIT-KEY1    indica que a janela inferior
                                          não necessita de ser limpa.
                SET  3,(TV-FLAG)          Senão, sinalizar «considerar
                                          que o modo se alterou».
15DE WAIT-KEY1  CALL 15E6,INPUT-AD        Chama a subrotina de entrada
                                          indirectamente por INPUT-AD.
                RET  C                    Retorno com códigos aceitáveis.
                JR   Z,15DE,WAIT-KEY1     Tanto a flag «carry» como a «zero»
                                          passam a zero se «não está a ser
                                          premida qualquer tecla»; senão,
                                          sinaliza erro.
```

Mensagem «8 — End of file».

```
15E4 REPORT-8   RST  0008,ERROR-1         Invocar a rotina de tratamento
                DEFB +07                  de erro.
```

A subrotina «INPUT-AD»

Os registos são salvaguardados, e HL aponta para o endereço de entrada.

```
15E6 INPUT-AD   EXX                       Guardar registos.
                PUSH HL
                LD   HL,(CURCHL)          Obter o endereço base da
                                          informação do canal actual.
                INC  HL                   Ultrapassar o endereço de saída.
                INC  HL
                JR   15F7,CALL-SUB        Saltar adiante.
```

A subrotina «Principal de impressão»

Esta subrotina é invocada com um valor absoluto ou um código válido de caracter no registo A.

```
15EF OUT-CODE   LD   E,+30                Aumentar o valor no registo
                ADD  A,E                  A de +30.
15F2 PRINT-A-2  EXX                       Guardar de novo os registos.
                PUSH HL
                LD   HL,(CURCHL)          Obter o endereço base do
                                          canal actual. Este apontará
                                          para um endereço de saída.
```


Chamar agora a subrotina propriamente dita. HL aponta para o endereço de saída ou entrada, conforme for apropriado.

```
15F7 CALL-SUB   LD   E,(HL)               Obter o byte baixo.
                INC  HL
                LD   D,(HL)               Obter o byte alto.
                EX   DE,HL                Passar o endereço para o par
                                          de registos HL.
                CALL 162C,CALL-JUMP       Chamar a subrotina.
                POP  HL                   Restaurar os registos.
                EXX
                RET                       Retorno a partir daqui a
                                          menos que haja erro.
```

A subrotina «CHAN-OPEN»

Esta subrotina é invocada contendo no registo A um número de stream válido — normalmente +FD a +03. Então, conforme os dados de stream, um dado canal passará a actual.

```
1601 CHAN-OPEN  ADD  A,A                  O valor no registo A é
                ADD  A,+16                duplicado e aumentado de
                LD   L,A                  +16. O resultado passa para L.
                LD   H,+5C                O endereço 5C16 é o endereço
                                          base do stream +00.
                LD   E,(HL)               Obter o primeiro byte dos
                INC  HL                   dados do stream desejado;
                LD   D,(HL)               depois, o segundo byte.
                LD   A,D                  Dar um erro se ambos os
                OR   E                    tipos são zero; senão,
                JR   NZ,1610,CHAN-OP-1    saltar para diante.
```

Mensagem «O — Invalid stream»

```
160E REPORT-O   RST  0008,ERROR-1         Invocar a rotina de tratamento
                DEFB +17                  de erro.
```

Usando os dados do stream, descobrir agora o endereço base da informação de canal associada a esse stream.

```
1610 CHAN-OP-1  DEC  DE                   Reduzir os dados de stream.
                LD   HL,(CHANS)           Endereço base de toda a
                                          área de informação de canal.
                ADD  HL,DE                Formar o endereço requerido
                                          nesta área.
```

A subrotina «CHAN-FLAG»

Esta subrotina define as flags apropriadas para os diferentes canais.

```
1615 CHAN-FLAG  LD   (CURCHL),HL          O par de registos HL
                                          contém o endereço base de
                                          um dado canal.
```

```
                RES   4,(FLAGS2)      Sinalizar «uso de um canal
                                       diferente de K».
                INC   HL               Passar os endereços de saída
                INC   HL               e entrada e fazer HL
                INC   HL               apontar para o código
                INC   HL               do canal.
                LD    C,(HL)           Obter o código.
                LD    HL,+162D         Endereço base da «tabela de
                                       códigos de canal».
                CALL  16DC,INDEXER     Indexar esta tabela e localizar
                                       o deslocamento requerido; mas
                RET   NC               retorno, se não existe um
                                       código de canal igual.
                LD    D,+00            Passar o deslocamento para o
                LD    E,(HL)           par de registos DE.
                ADD   HL,DE            Saltar para diante para a
162C CALL-JUMP  JP    (HL)             rotina que define a flag.
```

A tabela de «Códigos de canal»

```
162D DEFB  4B 06  —  canal «K»   desloc. +06   endereço 1634
162F DEFB  53 12  —  canal «S»   desloc. +12   endereço 1642
1631 DEFB  50 1B  —  canal «P»   desloc. +1B   endereço 164D
1633 DEFB  00     —  separador final
```

A subrotina «Flag do canal 'K'»

```
1634 CHAN-K     SET   0,(TV-FLAG)      Sinalizar «uso de janela inferior».
                RES   5,(FLAGS)        Sinalizar «pronto para tecla».
                SET   4,(FLAGS2)       Sinalizar «usando canal 'K'».
                JR    1646,CHAN-S-1    Saltar para diante.
```

A subrotina «Flag do Canal 'S'»

```
1642 CHAN-S     RES   0,(TV-FLAG)      Sinalizar «uso da janela principal».
1646 CHAN-S-1   RES   1,(FLAGS)        Sinalizar «impressora não usada».
                JP    0D4D,TEMPS       Sair por TEMPS de modo a definir
                                       as variáveis de sistema da cor.
```

A subrotina «Flag do canal 'P'»

```
164D CHAN-P     SET   1,(FLAGS)        Sinaliza «impressora em uso».
                RET
```

A subrotina «MAKE-ROOM»

Trata-se de uma subrotina muito importante. É invocada em muitas ocasiões para «abrir» uma área. Em todos os casos o par de registos HL aponta para a posição **seguinte** àquela onde é requerido «espaço», e o par de registos BC contém o comprimento desse «espaço».

Quando é apenas necessária uma posição, entra-se na subrotina por ONE-SPACE.


```
1652 ONE-SPACE  LD    BC,+0001         É requerida apenas uma
                                       posição.
1655 MAKE-ROOM  PUSH  HL               Guardar o indicador.
                CALL  1F05,TEST-ROOM   Verificar se existe memória
                                       suficiente para a tarefa
                                       que está a ser realizada.
                POP   HL               Restaurar o indicador.
                CALL  1664,POINTERS    Alterar todos os indicadores
                                       antes de abrir o «espaço».
                LD    HL,(STKEND)      Colocar em HL a nova STKEND.
                EX    DE,HL            Comutar «antigo» e «novo».
                LDDR                   Abrir o «espaço» e
                RET                    retorno.
```

Nota: Esta subrotina retorna com o par de registos HL apontando para a posição antes do novo «espaço», e o par de registos DE apontando para a última das novas posições. O novo «espaço» possui, portanto, a descrição: «(HL)+1» a «(DE)» inclusive.

No entanto, como as «novas posições» ainda mantêm os seus valores «antigos», é igualmente possível considerar que o novo «espaço» foi aberto **depois** da posição inicial «(HL)» e tem, portanto, a descrição «(HL)+2» a «(DE)+1».

De facto, o programador parece ter uma preferência pela «segunda descrição», e isto pode ser confuso.

A subrotina «POINTERS»

Sempre que é necessário «fazer» ou «reclamar» uma área, as variáveis de sistema que endereçam posições para além da «posição» alterada devem ser corrigidas da maneira adequada. Na entrada da rotina o par de registos BC contém o número de bytes envolvidos e o par de registos HL endereça o byte anterior a «posição».

```
1664 POINTERS   PUSH  AF             Guarda estes registos.
                PUSH  HL             Copia o endereço de
                                     «posição».
                LD    HL,+5C4B       Isto é VARS, o primeiro
                LD    A,+0E          de 14 indicadores de sistema.
```

Entra-se num ciclo que considera separadamente cada indicador. Só são alterados os indicadores que apontam para além de «posição».

```
1668 PTR-NEXT   LD    E,(HL)         Obter os dois bytes do
                INC   HL             indicador actual.
                LD    D,(HL)
                EX    (SP),HL        Trocar a variável de sistema
                                     pelo endereço de «posição».
                AND   A              A flag «carry» passará a um
                SBC   HL,DE          se o endereço da variável de
                ADD   HL,DE          sistema deve ser actualizado.
                EX    (SP),HL        Restaura «posição».
                JR    NC,167F,PTR-DONE  Salta para diante se o indicador
                                     é mantido; senão, altera-o.
                PUSH  DE             Guarda o valor antigo.
```

```
                EX      DE,HL       Soma agora o valor em BC
                ADD     HL,BC       ao valor antigo.
                EX      DE,HL
                LD      (HL),D      Introduz o novo valor na
                DEC     HL          variável de sistema — byte alto
                LD      (HL),E      antes do byte baixo.
                INC     HL          Aponta de novo para byte alto.
                POP     DE          Obtém o valor antigo.
167F PTR-DONE   INC     HL          Aponta para a variável de
                DEC     A           sistema seguinte e salta atrás
                JR      NZ,166B,PTR-NEXT  até terem sido consideradas as 14.
```

Determina-se agora a dimensão do bloco a deslocar.

```
                EX      DE,HL       Coloca o valor antigo de
                POP     DE          STKEND em HL e restauram-se
                POP     AF          os outros registos.
                AND     A           Descobre a diferença entre
                SBC     HL,DE       o valor antigo de STKEND
                LD      B,H         e «posição».
                LD      C,L         Transfere o resultado de BC
                INC     BC          e soma «1» para o byte presente.
                ADD     HL,DE       Forma o valor antigo de
                EX      DE,HL       STKEND e passa-o para DE
                RET                 antes do retorno.
```

A subrotina «Recolher um número de linha».

No início, o par de registos HL aponta para a posição que está a ser considerada. Se a posição contém um valor que constitui um byte alto apropriado para um número de linha, então o número de linha é reenviado para DE. No entanto, se assim não acontecer, é verificada a posição endereçada por DE; e se também não houver êxito é produzido o número de linha «zero».

```
168F LINE-ZERO  DEFB    +00         Linha número zero.
                DEFB    +00
1961 LINE-NO-A  EX      DE,HL       Considera o outro indicador.
                LD      DE,+168F    Usa a linha número zero.
```

O ponto normal de entrada é LINE-NO.

```
1695 LINE-NO    LD      A,(HL)      Obter o byte alto e
                AND     +C0         verificá-lo.
                JR      NZ,1691,LINE-NO-A  Saltar atrás se não apropriado.
                LD      D,(HL)      Obter o byte alto.
                INC     HL
                LD      E,(HL)      Obter o byte baixo e
                RET                 retorno.
```

A subrotina «RESERVE»

Esta subrotina é normalmente invocada usando RST 0030, BC-SPACES. Na entrada, o último valor no «stack» da máquina é WORKSP, e o valor acima deste é o número de espaços que devem ser «reservados».

Esta subrotina abre sempre «espaço» entre a área de trabalho existente e o «stack» do calculador.

```
169E RESERVE    LD      HL,(STKBOT) Obter o valor actual de STKBOT
                DEC     HL          e decrementá-lo de modo a
                                    obter a última posição da
                                    área de trabalho.
                CALL    1655,MAKE-ROOM  Fazer agora «BC» espaços.
                INC     HL          Apontar para o 1.º espaço novo
                INC     HL          e depois para o segundo.
                POP     BC          Obter o antigo valor de
                LD      (WORKSP),BC WORKSP e restaurá-lo.
                POP     BC          Restaurar BC — n.º de espaços.
                EX      DE,HL       Comutar indicadores.
                INC     HL          Fazer HL apontar para o
                                    primeiro dos bytes deslocados.
                RET                 Retorno.
```

Nota: Pode considerar-se também que o retorno da rotina é feito com o par de registos DE apontando para um primeiro «byte extra» e o par de registos HL para um último «byte extra», tendo estes bytes extra sido acrescentados depois da posição original «(HL)+1».

A subrotina «SET-MIN»

Esta subrotina passa a área de «editing» e as que se lhe seguem para as respectivas dimensões mínimas. De facto, «limpa» todas estas áreas.

```
16B0 SET-MIN    LD      HL,(E-LINE) Obter E-LINE.
                LD      (HL),+0D    Colocar apenas na área de
                LD      (K-CUR),HL  «montagem» o caracter «retorno
                INC     HL          de linha» e o separador final.
                LD      (HL),+80
                INC     HL          Passar à «limpeza» da área de
                LD      (WORKSP),HL trabalho.
```

Entrando aqui, «limpa-se» a área de trabalho e o «stack» do calculador.

```
168F SET-WORK   LD      HL,(WORKSP) Obter WORKSP.
                LD      (STKBOT),HL Limpa a área de trabalho.
```

Entrando aqui, «limpa-se» apenas o «stack» do calculador.

```
16C5 SET-STK    LD      HL,(STKBOT) Obter STKBOT.
                LD      (STKEND),HL Limpa o «stack».
```

Em todos os casos, levar MEM a endereçar a área de memória do calculador.

```
                PUSH    HL          Guardar STKEND.
                LD      HL,+5C92    Base da área de memória.
                LD      (MEM),HL    Passar este endereço para MEM.
                POP     HL          Restaurar STKEND no par de
                RET                 registos HL antes do retorno.
```

A subrotina «Reclamar a linha 'EDIT'»

```
16D4 REC-EDIT   LD   DE,(E-LINE)        Obter E-LINE.
                JP   19E5,RECLAIM-1     Reclamar a memória.
```

A subrotina «INDEXER»

Esta subrotina é usada em diversas ocasiões para consultar tabelas. O ponto de entrada é INDEXER.

```
16DB INDEXER-1  INC  HL                 Passar a considerar o par
                                        de entradas seguinte.
16DC INDEXER    LD   A,(HL)             Obter a primeira de duas
                AND  A                  entradas, mas retorno se for
                RET  Z                  zero — o separador final.
                CP   C                  Comparar com o código
                                        fornecido.
                INC  HL                 Apontar para a 2.ª entrada.
                JR   NZ,16DB,INDEXER-1  Saltar atrás se não foi
                                        encontrada a entrada correcta.
                SCF                     A flag «carry» passa a um
                RET                     após procura com êxito.
```

A rotina do comando «CLOSE #»

Este comando permite ao utilizador fechar streams. No entanto, no caso dos streams +00 a +03, os dados iniciais de stream são restaurados, pelo que estes não podem de facto ser fechados.

```
16E5 CLOSE      CALL 171E,STR-DATA      Obtém-se os dados existentes
                                        sobre o stream.
                CALL 1701,CLOSE-2       Verifica-se o código no
                                        canal desse stream.
                LD   BC,+0000           Prepara para passar a zero
                                        os dados desse stream.
                LD   DE,+A3E2           Prepara para identificar o uso
                EX   DE,HL              dos streams +00 a +03.
                ADD  HL,DE              A flag «carry» passa a um
                                        para os streams +04 a +0F.
                JR   C,16FC,CLOSE—1     Saltar para diante para estes
                LD   BC,+15D4           streams; senão, descobrir a
                ADD  HL,BC              entrada correcta na tabela
                                        «dados iniciais de stream».
                LD   C,(HL)             Obter os dados iniciais para
                INC  HL                 os streams +00 a +03.
                LD   B,(HL)
16FC CLOSE-1    EX   DE,HL              Introduzir os dados; zero
                LD   (HL),C             e zero, ou os valores
                INC  HL                 iniciais.
                LD   (HL),B
                RET
```


A subrotina «CLOSE-2»

O código do canal associado ao stream que é fechado deverá ser «K», «S» ou «P».

```
1701 CLOSE-2    PUSH HL                 Guardar o endereço dos dados do
                                        stream.
                LD   HL,(CHANS)         Obter o endereço base da área
                ADD  HL,BC              de informação de canal e
                                        descobrir os dados de canal
                                        correspondentes ao stream fechado.
                INC  HL                 Ultrapassar os endereços da
                INC  HL                 subrotina e recolher o código
                INC  HL                 desse canal.
                LD   C,(HL)
                EX   DE,HL              Guardar o indicador.
                LD   HL,+1716           Endereço base da tabela de
                                        «fecho de streams».
                CALL 16DC,INDEXER       Indexar esta tabela e localizar
                                        o deslocamento requerido.
                LD   C,(HL)             Passar o deslocamento para
                LD   B,+00              o par de registos BC.
                ADD  HL,BC              Saltar para diante para a
                JP   (HL)               rotina apropriada.
```

A tabela «Fecho de STREAMS»

```
1716 DEFB 4B 05 — canal «K»   deslocamento +05, endereço 171C
1718 DEFB 53 03 — canal «S»   deslocamento +03, endereço 171C
171A DEFB 50 01 — canal «P»   deslocamento +01, endereço 171C
```

Nota: Não existe separador final no fim desta tabela.

A subrotina «Fechar STREAM»

```
171C CLOSE-STR  POP  HL                 Obter o indicador da informação
                RET                     de canal e retorno.
```

A subrotina «Dados de STREAM»

Esta subrotina coloca no par de registos BC os dados correspondentes a um dado stream.

```
171E STR-DATA   CALL 1E94,STK-TO-A      O n.º de stream a considerar
                                        é obtido do «stack» do calculador.
                CP   +10                Dar um erro se o número é
                JR   C,1727,STR-DATA1   superior a +0F.
```

Mensagem «O — Invalid stream»

```
1725 REPORT-O   RST  0008,ERROR-1       Invocar a rotina de tratamento
                DEFB +17                de erro.
```

Continuar, no caso de números de stream válidos.

```
1727 STR-DATA1  ADD   A,+03       Gama agora +03 a +12;
                RLCA              e agora +06 a +24.
                LD    HL,+5C10    Endereço base da área
                                  de dados de stream.
                LD    C,A         Passar o código do stream para
                LD    B,+00       o par de registos BC.
                ADD   HL,BC       Indexar a área de dados e
                LD    C,(HL)      passar os dois bytes de
                INC   HL          dados para o par de registos BC.
                LD    B,(HL)
                DEC   HL          Apontar o indicador para o
                RET               primeiro dos bytes de dados
                                  antes do retorno.
```

A rotina de comando «OPEN #»

Este comando permite ao utilizador abrir streams. É necessário fornecer um código de canal, que terá de ser «K», «k», «S», «s», «P» ou «p».

Note-se que não é feita qualquer tentativa de dar aos streams +00 ou +03 os seus dados iniciais.

```
1736 OPEN       RST   0028,FP-CALC     Usar o calculador.
                DEFB  +01,exchange      Trocar o número de stream
                DEFB  +38,end-calc      e o código de canal.
                CALL  171E,STR-DATA     Obter os dados do stream.
                LD    A,B               Saltar para diante se ambos
                OR    C                 os bytes de dados forem zero,
                JR    Z,1756,OPEN-1     isto é, o stream estiver fechado.
                EX    DE,HL             Guardar DE.
                LD    HL,(CHANS)        Obter CHANS — endereço base
                ADD   HL,BC             da informação de canal,
                INC   HL                e descobrir o código de canal
                INC   HL                associado ao stream que é
                INC   HL                aberto.
                LD    A,(HL)
                EX    DE,HL             Recuperar DE.
                CP    +4B               O código obtido da área de
                JR    Z,1756,OPEN-1     informação de canal deve ser
                CP    +53               «K», «S» ou «P»; dar um
                JR    Z,1756,OPEN-1     erro no caso contrário.
                CP    +50
                JR    NZ,1725,REPORT-O
1756 OPEN-1     CALL  175D,OPEN-2       Recolher os dados apropriados
                                        em DE.
                LD    (HL),E            Introduzir os dados nos
                INC   HL                2 bytes na área de
                LD    (HL),D            informação de stream.
                RET                     Finalmente, retorno.
```

A subrotina «OPEN-2»

São recolhidos os bytes de dados de stream correspondente ao canal associado ao stream que é aberto.


```
175D OPEN-2     PUSH  HL          Guardar HL.
                CALL  2BF1,STK-FETCH  Obter os parâmetros do
                                      código de canal.
                LD    A,B         Dar um erro se a expressão
                OR    C           fornecida for vazia; isto
                JR    NZ,1767,OPEN-3  é, OPEN #5, « ».
```

Mensagem «F — Invalid file name»

```
1765 REPORT-F   RST   0008,ERROR-1     Invocar a rotina de tratamento
                DEFB  +0E               de erro.
```

Continuar, se não ocorreu qualquer erro.

```
1767 OPEN-3     PUSH  BC          Guarda o comprimento da
                                  expressão.
                LD    A,(DE)      Obtém o primeiro caracter.
                AND   +DF         Converte os códigos de
                                  minúsculas para maiúsculas.
                LD    C,A         Desloca o código para o registo C.
                LD    HL,+177A    Endereço base da tabela de
                                  «abertura de streams».
                CALL  16DC,INDEXER  Indexa esta tabela e define
                                    o deslocamento requerido.
                JR    NC,1765,REPORT-F  Saltar atrás se não encontra.
                LD    C,(HL)      Passa a deslocamento para o
                LD    B,+00       par de registos BC.
                ADD   HL,BC       Faz HL apontar para o início
                                  da subrotina apropriada.
                POP   BC          Obtém o comprimento da
                JP    (HL)        expressão antes de saltar
                                  para a subrotina.
```

A tabela «Abertura de STREAMS»

```
177A DEFB  4B 06  —  canal «K»   deslocamento +06,  endereço 1781
177C DEFB  53 08  —  canal «S»   deslocamento +08,  endereço 1785
177E DEFB  50 0A  —  canal «P»   deslocamento +0A,  endereço 1789
1780 DEFB  00     —  separador final
```

A subrotina «OPEN-K»

```
1781 OPEN-K     LD    E,+01       Os bytes de dados serão
                JR    178B,OPEN-END  +01 e +00.
```

A subrotina «OPEN-S»

```
1785 OPEN-S     LD    E,+06       Os bytes de dados serão
                JR    178B,OPEN-END  +06 e +00.
```

A subrotina «OPEN-P»

```
1789 OPEN-P     LD    E,+10       Os bytes de dados serão
                                  +10 e +00.
```

```
1788 OPEN-END   DEC   BC          Diminui o comprimento da
                LD    A,B         expressão e produz erro
                OR    C           se não se trata de um único
                JR    NZ,1765,REPORT-F  caracter; senão, limpa o
                LD    D,A         registo D, obtém HL, e
                POP   HL          retorno.
                RET
```

As rotinas dos comandos «CAT, ERASE, FORMAT e MOVE»

No sistema standard do Spectrum o uso destas instruções conduz à apresentação da mensagem «O — Invalid stream».

```
1793 CAT-ETC.   JR    1725,REPORT-O   Imprimir esta mensagem.
```

As rotinas dos comandos «LIST» e «LLIST»

As rotinas desta parte do programa monitor de 16 K são usadas para produzir listagens do programa Basic em memória central. É necessário avaliar o número de cada linha, expandir as suas palavras-chave e posicionar os cursores apropriados.

O ponto de entrada AUTO-LIST é usado pela rotina principal de execução e pelo EDITOR a fim de produzir uma página de listagem.

```
1795 AUTO-LIST  LD    (LIST-SP),SP    Guarda o indicador de «stack»,
                                      permitindo que o «stack» passe
                                      a zero quando a listagem termina
                                      (ver PO-SCR, 0C55).
                LD    (TV-FLAG),+10   Sinal «listagem automática
                                      no visor».
                CALL  0DAF,CL-ALL     Limpar a parte principal do visor.
                SET   0,(FLAGS)       Comutar para a área de «editing».
                LD    B,(DF-SZ)       Limpar agora a janela inferior
                CALL  0E44,CL-LINE    do visor.
                RES   0,(FLAGS)       Comutar de novo.
                SET   0,(FLAGS2)      Sinal «visor limpo».
                LD    HL,(E-PPC)      Obtém agora o número da linha
                LD    DE,(S-TOP)      actual e o número da linha
                                      «automática».
                AND   A               Se o n.º «actual» é menor
                SBC   HL,DE           que o n.º «automático»,
                ADD   HL,DE           saltar para diante
                JR    C,17E1,AUTO-L-2 para actualizar este.
```

O número «automático» deve agora ser alterado de modo a produzir uma listagem com a linha «actual» aparecendo perto da parte inferior do visor.

```
                PUSH  DE            Guardar o número «automático».
                CALL  196E,LINE-ADDR  Definir o endereço do início
                LD    DE,+02C0      da linha actual e produzir
                EX    DF,HL         um endereço cerca de «um
                SBC   HL,DE         visor antes» (negação).
                EX    (SP),HL       Guardar o «resultado» no
                CALL  196E,LINE-ADDR  «stack» da máquina enquanto é
                                      igualmente recolhido o endereço
                                      da linha «automática» (em HL).
```



```
                POP   BC            O «resultado» passa para o
                                    par de registos BC.
```

Entra-se agora num ciclo. O número da linha «automática» é aumentado em cada passagem até ser provável que a linha «actual» seja mostrada na listagem.

```
17CE AUTO-L-1   PUSH  BC            Guardar o «resultado».
                CALL  19B8,NEXT-ONE   Descobrir o endereço do
                                      início da linha depois da
                                      linha «automática» actual (em DE).
                POP   BC            Restaurar o «resultado».
                ADD   HL,BC         Realizar o cálculo e
                JR    C,17E4,AUTO-L-3  saltar quando termina.
                EX    DE,HL         Passar o endereço da linha
                LD    D,(HL)        seguinte para o par de
                INC   HL            registos HL e recolher
                                    o seu número.

                LD    E,(HL)        Pode actualizar-se agora
                DEC   HL            S—TOP, repetindo-se o teste
                LD    (S-TOP),DE    para a nova linha.
                JR    17CE,AUTO-L-1
```

Pode agora produzir-se a listagem «automática».

```
17E1 AUTO-L-2   LD    (S-TOP),HL    Quando E-PPC inferior a S-TOP.
17E4 AUTO-L-3   LD    HL,(S-TOP)    Obter o número da linha de cima
                CALL  196E,LINE-ADDR  e portanto o seu endereço.
                JR    Z,17ED,AUTO-L-4  Se a linha não pode ser
                EX    DE,HL         encontrada, usar DE.
17ED AUTO-L-4   CALL  1833,LIST-ALL   Produz-se a listagem.
                RES   4,(TV-FLAG)     Retorno para aqui a menos
                RET                   que seja preciso «scroll»
                                      para imprimir a linha actual.
```

O ponto de entrada «LLIST»

Será necessário abrir o canal da impressora.

```
17F5 LLIST      LD    A,+03         Usar stream +03.
                JR    17FB,LIST-1   Saltar para diante.
```

O ponto de entrada «LIST»

Será necessário abrir o canal correspondente à parte principal do visor.

```
17F9 LIST       LD    A,+02         Usar stream +02.
17FB LIST-1     LD    (TV-FLAG),+00   Sinal «listagem normal na
                                      janela principal do visor».
                CALL  2530,SYNTAX-Z   Abrir o canal a menos que se
                CALL  NZ,1601,CHAN-OPEN  verifique a sintaxe.
                RST   0018,GET-CHAR   Com o caracter presente no
                CALL  2070,STR-ALTER  registo A, verificar se
                                      se deve alterar o stream.
```

```
                JR      C,181F,LIST-4       Saltar para diante, se não.
                RST     0018,GET-CHAR       O caracter actual é um
                CP      +3B                 «:»?
                JR      Z,1814,LIST-2       Saltar, se for.
                CP      +2C                 É um «,»?
                JR      NZ,181A,LIST-3      Saltar, se não.
1814 LIST-2     RST     0020,NEXT-CHAR      Deve seguir-se uma expressão
                CALL    1C82,EXPT-1NUM      numérica, como LIST #5,20.
                JR      1822,LIST-5         Saltar para diante com ela.
181A LIST-3     CALL    1CE6,USE-ZERO       Senão, usar zero e saltar
                JR      1822,LIST-5         igualmente.
```

Vir aqui se o stream não foi alterado.

```
181F LIST-4     CALL    1CDE,FETCH-NUM      Obter qualquer linha ou
                                            usar zero se não houver uma.
1822 LIST-5     CALL    1BEE,CHECK-END      Se se verifica a sintaxe da
                                            linha em montagem passar à
                                            declaração seguinte.
                CALL    1E99,FIND-INT       Número de linha para BC.
                LD      A,B                 Byte alto para A.
                AND     +3F                 Limitar o byte alto à gama
                LD      H,A                 correcta e passar todo o
                LD      L,C                 número de linha para HL.
                LD      (E-PPC),HL          Definir E-PPC e descobrir o
                CALL    196E,LINE-ADDR      endereço do início desta linha
                                            ou a primeira linha seguinte
                                            se a actual não existe.
                LD      E,+01               Flag «antes da linha actual».
```

Segue-se o ciclo de controlo da impressão de uma série de linhas.

```
1835 LIST-ALL   CALL    1855,OUT-LINE       Imprimir uma linha inteira.
                RST     0010,PRINT-A-1      Isto será um «retorno de linha».
                BIT     4,(TV-FLAG)         Saltar atrás a menos que se
                JR      Z,1835,LIST-ALL     trate de listagem automática.
                LD      A,(DF-SZ)           Saltar também se ainda existir
                SUB     (S-POSN-hi)         uma parte da janela principal
                JR      NZ,1835,LIST-ALL    do visor que possa ser usada.
                XOR     E                   Retorno possível aqui se o
                RET     Z                   visor está cheio e a linha
                                            actual já foi impressa
                                            (E=+00).
                PUSH    HL                  Mas se falta a linha actual
                PUSH    DE                  na listagem, deve
                LD      HL,+5C6C            actualizar-se S-TOP e imprimir
                CALL    190F,LN-FETCH       uma nova linha (usando
                POP     DE                  scroll).
                POP     HL
                JR      1835,LIST-ALL
```

A subrotina «Imprimir uma linha Basic inteira»

O par de registos HL aponta para o início da linha — a posição que contém o byte alto do número de linha.

Antes de o número de linha ser impresso é verificado, a fim de determinar se se encontra antes da linha «actual», se é a linha «actual» ou se se segue a esta.


```
1855 OUT-LINE   LD      BC,(E-PPC)          Obter o número da linha
                CALL    1980,CP-LINES       «actual» e compará-lo.
                LD      D,+3E               Carregar no registo D o
                                            actual cursor de linha.
                JR      Z,1865,OUT-LINE1    Saltar para diante se se
                                            imprime a linha «actual».
                LD      DE,+0000            Carregar zero no registo D
                                            (não é o cursor) e passar
                RL      E                   E para +01 se a linha está
                                            antes da «actual» e para
                                            +00 se está depois (a flag «carry»
                                            vem de CP-LINES).
1865 OUT-LINE   LD      (BREG),E            Guardar o separador de linha.
                LD      A,(HL)              Obter o byte alto do número
                CP      +40                 de linha e retorno se a
                POP     BC                  listagem está já
                RET     NC                  terminada.
                PUSH    BC
                CALL    1A28,OUT-NUM-2      Pode imprimir-se agora o n.º
                                            de linha — com espaços iniciais.
                INC     HL                  Levar o indicador a endereçar
                INC     HL                  o primeiro código de comando
                INC     HL                  da linha.
                RES     0,(FLAGS)           Sinal «espaço inicial permitido».
                LD      A,D                 Obter o código do cursor e
                AND     A                   saltar para diante a menos
                JR      Z,1881,OUT-LINE3    que se deva imprimir cursor.
                RST     0010,PRINT-A-1      Imprimir agora o cursor.
187D OUT-LINE2  SET     0,(FLAGS)           Sinal «sem espaços agora».
1881 OUT-LINE3  PUSH    DE                  Guardar os registos.
                EX      DE,HL               Passar o indicador para DE.
                RES     2,(FLAGS2)          Sinal «não entre aspas».
                LD      HL,+5C3B            Isto é FLAGS.
                RES     2,(HL)              Sinal «imprimir em modo K».
                BIT     5,(FLAGX)           Saltar para diante se não
                JR      Z,1894,OUT-LINE4    for modo INPUT.
                SET     2,(HL)              Sinal «imprimir em modo L».
```

Entra-se, agora, num ciclo que imprime todos os códigos do resto da linha Basic — saltando sobre as representações em vírgula flutuante onde estas ocorrerem.

```
1894 OUT-LINE4  LD      HL,(X-PTR)          Obter o indicador de erro
                AND     A                   de sintaxe e saltar para
                SBC     HL,DE               diante se não se deve imprimir
                JR      NZ,18A1,OUT-LINE5   um marcador de erro.
                LD      A,+3F               Imprimir o marcador de erro.
                CALL    18C1,OUT-FLASH      É um «?» em FLASH.
18A1 OUT-LINE5  CALL    18E1,OUT-CURS       Verificar se imprime o
                                            cursor.
                EX      DE,HL               Passar o indicador para HL.
                LD      A,(HL)              Obter cada caracter em separado.
                CALL    18B6,NUMBER         Se o caracter é um «indicador
                                            de número», não imprimir a
                                            representação em vírgula flutuante.
                INC     HL                  Actualizar o indicador para a
                                            passagem seguinte.
                CP      +0D                 O caracter é um «retorno
                                            de linha»?
                JR      Z,18B4,OUT-LINE6    Saltar, se for.
```

```
                EX      DE,HL           Passar o indicador para DE.
                CALL    1937,OUT-CHAR   Imprimir caracter.
                JR      1894,OUT-LINE4  Percorrer o ciclo pelo
                                        menos mais uma vez.
```

A linha encontra-se assim impressa.

```
1884 OUT-LINE6  POP     DE              Restaurar o par de registos
                RET                     DE, e retorno.
```

A subrotina «number»

Se o registo A contém o «marcador de número», o par de registos HL é avançado de modo a passar à frente da representação em vírgula flutuante.

```
1886 NUMBER     CP      +0E             É um «marcador de linha»?
                RET     NZ              Saltar, se não.
                INC     HL              Avançar o indicador 6
                IND     HL              vezes de modo a passar o
                INC     HL              «marcador» e as 5 posições
                INC     HL              que contêm a representação
                INC     HL              em vírgula flutuante.
                INC     HL
                LD      A,(HL)          Obter o código actual antes
                RET                     do retorno.
```

A subrotina «imprimir um caracter em 'flash'»

O «cursor de erro» e os «cursores de modo» são impressos usando esta subrotina.

```
18C1 OUT-FLASH  EXX                     Guardar o registo actual.
                LD      HL,(ATTR-T)     Guardar ATTR-T e MASK-T no
                PUSH    HL              «stack» da máquina.
                RES     7,H             Verificar se FLASH está
                SET     7,L             activo.
                LD      (ATTR-T),HL     Usar estes valores alterados
                                        para ATTR-T e MASK-T.
                LD      HL,+5C91        Isto é P-FLAG.
                LD      D,(HL)          Guardar P-FLAG também no
                PUSH    DE              «stack» da máquina.
                LD      (HL),+00        Garantir INVERSE 0, OVER 0,
                                        e não PAPER 9 ou INK 9.
                CALL    09F4,PRINT-OUT  O caracter é impresso agora.
                POP     HL              Restaurar o valor incial
                LD      (P-FLAG),H      de P-FLAG.
                POP     HL              Os valores iniciais de ATTR-T
                LD      (ATTR-T),HL     e MASK-T são também
                EXX                     restaurados; retorno.
                RET
```

A subrotina «imprimir cursor»

É feito um retorno se não se trata do local correcto para imprimir o cursor; mas se é, será impresso o cursor «C», «L», «G», «K» ou «E».


```
18E1 OUT-CURS   LD      HL,(K-CUR)      Obter o endereço do cursor,
                AND     A               mas retorno se não está
                SBC     HL,DE           a ser considerado aqui
                RET     NZ              o local correcto.
                LD      A,(MODE)        Obtém-se e duplica-se o
                RLC     A               valor actual de MODE.
                JR      Z,18F3,OUT-C-1  Saltar para diante a menos que
                                        seja modo «Extended» ou «Graphics».
                ADD     A,+43           Somar o deslocamento correcto
                                        para obter «E» ou «G».
                JR      1909,OUT-C-2    Saltar atrás para o imprimir.
18F3 OUT-C-1    LD      HL,+5C3B        Isto é FLAGS.
                RES     3,(HL)          Sinal «modo K».
                LD      A,+4B           O caracter «K».
                BIT     2,(HL)          Saltar para diante para imprimir
                JR      Z,1909,OUT-C-2  «K» se «impressão a fazer em
                                        modo K».
                SET     3,(HL)          «Impressão a fazer em modo L»,
                                        pelo que sinalizar «modo-L».
                INC     A               Formar o caracter «L».
                BIT     3,(FLAGS2)      Saltar para diante se não se
                JR      Z,1909,OUT-C-2  está em modo «C».
                LD      A,+43           O caracter «C».
1909 OUT-C-2    PUSH    DE              Guardar o par de registos DE
                CALL    18C1,OUT-FLASH  enquanto o cursor é impresso —
                POP     DE              — em FLASH.
                RET                     Retorno em seguida.
```

Nota: É a consideração da letra a imprimir no cursor que determina o modo — «K» ou «L/C».

A subrotina «LN-FETCH»

Entra-se nesta subrotina com o par de registos HL endereçando uma variável de sistema — S-TOP ou E-PPC.

A subrotina termina colocando na variável de sistema o número da linha seguinte.

```
190F LN-FETCH   LD      E,(HL)          É recolhido o número de
                INC     HL              linha guardado na variável.
                LD      D,(HL)
                PUSH    HL              Guardar o indicador.
                EX      DE,HL           Passar o número de linha para
                INC     HL              HL e incrementa-se.
                CALL    196E,LINE-ADDR  Descobre-se o endereço do
                                        início desta linha, ou da linha
                                        seguinte se não é usado o
                                        número de linha.
                CALL    1695,LINE-NO    Obtém-se o número dessa
                                        linha.
                POP     HL              Restaurar o indicador da
                                        variável de sistema.
```

O ponto de entrada LN-STORE é usado pelo EDITOR.

```
191C LN-STORE   BIT     5,(FLAGX)       Retorno se se está em modo
                RET     NZ              «INPUT»; senão, passar o
                LD      (HL),D          número de linha para os
```

```
                DEC     HL              dois bytes da variável de
                LD      (HL),E          sistema.
                RET                     Retorno.
```

A subrotina «impressão de caracteres numa linha Basic»

Todos os caracteres e palavras-chave de uma linha Basic são impressos invocando repetidamente esta rotina.

O ponto de entrada OUT-SP-NO é usado quando se imprimem números de linha que podem necessitar de espaços iniciais.

```
1925 OUT-SP-2   LD      A,E             O registo A contém +20 para
                                        um espaço ou +FF noutro caso.
                AND     A               Comparar o valor e retorno
                RET     M               se não é necessário espaço.
                JR      1937,OUT-CHAR   Saltar para imprimir espaço.
192A OUT-SP-NO  XOR     A               Limpar o registo A.
```

O par de registos HL contém o número de linha, e o par de registos BC o valor para «subtracção repetida» (BC contém «-1000,», «-100» ou «-10».

```
1928 OUT-SP-1   ADD     HL,BC           «Subtracção de ensaio».
                INC     A               Contar cada «ensaio».
                JR      C,1928,OUT-SP-1 Saltar atrás até terminar.
                SBC     HL,BC           Restaurar última «subtracção»
                DEC     A               e descontá-la.
                JR      Z,1925,OUT-SP-2 Se não foram possíveis
                                        «subtracções», saltar atrás para
                                        ver se se deve imprimir espaço.
                JP      15EF,OUT-CODE   Senão, imprimir o algarismo.
```

O ponto de entrada OUT-CHAR é usado para todos os caracteres, palavras-chave e caracteres de comando.

```
1937 OUT-CHAR   CALL    2D1B,NUMERIC    Voltar com «carry» a zero
                                        se trata código de algarismo.
                JR      NC,196C,OUT-CH-3 Saltar para imprimir algarismo.
                CP      +21             Imprimir ainda caracteres de
                JR      C,196C,OUT-CH-3 controlo e «espaço».
                RES     2,(FLAGS)       Sinal «imprimir em modo K».
                CP      +CB             Saltar para diante se se trata
                JR      Z,196C,OUT-CH-3 a palavra-chave «THEN».
                CP      +3A             Saltar para diante a menos que
                JR      NZ,195A,OUT-CH-1 se trate «:».
                BIT     5,(FLAGX)       Saltar para diante para imprimir
                JR      NZ,1968,OUT-CH-2 «:» se o modo é INPUT.
                BIT     2,(FLAGS2)      Saltar para diante se «:»
                JR      Z,196C,OUT-CH-3 não está entre aspas, ou seja,
                                        é um separador.
                JR      1968,OUT-CH-2   «:» está entre aspas e pode
                                        agora ser impresso.
195A OUT-CH-1   CP      +22             Aceitar para impressão todos
                JR      NZ,1968,OUT-CH-2 os caracteres excepto «"».
                PUSH    AF              Guardar o código do caracter
                                        enquanto se sai do «modo aspas».
                LD      A,(FLAGS2)      Recuperar FLAGS2 e comutar
                XOR     +04             o bit 2.
```



```
                LD      (FLAGS2),A      Introduzir o valor emendado
                POP     AF              e restaurar o código do caracter.
1968 OUT-CH-2   SET     2,(FLAGS)       Sinalizar «caracter a imprimir
                                        em modo L».
196D OUT-CH-3   RST     0010,PRINT-A-1  O caracter é impresso antes do
                RET                     retorno.
```

Nota: São os testes executados sobre o caracter actual que determinam se o seguinte será impresso em modo «K» ou «L».

Note-se ainda que o programa não trata «:» em declarações REM.

A subrotina «LINE-ADDR»

Para um dado número de linha, guardado no par de registos HL, esta subrotina produz o endereço inicial dessa linha ou da primeira linha seguinte, colocando-o também no par de registos HL, e o início da linha anterior no par de registos DE.

Se está a ser usado o número de linha, a flag «zero» estará ao valor um. No entanto, se se usa a «primeira linha seguinte», a flag «zero» é passada a zero.

```
196E LINE-ADDR  PUSH    HL              Guardar o número de linha dado.
                LD      HL,(PROG)       Obter a variável de sistema
                LD      D,H             PROG e transferir o endereço
                LD      E,L             para o par de registos DE.
```

Entrar, agora, num ciclo que verifica o número de cada linha do programa em função do número de linha dado, até ser igual ou o exceder.

```
1974 LINE-AD-1  POP     BC              Número de linha dado.
                CALL    1980,CP-LINES   Compará-lo com o número
                                        da linha endereçada.
                RET     NC              Retorno se «carry» em zero;
                PUSH    BC              senão, endereçar o número
                CALL    1988,NEXT-ONE   da linha seguinte.
                EX      DE,HL           Comutar indicadores e
                JR      1974,LINE-AD-1  saltar atrás para considerar
                                        a linha seguinte do programa.
```

A subrotina «comparar números de linha»

O número de linha dado no par de registos BC é comparado com o número de linha endereçado.

```
1980 CP-LINES   LD      A,(HL)          Obter o byte alto do número
                CP      B               de linha endereçado e
                RET     NZ              compará-lo. Retorno, se não
                                        são iguais.
                INC     HL              Depois comparar bytes baixos.
                LD      A,(HL)          Retorno com «carry» ao valor
                DEC     HL              um se o número de linha
                CP      C               endereçado ainda não atingiu
                RET                     o número dado.
```

A subrotina «encontrar cada instrução»

Esta subrotina possui duas funções distintas.

a) Pode ser usada para determinar a instrução de ordem «D» numa linha Basic — terminando com o endereço da posição imediatamente anterior ao início da instrução no par de registos HL, e a flag «zero» ao valor um.
b) Pode igualmente ser usada para descobrir uma instrução, se existir, que se inicie por uma dada palavra-chave (no registo E).

```
1988            INC   HL              Não usado.
                INC   HL
                INC   HL
1988 EACH-STMT  LD    (CH-ADD),HL     Passar CH-ADD para byte actual.
                LD    C,+00           Flag «sem aspas» a um.
```

Entra-se num ciclo que trata cada instrução da linha Basic.

```
1990 EACH-S-1   DEC   D               Diminuir «D» e retorno se
                RET   Z               foi encontrada a declaração
                                      requerida.
                RST   0020,NEXT-CHAR  Obter o código de caracter
                CP    E               seguinte e saltar se não
                JR    NZ,199A,EACH-S-3 concorda com a palavra-chave.
                AND   A               Mas se concordar, retorno
                RET                   com ambas as flags «carry»
                                      e «zero» a zero.
```

Entrar agora num segundo ciclo para considerar os caracteres que se seguem na mesma linha e verificar onde termina a instrução.

```
1998 EACH-S-2   INC   HL              Actualizar o indicador
                LD    A,(HL)          e obter o novo código.
199A EACH-S-3   CALL  1886,NUMBER     Desprezar qualquer número.
                LD    (CH-ADD),HL     Actualizar CH-ADD.
                CP    +22             Saltar para diante se o
                JR    NZ,19A5,EACH-S-4 caracter não é «"».
                DEC   C               Senão, passar a um «flag aspas».
19A5 EACH-S-4   CP    +3A             Saltar para diante se o
                JR    Z,19AD,EACH-S-5 caracter é «:».
                CP    +CB             Saltar para diante a menos
                JR    NZ,19B1,EACH-S-6 que seja o código «THEN».
19AD EACH-S-5   BIT   0,C             Ler a «flag aspas» e saltar
                JR    Z,1990,EACH-S-1 atrás no final de cada
                                      declaração (incluindo
                                      após THEN).
19B1 EACH-S-6   CP    +0D             Saltar atrás a não ser no
                JR    NZ,1998,EACH-S-2 final de uma linha Basic.
                DEC   D               Diminuir o contador de
                SCF                   instruções e passar a um
                RET                   a flag «carry» antes do retorno.
```

A subrotina «NEXT-ONE»

Esta subrotina pode ser usada para descobrir a «linha seguinte» na área de programa ou a «variável seguinte» na área de variáveis. A subrotina trata os seis tipos diferentes de variável que são usados no sistema SPECTRUM.


```
19B8 NEXT-ONE   PUSH  HL              Guardar o endereço da
                                      linha ou variável actuais.
                LD    A,(HL)          Obter o primeiro byte.
                CP    +40             Saltar para diante se se
                JR    C,19D5,NEXT-0-3 procura a «linha seguinte».
                BIT   5,A             Saltar se se procura a
                JR    Z,19D6,NEXT-0-4 variável de string ou array seguinte.
                ADD   A,A             Saltar para diante para variáveis
                JP    M,19C7,NEXT-0-1 numéricas simples ou do tipo
                                      FOR-NEXT.
                CCF                   Apenas variáveis numéricas
                                      de nome extenso.
19C7 NEXT-0-1   LD    BC,+0005        Uma variável numérica ocupa
                JR    NC,19CE,NEXT-0-2 5 posições, mas uma variável de
                LD    C,+12           controlo FOR-NEXT necessitará
                                      de dezoito posições.
19CE NEXT-0-2   RLA                   A flag «carry» passa a zero
                                      apenas para variáveis de
                                      nome extenso; até se atingir
                                      o último caracter.
                INC   HL              Incrementar o indicador
                LD    A,(HL)          e obter o novo código.
                JR    NC,19CE,NEXT-0-2 Saltar atrás a menos que o
                                      código anterior fosse o
                                      último do nome da variável.
                JR    19DB,NEXT-0-5   Saltar para diante (BC=+0005 ou
                                      +0012).
19D5 NEXT-0-3   INC   HL              Passar o byte baixo do
                                      número de linha.
19D6 NEXT-0-4   INC   HL              Apontar para o byte baixo
                                      do comprimento.
                LD    C,(HL)          Passar o comprimento para
                INC   HL              o par de registos BC.
                LD    B,(HL)
                INC   HL              Ter em conta o byte a mais.
```

Em todos os casos é determinado o endereço da linha ou variável «seguinte».

```
19DB NEXT-0-5   ADD   HL,BC           Apontar para o 1.º byte da
                                      linha ou variável «seguinte».
                POP   DE              Obter o endereço da anterior
                                      e continuar para a
                                      subrotina «diferença».
```

A subrotina «diferença»

Forma-se no par de registos BC o «comprimento» entre dois inícios. Os indicadores são modificados mas voltam trocados.

```
19DD DIFFER     AND   A               Prepara para subtracção verdadeira.
                SBC   HL,DE           Determina comprimento entre
                LD    B,H             um início e o seguinte, e
                LD    C,L             passa-o para BC.
                ADD   HL,DE           Forma o endereço e troca
                EX    DE,HL           os registos antes
                RET                   do retorno.
```

A subrotina «reclamar posições»

O ponto de entrada RECLAIM-1 é usado quando o endereço da primeira posição a reclamar se encontra no par de registos DE e o endereço da primeira posição que deve ser deixada como está se encontra no par de registos HL. O ponto de entrada RECLAIM-2 é usado quando o par de registos HL aponta para a primeira posição a reclamar e o par de registos BC contém o número de bytes que devem ser reclamados.

```
19E5 RECLAIM-1   CALL   19DD,DIFFER      Usar a subrotina «diferença» para obter os valores apropriados.
19E8 RECLAIM-2   PUSH   BC               Guardar o número de bytes a reclamar.
                 LD     A,B              Todos os indicadores variáveis de sistema acima da área
                 CPL                     devem ser reduzidos de «BC» pelo que este número é
                 LD     B,A              complementado para 2 antes
                 LD     A,C              de os indicadores se alterarem.
                 CPL
                 LD     C,A
                 INC    BC
                 CALL   1664,POINTERS
                 EX     DE,HL            Enviar o endereço de «primeira posição» para o registo DE
                 POP    HL               e reformar o endereço da
                 ADD    HL,DE            1.ª posição que é deixada.
                 PUSH   DE               Guardar a primeira posição
                 LDIR                    enquanto é «reclamado» o
                 POP    HL               espaço.
                 RET                     Retorno.
```

A subrotina «E-LINE-NO»

Esta subrotina é usada para ler o número da linha que se encontra na área de montagem. Se não existe número de linha, se, portanto, é uma linha Basic directa, o número é considerado zero.

Em todos os casos o número de linha volta no par de registos BC.

```
19FB E-LINE-NO   LD     HL,(E-LINE)      Recolher o indicador da área de montagem.
                 DEC    HL               Levar CH-ADD a apontar para a
                 LD     (CH-ADD),HL      posição antes de um número.
                 RST    0020,NEXT-CHAR   Passar o primeiro código para o registo A.
                 LD     HL,+5C92         Mas antes de considerar o
                 LD     (STKEND),HL      código, transformar a área de memória do calculador numa área temporária de «stack» do calculador.
                 CALL   2D3B,INT-TO-FP   Ler agora os algarismos do número de linha. Voltar com zero se não existe.
                 CALL   2DA2,FP-TO-BC    Comprimir o número de linha para o par de registos BC.
                 JR     C,1A15,E-L-1     Saltar para diante se o número excede 65536.
```



```
                 LD     HL,+D8F0         Senão, compará-lo com
                 ADD    HL,BC            10 000.
                 JP     C,1C8A,REPORT-C  Mensagem C se maior que 9 999.
                 JP     16C5,SET-STK     Retorno por SET-STK, que coloca o «stack» do calculador no local certo.
```

A subrotina «impressão de mensagem e número de linha»

O ponto de entrada OUT-NUM-1 conduzirá à impressão do número disponível no par de registos BC. Qualquer valor superior a 9 999 não será porém impresso.

O ponto de entrada OUT-NUM-2 conduzirá à impressão do número indirectamente endereçado pelo par de registos HL. Desta vez ocorrerão quaisquer espaços iniciais que sejam necessários. O limite dos números a imprimir correctamente é de novo 9 999.

```
1A1B OUT-NUM-1   PUSH   DE               Guardar os outros registos
                 PUSH   HL               durante a subrotina.
                 XOR    A                Limpar o registo A.
                 BIT    7,B              Saltar para diante para imprimir «0» em vez de «-2»
                 JR     NZ,1A42,OUT-NUM-4 nas mensagens sobre a «edit-line».
                 LD     H,B              Passar o número para o par
                 LD     L,C              de registos HL.
                 LD     E,+FF            Flag «sem espaços iniciais».
                 JR     1A30,OUT-NUM-3   Saltar para diante para imprimir o número.
1A28 OUT-NUM-2   PUSH   DE               Guardar o par de registos DE.
                 LD     D,(HL)           Obter o número para o par
                 INC    HL               de registos DE e guardar o
                 LD     E,(HL)           indicador (actualizado).
                 PUSH   HL
                 EX     DE,HL            Passar o número para o par
                 LD     E,+20            de registos HL, e flag «espaço inicial a imprimir».
```

É impressa, agora, a forma inteira do número presente no par de registos HL.

```
1A30 OUT-NUM-3   LD     BC,+FC18         Isto é «-1 000».
                 CALL   192A,OUT-SP-NO   Imprimir primeiro algarismo.
                 LD     BC,+FF9C         Isto é «-100».
                 CALL   192A,OUT-SP-NO   Imprimir segundo algarismo.
                 LD     C,+F6            Isto é «-10».
                 CALL   192A,OUT-SP-NO   Imprimir terceiro algarismo.
                 LD     A,L              Passar qualquer parte restante do número para o registo A.
1A42 OUT-NUM-4   CALL   15EF,OUT-CODE    Imprimir o algarismo.
                 POP    HL               Restaurar os registos
                 POP    DE               antes do retorno.
                 RET
```

7
INTERPRETAÇÃO DE LINHAS E COMANDOS BASIC

As tabelas sintácticas

1. *Tabela de deslocamentos.*

Existe uma tabela de deslocamentos para cada um dos cinquenta comandos Basic.

	Comando	Endereço		Comando	Endereço
1A48 DEFB +B1	DEF FN	1AF9	1A61 DEFB +94	BORDER	1AF5
1A49 DEFB +CB	CAT	1B14	1A62 DEFB +56	CONTINUE	1AB8
1A4A DEFB +BC	FORMAT	1B06	1A63 DEFB +3F	DIM	1AA2
1A4B DEFB +BF	MOVE	180A	1A64 DEFB +41	REM	1AA5
1A4C DEFB +C4	ERASE	1B10	1A65 DEFB +2B	FOR	1A90
1A4D DEFB +AF	OPEN #	1AFC	1A66 DEFB +17	GO TO	1A7D
1A4E DEFB +B4	CLOSE #	1B02	1A67 DEFB +1F	GO SUB	1A86
1A4F DEFB +93	MERGE	1AE2	1A68 DEFB +37	INPUT	1A9F
1A50 DEFB +91	VERIFY	1AE1	1A69 DEFB +77	LOAD	1AE0
1A51 DEFB +92	BEEP	1AE3	1A6A DEFB +44	LIST	1AAE
1A52 DEFB +95	CIRCLE	1AE7	1A6B DEFB +0F	LET	1A7A
1A53 DEFB +98	INK	1AE8	1A6C DEFB +59	PAUSE	1AC5
1A54 DEFB +98	PAPER	1AEC	1A6D DEFB +2B	NEXT	1A98
1A55 DEFB +98	FLASH	1AED	1A6E DEFB +43	POKE	1AB1
1A56 DEFB +98	BRIGHT	1AEE	1A6F DEFB +2D	PRINT	1A9C
1A57 DEFB +98	INVERSE	1AEF	1A70 DEFB +51	PLOT	1AC1
1A58 DEFB +98	OVER	1AF0	1A71 DEFB +3A	RUN	1AA8
1A59 DEFB +98	OUT	1AF1	1A72 DEFB +6D	SAVE	1ADF
1A5A DEFB +7F	LPRINT	1AD9	1A73 DEFB +42	RANDOMIZE	1AB5
1A5B DEFB +B1	LLIST	1ADC	1A74 DEFB +0D	IF	1A81
1A5C DEFB +2E	STOP	1A8A	1A75 DEFB +49	CLS	1ABE
1A5D DEFB +6C	READ	1AC9	1A76 DEFB +5C	DRAW	1AD2
1A5E DEFB +6E	DATA	1ACC	1A77 DEFB +44	CLEAR	1AB8
1A5F DEFB +70	RESTORE	1ACF	1A78 DEFB +15	RETURN	1A8D
1A60 DEFB +48	NEW	1AA8	1A79 DEFB +5D	COPY	1AD6

2. *Tabela de parâmetros*

Para cada um dos cinquenta comandos Basic existem até oito entradas na tabela de parâmetros. Estas entradas incluem detalhes de classificação, separadores requeridos e, quando apropriado, endereços das rotinas de comando.

```
1A7A P-LET      DEFB +01        CLASS-01
                DEFB +3D        '='
```



```
                DEFB +02        CLASS-02
1A7D P-GO-TO    DEFB +06        CLASS-06
                DEFB +00        CLASS-00
                DEFB +67,+1E    GO-TO,1E67
1A81 P-IF       DEFB +06        CLASS-06
                DEFB +CB        'THEN'
                DEFB +05        CLASS-05
                DEFB +F0,+1C    IF,1CF0
1A86 P-GO-SUB   DEFB +06        CLASS-06
                DEFB +00        CLASS-00
                DEFB +ED,+1E    GO-SUB,1EED
1A8A P-STOP     DEFB +00        CLASS-00
                DEFB +EE,+1C    STOP,1CEE
1A8D P-RETURN   DEFB +00        CLASS-00
                DEFB +23,+1F    RETURN,1F23
1A90 P-FOR      DEFB +04        CLASS-04
                DEFB +3D        '='
                DEFB +06        CLASS-06
                DEFB +CC        'TO'
                DEFB +06        CLASS-06
                DEFB +05        CLASS-05
                DEFB +03,+1D    FOR,1D03
1A98 P-NEXT     DEFB +04        CLASS-04
                DEFB +00        CLASS-00
                DEFB +AB,+1D    NEXT,1DAB
1A9C P-PRINT    DEFB +05        CLASS-05
                DEFB +CD,+1F    PRINT,1FCD
1A9F P-INPUT    DEFB +05        CLASS-05
                DEFB +89,+20    INPUT,2089
1AA2 P-DIM      DEFB +05        CLASS-05
                DEFB +02,+2C    DIM,2C02
1AA5 P-REM      DEFB +05        CLASS-05
                DEFB +B2,+1B    REM,1BB2
1AA8 P-NEW      DEFB +00        CLASS-00
                DEFB +B7,+11    NEW,11B7
1AAB P-RUN      DEFB +03        CLASS-03
                DEFB +A1,+1E    RUN,1EA1
1AAE P-LIST     DEFB +05        CLASS-05
                DEFB +F9,+17    LIST,17F9
1AB1 P-POKE     DEFB +08        CLASS-08
                DEFB +00        CLASS-00
                DEFB +80,+1E    POKE,1E80
1AB5 P-RANDOM   DEFB +03        CLASS-03
                DEFB +4F,+1E    RANDOMIZE,1E4F
1AB8 P-CONT     DEFB +00        CLASS-00
                DEFB +5F,+1E    CONTINUE,1E5F
1ABB P-CLEAR    DEFB +03        CLASS-03
                DEFB +AC,+1E    CLEAR,1EAC
1ABE P-CLS      DEFB +00        CLASS-00
                DEFB +6B,+0D    CLS,0D6B
1AC1 P-PLOT     DEFB +09        CLASS-09
                DEFB +00        CLASS-00
                DEFB +DC,+22    PLOT,22DC
1AC5 P-PAUSE    DEFB +06        CLASS-06
                DEFB +00        CLASS-00
                DEFB +3A,+1F    PAUSE,1F3A
1AC8 P-READ     DEFB +05        CLASS-05
                DEFB +ED,+1D    READ,1DED
1ACC P-DATA     DEFB +05        CLASS-05
                DEFB +27,+1E    DATA,1E27
```

```
1ACF P-RESTORE  DEFB  +03       CLASS-03
                DEFB  +42,+1E   RESTORE,1E42
1AD2 P-DRAW     DEFB  +09       CLASS-09
                DEFB  +05       CLASS-05
                DEFB  +82,+23   DRAW,2382
1AD6 P-COPY     DEFB  +00       CLASS-00
                DEFB  +AC,+0E   COPY,0EAC
1AD9 P-LPRINT   DEFB  +05       CLASS-05
                DEFB  +C9,+1F   LPRINT,1FC9
1ADC P-LLIST    DEFB  +05       CLASS-05
                DEFB  +F5,+17   LLIST,17F5
1ADF P-SAVE     DEFB  +0B       CLASS-0B
1AE0 P-LOAD     DEFB  +0B       CLASS-0B
1AE1 P-VERIFY   DEFB  +0B       CLASS-0B
1AE2 P-MERGE    DEFB  +0B       CLASS-0B
1AE3 P-BEEP     DEFB  +08       CLASS-08
                DEFB  +00       CLASS-00
                DEFB  +F8,+03   BEEP,03F8
1AE7 P-CIRCLE   DEFB  +09       CLASS-09
                DEFB  +05       CLASS-05
                DEFB  +20,+23   CIRCLE,2320
1AEB P-INK      DEFB  +07       CLASS-07
1AEC P-PAPER    DEFB  +07       CLASS-07
1AED P-FLASH    DEFB  +07       CLASS-07
1AEE P-BRIGHT   DEFB  +07       CLASS-07
1AEF P-INVERSE  DEFB  +07       CLASS-07
1AF0 P-OVER     DEFB  +07       CLASS-07
1AF1 P-OUT      DEFB  +08       CLASS-08
                DEFB  +00       CLASS-00
                DEFB  +7A,+1E   OUT,1E7A
1AF5 P-BORDER   DEFB  +06       CLASS-06
                DEFB  +00       CLASS-00
                DEFB  +94,+22   BORDER,2294
1AF9 P-DEF-FN   DEFB  +05       CLASS-05
                DEFB  +60,+1F   DEF-FN,1F60
1AFC P-OPEN     DEFB  +06       CLASS-06
                DEFB  +2C       ' ,'
                DEFB  +0A       CLASS-0A
                DEFB  +00       CLASS-00
                DEFB  +36,+17   OPEN,1736
1B02 P-CLOSE    DEFB  +06       CLASS-06
                DEFB  +00       CLASS-00
                DEFB  +E5,+16   CLOSE,16E5
1B06 P-FORMAT   DEFB  +0A       CLASS-0A
                DEFB  +00       CLASS-00
                DEFB  +93,+17   CAT-ETC,1793
1B0A P-MOVE     DEFB  +0A       CLASS-0A
                DEFB  +2C       ' ,'
                DEFB  +0A       CLASS-0A
                DEFB  +00       CLASS-00
                DEFB  +93,+17   CAT-ETC,1793
1B10 P-ERASE    DEFB  +0A       CLASS-0A
                DEFB  +00       CLASS-00
                DEFB  +93,+17   CAT-ETC,1793
1B14 P-CAT      DEFB  +00       CLASS-00
                DEFB  +93,+17   CAT-ETC,1793
```

Nota: os requisitos das diferentes classes de comandos são os seguintes:

CLASSE 00 — Sem outros operandos.
CLASSE 01 — Usado em LET. É necessária uma variável.

CLASSE 02 — Usado em LET. Deve seguir-se uma expressão, numérica ou de cadeia.
CLASSE 03 — Deve seguir-se uma expressão numérica. Será usado zero quando não explícita.
CLASSE 04 — Deve seguir-se uma única variável alfanumérica.
CLASSE 05 — Pode ser fornecido um conjunto de elementos.
CLASSE 06 — Deve seguir-se uma expressão numérica.
CLASSE 07 — Trata elementos de cor.
CLASSE 08 — Devem seguir-se duas expressões numéricas, separadas por vírgula.
CLASSE 09 — Tal como para a Classe 08, mas expressões podem ser precedidas por elementos de cor.
CLASSE 0A — Deve seguir-se uma expressão de cadeia.
CLASSE 0B — Trata rotinas de cassete.

O «MAIN PARSER» no interpretador Basic

A rotina de «parsing» do interpretador Basic é acedida por LINE-SCAN, onde se verifica a sintaxe, e por LINE-RUN quando é executado um programa Basic com uma ou mais instruções.

Cada instrução é considerada por sua vez e usa-se a variável de sistema CH-ADD para apontar para cada código da declaração quando este ocorre na área de programa ou de montagem.

```
1B17 LINE-SCAN  RES   7,(FLAGS)      Sinal «verificação sintáctica».
                CALL  19FB,E-LINE-NO CH-ADD passa a apontar para o
                                     1.º código após número de linha.
                XOR   A              A variável de sistema SUBPPC é
                LD    (SUBPPC),A     inicializada para +00, e
                DEC   A              ERR-NR para +FF.
                LD    (ERR-NR),A
                JR    1B29,STMT-L-1  Saltar para diante para estudar
                                     a 1.ª instrução da linha.
```

O ciclo das instruções

Cada instrução é considerada por sua vez até ser atingido o final da linha.

```
1B28 STMT-LOOP  RST   0020,NEXT-CHAR Avançar CH-ADD na linha.
1B29 STMT-L-1   CALL  16BF,SET-WORK  Limpar área de trabalho.
                INC   (SUBPPC)       Aumentar SUBPPC em cada
                                     passagem pelo ciclo.
                JP    M,1C8A,REPORT-C Mas só podem existir 127
                                     instruções por linha.
                RST   0018,GET-CHAR  Obter um caracter.
                LD    B,+00          Limpar o registo.
                CP    +0D            O caracter é «retorno de
                JR    Z,1B83,LINE-END linha»? Saltar, se for.
                CP    +3A            Percorrer de novo o ciclo
                JR    Z,1B28,STMT-LOOP se for«:».
```

Foi identificada uma instrução e, portanto, considera-se primeiro o seu comando inicial.

```
                LD    HL,+1B76       Carregar no «stack» o
                PUSH  HL             endereço de retorno —
                                     — STMT-RET.
```

```
                LD      C,A             Guardar o comando tempo-
                RST     0020,NEXT-CHAR  rariamente no registo C
                LD      A,C             enquanto CH-ADD é avançada.
                SUB     +CE             Reduzir o código do comando
                                        de +CE; obtém-se a gama +00
                                        a +31 para os 50 comandos.
                JP      C,18CA,REPORT-C Produzir o erro apropriado
                                        se não é um código de comando.
                LD      C,A             Passar este código para o par
                                        de registos BC (contém +00).
                LD      HL,+1A48        Endereço base da tabela de
                                        deslocamentos de sintaxe.
                ADD     HL,BC           O deslocamento é passado
                LD      C,(HL)          para o registo C e usado
                ADD     HL,BC           para calcular o endereço
                                        base das entradas dos comandos
                                        na tabela de parâmetros.
                JR      1B55,GET-PARAM  Saltar para SCAN-LOOP com
                                        este endereço.
```

Cada uma das rotinas aplicáveis ao comando considerado é executada uma a uma. São igualmente considerados quaisquer separadores.

```
1B52 SCAN-LOOP  LD      HL,(T-ADDR)     Indicador temporário das
                                        entradas na tabela de parâmetros.
1B55 GET-PARAM  LD      A,(HL)          Obter cada entrada separadamente.
                INC     HL              Actualizar o indicador para as
                LD      (T-ADDR),HL     entradas da passagem seguinte.
                LD      BC,+1B52        Carregar no «stack» o
                PUSH    BC              endereço de retorno —
                                        — SCAN-LOOP.
                LD      C,A             Copiar a entrada para o
                                        registo C para uso ulterior.
                CP      +20             Saltar para diante se a entrada
                JR      NC,1B6F,SEPARATOR é um «separador».
                LD      HL,+1C01        Endereço base de tabela de
                                        «classes de comandos».
                LD      B,+00           Limpar o registo B e
                ADD     HL,BC           indexar a tabela.
                LD      C,(HL)          Obter o deslocamento e
                ADD     HL,BC           calcular o endereço inicial
                                        da necessária rotina de comando.
                PUSH    HL              Passar o endereço para o
                                        «stack» da máquina.
                RST     0018,GET-CHAR   Antes de realizar um salto
                DEC     B               indirecto para a rotina da
                RET                     classe de comandos, passar o
                                        código do comando para A,
                                        e passar B a +FF.
```

A subrotina «Separador»

É apresentada a mensagem «Nonsense in BASIC» no caso de não se encontrar presente o separador requerido. Mas note-se que, quando está a ser verificada a sintaxe, a mensagem não ocorre de facto no visor — apenas o «marcador de erro».


```
1B6F SEPARATOR  RST     0018,GET-CHAR   Obtém-se o caracter actual
                CP      C               e compara-se com a entrada
                                        na tabela de parâmetros.
                JP      NZ,1C8A,REPORT-C Dar a mensagem de erro se
                                        não concordam.
                RST     0020,NEXT-CHAR  Passar um caracter correcto
                RET                     e retorno.
```

A subrotina «STMT-RET»

Depois da interpretação correcta de uma declaração é feito um retorno para este ponto de entrada.

```
1B76 STMT-RET   CALL    1F54,BREAK-KEY  Verifica-se a tecla BREAK
                                        depois de cada declaração.
                JR      C,1B7D,STMT-R-1 Saltar para diante a menos
                                        que tenha sido premida.
```

Mensagem «L — BREAK into program»

```
1B7B REPORT-L   RST     0008,ERROR-1    Invocar a rotina de tratamento
                DEFB    +14             de erro.
```

Continuar aqui se a tecla BREAK não foi premida.

```
1B7D STMT-R-1   BIT     7,(NSPPC)       Saltar para diante se não
                JR      NZ,1BF4,STMT-NEXT for preciso realizar um «salto».
                LD      HL,(NEWPPC)     Obter o número da «linha nova»
                BIT     7,H             e saltar para diante a menos que
                JR      Z,1B9E,LINE-NEW se trate uma outra declaração
                                        na área de montagem.
```

O ponto de entrada «LINE-RUN»

Este ponto de entrada é usado sempre que se pretende executar uma linha presente na área de montagem. Neste caso, a flag sintaxe/execução estará ao nível um (bit 7 de FLAGS).

O ponto de entrada em causa será também usado na verificação da sintaxe de uma linha presente na área de montagem que possua mais do que uma declaração (o bit 7 de FLAGS terá o valor zero).

```
1B8A LINE-RUN   LD      HL,+FFFE        Uma linha na área de montagem
                LD      (PPC),HL        é considerada como linha «-2».
                LD      HL,(WORKSP)     Fazer HL apontar para o
                DEC     HL              separador final da área de
                LD      DE,(E-LINE)     montagem e DE para a posição
                DEC     DE              anterior ao início desta área.
                LD      A,(NSPPC)       Obter o número da declaração
                                        seguinte a tratar antes de
                JR      1BD1,NEXT-LINE  saltar para diante.
```

A subrotina «LINE-NEW»

Ocorreu um salto no programa e o endereço inicial da nova linha deve ser encontrado.

```
19BE LINE-NEW   CALL  196E,LINE-ADDR   Obtém endereço inicial da linha,
                                       ou da «primeira linha depois».
                LD    A,(NSPPC)        Obtém o número da declaração.
                JR    Z,1BBF,LINE-USE  Saltar para diante se encontra
                AND   A                linha requerida; senão,
                JR    NZ,18EC,REPORT-N verifica a validade do número
                                       da instrução — deve ser zero.
                LD    B,A              Verifica também se a «primeira
                LD    A,(HL)           linha depois» não está depois
                AND   +C0              de «fim do programa».
                LD    A,B
                JR    Z,1BBF,LINE-USE  Saltar para diante no caso de
                                       endereços válidos; senão,
                                       mensagem «OK».
```

Mensagem «0 — OK»

```
1BB0 REPORT-0   RST   0008,ERROR-1     Usar a rotina de tratamento
                DEFB  +FF              de erro.
```

Nota: Obviamente não se trata de um erro no sentido normal — mas de um salto para além do programa.

A rotina do comando «REM»

É libertado o endereço de retorno a STMT-RET, o que tem como efeito ignorar o resto da linha.

```
1BB2 REM        POP   BC               Eliminar endereço STMT-RET.
```

A rotina «LINE-END»

Se se verifica a sintaxe é realizado um simples retorno; mas quando se «executa» deve verificar-se o endereço guardado em NXTLIN antes de poder ser usado.

```
1BB3 LINE-END   CALL  2530,SYNTAX-Z    Retorno, se se verifica a
                RET   Z                sintaxe; senão, obter o
                LD    HL,(NXTLIN)      endereço em NXTLIN.
                LD    A,+C0            Retorno também se o endereço
                AND   (HL)             ocorre depois do fim do
                RET   NZ               programa — termina a execução.
                XOR   A                Sinal «declaração zero»
                                       antes de continuar.
```

A rotina «LINE-USE»

Esta curta rotina possui três funções: 1. Passar a declaração zero a declaração um; 2. Descobrir o número da nova linha e guardá-lo em PPC; 3. Formar o endereço do início da linha que se segue.


```
1BBF LINE-USE   CP    +01              A declaração zero passa
                ADC   A,+00            a declaração «um».
                LD    D,(HL)           O número da linha a usar
                INC   HL               é recolhido e passado
                LD    E,(HL)           para PPC.
                LD    (PPC),DE
                INC   HL               Descobrir agora o
                LD    E,(HL)           «comprimento» da linha.
                INC   HL
                LD    D,(HL)
                EX    DE,HL            Comutar os valores.
                ADD   HL,DE            Formar o endereço do
                INC   HL               início de linha seguinte
                                       em HL, e da posição antes
                                       do 1.º caracter da linha
                                       seguinte em DE.
```

A rotina «NEXT-LINE»

Quando se entra na rotina o par de registos HL aponta para a posição que se segue ao final da linha «seguinte» a tratar, e o par de registos DE para a posição antes do primeiro caracter da linha. Isto aplica-se às linhas existentes na área de programa e também a qualquer linha na área de montagem — onde a linha seguinte será novamente a mesma enquanto existem instruções para interpretar.

```
1BD1 NEXT-LINE  LD    (NXTLIN),HL      Definir NXTLIN para uso depois
                                       de terminada a linha actual.
                EX    DE,HL            Como é habitual, CH-ADD
                LD    (CH-ADD),HL      aponta para a posição antes
                                       do 1.º caracter a considerar.
                LD    D,A              Obtém o número da
                                       instrução.
                LD    E,+00            O registo E é limpo no
                                       caso de ser usado EACH-STMT.
                LD    (NSPPC),+FF      Sinal «sem salto».
                DEC   D                SUBPPC recebe o número da
                LD    (SUBPPC),D       instrução menos um.
                JP    Z,1B28,STMT-LOOP Pode considerar-se agora
                                       uma primeira instrução.
                INC   D                Mas para instruções
                CALL  1988,EACH-STMT   ulteriores é necessário
                                       encontrar o «endereço inicial».
                JR    Z,1BF4,STMT-NEXT Saltar para diante a menos
                                       que a instrução não exista.
```

Mensagem «N — Statement lost».

```
1BEC REPORT-N   RST   0008,ERROR-1     Invocar rotina de tratamento
                DEFB  +16              de erro.
```

A subrotina «CHECK-END»

Trata-se de uma rotina importante, que é invocada de muitos pontos do programa monitor quando é verificada a sintaxe da linha em montagem. O objectivo da rotina em causa é produzir uma mensagem de erro se **não** foi

atingido o final de uma declaração, e passar para a declaração seguinte se a sintaxe é correcta.

```
1BEE CHECK-END   CALL  2530,SYNTAX-Z   Não continuar a menos
                 RET   NZ              que se verifique sintaxe.
                 POP   BC              Libertar os endereços de
                 POP   BC              SCAN-LOOP e STMT-RET
                                       antes de continuar para
                                       STMT-NEXT.
```

A rotina «STMT-NEXT»

Se o caracter actual é um «retorno de linha», então a «instrução seguinte» está na «linha seguinte»; se for «:» estará na mesma linha; mas se for encontrado qualquer outro caracter existe um erro de sintaxe.

```
1BF4 STMT-NEXT   RST   0018,GET-CHAR   Obter o caracter actual.
                 CP    +0D             Considerar a «linha seguinte»
                 JR    Z,1BB3,LINE-END se for um retorno de linha.
                 CP    +3A             Considerar a «instrução seguinte»
                 JP    Z,1B28,STMT-LOOP se for «:».
                 JP    1C8A,REPORT-C   Senão, ocorreu um erro de
                                       sintaxe.
```

A tabela «Classes de comandos»

Endereço	Deslocamento	Classe	Endereço	Deslocamento	Classe
1C01	0F	CLASS-00-1C10	1C07	7B	CLASS-06,1C82
1C02	1D	CLASS-01,1C1F	1C08	8E	CLASS-07,1C96
1C03	4B	CLASS-02,1C4E	1C09	71	CLASS-08,1C7A
1C04	09	CLASS-03,1C0D	1C0A	B4	CLASS-09,1CBE
1C05	67	CLASS-04,1C6C	1C0B	81	CLASS-0A,1C8C
1C06	0B	CLASS-05,1C11	1C0C	CF	CLASS-0B,1CDB

As classes de comandos «00, 03 e 05»

Os comandos de classe 03 podem, ou não, ser seguidos de um número, por exemplo RUN ou RUN 200.

```
1C0D CLASS-03    CALL  1CDE,FETCH-NUM  Obtém número, mas usa zero
                                       se aquele não for explícito.
```

Os comandos de classe 00 não devem possuir operandos, por exemplo, COPY e CONTINUE.

```
1C10 CLASS-00    CP    A               Passar a uma flag «zero».
```

Os comandos de classe 05 podem ser seguidos de um conjunto de elementos, por exemplo, PRINT e PRINT «222».


```
1C11 CLASS-05    POP   BC              Em todos os casos, libertar
                                       o endereço — SCAN-LOOP.
                 CALL  Z,1BEE,CHECK-END Se se trata comandos das
                                       classes 00 e 03 e se veri-
                                       fica sintaxe, passar à
                                       instrução seguinte.
                 EX    DE,HL           Guardar o indicador de linha
                                       no par de registos DE.
```

A rotina «JUMP-C-R»

Depois das entradas das classes de comandos e das entradas de separador na tabela de parâmetros é realizado um salto para a rotina de comando apropriada.

```
1C16 JUMP-C-R    LD    HL,(T-ADDR)     Obter o indicador das
                 LD    C,(HL)          entradas na tabela de
                 INC   HL              parâmetros e o endereço
                 LD    B,(HL)          da rotina de comando.
                 EX    DE,HL           Trocar os indicadores
                 PUSH  BC              de novo e realizar um
                 RET                   salto indirecto para a
                                       rotina de comando.
```

As classes de comandos «01, 02 e 04»

Estas três classes de comandos são usadas pelos comandos de tratamento de variáveis — LET, FOR e NEXT, e, indirectamente, por READ e INPUT.

A classe de comandos 01 respeita à identificação da variável numa declaração LET, READ ou INPUT.

```
1C1F CLASS-01    CALL  28B2,LOOK-VARS  Observar a área de variáveis
                                       para determinar se a
                                       variável já foi ou não
                                       usada.
```

A subrotina «Variável em atribuição»

Esta subrotina desenvolve os valores apropriados para as variáveis de sistema DEST e STRLEN.

```
1C22 VAR-A-1     LD    (FLAGX),+00     Inicializar FLAGX para +00.
                 JR    NC,1C30,VAR-A-2 Saltar para diante se a
                                       variável já foi usada.
                 SET   1,(FLAGX)       Sinal «variável nova».
                 JR    NZ,1C46,VAR-A-3 Produzir erro se se tenta
                                       usar um array não dimensionado.
```

Mensagem «2 — Variable not found».

```
1C2E REPORT-2    RST   0008,ERROR-1    Invocar a rotina de tratamento de
                 DEFB  +01             erro.
```

Continuar o tratamento das variáveis existentes.

```
1C30 VAR-A-2    CALL  Z,2996,STK-VARS    Passam-se os parâmetros de
                                         variáveis de cadeia simples e
                                         variáveis de array para o
                                         «stack» do calculador (STK-VARS
                                         «divide» cadeia se necessário).
                BIT   6,(FLAGS)          Saltar para diante se se trata
                JR    NZ,1C46,VAR-A-3    uma variável numérica.
                XOR   A                  Limpar o registo A.
                CALL  2530,SYNTAX-Z      Obtêm-se os parâmetros da
                CALL  NZ,2BF1,STK-FETCH  cadeia ou array a menos que
                                         se esteja a verificar sintaxe.
                LD    HL,+5C71           Isto é FLAGX.
                OR    (HL)               O bit 0 passa a 1 só quando
                LD    (HL),A             se tratam cadeias simples
                                         completas, sinalizando «cópia
                                         antiga a eliminar».
                EX    DE,HL              HL aponta agora para a cadeia
                                         ou elemento de array.
```

Os trajectos juntam-se agora para definir STRLEN e DEST do modo apropriado. No caso de todas as variáveis numéricas e de variáveis novas de cadeia ou de array, o byte baixo de STRLEN contém a «letra» que identifica a variável. Mas no caso de variáveis de cadeia ou array «antigas», quer estejam completas ou «divididas» («sliced»), contém o «comprimento em atribuição».

```
1C46 VAR-A-3    LD    (STRLEN),BC        Define STRLEN como requerido.
```

DEST contém o endereço de destino de uma variável «antiga» e, de facto, a «fonte» de uma variável «nova».

```
                LD    (DEST),HL          Definir DEST como requerido
                RET                      e retorno.
```

A classe de comandos 02 respeita ao cálculo do valor a atribuir numa declaração LET.

```
1C4E CLASS-02   POP   BC                 Liberta o endereço
                                         SCAN-LOOP.
                CALL  1C56,VAL-FET-1     Realiza a atribuição.
                CALL  1BEE,CHECK-END     Passa à instrução seguinte
                                         através de CHECK-END se
                                         verifica sintaxe, ou de
                RET                      STMT-RET se «em execução».
```

A subrotina «Obter um valor»

Esta subrotina é usada por declarações LET, READ e INPUT primeiro para avaliar e depois para atribuir valores à variável previamente designada.

O ponto de entrada VAL-FET-1 é usado por LET e READ e considera FLAGS, enquanto o ponto de entrada VAL-FET-2 é usado por INPUT e considera FLAGX.


```
1C56 VAL-FET-1  LD    A,(FLAGS)          Usar FLAGS.
1C59 VAL-FET-2  PUSH  AF                 Guardar FLAGS ou FLAGX.
                CALL  24FB,SCANNING      Avaliar a expressão seguinte.
                POP   AF                 Obter FLAGS ou FLAGX
                                         antigas.
                LD    D,(FLAGS)          Obter a nova FLAGS.
                XOR   D                  A natureza (numérica ou
                AND   +40                cadeia) da variável e da
                                         expressão deve concordar.
                JR    NZ,1C8A,REPORT-C   Senão, produzir mensagem C.
                BIT   7,D                Saltar para diante para
                JP    NZ,2AFF,LET        realizar a atribuição a
                                         menos que se verifique
                RET                      sintaxe (retorno simples).
```

A rotina «Classe de comandos 04»

O ponto de entrada CLASS-04 é usado por declarações FOR e NEXT.

```
1C6C CLASS-04   CALL  28B2,LOOK-VARS     Procurar na área de variáveis
                                         a variável em uso.
                PUSH  AF                 Guardar o par de registos
                LD    A,C                AF enquanto o byte discrimi-
                OR    +9F                nador é comparado para
                INC   A                  verificar se se trata de uma
                JR    NZ,1C8A,REPORT-C   variável de controlo FOR-NEXT.
                POP   AF                 Restaurar o registo de flags
                JR    1C22,VAR-A-1       e saltar atrás para passar a
                                         variável encontrada a
                                         «variável em atribuição».
```

A subrotina «Esperar expressões numéricas/de cadeia»

Existe uma série de curtas subrotinas que são usadas para recuperar o resultado da avaliação da expressão seguinte. O resultado de uma expressão isolada é enviado como um «último valor» no «stack» do calculador.

O ponto de entrada NEXT-2NUM é usado quando CH-ADD necessita de actualização para apontar para o início da primeira expressão.

```
1C79 NEXT-2NUM  RST   0020,NEXT-CHAR     Avançar CH-ADD.
```

O ponto de entrada EXPT-2NUM (classe 08) permite a avaliação de duas expressões numéricas, separadas por uma vírgula.

```
1C7A EXPT-2NUM  CALL  1C82,EXPT-1NUM     Avaliar cada expressão por sua
     (CLASS-08)                          vez — começando pela primeira.
                CP    +2C                Produzir mensagem de erro
                JR    NZ,1C8A,REPORT-C   se o separador não é vírgula.
                RST   0020,NEXT-CHAR     Avançar CH-ADD.
```

O ponto de entrada EXPT-1NUM (classe 06) permite a avaliação de uma única expressão numérica.

```
1C82 EXPT-1NUM  CALL  24FB,SCANNING   Avaliar a expressão seguinte.
     (CLASS-06)
                BIT   6,(FLAGS)      Retorno, se o resultado é
                RET   NZ             numérico; senão, é erro.
```

Mensagem «C — Nonsense in BASIC»

```
1C8A REPORT-C   RST   0008,ERROR-1   Invocar a rotina de tratamento de
                DEFB  +0B            erro.
```

O ponto de entrada EXPT-EXP (Classe 0A) permite a avaliação de uma expressão de cadeia.

```
1C8C EXPT-EXP   CALL  24FB,SCANNING   Avaliar a expressão seguinte.
     (CLASS-0A)
                BIT   6,(FLAGS)      Desta vez, retorno se o
                RET   Z              resultado é uma cadeia;
                JR    1C8A,REPORT-C  senão, mensagem de erro.
```

A subrotina «Definir cores permanentes» (classe 07)

Esta subrotina permite transformar em permanentes, as actuais cores temporárias. A classe de comandos 07 respeita, de facto, ao controlo dos elementos de cor.

```
1C96 PERMS      BIT   7,(FLAGS)      É lida a flag sintaxe/execução.
     (CLASS-07)
                RES   0,(TV-FLAG)    Sinal «main screen».
                CALL  NZ,0D4D,TEMPS  Invocar TEMPS, apenas em
                                     «execução» para passar as
                                     cores temporárias a principais.
                POP   AF             Libertar o endereço de retorno —
                                     SCAN-LOOP.
                LD    A,(T-ADDR)     Obter o byte baixo de T-ADDR
                SUB   +13            e subtrair +13 para obter a
                                     gama +D9 a +DE que são as
                                     palavras-chave de INK a OVER.
                CALL  21FC,CO-TEMP-4 Saltar para diante para alterar as co-
                                     res temporárias como indicado
                                     em BASIC.
                CALL  1BEE,CHECK-END Passar à instrução seguinte
                                     se se verifica a sintaxe.
                LD    HL,(ATTR-T)    Os valores temporários de cor
                LD    (ATTR-P),HL    passam agora a permanentes
                                     (ATTR-P e MASK-P).
                LD    HL,+5C91       Isto é P-FLAG; e também
                LD    A,(HL)         deve ser considerada.
```

As instruções que se seguem, copiam os bits pares do byte fornecido para os bits ímpares; deste modo, tornam os bits permanentes iguais aos temporários.

```
                RLCA                 Deslocar a máscara para a es-
                                     querda.
                XOR   (HL)           Passar apenas os bits
                AND   +AA            pares do byte através da máscara.
                XOR   (HL)
                LD    (HL),A         Restaurar o resultado.
                RET
```


A rotina «Classe de comandos 09»

A rotina é usada por PLOT, DRAW e CIRCLE para especificar as condições de «defeito» de FLASH 8, BRIGHT 8, PAPER 8, que são definidas antes de serem considerados quaisquer elementos de cor explícitos.

```
1CBE CLASS-09   CALL  2530,SYNTAX-Z  Saltar para diante se se
                JR    Z,1CD6,CL-09-1 verifica a sintaxe.
                RES   0,(TV-FLAG)    Sinal «janela principal».
                CALL  0D4D,TEMPS     Definir as cores temporárias
                                     da janela principal.
                LD    HL,+5C90       Isto é MASK-T.
                LD    A,(HL)         Obter o seu valor actual,
                OR    +F8            mas manter apenas a parte
                                     INK sem máscara.
                LD    (HL),A         Restaurar o valor que agora
                                     indica «FLASH 8; BRIGHT 8;
                                     PAPER 8».
                RES   6,(P-FLAG)     Garantir ainda «NOT PAPER 9».
                RST   0018,GET-CHAR  Obter o caracter presente
                                     antes de passar ao tratamento
                                     dos elementos de cor explícitos.
1CD6 CL-09-1    CALL  21E2,CO-TEMP   Tratar os elementos de cor
                                     localmente dominantes.
                JR    1C7A,EXPT-2NUM Obter agora os dois primeiros
                                     operandos para PLOT, DRAW e
                                     CIRCLE.
```

A rotina «Classe de comandos 0B»

Esta rotina é usada pelas declarações SAVE, LOAD, VERIFY e MERGE.

```
1CDB CLASS-0B   JP    0605,SAVE-ETC  Saltar para a rotina de
                                     tratamento da cassette.
```

A subrotina «Obter um número»

Esta subrotina conduz à avaliação da expressão numérica seguinte, sendo utilizado zero se não existir tal expressão.

```
1CDE FETCH-NUM  CP    +0D            Saltar para diante no final
                JR    Z,1CE6,USE-ZERO de uma linha.
                CP    +3A            Mas saltar para EXPT-1NUM,
                JR    NZ,1C82,EXPT-1NUM excepto no fim de instrução.
```

Usa-se agora o calculador para somar o valor zero ao «stack» do calculador.

```
1CE6 USE-ZERO   CALL  2530,SYNTAX-Z   Não realizar a operação
                RET   Z               se se verifica sintaxe.
                RST   0028,FP-CALC    Usar o calculador.
                DEFB  +A0,stk-zero    0 «último valor» é zero.
                DEFB  +38,end-calc
                RET                   Retorno com zero acrescentado
                                      ao «stack».
```

As rotinas de comando

A secção do programa monitor de 16 K entre 1CEE e 23FA contém a maior parte das rotinas de comando do interpretador Basic.

A rotina do comando «STOP»

A rotina de comando STOP contém apenas uma chamada à rotina de tratamento de erro.

```
1CEE STOP       RST   0008,ERROR-1    Invoca a rotina de
     (REPORT-9) DEFB  +08             tratamento de erro.
```

A rotina do comando «IF»

Na entrada desta rotina o valor da expressão entre IF e THEN é o «último valor» no «stack» do calculador. Se este é logicamente verdadeiro, é considerada a declaração seguinte; senão, considera-se que a linha terminou.

```
1CF0 IF         POP   BC              Libertar o endereço de
                                      retorno — STMT-RET.
                CALL  2530,SYNTAX-Z   Saltar para diante se
                JR    Z,1D00,IF-1     se verifica sintaxe.
```

Usa-se agora o calculador para «apagar» o último valor do «stack» do calculador, mas deixa-se o par de registos DE endereçando o primeiro byte do valor.

```
                RST   0028,FP-CALC    Usar o calculador.
                DEFB  +02,delete      Elimina-se o «último
                DEFB  +38,end-calc    valor» actual.
                EX    DE,HL           Faz-se HL apontar para o 1.º
                CALL  34E9,TEST-ZERO  byte e chama-se TEST-ZERO.
                JP    C,1BB3,LINE-END Se o valor é falso, saltar
                                      para a linha seguinte.
1D00 IF-1       JP    1B29,STMT-L-1   Mas se é verdadeiro, passar
                                      à declaração seguinte.
```

A rotina do comando «FOR»

Entra-se nesta rotina de comando com o VALOR e o LIMITE da declaração FOR já guardados no topo do «stack» do calculador.


```
1D03 FOR        CP    +CD             Saltar para diante a menos
                JR    NZ,1D10,F-USE-1 que seja dado um STEP.
                RST   0020,NEXT-CHAR  Avançar CH-ADD e obter o
                CALL  1C82,EXPT-1NUM  valor de STEP.
                CALL  1BEE,CHECK-END  Passar à declaração seguinte
                JR    1D16,F-REORDER  se se verifica sintaxe; se-
                                      não, saltar para diante.
```

Não foi indicado um STEP, pelo que se usa o valor «1».

```
1D10 F-USE-1    CALL  1BEE,CHECK-END  Passar à instrução seguinte
                                      se se verifica sintaxe; senão,
                RST   0028,FP-CALC    usar o calculador para pôr um
                DEFB  +A1,stk-one     «1» no «stack» do calculador.
                DEFB  +38,end-calc
```

Os três valores presentes no «stack» do calculador são VALOR (v), LIMITE (l) e STEP (s). É agora necessário manipular estes valores.

```
1D16 F-REORDER  RST   0028,FP-CALC    v,l,s
                DEFB  +C0,st-mem-0    v,l,s          (mem-0=s)
                DEFB  +02,apagar      v,l
                DEFB  +01,trocar      l,v
                DEFB  +E0,obter-mem-0 l,v,s
                DEFB  +01,trocar      l,s,v
                DEFB  +38,fim-calc
```

Estabelece-se agora uma variável de comando FOR, que é tratada como uma área temporária de calculador.

```
                CALL  2AFF,LET        Encontra a variável, ou
                                      cria-a, se necessário (use v).
                LD    (MEM),HL        Transforma-a em «área de memó-
                                      ria».
```

A variável que foi encontrada poJe ser uma simples variável numérica usando apenas seis posições, caso em que necessitará de ser ampliada.

```
                DEC   HL              Obter o nome da variável
                LD    A,(HL)          (um só caracter).
                SET   7,(HL)          Passar a um o bit 7 do nome.
                LD    BC,+0006        Terá pelo menos 6 posições.
                ADD   HL,BC           Apontar HL para elas.
                RLCA                  Rodar o nome e saltar se
                JR    C,1D34,F-L&S    era já uma variável FOR.
                LD    C,+0D           Senão, criar mais treze
                CALL  1655,MAKE-ROOM  posições.
                INC   HL              Fazer HL apontar para a
                                      posição LIMITE.
```

São agora acrescentados os valores iniciais de LIMITE e STEP.

```
1D34 F-L&S      PUSH  HL              Guarda o indicador.
                RST   0028,FP-CLAC    l,s
                DEFB  +02,apagar      l
                DEFB  +02,apagar      —
                DEFB  +38,fim-calc    DE aponta ainda para «l».
```

```
POP     HL          Restaura o indicador e troca
EX      DE,HL       ambos os indicadores.
LD      C,+0A       São deslocados os 10 bytes
LDIR                de LIMITE e STEP.
```

Indicam-se agora o número de linha e o número de declaração do ciclo.

```
LD      HL,(PPC)    Número de linha actual.
EX      DE,HL       Trocar os registos antes de
LD      (HL),E      acrescentar o n.º de linha à
INC     HL          variável de controlo FOR.
LD      (HL),D
LD      D,(SUBPPC)  A instrução em ciclo é
INC     D           sempre a seguinte — quer
INC     HL          exista ou não.
LD      (HL),D
```

A subrotina NEXT-LOOP é invocada para verificar a possibilidade de uma «passagem», sendo realizado o retorno se for possível; senão, deve ser identificada a declaração que se segue no ciclo.

```
CALL    1DDA,NEXT-LOOP  É possível uma «passagem»?
RET     NC              Retorno, se for.
LD      B,(STRLEN-lo)   Obter o nome da variável.
LD      HL,(PPC)        Copiar o n.º da linha actual
LD      (NEWPPC),HL     para NEWPPC.
LD      A,(SUBPPC)      Obter o actual número de
NEG                     instrução e complementá-lo para 2.
LD      D,A             Transferir o resultado para o
                        registo D.
LD      HL,(CH-ADD)     Obter o valor actual de
                        CH-ADD.
LD      E,+F3           Procura «NEXT».
```

É agora realizada uma procura na área de programa, a partir do ponto actual, descobrindo a primeira ocorrência de NEXT seguido da variável correcta.

```
1D64 F-LOOP  PUSH    BC              Guardar o nome da variável.
             LD      BC,(NXTLIN)     Obter o valor actual de
                                     NXTLIN.
             CALL    1D86,LOOK-PROG  Procurar na área de programa,
                                     sendo BC alterado para cada
                                     nova linha examinada.
             LD      (NXTLIN),BC     Guardar o indicador.
             POP     BC              Restaurar o nome da variável.
             JR      C,1D84,REPORT-I Se não existir NEXT produzir
                                     erro.
             RST     0020,NEXT-CHAR  Avançar para além do NEXT
                                     encontrado.
             OR      +20             Permitir maiúsculas e
             CP      B               minúsculas ao verificar o
                                     nome da variável.
             JR      Z,1D7C,F-FOUND  Saltar para diante se concorda.
             RST     0020,NEXT-CHAR  Avançar CH-ADD de novo e
             JR      1D64,F-LOOP     saltar atrás se não for
                                     a variável correcta.
```


NEWPPC contém o número da linha onde se encontrou o NEXT correcto. É agora necessário descobrir e guardar em NSPPC o número da instrução.

```
1D7C F-FOUND  RST     0020,NEXT-CHAR  Avançar CH-ADD.
              LD      A,+01           O contador de instruções
              SUB     D               no registo D contou para trás
                                      a partir de zero devendo ser
                                      subtraído de «1».
              LD      (NSPPC),A       Guarda o resultado.
              RET                     Retorno para STMT-RET.
```

Mensagem «I — FOR without NEXT»

```
1D84 REPORT-I  RST    0008,ERROR-1    Invoca a rotina de
               DEFB   +11             tratamento de erro.
```

A subrotina «LOOK-PROG»

Esta subrotina é usada para descobrir ocorrências de DATA, DEF FN ou NEXT. É iniciada com o código da palavra-chave no registo E e com o par de registos HL apontando para o início da área de procura.

```
1D86 LOOK-PROG  LD      A,(HL)          Obter o caracter presente.
                CP      +3A             Saltar para diante se é «:»,
                JR      Z,1DA3,LOOK-P-2 indicando que existem outras
                                        instruções na linha.
```

Entra-se agora num ciclo que examina cada nova linha do programa.

```
1D88 LOOK-P-1   INC     HL              Obter o byte alto do número
                LD      A,(HL)          de linha e retorno com a flag
                AND     +C0             «carry» a um se não existirem
                SCF                     outras linhas no programa.
                RET     NZ
                LD      B,(HL)          Obtém número de linha, que
                INC     HL              é passado a NEWPPC.
                LD      C,(HL)
                LD      (NEWPPC),BC
                INC     HL              Obtém o comprimento.
                LD      C,(HL)
                INC     HL
                LD      B,(HL)
                PUSH    HL              O indicador é guardado
                ADD     HL,BC           enquanto se forma o endereço
                LD      B,H             do fim da linha no par de
                LD      C,L             registos BC.
                POP     HL              Restaura indicador.
                LD      D,+00           Contador de instruções
                                        passa a zero.
1DA3 LOOK-P-2   PUSH    BC              O indicador de fim de linha
                CALL    1988,EACH-STMT  é guardado enquanto são exami-
                POP     BC              nadas as instruções da linha.
                RET     NC              Retorno se houve uma «ocorrência»;
                JR      1D88,LOOK-P-1   senão, passa a considerar a
                                        linha seguinte.
```

A rotina do comando «NEXT»

A «variável em atribuição» já foi determinada (ver CLASS-04, 1C6C); e resta alterar VALOR da forma apropriada.

```
1DAB NEXT   BIT   1,(FLAGX)          Saltar para dar mensagem de
            JP    NZ,1C2E,REPORT-2    erro se não se encontra variável.
            LD    HL,(DEST)           Obtém o endereço da variável,
            BIT   7,(HL)              sendo o nome ainda melhor
            JR    Z,1DD8,REPORT-1     verificado.
```

Em seguida, VALOR e PASSO («STEP») são manipulados pelo calculador.

```
              HL                    Passar o nome.
              (MEM),HL              Passar a variável a «área de
                                    memória» variável.
              0028,FP-CALC          —
              +E0,obter-mem-0       v
              +E2,obter-mem-2       v,s
              +0F,soma              v+s
              +C0,st-mem-0          v+s
              +02,apagar            —
              +38,fim-calc          —
```

O resultado da soma de VALOR e PASSO é agora comparado com o LIMITE invocando NEXT-LOOP.

```
              CALL  1DDA,NEXT-LOOP  Verifica o novo VALOR em
                                    função de LIMITE.
              RET   C               Retorno, se o ciclo FOR-
                                    -NEXT terminou.
```

Senão, recupera o número de linha e a declaração de ciclo.

```
              LD    HL,(MEM)        Determina o endereço do
              LD    DE,+000F        byte baixo do número de
              ADD   HL,DE           linha do ciclo.
              LD    E,(HL)          Recupera este número de linha.
              INC   HL
              LD    D,(HL)
              INC   HL
              LD    H,(HL)          Segue-se o número de
                                    declaração.
              EX    DE,HL           Troca os números antes de
              JP    1E73,GO-TO-2    saltar para diante a fim de
                                    os tratar como linha de
                                    destino de uma instrução GO TO.
```

Mensagem «1 — NEXT without FOR»

```
1DD8 REPORT-1  RST   0008,ERROR-1   Invoca a rotina de
               DEFB  +00            tratamento de erro.
```

A subrotina «NEXT-LOOP»

Esta subrotina é usada para determinar se foi excedido LIMITE pelo VALOR presente. É necessário ter em conta o sinal de STEP.

A subrotina devolve a flag «carry» a um se LIMITE foi excedido.

```
1DDA NEXT-LOOP  RST   0028,FP-CALC      —
                DEFB  +E1,obter-mem-1   l
                DEFB  +E0,obter-mem-0   l,v
                DEFB  +E2,obter-mem-2   l,v,s
                DEFB  +36,menos-0       l,v,(1/0)
                DEFB  +00,saltar-verdade l,v,(1/0)
                DEFB  +02,para NEXT-1   l,v,(1/0)
                DEFB  +01,troca         v,l
1DE2 NEXT-1     DEFB  +03,subtrai       v-l ou l-v
                DEFB  +37,maior-0       (1/0)
                DEFB  +00,saltar-verdade (1/0)
                DEFB  +04,para NEXT-2   —
                DEFB  +38,fim-calc      —
                AND   A                 Limpar flag «carry» e
                RET                     retorno — ciclo possível.
```

Mas se o ciclo é impossível a flag deve ser passada a um.

```
1DE9 NEXT-2     DEFB  +38,end-calc      —
                SCF                     Passar a um a flag «carry»
                RET                     e retorno.
```

A rotina do comando «READ»

O comando READ permite a leitura de uma lista de dados e tem um efeito semelhante a uma série de declarações LET.

Cada atribuição no interior de uma única declaração READ é tratada separadamente. A variável de sistema X-PTR é usada como posição de armazenamento do indicador da declaração READ, enquanto CH-ADD é usada para percorrer a lista de dados.

```
1DEC READ-3     RST   0020,NEXT-CHAR    Vir aqui em cada passagem,
                                        após a primeira, para percorrer
                                        a declaração READ.
1DED READ       CALL  1C1F,CLASS-01     Verificar se a variável já foi
                                        lida antes; descobrir a
                                        entrada existente se houver.
                CALL  2530,SYNTAX-Z     Saltar para diante se se
                JR    Z,1E1E,READ-2     verifica sintaxe.
                RST   0018,GET-CHAR     Guardar o actual indicador
                LD    (X-PTR),HL        CH-ADD em X-PTR.
                LD    HL,(DATADD)       Obter o actual indicador da
                LD    A,(HL)            lista de dados e saltar para
                CP    +2C               diante a menos que se deva
                JR    Z,1E0A,READ-1     procurar outra instrução DATA.
                LD    E,+E4             Procura «DATA».
                CALL  1D86,LOOK-PROG    Saltar para diante se a procura
                JR    NC,1E0A,READ-1    tem êxito.
```

Mensagem «E — Out of DATA»

```
1E08 REPORT-E   RST   0008,ERROR-1      Invoca a rotina de tratamento
                DEFB  +0D               de erro.
```

Continua — recolhendo um valor da lista de dados.

```
1E0A READ-1     CALL  0077,TEMP-PTR1    Avançar o indicador pela lista
                                        de dados; definir CH-ADD.
                CALL  1C56,VAL-FET-1    Obter o valor e atribuí-lo à
                                        variável.
                RST   0018,GET-CHAR     Obter o valor actual de
                LD    (DATADD),HL       CH-ADD e guardá-lo em
                                        DATADD.
                LD    HL,(X-PTR)        Obter o indicador da
                LD    (X-PTR-hi),+00    declaração READ e limpar
                                        X-PTR.
                CALL  0078,TEMP-PTR2    Levar CH-ADD a apontar de
                                        novo para a instrução READ.
1E1E READ-2     RST   0018,GET-CHAR     Obter o caracter presente
                CP    +2C               e ver se é um ",".
                JR    Z,1DEC,READ-3     Se é, saltar atrás porque
                                        existem mais elementos, se-
                CALL  1BEE,CHECK-END    não, retorno por CHECK-END
                RET                     (se se verifica sintaxe)
                                        ou pela instrução RET
                                        (para STMT-RET).
```

A rotina do comando «DATA»

Durante a verificação de sintaxe as declarações DATA são verificadas a fim de garantir que contêm uma série de expressões válidas, separadas por vírgulas. Mas durante a execução, estas declarações são ultrapassadas pelo sistema.

```
1E27 DATA       CALL  2530,SYNTAX-Z     Saltar para diante a não ser
                JR    NZ,1E37,DATA-2    que se verifique a sintaxe.
```

Entra-se num ciclo que trata cada expressão da declaração DATA.

```
1E2C DATA-1     CALL  24FB,SCANNING     Observar a expressão seguinte.
                CP    +2C               Verificar se o separador é
                                        correcto — um ",";
                CALL  NZ,1BEE,CHECK-END mas passar adiante se
                                        não concorda.
                RST   0020,NEXT-CHAR    Percorrer ciclo enquanto
                JR    1E2C,DATA-1       houver expressões.
```

A declaração DATA deve ser desprezada durante a execução.

```
1E37 DATA-2     LD    A,+E4             É uma declaração DATA que
                                        deve ser passada.
```

A subrotina «PASS-BY»

No início, o registo A possuirá o código correspondente a «DATA» ou a «DEF FN», conforme o tipo de declaração que está a ser «ultrapassada».

```
1E39 PASS-BY    LD    B,A               Coloca em BC um número
                                        muito alto.
```



```
                CPDR                    Voltar atrás procurando
                                        a palavra-chave.
                LD    DE,+0200          Procurar a declaração
                JP    1988,EACH-STMT    seguinte na linha (a
                                        declaração «D-1» a partir
                                        da posição actual).
```

A rotina do comando «RESTORE»

O operando de um comando RESTORE é considerado como um número de linha, sendo usado zero se não for dado qualquer operando.
O ponto de entrada REST-RUN é usado pela rotina de comando RUN.

```
1E42 RESTORE    CALL  1E99,FIND-INT2    Comprimir o operando no par
                                        de registos BC.
1E45 REST-RUN   LD    H,B               Transferir o resultado para
                LD    L,C               o par de registos HL.
                CALL  196E,LINE-ADDR    Descobrir o endereço dessa
                                        linha ou da seguinte.
                DEC   HL                Fazer DATADD apontar para a
                LD    (DATADD),HL       posição anterior.
                RET                     Retorno.
```

A rotina do comando «RANDOMIZE»

O operando é mais uma vez colocado no par de registos BC e transferido para a variável de sistema apropriada. No entanto, se o operando for zero, é usado em vez dele o valor em FRAMES1 e FRAMES2.

```
1E4F RANDOMIZE  CALL  1E99,FIND-INT2    Recuperar o operando.
                LD    A,B               Saltar para diante a menos
                OR    C                 que o valor do operando
                JR    NZ,1E5A,RAND-1    seja zero.
                LD    BC,(FRAMES1)      Obter os dois bytes de
                                        menor ordem de FRAMES.
1E5A RAND-1     LD    (SEED),BC         Passar o resultado à
                RET                     variável de sistema SEED
                                        antes do retorno.
```

A rotina do comando «CONTINUE»

O número de linha e de instrução no interior dessa linha são objecto de um salto.

```
1E5F CONTINUE   LD    HL,(OLDPPC)       Número de linha.
                LD    D,(OSPCC)         Número de declaração.
                JR    1E73,GO-TO-2      Saltar para diante.
```

A rotina do comando «GO TO»

O operando de uma GO TO deve ser um número de linha na gama «1» a «9999», mas, de facto, compara-se com um valor superior de «61439».

```
1E67 GO-TO      CALL  1E99,FIND-INT2   Obter o operando e
                LD    H,B              transferi-lo para HL.
                LD    L,C
                LD    D,+00            Passar o número de instrução
                                       para zero.
                LD    A,H              Produzir a mensagem de erro
                CP    +F0              «Integer out of range»
                JR    NC,1E9F,REPORT-B para linhas acima de 61439.
```

É usado o ponto de entrada GO-TO-2 para determinar o número da linha seguinte para tratamento em diversas circunstâncias.

```
1E73 GO-TO-2    LD    (NEWPPC),HL      Indicar o número de linha e
                LD    (NSPPC),D        depois o número de instrução.
                RET                    Retorno; para STMT-RET.
```

A rotina do comando «OUT»

Os dois parâmetros da instrução OUT são obtidos do «stack» do calculador e usados do modo apropriado.

```
1E7A OUT        CALL  1E85,TWO-PARAM   Obtêm-se os operandos.
                OUT   (C),A            Instrução OUT.
                RET                    Retorno; a STMT-RET.
```

A rotina do comando «POKE»

A operação POKE é realizada de um modo semelhante.

```
1E80 POKE       CALL  1E85,TWO-PARAM   Obtém-se o operando.
                LD    (BC),A           Instrução POKE.
                RET                    Retorno; para STMT-RET.
```

A subrotina «TWO-PARAM»

O parâmetro do topo do «stack» do calculador deve poder ser comprimido num único registo. Encontra-se complementado para dois quando é negativo. O segundo parâmetro deve poder ser comprimido num par de registos.

```
1E85 TWO-PARAM  CALL  2DD5,FP-TO-A     Obter o parâmetro.
                JR    C,1E9F,REPORT-B  Produzir um erro se o
                                       número é muito alto.
                JR    Z,1E8E,TWO-P-1   Saltar para diante para números
                NEG                    positivos mas não para
                                       negativos em complemento para 2.
1E8E TWO-P-1    PUSH  AF               Guardar o 1.º parâmetro
                CALL  1E99,FIND-INT2   enquanto se obtém o 2.º.
                POP   AF               Restaura o primeiro parâmetro
                RET                    antes do retorno.
```


A subrotina «Descobrir inteiros»

Obtém-se o «último valor» no «stack» do calculador, que é passado a um único registo ou a um par de registos entrando por FIND-INT1 ou FIND-INT2 respectivamente.

```
1E94 FIND-INT1  CALL  2DD5,FP-TO-A     Obtém «último valor».
                JR    1E9C,FIND-I-1    Saltar para diante.
1E99 FIND-INT2  CALL  2DA2,FP-TO-BC    Obtém o «último valor».
1E9C FIND-I-1   JR    C,1E9F,REPORT-B  Em ambos os casos o «overflow»
                                       é indicado pela «carry» a um.
                RET   Z                Retorno para todos os n.os
                                       positivos que existem na gama.
```

Mensagem «B — Integer out of range».

```
1E9F REPORT-B   RST   0008,ERROR-1     Invocar a rotina de
                DEFB  +0A              tratamento de erro.
```

A rotina do comando «RUN»

O parâmetro do comando RUN é passado para NEWPPC invocando a rotina de comando GO TO. As operações de «RESTORE 0» e «CLEAR 0» são em seguida realizadas, antes do retorno.

```
1EA1 RUN        CALL  1E67,GO-TO       Definir NEWPPC como apropriado.
                LD    BC,+0000         Realizar um «RESTORE 0».
                CALL  1E45,REST-RUN
                JR    1EAF,CLEAR-1     Sair pela rotina do
                                       comando CLEAR.
```

A rotina do comando «CLEAR»

Esta rotina permite «limpar» a área das variáveis e a de imagem, deslocando também a RAMTOP. Descido a esta última operação, o «stack»-máquina é reconstruído, sendo igualmente limpo o «stack» de GO SUB.

```
1EAC CLEAR      CALL  1E99,FIND-INT2   Obter o operando — usando
                                       zero se não for explícito.
1EAF CLEAR-RUN  LD    A,B              Saltar adiante se o operando
                OR    C                não for zero. Quando chamada
                JR    NZ,1EB7,CLEAR-1  por RUN não há salto.
                LD    BC,(RAMTOP)      Se zero, usar o valor
                                       existente em RAMTOP.
1EB7 CLEAR-1    PUSH  BC               Guardar o valor.
                LD    DE,(VARS)        Reclamar o espaço da actual
                LD    HL,(E-LINE)      área de variáveis.
                DEC   HL
                CALL  19E5,RECLAIM-1
                CALL  0D6B,CLS         Limpar o ficheiro de imagem.
```

O valor no par de registos BC que será usado como RAMTOP é comparado, a fim de garantir que não seja demasiado elevado ou baixo.

```
                LD      HL,(STKEND)     O valor actual de STKEND
                LD      DE,+0032        é aumentado de «50» antes de
                ADD     HL,DE           ser comparado. Forma assim um
                POP     DE              limite inferior.
                SBC     HL,DE
                JR      NC,1EDA,REPORT-M A RAMTOP será muito baixa.
                LD      HL,(P-RAMT)     Para o teste superior verifica-
                AND     A               -se o valor de RAMTOP pelo de
                SBC     HL,DE           P-RAMT.
                JR      NC,1EDC,CLEAR-2 Saltar, se aceitável.
```

Mensagem «M — RAMTOP no good».

```
1EDA REPORT-M   RST     0008,ERROR-1    Invocar a rotina de
                DEFB    +15             tratamento de erro.
```

Continuar com a operação CLEAR.

```
1EDC CLEAR-3    EX      DE,HL           O valor pode agora ser passado
                LD      (RAMTOP,HL      para RAMTOP.
                POP     DE              Obter endereço — STMT-RET.
                POP     BC              Obter «endereço de erro».
                LD      (HL),+3E        Introduzir um separador
                                        final do «stack» de GO SUB.
                DEC     HL              Deixar uma posição.
                LD      SP,HL           Fazer o indicador de «stack» apontar
                                        para o «stack» GO SUB vazio.
                PUSH    BC              Passar o «endereço de erro» para
                LD      (ERR-SP),SP     o «stack» e guardar o seu endereço
                                        em ERR-SP.
                EX      DE,HL           Retorno indirecto para
                JP      (HL)            STMT-RET.
```

Nota: Quando a rotina é invocada por RUN, os valores de NEWPPC e NSPPC são afectados e não é possível encontrar quaisquer instruções que se sigam a RUN antes de o salto ser dado.

A rotina do comando «GO SUB»

O valor actual de PPC e o valor incrementado de SUBPPC são guardados no «stack» de GO SUB.

```
1EED GO-SUB     POP     DE              Guardar endereço — STMT-RET.
                LD      H,(SUBPPC)      Obter número de instrução e
                INC     H               incrementá-lo.
                EX      (SP),HL         Trocar o «endereço de erro»
                                        pelo número de instrução.
                INC     SP              Reclamar o uso de uma posição.
                LD      BC,(PPC)        Depois guardar o número da
                PUSH    BC              linha actual.
                PUSH    HL              Enviar o «endereço de erro»
                LD      (ERR-SP),SP     para o «stack»-máquina e fazer
                                        ERR-SP apontar para ele.
                PUSH    DE              Enviar o endereço —
                                        — STMT-RET.
```



```
                CALL    1E67,GO-TO-1    Passar NEWPPC e NSPPC para
                                        os valores requeridos.
                LD      BC,+0014        Mas antes de realizar o
                                        salto, ver se há espaço.
```

A subrotina «TEST-ROOM»

É realizada uma série de testes para garantir que existe suficiente memória livre para a tarefa que está a ser realizada.

```
1F05 TEST-ROOM  LD      HL,(STKEND)     Somar ao valor tirado de
                ADD     HL,BC           STKEND o trazido para a rotina
                                        pelo par de registos
                                        BC.
                JR      C,1F15,REPORT-4 Saltar para a frente se o
                                        resultado é maior que +FFFF.
                EX      DE,HL           Tentar novamente admitindo mais oi-
                LD      HL,+0050        tenta bytes.
                ADD     HL,DE
                JR      C,1F15,REPORT-4
                SBC     HL,SP           Finalmente verificar o valor em
                                        relação ao endereço no «stack».
                RET     C               Retorno se for satisfatório.
```

Mensagem «4 — Out of memory»

```
1F15 REPORT-4   LD      L,+03           É um erro de «execução» e o
                JP      0055,ERROR-3    marcador de erro não é usado.
```

A subrotina «memória livre»

Não existe uma instrução «FREE» no Spectrum mas existe a subrotina que executa essa tarefa.

Pode avaliar-se o espaço livre em qualquer momento usando:

PRINT 65536-USR 7962.

```
1F1A FREE-MEM   LD      BC,+0000        Não permitir um excesso.
                CALL    1F05,TEST-ROOM  Fazer o teste e passar o
                LD      B,H             resultado para o registo BC
                LD      C,L             antes do retorno.
                RET
```

A rotina do comando «RETURN»

O número de linha e o número de instrução que serão objecto de um «retorno» são obtidos do «stack» de GO SUB.

```
1F23 RETURN     POP     BC              Obter endereço — STMT-RET.
                POP     HL              Obter «endereço de erro».
                POP     DE              Obter a última entrada no
                                        «stack» de GO SUB.
                LD      A,D             A entrada é verificada para
                CP      +3E             testar se é o separador final
                JR      Z,1F36,REPORT-7 do «stack» de GO SUB; saltar se é.
```

```
                DEC   SP          A entrada usa apenas três
                                  posições.
                EX    (SP),HL     Trocar o número de instrução
                                  pelo «endereço de erro».
                EX    DE,HL       Deslocar o número de instrução.
                LD    (ERR-SP),SP Redefinir o indicador de erro.
                PUSH  BC          Substituir o endereço —
                                  — STMT-RET.
                JP    1E73,GO-TO-2 Saltar atrás para alterar
                                  NEWPPC e NSPPC.
```

Mensagem «7 — RETURN without GOSUB»

```
1F36 REPORT-7   PUSH  DE          Repor o separador final
                PUSH  HL          e o «endereço de erro».
                RST   0008,ERROR-1 Invocar a rotina de
                DEFB  +06         tratamento de erro.
```

A rotina do comando «PAUSE»

O período da pausa é determinado contando o número de interrupções mascaráveis que ocorrem 50 vezes por segundo.

A pausa é terminada ao fim do número adequado de interrupções ou quando a variável de sistema FLAGS indica que se premiu uma tecla.

```
1F3A PAUSE      CALL  1E99,FIND-INT2 Obter o operando.
1F3D PAUSE-1    HALT                Esperar uma interrupção mascarável.
                DEC   BC            Diminuir o contador.
                LD    A,B           Se o contador atinge zero,
                OR    C             a pausa terá chegado ao
                JR    Z,1F4F,PAUSE-END seu fim.
                LD    A,B           Se o operando fosse zero, BC
                AND   C             conteria agora +FFFF e este
                INC   A             valor seria passado a zero.
                JR    NZ,1F49,PAUSE-2 Salto para todos os
                INC   BC            outros valores do operando.
1F49 PAUSE-2    BIT   5,(FLAGS)     Salto atrás a menos que se
                JR    Z,1F3D,PAUSE-1 tenha premido uma tecla.
```

O período de pausa terminou agora.

```
1F4F PAUSE-END  RES   5,(FLAGS)   Sinal «tecla não premida».
                RET               Retorno — a STMT-RET.
```

A subrotina «BREAK-KEY»

Esta subrotina é invocada em vários casos para ler a tecla BREAK. A flag «carry» é devolvida ao valor zero apenas no caso de terem sido premidas simultaneamente as teclas SHIFT e BREAK.

```
1F54 BREAK-KEY  LD    A,+7F       Formar o endereço de porto
                IN    A,(+FE)     +7FFE e ler um byte dele.
                RRA               Examinar apenas o bit zero
                                  passando-o à posição «carry».
```



```
                RET   C           Retorno se não é premida a
                                  tecla BREAK.
                LD    A,+FE       Formar o endereço de porto
                IN    A,(+FE)     +FEFE e ler um byte dele.
                RRA               Examinar de novo o bit zero.
                RET               Retorno com «carry» em zero se
                                  estão a ser premidas ambas as teclas.
```

A rotina do comando «DEF FN»

Durante a verificação da sintaxe, as declarações DEF FN são observadas a fim de garantir que possuem a forma correcta. É igualmente dedicado espaço ao resultado da avaliação da função.

Mas, durante a execução, as declarações DEF FN são desprezadas.

```
1F60 DEF-FN     CALL  2530,SYNTAX-Z Saltar para diante se se
                JR    Z,1F6A,DEF-FN-1 verifica a sintaxe.
                LD    A,+CE       Senão, passar à frente da
                JP    1E39,PASS-BY declaração DEF FN.
```

Considerar primeiro a variável da função.

```
1F6A DEF-FN-1   SET   6,(FLAGS)   Sinal «uma variável numérica».
                CALL  2C8D,ALPHA  Verificar se o código presente
                                  é uma letra.
                JR    NC,1F89,DEF-FN-4 Saltar para diante, se não.
                RST   0020,NEXT-CHAR Obter o caracter seguinte.
                CP    +24         Saltar para diante a menos
                JR    NZ,1F7D,DEF-FN-2 que seja «$».
                RES   6,(FLAGS)   Alterar o bit 6 por ser uma
                                  variável de cadeia.
                RST   0020,NEXT-CHAR Obter o caracter seguinte.
1F7D DEF-FN-2   CP    +28         O nome da variável deve ser
                JR    NZ,1F8D,DEF-FN-7 seguido de um «(».
                RST   0020,NEXT-CHAR Obter o caracter seguinte.
                CP    +29         Saltar para diante se é um
                JR    Z,1FA6,DEF-FN-6 «)», dado que não existem
                                  parâmetros da função.
```

Entra-se agora num ciclo para tratar separadamente cada parâmetro.

```
1F86 DEF-FN-3   CALL  2C8D,ALPHA  O código actual deve ser
1F89 DEF-FN-4   JP    NC,1C8A,REPORT-C uma letra.
                EX    DE,HL       Guardar o indicador em DE.
                RST   0020,NEXT-CHAR Obter o caracter seguinte.
                CP    +24         Saltar para diante a menos
                JR    NZ,1F94,DEF-FN-5 que seja um «$».
                EX    DE,HL       Senão, guardar o novo indica-
                                  dor em DE.
                RST   0020,NEXT-CHAR Obter caracter seguinte.
1F94 DEF-FN-5   EX    DE,HL       Deslocar o indicador do
                                  último caracter do nome
                                  para o par de registos HL.
                LD    BC,+0006    Reservar 6 posições após
                CALL  1655,MAKE-ROOM esse último caracter e
                INC   HL          colocar um «marcador de nú-
```

```
                INC   HL              mero» na primeira das novas
                LD    (HL),+0E        posições.
                CP    +2C             Se o caracter actual é um
                JR    NZ,1FA6,DEF-FN-6 »,«, saltar atrás porque de-
                RST   0020,NEXT-CHAR  veria existir um outro parâ-
                JR    1F86,DEF-FN-3   metro; senão, sair do
                                      ciclo.
```

Em seguida, considera-se a definição da função.

```
1FA6 DEF-FN-6   CP    +29             Verificar se existe o
                JR    NZ,1FBD,DEF-FN-7 caracter »)«.
                RST   0020,NEXT-CHAR  Obter o caracter seguinte.
                CP    +3D             Deve ser um »=«.
                JR    NZ,1FBD,DEF-FN-7
                RST   0020,NEXT-CHAR  Obter o caracter seguinte.
                LD    A,(FLAGS)       Guardar a natureza — número
                PUSH  AF              ou cadeia — da variável.
                CALL  24FB,SCANNING   Considerar agora a definição
                                      como expressão.
                POP   AF              Obter a natureza da variável
                XOR   (FLAGS)         e verificar se é do mesmo
                AND   +40             tipo referenciado na
                                      definição.
1FBD DEF-FN-7   JP    NZ,1C8A,REPORT-C Produzir uma mensagem de
                                      erro se for caso disso.
                CALL  1BEE,CHECK-END  Sair por CHECK-END (passando
                                      portanto a considerar a
                                      declaração que se segue na
                                      linha).
```

A subrotina «UNSTACK-Z»

Esta subrotina é invocada em várias situações a fim de «sair cedo» de uma subrotina ao verificar sintaxe. A razão disto consiste em evitar a impressão de caracteres ou a passagem de valores entre o «stack» do calculador e outras posições.

```
1FC3 UNSTACK-Z  CALL  2530,SYNTAX-Z   Está a verificar sintaxe?
                POP   HL              Obter o endereço de retorno
                RET   Z               mas ignorá-lo ao ver sintaxe.
                JP    (HL)            Em execução, retorno simples
                                      à rotina original.
```

As rotinas de comando «LPRINT e PRINT»

É aberto o canal apropriado e consideram-se separadamente os elementos a imprimir.

```
1FC9 LPRINT     LD    A,+03           Preparar para abrir o canal »P«.
                JR    1FCF,PRINT-1    Saltar para diante.
1FCD PRINT      LD    A,+02           Preparar para abrir o canal »S«.
1FCF PRINT-1    CALL  2530,SYNTAX-Z   Abrir um canal a menos que se
                CALL  NZ,1601,CHAN-OPEN verifique sintaxe.
```



```
                CALL  0D4D,TEMPS      Definir as variáveis de sis-
                                      tema de cor temporária.
                CALL  1FDF,PRINT-2    Invocar a rotina de controlo
                                      da impressão.
                CALL  1BEE,CHECK-END  Passar a considerar a decla-
                RET                   ração seguinte; através de
                                      CHECK-END se verifica sintaxe.
```

A subrotina de controlo de impressão é indicada pelas subrotinas PRINT, LPRINT e INPUT.

```
1FDF PRINT-2    RST   0018,GET-CHAR   Obter o primeiro caracter.
                CALL  2045,PR-END-Z   Saltar para diante se já está
                JR    Z,1FF2,PRINT-4  no final da lista de elementos.
```

Entrar agora num ciclo que trate os «controladores de posição» e os elementos a imprimir.

```
1FE5 PRINT-3    CALL  204E,PR-POSN-1  Tratar quaisquer controlos de
                JR    Z,1FE5,PRINT-3  posição consecutivos.
                CALL  1FFC,PR-ITEM-1  Tratar um único elemento a imprimir.
                CALL  204E,PR-POSN-1  Verificar outros controlos de
                JR    Z,1FE5,PRINT-3  posição e imprimir elementos
                                      até não restar nenhum.
1FF2 PRINT-4    CP    +29             Retorno, agora, se o caracter
                RET   Z               actual é um »)«; senão,
                                      considerar a execução de um
                                      «retorno de linha».
```

A subrotina «Imprimir um retorno de linha»

```
1FF5 PRINT-CR   CALL  1FC3,UNSTACK-Z  Retorno se verifica sintaxe.
                LD    A,+0D           Imprimir um caracter de retorno
                RST   0010,PRINT-A-1  de linha, e retorno.
                RET
```

A subrotina «Imprimir elementos»

Esta subrotina é invocada pelas rotinas de PRINT, LPRINT e INPUT. Os diversos tipos de elemento a imprimir são identificados e impressos.

```
1FFC PR-ITEM-1  RST   0018,GET-CHAR   Obtém o 1.º caracter.
                CP    +AC             Saltar para diante a menos
                JR    NZ,200E,PR-ITEM-2 que seja um »AT«.
```

Tratar agora um »AT«.

```
                CALL  1C79,NEXT-2NUM  Os 2 parâmetros são transferi-
                                      dos para o »stack« do calculador.
                CALL  1FC3,UNSTACK-Z  Retorno se se verifica sintaxe.
                CALL  2307,STK-TO-BC  Os parâmetros são comprimidos
                                      para o par de registos BC.
                LD    A,+16           Carrega no registo A o carac-
                JR    201E,PR-AT-TAB  ter de controlo AT antes de
                                      executar o salto.
```

Procurar em seguida um TAB.

```
200E PR-ITEM-2   CP    +AD                  Saltar para diante a menos que
                 JR    NZ,2024,PR-ITEM-3    seja «TAB».
```

Tratar agora um TAB.

```
                 RST   0020,NEXT-CHAR       Obter o caracter seguinte.
                 CALL  1C82,EXPT-1NUM       Transferir um parâmetro para
                                            o «stack» do calculador.
                 CALL  1FC3,UNSTACK-Z       Retorno se verifica sintaxe.
                 CALL  1E99,FIND-INT2       O valor é comprimido no par
                                            de registos BC.
                 LD    A,+17                O registo A recebe o caracter de controlo TAB.
```

Os elementos de impressão AT e TAB são impressos invocando três vezes PRINT-OUT.

```
201E PR-AT-TAB   RST   0010,PRINT-A-1       Imprimir o caracter de controlo.
                 LD    A,C                  Segui-lo do primeiro valor.
                 RST   0010,PRINT-A-1
                 LD    A,B                  Finalmente, imprimir o segundo
                 RST   0010,PRINT-A-1       valor; e retorno.
                 RET
```

Considerar agora os elementos de cor.

```
2024 PR-ITEM-3   CALL  21F2,CO-TEMP-3       Retorno com a «carry» a zero
                                            se encontra elementos de cor.
                 RET   NC                   Continua, no caso contrário.
                 CALL  2070,STR-ALTER       Considerar se o «stream» deve
                                            ser alterado.
                 RET   NC                   Continuar, se não o foi.
```

O elemento a imprimir deve ser, a partir de agora, uma expressão, numérica ou de cadeia.

```
                 CALL  24FB,SCANNING        Avaliar a expressão, mas
                 CALL  1FC3,UNSTACK-Z       retorno se verifica sintaxe.
                 BIT   6,(FLAGS)            Verificar a natureza da
                                            expressão.
                 CALL  Z,2BF1,STK-FETCH     Se é cadeia, obter os parâmetros necessários; mas se é
                 JP    NZ,2DE3,PRINT-FP     numérica, sair por PRINT-FP.
```

Realiza-se agora um ciclo que trata separadamente cada um dos caracteres da cadeia.

```
203C PR-STRING   LD    A,B                  Retorno se não restam caracteres na cadeia; no caso
                 OR    C                    contrário, diminuir o
                 DEC   BC                   contador.
                 RET   Z                    Obter o código e incrementar
                 LD    A,(DE)               o indicador.
                 INC   DE                   Imprime o código, executando
                 RST   0010,PRINT-A-1       um salto para considerar
                 JR    203C,PR-STRING       quaisquer outros caracteres.
```


A subrotina «Final de impressão»

A flag «zero» passará a um se não houver nada mais para imprimir.

```
2045 PR-END-Z    CP    +29                  Retorno, se o caracter é
                 RET   Z                    um «)».
2048 PR-ST-END   CP    +0D                  Retorno, se o caracter é um
                 RET   Z                    «retorno de linha».
                 CP    +3A                  Comparar finalmente com «:»
                 RET                        antes do retorno.
```

A subrotina «Posição de impressão»

Esta subrotina considera os vários caracteres de controlo de posição.

```
204E PR-POSN-1   RST   0018,GET-CHAR        Obter o caracter actual.
                 CP    +3B                  Saltar para diante se é
                 JR    Z,2067,PR-POSN-3     um «;».
                 CP    +2C                  Saltar também para diante para
                 JR    NZ,2061,PR-POSN-2    qualquer caracter excepto «,»;
                 CALL  2530,SYNTAX-Z        mas não imprimir o caracter
                 JR    Z,2067,PR-POSN-3     se verifica sintaxe.
                 LD    A,+06                Carregar no registo A o
                 RST   0010,PRINT-A-1       código de controlo «vírgula» e
                 JR    2067,PR-POSN-3       imprimi-lo; saltar para diante.
2061 PR-POSN-2   CP    +27                  É um «'»?
                 RET   NZ                   Retorno, se não é nenhum dos
                                            controlos de posição.
                 CALL  1FF5,PR-CR           Imprimir «retorno de caracter»
                                            se não verifica sintaxe.
2067 PR-POSN-3   RST   0020,NEXT-CHAR       Obter o caracter seguinte.
                 CALL  2045,PR-END-Z        Se não está no fim de uma decla-
                 JR    NZ,206E,PR-POSN-4    ração de impressão saltar para
                 POP   BC                   diante; no caso contrário, re-
206E PR-POSN-4   CP    A                    torno à rotina original.
                 RET                        A flag «zero» passará a zero
                                            se não foi atingido o final da
                                            declaração de impressão.
```

A subrotina «Alterar STREAM»

Este subrotina é invocada sempre que há necessidade de considerar se o utilizador deseja recorrer a um «stream» diferente.

```
2070 STR-ALTER   CP    +23                  Se o caracter actual não é
                 SCF                        um «#», retorno com a flag
                 RET   NZ                   «carry» a um.
                 RST   0020,NEXT-CHAR       Avançar CH-ADD.
                 CALL  1C82,EXPT-1NUM       Passar os parâmetros para o
                                            «stack» do calculador.
                 AND   A                    Limpar a flag «carry».
                 CALL  1FC3,UNSTACK-Z       Retorno se verifica sintaxe.
                 CALL  1E94,FIND-INT1       O valor é passado para o
                                            registo A.
```

```
         CP    +10          Produzir a mensagem O se o
         JP    NC,160E,REPORT-O  valor é superior a +FF.
         CALL  1601,CHAN-OPEN  Usar o canal para o stream em
                             questão.
         AND   A            Limpar a flag «carry» e
         RET                retorno.
```

A rotina de comando «INPUT»

Esta rotina permite que sejam atribuídos a variáveis, valores introduzidos por teclado. É igualmente possível incluir elementos para impressão na declaração INPUT, sendo estes elementos impressos na parte inferior do visor.

```
2089 INPUT      CALL  2530,SYNTAX-Z   Saltar para a frente se se
                JR    Z,2096,INPUT-1  verifica sintaxe.
                LD    A,+01         Abrir o canal «K».
                CALL  1601,CHAN-OPEN
                CALL  0D6E,CLS-LOWER  Limpar a janela inferior do
                                    visor.
2096 INPUT-1    LD    (DF-SZ),+01   Definir esta janela de modo
                                    a ter apenas uma linha.
                CALL  20C1,IN-ITEM-1  Invocar a subrotina para
                                    tratar os elementos de INPUT.
                CALL  1BEE,CHECK-END  Passar à declaração seguinte
                                    se verifica sintaxe.
                LD    BC,(S-POSN)   Obter a actual posição de im-
                                    pressão.
                LD    A,(DF-SZ)     Saltar para diante se a posi-
                CP    B             ção actual está acima da
                JR    C,20AD,INPUT-2  janela inferior.
                LD    C,+21         Senão, definir a posição de
                LD    B,A           impressão no topo desta janela.
20AD INPUT-2    LD    (S-POSN),BC   Redefinir S-POSN.
                LD    A,+19         Definir o contador de «scroll».
                SUB   B
                LD    (SCR-CT),A
                RES   0,(TV-FLAG)   Sinal «janela principal».
                CALL  0DD9,CL-SET   Definir as variáveis de sistema
                JP    0D6E,CLS-LOWER  e sair por CLS-LOWER.
```

O ciclo que se segue trata separadamente os elementos de INPUT e os elementos de impressão contidos nesta instrução.

```
20C1 IN-ITEM-1  CALL  204E,PR-POSN-1  Considerar primeiro quaisquer
                JR    Z,20C1,IN-ITEM-1  caracteres de controlo de posição.
                CP    +28           Saltar para diante se o caracter
                JR    NZ,20D8,IN-ITEM-2  seguinte não é um «(».
                RST   0020,NEXT-CHAR  Obter o caracter seguinte.
                CALL  1FDF,PRINT-2  Invocar agora a rotina de comando
                                    PRINT para tratar os elementos
                                    dentro de parêntesis.
                RST   0018,GET-CHAR   Obter o caracter actual.
                CP    +29           Produzir a mensagem C se o
                JP    NZ,1C8A,REPORT-C  caracter não é um «)».
                RST   0020,NEXT-CHAR  Obter o caracter seguinte e
                JP    2182,IN-NEXT-2  saltar para diante para
                                    verificar se existem outros
                                    elementos de INPUT.
```


Considerar agora se se está a usar INPUT LINE.

```
20D8 IN-ITEM-2  CP    +CA           Saltar para diante se não é
                JR    NZ,20ED,IN-ITEM-3  LINE.
                RST   0020,NEXT-CHAR  Avançar CH-ADD.
                CALL  1C1F,CLASS-01   Determinar o endereço
                                    de destino da variável.
                SET   7,(FLAGX)     Sinal «uso de INPUT LINE».
                BIT   6,(FLAGS)     Produzir mensagem C se não
                JP    NZ,1C8A,REPORT-C  é uma variável de cadeia.
                JR    20FA,IN-PROMPT  Saltar para diante para exe-
                                    cutar o pedido de entrada.
```

Continuar tratando variáveis simples de INPUT.

```
20ED IN-ITEM-3  CALL  2C8D,ALPHA    Saltar para considerar nova
                JP    NC,21AF-IN-NEXT-1  execução do ciclo se o carac-
                                    ter actual não é uma letra.
                CALL  1C1F,CLASS-01   Determinar o endereço de
                                    destino da variável.
                RES   7,(FLAGX)     Sinal «não INPUT LINE».
```

A mensagem que acompanha o pedido de entrada é agora construída na área de trabalho.

```
20FA IN-PROMPT  CALL  2530,SUNTAX-Z   Saltar para diante se
                JP    Z,2182,IN-NEXT-2  apenas verifica sintaxe.
                CALL  16BF,SET-WORK   A área de trabalho é limpa.
                LD    HL,+5C71      Isto é FLAGX.
                RES   6,(HL)        Sinal «resultado cadeia».
                SET   5,(HL)        Sinal «modo INPUT».
                LD    BC,+0001      Atribuir à mensagem de
                                    entrada só uma posição.
                BIT   7,(HL)        Salto para a frente se LINE.
                JR    NZ,211C,IN-PR-2
                LD    A,(FLAGS)     Salto para a frente se
                AND   +40           espera uma entrada numérica.
                JR    NZ,211A,IN-PR-1
                LD    C,+03         Uma entrada de cadeia necessita
                                    de três posições.
211A IN-PR-1    OR    (HL)          O bit 6 de FLAGX passará a
                LD    (HL),A        um para uma entrada numérica.
211C IN-PR-2    RST   0030,BC-SPACES  É reservado o número neces-
                                    sário de posições.
                LD    (HL),+0D      Um «retorno de linha» vai
                                    para a última posição.
                LD    A,C           Verificar o bit 6 do registo
                RRCA                C e saltar para diante se é
                RRCA                apenas necessária uma posição.
                JR    NC,2129,IN-PR-3
                LD    A,+22         Um caracter «aspas» vai para
                LD    (DE),A        a primeira e segunda
                DEC   HL            posições.
                LD    (HL),A
2129 IN-PR-3    LD    (K-CUR),HL    Pode agora guardar-se a
                                    posição do cursor.
```

No caso de INPUT LINE pode invocar-se o EDITOR sem qualquer outra preparação mas para outros tipos de INPUT deve ser alterado o «stack» de erro a fim de contrariar a indicação de erro.

```
                BIT   7,(FLAGX)          Saltar para diante no caso «INPUT
                JR    NZ,215E,IN-VAR-3   LINE».
                LD    HL,(CH-ADD)        Guardar o valor actual de
                PUSH  HL                 CH-ADD e ERR-SP no
                LD    HL,(ERR-SP)        «stack»-máquina.
                PUSH  HL
213A IN-VAR-1   LD    HL,+213A           Este será o «ponto de
                PUSH  HL                 retorno» no caso de erros.
                BIT   4,(FLAGS2)         Só alterar o indicador do
                JR    Z,2148,IN-VAR-2    «stack» de erros se se usa
                LD    (ERR-SP),SP        o canal «K».
2148 IN-VAR-2   LD    HL,(WORKSP)        Definir HL para o início da
                CALL  11A7,REMOVE-FP     linha INPUT e eliminar quaisquer
                                         representações em vírgula flu-
                                         tuante (não haverá nenhuma
                                         excepto após um erro).
                LD    (ERR-NR),+FF       Sinal «não há erro ainda».
                CALL  0F2C,EDITOR        Obter agora o INPUT e, com a
                RES   7,(FLAGS)          flag sintaxe/erro indicando
                CALL  21B9,IN-ASSIGN     sintaxe, verificar erros de
                JR    2161,IN-VAR-4      INPUT; salto se em ordem; se-
                                         não, retorno para IN-VAR-1.
215E IN-VAR-3   CALL  0F2C,EDITOR        Obter uma «LINE».
```

Todas as variáveis de sistema devem ser redefinidas antes de ser possível realizar a atribuição de um valor.

```
2161 IN-VAR-4   LD    (K-CUR-hi),+00     O endereço do cursor passa a 0.
                CALL  21D6,IN-CHAN-K     Salto no caso de se usar
                JR    NZ,2174,IN-VAR-5   um canal que não seja «K».
                CALL  111D,ED-COPY       A linha de entrada é copia-
                LD    BC,(ECHO-E)        da para o visor e ECHO-E
                CALL  0DD9,CL-SET        passa a posição actual na
                                         janela inferior.
2174 IN-VARS-5  LD    HL,+5C71           Isto é FLAGX.
                RES   5,(HL)             Sinal «modo 'edit'».
                BIT   7,(HL)             Saltar para diante se trata
                RES   7,(HL)             INPUT LINE.
                JR    NZ,219B,IN-VAR-6
                POP   HL                 Libertar endereço IN-VAR-1.
                POP   HL                 Pôr em ERR-SP o seu valor
                LD    (ERR-SP),HL        original.
                POP   HL                 Guardar o endereço CH-ADD
                LD    (X-PTR),HL         original em X-PTR.
                SET   7,(FLAGS)          Fazer a atribuição para a
                CALL  2189,IN-ASSIGN     flag sintaxe/execução indicando
                                         «execução».
                LD    HL,(X-PTR)         Restaurar o endereço original
                LD    (X-PTR-hi),+00     da CH-ADD e limpar X-PTR.
                LD    (CH-ADD),HL
                JR    2182,IN-NEXT-2     Saltar para diante para verificar
                                         se existem outros elementos INPUT.
219B IN-VARS-6  LD    HL,(STKBOT)        Determina o comprimento de «LINE»
                LD    DE,(WORKSP)        da área de trabalho.
```



```
                SCF
                SBC,  HL,DE
                LD    B,H                DE aponta para o índice e BC
                LD    C,L                guarda o comprimento.
                CALL  2AB2,STK-ST-$      Estes parâmetros são guardados,
                CALL  2AFF,LET           sendo feita a atribuição.
                JR    21B2,IN-NEXT-2     Saltar também para diante para
                                         considerar outros elementos.
```

São considerados outros elementos na declaração INPUT.

```
21AF IN-NEXT-1  CALL  1FFC,PR-ITEM-1     Tratar elementos de impressão.
21B2 IN-NEXT-2  CALL  204E,PR-POSN-1     Tratar controlos de posição.
                JP    Z,20C1,IN-ITEM-1   Percorrer de novo o ciclo se
                                         existem mais elementos;
                RET                      senão, retorno.
```

A subrotina «IN-ASSIGN»

Esta subrotina é invocada duas vezes para cada valor INPUT. Uma vez com a flag sintaxe/run ao nível zero (sintaxe) e outra com o valor um (execução).

```
21B9 IN-ASSIGN  LD    HL,(WORKSP)        Levar CH-ADD a apontar para
                LD    (CH-ADD),HL        a primeira posição da área
                RST   0018,GET-CHAR      de trabalho e obter o caracter.
                CP    +E2                É «STOP»?
                JR    Z,21D0,IN-STOP     Saltar, se sim.
                LD    A,(FLAGX)          Senão, atribuir o «valor»
                CALL  1C59,VAL-FET-2     à variável.
                RST   0018,GET-CHAR      Obter o caracter actual e
                CP    +0D                verificar se é «retorno de
                RET   Z                  linha». Retorno se sim.
```

Mensagem «C — Nonsense in Basic»

```
21CE REPORT-C   RST   0008,ERROR-1       Invocar rotina de tratamento
                DEFB  +0B                de erro.
```

Vir aqui se a linha de INPUT começa por «STOP».

```
21D0 IN-STOP    CALL  2530,SYNTAX-Z      Mas não produzir mensagem de
                RET   Z                  erro se verifica sintaxe.
```

Mensagem «H — STOP in INPUT»

```
21D4 REPORT-H   RST   0008,ERROR-1       Invocar rotina de tratamento
                DEFB  +10                de erro.
```

A subrotina «IN-CHAN-K»

Esta subrotina termina com a flag «zero» a zero apenas no caso de estar a ser usado o canal «K».

```
21D6 IN-CHAN-K  LD   HL,(CURCHL)   Obtém o endereço base da
                INC  HL            informação do canal actual,
                INC  HL            sendo o código do canal
                INC  HL            comparado com o caracter
                INC  HL            «K».
                LD   A,(HL)
                CP   +4B
                RET                Retorno em seguida.
```

As rotinas «Elementos de cor»

Este conjunto de rotinas pode ser facilmente dividido em duas partes:

1) O tratamento do elemento de cor explícito.
2) O tratamento da «variável de sistema de cor».

1) Os elementos de cor explícitos são tratados invocando a subrotina PRINT-OUT do modo apropriado.

Entra-se num ciclo para tratar cada elemento por sua vez. O ponto de entrada é CO-TEMP-2.

```
21E1 CO-TEMP-1  RST  0020,NEXT-CHAR   Considerar o caracter seguinte
                                      da declaração BASIC.
21E2 CO-TEMP-2  CALL 21F2,CO-TEMP-3   Saltar para diante se o código
                                      actual representa uma cor
                                      «temporária» explícita. Retorno
                RET  C                com flag «carry» em zero se não
                                      é um elemento de cor.
                RST  0018,GET-CHAR    Obter o caracter seguinte.
                CP   +2C              Saltar atrás se é um «,» ou um
                JR   Z,21E1,CO-TEMP-1 «;»; senão, ocorreu um
                CP   +3B              erro.
                JR   Z,21E1,CO-TEMP-1
                JP   1C8A,REPORT-C    Saída através de «mensagem C».
21F2 CO-TEMP-3  CP   +D9              Retorno com «carry» a um se
                RET  C                o código não está na gama +D9
                CP   +DF              a +DE (INK a OVER).
                CCF
                RET  C
                PUSH AF               O código do elemento de cor
                RST  0020,NEXT-CHAR   é preservado enquanto CH-ADD
                POP  AF               avança para endereçar o
                                      parâmetro que o segue.
```

O código do elemento de cor e o parâmetro são agora impressos invocando PRINT-OUT em duas ocasiões.

```
21FC CO-TEMP-4  SUB  +C9              A gama de palavras-chave (+D9 a
                                      +DE) é reduzida à dos caracteres
                                      de controlo (+10 a +15).
                PUSH AF               O código do caracter de controlo
                CALL 1CB2,EXPT-1NUM   é preservado enquanto o parâmetro
                POP  AF               é passado ao «stack» do calculador.
                AND  A                Retorno neste ponto se se
                CALL 1FC3,UNSTACK-Z   verifica sintaxe.
                PUSH AF               O código do caracter de controlo
                CALL 1E94,FIND-INT1   é preservado enquanto o pa-
                LD   D,A              râmetro passa para o registo D.
                POP  AF
```



```
                RST  0010,PRINT-A-1   O caracter de controlo é aqui
                                      enviado.
                LD   A,D              Depois, obtém-se o parâmetro
                RST  0010,PRINT-A-1   e envia-se este antes do
                RET                   retorno.
```

2) As variáveis de sistema da cor — ATTR-T, MASK-T e P-FLAG — são alteradas do modo requerido.

Esta subrotina é invocada por PRINT-OUT; no início, o código do caracter de controlo encontra-se no registo A e o parâmetro no registo D.

Note-se que todas as alterações afectam apenas as variáveis de sistema «temporárias».

```
2211 CO-TEMP-5  SUB  +11              Reduzir a gama e saltar para
                ADC  A,+00            a frente para INK e PAPER.
                JR   Z,2234,CO-TEMP-7
                SUB  +02              Reduzir a gama uma vez mais e
                ADC  A,+00            saltar para diante para
                JR   Z,2273,CO-TEMP-C FLASH e BRIGHT.
```

O código de controlo de cor será agora +01 para INVERSE e +02 para OVER, sendo a variável de sistema P-FLAG alterada de modo correspondente.

```
                CP   +01              Preparar para salto se OVER.
                LD   A,D              Obter o parâmetro.
                LD   B,+01            Preparar a máscara para OVER.
                JR   NZ,2228,CO-TEMP-6 Saltar agora.
                RLCA                  O bit 2 do registo A deve
                RLCA                  passar a zero se INVERSE 0, e a
                LD   B,+04            um para INVERSE 1; a máscara terá
                                      o bit 2 ao valor um.
2228 CO-TEMP-6  LD   C,A              Guardar o registo A enquanto
                                      é verificada a gama.
                LD   A,D              A gama correcta para INVERSE e
                CP   +02              OVER é apenas
                JR   NC,2244,REPORT-K «0 a 1».
                LD   A,C              Obter o registo A.
                LD   HL,+5C91         É P-FLAG que deve ser alte-
                                      rada.
                JR   226C,CO-CHANGE   Saída por CO-CHANGE e alterar
                                      P-FLAG usando «B» como máscara,
                                      isto é, bit 0 para OVER e bit 2
                                      para INVERSE.
```

PAPER e INK são tratados pela rotina seguinte. Na entrada, a flag «carry» está definida para INK.

```
2234 CO-TEMP-7  LD   A,D              Obter o parâmetro.
                LD   B,+07            Preparar a máscara para INK.
                JR   C,223E,CO-TEMP-8 Saltar para diante se INK.
                RLCA                  Multiplicar o parâmetro por
                RLCA                  oito para PAPER.
                RLCA
                LD   B,+38            Preparar a máscara para PAPER.
223E CO-TEMP-8  LD   C,A              Guardar o parâmetro no registo
                                      C enquanto é verificada a gama
                                      do parâmetro.
```

```
                LD      A,D             Obter o valor original.
                CP      +0A             Só permitir PAPER/INK
                JR      C, 2246,CO-TEMP-9  numa escala «0 a 9».
```

Mensagem «K — Invalid colour»

```
2244 REPORT     RST     0008,ERROR-1    Invocar a rotina de tratamento de
                DEFB    +13             erro.
```

Continuar a tratar PAPER e INK.

```
2246 CO-TEMP-9  LD      HL,+5C8F        Preparar para alterar ATTR-T,
                                        MASK-T e P-FLAG.
                CP      +08             Saltar para diante para PAPER/
                JR      C,2258,CO-TEMP-B  /INK 0 a 7.
                LD      A,(HL)          Obter o valor actual do
                JR      Z,2257,CO-TEMP-A  ATTR-T e usá-lo sem alterações, sal-
                                        tando para diante,
                                        para PAPER /INK «8».
                OR      B               Mas para PAPER/INK «9» as
                CPL                     cores de PAPER e INK devem
                AND     +24             ser preto ou branco.
                JR      Z,2257,CO-TEMP-A  Saltar para INK/PAPER preto;
                LD      A,B             mas continuar para INK/PAPER
                                        branco.
2257 CO-TEMP-A  LD      C,A             Passar o valor para o registo C.
```

Usam-se agora a máscara (B) e o valor (C) para alterar ATTR-T.

```
2258 CO-TEMP-B  LD      A,C             Deslocar o valor.
                CALL    226C,CO-CHANGE  Alterar agora ATTR-T conforme
                                        necessário.
```

Considera-se em seguida MASK-T.

```
                LD      A,+07           Os bits de MASK-T são «um»
                CP      D               apenas para PAPER/INK «8»
                SBC     A,A             ou «9».
                CALL    226C,CO-CHANGE  Alterar agora MASK-T como for ne-
                                        cessário.
```

Em seguida, considera-se P-FLAG.

```
                RLCA                    É construída a máscara
                RLCA                    apropriada no registo B,
                AND     +50             alterando os bits 4 e 6 como
                LD      B,A             necessário.
                LD      A,+08           Os bits de P-FLAG estão a
                CP      D               «um» apenas para PAPER/INK
                SBC     A,A             «9».
                                        Continuar para CO-CHANGE a
                                        fim de manipular P-FLAG.
```

A Subrotina «CO-CHANGE»

Esta subrotina é usada para «imprimir» numa variável de sistema a «natureza» dos bits no registo A. O registo B contém uma máscara que mostra quais os bits que devem ser copiados de A para (HL).


```
226C CO-CHANGE  XOR     (HL)            Os bits especificados na máscara
                AND     B               do registo B são altera-
                XOR     (HL)            dos, sendo o resultado usado
                LD      (HL),A          para construir a variável de
                                        sistema.
                INC     HL              Passar a endereçar a variável
                                        de sistema seguinte.
                LD      A,B             Retorno com a máscara no
                RET                     registo A.
```

FLASH e BRIGHT são tratados pela rotina seguinte.

```
2273 CO-TEMP-C  SBC     A,A             A flag «zero» estará a um
                                        para BRIGHT.
                LD      A,D             O parâmetro é obtido e ro-
                RRCA                    dado.
                LD      B,+80           Preparar a máscara para FLASH.
                JR      NZ,227D,CO-TEMP-D  Saltar para diante se FLASH.
                RRCA                    Rodar mais uma vez e preparar
                LD      B,+40           uma máscara para BRIGHT.
227D CO-TEMP-D  LD      C,A             Guardar o valor no registo C.
                LD      A,D             Obter o parâmetro e verificar
                CP      +08             a sua gama; só são permitidos
                JR      Z,2287,CO-TEMP-E  «0», «1» e «8».
                CP      +02
                JR      NC,2244,REPORT-K
```

Pode agora alterar-se a variável de sistema ATTR-T.

```
2287 CO-TEMP-E  LD      A,C             Obter o valor.
                LD      HL,+5C8F        Isto é ATTR-T.
                CALL    226C,CO-CHANGE  Alterar a variável de sistema.
```

Considera-se agora o valor em MASK-T.

```
                LD      A,C             O valor é de novo obtido.
                RRCA                    O bit passado a um (bit 3) de
                RRCA                    FLASH/BRIGHT «8» é passado para
                RRCA                    o bit 7 (para FLASH) ou bit 6
                                        (para BRIGHT).
                JR      226C,CO-CHANGE  Saída por CO-CHANGE.
```

A rotina de comando «BORDER»

O parâmetro do comando BORDER é usado com uma ordem OUT para alterar de facto a cor da margem. O parâmetro é depois guardado na variável de sistema BORDCR.

```
2294 BORDER     CALL    1E94,FIND-INT1  Obter parâmetro e verifi-
                CP      +08             car a sua gama.
                JR      NC,2244,REPORT-K
                OUT     (+FE),A         Usa-se então a instrução OUT
                                        para definir a cor de margem.
                RLCA                    O parâmetro é então multi-
                RLCA                    plicado por oito.
                RLCA
                BIT     5,A             Se a cor de margem é «clara»,
```

```
            JR      NZ,22A6,BORDER-1    a cor de INK na área de montagem
                                        será negra — executar
                                        o salto.
            XOR     +07                 Altera a cor de INK.
22A6 BORDER-1 LD    (BORDCR),A          Definir a variável de sistema
            RET                         como requerido e retorno.
```

A subrotina «Endereço de PIXEL»

Esta subrotina é invocada pela subrotina POINT e pela rotina do comando PLOT. No início, as coordenadas de um pixel encontram-se no par de registos BC, e a rotina coloca em HL o endereço do byte do ficheiro de imagem que contém esse pixel e aponta A para a posição do pixel no interior do byte.

```
22AA PIXEL-ADD  LD      A,+AF           Verificar se a coordenada (em
                SUB     B               B) não é maior do que 175.
                JP      C,24F9,REPORT-B
                LD      B,A             B contém agora 175 menos y.
                AND     A               A contém b7b6b5b4b3b2b1b0, o
                RRA                     byte de B. E agora
                                        0b7b6b5b4b3b2b1.
                SCF
                RRA                     Agora 10b7b6b5b4b3b2.
                AND     A
                RRA                     Agora 010b7b6b5b4b3.
                XOR     B
                AND     +F8             Finalmente, 010b7b6b2b1b0, de tal
                XOR     B               modo que H passa a 64+8*INT
                LD      H,A             (B/64)+B (mod 8), byte alto do
                LD      A,C             endereço do pixel. C contém X.
                RLCA                    A começa em c7c6c5c4c3c2c1c0.
                RLCA
                RLCA                    E agora é c2c1c0c7c6c5c4c3.
                XOR     B
                AND     +C7
                XOR     B               Agora é c2c1b5b4b3c5c4c3.
                RLCA
                RLCA                    Finalmente b5b4b3c7c6c5c4c3, de
                LD      L,A             tal modo que L passa a 32*INT (B
                LD      A,C             (mod 64)/8)+INT (x/8), byte baixo.
                AND     +07             A contém x(mod 8): o pixel é o
                RET                     bit (A-7) no interior do byte.
```

A subrotina «POINT»

Esta subrotina é invocada pela função POINT em SCANNING. É iniciada com as coordenadas de um pixel no «stack» do calculador, e devolve um último valor 1 se esse pixel for cor de INK, ou 0 se for cor de PAPER.

```
22CB POINT-SUB  CALL    2307,STK-TO-BC  Coordenadas Y para B, X para C.
                CALL    22AA,PIXEL-ADD  Endereço de pixel para HL.
                LD      B,A             B contará A+1 ciclos para
                INC     B               obter o bit desejado de (HL)
                LD      A,(HL)          na posição 0.
22D4 POINT-LP   RLCA                    Deslocamentos.
                DJNZ    22D4,POINT-LP
                AND     +01             O bit é 1 para tinta, 0 para papel.
                JP      2D28,STACK-A    É colocado no «stack» do calculador.
```


A rotina de comando «PLOT»

Esta rotina consiste numa subrotina principal além de uma linha que a invoca e outra que dela sai. A rotina principal é usada duas vezes por CIRCLE, e a subrotina é invocada por DRAW. Entra-se na rotina com as coordenadas de um pixel no «stack» do calculador. Descobre o endereço desse pixel e marca-o, tendo em conta o estatuto de INVERSE e OVER, guardado em P-FLAG.

```
22DC PLOT       CALL    2307,STK-TO-BC  Coordenada Y para B, X para C.
                CALL    22E5,PLOT-SUB   Invoca a subrotina.
                JP      0D4D,TEMPS      Sair, definindo cores temporárias.
22E5 PLOT-SUB   LD      (COORDS),BC     Define a variável de sistema.
                CALL    22AA,PIXEL-ADD  Endereço do pixel para HL.
                LD      B,A             B contará A+1 ciclos para
                INC     B               colocar um 0 na posição certa em A.
                LD      A,+FE           Introduz o zero.
22F0 PLOT-LOOP  RRCA                    Alinha pela posição do bit do
                DJNZ    22F0,PLOT-LOOP  pixel no byte.
                LD      B,A             Copia para B.
                LD      A,(HL)          Obtém o pixel-byte em A.
                LD      C,(P-FLAG)      Obtém P-FLAG, comparando
                BIT     0,C             primeiro com OVER.
                JR      NZ,22FD,PL-TST-IN Salta se OVER 1.
                AND     B               OVER 0 começa por passar
                                        o pixel a zero.
22FD PL-TST-IN  BIT     2,C             Verificar INVERSE.
                JR      NZ,2303,PLOT-END INVERSE 1 deixa o pixel como
                                        estava (OVER 1) ou em
                                        zero (OVER 0).
                XOR     B               INVERSE 0 deixa o pixel
                CPL                     complementado (OVER 1) ou
                                        em 1 (OVER 0).
2303 PLOT-END   LD      (HL),A          Introduz o byte. Os seus
                                        outros bits não são alterados.
                JP      0BDB,PO-ATTR    Sair, definindo byte de atributos.
```

A subrotina «STK-TO-BC»

Esta subrotina carrega dois números em vírgula flutuante para o par de registos BC. É portanto usada para recolher parâmetros na gama +00 a +FF. Obtém, igualmente, em DE, os valores de «deslocamento em diagonal» (±1, ±1) que são usados na subrotina de desenho a traço de DRAW.

```
2307 STK-TO-BC  CALL    2314,STK-TO-A   Primeiro número em A.
                LD      B,A             Daí para B.
                PUSH    BC              Guardar temporariamente.
                CALL    2314,STK-TO-A   Segundo número em A.
                LD      E,C             O seu indicador de sinal para E.
                POP     BC              Restaurar primeiro número.
```

```
        LD      D,C             O seu indicador de sinal para D.
        LD      C,A             Segundo número para C.
        RET                     BC, DE são os pretendidos.
```

A subrotina «STK-TO-A»

Esta subrotina carrega no registo A o número em vírgula flutuante guardado no topo do «stack» do calculador. O número deve encontrar-se na gama 00 a FF.

```
2314 STK-TO-A   CALL    2DD5,FP-TO-A    Módulo do último valor arredondado
                JP      C,24F9,REPORT-B para A se possível; senão, erro.
                LD      C,+01           Um para C se último valor positivo.
                RET     Z               Retorno, se o valor é positivo.
                LD      C,+FF           Senão, passar C para +FF (- 1).
                RET                     Fim.
```

A rotina de comando «CIRCLE»

Esta rotina desenha um círculo aproximado com centro nas coordenadas X e Y e raio Z. Estes números são arredondados para o inteiro mais próximo antes de serem usados. Assim, Z deve ser menos de 87,5, mesmo quando (X, Y) se encontra no centro do visor. O método usado consiste em desenhar uma série de arcos representados aproximadamente por linhas rectas. No apêndice a esta obra ilustra-se o método em Basic. Segue-se aqui a notação desse programa.

CIRCLE possui quatro partes:

1. Verifica o raio. Se o seu módulo é inferior a um, limita-se a «PLOT» X, Y.
2. Invoca CD-PRMS1 em 2470-24B6, usada para os parâmetros iniciais tanto de CIRCLE como de DRAW;
3. Define os parâmetros restantes de CIRCLE, incluindo o deslocamento inicial para o primeiro «arco» (de facto uma linha recta);
4. Salta para DRAW a fim de usar o ciclo que desenha os arcos em 2420-24FA.

As partes 1 a 4 serão agora explicadas separadamente.

1. 2320-23AA. O raio, por exemplo Z', é obtido no «stack» do calculador. É formado o seu módulo Z, que será usado daqui em diante. Se Z for inferior a um, é eliminado do «stack», sendo marcado o ponto X, Y por salto para PLOT.

```
2320 CIRCLE     RST     0018,GET-CHAR   Obter o caracter actual.
                CP      +2C             Verificar se é vírgula.
                JP      NZ,1C8A,REPORT-C Se não, indicar um erro.
                RST     0020,NEXT-CHAR  Obter caracter seguinte (raio).
                CALL    1C82,EXPT-1NUM  Raio para o «stack» do calculador.
                CALL    1BEE,CHECK-END  Passar a considerar a declaração
                                        seguinte, se verifica sintaxe.
```



```
                RST     0028,FP-CALC    Usa o calculador; o «stack»
                DEFB    +2A,abs         contém X, Y, Z.
                DEFB    +3D,re-stack»   Z é de novo passado ao «stack»; dis-
                DEFB    +38,fim-calc    pomos assim do seu expoente.
                LD      A,(HL)          Obter expoente do raio.
                CP      +81             Verificar se raio menor que um.
                JR      NC,233B,C-R-GRE-1 Se não, salto.
                RST     0028,FP-CALC    Se sim, eliminar do «stack».
                DEFB    +02,apagar      O «stack» contém X, Y.
                DEFB    +38,fim-calc
                JR      22DC,PLOT       Marcar o ponto X, Y.
```

2. 233B-2346 e invocar CD-PRMS1. Guarda-se 2•PI em mem-5 e invoca-se CD-PRMS1. Esta subrotina guarda no registo B o número de arcos requerido para o círculo, a saber, A=4•INT(PI•SQR Z/4)+4, donde 4, 8, 12, etc., até um máximo de 32. Guarda também em mem-0 a mem-4 as quantidades 2•PI/A, SIN(PI/A), 0, COS (2•PI/A) e SIN (2•PI/A).

```
                RST     0028,FP-CALC
                DEFB    +A3,stk-pi/2    X, Y, Z, PI/2.
                DEFB    +38,fim-calc    Aumentar agora expoente para
                LD      (HL),+83        83 hex, passando PI/2 a 2•PI.
                RST     0028,FP-CALC    X, Y, Z, 2•PI.
                DEFB    +C5,st-mem-5    (2•PI é copiado para mem-5).
                DEFB    +02,apagar      X, Y, Z.
                DEFB    +38,fim-calc
                CALL    247D,CD-PRMS1   Definir parâmetros iniciais.
```

3. 2347-2381: parâmetros restantes e salto para DRAW. É verificado se o comprimento inicial do arco é inferior a um. Se é, realiza um salto apenas para marcar o ponto X, Y. Se não, define os parâmetros: X+Z e Y—Z•SIN(PI/A) são guardados duas vezes como ponto inicial e final, e copiados também para COORDS; zero e 2•Z•SIN(PI/A) são guardados em mem-1 e mem-2 como incrementos iniciais, dando como primeiro «arco» a linha recta vertical que liga X+Z, Y—Z•SIN (PI/A) e X+Z, Y+Z•SIN(PI/A). O ciclo de desenho do arco usado em DRAW garante que todos os outros pontos se mantêm no mesmo círculo que estes dois pontos, com um incremento angular de 2•PI/A. Mas é óbvio que estes dois pontos subtendem de facto este ângulo no ponto X+Z•(1-COS (PI/A)), Y e não em X, Y. Nestas condições, os pontos finais de cada arco de círculo são deslocados para a direita de 2•(1-COS (PI/A), que é menos de meio pixel, e arredonda no máximo para um pixel.

```
2347            PUSH    BC              Guardar a contagem de arcos em B.
                RST     0028,FP-CALC    X, Y, Z.
                DEFB    +31,copiar      X, Y, Z, Z.
                DEFB    +E1,obter-mem-1 X, Y, Z, Z, SIN (PI/A)
                DEFB    +04,multiplicar X, Y, Z, Z•SIN(PI/A)
                DEFB    +38,fim-calc    SIN (PI/A) é metade do com-
                LD      A,(HL)          primento inicial do arco; ve-
                CP      +80             rifica-se se é menos de 0,5.
                JR      NC,235A,C-ARC-GE1 Se não, executar salto.
                RST     0028,FP-CALC    De outro modo, Z é eliminado
                DEFB    +02,apagar      do «stack», junto com meio-arco;
                DEFB    +02,apagar      o «stack»-máquina é limpo; e
                DEFB    +38,fim-calc.   salta para marcar X, Y.
                POP     BC
```

```
                JP      22DC,PLOT
235A C-ARC-GE1  RST     0028,FP-CALC        X, Y, Z, Z*SIN (PI/A).
                DEFB    +C2,st-mem-2        (Z*SIN (PI/A) para mem-2 por
                                            agora).
                DEFB    +01,troca           X, Y, Z*SIN (PI/A), Z
                DEFB    +C0,st-mem-0        X, Y, Z*SIN (PI/A), Z
                DEFB    +02,apagar          X, Y, Z*SIN (PI/A)
                DEFB    +03,subtrair        X, Y - Z*SIN (PI/A)
                DEFB    +01,troca           X, Y, Z*SIN (PI/A)
                DEFB    +E0,obter-mem-0     Y - Z*SIN (PI/A), X, Z
                DEFB    +0F,soma            Y - Z*SIN (PI/A), X+Z
                DEFB    +C0,st-mem-0        (X+Z é copiado para mem-0)
                DEFB    +01,troca           X+Z, Y - Z*SIN (PI/A)
                DEFB    +31,copiar          X+Z, Y - Z*SIN (PI/A),
                                            Y - Z*SIN (PI/A)
                DEFB    +E0,obter-mem-0     sa, sb, sb, sa
                DEFB    +01,troca           sa, sb, sa, sb
                DEFB    +31,copiar          sa, sb, sa, sb, sb
                DEFN    +E0,obter-mem-0     sa, sb, sa, sb, sb, sa
                DEFB    +A0,stk-zero        sa, sb, sa, sb, sb, sa, 0
                DEFB    +C1,st-mem-1        (mem-1 passa a zero)
                DEFB    +02,apagar          sa, sb, sa, sb, sb, sa
                DEFB    +38,fim-calc
```

Aqui **sa** indica X+Z, e **sb** indica Y - Z*SIN (PI/A).

```
                INC     (mem-2-1°)          Incrementa o byte expoente
                                            de mem-2, passa mem-2 para
                                            2*Z*SIN (PI/A).
                CALL    1E94,FIND-INT1      O último valor X+Z passa
                LD      L,A                 do «stack» para A e é copiado
                                            para L.
                PUSH    HL                  Guardado em HL.
                CALL    1E94,FIND-INT1      Y - Z*SIN (PI/A) vai do «stack»
                POP     HL                  para A e é copiado para H.
                LD      H,A                 HL contém agora o ponto inicial.
                LD      (COORDS),HL         É copiado para COORDS.
                POP     BC                  Restaura contagem de arcos.
                JP      2420,DRW-STEPS      Salto para DRAW.
```

O «stack» contém agora X+Z, Y - Z*SIN (PI/A), Y - Z*SIN (PI/A), X+Z.

A rotina de comando «DRAW»

Entra-se nesta rotina com as coordenadas de um ponto X0, Y0, por exemplo, em COORDS. Se são dados apenas dois parâmetros X, Y na ordem DRAW, esta desenha uma aproximação de uma linha recta a partir do ponto X0, Y0 para X0+X, Y0+Y. Se for fornecido um terceiro parâmetro, G, desenha uma aproximação de um arco de círculo a partir de X0, Y0 até X0+X, Y0+Y rodando no sentido contrário ao dos ponteiros do relógio um ângulo de G radianos.

A rotina possui quatro partes:

1. Desenha simplesmente uma linha se só recebe dois parâmetros ou se o diâmetro do círculo implícito é inferior a um;

2. Invoca CD-PRMS1 em 247D-24B6 para definir os primeiros parâmetros;
3. Define os parâmetros restantes, incluindo os deslocamentos iniciais do primeiro arco;
4. Entra no ciclo de desenho do arco, representando-o sob a forma de uma série de arcos menores definidos por linhas rectas, invocando a subrotina de desenho de traços em 24B7-24FA.

A rotina principal é seguida de duas subrotinas, CD-PRMS1 e DRAW-LINE. As quatro partes acima citadas na rotina principal serão tratadas separadamente.

1. Se apenas existem dois parâmetros, é realizado um salto para LINE-DRAW em 2477. É igualmente desenhada uma linha se a quantidade Z=(ABS X + ABS Y) / ABS SIN (G/2) for inferior a um. Z encontra-se entre 1 e 1,5 vezes o diâmetro do círculo implícito. Nesta secção, mem-0 passa a conter SIN (G/2), mem-1 Y, e mem-5 G.

```
2382 DRAW       RST     0018,GET-CHAR       Obter o caracter actual.
                CP      +2C                 Se é uma vírgula,
                JR      Z,238D,DR-3-PRMS    saltar.
                CALL    1BEE,CHECK-END      Passar à instrução seguinte
                                            se verifica sintaxe.
                JP      2477,LINE-DRAW      Salto para desenhar a linha.
238D DR-3-PRMS  RST     0020,NEXT-CHAR      Obter caracter seguinte (ângulo).
                CALL    1C82,EXPT-1NUM      Ângulo para «stack» do calcula-
                                            dor.
                CALL    1BEE,CHECK-END      Passar à instrução seguinte se
                                            verifica sintaxe.
                RST     0028,FP-CALC        X, Y, G no «stack».
                DEFB    +C5,st-mem-5        (G é copiado para mem-5)
                DEFB    +A2,stk-meio        X, Y, G, «0,5»
                DEFB    +04,multiplicar     X, Y, G/2
                DEFB    +1F,seno            X, Y, SIN (G/2)
                DEFB    +31,copiar          X, Y, SIN (G/2), SIN (G/2)
                DEFB    +30,negação         X, Y, SIN (G/2), (0/1)
                DEFB    +30,negação         X, Y, SIN (G/2), (1/0)
                DEFB    +00,salto-verdade   X, Y, SIN (G/2)
                DEFB    +06,para DR-SIN-NZ  (Se SIN (G/2)=0, isto é, G=
                DEFB    +02,apagar          2*PI*N, desenha uma recta).
                DEFB    +38, fim-calc       X, Y.
                JP      2477,LINE-DRAW      Linha X0, Y0 para X0+X, Y0+Y.
23A3 DR-SIN-NZ  DEFB    +C0,st-mem-0        (SIN G/2 copiado para mem-0).
                DEFB    +02,apagar          X, Y estão agora no «stack».
                DEFB    +C1,st-mem-1        (Y é copiado para mem-1).
                DEFB    +02,apagar          X
                DEFB    +31,copiar          X, X
                DEFB    +2A,abs             X, X' (X'=ABS X).
                DEFB    +E1,obter-mem-1     X, X', Y
                DEFB    +01,troca           X, Y, X'
                DEFB    +E1,obter-mem-1     X, Y, X', Y
                DEFB    +2A,abs             X, Y, X', Y' (Y'=ABS Y)
                DEFB    +0F,soma            X, Y, X'+Y'
                DEFB    +E0,obter-mem-0     X, Y, X'+Y', SIN (G/2)
                DEFB    +05,divisão         X, Y, (X'+Y')/SIN (G/2)=Z', p. ex.
                DEFB    +2A, abs            X, Y, Z (Z=ABS Z')
                DEFB    +E0,obter-mem-0     X, Y, Z, SIN (G/2)
```

```
DEFB   +01,troca          X, Y, SIN (G/2), Z
DEFB   +3D,re-«stack»     (Z passa de novo ao «stack» para
DEFB   +38,fim-calc       garantir a disponibilidade do
                          expoente)
LD     A,(HL)             Obter expoente de Z.
CP     +81                Se Z é maior ou igual
JR     NC,23C1,DR-PRMS    a 1, saltar.
RST    0028,FP-CALC       X, Y, SIN (G/2), Z
DEFB   +02,apagar         X, Y, SIN (G/2)
DEFB   +02,apagar         X, Y
DEFB   +38,fim-calc       Desenhar a recta de X0, Y0
JP     2477,LINE-DRAW     a X0+X, Y0+Y
```

2. Invoca CD-PRMS1. Esta subrotina guarda no registo B o número de arcos mais curtos requeridos para representar o arco completo, isto é, A=4•INT (G'•SQR Z/8)+4, onde G'=mod G, ou 252 se esta expressão excede 252 (como pode acontecer para uma corda extensa a um ângulo pequeno). A é, portanto, 4, 8, 12,... até 252. A subrotina guarda igualmente em mem-0 a mem-4 as quantidades G/A, SIN (G/2•A), 0, COS (G/A), SIN (G/A).

```
23C1  DR-PRMS     CALL  247D,CD-PRMS1     Invoca a subrotina.
```

3. Define o resto dos parâmetros do modo indicado em seguida. O «stack» conterá estes quatro elementos, lendo a partir da parte superior: X0+X e Y0+Y como final do último arco; em seguida, X0 e Y0 como início do primeiro arco. Mem-0 contém X0 e mem-5, Y0. Mem-1 e mem-2 conterão os deslocamentos iniciais do primeiro arco, U e V; e mem-3 e mem-4 conterão COS (G/A) e SIN (G/A) para uso no ciclo de desenho do arco.

As fórmulas de U e V podem ser explicadas do seguinte modo. Em vez de percorrer a corda final, de comprimento L, por exemplo, com deslocamentos X e Y, queremos percorrer uma corda inicial (que pode ser mais extensa) de comprimento L•W, onde W=SIN (G/2•A)/SIN (G/2), com deslocamentos X•W e Y•W, mas rodado de um ângulo — (G/2 — G/2•A), isto é, com os seguintes deslocamentos verdadeiros:

$$U=Y\cdot W\cdot \text{SIN}\ (G/2-G/2\cdot A)+X\cdot W\cdot \text{COS}\ (G/2-G/2\cdot A)$$
$$V=Y\cdot W\cdot \text{COS}\ (G/2-G/2\cdot A)-X\cdot W\cdot \text{SIN}\ (G/2-G/2\cdot A)$$

Estas fórmulas podem ser verificadas a partir de um diagrama, usando as expansões normais:

$$\text{COS}\ (P-Q)$$

e

$$\text{SIN}\ (P-Q),$$

onde $Q=G/2-G/2\cdot A$.

```
23C4   PUSH   BC              Guardar o contador de arcos em B.
       RST    0028,FP-CALC    X, Y, SIN (G/2), Z
       DEFB   +02,apagar      X,Y,SIN(G/2)
       DEFB   +E1,obter-mem-1 X,Y,SIN(G/2),SIN(G/2*A)
       DEFB   +01,trocar      X,Y,SIN(G/2*A),SIN(G/2)
       DEFB   +05,divisão     X,Y,SIN(G/2*A)/SIN(G/2)=W
       DEFB   +C1,st-mem-1    (W é copiado para mem-1)
       DEFB   +02,apagar      X,Y
```



```
DEFB   +01,trocar      Y,X
DEFB   +31,copiar      Y,X,X
DEFB   +E1,obter-mem-1 Y,X,X,W
DEFB   +04,multiplicar Y,X,X*W
DEFB   +C2,st-mem-2    (X•W é copiado para mem-2)
DEFB   +02,apagar      Y,X
DEFB   +01,trocar      X,Y
DEFB   +31,copiar      X,Y,Y
DEFB   +E1,obter-mem-1 X,Y,Y,W
DEFB   +04,multiplicar X,Y,Y*W
DEFB   +E2,obter-mem-2 X,Y,Y*W,X*W
DEFB   +E5,obter-mem-5 X,Y,Y*W,X*W,G
DEFB   +E0,obter-mem-0 X,Y,Y*W,X*W,G,G/A
DEFB   +03,subtrair    X,Y,Y*W,X*W,G — G/A
DEFB   +A2,stk-meio    X,Y,Y*W,X*W,G - G/A, ½
DEFB   +04,multiplicar X,Y,Y*W,X*W, G/2 - G/2*A=F
DEFB   +31,copiar      X,Y,Y*W,X*W, F, F
DEFB   +1F,seno        X,Y,Y*W,X*W, F, SIN F
DEFB   +C5,st-mem-5    (SIN F é copiado para mem-5).
DEFB   +02,apagar      X,Y,Y*W,X*W,F
DEFB   +20,co-seno     X,Y,Y*W,X*W, COS F
DEFB   +C0,st-mem-0    (COS F é copiado para mem-0).
DEFB   +02,apagar      X,Y,Y*W,X*W
DEFB   +C2,st-mem-2    (X•W é copiado para mem-2)
DEFB   +02,apagar      X,Y,Y*W
DEFB   +C1,st-mem-1    (Y•X é copiado para mem-1)
DEFB   +E5,obter-mem-5 X,Y,Y*W,SIN F
DEFB   +04 multiplicar X,Y,Y*W*SIN F
DEFB   +E0,obter-mem-0 X,Y,Y*W*SIN F,X*W
DEFB   +E2,obter-mem-2 X,Y,Y*W*SIN F,X*W, COS F
DEFB   +04,multiplicar X,Y,Y*W*SIN F,X*W*COS F
DEFB   +0F,somar       X,Y,Y*W*SIN F+X*WCOS F=U
DEFB   +E1,obter-mem-1 X,Y,U,Y*W
DEFB   +01,trocar      X,Y,Y*W,U
DEFB   +C1,st-mem-1    (U é copiado para mem-1).
DEFB   +02,apagar      X,Y,Y*W
DEFB   +E0,obter-mem-0 X,Y,Y*W, COS F
DEFB   +04,multiplicar X,Y,Y*W*COS F
DEFB   +E2,obter-mem-2 X,Y,Y*W*COS F,X*W
DEFB   +E5,obter-mem-5 X,Y,Y*W*COS F,X*W, SIN F
DEFB   +04,multiplicar X,Y,Y*W*COS F,X*W*SIN F
DEFB   +03,subtrair    X,Y,Y*W*COS F — X*W*SIN
                       F = V
DEFB   +C2,st-mem-2    (V é copiado para mem-2).
DEFB   +2A,abs         X, Y, V' (V' = ABS V)
DEFB   +E1,obter-mem-1 X, Y, V', U
DEFB   +2A,abs         X, Y, V', U' (U' = ABS U)
DEFB   +0F,somar       X, Y, U' + V'
DEFB   +02,apagar      X, Y
DEFB   +38,fim-calc    (DE aponta agora para U'+V').
LD     A,(DE)          Obter expoente de U'+V'.
CP     +81             Se U'+V' menor que 1, limpar
POP    BC              o «stack» e desenhar a linha
JP     C,2477,LINE-DRAW de X0, Y0 até X0+X, Y0+Y.
PUSH   BC              Senão, continuar com os parâ-
RST    0028,FP-CALC    metros X, Y no «stack».
DEFB   +01,trocar      Y, X
DEFB   +38,fim-calc
LD     A,(COORDS-baixo) Passar X0 para A e depois
CALL   2D28,STACK-A    para o «stack».
```

```
     RST    0028,FP-CALC      Y, X, X0
     DEFB   +C0,st-mem-0      (X0 é copiado para mem-0).
     DEFB   +0F,somar         Y, X0 + X
     DEFB   +01,trocar        X0+X, Y
     DEFB   +38,fim-calc
     LD     A,(COORDS-alto)   Passar Y0 para A e depois
     CALL   2D28,STACK-A      para o «stack».
     RST    0028,FP-CALC      X0+X, Y, Y0.
     DEFB   +C5,st-mem-5      (Y0 é copiado para mem-5).
     DEFB   +0F,somar         X0+X, Y0+Y
     DEFB   +E0,obter-mem-0   X0+X, Y0+Y, X0
     DEFB   +E5,obter-mem-5   X0+X, Y0+Y, X0, Y0
     DEFB   +38,fim-calc
     POP    BC                Restaura contador de arcos em B.
```

4. Ciclo de desenho do arco. Entra-se por 2439, com as coordenadas do ponto inicial no topo do «stack», e os deslocamentos iniciais para o primeiro arco em mem-1 e mem-2. Usa cálculos trigonométricos simples para garantir que todos os arcos subsequentes serão desenhados para pontos que se encontram no mesmo círculo que os dois primeiros, subtendendo o mesmo ângulo ao centro. Pode demonstrar-se que se 2 pontos X1, Y1 e X2, Y2 se encontram num círculo e subtendem um ângulo N no centro, que é também a origem das coordenadas, então X2=X1•COS N − Y1•SIN N, e Y2=X1•SIN N+Y1•COS N. Mas como a origem se encontra neste caso no canto inferior esquerdo do visor, o ciclo de desenho do arco aplica estas relações e incrementos, por exemplo, Un=Xn+1 − Xn e Vn=Yn+1 − Yn, conseguindo assim o resultado desejado. Mostra-se abaixo o «stack» na passagem (n+1) pelo ciclo, quando Xn e Yn são incrementados por Un e Vn, depois de estes serem obtidos a partir de Un − 1 e Vn − 1. Os quatro valores no topo do «stack» em 2425 são, em DRAW, e lendo para cima, X0+X, Y0+Y, Xn e Yn, mas para poupar espaços, estes não são apresentados até 2439. Quanto aos valores iniciais em CIRCLE, ver acima o final de CIRCLE. Ainda em CIRCLE, o ângulo G deve ser considerado 2 • PI.

```
2420 DRW-STEPS  DEC    B                 B conta as passagens pelo
                                         ciclo.
                JR     Z,245F,ARC-END    Salto quando B atinge zero.
                JR     2439,ARC-START    Salto para o ciclo no início
2425 ARC-LOOP   RST    0028,FP-CALC      (ver texto acima quanto ao
                                         «stack»).
                DEFB   +E1,obter-mem-1   Un-1
                DEFB   +31,copiar        Un-1,Un-1
                DEBF   +E3,obter-mem-3   Un-1,Un-1,COS(G/A)
                DEFB   +04,multiplicar   Un-1,Un-1*COS(G/A)
                DEFB   +E2,obter-mem-2   Un-1,Un-1*COS(G/A),Vn-1
                DEFB   +E4,obter-mem-4   Un-1,Un-1*COS(G/A),Vn-1,
                                         SIN(G/A)
                DEFB   +04,multiplicar   Un-1,Un-1*COS(G/A),Vn-1*
                                         SIN(G/A)
                DEFB   +03,subtrair      Un-1,Un-1*COS(G/A)-Vn-1*
                                         SIN(G/A)=Un
                DEFB   +C1,st-mem-1      (Un é copiado para mem-1).
                DEFB   +02,apagar        Un-1
                DEFB   +E4,obter-mem-4   Un-1,SIN(G/A)
                DEFB   +04,multiplicar   Un-1*SIN(G/A)
```



```
                DEFB   +E2,obter-mem-2   Un-1*SIN(G/A),Vn-1
                DEFB   +E3,obter-mem-3   Un-1*SIN(G/A),Vn-1,COS(G/A)
                DEFB   +04,multiplicar   Un-1*SIN(G/A),Vn-1*COS(G/A)
                DEFB   +0F,somar         Un-1*SIN(G/A)+Vn-1*COS
                                         (G/A)=Vn
                DEFB   +C2,st-mem-2      (Vn é copiado para mem-2).
                DEFB   +02,apagar        (Como se diz no texto, o
                DEFB   +38,fim-calc      «stack» contém, de facto, X0+X,
                                         Y0+Y, Xn e Yn).
2439 ARC-START  PUSH   BC                Guardar contador de arcos.
                RST    0028,FP-CALC      X0+X, Y0+Y, Xn, Yn
                DEFB   +C0,st-mem-0      (Yn é copiado para mem-0).
                DEFB   +02,apagar        X0+X, Y0+Y, Xn
                DEFB   +E1,obter-mem-1   X0+X, Y0+Y, Xn, Un
                DEFB   +0F,somar         X0+X, Y0+Y, Xn+Un = Xn+1
                DEFB   +31,copiar        X0+X, Y0+Y, Xn+1, Xn+1
                DEFB   +38,fim-calc      Depois de Xn', o valor aprox.
                                         de Xn é obtido pela subrotina
                                         de desenho de linhas e copiado
                LD     A,(COORDS-baixo)  para A e daí para o
                CALL   2D28,STACK-A      «stack».
                RST    0028,FP-CALC      X0+X,Y0+Y,Xn+1,Xn'
                DEFB   +03,subtrair      X0+X,Y0+Y,Xn+1,Xn+1,Xn'
                                         - Xn' = Un'
                DEFB   +E0,obter-mem-0   X0+X,Y0+Y,Xn+1,Un',Yn
                DEFB   +E2,obter-mem-2   X0+X,Y0+Y,Xn+1,Un',Yn,Vn
                DEFB   +0F,somar         X0+X,Y0+Y,Xn+1,Un',Yn +
                                         Vn = Yn+1
                DEFB   +C0,st-mem-0      (Yn+1 é copiado para mem-0).
                DEFB   +01,trocar        X0+X,Y0+Y,Xn+1,Yn+1,Un'
                DEFB   +E0,obter-mem-0   X0+X,Y0+Y,Xn+1,Yn+1,
                                         Un',Yn+1
                DEFB   +38,fim-calc
                LD     A,(COORDS-alto)   Yn' aproximado por Xn', é
                CALL   2D28,STACK-A      copiado para A e daí para o
                                         «stack».
                RST    0028,FP-CALC      X0+X,Y0+Y,Xn+1,Yn+1,
                                         Un',Yn+1,Yn'
                DEFB   +03,subtrair      X0+X,Y0+Y,Xn+1,Yn+1,
                                         Un',Vn'
                DEFB   +38,fim-calc
                CALL   24B7,DRAW-LINE    É desenhado o arco seguinte.
                POP    BC                Restaura o contador de arcos.
                DJNZ   2425,ARC-LOOP     Salto se desenha mais arcos.
245F ARC-END    RST    0028,FP-CALC      As coordenadas do final do
                DEFB   +02,apagar        último arco desenhado são
                DEFB   +02,apagar        eliminadas do «stack».
                DEFB   +01,trocar        Y0+Y, X0+X
                DEFB   +38,fim-calc
                LD     A,(COORDS-baixo)  A coordenada X do final do
                CALL   2D28,STACK-A      último arco desenhado, p. ex.
                RST    0028,FP-CALC      Xz', é copiada para o «stack»
                DEFB   +03,subtrair      Y0+Y, X0+X - Xz'
                DEFB   +01,trocar        X0+X - Xz', Y0+Y
                DEFB   +38,fim-calc
                LD     A,(COORDS-alto)   Obtém a coordenada Y.
                CALL   2D28,STACK-A
                RST    0028,FP-CALC      X0+X - Xz', Y0+Y, Yz'
                DEFB   +03,subtrair      X0+X - Xz', Y0+Y - Yz'
                DEFB   +38,fim-calc
```

```
2477 LINE-DRAW   CALL  2487,DRAW-LINE   Desenha o último arco para
                                        atingir X0+X, Y0+Y (ou fechar
                                        o círculo).
                 JP    0D4D,TEMPS      Sair, definindo cores temporárias.
```

A subrotina «Parâmetros iniciais»

Esta subrotina é invocada tanto por CIRCLE como por DRAW para definir os parâmetros iniciais. É invocada por CIRCLE com X, Y e o raio Z no topo do «stack», lendo para cima. É invocada por DRAW com as suas próprias X, Y, SIN (G/2) e Z, como definido acima, no topo do «stack». No que se segue o «stack» só é apresentado de Z para cima. A subrotina fornece em B a contagem de arcos em A, como se explica em CIRCLE e DRAW acima, e em mem-0 a mem-5 as quantidades G/A, SIN (G/2•A), 0, COS (G/A), SIN (G/A) e G. No caso de um círculo, G deve ser considerado como igual a 2•PI.

```
247D CD-PRMS1   RST   0028,FP-CALC      Z
                DEFB  +31,copiar        Z, Z
                DEFB  +28,sqr           Z, SQR Z
                DEFB  +34,stk-dados     Z, SQR Z, 2
                DEFB  +32,expoente+82
                DEFB  +00,(+00,+00,+00)
                DEFB  +01,trocar        Z, 2, SQR Z
                DEFB  +05,divisão       Z, 2/SQR Z
                DEFB  +E5,obter-mem-5   Z, 2/SQR Z, G
                DEFB  +01,trocar        Z, G, 2/SQR Z
                DEFB  +05,divisão       Z, G*SQR Z/2
                DEFB  +2A,abs           Z, G'*SQR Z/2 (G' = mod G)
                DEFB  +38,fim-calc      Z, G'*SQR Z/2 = A1, say
                CALL  2DD5,FP-TO-A      A1 para A do «stack», se
                                        possível.
                JR    C,2495,USE-252    Se A1 arredonda para 256 ou
                                        mais, usar 252.
                AND   +FC               4•INT(A1/4) para A.
                ADD   A,+04             Somar 4, dando a contagem de
                                        arcos A.
                JR    NC,2497,DRAW-SAVE Saltar, se for ainda menos de
                                        256.
2495 USE-252    LD    A,+FC             Aqui, usar apenas 252 decimal.
                PUSH  AF                Agora guardar contagem de ar-
                                        cos.
                CALL  2D28,STACK-A      Copiar também para o «stack» do
                                        calculador.
                RST   0028,FP-CALC      Z, A
                DEFB  +E5, obter-mem-5  Z, A, G
                DEFB  +01,trocar        Z, G, A
                DEFB  +05,dividir       Z, G/A
                DEFB  +31,copiar        Z,G/A, G/A
                DEFB  +1F,seno          Z, G/A, SIN (G/A)
                DEFB  +C4,st-mem-4      (SIN (G/A) é copiado para mem-4).
                DEFB  +02,apagar        Z, G/A
                DEFB  +31,copiar        Z, G/A, G/A
                DEFB  +A2,stk-meio      Z, G/A, G/A, 0.5
                DEFB  +04,multiplicar   Z, G/A, G/2*A
```



```
                DEFB  +1F,seno          Z, G/A, SIN (G/2*A)
                DEFB  +C1,st-mem-1      (SIN (G/2•A é copiado para
                                        mem-1).
                DEFB  +01,trocar        Z, SIN (G/2*A), G/A
                DEFB  +C0,st-mem-0      (G/A é copiado para mem-0).
                DEFB  +02,apagar        Z, SIN (G/2*A) = S
                DEFB  +31,copiar        Z, S, S
                DEFB  +04,multiplicar   Z, S*S
                DEFB  +31,copiar        Z, S*S, S*S
                DEFB  +0F,somar         Z, 2*S*S
                DEFB  +A1,stk-um        Z, 2*S*S, 1
                DEFB  +03,subtrair      Z, 2*S*S - 1
                DEFB  +1B,negar         Z, 1 - 2*S*S = COS (G/A)
                DEFB  +C3,st-mem-3      (COS (G/A) é copiado para mem-4).
                DEFB  +02,apagar        Z
                DEFB  +38,fim-calc
                POP   BC                Restaurar a contagem de arcos
                                        em B.
                RET                     Final.
```

A subrotina de desenho de linhas

Esta subrotina é invocada por DRAW para desenhar uma linha recta aproximada a partir do ponto X0, Y0, guardado em COORDS até ao ponto X0+X, Y0+Y, onde os incrementos X e Y se encontram no topo do «stack» do calculador. A subrotina foi originalmente pensada para a ROM de 8K do ZX80 e do ZX81, e é descrita num programa Basic apresentado na página 121 do manual do ZX81. É igualmente ilustrada aqui no programa «Círculo» apresentado no apêndice.

O método consiste em recorrer a tantos passos horizontais ou verticais quantos os necessários para um conjunto básico de traços diagonais, usando um algoritmo que espaça os passos horizontais e verticais da forma mais regular possível.

```
2487 DRAW-LINE  CALL  2307,STK-TO-BC   ABS Y para B; ABS X para C;
                                       SGN Y para D; SGN X para E.
                LD    A,C              Salto se ABS X maior ou igual
                CP    B                a ABS Y, pelo que o menor vai
                JR    NC,24C4,DL-X-GE-Y para L, e o maior (depois)
                LD    L,C              para H.
                PUSH  DE               Guardar passo diagonal (±1, ±1)
                                       em DE.
                XOR   A                Inserir um passo vertical (±1,0)
                LD    E,A              em DE (D contém SGN Y).
                JR    24CB,DL-LARGER   Saltar para definir H.
24C4 DL-X-GE-Y  OR    C                Retorno se ABS X e ABS Y
                RET   Z                são ambos zero.
                LD    L,B              O menor (aqui ABS Y vai
                                       para L).
                LD    B,C              ABS X para B aqui.
                PUSH  DE               Guardar o passo diagonal.
                LD    D,+00            Passo horiz. (0, ±1) para DE.
24CB DL-LARGER  LD    H,B              Maior de ABS X, ABS Y para H.
```

O algoritmo inicia-se aqui. O maior de ABS X e ABS Y, digamos H, é colocado em A e reduzido a INT (H/2). Os passos horizontal ou vertical H — L e os passos diagonais L são obtidos do seguinte modo (sendo L o menor

da ABS X e ABS Y): L é somado a A; se A é igual ou excede H, é reduzido de H e obtém-se um passo diagonal; senão, obtém-se um passo horizontal ou vertical. Isto é repetido H vezes (B também guarda H). Note-se que, entretanto, os registos alternativos H' e L' são usados para guardar COORDS.

```
                LD      A,B         B para A tal como para H.
                RRA                 A começa em INT (H/2).
24CE D-L-LOOP   ADD     A,L         L é somado a A.
                JR      C,24D4,D-L-DIAG  Se 256, ou mais, salto —
                                    — passo diagonal.
                CP      H           Se A menos do que H, salto para
                JR      C,24DB,D-L-HR-VT  passo horizontal ou vertical.
24D4 D-L-DIAG   SUB     H           Reduzir A de H.
                LD      C,A         Restaurá-lo em C.
                EXX                 Usar os registos alternativos.
                POP     BC          Passo diagonal para B'C'.
                PUSH    BC          Guardar também.
                JR      24DF,D-L-STEP  Saltar para obter passo.
24DB D-L-HR-VT  LD      C,A         Guardar A (intacto) em C.
                PUSH    DE          Passo para o «stack».
                EXX                 Obter registos alternativos.
                POP     BC          Passo para B'C'.
24DF D-L-STEP   LD      HL,(COORDS) Obter o passo: primeiro,
                                    COORDS para H'L' como ponto
                                    inicial.
                LD      A,B         Passo Y de B' para A.
                ADD     A,H         Somar em H'.
                LD      B,A         Resultado para B'.
                LD      A,C         Agora passo X; será verificada
                INC     A           a sua gama (Y será testado em
                                    PLOT).
                ADD     A,L         Somar L' a C' em A, salto
                JR      C,24F7,D-L-RANGE  para melhor verificação.
                JR      Z,24F9,REPORT-B  Zero após «carry» 0 indica
                                    posição X —1, fora da gama.
24EC D-L-PLOT   DEC     A           Restaurar verdadeiro valor em A.
                LD      C,A         Valor para C' para marcar ponto.
                CALL    22E5,PLOT-SUB  Marcar o passo.
                EXX                 Restaurar registos normais.
                LD      A,C         C volta a A para continuar
                                    algoritmo.
                DJNZ    24CE,D-L-LOOP  Voltar atrás para B passos (ou
                                    seja, H passos).
                POP     DE          Limpar «stack»-máquina.
                RET                 Terminar.
24F7 D-L-RANGE  JR      Z,24EC,D-L-PLOT  Zero após «carry» 1 indica
                                    posição X 255, na gama.
```

Mensagem «B — Integer out of range»

```
24F9 REPORT-B   RST     0008,ERROR-1  Invocar a rotina de
                DEFB    +0A           tratamento de erro.
```


A subrotina «SCANNING»

Esta subrotina é usada para produzir uma avaliação do resultado da «expressão seguinte».

O resultado é devolvido como «último valor» do «stack» do calculador.

No caso de um resultado numérico, o último valor será o número verdadeiro em vírgula flutuante. No entanto, no caso de um resultado alfanumérico, o último valor consistirá num conjunto de parâmetros.

O primeiro dos cinco bytes não é especificado, o segundo e o terceiro contêm o endereço do início da cadeia e o quarto e o quinto o comprimento desta.

O bit 6 de FLAGS está a um no caso de um resultado numérico, e a zero no caso de uma cadeia.

Quando a expressão seguinte consiste num único operando, por exemplo . . . A . . . , . . . RND. . . , . . . A$ (4,3 TO 7) . . . , o último valor é apenas o valor que é obtido por avaliação do operando.

No entanto, quando a expressão seguinte contém uma função e um operando, por exemplo . . . CHR$ A . . . , . . . NOT A . . . SIN 1 . . . , o código de operação da função é guardado no «stack»-máquina até o último valor do operando ter sido calculado. Este último valor é então sujeito à operação apropriada para dar um novo último valor.

No caso de ser necessário realizar uma operação aritmética ou lógica, por exemplo . . . A+B. . . , A*B. . . , . . . A=B. . . , tanto o último valor do primeiro argumento como o código de operação devem ser guardados até o último valor do segundo argumento ter sido encontrado.

De facto, o cálculo do último valor do segundo argumento pode também envolver o armazenamento dos últimos valores e códigos de operação enquanto a operação é realizada.

Pode portanto mostrar-se que enquanto é avaliada uma expressão complexa, por exemplo . . . CHR$ (T+A — 26*INT ((T+A)/26)+65) . . . , é construída uma hierarquia de operações ainda a realizar, até ser atingido o ponto a partir do qual deve ser desmantelada a fim de produzir o último resultado final.

Cada código de operação tem associado a si um código de prioridade apropriado, e as operações de maior prioridade são sempre realizadas antes das de menor prioridade.

A subrotina começa com o registo A preparado para guardar o primeiro caracter da expressão, e um marcador de prioridade inicial — zero — colocado no «stack»-máquina.

```
24FB SCANNING   RST   0018,GET-CHAR     Obtém o primeiro caracter.
                LD    B,+00             Marcador de prioridade inicial.
                PUSH  BC                É posto no «stack».
24FF S-LOOP-1   LD    C,A               Ponto de entrada principal.
                LD    HL,+2596          Indexar a tabela de «scanning»
                CALL  16DC,INDEXER      com o código em C.
                LD    A,C               Restaurar o código em A.
                JP    NC,2684,S-ALPHNUM Saltar se o código não está na tabela.
                LD    B,+00             Usar a entrada descoberta na tabela
                LD    C,(HL)            para construir o endereço requerido
                ADD   HL,BC             em HL, e saltar para este.
                JP    (HL)
```

Seguem-se quatro subrotinas; são invocadas por rotinas da tabela de «scanning» de funções. A primeira, a subrotina de «scanning» de aspas, é usada por S-QUOTE para verificar se cada par de aspas inicial é acompanhado pelo correspondente par de aspas final.

```
250F S-QUOTE-S  CALL  0074,CH-ADD+1     Aponta para o caracter seguinte.
                INC   BC                Aumentar a contagem de compri-
                                        mento de 1.
                CP    +0D               É um retorno de linha?
                JP    Z,1C8A,REPORT-C   Mensagem de erro se apropriado.
                CP    +22               É outro «"»?
                JR    NZ,250F,S-QUOTE-S Salto atrás, se não é.
                CALL  0074,CH-ADD+1     Apontar para o caracter seguinte;
                CP    +22               passar flag «zero» a um se for «"».
                RET                     Terminar.
```

A subrotina seguinte, a de «scanning» de duas coordenadas, é chamada por S-SCREEN$, S-ATTR e S-POINT para garantir que as duas coordenadas requeridas são dadas na forma correcta.

```
2522 S-2-COORD  RST   0020,NEXT-CHAR    Obter o caracter seguinte.
                CP    +28               É um «(»?
                JR    NZ,252D,S-RPORT-C Mensagem de erro se não for.
                CALL  1C79,NEXT-2NUM    Coordenadas para «stack» do
                                        calculador.
                RST   0018,GET-CHAR     Obter o caracter actual.
                CP    +29               É um «)»?
252D S-RPORT-C  JP    NZ,1C8A,REPORT-C  Mensagem de erro, se não for.
```

A subrotina «SYNTAX-Z»

Neste ponto é interpolada a subrotina «SYNTAX-Z». É invocada 32 vezes, guardando apenas um byte de cada vez. Uma simples verificação do bit 7 de FLAGS, colocará em zero a flag «zero» durante a execução e em um durante a verificação de sintaxe.

```
2530 SYNTAX-Z   BIT   7,(FLAGS)         Verificar o bit 7 de FLAGS.
                RET                     Terminado.
```

A subrotina seguinte é a de «scanning» de SCREEN$, usada por S--SCREEN$ para descobrir o caracter que ocorre na linha **x**, coluna **y** do visor. Apenas procura o caracter apontado por CHARS.

Nota: Trata-se normalmente do caracter +20 (espaço) a +7F (©), se bem que o utilizador possa alterar CHARS para procurar qualquer outro caracter, incluindo gráficos por si definidos.

```
2535 S-SCRN$-S  CALL  2307,STK-TO-BC    x para C, y para B; 0<=x<=23
                LD    HL,(CHARS)        em decimal; 0<=y<=31 em decimal.
                LD    DE,+0100          CHARS mais 256 decimal produz
                ADD   HL,DE             HL apontando para conj. de
                                        caracteres.
                LD    A,C               x é copiado para A.
                RRCA                    Forma em A e copia para E o
                RRCA                    número 32 (decimal)+(x mod 8)+
                                        +y.
                RRCA                    É o byte baixo do endereço
                AND   +E0               requerido no visor.
                XOR   B
                LD    E,A
                LD    A,C               x é copiado de novo para A.
                AND   +18               Insere o número 64 (decimal)+
                XOR   +40               +8*INT (x/8) em D.
                LD    D,A               DE contém o endereço do visor.
                LD    B,+60             B conta 96 caracteres.
25AF S-SCRN-LP  PUSH  BC                Guarda a contagem.
                PUSH  DE                E o indicador de visor.
                PUSH  HL                E o indicador do conjunto de
                                        caracteres.
                LD    A,(DE)            Obtém primeira linha do caracter.
                XOR   (HL)              Compara com linha do conjunto
                                        de caracteres.
                JR    Z,255A,S-SC-MATCH Salta se corresponderm.
                INC   A                 Verifica agora com caracter
                                        invertido (obtém +00 em A a
                                        partir de +FF).
                JR    NZ,2573,S-SCR-NXT Salta, se não corresponde.
                DEC   A                 Restaura +FF em A.
255A S-SC-MTCH  LD    C,A               Estado inverso (+00 ou +FF) para C.
                LD    B,+07             B conta as outras sete
                                        linhas.
255D S-SC-ROWS  INC   D                 Passa DE para linha seguinte
                                        (soma 256 decimal).
                INC   HL                Passa HL para linha seguinte
                                        (isto é, byte seguinte).
                LD    A,(DE)            Obtém a linha do visor.
                XOR   (HL)              Compara com linha da ROM.
                XOR   C                 Inclui o estado inverso.
                JR    NZ,2573,S-SCR-NXT Salta, se linha não concorda.
                DJNZ  255D,S-SC-ROWS    Salta atrás até ver todas as linhas.
                POP   BC                Elimina indicador do conj. de
                                        caracteres.
                POP   BC                E o indicador do visor.
                POP   BC                Contagem final para BC.
                LD    A,+80             Último código de caracter do
                                        conjunto, mais um.
                SUB   B                 A contém o código requerido.
                LD    BC,+0001          Necessário um espaço na área
                                        de trabalho.
                RST   0030,BC-SPACES    Construir o espaço.
                LD    (DE),A            Colocar o caracter nele.
                JR    257D,S-SCR-STO    Saltar para pôr caracter no «stack».
```

```
2573 S-SCR-NXT   POP   HL          Restaura o indicador do conj.
                                   de caracteres.
                 LD    DE,+0008    Avançá-lo 8 bytes, para o caracter
                 ADD   HL,DE       seguinte do conjunto.
                 POP   DE          Restaura o indicador do
                                   visor.
                 POP   BC          E o contador.
                 DJNZ  254F,S-SCRN-LP  Salto atrás para os 96 caracteres.
                 LD    C,B         «Stack» a cadeia vazia
                                   (comprimento zero).
257D S-SCR-STO   JP    2AB2,STK-STO-$  Saltar para «stack» o caracter
                                   correspondente, ou a cadeia nula,
                                   se não houve concordância.
```

Nota: A saída, através de STK-STO-$, constitui um erro porque conduz a um «duplo armazenamento» do resultado em cadeia (ver S-STRING, 25DB). A instrução deveria ser RET.

A última destas quatro subrotinas é a de «scanning» de atributos. É invocada por S-ATTR para produzir o valor de ATTR (x,y) que corresponde aos atributos da linha x, coluna y do visor.

```
2580 S-ATTR-S    CALL  2307,STK-TO-BC  x para C, y para B. 0<=x<=23
                 LD    A,C         decimal; 0<=y<=31 decimal.
                 RRCA              x é copiado para A e o número
                 RRCA              32 (decimal) * x (mod 8)+y é
                 RRCA              formado em A e copiado para L.
                 LD    C,A         32*x(mod 8)+INT(x/8) é também
                 AND   +E0         copiado para C.
                 XOR   B
                 LD    L,A         L contém byte baixo do endereço
                                   dos atributos.
                 LD    A,C         32*x(mod 8)+INT (x/8) é
                                   copiado para A.
                 AND   +03         88(decimal)+INT(x/8) é for-
                 XOR   +58         mado em A e copiado para H.
                 LD    H,A         H contém byte alto do ende-
                                   reço de atributos.
                 LD    A,(HL)      O byte de atributos é
                                   copiado para A.
                 JP    2D28,STACK-A  Saída, guardando o byte.
```

A tabela de procura de funções

Esta tabela contém 8 funções e 4 operadores. Incorpora assim 5 novas funções ao Spectrum e constitui um modo fácil de aceder algumas funções e operadores que já existiam no ZX81.

Posição	Código	Deslocamento	Nome	Endereço da rotina de tratamento
2596	22	1C	S-QUOTE	25B3
2598	28	4F	S-BRACKET	25E8
259A	2E	F2	S-DECIMAL	268D
259C	28	12	S-U-PLUS	25AF
259E	A8	56	S-FN	25F5
25A0	A5	57	S-RND	25F8
25A2	A7	84	S-PI	2627

Posição	Código	Deslocamento	Nome	Endereço da rotina de tratamento
25A4	A6	8F	S-INKEY$	2634
25A6	C4	E6	S-BIN (EQU, S-DECIMAL)	268D
25A8	AA	BF	S-SCREEN$	2668
25AA	AB	C7	S-ATTR	2672
25AC	A9	CE	S-POINT	2678
25AE	00			Marcador final

As rotinas de procura de funções

```
25AF S-U-PLUS    RST   0020,NEXT-CHAR  Passar apenas ao caracter
                 JP    24FF,S-LOOP-1   seguinte e saltar atrás para o
                                       ponto de entrada principal
                                       de SCANNING.
```

A «rotina de 'scanning' de aspas»: esta rotina trata as aspas de cadeias, quer simples como «nome», quer mais complexas como expressões que incluam aspas internas («»branco»») ou a aparentemente redundante VAL$«««a»»».

```
25B3 S-QUOTE     RST   0018,GET-CHAR  Obter o caracter actual.
                 INC   HL            Apontar para o início da cadeia.
                 PUSH  HL            Guardar o endereço inicial.
                 LD    BC,+0000      Passar comprimento para zero.
                 CALL  250F,S-QUOTE-S  Invocar a subrotina «verificação».
                 JR    NZ,25D9,S-Q-PRMS  Saltar, se flag «zero» em zero —
                                       — sem mais aspas.
25BE S-Q-AGAIN   CALL  250F,S-QUOTE-S  Invocar de novo para 3ªs aspas.
                 JR    Z,25BE,S-Q-AGAIN  E de novo para quintas,
                                       sétimas, etc.
                 CALL  2530,SYNTAX-Z   Se verifica sintaxe, saltar
                 JR    Z,25D9,S-Q-PRMS  para passar a zero o bit 6 de
                                       flags e continuar procura.
                 RST   0030,BC-SPACES  Abrir espaço na área de
                                       trabalho para a cadeia e
                                       aspas finais.
                 POP   HL            Obter indicador do início.
                 PUSH  DE            Guardar o indicador do
                                       primeiro espaço.
25CB S-Q-COPY    LD    A,(HL)        Obter um caracter da cadeia.
                 INC   HL            Apontar para o seguinte.
                 LD    (DE),A        Copiar o último para área de trabalho.
                 INC   DE            Apontar para o espaço seguinte.
                 CP    +22           O último caracter é «"»?
                 JR    NZ,25CB,S-Q-COPY  Se não, saltar para copiar o seguinte.
                 LD    A,(HL)        Mas se é, não copiar seguinte;
                 INC   HL            se este for «"», saltar para
                 CP    +22           copiar o que se segue a ele;
                 JR    Z,25CB,S-Q-COPY  senão, terminar cópia.
25D9 S-Q-PRMS    DEC   BC            Obter comprimento real para BC.
```

Notar que as primeiras aspas não foram contadas no comprimento; as aspas finais foram-no, e devem ser eliminadas agora. **No interior** da cadeia, as primeiras, terceiras, quintas, etc., aspas foram contadas, mas as segundas, quartas, etc., não.

```
                POP     DE              Restaurar início da cadeia copiada.
25DB S-STRING   LD      HL,+5C3B        Isto é FLAGS; esta entrada é
                RES     6,(HL)          usada sempre que o bit 6 deve
                BIT     7,(HL)          passar a 0 e uma cadeia guardada
                CALL    NZ,2AB2,STK-STO-$ em «stack» se executa uma linha.
                JP      2712,S-CONT-2   Saltar para continuar a varrer
                                        a linha.
```

Note-se que ao copiar a cadeia para a área de trabalho, cada dois pares de aspas **no interior** da cadeia (« ») devem ser reduzidos a um par de aspas de cadeia (»).

```
25E8 S-BRACKET  RST     0020,NEXT-CHAR  A rotina de procura de
                CALL    24FB,SCANNING   parêntesis obtém apenas o
                CP      +29             caracter seguinte e invoca
                                        SCANNING.
                JP      NZ,1C8A,REPORT-C Indicar erro se não existe
                RST     0020,NEXT-CHAR  2.º parêntesis; continuar procura.
                JP      2712,S-CONT-2

25F5 S-FN       JP      27BD,S-FN-SBRN  Rotina de procura de FN.
```

Esta rotina, para funções definidas pelo utilizador, salta simplesmente para a subrotina de procura de FN.

```
25F8 S-RND      CALL    2530,SYNTAX-Z   Se não se verifica sintaxe,
                JR      Z,2625,S-RND-END saltar para calcular um
                                        número aleatório.
                LD      BC,(SEED)       Obter o valor actual de
                                        SEED.
                CALL    2D28,STACK-BC   Colocá-lo no «stack» do calculador.
                RST     0028,FP-CALC    Usar agora o calculador.
                DEFB    +A1,stk-um      O «último valor» é agora
                DEFB    +0F,somar       SEED+1.
                DEFB    +34,stk-dado    Colocar o número decimal 75
                DEFB    +37,expoente+87 no «stack» do calculador.
                DEFB    +16,(+00,+00,+00)
                DEFB    +04,multiplicar «Último valor» (SEED+1)*75.
                DEFB    +34,stk-dado    Ver «STACK» LITERAIS para saber
                DEFB    +80,(4 bytes)   como se expandem os bytes altos
                DEFB    +41,expoente +91 de modo a calcular o número
                DEFB    +00,+00,+80,(+00) decimal 65537 no «stack» do
                                        calculador.
                DEFB    +32,n-mod-m     Dividir (SEED+1)*75 por 65537
                                        para obter um «resto» e uma
                                        «resposta».
                DEFB    +02,apagar      Eliminar a «resposta».
                DEFB    +A1,stk-um      Este «último valor» é agora
                DEFB    +03,subtrair    «resto» – 1.
                DEFB    +31,copiar      Fazer uma cópia do «último valor».
                DEFB    +38,fim-calc    O cálculo está terminado.
                CALL    2DA2,FP-TO-BC   Usar o «último valor» para dar
                LD      (SEED),BC       novo valor a SEED.
                LD      A,(HL)          Obter o expoente de
                                        «último valor».
                AND     A               Saltar para diante se o
                JR      Z,2625,S-RND-END expoente é zero.
                SUB     +10             Reduzir o expoente, isto é,
```



```
                LD      (HL),A          dividir «último valor» por 65536
                                        para obter «últ. valor» requerido.
2625 S-RND-END  JR      2630,S-PI-END   Saltar para depois da rotina PI.
```

A rotina de procura de PI: se não se verifica sintaxe, o valor de PI é calculado e forma o «último valor» no «stack» do calculador.

```
2627 S-PI       CALL    2530,SYNTAX-Z   Observar se se verifica sintaxe.
                JR      Z,2630,S-PI-END Saltar, se requerido.
                RST     0028,FP-CALC    Usar agora o calculador.
                DEFB    +A3,stk-pi/2    O valor de PI/2 é colocado no
                DEFB    +38,fim-calc    «stack» do calculador como
                                        «último valor».
                INC     (HL)            O expoente é incrementado,
                                        duplicando assim o «último
                                        valor» e obtendo PI.
2630 S-PI-END   RST     0020,NEXT-CHAR  Passar ao caracter seguinte.
                JP      26C3,S-NUMERIC  Saltar para diante.
2634 S-INKEY$   LD      BC,+105A        Prioridade +10 hex, código de
                RST     0020,NEXT-CHAR  operação +5A para a subrotina
                                        «leitura».
                CP      +23             Se o caracter seg. é «#», saltar.
                JP      Z,270D,S-PUSH-PO Haverá um argumento
                                        numérico.
                LD      HL,+5C3B        Isto é FLAGS.
                RES     6,(HL)          Bit 6 a zero para resultado «cadeia».
                BIT     7,(HL)          Testar se se verifica sintaxe.
                JR      Z,2665,S-INK$-EN Salto, se requerido.
                CALL    028E,KEY-SCAN   Obter o valor da tecla em DE.
                LD      C,+00           Preparar cadeia vazia; pôr no «stack»
                JR      NZ,2660,S-IK$-STK se se carrega em muitas teclas.
                CALL    031E,K-TEST     Verificar valor da tecla; «stack»
                JR      NC,2660,S-IK$-STK cadeia vazia se não satisfatório.
                DEC     D               +FF para D se modo L (bit 3 em «1»).
                LD      E,A             Valor da tecla em E para descodificar.
                CALL    0333,K-DECODE   Descodificar valor da tecla.
                PUSH    AF              Guardar o valor ASCII.
                LD      BC,+0001        Um espaço necessário na área
                                        de trabalho.
                RST     0030,BC-SPACES  Construí-lo agora.
                POP     AF              Restaurar o valor ASCII.
                LD      (DE),A          Preparar para guardá-lo como cadeia.
                LD      C,+01           Comprimento um.
2660 S-IK$-STK  LD      B,+00           Completar o parâmetro comprimento.
                CALL    2AB2,STK-STO-$  Guardar a cadeia requerida.
2665 S-INK$-EN  JP      2712,S-CONT-2   Saltar para diante.
2668 S-SCREEN$  CALL    2522,S-2-COORD  Verificar se são dadas duas
                                        coordenadas.
                CALL    NZ,2535,S-SCRN$-S Invocar a subrotina a menos
                RST     0020,NEXT-CHAR  que verifique sintaxe; obter
                JP      25DB,S-STRING   caracter seguinte e saltar.
2672 S-ATTR     CALL    2522,S-2-COORD  Verificar se são dadas duas
                                        coordenadas.
                CALL    NZ,2580,S-ATTR-S Invocar a subrotina a menos
                RST     0020,NEXT-CHAR  que verifique sintaxe; obter
                JR      26C3,S-NUMERIC  caracter seguinte e saltar
                                        para diante.
267B S-POINT    CALL    2522,S-2-COORD  Verificar se são dadas duas
                                        coordenadas.
```

```
                CALL    NZ,22C8,POINT-SUB   Invocar a subrotina a menos
                RST     0020,NEXT-CHAR      que verifique sintaxe; obter
                JR      26C3,S-NUMERIC      caracter seguinte e saltar
                                            para diante.
2684 S-ALPHNUM  CALL    2C88,ALPHANUM       É um caracter alfanumérico?
                JR      NC,26DF,S-NEGATE    Saltar se não é letra ou algarismo.
                CP      +41                 Saltar se é letra;
                JR      NC,26C9,S-LETTER    senão, continuar para
                                            S-DECIMAL.
```

A rotina de procura DECIMAL que se segue trata uma vírgula decimal ou um número que comece por um algarismo. Tem em conta igualmente a expressão BIN, que é tratada pela subrotina «decimal para vírgula flutuante».

```
268D S-DECIMAL  CALL    2530,SYNTAX-Z       Saltar para diante se está a
     (EQU. S-BIN) JR    NZ,26B5,S-STK-DEC   ser executada uma linha.
```

A acção realizada é agora bastante diferente conforme se verifica a sintaxe ou se executa. Se se verifica a sintaxe, é necessário calcular a forma em vírgula flutuante e copiá-la para a linha Basic. No entanto, quando está a ser executada uma linha, a forma em vírgula flutuante estará sempre disponível, pelo que é copiada para o «stack» do calculador para formar um «último valor».

Durante a verificação da sintaxe:

```
                CALL    2C9B,DEC-TO-FP      Descoberta a forma em
                                            vírgula flutuante.
                RST     0018,GET-CHAR       Leva HL a apontar para o
                                            algarismo depois do último.
                LD      BC,+0006            São necessárias seis posições.
                CALL    1655,MAKE-ROOM      Abrir espaço na linha
                                            Basic.
                INC     HL                  Apontar para o 1.º espaço livre.
                LD      (HL),+0E            Introduzir o código indicador
                                            de número.
                INC     HL                  Apontar para a segunda posição.
                EX      DE,HL               Este indicador deve estar em DE.
                LD      HL,(STKEND)         Obter a «antiga» STKEND.
                LD      C,+05               Deve-se mover 5 bytes.
                AND     A                   Limpar a flag «carry».
                SBC     HL,BC               «Nova» STKEND=«antiga»
                                            STKEND-5.
                LD      (STKEND),HL         Passar o n.º em vírgula flu-
                LDIR                        tuante do «stack» do calculador
                                            para a linha.
                EX      DE,HL               Passar o indicador de linha para HL.
                DEC     HL                  Apontar para o último byte somado.
                CALL    0077,TEMP-PTR1      Definir CH-ADD.
                JR      26C3,S-NUMERIC      Saltar para diante.
```

Durante a execução da linha:

```
26B5 S-STK-DEC  RST     0018,GET-CHAR       Obter o caracter actual.
26B6 S-SD-SKIP  INC     HL                  Passar ao caracter seguinte,
                LD      A,(HL)              continuamente, até ser
                CP      +0E                 encontrado o código indicador
                JR      NZ,26B6,S-SD-SKIP   de número.
```



```
                INC     HL                  Apontar para o 1.º byte do
                                            número.
                CALL    33B4,STACK-NUM      Mover o n.º em vírgula flutuante.
                LD      (CH-ADD),HL         Definir CH-ADD.
```

Foi identificado um resultado numérico, vindo de RND, PI, ATTR, POINT ou um número decimal, pelo que deve ser passado ao valor um o bit 6 de FLAGS.

```
26C3 S-NUMERIC  SET     6,(FLAGS)           Passa a 1 a flag «numérico».
                JR      26DD,S-CONT-1       Saltar para diante.
```

A rotina de procura de variáveis

Quando é identificado o nome de uma variável é invocada LOOK-VARS que procura nas variáveis que já existem na área de variáveis (ou na área de programa nas declarações DEF FN no caso de uma função FN definida pelo utilizador). Se for encontrado um valor numérico apropriado, este é copiado para o «stack» do calculador usando STACK-NUM. No entanto, uma entrada de cadeia ou «array» deve ter os parâmetros apropriados passados para o «stack» do calculador pela subrotina STK-VAR (ou no caso de uma função definida pelo utilizador, pela subrotina STK-F-ARG chamada por LOOK-VARS).

```
26C9 S-LETTER   CALL    2BB2,LOOK-VARS      Procurar nas variáveis existentes
                                            a entrada correspondente.
                JP      C,1C2E,REPORT-2     Indicar erro se não existe
                                            essa entrada.
                CALL    Z,2996,STK-VARS     Guardar no «stack» os parâmetros
                                            da entrada de cadeia/devolver
                                            endereço numérico base.
                LD      A,(FLAGS)           Obter FLAGS.
                CP      +C0                 Testar juntamente bits 6 e 7.
                JR      C,26DD,S-CONT-1     Um ou ambos passam a zero.
                INC     HL                  Será guardado um valor numérico.
                CALL    33B4,STACK-NUM      Deslocar o número.
26DD S-CONT-1   JR      2712,S-CONT-2       Saltar para diante.
```

O caracter é verificado comparando com o código de «-», identificando-se assim a operação de diminuição.

Antes de realizar o teste, o registo B passa a guardar a prioridade +09 e o registo C o código de operação +D8, necessários para esta operação.

```
26DF S-NEGATE   LD      BC,+09D8            Prioridade +09, código de
                                            operação +D8.
                CP      +2D                 É um «-»?
                JR      Z,270D,S-PUSH-PO    Saltar para diante se
                                            sim.
```

Em seguida, o caracter é novamente comparado com o código de VAL$, com uma prioridade 16 (decimal) e um código de operação 18 (hexadecimal).

```
                LD      BC,+1018            Prioridade 16 dec., código de
                                            operação +18 hex.
                CP      +AE                 É VAL$?
                JR      Z,270D,S-PUSH-PO    Saltar para diante, se sim.
```

O caracter actual deve representar uma das funções CODE a NOT, com códigos +AF a +C3.

```
        SUB   +AF              A gama de funções passa
                               de +AF a +C3 para
                               +00 a +14 hex.
        JP    C,1C8A,REPORT-C  Dar erro se fora da gama.
```

É identificada a função NOT, e tratada separadamente das outras funções.

```
        LD    BC,+04F0         Prioridade +04, código de
                               operação +F0.
        CP    +14              É a função NOT?
        JR    Z,270D,S-PUSH-PO Saltar, se sim.
        JP    NC,1C8A,REPORT-C Verificar gama de novo.
```

As funções restantes têm uma prioridade 16 decimal. Os códigos de operações destas funções são agora calculados. As funções que actuam sobre cadeias necessitam de ter o bit 6 em zero, e as funções que produzem resultados de cadeia devem ter o bit 7 em zero nos seus códigos de operação.

```
        LD    B,+10            Prioridade 16 decimal.
        ADD   A,+DC            A gama de funções é agora
                               +DC a +EF.
        LD    C,A              Transferir o código de operação.
        CP    +DF              Separar CODE, VAL e LEN
        JR    NC,2707,S-NO-TO-$ que actuam sobre cadeias
        RES   6,C              para obter resultados numéricos.
2707 S-NO-TO-$ CP +EE          Separar STR$ e CHR$
        JR    C,270D,S-PUSH-PO que actuam sobre números
                               para dar cadeias.
        RES   7,C              Marcar os códigos de operação.
                               Os outros códigos possuem
                               os bits 6 e 7 a um.
```

O código de prioridade e o código de operação da função que está a ser considerada são agora passados para o «stack»-máquina. É assim construída uma hierarquia de operações.

```
270D S-PUSH-PO  PUSH  BC       Guardar no «stack» os códigos
                RST   0020,NEXT-CHAR de prioridade e operação antes
                JP    24FF,S-LOOP-1 de considerar a parte da ex-
                                    pressão que se segue.
```

Continua agora a procura na linha. O argumento seguinte pode ser seguido de um «(», um operador binário ou, se foi atingido o final da expressão, por exemplo, por um caracter de retorno de linha ou uma vírgula, um separador ou um «THEN».

```
2712 S-CONT-2   RST   0018,GET-CHAR  Obter o caracter actual.
2713 S-CONT-3   CP    +28           Saltar para diante se não é
                JR    NZ,2723,S-OPERTR «(», que indica uma expressão
                                       entre parêntesis.
```

Se o «último valor» é numérico, a expressão entre parêntesis é uma verdadeira sub-expressão, e deve ser avaliada separadamente. No entanto, se o «último valor» é uma cadeia, a expressão entre parêntesis representa um elemento de um array ou uma parte de uma cadeia. Uma chamada a SLICING modifica os parâmetros da cadeia como for apropriado.

```
        BIT   6,(FLAGS)        Saltar para diante se trata
        JR    NZ,2734,S-LOOP   uma expressão numérica entre
                               parêntesis.
        CALL  2A52,SLICING     Modificar os parâmetros do
                               «último valor».
        RST   0020,NEXT-CHAR   Passar a considerar o caracter
        JR    2713,S-CONT-3    seguinte.
```

Se o caracter actual for de facto um operador binário receberá um código de operação na gama +C3 a +CF hexadecimal, e o código de prioridade apropriado.

```
2723 S-OPERTR   LD    B,+00    Código original para BC para
                LD    C,A      indexar tabela de operadores.
                LD    HL,+2795 Indicador da tabela.
                CALL  16DC,INDEXER Indexar a tabela.
                JR    NC,2734,S-LOOP Saltar para diante, se não
                                     se encontra operação.
                LD    C,(HL)   Obter código requerido na
                                tabela.
                LD    HL,+26ED Indicador da tabela de prio-
                                ridade: p. ex., 26ED+C3 dá
                                27B0 como 1.º endereço.
                ADD   HL,BC    Indexar a tabela.
                LD    B,(HL)   Obter a prioridade apropriada.
```

Entra-se agora no ciclo principal desta subrotina. Nesta fase existem:

1. Um «último valor» no «stack» do calculador.
2. O marcador de prioridade inicial no «stack»-máquina segundo uma hierarquia, de dimensões desconhecidas, de funções e códigos de operação binários. Esta hierarquia pode ser nula.
3. O par de registos BC que contém a operação e a prioridade «actuais» que, no caso de ter sido atingido o final de uma expressão, terão prioridade zero.

Inicialmente as «últimas» operação e prioridade são obtidas no «stack»-máquina e comparadas com as «actuais» operação e prioridade. Se a prioridade «actual» é superior à «última», sai-se do ciclo porque a prioridade «actual» é considerada mais forte do que a «última».

No entanto, se a prioridade actual é menos forte, é realizada a operação especificada como «última». A operação e a prioridade «actuais» voltam para o «stack»-máquina para percorrerem novamente o ciclo. Deste modo, é tratada a hierarquia das funções e operações binárias que foram colocadas em fila.

```
2734 S-LOOP     POP   DE       Obter a «última» operação
                               e a «última» prioridade.
                LD    A,D      A prioridade passa para o
                               registo A.
```

```
         CP    B              Comparar «último» com «actual».
         JR    C,2773,S-TIGHTER  Sair para esperar pelo argumento.
         AND   A              Ambas as prioridades são zero?
         JP    Z,0018,GET-CHAR  Sair por GET-CHAR, passando
                                assim «último valor» a re-
                                sultado pretendido.
```

Antes de ser executada a «última» operação, espera-se a função USR em «USR número» e «USR cadeia», conforme o bit 6 de FLAGS estava a um ou a zero quando o argumento de função foi guardado como «último valor».

```
         PUSH  BC             Guardar valores «actuais».
         LD    HL,+5C3B       Isto é FLAGS.
         LD    A,E            A «última» operação é
         CP    +ED            comparada com o código de USR, o
         JR    NZ,274C,S-STK-LST  que dará «USR número» se não é
                                modificada; salto se não «USR».
         BIT   6,(HL)         Testar bit 6 de FLAGS.
         JR    NZ,274C,S-STK-LST  Saltar se está a um («USR número»).
         LD    E,+99          Modificar o «último» código de
                                operação: «deslocamento» 19, +80
                                se entrada em cadeia e saída
                                numérica («USR cadeia»).
274C S-STK-LST
         PUSH  DE             Guardar os «últimos valores».
         CALL  2530,SYNTAX-Z  Não realizar a operação actual
         JR    Z,275B,S-SYNTEST  se se está a verificar a
                                sintaxe.
         LD    A,E            «Último» código de operação.
         AND   +3F            Eliminar bits 6 e 7 para con-
         LD    B,A            verter o código de operação
                                num deslocamento de calculador.
         RST   0028,FP-CALC   Usar agora o calculador.
         DEFB  +38,fp-calc 2  Realizar a operação.
         DEFB  +38,end-calc   Foi realizada.
         JR    2764,S-RUNTEST  Saltar para diante.
```

Uma parte importante da verificação sintáctica envolve o teste da operação para verificar se a natureza do «último valor» é do tipo correcto para a operação considerada.

```
2758 S-SYNTEST
         LD    A,E            Obter o «último» código de operação.
         XOR   (FLAGS)        Teste da natureza do «último valor»
         AND   +40            pelos requisitos da operação.
                              Devem ser iguais para que a
                              sintaxe seja correcta.
2761 S-RPORT-C
         JP    NZ,1C8A,REPORT-C  Saltar, se sintaxe errada.
```

Antes de saltar atrás para percorrer novamente o ciclo deve-se registar a natureza do «último valor» em FLAGS.

```
2764 S-RUNTEST
         POP   DE             Obter «último» código de operação.
         LD    HL,+5C3B       Isto é FLAGS.
         SET   6,(HL)         Considerar resultado numérico.
         BIT   7,E            Saltar para diante se o «último
         JR    NZ,2770,S-LOOPEND  valor» tem natureza numérica.
         RES   6,(HL)         É uma cadeia.
2770 S-LOOPEND
         POP   BC             Obter os valores «actuais» em BC.
         JR    2734,S-LOOP    Saltar atrás.
```


Sempre que a operação «actual» é mais forte, os valores «último» e «actual» voltam para o «stack»-máquina. No entanto, se a operação «actual» requer uma cadeia como operando, então o código de operação é modificado a fim de indicar este requisito.

```
2773 S-TIGHTER
         PUSH  DE             «Últimos» valores vão para «stack».
         LD    A,C            Obter o código de operação
                              «actual».
         BIT   6,(FLAGS)      Não modificar o código de
         JR    NZ,2790,S-NEXT  operação se trata um operando
                              numérico.
         AND   +3F            Limpar bits 6 e 7.
         ADD   A,+08          Aumentar o código de +08 hex.
         LD    C,A            Devolver o código ao
                              registo C.
         CP    +10            É a operação «AND»?
         JR    NZ,2788,S-NOT-AND  Saltar, se não.
         SET   6,C            «AND» exige um operando
                              numérico.
         JR    2790,S-NEXT    Saltar para diante.
2788 S-NOT-AND
         JR    C,2761,S-RPORT-C  As operações −, *, /, ↑ e OR
                              não são possíveis entre cadeias.
         CP    +17            A operação é um «+»?
         JR    Z,2790,S-NEXT  Saltar, se sim.
         SET   7,C            As outras operações produ-
                              zem um resultado numérico.
2790 S-NEXT
         PUSH  BC             Os valores «actuais» vão
                              para o «stack»-máquina.
         RST   0020,NEXT-CHAR Considerar caracter seguinte.
         JP    24FF,S-LOOP-1  Percorrer de novo o ciclo.
```

A tabela de operadores

Posição	Código	Código de operador	Operador	Posição	Código	Código de operador	Operador
2795	2B	CF	+	27A3	3C	CD	<
2797	2D	C3	−	27A5	C7	C9	<=
2799	2A	C4	*	27A7	C8	CA	>=
2898	2F	C5	/	27A9	C9	CB	<>
279D	5E	C6	↑	27AB	C5	C7	OR
279F	3D	CE	=	27AD	C6	C8	AND
27A1	3E	CC	>	27AF	00	Separador final	

A tabela de prioridades (tabela de precedência)

Posição	Prioridade	Operador	Posição	Prioridade	Operador
27B0	06	−	27B7	05	>=
27B1	08	*	27B8	05	<>
27B2	08	/	27B9	05	>
27B3	0A	↑	27BA	05	<
27B4	02	OR	27BB	05	=
27B5	03	AND	27BC	06	+
27B6	05	<=			

A subrotina «Procura de funções»

Esta subrotina é invocada pela «rotina de procura de FN» a fim de avaliar uma função definida pelo utilizador que ocorra numa linha Basic. A subrotina pode ser considerada em quatro partes:

1) A sintaxe da declaração FN é verificada.
2) Durante a execução, procura-se no programa uma declaração DEF FN, sendo comparados os nomes das funções até concordarem — ou ser impressa uma mensagem de erro.
3) Os argumentos de FN são avaliados por chamadas à subrotina SCANNING.
4) A função propriamente dita é avaliada invocando SCANNING, que por sua vez, chama LOOK-VARS e, portanto, a subrotina «GUARDAR EM «STACK» O ARGUMENTO DA FUNÇÃO».

```
27BD S-FN-SBRN   CALL  2530,SYNTAX-Z   Se não se verifica sintaxe,
                 JR    NZ,27F7,SF-RUN  salto para SF-RUN.
                 RST   0020,NEXT-CHAR  Obter o primeiro caracter
                                       do nome.
                 CALL  2C8D,ALPHA      Se não é alfabético, indicar
                 JP    NC,1C8A,REPORT-C erro.
                 RST   0020,NEXT-CHAR  Obter caracter seguinte.
                 CP    +24             É um «$»?
                 PUSH  AF              Guardar a flag «zero» no «stack».
                 JR    NZ,27D0,SF-BRKT-1 Saltar, se não era um «$».
                 RST   0020,NEXT-CHAR  Mas obter caracter seguinte
                                       se era.
27D0 SF-BRKT-1   CP    +28             Se um caracter não é um «(», indi-
                 JR    NZ,27E6,SF-RPRT-C car o erro.
                 RST   0020,NEXT-CHAR  Obter o caracter seguinte.
                 CP    +29             É um «)»?
                 JR    Z,27E9,SF-FLAG-6 Salto se for; não existem
                                       argumentos.
27D9 SF-ARGMTS   CALL  24FB,SCANNING   No interior do ciclo, chamar
                                       SCANNING para verificar a sin-
                                       taxe de cada argumento e inserir
                                       números em vírgula flutuante.
                 RST   0018,GET-CHAR   Obter o caracter que se segue
                 CP    +2C             ao argumento; se não é um «,»
                 JR    NZ,27E4,SF-BRKT-2 saltar — sem mais argumentos.
                 RST   0020,NEXT-CHAR  Obter o primeiro caracter no
                                       argumento seguinte.
                 JR    27D9,SF-ARGMTS  Saltar atrás para considerar
                                       este argumento.
27E4 SF-BRKT-2   CP    +29             O caracter actual é um «)»?
27E6 SF-RPRT-C   JP    NZ,1C8A,REPORT-C Indicar erro, se não.
27E9 SF-FLAG-6   RST   0020,NEXT-CHAR  Apontar para o caracter seguinte
                                       na linha Basic.
                 LD    HL,+5C3B        Isto é FLAGS; considerar uma
                 RES   6,(HL)          função avaliada como cadeia
                                       e passar a zero o bit 6 de FLAGS.
                 POP   AF              Restaurar a flag «zero»; saltar se
                 JR    Z,27F4,SF-SYN-EN FN é de facto avaliada como cadeia.
                 SET   6,(HL)          Senão, passa a 1 o bit 6 de FLAGS.
27F4 SF-SYN-EN   JP    2712,S-CONT-2   Saltar atrás para continuar
                                       a varrer a linha.
```


2) Durante a execução da linha, deve-se procurar uma declaração DEF FN.

```
27F7 SF-RUN      RST   0020,NEXT-CHAR  Obter o 1.º caracter do
                                       nome.
                 AND   +DF             Passar a 0 o bit 5 para maiúsculas.
                 LD    B,A             Copiar o nome para B.
                 RST   0020,NEXT-CHAR  Obter o caracter seguinte.
                 SUB   +24             Subtrair 24 hex, o código de «$».
                 LD    C,A             Copiar o resultado para C (0 para
                                       cadeia, não-zero para uma função
                                       numérica).
                 JR    NZ,2802,SF-ARGMT1 Saltar se não for zero; função
                                       numérica.
                 RST   0020,NEXT-CHAR  Obter caracter seguinte, «(».
2802 SF-ARGMT1   RST   0020,NEXT-CHAR  Obter 1.º caracter do primeiro
                                       argumento.
                 PUSH  HL              Guardar o indicador deste no
                                       «stack».
                 LD    HL,(PROG)       Apontar para o início do
                                       programa.
                 DEC   HL              Voltar atrás uma posição.
2808 SF-FND-DF   LD    DE,+00CE        Procurar «DEF FN».
                 PUSH  BC              Guardar o nome e o «estado cadela».
                 CALL  1D86,LOOK-PROG  Procurar agora no programa.
                 POP   BC              Restaurar o nome e o estado.
                 JR    NC,2814,SF-CP-DEF Saltar se encontra uma decla-
                                       ração DEF FN.
```

Mensagem «P — FN without DEF»

```
2812 REPORT-P    RST   0008,ERROR-1    Invocar rotina de
                 DEFB  +18             tratamento de erro.
```

Quando é encontrada uma declaração DEF FN, são comparados o nome e o estado das duas funções: se não concordam, retoma-se a procura.

```
2814 SF-CP-DEF   PUSH  HL              Guardar o indicador de DEF FN
                                       para o caso de ser necessário
                                       retomar a procura.
                 CALL  28AB,FN-SKPOVR  Obter o nome da função DEF
                                       FN.
                 AND   +DF             Passa o bit 5 a 0 para maiúsculas.
                 CP    B               Concorda com o nome de FN?
                 JR    NZ,2825,SF-NOT-FD Saltar, se não.
                 CALL  28AB,FN-SKPOVR  Obter o caracter seguinte em
                                       DEF FN.
                 SUB   +24             Subtrair 24 hex, o código de
                                       «$».
                 CP    C               Compara o estado com o de
                                       FN.
                 JR    Z,2831,SF-VALUES Saltar se encontra corres-
                                       pondência total.
2825 SF-NOT-FD   POP   HL              Restaurar o indicador de
                                       «DEF FN».
                 DEC   HL              Voltar atrás uma posição.
                 LD    DE,+0200        Usar a rotina de procura para
                 PUSH  BC              descobrir o fim da declaração
```

```
         CALL  1988,EACH-STMT   DEF FN, preparando a procura
         POP   BC               seguinte; guardar o nome e o
                                estado, entretanto.
         JR    2808,SF-FND-DF   Saltar atrás para nova procura.
```

3) Foi agora encontrada a declaração DEF FN correcta. Os argumentos da declaração FN serão avaliados através de repetidas chamadas a SCANNING, e os seus valores de 5 bytes (ou parâmetros, no caso das cadeias) serão inseridos na declaração DEF FN nos espaços reservados durante a verificação de sintaxe. HL será usado para apontar a posição ao longo da declaração DEF FN (invocando FN-SKPOVR conforme necessário) enquanto CH-ADD aponta ao longo da declaração FN (invocando RST 0020, NEXT-CHAR, quando necessário).

```
2831 SF-VALUES  AND   A              Se HL aponta para um «$»,
                CALL  Z,28AB,FN-SKPOVR passar para o «(».
                POP   DE             Eliminar o indicador de
                                     «DEF FN».
                POP   DE             Obter o indicador do 1.º
                LD    (CH-ADD),DE    argumento de FN, e copiá-
                                     -lo para CH-ADD.
                CALL  28AB,FN-SKPOVR Passar agora o «(».
                PUSH  HL             Guardar este indicador no «stack».
                CP    +29            Aponta para um «)»?
                JR    Z,2885,SF-R-BR-2 Se sim, saltar: FN não tem
                                     argumentos.
2843 SF-ARG-LP  INC   HL             Apontar para o código seguinte.
                LD    A,(HL)         Pôr o código em A.
                CP    +0E            É o código indicador de número,
                                     0E hex?
                LD    D,+40          Passar a um o bit 6 de D para
                                     um argumento numérico.
                JR    Z,2852,SF-ARG-VL Saltar, se zero: argumento
                                     numérico.
                DEC   HL             Garantir que HL aponta para
                                     o caracter «$» (e não, p. ex., para
                CALL  28AB,FN-SKPOVR um caracter de controlo).
                INC   HL             HL aponta agora para o «indi-
                                     cador de número».
                LD    D,+00          O bit 6 de D passa a zero: argumen-
                                     to alfanumérico.
2852 SF-ARG-VL  INC   HL             Apontar para o primeiro dos 5
                                     bytes de DEF FN.
                PUSH  HL             Guardar este indicador no «stack».
                PUSH  DE             Guardar o «estado cadeia» do
                                     argumento.
                CALL  24FB,SCANNING  Avaliar agora o argumento.
                POP   AF             Obter a flag n.º/cadeia em A.
                XOR   (FLAGS)        Comparar o seu bit 6 com o resul-
                AND   +40            tado de SCANNING.
                JR    NZ,2888,REPORT-Q Dar mensagem Q se não con-
                                     cordam.
                POP   HL             Obter o indicador do primeiro
                EX    DE,HL          dos 5 espaços de DEF FN para os
                                     registos DE.
                LD    HL,(STKEND)    Apontar HL para STKEND.
                LD    BC,+0005       BC contará 5 bytes a des-
                                     locar.
```



```
                SBC   HL,BC          Primeiro, diminuir STKEND de 5,
                LD    (STKEND),HL    eliminando o «último valor» do
                                     «stack».
                LDIR                 Copiar os 5 bytes para os espa-
                                     ços em DEF FN.
                EX    DE,HL          Apontar HL para o código seguinte.
                DEC   HL             Garantir que HL aponta para o
                CALL  28AB,FN-SKPOVR caracter após os 5 bytes.
                CP    +29            É um «)»?
                JR    Z,2885,SF-R-BR-2 Saltar, se sim: não há mais
                                     argumentos na declaração
                                     DEF FN.
                PUSH  HL             É um «,»; guardar o seu indicador.
                RST   0018,GET-CHAR  Obter o caracter que se segue ao
                                     último caracter de FN que foi
                                     avaliado.
                CP    +2C            Se não é um «,», saltar: argu-
                JR    NZ,2888,REPORT-Q mentos não concordantes de FN e
                                     DEF FN.
                RST   0020,NEXT-CHAR Apontar CH-ADD para o argumento
                                     seguinte de FN.
                POP   HL             Apontar HL para «,» em DEF FN
                                     novamente.
                CALL  28AB,FN-SKPOVR Passar HL para o argumento
                                     seguinte de DEF FN.
                JR    2843,SF-ARG-LP Saltar atrás para considerar
                                     este argumento.
2885 SF-R-BR-2  PUSH  HL             Guardar o indicador de «)» em
                                     DEF FN.
                RST   0018,GET-CHAR  Obter o caracter após o último
                                     argumento de FN.
                CP    +29            É um «)»?
                JR    Z,288D,SF-VALUE Se sim, saltar para avaliar a
                                     função; mas se não, apresentar
                                     mensagem Q.
```

Mensagem «Q — Parameter error».

```
2888 REPORT-Q   RST   0008,ERROR-1   Invocar rotina de
                DEFB  +19            tratamento de erro.
```

4) Finalmente, é avaliada a própria função invocando SCANNING, depois de começar por guardar em DEFADD o endereço dos argumentos tal como ocorrem na declaração DEF FN. Garante-se assim que LOOK-VARS, quando invocado por SCANNING, comece por procurar nestes argumentos os valores requeridos, antes de iniciar a procura na área de variáveis.

```
288D SF-VALUE   POP   DE             Restaurar o indicador de «)»
                                     em DEF FN.
                EX    DE,HL          Passar este indicador para HL.
                LD    (CH-ADD),HL    Inseri-lo em CH-ADD.
                LD    HL,(DEFADD)    Obter o valor antigo de DEFADD.
                EX    (SP),HL        Guardá-lo em «stack», e obter o
                LD    (DEFADD),HL    endereço inicial da área de argu-
                                     mentos de DEF FN em DEFADD.
                PUSH  DE             Guardar endereço de «)» em FN.
                RST   0020,NEXT-CHAR Deslocar CH-ADD para além de «)»
```

```
        RST     0020,NEXT-CHAR  e «=» para o início da expressão
                                em DEF FN.
        CALL    24FB,SCANNING   Avaliar agora a função.
        POP     HL              Restaurar o endereço de «)»
                                em FN.
        LD      (CH-ADD),HL     Guardá-lo em CH-ADD.
        POP     HL              Restaurar o valor original
                                de DEFADD.
        LD      (DEFADD),HL     Pô-lo de novo em DEFADD.
        RST     0020,NEXT-CHAR  Obter o caracter seguinte da
                                linha Basic.
        JP      2712,S-CONT-2   Saltar atrás para continuar o
                                varrimento.
```

A subrotina «SKIPOVER»

Esta subrotina é usada por FN e por STK-F-ARG para deslocar HL ao longo da declaração DEF FN enquanto CH-ADD se mantém, dado que aponta para a declaração FN.

```
28AB FN-SKPOVR  INC     HL              Apontar para o código seguinte
                                        na declaração.
        LD      A,(HL)          Copiar o código para A.
        CP      +21             Saltar atrás para passar adiante
        JR      C,28AB,FN-SKPOVR se for espaço ou caracter de controlo.
        RET                     Final.
```

A subrotina «LOOK-VARS»

Esta subrotina é invocada sempre que é necessária uma procura na área de variáveis ou nos argumentos de uma declaração DEF FN. Entra-se na subrotina com a variável de sistema CH-ADD apontada para a primeira letra do nome da variável cuja posição está a ser procurada. O nome encontrar-se-á na área de programa ou na área de trabalho. A subrotina constrói inicialmente um byte discriminador, no registo C, que se baseia na primeira letra do nome da variável. Os bits 5 e 6 deste byte indicam o tipo da variável que está a ser tratada.

O registo B é usado para guardar flags bit a bit.

```
2882 LOOK-VARS  SET     6,(FLAGS)       Pressupor uma variável numérica.
        RST     0018,GET-CHAR   Obter em A o primeiro caracter.
        CALL    2C8D,ALPHA      É alfabético?
        JP      NC,1C8A,REPORT-C Dar uma mensagem de erro se não
                                for assim.
        PUSH    HL              Guardar o indicador da primeira
                                letra.
        AND     +1F             Transferir os bits 0 a 4 da
        LD      C,A             letra para o registo C; os bits
                                5 e 7 são sempre passados a zero.
        RST     0020,NEXT-CHAR  Obter em A o 2.º caracter.
        PUSH    HL              Guardar também este indicador.
        CP      +28             O 2.º caracter é um «(»?
        JR      Z,28EF,V-RUN/SYN Separar arrays de números.
        SET     6,C             Passar o bit 6 a um.
```



```
        CP      +24             O 2.º caracter é um «$»?
        JR      Z,28DE,V-STR-VAR Separar todas as cadeias.
        SET     5,C             Passar a um o bit 5.
        CALL    2C88,ALPHANUM   Se o nome da variável possui
        JR      NC,28E3,V-TEST-FN apenas um caracter saltar
                                para diante.
```

Descobrir agora o caracter final de um nome com mais de um caracter.

```
28D4 V-CHAR     CALL    2C88,ALPHANUM   É um caracter alfanumérico?
        JR      NC,28EF,V-RUN/SYN Sair do ciclo quando encontra
                                o final do nome.
        RES     6,C             Marcar o byte discriminador.
        RST     0020,NEXT-CHAR  Obter o caracter seguinte.
        JR      28D4,V-CHAR     Voltar atrás para testá-lo.
```

As cadeias simples e arrays de cadeias requerem que o bit 6 de FLAGS seja passado a zero.

```
28DE V-STR-VAR  RST     0020,NEXT-CHAR  Mover CH-ADD para além de «$».
        RES     6,(FLAGS)       Passar a 0 o bit 6 para indicar
                                uma cadeia.
```

Se o byte alto de DEFADD não é zero, indicando que está a ser avaliada uma função (uma «FN»), e se se está em execução, serão procurados os argumentos na declaração DEF FN.

```
28E3 V-TEST-FN  LD      A,(DEFADD-hi)   DEFADD-alto é zero?
        AND     A
        JR      Z,28EF,V-RUN/SYN Se sim, saltar para diante.
        CALL    2530,SYNTAX-Z   Em execução?
        JP      NZ,2951,STK-F-ARG Se sim, saltar para diante para
                                procurar a declaração DEF FN.
```

Senão (ou se a variável não foi encontrada na declaração DEF FN), será efectuada uma procura na área de variáveis, a menos que se verifique a sintaxe.

```
28EF V-RUN/SYN  LD      B,C             Copiar o byte discriminador
                                para o registo B.
        CALL    2530,SYNTAX-Z   Saltar para diante se «em
        JR      NZ,28FD,V-RUN   execução».
        LD      A,C             Passar o discriminador para A.
        AND     +E0             Eliminar a parte de código de caracter.
        SET     7,A             Indicar «sintaxe» com bit 7 a «1».
        LD      C,A             Restaurar o discriminador.
        JR      2934,V-SYNTAX   Saltar para diante para continuar.
```

Está a ser executada uma linha Basic, pelo que se deve procurar na área de variáveis.

```
28FD V-RUN      LD      HL,(VARS)       Recolher o indicador VARS.
```

Entrar agora num ciclo a fim de considerar os nomes das variáveis existentes.

```
2900 V-EACH     LD      A,(HL)          A primeira letra de cada
                                        variável existente.
                AND     +7F             Comparar os bits 0 a 6.
                JR      Z,2932,V-80-BYTE Saltar quando se atinge o
                                        «byte-80».
                CP      C               Comparação.
                JR      NZ,292A,V-NEXT  Saltar para diante se os 1.os
                                        caracteres não concordam.
                RLA                     Rodar A para a esquerda e depois
                ADD     A,A             duplicá-lo para testar os bits 5 e 6.
                JP      P,293F,V-FOUND-2 Variáveis de cadeia e de array.
                JR      C,293F,V-FOUND-2 Variáveis numéricas simples
                                        e FOR NEXT.
```

É necessário comparar completamente os nomes compridos.

```
                POP     DE              Fazer uma cópia do indicador
                PUSH    DE              para o 2.º caracter.
                PUSH    HL              Guardar o indicador da 1.ª letra.
2912 V-MATCHES  INC     HL              Considerar o caracter seguinte.
2913 V-SPACES   LD      A,(DE)          Obter cada caracter por sua vez.
                INC     DE              Apontar para o caracter seguinte.
                CP      +20             O caracter é um «espaço»?
                JR      Z,2913,V-SPACES Ignorar os espaços.
                OR      +20             Passar a «1» o bit 5 para com-
                                        parar minúsculas a maiúsculas.
                CP      (HL)            Fazer a comparação.
                JR      Z,2912,V-MATCHES Voltar atrás para outro caracter
                                        se concordar.
                OR      +80             Concorda se bit 7 a um?
                CP      (HL)            Tentar.
                JR      NZ,2929,V-GET-PTR Saltar para diante se os últi-
                                        mos caracteres não concordam.
                LD      A,(DE)          Verificar se foi atingido o
                CALL    2C88,ALPHANUM   fim do nome antes de saltar
                JR      NC,293E,V-FOUND-1 para diante.
```

Em todos os casos em que os nomes não concordam, o par de registos HL deve ser apontado para a variável seguinte na área de variáveis.

```
2929 V-GET-PTR  POP     HL              Obter o indicador.
292A V-NEXT     PUSH    BC              Guardar B e C brevemente.
                CALL    1988,NEXT-ONE   DE é levado a apontar para
                                        a variável seguinte.
                EX      DE,HL           Comutar os 2 indicadores.
                POP     BC              Recuperar B e C.
                JR      2900,V-EACH     Percorrer de novo o ciclo.
```

Vir aqui se não se encontrou nenhuma entrada com o nome correcto.

```
2932 V-80-BYTE  SET     7,B             Sinal «variável não encontrada».
```

Vir aqui se verifica sintaxe.

```
2934 V-SYNTAX   POP     DE              Libertar o indicador do 2.º
                                        caracter.
                RST     0018,GET-CHAR   Obter o caracter actual.
```



```
                CP      +28             É um «(»?
                JR      Z,2943,V-PASS   Saltar para diante.
                SET     5,B             Indicar que não trata um array
                JR      294B,V-END      e saltar para diante.
```

Vir aqui quando for encontrada uma entrada com o nome correcto.

```
293E V-FOUND-1  POP     DE              Libertar o indicador da variável.
293F V-FOUND-2  POP     DE              E o indicador do 2.º caracter.
                POP     DE              E o indicador da 1.ª letra.
                PUSH    HL              Guardar o indicador da «última» letra.
                RST     0018,GET-CHAR   Obter o caracter actual.
```

Se o nome de variável em comparação tem mais de uma letra, os outros caracteres devem ser passados.

Nota: Isto parece já ter sido feito por V-CHAR.

```
2943 V-PASS     CALL    2C88,ALPHANUM   É alfanumérico?
                JR      NC,294B,V-END   Salto quando foi encontrado
                                        o fim do nome.
                RST     0020,NEXT-CHAR  Obter o caracter seguinte.
                JR      2943,V-PASS     Voltar atrás e testar.
```

Os parâmetros de saída são agora definidos.

```
294B V-END      POP     HL              HL contém o indicador da letra
                                        de um nome curto ou o «últi-
                                        mo» caracter de um comprido.
                RL      B               Rodar o registo inteiro.
                BIT     6,B             Especificar o estado do bit 6.
                RET                     Terminar.
```

Os parâmetros de saída da subrotina podem ser resumidos do seguinte modo: a variável de sistema CH-ADD aponta para a primeira posição depois do nome da variável tal como ocorre na linha Basic.
Quando se obtém «variable not found»:

1) A flag «carry» passa a um;
2) A flag «zero» só se encontra a um quando se procura uma variável de array.
3) O par de registos HL aponta para a primeira letra do nome da variável tal como ocorre na linha Basic.

Quando a variável é encontrada:

1) A flag «carry» passa a zero.
2) A flag «zero» fica a um, tanto para variáveis simples de cadeia, como para todas as variáveis de array.
3) O par de registos HL aponta para a letra de um nome «curto», ou para o último caracter de um nome «comprido», da entrada existente na área de variáveis.

Em todos os casos os bits 5 e 6 do registo C indicam o tipo de variável que se está a tratar. O bit 7 é o complemento da flag SINTAXE/EXECUÇÃO. Mas só quando a subrotina é usada em execução, passam os bits 0 a 4 a conter o código da letra da variável.

Ao verificar sintaxe, o retorno é sempre feito com a flag «carry» a zero. A flag «zero» encontra-se a um no caso dos arrays e a zero no de todas as outras variáveis, excepto quando um nome de cadeia simples é incorrectamente seguido de um «(», que passa a um a flag «zero» e, no caso de SAVE «nome» DATA a$(), é também admitido em termos sintácticos.

A subrotina «'STACK' de argumentos de funções»

Esta subrotina é invocada por LOOK-VARS quando o byte alto de DEFADD não é zero, a fim de investigar a área de argumentos de uma declaração DEF FN, antes de realizar a procura na área de variáveis. Se se encontra a variável na declaração DEF FN, são guardados em «stack» os parâmetros da variável de cadeia e é enviado um sinal indicativo de que não é necessário invocar STK/VAR. Mas é deixada a SCANNING a função de guardar em «stack» o valor das variáveis numéricas em 26DA, do modo habitual.

```
2951 STK-F-ARG   LD    HL,(DEFADD)     Apontar para o 1.º caracter da
                 LD    A,(HL)          área de argumentos e pô-lo em A.
                 CP    +29             É um «)»?
                 JP    Z,28EF,V-RUN/SYN Saltar para procurar na área
                                       de variáveis.
295A SFA-LOOP    LD    A,(HL)          Obter o argumento seguinte
                                       no ciclo.
                 OR    +60             Passar a «1» os bits 5 e 6, se
                 LD    B,A             for uma variável numérica sim-
                                       ples; copiá-la para B.
                 INC   HL              Apontar para o código seguinte.
                 LD    A,(HL)          Pô-lo no registo A.
                 CP    +0E             É o indicador de número,
                                       0E hex?
                 JR    Z,2968,SFA-CP-VR Saltar, se sim; variável numérica.
                 DEC   HL              Garantir que HL aponta para
                 CALL  28A8,FN-SKPOVR  o caracter «$», e não para um
                                       espaço ou código de controlo.
                 INC   HL              HL aponta agora para o «indica-
                                       dor de número».
                 RES   5,B             Passa a zero o bit 5 de B:
                                       var. de cadeia.
296B SFA-CP-VR   LD    A,B             Obter o nome da variável em A.
                 CP    C               É aquela que procuramos?
                 JR    Z,2981,SFA-MATCH Saltar se concorda.
                 INC   HL              Passar os 5 bytes do número em
                 INC   HL              vírgula flutuante ou dos parâme-
                 INC   HL              tros de cadeia a fim de obter
                 INC   HL              o argumento seguinte.
                 INC   HL
                 CALL  28AB,FN-SKPOVR  Passar ao caracter seguinte.
                 CP    +29             É um «)»?
                 JP    Z,28EF,V-RUN/SYN Se sim, saltar para procurar na
                                       área de variáveis.
                 CALL  28AB,FN-SKPOVR  Apontar para o argumento seguinte.
                 JR    295A,SFA-LOOP   Saltar atrás para o considerar.
```


Foi encontrada uma concordância de nomes. No caso de uma variável de cadeia são guardados no «stack» os seus parâmetros, evitando a necessidade de invocar a subrotina STK-VAR.

```
2981 SFA-MATCH   BIT   5,C             Verificar se é variável numérica.
                 JR    NZ,2991,SFA-END Saltar, se sim; SCANNING
                                       guardá-la-á.
                 INC   HL              Apontar para o primeiro dos
                                       5 bytes a guardar.
                 LD    DE,(STKEND)     Apontar DE para STKEND.
                 CALL  33C0,MOVE-FP    Guardar em «stack» os 5 bytes.
                 EX    DE,HL           Apontar HL para a nova posição
                 LD    (STKEND),HL     de STKEND, e redefinir a
                                       variável de sistema.
2991 SFA-END     POP   DE              Eliminar os indicadores
                 POP   DE              LOOK-VARS (2.º e 1.º indicadores
                                       de caracteres).
                 XOR   A               Retorno com as flags «zero» e
                 INC   A               «carry» a zero — indicando
                                       que deixa de ser necessário
                                       invocar STK-VAR.
                 RET                   Terminado.
```

A subrotina «STK-VAR»

Esta subrotina é geralmente usada para descobrir os parâmetros que definem uma entrada de cadeia existente na área de variáveis, ou para devolver no par de registos HL o endereço base de um elemento particular ou de um array de números. Quando invocada por DIM, a subrotina limita-se a verificar a sintaxe da declaração Basic.

Note-se que os parâmetros que definem uma cadeia podem ser alterados invocando SLICING se isto for especificado.

Inicialmente, os registos A e B são limpos, sendo verificado o bit 7 do registo C a fim de determinar se se está a verificar a sintaxe.

```
2996 STK-VAR     XOR   A               Limpar a flag «array».
                 LD    B,A             Limpar o registo B.
                 BIT   7,C             Saltar adiante se se verifica
                 JR    NZ,29E7,SV-COUNT a sintaxe.
```

Em seguida, separam-se as cadeias simples das variáveis de array.

```
                 BIT   7,(HL)          Saltar para diante se se trata
                 JR    NZ,29AE,SV-ARRAYS de uma variável de array.
```

Os parâmetros de uma cadeia simples são fáceis de encontrar.

```
                 INC   A               Sinalizar «uma cadeia simples».
29A1 SV-SIMPLE$  INC   HL              Percorrer a entrada.
                 LD    C,(HL)          Obter byte baixo do contador
                                       de comprimento.
                 INC   HL              Avançar o indicador.
                 LD    B,(HL)          Obter o byte alto do contador
                                       de comprimento.
                 INC   HL              Avançar o indicador.
```

```
            EX      DE,HL           Transferir o indicador
                                    para a cadeia.
            CALL    2AB2,STK-STORE  Passar estes parâmetros para
                                    o «stack» do calculador.
            RST     0018,GET-CHAR   Obter o caracter actual e
                                    saltar para diante para
            JP      2A49,SV-SLICE?  verificar se é necessário «slice».
```

É agora encontrado o endereço base de um elemento de um array. Inicialmente, é obtido o «número de dimensões».

```
29AE SV-ARRAYS  INC     HL              Passar os bytes de com-
                INC     HL              primento.
                INC     HL
                LD      B,(HL)          Obter o «número de di-
                                        mensões».
                BIT     6,C             Saltar para diante se se trata
                JR      Z,29C0,SV-PTR   um array de números.
```

Se um array de cadeias possui um «número de dimensões» igual a «1», este array pode ser tratado como uma cadeia simples.

```
                DEC     B               Diminuir o «número de dimen-
                JR      Z,29A1,SV-SIMPLE$ sões» e saltar, se o número
                                        for agora zero.
```

Em seguida, verificar se a variável é seguida de um índice na linha Basic.

```
                EX      DE,HL           Guardar o indicador em DE.
                RST     0018,GET-CHAR   Obter o caracter actual.
                CP      +28             É um «(»?
                JR      NZ,2A20,REPORT-3 Indicar erro, se não for.
                EX      DE,HL           Restaurar o indicador.
```

Tanto no caso de arrays numéricos como no de arrays de cadeias, o indicador da variável é transferido para o par de registos DE antes de o índice ser avaliado.

```
29C0 SV-PTR     EX      DE,HL           Passar o indicador para DE.
                JR      29E7,SV-COUNT   Saltar para diante.
```

O ciclo que se segue, é usado para descobrir os parâmetros de um elemento especificado do array. Entra-se no ciclo no ponto médio SV-COUNT, onde o contador de elementos é passado para zero.

O ciclo é acedido «B» vezes, sendo estas, no caso de um array numérico, iguais ao número de dimensões que estão a ser usadas; mas no caso de um array de cadeias, «B» é menos um do que o número de dimensões em uso, porque o último índice é usado para especificar uma «divisão» («slice») da cadeia.

```
29C3 SV-COMMA   PUSH    HL              Guardar o contador.
                RST     0018,GET-CHAR   Obter o caracter actual.
                POP     HL              Restaurar o contador.
                CP      +2C             O caracter presente é um «,»?
                JR      Z,29EA,SV-LOOP  Saltar para diante a fim de
                                        considerar outro índice.
```



```
                BIT     7,C             Se a linha está em execução
                JR      Z,2A20,REPORT-3 existe um erro.
                BIT     6,C             Saltar para diante, se trata
                JR      NZ,29D8,SV-CLOSE um array de cadeias.
                CP      +29             O caracter actual é um «)»?
                JR      NZ,2A12,SV-RPT-C Indicar erro se não.
                RST     0020,NEXT-CHAR  Avançar CH-ADD.
                RET                     Retorno porque sintaxe correcta.
```

No caso de um array de cadeias o índice actual pode representar um «slice», ou pode estar presente o índice de um «slice» na linha Basic.

```
29D8 SV-CLOSE   CP      +29             O caracter actual é um «)»?
                JR      Z,2A48,SV-DIM   Saltar para diante e verifi-
                                        car se existe algum outro
                                        índice.
                CP      +CC             O caracter presente é «TO»?
                JR      NZ,2A12,SV-RPT-C Não deve ser outro.
29E0 SV-CH-ADD  RST     0018,GET-CHAR   Obter o caracter actual.
                DEC     HL              Apontar para o caracter
                LD      (CH-ADD),HL     anterior e definir CH-ADD.
                JR      2A45,SV-SLICE   Avaliar o «slice».
```

Entrar no ciclo, aqui.

```
29E7 SV-COUNT   LD      HL,+0000        Passar o contador para zero.
29EA SV-LOOP    PUSH    HL              Guardar o contador.
                RST     0020,NEXT-CHAR  Avançar CH-ADD.
                POP     HL              Restaurar o contador.
                LD      A,C             Obter o byte discriminador.
                CP      +C0             Saltar se não se verifica a
                JR      NZ,29FB,SV-MULT sintaxe de um array de cadeias.
                RST     0018,GET-CHAR   Obter o caracter actual.
                CP      +29             É um «)»?
                JR      Z,2A48,SV-DIM   Saltar para diante se terminou
                                        contagem de elementos.
                CP      +CC             É um «TO»?
                JR      Z,29E0,SV-CH-ADD Saltar atrás se trata um
                                        «slice».
29FB SV-MULT    PUSH    BC              Guardar o contador de dimen-
                                        sões e o byte
                                        discriminador.
                PUSH    HL              Guardar o contador de elementos.
                CALL    2AEE,DE,(DE+1)  Pôr em DE um tamanho-dimensão.
                EX      (SP),HL         O contador passa para HL e o
                                        indicador da variável vai para o
                                        «stack».
                EX      DE,HL           Contador em DE e tamanho-
                                        -dimensão em HL.
                CALL    2ACC,INT-EXP1   Avaliar o índice seguinte.
                JR      C,2A20,REPORT-3 Dar erro se fora da gama.
                DEC     BC              O resultado da avaliação é de-
                                        crementado porque o contador
                                        deve contar os elementos que
                                        ocorrem antes do elemento indicado.
                CALL    2AF4,GET-HL*DE  Multiplicar o contador pelo
                                        tamanho-dimensão.
                ADD     HL,BC           Somar o resultado de «INT-
                                        -EXP1» ao contador presente.
```

```
                POP     DE          Obter o indicador de variável.
                POP     BC          Obter o número de dimensões e
                                    o byte discriminador.
                DJNZ    29C3,SV-COMMA  Percorrer o ciclo, até B
                                    ser igual a zero.
```

A flag SINTAXE/EXECUÇÃO é verificada antes de os arrays de cadeias serem separados dos arrays de números.

```
2A12 SV-RPT-C   BIT     7,C         Indicar erro se verifica
                JR      NZ,2A7A,SL-RPT-C  sintaxe neste ponto.
                PUSH    HL          Guardar o contador.
                BIT     6,C         Saltar para diante se trata
                JR      NZ,2A2C,SV-ELEM$  um array de cadeias.
```

Ao tratar um array de números o caracter actual deve ser um «)».

```
                LD      B,D         Transferir o indicador da
                LD      C,E         variável para BC.
                RST     0018,GET-CHAR  Obter o caracter actual.
                CP      +29         É um «)»?
                JR      Z,2A22,SV-NUMBER  Saltar para além da mensagem
                                    de erro se não for necessária.
```

Mensagem «3 — Subscript out of range»

```
2A20 REPORT-3   RST     0008,ERROR-1  Invocar a rotina de
                DEFB    +02         tratamento de erro.
```

Pode calcular-se agora o endereço da posição antes da actual forma em vírgula flutuante.

```
2A22 SV-NUMBER  RST     0020,NEXT-CHAR  Avançar CH-ADD.
                POP     HL          Obter contador.
                LD      DE,+0005    Existem 5 bytes para cada
                                    elemento de um array de números.
                CALL    2AF4,GET-HL*DE  Calcular o número total de
                                    bytes antes do elemento
                                    requerido.
                ADD     HL,BC       Fazer HL apontar para a
                                    posição antes do elemento.
                RET                 Retorno com este endereço.
```

Ao tratar um array de cadeias, o comprimento de um elemento é dado pelo último «tamanho-dimensão». Os parâmetros apropriados são assim calculados e passados para o «stack» do calculador.

```
2A2C SV-ELEM$   CALL    2AEE,DE,(DE+1)  Obter último «tamanho-dimensão».
                EX      (SP),HL     O indicador da variável vai para
                                    o «stack» e o contador para HL.
                CALL    2AF4,GET-HL*DE  Multiplicar «contador» por
                                    «tamanho-dimensão».
                POP     BC          Obter o indicador da variável.
                ADD     HL,BC       HL aponta assim para a posição
                                    antes da cadeia.
                INC     HL          Apontar portanto para «início».
                LD      B,D         Transferir último tamanho-dimensão
                LD      C,E         para BC para formar «comprimento».
```



```
                EX      DE,HL       Passar «início» para DE.
                CALL    2AB1,STK-ST-0  Passar estes parâmetros para o
                                    «stack» do calculador. Nota: o
                                    1.º parâmetro é zero, indicando
                                    uma cadeia de um array de cadeias
                                    e, portanto, a entrada existente não
                                    é reclamada.
```

O último índice pode apresentar três formas diferentes. A primeira, é ilustrada por A$(2,4 TO 8); a segunda, por A$(2)(4 TO 8); e a terceira, por A$(2) — forma que indica por defeito que vai ser utilizada a cadeia total.

```
                RST     0018.GET-CHAR  Obter o caracter actual.
                CP      +29         É um «)»?
                JR      Z,2A48,SV-DIM  Saltar, se sim.
                CP      +2C         É um «,»?
                JR      NZ,2A20,REPORT-3  Indicar erro, se não.
2A45 SV-SLICE   CALL    2A52,SLICING  Usar SLICING para alterar
                                    o conjunto de parâmetros.
2A48 SV-DIM     RST     0020,NEXT-CHAR  Obter o caracter seguinte.
2A49 SV-SLICE?  CP      +28         É um «(»?
                JR      Z,2A45,SV-SLICE  Saltar atrás se deve consi-
                                    derar um «slice».
```

Ao acabar de considerar o último índice pode executar-se um retorno.

```
                RES     6,(FLAGS)   Sinalizar «resultado cadeia».
                RET                 Retorno com os parâmetros da
                                    cadeia requerida formando
                                    um «último valor» no «stack»
                                    do calculador.
```

A subrotina «SLICING»

A cadeia actual pode ser dividida («sliced») usando esta subrotina. Entra-se na subrotina com os parâmetros da cadeia presentes no topo do «stack» do calculador e nos registos A, B, C, D e E. Inicialmente, é verificada a flag SINTAXE/EXECUÇÃO, e os parâmetros da cadeia só são recuperados se está a ser executada uma linha.

```
2A52 SLICING    CALL    2530,SYNTAX-Z  Verificar a flag.
                CALL    NZ,2BF1,STK-FETCH  Retirar os parâmetros do
                                    «stack» em execução.
```

É necessário considerar a possibilidade de o «slice» ser «()».

```
                RST     0020,NEXT-CHAR  Obter o caracter seguinte.
                CP      +29         É um «)»?
                JR      Z,2AAD,SL-STORE  Saltar para diante, se sim.
```

Antes de continuar, os registos são manipulados do seguinte modo:

```
                PUSH    DE          O «início» passa para o «stack»-
                                    -máquina.
                XOR     A           O registo A é limpo e
                PUSH    AF          guardado.
```

```
                PUSH    BC          O «comprimento» é guardado.
                LD      DE,+0001    Pressupor que o «slice» deve
                                    começar pelo 1.º caracter.
                RST     0018,GET-CHAR  Obter o 1.º caracter.
                POP     HL          Passar «comprimento» para HL.
```

Avalia-se, agora, o primeiro parâmetro do «slice».

```
                CP      +CC         O caracter actual é um
                                    «TO»?
                JR      Z,2A81,SL-SECOND  O 1.º parâmetro, por defeito,
                                    será um «1» se há salto.
                POP     AF          Nesta fase, A é zero.
                CALL    2ACD,INT-EXP2  BC aponta para o primeiro
                                    parâmetro. A conterá +FF se
                                    houve um erro do tipo «fora
                                    da gama».
                PUSH    AF          Guardar o valor sempre.
                LD      D,B         Transferir o primeiro
                LD      E,C         parâmetro para DE.
                PUSH    HL          Guardar o «comprimento».
                RST     0018,GET-CHAR  Obter o caracter actual.
                POP     HL          Restaurar «comprimento».
                CP      +CC         O caracter actual é «TO»?
                JR      Z,2A81,SL-SECOND  Saltar para diante para
                                    considerar o 2.º parâmetro se
                CP      +29         sim; se não, verificar que
2A7A SL-RPT-C   JP      NZ,1C8A,REPORT-C  existe um parêntesis final.
```

Neste ponto foi já identificado um «slice» de um só caracter, por exemplo, A$(4).

```
                LD      H,D         O último caracter do «slice»
                LD      L,E         é também o 1.º caracter.
                JR      2A94,SL-DEFINE  Saltar para diante.
```

É agora avaliado o segundo parâmetro de um «slice».

```
2A81 SL-SECOND  PUSH    HL          Guardar o «comprimento».
                RST     0020,NEXT-CHAR  Obter caracter seguinte.
                POP     HL          Restaurar o «comprimento».
                CP      +29         O caracter actual é um
                                    «)»?
                JR      Z,2A94,SL-DEFINE  Saltar, se não existe um
                                    segundo parâmetro.
                POP     AF          Se o primeiro parâmetro se
                                    encontrava na gama A conterá
                                    zero; senão, +FF.
                CALL    2ACD,INT-EXP2  BC passa a conter o segundo
                                    parâmetro.
                PUSH    AF          Guardar o «registo de erro».
                RST     0018,GET-CHAR  Obter o caracter actual.
                LD      H,B         Passar o resultado obtido
                LD      L,C         de INT-EXP2 para o par de
                                    registos HL.
                CP      +29         Verificar agora se existe
                JR      NZ,2A7A,SL-RPT-C  um parêntesis final.
```


São agora definidos os «novos» parâmetros.

```
2A94 SL-DEFINE  POP     AF          Obter o «registo de erro».
                EX      (SP),HL     O segundo parâmetro vai para
                                    o «stack» e o «início» para
                                    HL.
                ADD     HL,DE       O primeiro parâmetro é
                                    somado a «início».
                DEC     HL          Voltar atrás uma posição
                                    para obtê-lo certo.
                EX      (SP),HL     O novo «início» vai para o
                                    «stack» e o segundo para
                                    HL.
                AND     A           Subtrair os primeiros
                SBC     HL,DE       parâmetros do 2.º para descobrir
                                    o comprimento do «slice».
                LD      BC,+0000    Inicializar o «novo comprimento».
                JR      C,2AA8,SL-OVER  Um «slice» negativo é uma cadeia
                                    nula e não uma condição de
                                    erro (ver o Manual).
                INC     HL          Ter em conta byte inclusivo.
                AND     A           Só agora testar o «registo
                                    de erro».
                JP      M,2A20,REPORT-3  Saltar, se qualquer dos parâ-
                                    metros está fora da gama.
                LD      B,H         Transferir o «novo compri-
                LD      C,L         mento» para BC.
2AA8 SL-OVER    POP     DE          Obter o «novo início».
                RES     6,(FLAGS)   Garantir que a cadeia é
                                    ainda indicada.
2AAD SL-STORE   CALL    2530,SYNTAX-Z  Retorno neste ponto se veri-
                RET     Z           fica sintaxe; senão, con-
                                    tinuar para a subrotina
                                    STK-STORE.
```

A subrotina «STK-STORE»

Esta subrotina passa os valores guardados nos registos A, B, C, D e E para o «stack» do calculador. O «stack» aumenta portanto de tamanho em cinco bytes por cada invocação desta subrotina.

Esta subrotina é normalmente usada para transferir os parâmetros das cadeias, mas é também usada por «STACK»-BC e LOG (2 ↑ A) para transferir «inteiros pequenos» para o «stack».

Notar que, ao armazenar os parâmetros de uma cadeia, o primeiro valor guardado (vindo do registo A) será um zero se a cadeia vem de um array de cadeias ou é um «slice» de uma cadeia. O valor será «1» para uma cadeia simples completa. Esta «flag» é usada na rotina de comando LET, onde o «1» indica que deve ser «reclamada» a cópia antiga da cadeia.

```
2AB1 STK-ST-0   XOR     A           Sinal – uma cadeia de um
                                    array de cadeias ou uma
                                    cadeia «sliced».
2AB2 STK-STO-$  RES     6,(FLAGS)   Garantir que a flag indica
                                    um resultado de cadeia.
2AB6 STK-STORE  PUSH    BC          Guardar B e C.
```

```
        CALL   33A9,TEST-5-SP   Há espaço para 5 bytes?
                                Não sair aqui a menos que
                                haja espaço disponível.
        POP    BC               Restaurar B e C.
        LD     HL,(STKEND)      Obter o endereço da primeira
                                posição acima do «stack»
                                actual.
        LD     (HL),A           Transferir o 1.º byte.
        INC    HL               Continuar.
        LD     (HL),E           Transferir o segundo e
        INC    HL               terceiro bytes; numa cadeia,
        LD     (HL),D           formarão o «início».
        INC    HL               Continuar.
        LD     (HL),C           Transferir o quarto e
        INC    HL               quinto bytes; no caso de uma
        LD     (HL),B           cadeia, formarão o «comprimento».
        INC    HL               Continuar a fim de apontar
                                para a posição acima do «stack».
        LD     (STKEND),HL      Guardar este endereço em
        RET                     STKEND e retorno.
```

A subrotina «INT-EXP»

Esta subrotina devolve o resultado da avaliação da «expressão seguinte» sob a forma de um valor inteiro guardado no par de registos BC. Esta subrotina verifica ainda este resultado em função de um valor-limite fornecido pelo par de registos HL. A flag «carry» passa ao valor um se ocorre um erro «fora da gama».

O registo A é usado como «registo de erro» e guarda +00 se não existe «erro prévio» e +FF se ocorreu um.

```
2ACC INT-EXP1   XOR    A                Limpar o «registo de erro».
2ACD INT-EXP2   PUSH   DE               Guardar daqui em diante os
                PUSH   HL               registos DE e HL.
                PUSH   AF               Guardar o «registo de
                                        erro» brevemente.
                CALL   1C82,EXPT-1NUM   Avalia a «expressão seguin-
                                        te» de modo a obter um «último
                                        valor» no «stack» do calculador.
                POP    AF               Restaurar o «registo de erro».
                CALL   2530,SYNTAX-Z    Saltar para diante, se verifica
                                        sintaxe.
                JR     Z,2AEB,I-RESTORE Guardar o registo de erro.
                PUSH   AF               Comprimir «último valor»
                CALL   1E99,FIND-INT2   em BC.
                                        Registo de erro em D.
                POP    DE               Uma «expressão seguinte»
                LD     A,B              que dá zero é sempre um erro;
                OR     C                saltar para diante, se assim
                SCF                     for.
                JR     Z,2AE8,I-CARRY   Copiar o valor-limite. Este
                POP    HL               será um «tamanho-dimensão», um
                PUSH   HL               «limite-DIM» ou um «compri-
                                        mento de cadeia».
                AND    A                Comparar o resultado da ava-
                SBC    HL,BC            liação da expressão com o
                                        limite.
```


O estado da flag «carry» e o valor guardado no registo D são agora manipulados de modo a fornecerem o valor apropriado ao «registo de erro».

```
2AEB I-CARRY    LD     A,D              Obter o «valor antigo de erro».
                SBC    A,+00            Formar o «novo valor de erro»;
                                        +00 se erro ausente, +FF se
                                        «fora da gama» nesta passa-
                                        gem ou nas anteriores.
```

Restaurar os registos antes do retorno.

```
2AEB I-RESTORE  POP    HL               Restaurar HL e DE.
                POP    DE
                RET                     Retorno; o «registo de erro»
                                        é o registo A.
```

A subrotina «DE,(DE+1)»

Esta subrotina realiza a construção — LD DE,(DE+1) — e devolve HL apontando para «DE+2».

```
2AEE DE,(DE+1)  EX     DE,HL            Usar HL para a construção.
                INC    HL               Apontar para «DE+1».
                LD     E,(HL)           De facto, — LD E,(DE+1).
                INC    HL               Apontar para «DE+2».
                LD     D,(HL)           De facto, LD D,(DE+2).
                RET                     Final.
```

A subrotina «GET-HL*DE»

A menos que se esteja a verificar a sintaxe, esta subrotina invoca «HL=HL*DE», que executa a construção implícita.

Ao passar os 16 bits disponíveis no par de registos, obtém-se a mensagem «out of memory». A mensagem não retrata a verdadeira situação, mas implica que a memória não é suficientemente extensa para a tarefa considerada pelo utilizador.

```
2AF4 GET-HL*DE  CALL   2530,SYNTAX-Z    Retorno directo se se veri-
                RET    Z                fica sintaxe.
                CALL   30A9,HL=HL*DE    Realizar a multiplicação.
                JP     C,1F15,REPORT-4  Mensagem «Out of Memory».
                RET                     Final.
```

A rotina de comando «LET»

É esta a rotina de atribuição usada nos comandos LET, READ e INPUT.

Quando a variável de destino é uma «variável recém-declarada», DEST apontará para a primeira letra do nome da variável tal como ocorre na linha Basic. O bit 1 de FLAGX passará a um.

No entanto, se a variável de destino «já existe», o bit 1 de FLAGX será passado a zero e DEST apontará no caso de uma variável numérica para

a posição *antes* dos cinco bytes do «número antigo»; no caso de uma variável de cadeia, apontará para a *primeira* posição da «cadeia antiga». O uso de DEST, deste modo, aplica-se a variáveis simples e a elementos de arrays.

O bit 0 de FLAGX é passado a um se a variável de destino é uma variável de cadeia simples «completa» (sinalizando — eliminar cópia antiga).

Inicialmente, é recolhido o valor actual de DEST, e verifica-se o bit 1 de FLAGS.

```
2AFF LET    LD    HL,(DEST)          Obter o endereço actual
                                     em DEST.
            BIT   1,(FLAGX)          Saltar, se trata uma variável
            JR    Z,2B66,L-EXISTS    «já existente».
```

Está a ser usada uma «variável recém-declarada». Descobre-se, portanto, primeiro, o comprimento do nome.

```
            LD    BC,+0005           Pressupor que é uma variável
                                     numérica — 5 bytes.
```

Entrar num ciclo que trata os caracteres de um nome comprido. São ignorados quaisquer espaços ou códigos de cor no nome.

```
2B08 L-EACH-CH  INC   BC             Somar «1» ao contador por
                                     cada caracter do nome.
2B0C L-NO-SP    INC   HL             Percorrer o nome da variável.
                LD    A,(HL)         Obter o «código actual».
                CP    +20            Saltar atrás se é «espaço»;
                JR    Z,2B0C,L-NO-SP ignora portanto os espaços.
                JR    NC,2B1F,L-TEST-CH Saltar para diante se o código
                                     é +21 a +FF.
                CP    +10            Aceitar, como código final,
                JR    C,2B29,L-SPACES os na gama +00 a +0F.
                CP    +16            Aceitar também a gama
                JR    NC,2B29,L-SPACES +16 a +1F.
                INC   HL             Passar além do código de
                                     controlo após INK a OVER.
                JR    2B0C,L-NO-SP   Saltar atrás tratando estes
                                     códigos como espaços.
```

Separar nomes «numéricos» e de «cadeia».

```
2B1F L-TEST-CH  CALL  2C88,ALPHANUM  O código é alfanumérico?
                JR    C,2B08,L-EACH-CH Se for, aceitá-lo como caracter
                                     de um nome comprido.
                CP    +24            O código actual é um «$»?
                JP    Z,2BC0,L-NEW$  Salto para diante para tratar
                                     cadeia simples «recém-declarada».
```

A «variável numérica recém-declarada» que neste momento está a ser tratada necessitará de «BC» espaços na área de variáveis para o seu nome e o seu valor. Este espaço é colocado à sua disposição, sendo o nome da variável copiado com os caracteres «marcados» do modo requerido.

```
2B29 L-SPACES   LD    A,C            Copiar o «comprimento» de A.
                LD    HL,(E-LINE)    Fazer HL apontar para o
                DEC   HL             «byte-80» no final da área
                                     de variáveis.
```



```
                CALL  1655,MAKE-ROOM Abrir agora a área de variáveis.
                                     Nota: de facto, são libertados
                                     «BC» espaços antes do «byte-
                                     -80» deslocado.
                INC   HL             Apontar par o 1.º byte «novo».
                INC   HL             Fazer DE apontar para o segundo
                EX    DE,HL          byte «novo».
                PUSH  DE             Guardar este indicador.
                LD    HL,(DEST)      Obter o indicador do início
                                     do nome.
                DEC   DE             Fazer DE apontar para o 1.º
                                     byte «novo».
                SUB   +06            Fazer B guardar o «número de
                LD    B,A            letras extra» que se encontram
                                     num «nome comprido».
                JR    Z,2B4F,L-SINGLE Saltar para diante, se trata
                                     uma variável com «nome curto».
```

Os códigos «extra» de um nome comprido são passados para a área das variáveis.

```
2B3E L-CHAR     INC   HL             Apontar para cada código «extra».
                LD    A,(HL)         Obter o código.
                CP    +21            Aceitar códigos de +21 a +FF;
                JR    C,2B3E,L-CHAR  ignorar os códigos +00 a +20.
                OR    +20            Passar a um o bit 5, para
                                     minúsculas.
                INC   DE             Transferir os códigos um a um
                LD    (DE),A         para o 2.º byte «novo» e daí em
                                     diante.
                DJNZ  2B3E,L-CHAR    Percorrer o ciclo para todos os
                                     códigos «extra».
```

O último código de um nome «comprido» deve ser «ORed» com +80.

```
                OR    +80            Marcar o código como requerido
                LD    (DE),A         e substituir o anterior.
```

É agora considerada a primeira letra do nome da variável que está a ser tratada.

```
                LD    A,+C0          Preparar a marca para um
                                     «nome comprido».
2B4F L-SINGLE   LD    HL,(DEST)      Obter o indicador da letra.
                XOR   (HL)           A contém +00 no caso de um nome
                                     «curto» e +C0 no de um «com-
                                     prido».
                OR    +20            Passar o bit 5 a 1, para
                                     minúsculas.
                POP   HL             Libertar o indicador.
```

A subrotina L-FIRST é agora invocada para introduzir a «letra» na sua posição apropriada.

```
                CALL  2BEA,L-FIRST   Introduzir a letra e retorno
                                     com HL apontando para «novo
                                     byte-80»
```

O «último valor» pode agora ser transferido para a área das variáveis. Note-se, que neste ponto, HL aponta sempre para a posição após as cinco posições atribuídas ao número.

É usada uma instrução «RST 0028» para invocar o calculador, e é eliminado o «último valor». No entanto, este valor não é substituído.

```
2B59 L-NUMERIC  PUSH  HL           Guardar o indicador «destino».
                RST   0028,FP-CALC  Usar o calculador.
                DEFB  +02,delete    Isto passa STKEND 5 bytes
                DEFB  +38,end-calc  para trás.
                POP   HL            Restaurar o indicador.
                LD    BC,+0005      Dar ao número um «comprimento»
                                    de 5 bytes.
                AND   A             Fazer HL apontar para a
                SBC   HL,BC         primeira das cinco posições
                JR    2BA6,L-ENTER  e saltar para a frente para
                                    fazer a transferência.
```

Vir aqui se se considera uma variável que «já existe». Primeiro, verifica-se o bit 6 de FLAGS a fim de separar as variáveis numéricas das variáveis de cadeia ou arrays de cadeias.

```
2B66 L-EXISTS   BIT   6,(FLAGS)          Saltar para diante, se trata
                JR    Z,2B72,L-DELETE$   qualquer tipo de variável de cadeia.
```

No caso de variáveis numéricas, o «novo» número substitui o número «antigo». É portanto necessário, primeiro, que HL aponte para a posição *após* os 5 bytes da entrada existente. Neste momento, HL aponta para a posição *antes* dos cinco bytes.

```
                LD    DE,+0006      Os 5 bytes de um número +«1».
                ADD   HL,DE         HL aponta agora «depois».
                JR    2B59,L-NUMERIC Saltar atrás para fazer a
                                    transferência.
```

Os parâmetros da variável de cadeia são recuperados, separando-se as cadeias simples completas das cadeias «sliced» e dos arrays de cadeias.

```
2B72 L-DELETE$  LD    HL,(DEST)     Obter o «início». Nota: esta
                                    linha é redundante.
                LD    BC,(STRLEN)   Obter o «comprimento».
                BIT   0,(FLAGX)     Saltar, se trata com uma cadeia
                JR    NZ,2BAF,L-ADD$ simples completa; a cadeia
                                    antiga necessitará de ser
                                    eliminada apenas neste caso.
```

Quando se trata um «slice» de uma cadeia simples existentes, um «slice» de uma cadeia pertencente a um array de cadeias, ou uma cadeia completa de um array de cadeias, são envolvidas duas fases distintas. A primeira consiste em construir a «nova» cadeia na área de trabalho, aumentando-a ou encurtando-a como for necessário. A segunda fase consiste então em copiar a «nova» cadeia para o espaço que lhe foi reservado na área de variáveis.

No entanto, nada se faz no caso de a cadeia não ter «comprimento».

```
                LD    A,B           Retorno, se a cadeia for
                OR    C             vazia.
                RET   Z
```


Reservar então o número de espaços necessários na área de trabalho.

```
                PUSH  HL            Guardar o «início» (DEST).
                RST   0030,BC-SPACES Reservar o espaço necessário
                                    na área de trabalho.
                PUSH  DE            Guardar o indicador da
                                    primeira posição.
                PUSH  BC            Guardar o «comprimento»
                                    para uso mais tarde.
                LD    D,H           Fazer DE apontar para a
                LD    E,L           última posição.
                INC   HL            Fazer HL apontar para «um
                                    depois» das novas posições.
                LD    (HL),+20      Inserir um caracter «espaço».
                LDDR                Copiar este caracter para todas
                                    as posições novas. Terminar
                                    com HL apontando para a
                                    1.ª posição nova.
```

São agora recuperados os parâmetros da cadeia que está a ser tratada, a partir do «stack» do calculador.

```
                PUSH  HL            Guardar o indicador.
                CALL  2BF1,STK-FETCH Obter os «novos» parâmetros.
                POP   HL            Restaurar o indicador.
```

Nota: Neste ponto foi já reservado na área de trabalho, o espaço necessário para a «variável em atribuição»; por exemplo, para o tratamento LET A$(4 TO 8)="abcdefg" — são reservadas cinco posições.

Os parâmetros acima obtidos como «último valor» representam a cadeia que deve ser copiada para as novas posições com aumento ou encurtamento, conforme necessário.

O comprimento da «nova» cadeia é comparado com o do espaço que foi colocado à sua disposição.

```
                EX    (SP),HL       «Comprimento» da nova área para HL.
                                    «Indicador» da nova área para o
                                    «stack».
                AND   A             Comparar os 2 comprimentos e
                SBC   HL,BC         saltar para diante se a cadeia
                ADD   HL,BC         «nova» couber no espaço, isto
                JR    NC,2B9B,L-LENGTH é, não é necessário encurtamento.
                LD    B,H           No entanto, modificar o «novo»
                LD    C,L           comprimento se for excessivo.
2B9B L-LENGTH   EX    (SP),HL       «Comprimento» da nova área
                                    para «stack».
                                    «Indicador» da nova área para HL.
```

Desde que a nova cadeia não seja vazia, é copiada para a área de trabalho. O aumento é realizado automaticamente se a «nova» cadeia for mais curta do que o espaço que lhe está destinado.

```
                EX    DE,HL         «Início» da cadeia nova para HL.
                                    «Indicador» da nova área para DE.
                LD    A,B           Saltar para diante se a
                OR    C             cadeia «nova» é uma cadeia
                JR    Z,2BA3,L-IN-W/S vazia.
```

```
                LDIR                    Senão, passar a cadeia «nova»
                                        para a área de trabalho.
```

Os valores que foram passados para o «stack»-máquina são recuperados.

```
2BA3 L-IN-W/S   POP   BC      «Comprimento» da nova área.
                POP   DE      «Indicador» da nova área.
                POP   HL      «Início» — indicador da
                              «variável em atribuição»
                              que estava em DEST.
                              L-ENTER é usada para
                              passar a «nova» cadeia para
                              a área das variáveis.
```

A subrotina «L-ENTER»

Esta curta subrotina é usada para passar um valor numérico, do «stack» do calculador, ou uma cadeia, da área de trabalho, para a sua posição apropriada na área de variáveis.

Esta subrotina é portanto usada para tudo, excepto cadeias simples «recém-declaradas» e cadeias simples, «completas e existentes».

```
2BA6 L-ENTER    EX    DE,HL     Trocar os indicadores.
                LD    A,B       Verificar mais uma vez se
                OR    C         o comprimento não é zero.
                RET   Z
                PUSH  DE        Guardar o indicador de destino.
                LDIR            Transferir o valor numérico
                                ou a cadeia.
                POP   HL        Retorno com o par de registos
                RET             HL apontando para o primeiro
                                byte do valor numérico ou
                                da cadeia.
```

Continuação da subrotina «LET»

Quando se trata uma cadeia simples «completa e existente», a nova cadeia é introduzida como se fosse uma cadeia simples «recém-declarada» antes de ser «reclamada» a versão existente.

```
2BAF L-ADD$     DEC   HL        Fazer HL apontar para a letra
                DEC   HL        do nome da variável,
                DEC   HL        isto é, DEST — 3.
                LD    A,(HL)    Recolher a letra.
                PUSH  HL        Guardar o indicador da
                                «versão existente».
                PUSH  BC        Guardar o «comprimento»
                                da «cadeia existente».
                CALL  2BC6,L-STRING
                                Usar L-STRING para colocar a
                                nova cadeia na área de variáveis.
                POP   BC        Restaurar o «comprimento».
                POP   HL        Restaurar o indicador.
                INC   BC        Atribuir um byte para a letra
                INC   BC        e dois para o comprimento.
```



```
                INC   BC
                JP    19E8,RECLAIM-2
                                Sair, saltando para RECLAIM-2
                                que reclamará toda a versão
                                existente.
```

As cadeias simples «recém-declaradas» são tratadas do seguinte modo:

```
2BC0 L-NEW$     LD    A,+DF     Preparar para marcar a letra
                                da variável.
                LD    HL,(DEST) Obter o indicador da letra.
                AND   (HL)      Marcar a letra como requerido.
                                L-STRING é agora usada para
                                colocar a nova cadeia na
                                área de variáveis.
```

A subrotina «L-STRING»

São recuperados os parâmetros da «nova» cadeia, é reservado espaço suficiente para ela e transfere-se, em seguida, a cadeia.

```
2BC6 L-STRING   PUSH  AF        Guardar a letra da variável.
                CALL  2BF1,STK-FETCH
                                Obter o «início» e o «compri-
                                mento» da «nova» cadeia.
                EX    DE,HL     Passar o «início» para HL.
                ADD   HL,BC     Fazer HL apontar para «um de-
                                pois» da cadeia.
                PUSH  BC        Guardar o «comprimento».
                DEC   HL        Fazer HL apontar para o
                                final da cadeia.
                LD    (DEST),HL Guardar o indicador.
                INC   BC        Reservar um byte para a letra
                INC   BC        e dois bytes para o comprimento.
                INC   BC
                LD    HL,(E-LINE)
                                Fazer HL apontar para o
                DEC   HL        «byte-80» no final da área
                                de variáveis.
                CALL  1655,MAKE-ROOM
                                Abrir agora a área de variáveis.
                                Nota: de facto, são reservados «BC»
                                espaços antes do «byte-80»
                                deslocado.
                LD    HL,(DEST) Restaurar o indicador para o
                                fim da «nova» cadeia.
                POP   BC        Copiar o comprimento da
                PUSH  BC        «nova» cadeia.
                INC   BC        Acrescentar um ao comprimento
                                no caso de a «nova» cadeia
                                ser uma cadeia «vazia».
                LDDR            Copiar agora a «nova» cadeia
                                mais um byte.
                EX    DE,HL     Fazer HL apontar para o byte
                INC   HL        maior do comprimento.
                POP   BC        Obter o «comprimento».
                LD    (HL),B    Inserir o byte alto do comprimento.
                DEC   HL        Atrasar um.
                LD    (HL),C    Inserir o byte baixo do comprimento.
                POP   AF        Obter a letra da variável.
```

A subrotina «L-FIRST»

Entra-se nesta subrotina com uma letra da variável, convenientemente marcada, no registo A. Esta letra sobrepõe-se ao «antigo byte-80» na área de variáveis. A subrotina devolve o par de registos HL apontando para o «novo byte-80».

```
2BEA L-FIRST    DEC   HL          Fazer HL apontar para o
                                  «antigo byte-80».
                LD    (HL),A      É substituído pela letra
                                  da variável.
                LD    HL,(E-LINE) Fazer HL apontar para o
                DEC   HL          «novo byte-80».
                RET               Termina todas as variáveis
                                  «recém-declaradas».
```

A subrotina «STK-FETCH»

Esta importante subrotina recolhe o «último valor» do «stack» do calculador. Os cinco bytes podem corresponder a um número em vírgula flutuante, em forma «curta» ou «comprida», ou a um conjunto de parâmetros definindo uma cadeia.

```
28F1 STK-FETCH  LD    HL,(STKEND) Obter STKEND.
                DEC   HL          Atrasar um.
                LD    B,(HL)      O quinto valor.
                DEC   HL          Atrasar um.
                LD    C,(HL)      O quarto.
                DEC   HL          Atrasar um.
                LD    D,(HL)      O terceiro.
                DEC   HL          Atrasar um.
                LD    E,(HL)      O segundo valor.
                DEC   HL          Atrasar um.
                LD    A,(HL)      O primeiro.
                LD    (STKEND),HL Redefinir STKEND para a sua
                                  nova posição.
                RET               Final.
```

A rotina de comando «DIM»

Esta rotina define novos arrays na área de variáveis. Começa por procurar na actual área de variáveis por um eventual array com o mesmo nome. Se o encontra, «reclama-o» antes de ser estabelecido o novo array.

Um array novo terá todos os seus elementos a zero, se for numérico, ou iguais a «espaços», se se tratar de um array de cadeias.

```
2C02 DIM        CALL  2882,LOOK-VARS    Investigar a área de variáveis.
2C05 D-RPORT-C  JP    NZ,1C8A,REPORT-C  Dar mensagem C se ocorreu um
                                        erro.
                CALL  2530,SYNTAX-Z     Saltar para diante se está em
                JR    NZ,2C15,D-RUN     «execução».
                RES   6,C               Verificar a sintaxe de arrays de
                                        cadeias como se fossem numéricos.
```



```
                CALL  2996,STK-VAR      Verificar a sintaxe da expressão
                                        entre parêntesis.
                CALL  1BEE,CHECK-END    Passar à declaração seguinte,
                                        se a sintaxe estava correcta.
```

É reclamado um «array existente».

```
2C15 D-RUN      JR    C,2C1F,D-LETTER   Saltar para diante se não
                                        houver já esse array.
                PUSH  BC                Guardar o byte discriminador.
                CALL  1988,NEXT-ONE     Descobrir o início da variável
                                        seguinte.
                CALL  19E8,RECLAIM-2    Reclamar o «array existente».
                POP   BC                Restaurar o byte discriminador.
```

São descobertos os parâmetros iniciais do novo array.

```
2C1F D-LETTER   SET   7,C               Passar o bit 7 do byte dis-
                                        criminador a um.
                LD    B,+00             Passar o contador de dimensões
                                        a zero.
                PUSH  BC                Guardar o contador e o byte
                                        discriminador.
                LD    HL,+0001          O par de registos HL guarda a
                BIT   6,C               dimensão dos elementos do array,
                JR    NZ,2C2D,D-SIZE    «1» no caso de um array de
                LD    L,+05             cadeias/«5», se numérico.
2C2D D-SIZE     EX    DE,HL             Tamanho dos elementos em DE.
```

O ciclo que se segue é acedido para cada dimensão especificada na expressão entre parêntesis da declaração DIM. O número total de bytes requeridos para os elementos do array é formado no par de registos DE.

```
2C2E D-NO-LOOP  RST   0020,NEXT-CHAR   Avançar CH-ADD em cada passa-
                                        gem.
                LD    H,+FF             Definir um «valor-limite».
                CALL  2ACC,INT-EXP1     Avaliar um parâmetro.
                JP    C,2A20,REPORT-3   Dar um erro se «fora da gama».
                POP   HL                Obter contador-dimensões e
                                        byte discriminador.
                PUSH  BC                Guardar o parâmetro em cada
                                        passagem pelo ciclo.
                INC   H                 Aumentar o contador-dimensões
                                        também em cada passagem.
                PUSH  HL                «Stack» o contador-dimensões
                                        e o byte discriminador.
                LD    H,B               O parâmetro é passado para
                LD    L,C               o par de registos HL.
                CALL  2AF4,GET-HL*DE    O byte total é formado em
                EX    DE,HL             HL e transferido para
                                        DE.
                RST   0018,GET-CHAR     Obter o caracter actual e
                CP    +2C               percorrer de novo o ciclo se
                JR    Z,2C2E,D-NO-LOOP  existe outra dimensão.
```

Nota: Neste ponto o par de registos DE indica o número de bytes requerido para os elementos do novo array, sendo guardado no «stack»-máquina o tamanho de cada dimensão.

Verifica-se agora se existe de facto um parêntesis no final da expressão entre parêntesis.

```
CP      +29                 É um ')'?
JR      NZ,2C05,D-RPORT-C   Saltar atrás, se não.
RST     0020,NEXT-CHAR      Avançar CH-ADD para além dele.
```

São tomados em conta os tamanhos das dimensões.

```
POP     BC                  Obter o contador-dimensões
                            e o byte discriminador.
LD      A,C                 Passar o byte discriminador
                            para o registo A.
LD      L,B                 Passar o contador para L.
LD      H,+00               Limpar o registo H.
INC     HL                  Aumentar o contador-dimen-
INC     HL                  sões de 2 e duplicar o resul-
ADD     HL,HL               tado, formando o comprimento
ADD     HL,DE               total correcto da variável por
                            soma do total de bytes dos elementos.
JP      C,1F15,REPORT-4     Dar mensagem «Out of memory»,
                            se necessário.
PUSH    DE                  Guardar o total de bytes-elementos.
PUSH    BC                  Guardar contador-dimensões
                            e byte discriminador.
PUSH    HL                  Guardar comprimento total.
LD      B,H                 Passar este para BC.
LD      C,L
```

É reservada a quantidade de memória requerida para o novo array no fim da área de memória.

```
LD      HL,(E-LINE)         Fazer o par de registos HL
DEC     HL                  apontar para o «byte-80».
CALL    1655,MAKE-ROOM      O espaço é reservado.
INC     HL                  HL aponta para a primeira
                            posição nova.
```

São agora introduzidos os parâmetros.

```
LD      (HL),A              A letra, adequadamente marcada,
                            é inserida primeiro.
POP     BC                  O comprimento global é obtido
DEC     BC                  e diminuído de «3».
DEC     BC
DEC     BC
INC     HL                  Avançar HL.
LD      (HL),C              Inserir byte baixo de comprimento.
INC     HL                  Avançar HL.
LD      (HL),B              Inserir byte alto.
POP     BC                  Obter contador de dimensões.
LD      A,B                 Passar para o registo A.
INC     HL                  Avançar HL.
LD      (HL),A              Inserir contagem de dimensões.
```

São agora «limpos» os elementos do novo array.

```
LD      H,D                 HL é levado a apontar para
LD      L,E                 a última posição do array
```



```
DEC     DE                  e DE para a posição anterior
                            a essa.
LD      (HL),+00            Inserir um zero na última
BIT     6,C                 posição, mas substitul-lo por
JR      Z,2C7C,DIM-CLEAR    um «espaço» se se trata um
LD      (HL),+20            array de cadeias.
2C7C DIM-CLEAR
POP     BC                  Obter o total de bytes-elementos.
LDDR                        Limpar o array+uma posição
                            extra.
```

São agora introduzidos os «tamanhos-dimensões».

```
2C7F DIM-SIZES
POP     BC                  Obter um tamanho-dimensão.
LD      (HL),B              Inserir o byte alto.
DEC     HL                  Atrasar um.
LD      (HL),C              Inserir o byte baixo.
DEC     HL                  Atrasar um.
DEC     A                   Diminuir o contador de
                            dimensões.
JR      NZ,2C7F,DIM-SIZES   Repetir a operação até
                            terem sido consideradas todas
RET                         as dimensões; retorno.
```

A subrotina «ALPHANUM»

Esta subrotina termina com a flag «carry» a um se o valor actual do registo A indicar um algarismo ou letra válidos.

```
2C88 ALPHANUM
CALL    2D1B,NUMERIC        Verificar se é algarismo, o
                            que passará a «carry» a zero.
CCF                         Complementar a flag «carry».
RET     C                   Retorno, se é algarismo; se-
                            não, continuar para «ALPHA».
```

A subrotina «ALPHA»

Esta subrotina termina com a flag «carry» em um se o valor actual do registo A indica uma letra válida do alfabeto.

```
2C8D ALPHA
CP      +41                 Comparar com 41 hex,
                            código de «A».
CCF                         Complementar a flag «carry».
RET     NC                  Retorno, se não é um código
                            válido.
CP      +5B                 Comparar com 5B hex, um mais
                            do que o código de «Z».
RET     C                   Retorno, se é maiúscula.
CP      +61                 Comparar com 61 hex, código
                            de «a».
CCF                         Complementar a flag «carry».
RET     NC                  Retorno, se não é um código
                            válido de caracter.
CP      +7B                 Comparar com 7B hex, mais
                            um do que o código de «z».
RET                         Final.
```

A subrotina «Decimal para vírgula flutuante»

Como parte da verificação de sintaxe, os números decimais que ocorrem numa linha Basic são convertidos para as suas formas em vírgula flutuante. Esta subrotina lê o número decimal, algarismo a algarismo, e apresenta o seu resultado como um «último valor» no «stack» do calculador. Mas primeiro trata a notação alternativa BIN, que introduz uma sequência de zeros e uns dando a representação binária do número requerido.

```
2C9B DEC-TO-FP  CP    +C4              O caracter é um «BIN»?
                JR    NZ,2CB8,NOT-BIN  Saltar, se não for «BIN».
                LD    DE,+0000         Inicializar resultado para 0 em DE.
2CA2 BIN-DIGIT  RST   0020,NEXT-CHAR   Obter o caracter seguinte.
                SUB   +31              Subtrair o código de caracter
                                       de «1».
                ADC   A,+00            0 dá agora 0 com «carry» a um;
                                       1 dá 0 com «carry» em zero.
                JR    NZ,2CB3,BIN-END  Qualquer outro caracter produz
                                       um salto para BIN-END e verá a
                                       sintaxe verificada durante ou
                                       após o «scanning».
                EX    DE,HL            O resultado actual em HL, baixo.
                CCF                    Complementar a flag «carry».
                ADC   HL,HL            Desloca o resultado para a
                                       esquerda, passando a
                                       «carry» para o bit 0.
                JP    C,31AD,REPORT-6  Indicar excesso se mais de
                                       65535.
                EX    DE,HL            Reenvia o resultado actual para DE.
                JR    2CA2,BIN-DIGIT   Salto atrás para o 0 ou 1 seguinte.
2CB3 BIN-END    LD    B,D              Copia resultado em BC,
                LD    C,E              para «stack».
                JP    2D2B,STACK-BC    Saltar para diante para guardar
                                       o resultado no «stack».
```

No caso de outros números, é primeiramente convertida qualquer parte inteira; se o caracter seguinte é um decimal, é considerada a fracção decimal.

```
2CB8 NOT-BIN    CP    +2E              O 1.º caracter é um «.»?
                JR    Z,2CCB,DECIMAL   Se sim, salto para diante.
                CALL  2D3B,INT-TO-FP   Senão, formar um «último
                                       valor» do inteiro.
                CP    +2E              O caracter seguinte é «.»?
                JR    NZ,2CEB,E-FORMAT Salto adiante para verificar
                                       se é «E».
                RST   0020,NEXT-CHAR   Obter caracter seguinte.
                CALL  2D1B,NUMERIC     É um algarismo?
                JR    C,2CEB,E-FORMAT  Saltar, se não (p. ex., é
                                       possível 1.E4).
                JR    2CD5,DEC-STO-1   Saltar para diante para tratar
                                       os algarismos após vírgula decimal.
2CCB DECIMAL    RST   0020,NEXT-CHAR   Se o número começa por vírgula,
                CALL  2D1B,NUMERIC     ver se o caracter seguinte é
                                       um algarismo.
2CCF DEC-RPT-C  JP    C,1C8A,REPORT-C  Indicar erro se não.
                RST   0028,FP-CALC     Usar o calculador para guardar
                DEFB  +A0,stk-zero     zero como parte inteira destes
                DEFB  +38,fim-calc     números.
2CD5 DEC-STO-1  RST   0028,FP-CALC     Usar o calculador novamente.
```



```
                DEFB  +A1,stk-um       Descobrir a forma de vírgula
                DEFB  +C0,st-mem-0     flutuante do decimal «1», e
                DEFB  +02,apagar       guardá-la na área de memória.
                DEFB  +38,fim-calc
2CDA NXT-DGT-1  RST   0018,GET-CHAR    Obter o caracter actual.
                CALL  2D22,STK-DIGIT   Se é um algarismo, guardá-lo.
                JR    C,2CEB,E-FORMAT  Senão, saltar para diante.
                RST   0028,FP-CALC     Usar agora o calculador.
                DEFB  +E0,obter-mem-0  Em cada passagem no ciclo,
                DEFB  +A4,stk-dez      o número guardado em memória
                DEFB  +05,dividir      é obtido, dividido por 10 e
                DEFB  +C0,st-mem-0     guardado; passa de .1 para
                                       .01, .001, etc.
                DEFN  +04,multiplicar  O algarismo actual é multi-
                DEFB  +0F,somar        plicado pelo «n.º guardado»
                DEFB  +38,fim-calc     e somado a «último valor».
                RST   0020,NEXT-CHAR   Obter o caracter seguinte.
                JR    2CDA,NXT-DGT-1   Saltar atrás (mais um byte do
                                       que necessário) para o considerar.
```

Ter em conta, seguidamente, qualquer «notação E», isto é, a forma xEm, ou xem, onde m é um inteiro positivo ou negativo.

```
2CEB E-FORMAT   CP    +45              O caracter actual é um «E»?
                JR    Z,2CF2,SIGN-FLAG Saltar para diante, se sim.
                CP    +65              É um «e»?
                RET   NZ               Terminar, se não for.
2CF2 SIGN-FLAG  LD    B,+FF            Usar B como flag de sinal (FF=+).
                RST   0020,NEXT-CHAR   Obter o caracter seguinte.
                CP    +2B              É um «+»?
                JR    Z,2CFE,SIGN-DONE Saltar para diante.
                CP    +2D              É um «-»?
                JR    NZ,2CFF,ST-E-PART Saltar, se não é «+» nem «-».
                INC   B                Mudar o sinal da flag.
2CFE SIGN-DONE  RST   0020,NEXT-CHAR   Apontar para o 1.º algarismo.
2CFF ST-E-PART  CALL  2D1B,NUMERIC     É de facto um algarismo?
                JR    C,2CCF,DEC-RPT-C Indicar erro, se não.
                PUSH  BC               Guardar a flag em B.
                CALL  2D3B,INT-TO-FP   Guardar em «stack» ABS m, onde
                                       m é o expoente.
                CALL  2DD5,FP-TO-A     Transferir ABS m para A.
                POP   BC               Passar a flag de sinal para B.
                JP    C,31AD,REPORT-6  Indicar o excesso se ABS m é
                AND   A                maior do que 255 ou, aliás,
                JP    M,31AD,REPORT-6  maior do que 127 (os valores
                                       maiores do que cerca de 39
                                       serão detectados mais tarde).
                INC   B                Verificar a flag de sinal em B;
                                       «+» (isto é, +FF) passa a um a
                                       flag «sinal».
                JR    Z,2D18,E-FP-JUMP Saltar, se sinal de m é «+».
                NEG                    Negar m se sinal é «-».
2D18 E-FP-JUMP  JP    2D4F,E-TO-FP     Saltar para atribuir a «último
                                       valor» o resultado de x*10↑m.
```

A subrotina «Numérico»

Esta subrotina termina com a flag «carry» em zero se o valor actual do registo A indica um algarismo válido.

```
2D1B NUMERIC   CP    +30      Comparar com 30 hex, código
                              de «0».
               RET   C        Retorno, se não é um código
                              de caracter válido.
               CP    +3A      Comparar com o limite superior.
               CCF            Complementar a flag «carry».
               RET            Final.
```

A subrotina «STK-DIGIT»

Esta subrotina limita-se a efectuar um retorno no caso de o valor actual no registo A não representar um algarismo; mas se representa, a forma deste algarismo em vírgula flutuante passa a «último valor» no «stack» do calculador.

```
2D22 STK-DIGIT CALL  2D1B,NUMERIC  O caracter é um algarismo?
               RET   C             Retorno, se não é aceitável.
               SUB   +30           Substituir o código pelo
                                   algarismo exacto.
```

A subrotina «'STACK¹A»

Esta subrotina produz a forma em vírgula flutuante do valor binário absoluto actualmente guardado no registo A.

```
2D28 STACK-A   LD    C,A       Transferir o valor para o
                               registo C.
               LD    B,+00     Limpar o registo B.
```

A subrotina «'STACK¹BC»

Esta subrotina produz a forma de vírgula flutuante do valor binário absoluto actualmente guardado no par de registos BC.

A forma usada nesta e, portanto, nas duas subrotinas anteriores é a reservada no Spectrum aos inteiros pequenos n, onde — 65535<=n<=65535. O primeiro e o quinto bytes são passados a zero; o terceiro e o quarto bytes são o menos significativo e o mais significativo do inteiro n com 16 bits, na forma «complemento para dois» (se n é negativo, estes bytes contêm 65536+n); e o segundo byte é o byte de sinal, 00 para «+» e FF para «-».

```
2D2B STACK-BC  LD    IY,+5C3A  Reinicializar IY para ERR-NR.
               XOR   A         Limpar o registo A.
               LD    E,A       E o registo E, para indicar
                               «+».
               LD    D,C       Copiar o byte menos signifi-
                               cativo para D.
               LD    C,B       E o mais significativo para
                               C.
               LD    B,A       Limpar o registo B.
               CALL  2AB6,STK-STORE  Guardar o número.
               RST   0028,FP-CALC    Fazer HL apontar para
               DEFB  +38,fim-calc    STKEND-5.
               AND   A         Limpar a flag «carry».
               RET             Final.
```


A subrotina «Inteiro para vírgula flutuante»

Esta subrotina produz um «último valor» no «stack» do calculador que é o resultado da conversão de um inteiro presente numa linha Basic, ou seja, a parte inteira do número decimal ou número de linha, para a sua forma de vírgula flutuante.

Repetidos chamamentos a CH-ADD-1 permitem obter cada um dos algarismos do inteiro original. Sai-se da subrotina quando é encontrado um código que não representa um algarismo.

```
2D3B INT-TO-FP PUSH  AF          Guardar o 1.º algarismo em A.
               RST   0028,FP-CALC  Usar o calculador.
               DEFB  +A0,stk-zero  O «último valor» é zero.
               DEFB  +38,fim-calc
               POP   AF          Restaurar o 1.º algarismo.
```

É agora construído um ciclo. Enquanto o código representar um algarismo, este ciclo determina a forma em vírgula flutuante e guarda-a como «último valor». Este «último valor» é depois multiplicado pelo número 10 (decimal) e somado ao «algarismo», de modo a formar um novo «último valor» que é depois usado para a passagem seguinte pelo ciclo.

```
2D40 NXT-DGT-2 CALL  2D22,STK-DIGIT  Se o código representa um
               RET   C               algarismo, guardar a sua
                                     forma em vírgula flutuante.
               RST   0028,FP-CALC    Usar o calculador.
               DEFB  +01,trocar      «Algarismo» passa a «último va-
                                     lor».
               DEFB  +A4,stk-dez     Definir 10 decimal.
               DEFB  +04,multiplicar «Último valor»=«último valor»*10.
               DEFB  +0F,somar       «Último valor»=«último valor»+
                                     +«algarismo».
               DEFB  +38,fim-calc
               CALL  0074,CH-ADD+1   O código seguinte vai para A.
               JR    2D40,NXT-DGT-2  Retomar o ciclo com este código.
```

9
AS ROTINAS ARITMÉTICAS

A subrotina «Formato-E para virgula flutuante»
(Deslocamento 3C — ver CALCULATE, mais abaixo: «E-TO-FP»)

Esta subrotina produz um «último valor» no topo do «stack» do calculador que é o resultado da conversão de um número indicado na forma xEm, onde m é um inteiro positivo ou negativo. Entra-se na subrotina com x no topo do «stack» do calculador e m no registo A.

O método usado consiste em descobrir o valor absoluto de m, digamos p, e em multiplicar ou dividir x por 10↑p, conforme m é positivo ou negativo.

Para conseguir isto, p é rodado para a direita até ser zero, e x é multiplicado ou dividido por 10↑(2↑n) por cada bit b(n), ao valor um, de p. Como p nunca é muito mais do que o decimal 39, os bits 6 e 7 não se encontrarão geralmente ao valor um.

```
2D4F E-TO-FP    RLCA                    Verificar o sinal de m
                RRCA                    rodando o bit 7 de A para a
                                        «carry» sem alterar A.
                JR      NC,2D55,E-SAVE  Saltar, se m é positivo.
                CPL                     Negar m em A sem perturbar
                INC     A               a flag «carry».
2D55 E-SAVE     PUSH    AF              Guardar m em A.
                LD      HL,+5C92        Isto é MEMBOT: é agora
                CALL    3508,FP-0/1     guardada uma flag de sinal no 1.º
                                        byte de mem-0, 0 para «+» e
                                        1 para «-».
                RST     0028,FP-CALC    O «stack» contém x.
                DEFB    +A4,stk-dez     x, 10 (decimal).
                DEFB    +38,fim-calc    x,10
                POP     AF              Restaurar m em A.
                SRL     A
2D60 E-LOOP                             No ciclo, obter o bit se-
                                        guinte de m, modificando as
                                        flags «carry» e «zero» do modo
                JR      NC,2D71,E-TST-END apropriado; salto, se «carry»=0.
                PUSH    AF              Guardar o resto de m e as flags.
                RST     0028,FP-CALC    O «stack» contém x' e
                                        10↑(2↑n), onde x' é uma
                                        fase intermédia na multiplica-
                                        ção de x por 10↑m, e n é igual
                                        a 0, 1, 2, 3, 4, 5.
                DEFB    +C1,st-mem-1    (10↑(2↑n) é copiado para mem-1).
                DEFB    +E0,obter-mem-0 x', 10↑(2↑n), (1/0)
                DEFB    +00,saltar-verdade x', 10↑(2↑n)
                DEFB    +04,para E-DIVSN x', 10↑(2↑n)
```



```
                DEFB    +04,multiplicar x' • 10↑(2↑n)
                DEFB    +33,saltar      x''
                DEFB    +02, para E-FETCH x''
2D6D E-DIVSN    DEFB    +05,dividir     x'/10↑(2↑n)=x'' (x'' é N' • 10↑
                                        (2↑n) ou x'/10↑(2↑n)
                                        conforme m é «+» ou «-»).
2D6E E-FETCH    DEFB    +E1,obter-mem-1 x'', 10↑(2↑n)
                DEFB    +38,fim-calc    x'', 10↑(2↑n)
                POP     AF              Restaurar o resto de m em A, e
                                        as flags.
2D71 E-TST-END  JR      Z,2D7B,E-END    Saltar se m foi reduzido
                                        a zero.
                PUSH    AF              Guardar o resto de m em A.
                RST     0028,FP-CALC    x'', 10↑(2↑n)
                DEFB    +31,copiar      x'', 10↑(2↑n),10↑(2↑n)
                DEFB    +04,multiplicar x'', 10↑(2↑(n+1))
                DEFB    +38,fim-calc    x'', 10↑(2↑(n+1))
                POP     AF              Restaurar o resto de m em A.
                JR      2D60,E-LOOP     Atrasar para todos os bits de m.
2D7B E-END      RST     0028,FP-CALC    Usar o calculador para
                DEFB    +02,apagar      eliminar a última potência de 10
                DEFB    +38,fim-calc    obtida, deixando «último valor»
                RET                     x • 10↑m no «stack».
```

A subrotina «INT-FETCH»

Esta subrotina recolhe em DE um pequeno inteiro n de valor compreendido entre -65535 e +65535 a partir da posição endereçada por HL; ou seja, n é normalmente o primeiro (ou segundo) número no topo do «stack» do calculador; mas HL pode também aceder (por troca com DE) um número que foi eliminado do «stack».

A subrotina não limpa, de facto, o número do «stack» ou da memória; devolve HL apontado para o quarto byte do número na sua posição original.

```
2D7F INT-FETCH  INC     HL              Apontar para o byte de sinal
                                        do número.
                LD      C,(HL)          Copiar o byte de sinal para C.
```

O mecanismo seguinte executará a «complementação para dois» no caso de o número ser negativo (C é FF), mas deixá-lo-á na mesma se for positivo (C é 00).

```
                INC     HL              Apontar para o byte menos
                                        significativo.
                LD      A,(HL)          Recolher o byte em A.
                XOR     C               Complemento para um, se
                                        negativo.
                SUB     C               Soma 1 no caso de n.ºs nega-
                                        tivos; passa «carry» a um
                                        se byte não era zero.
                LD      E,A             Byte baixo para E.
                INC     HL              Apontar para o byte mais
                                        significativo.
                LD      A,(HL)          Recolhê-lo em A.
                ADC     A,C             Terminar a complementação
                XOR     C               para 2 se o número é negativo;
```

```
                LD      D,A             notar que a «carry» fica
                                        sempre a zero.
                RET                     Byte alto para D agora.
                                        Final.
```

A subrotina «INT-STORE»

Esta subrotina guarda um inteiro pequeno n (compreendido entre — 65535 e + 65535) na posição endereçada por HL e nas quatro posições seguintes; ou seja, n substitui o primeiro (ou segundo) número no topo do «stack» do calculador.

A subrotina termina com HL apontado para o primeiro byte de n no «stack».

```
2D8C P-INT-STO  LD      C,+00           Este ponto de entrada guarda um
                                        número que se sabe ser positivo.
2D8E INT-STORE  PUSH    HL              Guarda o indicador da primeira
                                        posição.
                LD      (HL),+00        O 1.º byte passa a zero.
                INC     HL              Aponta para a 2.ª posição.
                LD      (HL),C          Inserir o byte de sinal.
```

Usa-se agora o mesmo mecanismo que em «INT-FETCH» para complementar para dois os números negativos. Isto é necessário, por exemplo, antes e depois da multiplicação de números inteiros. A soma é, no entanto, realizada sem complementação para dois antes ou depois.

```
                INC     HL              Apontar para a terceira posição.
                LD      A,E             Recolher o byte menos signi-
                                        ficativo.
                XOR     C               Complementar para dois se o
                SUB     C               número é negativo.
                LD      (HL),A          Guardar o byte.
                INC     HL              Apontar para a 4.ª posição.
                LD      A,D             Recolher o byte mais
                                        significativo.
                ADC,    A,C             Complementar para dois se o
                XOR     C               número é negativo.
                LD      (HL),A          Guardar o byte.
                INC     HL              Apontar para a 5.ª posição.
                LD      (HL),+00        O 5.º byte é passado a zero.
                POP     HL              Retorno com HL apontando para
                RET                     o 1.º byte de n no «stack».
```

A subrotina «Vírgula flutuante para BC»

Esta subrotina é invocada de quatro pontos diferentes para diversos fins, sendo usada para comprimir o «último valor» em vírgula flutuante para o par de registos BC.

Se o resultado é excessivamente grande, ou seja, superior a 65535 em decimal, a subrotina termina com a flag «carry» em um. Se o «último valor» é negativo, a flag «zero» passa a zero.

O byte baixo do resultado é, por outro lado, copiado para o registo A.


```
2DA2 FP-TO-BC   RST     0028,FP-CALC    Usar o calculador para fazer
                DEFB    +38,fim-calc    HL apontar para STKEND-5.
                LD      A,(HL)          Recolher o byte expoente
                AND     A               do «último valor»; saltar,
                JR      Z,2DAD,FP-DELETE se for 0 (inteiro pequeno).
                RST     0028,FP-CALC    Usar o calculador para arre-
                DEFB    +A2,stk-meio    dondar «último valor» para inteiro
                DEFB    +0F,somar       + próximo, passando-o também a
                DEFB    +27,int         «inteiro pequeno» no «stack» do
                DEFB    +38,fim-calc    calculador se for possível, ou
                                        seja, se — 65535,5 <=x e
                                        x <:65535,5
2DAD FP-DELETE  RST     0028,FP-CALC    Usar o calculador para eliminar
                DEFB    +02,apagar      o inteiro do «stack»; DE aponta
                DEFB    +38,fim-calc    ainda para ele em memória (em
                                        STKEND).
                PUSH    HL              Guardar os indicadores de «stack».
                PUSH    DE
                EX      DE,HL           HL aponta para o número.
                LD      B,(HL)          Copiar o 1.º byte para B.
                CALL    2D7F,INT-FETCH  Copiar os bytes 2, 3 e 4 para
                                        C, E e D.
                XOR     A               Limpar o registo A.
                SUB     B               «Carry» a um a menos que
                                        B seja zero.
                BIT     7,C             Isto passa a 1 a flag «zero» se
                                        o número é positivo (NZ indica
                                        negativo).
                LD      B,D             Copiar o byte alto para B.
                LD      C,E             E o byte baixo para C.
                LD      A,E             Copiar o byte baixo também para A.
                POP     DE              Restaurar os indicadores de
                                        «stack».
                POP     HL
                RET                     Final.
```

A subrotina «LOG (2↑A)»

Esta subrotina é invocada pela subrotina «PRINT-FP» a fim de calcular o número aproximado de algarismos antes da vírgula decimal em x, o número a imprimir ou, se não existirem algarismos antes da vírgula decimal, o número aproximado de zeros após a vírgula. Entra-se nela com o registo A contendo e', o expoente verdadeiro de x, ou e' — 2, e calcula z (logaritmo decimal) de (2↑A). Torna, em seguida, A igual a ABS INT (z+0,5), como pretendido, usando FP-TO-A para este fim.

```
2DC1 LOG (2↑A)  LD      D,A             Guarda o inteiro A, quer sob
                RLA                     a forma 00 00 A 00 00 (A posi-
                SBC     A,A             tivo) ou 00 FF A FF 00 (A ne-
                                        gativo).
                LD      E,A             Estes bytes são guardados em
                LD      C,A             A, E, D, C, B, e depois é
                XOR     A               chamada STK-STORE para pôr
                LD      B,A             o número no «stack» do
                                        calculador.
                CALL    2AB6,STK-STORE  Usa o calculador.
                RST     0028,FP-CALC    Guardado log 2 na base
                DEFB    +34,stk-dado    decimal.
```

```
        DEFB  +EF,expoente+7F     O «stack» contém A, log 2.
        DEFB  +1A,+20,+9A,+85
        DEFB  +04,multiplicar     A=log 2, isto é, log (21A)
        DEFB  +27,int             INT log (21A)
        DEFB  +38,fim-calc
```

A subrotina continua para FP-TO-A a fim de terminar o cálculo.

A subrotina «Vírgula flutuante para A»

Esta subrotina curta mas vital é invocada pelo menos oito vezes, para diversos fins. Utiliza a subrotina FP-TO-BC, também muito importante, para obter o «último valor» no registo A quando possível. Verifica, em seguida, se o módulo do número arredonda para mais de 255, e se tal acontecer, termina com a flag «carry» em um. Senão, termina com o módulo do número, arredondado para o inteiro mais próximo, no registo A, e a flag «zero» a um para indicar que o número é positivo, ou em zero para indicar que é negativo.

```
2DD5 FP-TO-A    CALL  2DA2,FP-TO-BC     Comprimir o «último valor»
                                        em BC.
                RET   C                 Retorno, se está fora da gama.
                PUSH  AF                Guardar resultado e flags.
                DEC   B                 Se B não contém zero, o valor
                INC   B                 encontra-se novamente fora
                                        da gama.
                JR    Z,2DE1,FP-A-END   Salto, se o número é válido.
                POP   AF                Obter o resultado e as flags.
                SCF                     Sinal «resultado fora da gama».
                RET                     Terminado — sem êxito.
2DE1 FP-A-END   POP   AF                Obter o resultado e as flags.
                RET                     Terminado — com êxito.
```

A subrotina «Imprimir um número em vírgula flutuante»

Esta subrotina é invocada pela ordem PRINT em 2039 e por STR$ em 3630, que converte para cadeia o número que deveria ser impresso. A subrotina imprime x, o «último valor» do «stack» do calculador. O formato de impressão nunca ocupa mais de 14 espaços.

Os oito algarismos mais significativos de x, correctamente arredondado, são guardados num buffer de impressão *ad hoc* em mem-3 e mem-4. Os pequenos números, inferiores a 1, e os números grandes, superiores a 2127, são tratados separadamente. Os primeiros são multiplicados por 10↑n, sendo n o número aproximado de zeros após a vírgula decimal, enquantro os últimos são divididos por 10↑(n-7), onde n é o número aproximado de algarismos antes da vírgula decimal. Isto traz todos os números para a gama intermédia, e o número de algarismos antes da decimal é formado no segundo byte de mem-5. Finalmente é realizada a impressão, usando o formato E se existem mais de oito algarismos antes da vírgula ou, no caso dos números pequenos, mais de quatro zeros após a vírgula.

O programa seguinte mostra a gama de formatos de impressão:

```
10 FOR a= — 11 TO 12: PRINT SGN a*9↑a,: NEXT a
```


1. Primeiramente, é tratado o sinal de x:

Se x é negativo, a subrotina salta para PF-NEGATIVE, considera ABS x e imprime o sinal menos.

Se x é zero, é eliminado do «stack» do calculador, é impresso um «0» e feito o retorno.

Se x é positivo, a subrotina continua.

```
2DE3 PRINT-FP   RST   0028,FP-CALC      Usar o calculador.
                DEFB  +31,copiar        x,x
                DEFB  +36,menos-0       x,(1/0) Valor lógico de x.
                DEFB  +00,saltar-verdade  x
                DEFB  +0B, para PF-NEGTVE  x
                DEFB  +31,copiar        x,x
                DEFB  +37,maior-0       x(1/0) Valor lógico de x.
                DEFB  +00,saltar-verdade  x
                DEFB  +0D, para PF-POSTVE  x Daqui em diante x'=ABS x.
                DEFB  +02,apagar        -
                DEFB  +38,fim-calc      -
                LD    A,+30             Indicar código de caracter de «0».
                RST   0010,PRINT-A-1    Imprimir o zero.
                RET                     Final, porque «último valor»
                                        é zero.
2DF2 PF-NEGTVE  DEFB  +2A,abs           x' x'=ABS x.
                DEFB  +38,fim-calc      x'
                LD    A,+2D             Inserir código de «-».
                RST   0010,PRINT-A-1    Imprimir «-».
                RST   0028,FP-CALC      Usar de novo o calculador.
2DF8 PF-POSTVE  DEFB  +A0,stk-zero      Os 15 bytes de mem-3, mem-4
                DEFB  +C3,st-mem-3      e mem-5 são agora inicializados para zero para uso como
                DEFB  +C4,st-mem-4
                DEFB  +C5,st-mem-5      buffer e 2 contadores.
                DEFB  +02,apagar        O «stack» é limpo, excepto
                                        quanto a x'.
                DEFB  +38,fim-calc      x'
                EXX                     H'L', usado para guardar
                PUSH  HL                deslocamentos do calculador
                EXX                     (p. ex. para STR$), é guardado
                                        no «stack»-máquina.
```

2. Este é o início de um ciclo que trata números grandes. No entanto, qualquer número pequeno x é primeiro dividido na sua parte inteira i e na parte fraccionária f. Se se trata de um inteiro pequeno, ou seja, se — 65535 <=|x|<=65535, é guardado em D'E' para inserção no buffer da impressora.

```
2E01 PF-LOOP    RST   0028,FP-CALC      Usar de novo o calculador.
                DEFB  +31,copiar        x',x'
                DEFB  +27,int           x',INT (x')=i
                DEFB  +C2,st-mem-2      (i é guardado em mem-2).
                DEFB  +03,subtrair      x'—i=f
                DEFB  +E2,obter-mem-2   f,i
                DEFB  +01,trocar        i,f
                DEFB  +C2,st-mem-2      (f é guardado em mem-2).
                DEFB  +02,apagar        i
                DEFB  +38,fim-calc      i
                LD    A,(HL)            «i» é inteiro pequeno (1.ª
                AND   A                 byte zero), isto é, ABSi <=65535?
```

```
                JR      NZ,2E56,PF-LARGE    Saltar, se não.
                CALL    2D7F,INT-FETCH      i é copiado para DE (i, como x',
                                            >=*).
                LD      B,+10               B passa a contar 16 bits.
                LD      A,D                 D é copiado para A.
                AND     A                   É zero?
                JR      NZ,2E1E,PF-SAVE     Saltar se não é zero.
                OR      E                   Verificar agora E.
                JR      Z,2E24,PF-SMALL     Saltar se DE=zero: x é uma
                                            fracção pura.
                LD      D,E                 Passar E para D e definir
                LD      B,+08               B para 8 bits: D=0 e E não.
2E1E PF-SAVE    PUSH    DE                  Transferir DE para D'E', através
                EXX                         do «stack»-máquina, para passar
                POP     DE                  ao buffer da impressora em
                EXX                         PF-BITS.
                JR      2E7B,PF-BITS        Saltar para diante.
```

3. As fracções puras são multiplicadas por 10↑n, onde n é o número aproximado de zeros após a vírgula decimal; e soma-se −n ao segundo byte de mem-5, que guarda o número de algarismos necessários antes da vírgula decimal; um número negativo aqui indica zeros após a decimal;

```
2E24 PF-SMALL   RST     0028,FP-CALC        i (i=zero aqui)
                DEFB    +E2,obter-mem-2     i,f
                DEFB    +38,fim-calc        i,f
```

Note-se que o «stack» está agora desequilibrado. É necessário um byte extra «DEFB +02, apagar» em 2E25, imediatamente após o RST 0028. Uma expressão como «2»+STR$ 0,5 é avaliada incorrectamente como 0,5; o zero deixado no «stack» desloca o «2», que é tratado como uma cadeia vazia. Do mesmo modo, todas as comparações entre cadeias podem produzir valores incorrectos se a segunda cadeia assumir a forma STR$ x, onde x seja numericamente inferior a 1; por exemplo, a expressão «50» <STR$ 0,1 produz o valor lógico «verdadeiro». Uma vez mais usa-se aqui « » em vez de «50».

```
                LD      A,(HL)              O byte expoente e de f é
                                            copiado para A.
                SUB     +7E                 A passa a e-126 dec, isto
                                            é, e' +2, onde e' é o expoente
                                            verdadeiro de f.
                CALL    2DC1,LOG (2↑A)      A construção A=ABS INT
                                            (LOG (21A)) é realizada
                                            (LOG na base 10); A=n, por
                LD      D,A                 exemplo, n copiado de A para D.
                LD      A,(mem-5-2.º)       A contagem actual é recolhida
                SUB     D                   do 2.º byte de mem-5 e é-lhe
                LD      (mem-5-2.º),A       subtraído n.
                LD      A,D                 n copiado de D para A.
                CALL    2D4F,E-TO-FP        Forma e guarda no «stack»
                                            y = f+10↑n.
                RST     0028,FP-CALC        i,y
                DEFB    +31,copiar          i, y, y
                DEFB    +27,int             i, y, INT (y) = i2
                DEFB    +C1,st-mem-1        (i2 copiado para mem-1).
```



```
                DEFB    +03,subtrair        i, y - i2
                DEFB    +E1,obter-mem-1     i,y - i2, i2
                DEFB    +38,fim-calc        i, f2, i2 (f2=y-i2)
                CALL    2DD5,FP-TO-A        i2 é transferido do «stack»
                                            para A.
                PUSH    HL                  Guarda indicador de f2.
                LD      (mem-3-1st),A       i2 é guardado no 1.º byte de
                                            mem-3: um algarismo para
                                            imprimir.
                DEC     A                   i2 não conta como algarismo para
                RLA                         imprimir se é zero; A é mani-
                SBC     A,A                 pulado de tal modo que zero
                INC     A                   produzirá 0, mas não-zero
                                            produzirá 1.
                LD      HL,+5CAB            O zero ou um é inserido no 1.º
                LD      (HL),A              byte de mem-5 (número de al-
                INC     HL                  garismos para impressão) e
                ADD     A,(HL)              somado ao 2.º byte de mem-5
                LD      (HL),A              (número de algarismos antes
                                            da vírgula decimal).
                POP     HL                  Restaura o indicador de f2.
                JP      2ECF,PF-FRACTN      Salto para guardar f2 no buffer
                                            (HL aponta f2, DE aponta i2).
```

4. Os números maiores do que 2↑27 são do mesmo modo multiplicados por 2↑(−n+7), reduzindo o número de algarismos antes da vírgula decimal a oito, e reentra-se no ciclo em PF-LOOP.

```
2E56 PF-LARGE   SUB     +80                 e-80 hex=e', expoente ver-
                                            dadeiro de i.
                CP      +1C                 e' inferior a 28 decimal?
                JR      C,2E6F,PF-MEDIUM    Saltar, se sim.
                CALL    2DC1,LOG (2↑A)      n é formado em A.
                SUB     +07                 E reduzido a n-7.
                LD      B,A                 Depois copiado para B.
                LD      HL,+5CAC            n-7 é somado ao segundo
                ADD     A,(HL)              byte de mem-5, o número de
                LD      (HL),A              algarismos requerido antes
                                            da vírgula decimal em x.
                LD      A,B                 Depois é multiplicado por
                                            10↑(-n+7).
                NEG                         Isto trá-lo-á à gama inter-
                CALL    2D4F,E-TO-FP        média para impressão.
                JR      2E01,PF-LOOP        Percorre de novo o ciclo
                                            para tratar o número de dimen-
                                            são média.
```

5. A parte inteira de x é agora guardada no buffer da impressora, em mem-3 e mem-4.

```
2E6F PF-MEDIUM  EX      DE,HL               DE aponta agora para i, HL para
                                            f.
                CALL    2FBA,FETCH-TWO      A mantissa de i está
                                            em D', E', D, E.
                EXX                         Obter registos alternativos.
                SET     7,D                 Bit 7 de facto numérico para D'.
                LD      A,L                 Byte expoente e de i para A.
                EXX                         Volta aos registos principais.
```

```
                SUB     +80         Expoente verdadeiro e' =e-80 hex
                                    para A.
                LD      B,A         Isto dá a requerida conta-
                                    gem de bits.
```

Note-se que é retomado aqui o caso em que i é um inteiro pequeno (inferior a 65536).

```
2E7B PF-BITS    SLA     E           A mantissa é rodada para a
                RL      D           esquerda, sendo todos os bits
                EXX                 de i passados a mem-4 e sendo
                RL      E           a vírgula de cada byte ajustada
                                    em cada deslocamento.
                RL      D           Os quatro bytes de i.
                EXX                 Volta aos registos principais.
                LD      HL,+5CAA    Endereço de 5.º byte de mem-4
                LD      C,+05       para HL; contagem de 5 para C.
2E8A PF-BYTES   LD      A,(HL)      Obter o byte de mem-4.
                ADC     A,A         Rodá-lo para a esquerda, acei-
                                    tando o novo bit.
                DAA                 Ajustar a vírgula no byte.
                LD      (HL),A      Passá-lo a mem-4.
                DEC     HL          Apontar para o novo byte de
                                    mem-4.
                DEC     C           Diminuir a contagem de bytes.
                JR      NZ,2E8A,PF-BYTES  Saltar para cada byte de mem-4.
                DJNZ    2E7B,PF-BITS      Saltar para cada bit de INT(x).
```

O ajustamento da vírgula de cada byte de mem-4 produziu 2 algarismos decimais por byte, existindo no máximo 9 algarismos. Estes serão compactados, um em cada byte, em mem-3 e mem-4, usando a instrução RLD.

```
                XOR     A           A é limpo para receber algarismos.
                LD      HL,+5CA6    Endereço-origem: primeiro byte
                                    de mem-4.
                LD      DE,+5CA1    Destino: primeiro byte de
                                    mem-3.
                LD      B,+09       Há no máximo 9 algarismos.
                RLD                 Eliminados 4 bits esquerdos de
                                    mem-4.
                LD      C,+FF       FF em C sinaliza um zero
                                    inicial, 00 sinaliza um
                                    zero fora dessa posição.
2EA1 PF-DIGITS  RLD                 4 bits esquerdos de (HL) para
                                    A, 4 direitos de (HL) para esquerda.
                JR      NZ,2EA9,PF-INSERT  Saltar, se algarismo em A não-0.
                DEC     C           Verificar zero inicial:
                INC     C           põe 0 em flag «zero».
                JR      NZ,2EB3,PF-TEST-2  Saltar se era zero inicial.
2EA9 PF-INSERT  LD      (DE),A      Inserir algarismo agora.
                INC     DE          Apontar destino seguinte.
                INC     (mem-5-1.º) Mais 1 algarismo para imprimir,
                INC     (mem-5-2.º) e outro antes da vírgula.
                LD      C,+00       Passar a flag de zero inicial
                                    para outro.
2EB3 PF-TEST-2  BIT     0,B         O indicador de origem deve ser
                JR      Z,2EB8,PF,ALL-9   incrementado em cada 2.ª
                INC     HL          passagem pelo ciclo, quando B
                                    é ímpar.
```



```
2EB8 PF-ALL-9   DJNZ    2EA1,PF-DIGITS  Saltar atrás para os 9 algarismos.
                LD      A,(mem-5-1.º)   Obter contagem: havia 9 alga-
                SUB     +09             rismos excluindo zeros iniciais?
                JR      C,2ECB,PF-MORE  Se não, saltar para obter mais.
                DEC     (mem-5-1.º)     Preparar para arredondar; redu-
                                        zir contagem a oito.
                LD      A,+04           Comparar 9.º algarismo, byte 4
                CP      (mem-4-4.º)     de mem-4, com 4 para passar «carry»
                                        a um (para arredondamento).
                JR      2F0C,PF-ROUND   Saltar para arredondar.
2ECB PF-MORE    RST     0028,FP-CALC    Usar de novo calculador.
                DEFB    +02,apagar      - (i é agora eliminado).
                DEFB    +E2,obter mem-2 i
                DEFB    +38,fim-calc    i
```

6. Guarda-se agora no buffer da impressora a parte fraccionária de x.

```
2ECF PF-FRACTN  EX      DE,HL           DE aponta agora para f.
                CALL    2FBA,FETCH-TWO  A mantissa de f está agora
                                        em D',E',D,E.
                EXX                     Obter registos alternativos.
                LD      A,+80           Expoente de f reduzido a zero,
                SUB     L               deslocando os bits de f 80
                LD      L,+00           hex — e posições à direita,
                                        onde L' continha e.
                SET     7,D             Bit numérico verdadeiro
                                        para bit 7 de D'.
                EXX                     Restaurar registos principais.
                CALL    2FDD,SHIFT-FP   Fazer deslocamento.
2EDF PF-FRN-LP  LP      A,(mem-5-1.º)   Obter contagem de algarismos.
                CP      +08             Já existem 8 algarismos?
                JR      C,2EEC,PR-FR-DGT  Se não, saltar para diante.
                EXX                     Se 8 algarismos, usar apenas
                RL      D               f para arredondar i, rodando
                                        D' para esquerda (define «carry»).
                EXX                     Obter registos principais e
                JR      2F0C,PF-ROUND   saltar para arredondar.
2EEC PF-FR-DGT  LD      BC,+0200        Zero inicial para C, contagem
                                        de 2 em B.
2EEF PF-FR-EXX  LD      A,E             D'E'DE multiplica por 10 em 2
                CALL    2F8B,CA=10*A+C  fases, primeiro DE depois D'E',
                LD      E,A             cada byte a byte em 2 fases,
                LD      A,D             e a parte inteira do resultado
                CALL    2F8B,CA=10*A+C  é obtida em C para ser passada
                LD      D,A             ao buffer da impressora.
                PUSH    BC              Contagem e resultado alternam
                EXX                     entre BC e B'C'.
                POP     BC
                DJNZ    2EEF,PF-FR-EXX  Ver atrás uma vez através dos
                                        registos alternativos.
                LD      HL,+5CA1        Início — 1.º byte de mem-3.
                LD      A,C             Resultado em A para guardar.
                LD      C,(mem-5-1.º)   Contagem dos algarismos até
                                        agora em C.
                ADD     HL,BC           Endereçar o 1.º byte vazio.
                LD      (HL),A          Guardar o algarismo seguinte.
                INC     (mem-5-1.º)     Aumentar contador.
                JR      2EDF,PF-FRN-LP  Percorrer ciclo até haver oito
                                        algarismos.
```

7. Os algarismos guardados no buffer da impressora são arredondados até um máximo de oito algarismos para impressão.

```
2F0C PF-ROUND     PUSH   AF              Guardar a flag «carry» para
                                         arredondamento.
                  LD     HL,+5CA1        Endereço-base do número: mem-3,
                                         byte 1.
                  LD     C,(mem-5-1.º)   Deslocamento (número de alga-
                  LD     B,+00           rismos) para BC.
                  ADD    HL,BC           Endereçar o último byte do
                                         número.
                  LD     B,C             Copiar C para B como contador.
                  POP    AF              Restaurar a flag «carry».
2F18 PF-RND-LP    DEC    HL              Este é o último
                                         byte do número.
                  LD     A,(HL)          Colocar o byte em A.
                  ADC    A,+00           Somar a «carry», ou
                                         seja, arredondar para mais.
                  LD     (HL),A          Guardar o byte arrendondado
                                         no buffer.
                  AND    A               Se o byte é 0 ou 10, B será
                  JR     Z,2F25,PF-R-BACK decrementado e o zero final
                  CP     +0A             (ou o 10) não será contado
                                         para impressão.
                  CCF                    «Carry» a zero se algarismo válido.
                  JR     NC,2F2D,PF-COUNT Saltar, se «carry» é zero.
2F25 PF-R-BACK    DJNZ   2F18,PF-RND-LP  Saltar atrás para maior arre-
                                         dondamento ou mais zeros finais.
                  LD     (HL),+01        Excesso para a esquerda; é ne-
                  INC    B               cessário aqui mais um «1».
                  INC    (mem-5-2.º)     Também um algarismo extra
                                         antes da vírgula decimal.
2F2D PF-COUNT     LD     (mem-5-1.º),B   B define a quantidade de alga-
                                         rismos a imprimir (os zeros
                                         finais não serão impressos).
                  RST    0028,FP-CALC    f deve ser eliminado.
                  DEFB   +02,apagar      .
                  DEFB   +38,fim-calc    .
                  EXX                    O deslocamento guardado em
                  POP    HL              «stack» passa para H'L'.
                  EXX
```

8. O número pode agora ser impresso. Primeiro C passa a guardar o número de algarismos a imprimir, não contando os zeros finais, enquanto B guarda o número de algarismos requeridos antes da vírgula decimal.

```
                  LD     BC,(mem-5-1.º)  Os contadores são definidos.
                  LD     HL,+5CA1        Início dos algarismos.
                  LD     A,B             Se mais de 9, ou menos de
                  CP     +09             – 4, são precisos algarismos
                  JR     C,2F46,PF-NOT-E antes da vírgula decimal, ou
                                         seja, o formato E.
                  CP     +FC             Menos de 4 significa mais de
                  JR     C,2F6C,PF-E-FRMT 4 zeros iniciais após a decimal.
                  AND    A               Não há algarismos antes da
                  CALL   Z,15EF,OUT-CODE vírgula? Então, imprimir um
                                         zero inicial.
```


O ponto de entrada seguinte é igualmente usado para imprimir os algarismos necessários para impressão no formato E.

```
2F4A PF-E-SBRN    XOR    A               Começa por passar A a zero.
                  SUB    B               Subtrair B: «menos» indica que
                  JR     M,2F52,PF-OUT-LP existem algarismos antes da
                                         vírgula; saltar para diante
                                         para os imprimir.
                  LD     B,A             A é agora preciso como contador.
                  JR     2F5E,PF-DC-OUT  Saltar para diante para imprimir
                                         a parte decimal.
2F52 PF-OUT-LP    LD     A,C             Copiar o número de algarismos a
                  AND    A               imprimir para A. Se A é zero, há
                  JR     Z,2F59,PF-OUT-DT ainda zeros finais para imprimir
                                         (B não é zero), e portanto salto.
                  LD     A,(HL)          Obter algarismo no buffer
                                         da impressora.
                  INC    HL              Apontar para o algarismo seguinte.
                  DEC    C               Diminuir a contagem de um.
2F59 PF-OUT-DT    CALL   15EF,OUT-CODE   Apontar para o algarismo apropriado.
                  DJNZ   2F52,PF-OUT-LP  Percorrer ciclo até B ser zero.
2F5E PF-DC-OUT    LD     A,C             É o momento de imprimir a vírgula,
                  AND    A               se C não for zero; nesse caso,
                  RET    Z               retorno — terminado.
                  INC    B               Somar 1 a B — incluir a vírgula
                                         decimal (em inglês, o «ponto»)
                  LD     A,+2E           Colocar o código de «.» em A.
2F64 PF-DEC-0S    RST    0010,PRINT-A-1  Imprimir o «.».
                  LD     A,+30           Inserir o código do caracter
                                         «0».
                  DJNZ   2F64,PF-DEC-0S  Voltar atrás para imprimir todos
                                         os zeros necessários.
                  LD     B,C             Definir o contador para os
                                         algarismos restantes.
                  JR     2F52,PF-OUT-LP  Salto atrás para os imprimir.
2F6C PF-E-FRMT    LD     D,B             A contagem de algarismos é
                                         copiada para D.
                  DEC    D               É decrementada para dar o
                                         expoente.
                  LD     B,+01           É requerido um algarismo antes
                                         da vírgula no formato E.
                  CALL   2F4A,PF-E-SBRN  Toda a parte do número antes de
                                         «E» é agora impressa.
                  LD     A,+45           Indicar o código de caracter
                                         para «E».
                  RST    0010,PRINT-A-1  Imprimir o «E».
                  LD     C,D             Expoente para C para impressão.
                  LD     A,C             E para A para teste.
                  AND    A               O seu sinal é testado.
                  JP     P,2F83,PF-E-POS Saltar se é positivo.
                  NEG                    Senão, negá-lo em A.
                  LD     C,A             Depois copiá-lo de novo para
                                         C para impressão.
                  LD     A,+2D           Inserir o código de «-».
                  JR     2F85,PF-E-SIGN  Saltar para imprimir sinal.
2F83 PF-E-POS     LD     A,+2B           Inserir o código do caracter
                                         «+».
2F85 PF-E-SIGN    RST    0010,PRINT-A-1  Imprimir o sinal «+» ou «-».
                  LD     B,+00           BC contém o expoente para
```

```
                JP      1A1B,OUT-NUM    impressão.
                                        Saltar atrás para imprimi-lo e
                                        terminar.
```

A subrotina «CA=10*A+C»

Esta subrotina é invocada pela subrotina PRINT-FP a fim de multiplicar cada byte de D'E'DE por 10 e devolver a parte inteira do resultado no registo C. Na entrada, o registo A contém o byte a multiplicar por 10, e o registo C contém o transporte do byte anterior. Ao terminar, o registo A contém o byte resultante e o registo C o transporte («carry») para o byte seguinte.

```
2F88 CA=10*A+C  PUSH    DE      Guardar o par DE em uso.
                LD      L,A     Copiar o multiplicando de A
                LD      H,+00   para HL.
                LD      E,L     Copiá-lo também para DE.
                LD      D,H
                ADD     HL,HL   Duplicar HL.
                ADD     HL,HL   Duplicá-lo de novo.
                ADD     HL,DE   Somar em DE para obter HL=5*A.
                ADD     HL,HL   Duplicar de novo: agora HL=10*A.
                LD      E,C     Copiar C para DE (D é zero) para
                                soma.
                ADD     HL,DE   Agora HL=10*A+C.
                LD      C,H     H é copiado para C.
                LD      A,L     L é copiado para A, terminando
                                a tarefa.
                POP     DE      O par de registos DE é restaurado.
                RET             Final.
```

A subrotina «Preparar para somar»

Esta subrotina é a primeira de quatro subrotinas usadas pelas rotinas aritméticas principais — SUBTRACÇÃO, ADIÇÃO, MULTIPLICAÇÃO e DIVISÃO. Esta subrotina, em particular, prepara um número em vírgula flutuante para a soma, principalmente substituindo o bit de sinal por um verdadeiro bit numérico 1, e negando o número (complemento para dois) se for negativo. O expoente é devolvido no registo A, e o primeiro byte passa a 00 hex no caso de um número positivo e a FF no de um número negativo.

```
2F9B PREP-ADD   LD      A,(HL)  Transferir o expoente para A.
                LD      (HL),+00 Pressupor um número positivo.
                AND     A       Se o número é zero, a preparação
                RET     Z       está terminada.
                INC     HL      Apontar para o byte de sinal.
                BIT     7,(HL)  Definir a flag «zero» para um
                                número positivo.
                SET     7,(HL)  Restaurar o bit de facto numérico.
                DEC     HL      Apontar de novo para o 1.º byte.
                RET     Z       Foram preparados os n.os positivos,
                                mas os negativos devem ainda
                                ser complementados para dois.
                PUSH    BC      Guardar um expoente anterior.
                LD      BC,+0005 Deve-se tratar 5 bytes.
                ADD     HL,BC   Apontar para um após o último byte.
                LD      B,C     Transferir o «5» para B.
                LD      C,A     Guardar o expoente em C.
```



```
                SCF             Definir flag «carry» para negação.
2FAF NEG-BYTE   DEC     HL      Apontar para cada byte.
                LD      A,(HL)  Obter cada byte.
                CPL             Complementá-lo para um.
                ADC     A,+00   Somar «carry» para negação.
                LD      (HL),A  Restaurar o byte.
                DJNZ    2FAF,NEG-BYTE Restaurar ciclo «5» vezes.
                LD      A,C     Restaurar o expoente para A.
                POP     BC      Restaurar qualquer expoente anterior.
                RET             Final.
```

A subrotina «Obter dois números»

Esta subrotina é invocada por ADDITION, MULTIPLICATION e DIVISION para obter dois números no «stack» do calculador e colocá-los no registo, incluindo nos registos alternativos.

No início da subrotina o par de registos HL aponta para o primeiro byte do primeiro número e o par de registos DE aponta para o primeiro byte do segundo número.

Quando a subrotina é invocada por MULTIPLICATION ou DIVISION o sinal do resultado é guardado no segundo byte do primeiro número.

```
2FBA FETCH-TWO  PUSH    HL      É preservado HL.
                PUSH    AF      É preservado AF.

    Invocar os cinco bytes do primeiro número  — M1, M2, M3, M4 e M5.
    E do segundo número                        — N1, N2, N3, N4 e N5.

                LD      C,(HL)  M1 para C.
                INC     HL      O seguinte.
                LD      B,(HL)  M2 para B.
                LD      (HL),A  Copiar o sinal do resultado
                                para (HL).
                INC     HL      O seguinte.
                LD      A,C     M1 para A.
                LD      C,(HL)  M3 para C.
                PUSH    BC      Guardar M2 e M3 so «stack»-
                                -máquina.
                INC     HL      O seguinte.
                LD      C,(HL)  M4 para C.
                INC     HL      O seguinte.
                LD      B,(HL)  M5 para B.
                EX      DE,HL   HL aponta agora para N1.
                LD      D,A     M1 para D.
                LD      E,(HL)  N1 para E.
                PUSH    DE      Guardar M1 e N1 no «stack»-
                                -máquina.
                INC     HL      O seguinte.
                LD      D,(HL)  N2 para D.
                INC     HL      O seguinte.
                LD      E,(HL)  N3 para E.
                PUSH    DE      Guardar N2 e N3 no «stack»-
                                -máquina.
                EXX             Obter os registos alternativos.
                POP     DE      N2 para D' e N3 para E'.
                POP     HL      M1 para H' e N1 para L'.
                POP     BC      M2 para B' e M3 para C'.
                EXX             Obter os registos originais.
```

```
        INC     HL          O seguinte.
        LD      D,(HL)      N4 para D.
        INC     HL          O seguinte.
        LD      E,(HL)      N5 para E.
        POP     AF          Restaurar AF original.
        POP     HL          Restaurar HL original.
        RET                 Final.
```

Resumo: M1 — M5 estão em H', B', C', C, B.
N1 — N5 estão em L', D', E', D, E.
HL aponta para o primeiro byte do primeiro número.

A subrotina «Deslocar o 2.º termo»

Esta subrotina desloca um número em vírgula flutuante até 32 decimal (20 hexa) posições para a direita a fim de o alinhar convenientemente para a soma. O número com o menor expoente foi colocado na posição de 2.º termo antes de a subrotina ser chamada. Qualquer valor que saia para a direita, para «carry», é de novo somado ao número. Se a diferença de expoente é superior a 32 decimal, ou a «carry» retorna ao início do número, este é passado a zero de modo a que a soma não altere o outro número (o 1.º termo).

A subrotina «SHIFT ADDEND»

```
2FDD SHIFT-FP       AND     A           Se a diferença de expoente
                    RET     Z           é zero, a subrotina volta imedia-
                    CP      +21         tamente. Se é maior do que
                    JR      NC,2FF9,ADDEND-0  20 hex, saltar para diante.
                    PUSH    BC          Guardar BC brevemente.
                    LD      B,A         Transferir a diferença de expoente
                                        para B para contar as rotações.
2FE5 ONE-SHIFT      EXX                 Rotação aritmética para a direita
                    SRA     L           de L', mantendo os bits de sinal.
                    RR      D           Rodar para a direita com «carry»
                    RR      E           D', E', D e E.
                    EXX
                    RR      D           Rodando assim os 5 bytes do
                    RR      E           número para a direita tantas
                    DJNZ    2FE5,ONE-SHIFT  vezes quantas a contagem de B.
                    POP     BC          Voltar atrás até B ser zero.
                    RET     NC          Restaurar o BC original.
                    CALL    3004,ADD-BACK  Fim, se não há «carry» para recuperar.
                    RET     NZ          Recuperar «carry».
                                        Retorno a menos que «carry» ocorra
                                        de novo (neste caso não há
                                        nada a somar).
2FF9 ADDEND-0       EXX                 Obter L', D' e E'.
                    XOR     A           Limpar o registo A.
2FFB ZEROS-4/5      LD      L,+00       Passar o número para zero em D',
                    LD      D,A         E', D e E, junto com o byte
                    LD      E,L         indicador de sinal L', que é
                    EXX                 00 hex para um n.º positivo,
                    LD      DE,+0000    FF hex para um número ne-
                                        gativo. ZEROS-4/5 produz
                                        apenas 4 bytes zero
                                        quando invocada na posição
                                        3160.
                    RET                 Final.
```


A subrotina «ADD-BACK»

Esta subrotina soma ao número qualquer «carry» que tenha sobrado para a direita. No caso extremo, o «carry» volta ao lado esquerdo do número.

Quando esta subrotina é invocada durante a adição, esta rotação completa indica que foi deslocada uma mantissa de 0,5 32 espaços para a direita, e que o número a que se soma será agora passado a zero; quando invocada por MULTIPLICAÇÃO, significa que o expoente deve ser incrementado, e isto pode resultar num «overflow».

```
3004 ADD-BACK       INC     E           Somar «carry» ao byte da direita.
                    RET     NZ          Retorno se não há excesso
                                        à esquerda.
                    INC     D           Continuar para o byte seguinte.
                    RET     NZ          Retorno se não há
                                        excesso à esquerda.
                    EXX                 Obter o byte seguinte.
                    INC     E           Incrementá-lo também.
                    JR      NZ,300D,ALL-ADDED  Saltar se não há excesso.
                    INC     D           Incrementar o último byte.
300D ALL-ADDED      EXX                 Restaurar os registos
                                        originais.
                    RET                 Final.
```

A operação «Subtracção»
(Deslocamento 03 — ver CALCULATE, adiante: «subtrair»)

Esta rotina limita-se a alterar o sinal do número de que se subtrai e passa a ADDITION.

Note-se que HL aponta para o diminuendo e DE para o subtraendo (ver mais pormenores em ADDITION).

```
300F SUBTRACT       EX      DE,HL       Trocar os registos.
                    CALL    346E,NEGATE Mudar o sinal do
                                        subtraendo.
                    EX      DE,HL       Trocar os indicadores e
                                        passar a ADDITION.
```

A operação «Adição»
(Deslocamento 0F — ver CALCULATE, adiante: «somar»)

Primeira das três subrotinas aritméticas principais, esta realiza a soma em vírgula flutuante de dois números, tendo cada um deles uma mantissa de 4 bytes e um expoente de um byte. Nestas três subrotinas, os dois números no topo do «stack» do calculador são somados/multiplicados/divididos de modo a fornecerem um número no topo do mesmo «stack», um «último valor».

HL aponta para o segundo número a partir do topo, o que soma/multiplica ou é dividido; DE aponta para o número no topo do «stack» do calculador, o adendo/multiplicando/divisor. Depois HL aponta para o «último valor» cujo endereço pode também ser considerado como STKEND-5.

Mas a rotina de adição verifica primeiro se os dois números a somar são «inteiros pequenos». Se são, soma-os muito simplesmente em HL e BC, e coloca o resultado directamente no «stack». Não é necessária uma complementação para dois antes ou depois da adição, dado que estes números são guardados na forma de complemento para dois, prontos para serem somados.

```
3014 adição   LD     A,(DE)          Verificar se os 1.os bytes dos
              OR     (HL)            dois números são zero.
              JR     NZ,303E,FULL-ADDN Se não, saltar para soma normal.
              PUSH   DE              Guardar o indicador do segundo
                                     número.
              INC    HL              Apontar para o segundo byte do
              PUSH   HL              primeiro número e guardar tam-
                                     bém esse indicador.
              INC    HL              Apontar para o byte menos
                                     significativo.
              LD     E,(HL)          Guardá-lo em E.
              INC    HL              Apontar para o byte mais
                                     significativo.
              LD     D,(HL)          Guardá-lo em D.
              INC    HL              Passar ao segundo byte do
              INC    HL              segundo número.
              INC    HL
              LD     A,(HL)          Obtê-lo em A (é o byte de
                                     sinal).
              INC    HL              Apontar para o byte menos
                                     significativo.
              LD     C,(HL)          Guardá-lo em C.
              INC    HL              Apontar para o byte mais
                                     significativo.
              LD     B,(HL)          Guardá-lo em B.
              POP    HL              Obter o indicador do byte de
              EX     DE,HL           sinal do primeiro número; pô-lo
                                     em DE, e o número em HL.
              ADD    HL,BC           Realizar a soma: resultado em
                                     HL.
              EX     DE,HL           Resultado para DE, byte-sinal em HL.
              ADC    A,(HL)          Somar os bytes de sinal e a «carry»
              RRCA                   em A; detecta assim qualquer
                                     «overflow».
              ADC    A,+00           Um A não-zero indica
                                     «overflow».
              JR     NZ,303C,ADDN-OFLW Saltar para anular indicadores
                                     e executar a soma completa.
              SBC    A,A             Definir o byte de sinal correcto
                                     do resultado.
3032          LD     (HL),A          Guardá-lo no «stack».
              INC    HL              Apontar para a posição seguinte.
              LD     (HL),E          Guardar o byte baixo do resultado.
              INC    HL              Apontar para a posição seguinte.
              LD     (HL),D          Guardar agora o byte alto do
                                     resultado.
              DEC    HL              Passar o indicador para o
              DEC    HL              endereço do primeiro byte do
              DEC    HL              resultado.
              POP    DE              Restaurar STKEND em DE.
              RET                    Final.
```

Notar que pode surgir aqui o número -65536 decimal sob a forma 00 FF 00 00 00 como resultado da soma de dois inteiros negativos mais pequenos, por exemplo, -65000 e -536. É simplesmente guardado nesta forma. Trata-se de um erro. O sistema do Spectrum não pode tratar este número.

A maior parte das funções tratam-no como zero, e é impresso sob a forma -1E-38, obtida tratando-o como «menos zero» num formato ilegítimo.

Um remédio possível consiste em comparar este número por volta do byte 3032, e se estiver presente passar o segundo byte a 80 hex e o primeiro a 91 hex, produzindo assim a forma completa em vírgula flutuante correspondente a este número, ou seja, 91 80 00 00 00, o que não provoca problemas. Ver igualmente as observações em «truncar», mais abaixo, antes do byte 3225, e o Apêndice.

```
303C ADDN-OFLW  DEC    HL              Restaurar o indicador do
                                       primeiro número.
                POP    DE              Restaurar o indicador do
                                       segundo número.
303E FULL-ADDN  CALL   3293,RE-ST-TWO  Guardar de novo ambos os n.os
                                       em vírgula flutuante.
```

A subrotina «adição» chama primeiramente PREP-ADD para cada número, obtém em seguida os dois números do «stack» do calculador, e coloca o que possui o menor expoente da posição do «adendo». Invoca em seguida SHIFT-FP para deslocar aquele número 32 casas decimais para a direita de modo a alinhá-lo para execução da soma. Esta é realizada em poucos bytes, sendo realizado um único deslocamento para a «carry» (excesso — «overflow» — à esquerda) se necessário, o resultado é complementado para dois se for negativo, e é indicado qualquer «overflow» aritmético; senão, a subrotina salta para TEST-NORM a fim de normalizar o resultado e reenviá-lo para o «stack» com o bit correcto de sinal inserido no segundo byte.

```
                EXX                    Trocar os registos.
                PUSH   HL              Guardar o endereço literal seguinte.
                EXX                    Trocar os registos.
                PUSH   DE              Indicador do número somado.
                PUSH   HL              Indicador do número que soma.
                CALL   2F9B,PREP-ADD   Preparar este.
                LD     B,A             Guardar o seu expoente em B.
                EX     DE,HL           Trocar os indicadores.
                CALL   2F9B,PREP-ADD   Preparar o número somado.
                LD     C,A             Guardar o seu expoente em C.
                CP     B               Se o 1.º expoente é menor, manter
                JR     NC,3055,SHIFT-LEN o 1.º número como «somado»; se-
                LD     A,B             não, mudar novamente os
                LD     B,C             expoentes juntamente com os
                EX     DE,HL           indicadores.
3055 SHIFT-LEN  PUSH   AF              Guardar o expoente maior em A.
                SUB    B               A diferença entre os expoentes
                                       é o comprimento do desloca-
                                       mento para a direita.
                CALL   2FBA,FETCH-TWO  Obter os dois números no
                                       «stack».
                CALL   2FDD,SHIFT-FP   Deslocar o primeiro para a direita.
                POP    AF              Restaurar o expoente maior.
                POP    HL              HL aponta para o resultado.
                LD     (HL),A          Guardar o expoente do
                                       resultado.
                PUSH   HL              Guardar de novo o indicador.
                LD     L,B             M4 para H e M5 para L
                LD     H,C             (ver FETCH-TWO).
                ADD    HL,DE           Somar os 2 bytes da direita.
                EXX                    N2 para H' e N3 para L
                EX     DE,HL           (ver FETCH-TWO).
```

```
                ADC     HL,BC       Somar bytes da esquerda com carry.
                EX      DE,HL       Resultado de novo em D'E'.
                LD      A,H         Somar H', L' e «carry»; os meca-
                ADC     A,L         nismos resultantes garantirão
                LD      L,A         que seja efectuado 1 deslocamento
                RRA                 para a direita se a soma de 2 n.os
                XOR     L           positivos tiver dado «overflow», ou
                EXX                 a de 2 números negativos não o
                                    tiver produzido.
                EX      DE,HL       Resultado agora em DED'E'.
                POP     HL          Obter o indicador do
                                    expoente.
                RRA                 Teste de deslocamento (H',L'
                JR      NC,307C,TEST-NEG  eram 00 hex para n.os positivos
                                    e FF hex no caso de números
                                    negativos).
                LD      A,+01       A conta um deslocamento.
                CALL    2FDD,SHIFT-FP  Realiza deslocamento.
                INC     (HL)        Somar 1 ao expoente; isto
                JR      Z,309F,ADD-REP-6  pode conduzir a overflow.
307C TEST-NEG   EXX                 Verificar se resultado negativo:
                LD      A,L         obter bit de sinal de L' em A
                AND     +80         (isto indica agora correctamente
                EXX                 o sinal do resultado).
                INC     HL          Guardá-lo na posição do segundo
                LD      (HL),A      byte do resultado no «stack»
                DEC     HL          do calculador.
                JR      Z,30A5,GO-NC-MLT  Se é zero, não complementar
                                    para dois o resultado.
                LD      A,E         Obter o primeiro byte.
                NEG                 Negá-lo.
                CCF                 Complementar o «carry» para
                LD      E,A         nova negação, e guardar
                                    o byte.
                LD      A,D         Obter o byte seguinte.
                CPL                 Complementá-lo para um.
                ADC     A,+00       Somar o carry para negação.
                LD      D,A         Guardar o byte.
                EXX                 Obter o byte seguinte no
                LD      A,E         registo A.
                CPL                 Complementá-lo para um.
                ADC     A,+00       Somar o carry para negação.
                LD      E,A         Guardar o byte.
                LD      A,D         Obter último byte.
                CPL                 Complementá-lo para um.
                ADC     A,+00       Somar o carry para negação.
                JR      NC,30A3,END-COMPL  Final, se não há «carry».
                RRA                 Senão, obter .5 na mantissa e
                EXX                 somar 1 ao expoente; isto será
                INC     (HL)        necessário quando se somam dois
                                    números negativos para obter uma
                                    potência exacta de 2, e pode
                                    conduzir a overflow.
309F ADD-REP-6  JP      Z,31AD,REPORT-6  Dar erro se necessário.
                EXX
30A3 END-COMPL  LD      D,A         Guardar o último byte.
                EXX
30A5 GO-NC-MLT  XOR     A           Limpar a flag «carry».
                JP      3155,TEST-NORM  Sair por TEST-NORM.
```


A subrotina «HL=HL*DE»

Esta subrotina é invocada por «GET-HL*DE» e por «MULTIPLICATION» a fim de realizar a multiplicação de 16 bits.

Qualquer excesso («overflow») relativamente aos 16 bits é tratado após o retorno da subrotina.

```
30A9 HL=HL*DE   PUSH    BC          BC é guardado.
                LD      B,+10       Será uma multiplicação a
                                    16 bits.
                LD      A,H         A contém o byte alto.
                LD      C,L         C contém o byte baixo.
                LD      HL,+0000    Inicializar o resultado para zero.
30B1 HL-LOOP    ADD     HL,HL       Duplicar o resultado.
                JR      C,30BE,HL-END  Saltar se há excesso.
                RL      C           Rodar o bit 7 de C para «carry».
                RLA                 Rodar o bit «carry» para o bit 0
                                    e o bit 7 para a flag «carry».
                JR      NC,30BC,HL-AGAIN  Saltar se a flag «carry» é zero.
                ADD     HL,DE       Senão, somar DE uma vez.
                JR      C,30BE,HL-END  Saltar sb ocorre overflow.
30BC HL-AGAIN   DJNZ    30B1,HL-LOOP  Foram realizadas até 16 passagens.
30BE HL-END     POP     BC          Restaurar BC.
                RET                 Final.
```

A subrotina «Preparar para multiplicar ou dividir»

Esta subrotina prepara um número em vírgula flutuante para a multiplicação ou divisão, terminando com «carry» a um se o número é zero, colocando o sinal do resultado no registo A, e substituindo o bit de sinal no número pelo bit numérico verdadeiro, 1.

```
30C0 PREP-M/D   CALL    34E9,TEST-ZERO  Se o número é zero, retorno
                RET     C           com flag «carry» a um.
                INC     HL          Apontar para o byte de sinal.
                XOR     (HL)        Obter sinal do resultado em A
                                    (sinais iguais dão +, desiguais
                                    dão menos); passa a 0 a flag «carry».
                SET     7,(HL)      Define o bit numérico verdadeiro.
                DEC     HL          Aponta de novo para expoente.
                RET                 Retorno com flag «carry» a zero.
```

A operação «Multiplicação»

(Deslocamento 04 — ver CALCULATE, adiante: «multiplicar»)

Esta subrotina verifica primeiro se os dois números a multiplicar são «inteiros pequenos». Se são, usa INT-FETCH para os obter do «stack», HL=HL*DE para os multiplicar e INT-STORE para devolver o resultado ao «stack». Qualquer excesso («overflow») desta «multiplicação curta» (ou seja, se o resultado não é, ele mesmo, um «inteiro pequeno») provoca um salto para a multiplicação na forma de vírgula flutuante (ver adiante).

```
30CA multiplicar  LD    A,(DE)          Verificar se os 1.os bytes
                  OR    (HL)            dos 2 números são zero.
                  JR    NZ,30F0,MULT-LONG Se não, saltar para multi-
                                        plicação «comprida».
                  PUSH  DE              Guardar indicadores: para
                                        o segundo número.
                  PUSH  HL              E para o primeiro número.
                  PUSH  DE              E de novo para o segundo
                                        número.
                  CALL  2D7F,INT-FETCH  Obter sinal em C, e n.º em DE.
                  EX    DE,HL           Número para HL agora.
                  EX    (SP),HL         Número para «stack», segundo
                                        indicador para HL.
                  LD    B,C             Guardar 1.º sinal em B.
                  CALL  2D7F,INT-FETCH  Obter segundo sinal em C, e
                                        número em DE.
                  LD    A,B             Formar sinal do resultado em
                  XOR   C               A: sinais iguais dão mais (00),
                                        e desiguais dão menos (FF).
                  LD    C,A             Guardar sinal do resultado em C.
                  POP   HL              Restaurar o 1.º número para HL.
                  CALL  30A9,HL=HL*DE   Realizar agora a multipli-
                                        cação.
                  EX    DE,HL           Guardar o resultado em DE.
                  POP   HL              Restaurar o indicador do pri-
                                        meiro número.
                  JR    C,30EF,MULT-OFLW Saltar se «overflow» para a
                                        multiplicação «completa».
30E5              LD    A,D             Estes 5 bytes garantem que
                  OR    E               00 FF 00 00 00 é substituído
                  JR    NZ,30EA,MULT-RSLT por zero; que não são ne-
                  LD    C,A             cessários se este número losse
                                        excluído do sistema (ver acima, após
                                        303B).
30EA MULT-RSLT    CALL  2D8E,INT-STORE  Guardar agora o resultado no
                                        «stack».
                  POP   DE              Restaurar STKEND em DE.
                  RET                   Terminar.
30EF MULT-OFLW    POP   DE              Restaurar o indicador do
                                        segundo número.
30F0 MULT-LONG    CALL  3293,RE-ST-TWO  Guardar de novo os 2 números
                                        em forma de vírgula flutuante.
```

A subrotina de multiplicação «completa» prepara o primeiro número para a multiplicação invocando PREP-M/D, e fazendo retorno se é zero; senão, prepara o segundo número chamando novamente PREP-M/D, e se é zero a subrotina passa a definir o resultado zero. Em seguida, recupera os dois números do «stack» do calculador, e multiplica as respectivas mantissas do modo habitual, rodando o primeiro número (tratado como multiplicador) para a direita e somando o segundo número (o multiplicando) ao resultado sempre que o bit multiplicador é 1. Os expoentes são em seguida somados, verificando-se se ocorre «overflow» ou «underflow» (dando resultado zero). Finalmente, normaliza-se o resultado que é devolvido ao «stack» do calculador com o bit de sinal correcto no segundo byte.

```
                  XOR   A               A passa a 00 hex., de tal modo
                                        que o sinal do 1.º número irá
                                        para A.
```



```
                  CALL  30C0,PREP-M/D   Preparar o primeiro número, e
                  RET   C               retorno se for zero (o resul-
                                        tado já é zero).
                  EXX                   Trocar os registos.
                  PUSH  HL              Guardar o endereço literal seguinte.
                  EXX                   Trocar os registos.
                  PUSH  DE              Guardar o indicador do mul-
                                        tiplicando.
                  EX    DE,HL           Trocar os indicadores.
                  CALL  30C0,PREP-M/D   Preparar o 2.º número.
                  EX    DE,HL           Trocar de novo os indicadores.
                  JR    C,315D,ZERO-RSLT Saltar para diante se o 2.º
                                        número é zero.
                  PUSH  HL              Guardar o indicador do resultado.
                  CALL  2FBA,FETCH-TWO  Obter os 2 números do
                                        «stack».
                  LD    A,B             M5 para A (ver FETCH-TWO).
                  AND   A               Preparar para subtracção.
                  SBC   HL,HL           Inicializar HL a zero para
                                        resultado.
                  EXX                   Trocar os registos.
                  PUSH  HL              Guardar M1 e N1 (ver
                                        FETCH-TWO).
                  SBC   HL,HL           Inicializar também H'L' para
                                        o resultado.
                  EXX                   Trocar os registos.
                  LD    B,+21           B conta 33 decimal, 21
                                        hex., deslocamentos.
                  JR    3125,STRT-MLT   Saltar para o ciclo adiante.
```

Agora entrar no ciclo multiplicador.

```
3114 MLT-LOOP     JR    NC,311B,NO-ADD  Saltar para diante para NO-ADD
                                        se não «carry», ou seja, se o bit
                                        multiplicador é zero.
                  ADD   HL,DE           Senão, somar o multiplicando
                  EXX                   em D'E'DE (ver FETCH-TWO) ao
                  ADC   HL,DE           resultado construído em
                  EXX                   H'L'HL.
311B NO-ADD       EXX                   Quer o multiplicando seja
                  RR    H               somado ou não, deslocar para a
                  RR    L               direita H'L'HL, rodando cada
                  EXX                   byte com «carry», de tal modo
                  RR    H               que qualquer bit que passe a
                  RR    L               «carry» seja recebido pelo byte
                                        seguinte, e o deslocamento
                                        continue para B'C'CA.
3125 STRT-MLT     EXX                   Deslocar multiplicador para
                  RR    B               a direita em B'C'CA (ver
                                        FETCH-TWO e acima).
                  RR    C               Uma última passagem de bit para
                  EXX                   «carry» provocará outra soma
                  RR    C               do multiplicando ao resultado.
                  RRA
                  DJNZ  3114,MLT-LOOP   Percorrer o ciclo 33 vezes.
                  EX    DE,HL           Mover o resultado de:
                  EXX
                  EX    DE,HL           H'L'HL para D'E'DE.
                  EXX
```

Somar agora os expoentes.

```
                POP   BC                  Restaurar os expoentes – M1
                                          e N1.
                POP   HL                  Restaurar o indicador do
                                          byte expoente.
                LD    A,B                 Obter a soma dos dois
                ADD   A,C                 bytes expoentes em A, e o
                                          «carry» correcto.
                JR    NZ,3138,MAKE-EXPT   Se a soma é igual a zero
                AND   A                   limpar «carry»; senão,
                                          deixá-la como está.
3138 MAKE-EXPT  DEC   A                   Preparar para aumentar o
                CCF                       expoente de 80 hex.
```

O resto da subrotina é comum a MULTIPLICAÇÃO e DIVISÃO.

```
313D DIVN-EXPT  RLA                       Estes poucos bytes produzem
                CCF                       o byte expoente
                                          correcto.
                RRA                       Rodando para a esquerda e depois
                                          para a direita passa-se o byte
                                          expoente (exp. real+80 hex) para A.
                JP    P,3146,OFLW1-CLR    Se a flag «sinal» é zero, não é
                                          necessária mensagem de «overflow».
                JR    NC,31AD,REPORT-6    Indicar o excesso se a «carry» é
                                          zero.
                AND   A                   Limpar agora a «carry».
3146 OFLW-1-CLR INC   A                   O byte expoente está completo;
                JR    NZ,3151,OFLW2-CLR   mas se A é zero é necessário
                JR    C,3151,OFLW2-CLR    novo teste de overflow.
                EXX                       Se a «carry» não é um e o
                BIT   7,D                 resultado está já em forma
                EXX                       normal (bit 7 de D' é 1) deve
                JR    NZ,31AD,REPORT-6    indicar «overflow»; mas se esse
                                          bit é zero, o resultado é
                                          válido (inferior a 2↑↑127).
3151 OFLW2-CLR  LD    (HL),A              Guardar byte expoente.
                EXX                       Passar o 5.º byte do resultado
                LD    A,B                 para A para normalização,
                EXX                       isto é, o excesso de L para B'.
```

O resto da subrotina trata da normalização e é comum a todas as rotinas aritméticas.

```
3155 TEST-NORM  JR    NC,316C,NORMALISE   Se não carry, normalizar.
                LD    A,(HL)              Senão, tratar «underflow» (re-
                AND   A                   sultado zero) ou quase-underflow
3159 NEAR-ZERO  LD    A,+80               (resultado 2↑↑128):
                JR    Z,315E,SKIP-ZERO    devolver expoente a A, ver
315D ZERO-RSLT  XOR   A                   se A=0 (caso 2↑↑128) e se sim,
315E SKIP-ZERO  EXX                       produzir 2↑↑128 se número é
                AND   D                   normal; senão, produzir zero.
                CALL  2FFB,ZEROS-4/5      O expoente deve então passar
                RLCA                      a zero (se zero) ou um (se
                                          2↑↑128).
                LD    (HL),A              Restaurar o byte expoente.
                JR    C,3195,OFLOW-CLR    Saltar no caso de 2↑↑128.
```



```
                INC   HL                  Senão, pôr zero no segundo
                LD    (HL),A              byte do resultado no «stack»
                DEC   HL                  do calculador.
                JR    3195,OFLOW-CLR      Saltar para diante para trans-
                                          ferir o resultado.
```

Operação de normalização:

```
316C NORMALISE  LD    B,+20               Normalizar o resultado até
316E SHIFT-ONE  EXX                       32 decimal, 20 hex., desloca-
                BIT   7,D                 mentos para a esquerda de D'E'DE
                EXX                       (juntando A) até bit 7 de D'=1. A
                JR    NZ,3186,NORML-NOW   contém 0 após a soma, pelo que não
                RLCA                      se perde rigor; A contém o 5.º byte
                RL    E                   de B' após multiplicação ou di-
                RL    D                   visão; mas como só podem ser
                EXX                       correctos 32 bits, não perde
                RL    E                   rigor. Notar que A é rodado
                RL    D                   circularmente, através de
                EXX                       «carry» ... sendo um processo
                                          aleatório.
                DEC   (HL)                O expoente é decrementado em
                                          cada deslocamento.
                JR    Z,3159,NEAR-ZERO    Se o expoente passa a zero,
                                          o número 2↑↑129 é arredon-
                                          dado para 2↑↑128.
                DJNZ  316E,SHIFT-ONE      Volta atrás até 32 vezes.
                JR    315D,ZERO-RSLT      Se o bit 7 nunca foi 1 então
                                          o resultado deve ser zero.
```

Terminar a normalização considerando «carry».

```
3186 NORML-NOW  RLA                       Após normalização somar qualquer
                JR    NC,3195,OFLW-CLR    carry final colocada em A.
                CALL  3004,ADD-BACK       Saltar para diante se a carry
                JR    NZ,3195,OFLW-CLR    não volta à posição.
                EXX                       Se voltar, definir a
                LD    D,+80               mantissa como 0,5 e incre-
                EXX                       mentar o expoente.
                INC   (HL)                Esta acção pode conduzir a
                JR    Z,31AD,REPORT-6     overflow aritmético (caso final).
```

A parte final da subrotina envolve a passagem do resultado para os bytes que lhes foram reservados no «stack» do calculador e a redefinição dos indicadores.

```
3195 OFLOW-CLR  PUSH  HL                  Guardar o indicador-resultado.
                INC   HL                  Apontar para o byte de sinal
                                          no resultado.
                EXX                       O resultado passa dos registos
                PUSH  DE                  actuais, D'E'DE, para BCDE;
                EXX                       e depois para ACDE.
                POP   BC
                LD    A,B                 O bit de sinal é obtido da sua
                RLA                       posição temporária e transfe-
                RL    (HL)                rido para a sua posição correcta
                RRA                       como bit 7 do primeiro byte
                                          de mantissa.
```

```
                LD      (HL),A      Guarda o primeiro byte.
                INC     HL          O seguinte.
                LD      (HL),C      Guardar segundo byte.
                INC     HL          O seguinte.
                LD      (HL),D      Guardar terceiro byte.
                INC     HL          O seguinte.
                LD      (HL),E      Guardar quarto byte.
                POP     HL          Restaurar o indicador do
                                    resultado.
                POP     DE          Restaurar o indicador do
                                    segundo número.
                EXX                 Trocar registos.
                POP     HL          Restaurar endereço literal seguinte.
                EXX                 Trocar registos.
                RET                 Final.
```

Mensagem «6 — Arithmetic overflow»

```
31AD REPORT-6   RST     0008,ERROR-1    Invocar rotina de
                DEFB    +05             tratamento de erro.
```

A operação «Divisão»
(Deslocamento 05 — ver CALCULATE, adiante: «dividir»)

Esta subrotina prepara primeiro o divisor invocando PREP-M/D, indicando «excesso» aritmético se é zero; depois prepara o dividendo chamando novamente PREP-M/D, e executando retorno se for zero. Em seguida, recupera os dois números do «stack» do calculador e divide as suas mantissas recorrendo à habitual divisão, subtraindo por tentativas o divisor do dividendo e restaurando se existe transporte («carry») ou somando 1 ao quociente no caso contrário. O rigor máximo é obtido na divisão de 4 bytes, e após a subtracção dos expoentes, a subrotina envia para a parte final de MULTIPLICAÇÃO.

```
31AF divisão    CALL    3293,RE-ST-TWO  Usar virgula flutuante.
                EX      DE,HL           Trocar indicadores.
                XOR     A               A passa a 00 hex., pelo que o
                                        sinal do 1.º número irá para
                                        A.
                CALL    30C0,PREP-M/D   Preparar o divisor e produzir
                JR      C,31AD,REPORT-6 mensagem de excesso aritmético
                                        se for zero.
                EX      DE,HL           Trocar os indicadores.
                CALL    30C0,PREP-M/D   Preparar o dividendo e retorno
                RET     C               se é zero (resultado=zero).
                EXX                     Trocar os registos.
                PUSH    HL              Guardar o endereço literal seguinte.
                EXX                     Trocar os registos.
                PUSH    DE              Guardar indicador do divisor.
                PUSH    HL              Guardar indicador do dividendo.
                CALL    2FBA,FETCH-TWO  Obter os 2 números do
                                        «stack».
                EXX                     Trocar os registos.
                PUSH    HL              Guardar M1 e N1 no «stack»
                                        -máquina.
                LD      H,B             Copiar os 4 bytes do divi-
                LD      L,C             dendo dos registos B'C'CB
```



```
                EXX                     (M2, M3, M4, M5; ver
                LD      H,C             FETCH-TWO) para os registos
                                        H'L'HL.
                LD      L,B
                XOR     A               Limpar A e passar «carry» para zero.
                LD      B,+DF           B contará de -33 até -1, complemen-
                                        tos para dois, DF hex a FF, percor-
                                        rendo o ciclo se «menos», e saltará
                                        ainda, se zero, para obter maior
                                        rigor.
                JR      31E2,DIV-START  Saltar para diante (ciclo de
                                        divisão) para a primeira divisão
                                        por tentativa.
```

Entrar em seguida no ciclo de divisão.

```
31D2 DIV-LOOP   RLA                     Deslocar o resultado que resta
                RL      C               em B'C'CA, rodando os bits, re-
                EXX                     cebendo 1 de «carry» sempre que
                RL      C               esteja a um, e rodando para a
                RL      B               esquerda cada byte com carry de
                EXX                     modo a obter o deslocamento de
                                        32 bits.
31DB DIV-34TH   ADD     HL,HL           Mover o que resta do dividendo
                EXX                     em H'L'HL antes da subtracção
                ADC     HL,HL           que se segue; se um bit
                EXX                     passa a «carry», forçar
                                        um bit para o quociente,
                                        recuperando assim o bit
                JR      C,31F2,SUBN-ONLY perdido e admitindo um
                                        divisor completo, de 32 bits.
31E2 DIV-START  SBC     HL,DE           Subtrair o divisor em D'E'DE
                EXX                     do resto do dividendo em H'L'HL;
                SBC     HL,DE           não existe carry inicial (ver
                EXX                     passo anterior).
                JR      NC,31F9,NO-RSTORE Saltar para diante se não há
                                        carry.
                ADD     HL,DE           Senão, somar o divisor.
                EXX                     Depois limpar a carry de
                ADC     HL,DE           modo que não haja bit
                EXX                     para o quociente.
                AND     A
                JR      31FA,COUNT-ONE  Saltar para contador.
31F2 SUBN-ONLY  AND     A               Subtrair apenas, e continuar
                SBC     HL,DE           para pôr a um a flag carry
                EXX                     porque o bit perdido do
                SBC     HL,DE           dividendo deve ser recuperado
                EXX                     e usado para o quociente.
31F9 NO-RSTORE  SCF                     Um para o quociente em B'C'CA.
31FA COUNT-ONE  INC     B               Incrementar o contador.
                JP      M,31D2,DIV-LOOP Repetir 32 vezes para todos os bits.
                PUSH    AF              Guardar um 33.º bit para maior
                                        rigor («carry» actual).
                JR      Z,31E2,DIV-START Subtrair de novo para qualquer
                                        34.º bit; a PUSH AF acima
                                        também o guarda.
```

Nota: este salto é realizado para o local errado. Nunca será obtido um 34.º bit sem deslocar primeiro o dividendo. Assim, resultados importantes como

1/10 e 1/1000 não são arredondados da forma adequada. O arredondamento para mais nunca ocorre quando depende do 34.º bit. O salto deveria ser feito para 31DB DIV-34TH; isto é, o byte 3200 hex da ROM deveria conter DA hexadecimal (218 decimal) em vez de E1 hex (225 decimal).

```
        LD      E,A         Deslocar agora os 4 bytes
        LD      D,C         que formam a mantissa do
        EXX                 resultado de B'C'CA para D'E'DE.
        LD      E,C
        LD      D,B
        POP     AF          Pôr depois os bits 34 e 33
        RR      B           em B' para serem usados na
        POP     AF          normalização.
        RR      B
        EXX
        POP     BC          Restaurar os bytes expoentes,
                            M1 e N1.
        POP     HL          Restaurar o indicador do resultado.
        LD      A,B         Obter a diferença entre os
        SUB     C           2 bytes expoentes em A e
                            passar «carry» a 1 se necessário.
        JP      313D,DIVN-EXPT  Sair por DIVN-EXPT.
```

A subrotina de «Truncatura inteira para zero»
(Deslocamento 3A — ver CALCULATE adiante: «truncar»)

Esta subrotina (digamos I(x)) produz o resultado da truncatura inteira de x, o «último valor» para zero. Assim, I(2,4) é 2 e I(-2,4) é -2. A subrotina termina imediatamente se x se encontra na forma de «inteiro curto». Produz zero se o byte expoente de x é menor do que 81 hex (ABS x é menos de 1). Se I(x) é um «inteiro curto» a subrotina devolve-o nesse formato. Produz x se o byte expoente é A0 hex ou superior (x não possui uma parte não inteira significativa). Senão, o número correcto de bytes de x é passado a zero e, se necessário, é dividido mais um byte com uma máscara.

```
3214 truncar    LD      A,(HL)      Obter em A o byte expoente
                                    de x.
                AND     A           Se A é zero, retorno, porque
                RET     Z           x já é um inteiro pequeno.
                CP      +81         Comparar e, o expoente, com
                                    81 hex.
                JR      NC,3221,T-GR-ZERO   Saltar, se e é maior do que 80 hex.
                LD      (HL),+00    Senão, passar expoente a zero;
                LD      A,+20       pôr 32 decimal, 20 hex, em A
                JR      3272,NIL-BYTES  e saltar para diante para NIL-
                                    BYTES a fim de passar a zero
                                    todos os bits de x.
3221 T-GR-ZERO  CP      +91         Comparar e com 91 hex, 145
                                    decimal.
3223            JR      NZ,323F,T-SMALL  Saltar, se e não é 91 hex.
```

Os 26 bytes que se seguem parecem pensados para verificar se x é de facto -65536 decimal, isto é, 91 80 00 00 00, e se for, passá-lo para 00 FF 00 00 00. Isto é um erro. Como já se afirmou no byte 303B acima, o sistema

do Spectrum não pode tratar este número. O resultado aqui consiste simplesmente em produzir o valor -1 com INT (-65536). É uma pena, porque o número ficaria bem como estava. O remédio parece consistir simplesmente em omitir do programa os 28 bytes entre 3223 e 323E, inclusive.

```
3225            INC     HL          HL é apontado para o 4.º byte
                INC     HL          de x, onde terminam os 17
                INC     HL          bits da parte inteira de x
                                    após o primeiro bit.
                LD      A,+80       O primeiro bit é obtido em A,
                AND     (HL)        usando uma máscara 80 hex.
                DEC     HL          Este bit e os 8 anteriores
                OR      (HL)        são comparados com zero.
                DEC     HL          HL aponta para o segundo
                                    byte de x.
                JR      NZ,3233,T-FIRST  Se já não-zero, o teste
                                    pode terminar.
                LD      A,+80       Senão, o teste de -65536 é
                XOR     (HL)        completado: 91 80 00 00 00
                                    deixará a flag «zero» em um.
3233 T-FIRST    DEC     HL          HL aponta para o primeiro
                                    byte de x.
                JR      NZ,326C,T-EXPNENT  Se «zero» em zero, salto.
                LD      (HL),A      O 1.º byte é passado a zero.
                INC     HL          HL aponta para o 2.º byte.
                LD      (HL),+FF    O 2.º byte é passado a FF.
                DEC     HL          HL aponta novamente para
                                    o primeiro byte.
                LD      A,+18       Os últimos 24 bits serão zero.
                JR      3272,NIL-BYTES  O salto para NIL-BYTES
                                    completa o número
                                    00 FF 00 00 00.
```

Se o byte expoente de x se encontra entre 81 e 90 hex (129 e 144 decimal) inclusive, I(x) é um «inteiro pequeno», e será comprimido num ou dois bytes. Mas, primeiro, verifica-se se x é grande.

```
323F T-SMALL    JR      NC,326D,X-LARGE  Saltar para expoente 92
                                    ou mais (seria melhor
                                    também para 91).
                PUSH    DE          Guardar STKEND em DE.
                CPL                 Gama 129 <=A <=144 passa
                                    a 126 >=A >=111.
                ADD     A,+91       A gama é agora 15 dec >=A >=0.
                INC     HL          Apontar HL para segundo byte.
                LD      D,(HL)      Segundo byte para D.
                INC     HL          Apontar HL para 3.º byte.
                LD      E,(HL)      Terceiro byte para E.
                DEC     HL          Apontar HL para 1.º byte de novo.
                DEC     HL
                LD      C,+00       Pressupor um número positivo.
                BIT     7,D         Verificar agora se negativo
                                    (bit 7 a um).
                JR      Z,3252,T-NUMERIC  Saltar se positivo.
                DEC     C           Alterar o sinal.
3252 T-NUMERIC  SET     7,D         Inserir bit numérico 1 em D.
                LD      B,+08       Verificar se A >=8 (um byte
                SUB     B           apenas) ou se necessita de 2 bytes.
```

```
                ADD     A,B         Deixar A como está.
                JR      C,325E,T-TEST   Saltar se necessita de 2 bytes.
                LD      E,D         Pôr um em E.
                LD      D,+00       E passar D a zero.
                SUB     B           Agora 1 <=A <=7 para contar
                                    os deslocamentos necessários.
325E T-TEST     JR      Z,3267,T-STORE  Saltar se não precisa de
                                    deslocamento.
                LD      B,A         B contará os deslocamentos.
3261 T-SHIFT    SRL     D           Deslocar D e E para a direita B
                RR      E           vezes para produzir n.º correcto.
                DJNZ    3261,T-SHIFT    Repetir até B ser zero.
3267 T-STORE    CALL    2D8E,INT-STORE  Guardar resultado no «stack».
                POP     DE          Passar STKEND para DE.
                RET                 Final.
```

Resta considerar os valores elevados de x.

```
326C T-EXPNENT  LD      A,(HL)      Obter o byte expoente de x
                                    em A.
326D X-LARGE    SUB     +A0         Subtrair 160 decimal, A0
                                    hex, de e.
                RET     P           Retorno se x não tem uma
                                    parte não inteira significativa
                                    (se o expoente verdadeiro
                                    fosse reduzido a zero, a vír-
                                    gula binária surgiria no fim
                                    ou após os 4 bytes da
                                    mantissa).
                NEG                 Senão, negar o resto; isto
                                    dá o número de bits que
                                    passam a zero (n.º de bits
                                    após «vírgula binária»).
```

Podem limpar-se agora os bits da mantissa.

```
3272 NIL-BYTES  PUSH    DE          Guardar o valor actual de DE
                                    (STKEND).
                EX      DE,HL       Fazer HL apontar para um
                                    depois do 5.º byte.
                DEC     HL          HL aponta agora para
                                    o 5.º byte de x.
                LD      B,A         Obter o n.º de bits a
                SRL     B           passar a zero em B, e
                SRL     B           dividi-lo por 8 para obter
                SRL     B           o n.º de bytes inteiros.
                JR      Z,3283,BITS-ZERO    Saltar para diante se o
                                    resultado é zero.
327E BYTE-ZERO  LD      (HL),+00    Senão, passar bytes a zero;
                DEC     HL          B conta-os.
                DJNZ    327E,BYTE-ZERO

3283 BITS-ZERO  AND     +07         Obter A (mod 8); é o n.º de
                                    bits ainda a passar a zero.
                JR      Z,3290,IX-END   Saltar para o fim se não
                                    há mais nada a fazer.
                LD      B,A         B contará agora os bits.
                LD      A,+FF       Preparar a máscara.
328A LESS-MASK  SLA     A           Em cada ciclo um zero entra
                DJNZ    328A,LESS-MASK  na máscara pela direita,
```



```
                                    acabando por se obter uma
                                    máscara com o comprimento certo.
                AND     (HL)        Os bits não desejados de (HL)
                LD      (HL),A      são perdidos.
3290 IX-END     EX      DE,HL       Enviar indicador para HL.
                POP     DE          Enviar STKEND para DE.
                RET                 Final.
```

A subrotina «RE-STACK TWO»

Esta subrotina é invocada para guardar em «stack» dois «inteiros pequenos» na forma de vírgula flutuante com cinco bytes, para as operações binárias de adição, multiplicação e divisão. Faz isto, invocando a subrotina seguinte duas vezes.

```
3293 RE-ST-TWO  CALL    3296,RESTK-SUB  Chama a subrotina, e
                                    continua através dela para
                                    a segunda chamada.
3296 RESTK-SUB  EX      DE,HL       Troca os indicadores
                                    em cada chamada.
```

A subrotina «RE-STACK»
(Deslocamento 3D — ver CALCULATE, adiante: «re-stack»)

Esta subrotina é invocada para guardar novamente um número em «stack» (que poderá ser um «inteiro pequeno») na forma de vírgula flutuante com cinco bytes. É usada para um número único por ARCTAN e também, através do deslocamento do calculador, por EXP, LN e «obter-arg».

```
3297 RE-STACK   LD      A,(HL)      Se o 1.º byte não é zero,
                AND     A           retorno — o número não é
                RET     NZ          um «inteiro pequeno».
                PUSH    DE          Guardar o «outro» indicador em DE.
                CALL    2D7F,INT-FETCH  Obter o sinal em C e o
                                    número em DE.
                XOR     A           Limpar o registo A.
                INC     HL          Apontar para a 5.ª posição.
                LD      (HL),A      Passar 5.º byte para zero.
                DEC     HL          Apontar para a 4.ª posição.
                LD      (HL),A      Passar 4.º byte para zero:
                                    os bits 2 e 3 conterão a
                                    mantissa.
                LD      B,+91       Passar B para 145 dec para
                                    expoente, isto é, para até
                                    16 bits no inteiro.
                LD      A,D         Verificar se D é zero para
                AND     A           que no máximo se usem 8 bits.
                JR      NZ,32B1,RS-NRMLSE   Saltar se precisa de mais bits.
                OR      E           Verificar agora E.
                LD      B,D         Guardar o zero em B (dará ex-
                                    poente zero se E=0 também).
                JR      Z,32BD,RS-STORE     Saltar se E é zero.
                LD      D,E         Passar E para D (D era zero,
                                    E não).
                LD      E,B         Passar E para zero agora.
```

```
                LD      B,+89           Passar B a 137 dec para ex-
                                        poente — agora não excede
                                        os 8 bits.
32B1 RS-NRMLSE  EX      DE,HL           Apontar para DE, número em HL.
32B2 RSTK-LOOP  DEC     B               Decrementar o expoente em
                                        cada deslocamento.
                ADD     HL,HL           Deslocar o número para a
                                        direita uma posição.
                JR      NC,32B2,RSTK-LOOP Até «carry» estar a um.
                RRC     C               Bit de sinal para flag «carry».
                RR      H               Inseri-lo no seu lugar quando
                RR      L               o número é deslocado para
                                        trás uma posição.
                EX      DE,HL           Indicador do byte 4 de novo em HL.
32BD RS-STORE   DEC     HL              Apontar para a terceira posição.
                LD      (HL),E          Guardar o terceiro byte.
                DEC     HL              Apontar para a segunda posição.
                LD      (HL),D          Guardar o segundo byte.
                DEC     HL              Apontar para a primeira posição.
                LD      (HL),B          Guardar o byte expoente.
                POP     DE              Restaurar o «outro» indicador
                                        em DE.
                RET                     Final.
```


A tabela de constantes

A primeira tabela guarda os cinco números zero, um, 0,5, 0,5 PI e dez, bastante úteis e frequentemente necessários. Os números são guardados numa forma condensada, dilatada pela subrotina STACK LITERALS (ver adiante), de modo a adoptarem a forma em vírgula flutuante.

```
                    Dados           Constante   Quando a mantissa dá:
                                                exp. mantissa (hex):

32C5 stk-zero       DEFB +00        zero        00 00 00 00 00
                    DEFB +B0
                    DEFB +00

32C8 stk-um         DEFB +40        um          00 00 01 00 00
                    DEFB +B0
                    DEFB +00
                    DEFB +01

32CC stk-meio       DEFB +30        metade      80 00 00 00 00
                    DEFB +00

32CE stk-pi/2       DEFB +F1        meio PI     81 49 0F DA A2
                    DEFB +49
                    DEFB +0F
                    DEFB +DA
                    DEFB +A2

32D3 stk-dez        DEFB +40        dez         00 00 0A 00 00
                    DEFB +B0
                    DEFB +00
                    DEFB +0A
```

A tabela de endereços

Esta segunda tabela serve para determinar os endereços das sessenta e seis subrotinas operacionais do calculador. Os deslocamentos usados para indexar a tabela são obtidos a partir dos códigos de operação usados em SCANNING, ver 2734, etc., ou dos literais que se seguem a uma instrução RST 0028.

	Deslocamento	Etiqueta	Endereço		Deslocamento	Etiqueta	Endereço
32D7	00	saltar-verdade	8F 36	3311	1D	val	DE 35
32D9	01	trocar	3C 34	3313	1E	len	74 36
32DB	02	apagar	A1 33	3315	1F	sin	B5 37
32DD	03	subtrair	0F 30	3317	20	cos	AA 37
32DF	04	multiplicar	CA 30	3319	21	tan	DA 37
32E1	05	dividir	AF 31	331B	22	asn	33 38
32E3	06	a-potência	51 38	331D	23	acs	43 38
32E5	07	ou	1B 35	331F	24	atn	E2 37
32E7	08	n.ºe-n.º	24 35	3321	25	ln	13 37
32E9	09	n.ºmenor-igl	3B 35	3323	26	exp	C4 36
32EB	0A	n.ºmaior-igl	3B 35	3325	27	int	AF 36
32ED	0B	n.ºº-nigl	3B 35	3327	28	sqr	4A 38
32EF	0C	n.ºmaior	3B 35	3329	29	sgn	92 34
32F1	0D	n.ºmenor	3B 35	332B	2A	abs	6A 34
32F3	0E	nºº-igual	3B 35	332D	2B	peek	AC 34
32F5	0F	somar	14 30	332F	2C	in	A5 34
32F7	10	cad-e-n.º	2D 35	3331	2D	usr-n.º	B3 34
32F9	11	cad-menor-igl	3B 35	3333	2E	str$	1F 36
32FB	12	cad-maior-igl	3B 35	3335	2F	chr$	C9 35
32FD	13	cads-nigl	3B 35	3337	30	not	01 35
32FF	14	cad-maior	3B 35	3339	31	copiar	C0 33
3301	15	cad-menor	3B 35	333B	32	n-mod-m	A0 36
3303	16	cads-igual	3B 35	333D	33	saltar	86 36
3305	17	cads-soma	9C 35	333F	34	stk-dados	C6 33
3307	18	val$	DE 35	3341	35	dec-jr-nz	7A 36
3309	19	usr-$	BC 34	3343	36	menor-0	06 35
330B	1A	leitura	45 36	3345	37	maior-0	F9 34
330D	1B	negação	6E 34	3347	38	fim-calc	9B 36
330F	1C	código	69 36	3349	39	obter-arg	83 37

	Deslocamento	Etiqueta	Endereço		Deslocamento	Etiqueta	Endereço
334B	3A	truncar	14 32	3353	3E	série-06 etc.	49 34
334D	3B	vf-calc-2	A2 33	3355	3F	stk-zero etc.	1B 34
334F	3C	e-para-vf	4F 2D	3357	40	st-mem-0 etc.	2D 34
3351	3D	re-«stack»	97 32	3359	41	obter-mem-0 etc.	0F 34

Nota: As últimas quatro subrotinas são de uso geral, entrando-se nelas com um parâmetro que é uma cópia dos cinco bytes da direita do literal original. Segue-se o conjunto completo:

Deslocamento 3E: série-06, série-08 e série-0C; literais 86, 88 e 8C.
Deslocamento 3F: stk-zero, stk-um, stk-meio, stk-pi/2 e stk-dez; literais A0 a A4.
Deslocamento 40: st-mem-0, st-mem-1, st-mem-2, st-mem-3, st-mem-4 e st-mem-5; literais C0 a C5.
Deslocamento 41: obter-mem-0, obter-mem-1, obter-mem-2, obter-mem-3, obter-mem-4 e obter-mem-5; literais E0 a E5.

A subrotina «Calcular» (CALCULATE)

Esta subrotina é usada para realizar cálculos em vírgula flutuante. Estes podem ser considerados como pertencendo a três tipos:

1. Operações binárias, por exemplo, adição, onde dois números em forma vírgula flutuante são somados de modo a produzirem um «último valor».

2. Operações unárias, como o seno, nas quais o «último valor» é alterado de modo a dar a função resultante apropriada como novo «último valor».

3. Operações de manipulação, por exemplo, st-mem-0, onde o «último valor» é copiado para os primeiros cinco bytes da área de memória do calculador.

As operações a realizar são especificadas como uma série de bytes de dados, os literais, que se seguem a uma instrução RST 0028 que invoca essa subrotina. O último literal da lista é sempre «38», o que conduz ao final da operação.

No caso de uma única operação a realizar, o deslocamento da operação pode ser passado ao calculador no registo B, e pode ser realizada a operação «3B», de cálculo simples.

É igualmente possível invocar esta rotina por recorrência, ou seja, a partir de si mesma, e neste caso é possível usar a variável de sistema BREG como contador que controla o número de operações que são realizadas antes do retorno.

A primeira parte desta subrotina é complicada, mas realiza, essencialmente, as duas tarefas de armazenamento dos valores requeridos nos regis-

tos, e de produção de um deslocamento, e possivelmente um parâmetro, a partir do literal que está a ser considerado.

O deslocamento é usado para indexar a tabela de endereços do calculador (ver acima), para determinar o endereço da subrotina requerida. O parâmetro é usado quando são invocadas as subrotinas de uso geral.

Nota: Um número em vírgula flutuante pode na realidade ser um conjunto de parâmetros de cadeia.

```
335B CALCULATE   CALL   358F,STK-PNTRS   Pressupor uma operação unária
                                         e portanto apontar HL para o
                                         início do «último valor» no
                                         «stack» do calculador e DE para
                                         um depois deste n.º em vírgula
                                         flutuante (STKEND).
335E GEN-ENT-1   LD     A,B              Transferir um único deslocamento
                 LD     (BREG),A         de operação para BREG
                                         ou, ao usar a subrotina de
                                         modo recorrente, passar o
                                         parâmetro para BREG para uso
                                         como contador.
3362 GEN-ENT-2   EXX                     Endereço de retorno da subro-
                 EX     (SP),HL          tina em H'L'. Isto guarda o
                 EXX                     indicador no primeiro lite-
                                         ral. A entrada no calculador
                                         por GEN-ENT-2 é usada sempre
                                         que BREG é usado como con-
                                         tador e não pode ser afec-
                                         tada.
3365 RE-ENTRY    LD     (STKEND),DE      Entra-se num ciclo que trata
                                         cada literal da lista que se
                                         segue à instrução que o chama;
                                         primeiro, define sempre STKEND.
                 EXX                     Passar aos registos alterna-
                 LD     A,(HL)           tivos, e obter o literal
                                         para este ciclo.
                 INC    HL               Levar H'L' a apontar para o
                                         literal seguinte.
336C SCAN-ENT    PUSH   HL               Este indicador é guardado
                                         no «stack»-máquina. É usado
                                         SCAN-ENT pela subrotina «cál-
                                         culo simples» para descobrir
                                         a subrotina requerida.
                 AND    A                Verificar o registo A.
                 JP     P,3380,FIRST-3D  Separar os literais simples
                                         dos de uso múltiplo. Saltar
                                         se literais 00-3D.
                 LD     D,A              Guardar o literal em D.
                 AND    +60              Continuar apenas com bits 5 e 6.
                 RRCA                    4 deslocamentos passam-nos
                 RRCA                    agora a bits 1 e 2.
                 RRCA
                 RRCA
                 ADD    A,+7C            Os deslocamentos requeridos
                 LD     L,A              são 3E-41, e L conterá o dobro
                                         do deslocamento pretendido.
                 LD     A,D              Construir o parâmetro partindo
                 AND    +1F              dos bits 0, 1, 2, 3 e 4 do
                                         literal; manter parâmetro em A.
```



```
                 JR     338E,ENT-TABLE   Saltar para diante para desco-
                                         brir o endereço da subrotina
                                         requerida.
3380 FIRST-3D    CP     +18              Saltar para diante se realiza
                 JR     NC,338C,DOUBLE-A operação unária.
                 EXX                     Todas as subrotinas que
                 LD     BC,+FFFB         realizam operações binárias exigem
                 LD     D,H              que HL aponte para o 1.º operando
                 LD     E,L              e DE para o segundo operando
                 ADD    HL,BC            (o «último valor») tal como
                 EXX                     ocorrem no «stack» do calculador.
338C DOUBLE-A    RLCA                    Como cada entrada na tabela dos
                 LD     L,A              endereços ocupa dois bytes o
                                         deslocamento é duplicado.
338E ENT-TABLE   LD     DE,+32D7         Endereço base da tabela.
                 LD     H,+00            O endereço da entrada requerida
                 ADD    HL,DE            na tabela é formado em HL; e
                 LD     E,(HL)           o endereço da subrotina reque-
                 INC    HL               rida é carregado no par de
                 LD     D,(HL)           registos DE.
                 LD     HL,+3365         O endereço RE-ENTRY de 3365
                 EX     (SP),HL          é posto no «stack»-máquina sob
                 PUSH   DE               o endereço da subrotina.
                 EXX                     Retorno aos registos
                                         principais.
                 LD     BC,(STKEND-alto) O valor actual de BREG é
                                         transferido para o registo B
                                         definindo assim o deslocamento
                                         da operação.
                                         (ver COMPARAÇÃO em 353B).
33A1 apagar      RET                     Um salto indirecto para a
                                         subrotina requerida.
```

A subrotina «Apagar» (DELETE)
(Deslocamento 02: «apagar»)

Esta subrotina contém apenas a instrução RET em 33A1, acima. O literal «02» leva a considerar esta subrotina como uma operação binária que deve ser iniciada com um primeiro número endereçado pelo par de registos HL e um segundo número endereçado pelo par de registos DE, sendo o resultado novamente endereçado pelo par de registos HL.

A simples instrução RET conduz portanto a considerar o primeiro número como o «último valor» resultante, aceitando o segundo número como tendo sido eliminado. O número não foi evidentemente eliminado da memória, mas mantém-se inactivo e provavelmente será dentro em pouco substituído.

A subrotina «Operação única»
(Deslocamento 3B: «vf-calc-2»)

Esta subrotina é apenas invocada por SCANNING em 2757 hex e é usada para realizar uma única operação aritmética. O deslocamento que especifica qual a operação a realizar é fornecido ao calculador no registo B e, subsequentemente, transferido para a variável de sistema BREG.

O efeito da chamada a esta subrotina consiste essencialmente em executar um salto para a subrotina apropriada à operação em causa.

```
33A2 vf-calc-2   POP   AF              Eliminar endereço RE-ENTRY.
                 LD    A,(BREG)        Transferir deslocamento para A.
                 EXX                   Aceitar registos alternativos.
                 JR    336C,SCAN-ENT   Saltar atrás para descobrir o
                                       endereço requerido; guardar o
                                       endereço RE-ENTRY e saltar
                                       para a subrotina da operação.
```

A subrotina «Verificar 5 espaços»

Esta subrotina verifica se existe espaço suficiente em memória para o acrescento de um número em vírgula flutuante de 5 bytes ao «stack» do calculador.

```
33A9 TEST-5-SP   PUSH  DE              Guardar DE.
                 PUSH  HL              Guardar HL.
                 LD    BC,+0005        Especificar o teste (5 bytes).
                 CALL  1F05,TEST-ROOM  Realizar o teste.
                 POP   HL              Restaurar HL.
                 POP   DE              Restaurar DE.
                 RET                   Final.
```

A subrotina «Guardar número no 'stack'»

Esta subrotina é invocada duas vezes por BEEP e por SCANNING para copiar STKEND para DE. Passa um número em vírgula flutuante para o «stack» do calculador e redefine STKEND a partir de DE. Invoca «MOVE-FP» para fazer este deslocamento.

```
33B4 STACK-NUM   LD    DE,(STKEND)     Copiar STKEND para DE como
                                       endereço de destino.
                 CALL  33C0,MOVE-FP    Deslocar o número.
                 LD    (STKEND),DE     Redefinir STKEND a partir de DE.
                 RET                   Final.
```

A subrotina «Mover um número em vírgula flutuante»
(Deslocamento 31: «copiar»)

Esta subrotina passa um número em vírgula flutuante para o topo do «stack» do calculador (3 casas) ou do topo do «stack» para a área de memória do calculador (uma casa). É igualmente invocada através do calculador quando copia simplesmente o número no topo do «stack» do calculador, o «último valor», ampliando assim o «stack» em cinco bytes.

```
33C0 MOVE-FP     CALL  33A9,TEST-5-SP  Verifica-se o espaço.
                 LDIR                  Movem-se os 5 bytes em causa.
                 RET                   Final.
```

A subrotina «Guardar literais»
(Deslocamento 34: «stk-dados»)

Esta subrotina coloca no «stack» do calculador, como «último valor», o número em vírgula flutuante que lhe é fornecido como 2, 3, 4 ou 5 literais.

Quando invocada usando o deslocamento «34», os literais seguem-se a «34» na lista dos literais; quando invocada pelo Gerador de Séries (ver abaixo), os literais são fornecidos pela subrotina que pediu a produção de uma série; e quando invocada por SKIP CONSTANTS e STACK A CONSTANT, os literais são obtidos a partir da tabela de constantes do calculador (32C5-32D6).

Em cada caso, o primeiro literal fornecido é dividido por 40 hex, e o quociente inteiro mais 1 determina se serão considerados, 1, 2, 3 ou 4 literais da fonte para formar a mantissa do número. Todos os bytes não preenchidos nos cinco bytes que irão formar o número em vírgula flutuante são passados a zero. O primeiro literal é igualmente usado para determinar o expoente, após reduzir módulo 40 hex, a menos que o resto seja zero, caso em que se usa o segundo literal, tal como está, sem reduzir o valor indicado. Em qualquer dos casos, soma-se 50 hex ao literal, tendo o byte expoente ampliado, e (o expoente verdadeiro e' mais 80 hex). O resto dos cinco bytes são guardados, incluindo quaisquer zeros necessários, e a subrotina termina.

```
33C6 STK-DATA    LD    H,D             Esta subrotina realiza a
                 LD    L,E             operação de soma de um
                                       «último valor» ao «stack» do
                                       calculador; HL passa portanto
                                       a apontar para um mais do que
                                       o «último valor» actual, isto
                                       é, para o resultado.
33C8 STK-CONST   CALL  33A9,TEST-5-SP  Verificar agora se existe de
                                       facto espaço.
                 EXX                   Passar aos registos alterna-
                 PUSH  HL              tivos e guardar o indicador
                 EXX                   do literal seguinte.
                 EX    (SP),HL         Comutar o indicador do resul-
                                       tado e o do literal seguinte.
                 PUSH  BC              Guardar BC brevemente.
                 LD    A,(HL)          O 1.º literal é posto em A
                 AND   +C0             e dividido por 40 hex para dar
                 RLCA                  os inteiros 0, 1, 2 ou 3.
                 RLCA
                 LD    C,A             O valor inteiro é transferido
                 INC   C               para C e incrementado, dando
                                       assim a gama 1, 2, 3 ou 4
                                       para o número de literais
                                       que serão necessários.
                 LD    A,(HL)          O literal é de novo obtido,
                 AND   +3F             reduzido mód. 40 hex e elimi-
                 JR    NZ,33DE,FORM-EXP nado como não apropriado se
                 INC   HL              o resto é zero; neste caso,
                 LD    A,(HL)          obtém-se o literal seguinte,
                                       usando-o sem redução.
33DE FORM-EXP    ADD   A,+50           O expoente, e, é formado
                 LD    (DE),A          por soma de 50 hex e passa-
                                       do ao «stack» do calculador
                                       como primeiro dos cinco
                                       bytes do resultado.
                 LD    A,+05           O número de literais especi-
                 SUB   C               ficado em C é obtido da
                 INC   HL              fonte e introduzido nos
                 INC   DE              bytes do resultado.
                 LD    B,+00
                 LDIR
```

```
                POP     BC          Restaurar BC.
                EX      (SP),HL     Devolver o indicador do re-
                EXX                 sultado a HL e o indicador
                POP     HL          do literal seguinte à sua
                                    posição habitual em H' e L'.
                EXX
                LD      B,A         O n.º de bytes zero necessá-
                XOR     A           rios nesta fase é dado por
33F1 STK-ZEROS  DEC     B           5-C-1; e este n.º de zeros é
                RET     Z           somado ao resultado de modo
                LD      (DE),A      a produzir os 5 bytes desejados.
                INC     DE
                JR      33F1,STK-ZEROS
```

A subrotina «Eliminar constantes»

Entra-se nesta subrotina com o endereço base da tabela de constantes do calculador no par de registos HL e no registo A um parâmetro que indica qual das cinco constantes está a ser pedida.

A subrotina realiza as operações nulas de carga dos cinco bytes de cada constante não pretendida para as posições 0000, 0001, 0002, 0003 e 0004 no início da ROM, até ser atingida a constante desejada.

A subrotina termina com o endereço base da constante requerida no par de registos HL.

```
33F7 SKIP-CONS  AND     A           A subrotina termina se o parâ-
33F8 SKIP-NEXT  RET     Z           metro é zero, ou quando a cons-
                                    tante requerida ainda não foi
                                    atingida.
                PUSH    AF          Guardar o parâmetro.
                PUSH    DE          Guardar o indicador do resultado.
                LD      DE,+0000    Endereço falso.
                CALL    33C8,STK-CONST  Guardar imaginariamente uma
                                    constante expandida.
                POP     DE          Restaurar o indicador do resultado.
                POP     AF          Restaurar o parâmetro.
                DEC     A           Contar os ciclos.
                JR      33F8,SKIP-NEXT  Saltar atrás para consi-
                                    derar o valor do contador.
```

A subrotina «Posição em memória»

Esta subrotina determina o endereço base de cada porção de 5 bytes da área de memória do calculador de onde ou para onde se pretende deslocar um número em vírgula flutuante a partir do «stack» do calculador. Esta operação é realizada somando cinco vezes o parâmetro fornecido ao endereço-base da área que é guardada no par de registos HL.

Note-se que quando é tratada uma variável FOR-NEXT os indicadores são trocados de tal modo que a variável é tratada como se fosse a área de memória do calculador (ver o endereço 1D20).

```
3406 LOC-MEM    LD      C,A         Copiar o parâmetro para 6.
                RLCA                Duplicar o parâmetro.
```



```
                RLCA                Duplicar esse resultado.
                ADD     A,C         Somar o valor do parâmetro
                                    a fim de obter cinco vezes o
                                    valor original.
                LD      C,A         Este resultado é desejado no
                LD      B,+00       par de registos BC.
                ADD     HL,BC       Produzir o novo endereço-base.
                RET                 Final.
```

A subrotina «Obter na área de memória»
(Deslocamentos E0 a E5: «obter-mem-0» a «obter-mem-5»)

Esta subrotina é invocada usando os literais E0 a E5, e o parâmetro derivado destes literais é guardado no registo A. A subrotina invoca POSIÇÃO DE MEMÓRIA a fim de colocar o endereço-base requerido no par de registos HL, e DESLOCAR UM NÚMERO EM VÍRGULA FLUTUANTE para copiar os cinco bytes em causa, da área de memória do calculador para o topo do «stack» do calculador, de modo a formar um novo «último valor».

```
340F obter-mem-0  PUSH    DE          Guardar o indicador do resultado.
      etc.        LD      HL,(MEM)    Obter o indicador da área de
                                      memória actual (ver acima).
                  CALL    3406,LOC-MEM  Determina o endereço-base.
                  CALL    33C0,MOVE-FP  Os cinco bytes são movidos.
                  POP     HL          Definir o indicador do resultado.
                  RET                 Final.
```

A subrotina «Guardar uma constante»
(Deslocamentos A0 a A4: «stk-zero», «stk-um», «stk-meio», «stk-pi/2» e «stk-dez»)

Esta subrotina usa ELIMINAR CONSTANTES para determinar o endereço-base das constantes requeridas na tabela de constantes do calculador e em seguida invoca GUARDAR LITERAIS3 entrando em STK-CONST, a fim de transformar a forma dilatada da constante em «último valor» no «stack» do calculador.

```
341B stk-zero   LD      H,D         HL passa a guardar o indi-
      etc.                          cador do resultado.
                LD      L,E
                EXX                 Passa aos registos alternativos e
                PUSH    HL          guarda o indicador literal seguinte.
                LD      HL,+32C5    Endereço-base da tabela de
                                    constantes do calculador.
                EXX                 Voltar aos registos principais.
                CALL    33F7,SKIP-CONS  Determinar o endereço-base.
                CALL    33C8,STK-CONST  Expandir a constante.
                EXX
                POP     HL          Restaurar o indicador literal seguinte.
                EXX
                RET                 Final.
```

A subrotina «Guardar na área de memória»
(Deslocamentos C0 a C5: «st-mem-0» a «st-mem-5»)

Esta subrotina é invocada usando os literais C0 a C5 e o parâmetro obtido destes literais é guardado no registo A. Esta subrotina é muito semelhante à subrotina OBTER EM MEMÓRIA, mas os indicadores de fonte e de destino são trocados.

```
342D st-mem-0   PUSH  HL          Guardar o indicador do resultado.
      etc.      EX    DE,HL       Fonte para DE.
                LD    HL,(MEM)    Guardar o indicador da área de
                                  memória actual.
                CALL  3406,LOC-MEM Define o endereço base.
                EX    DE,HL       Troca indicadores de
                                  fonte e destino.
                CALL  33C0,MOVE-FP Move os cinco bytes.
                EX    DE,HL       «Último valor»+5, ou seja,
                                  STKEND, para DE.
                POP   HL          Indicador do resultado em HL.
                RET               Final.
```

Note-se que os indicadores HL e DE se mantêm onde estavam, apontando para STKEND-5 e STKEND, respectivamente, de tal modo que o «último valor» se matém no «stack» do calculador. Se necessário, pode ser eliminado usando «apagar».

A subrotina «Troca»
(Deslocamento 01: «troca»)

Esta operação binária «troca» o primeiro número com o segundo número, isto é, são trocados os dois números que se encontram mais acima no «stack» do calculador.

```
343C EXCHANGE   LD    B,+05       Estão envolvidos 5 bytes.
343E SWAP-BYTE  LD    A,(DE)      Cada byte do 2.º número.
                LD    C,(HL)      Cada byte do 1.º número.
                EX    DE,HL       Comutar fonte e destino.
                LD    (DE),A      Agora para o 1.º número.
                LD    (HL),C      Agora para o 2.º número.
                INC   HL          Passar a considerar o par
                INC   DE          de bytes seguintes.
                DJNZ  343E,SWAP-BYTE Trocar os cinco bytes.
                EX    DE,HL       Obter os indicadores certos,
                                  dado que 5 é ímpar.
                RET               Final.
```

A subrotina «Gerador em série»
(Deslocamentos 86, 88 e 8C: «série-06», «série-08» e «série-0C»)

Esta importante subrotina produz séries de polinómios Chebyshev que são usadas para obter o resultado aproximado de SIN, ATN, LN e EXP e portanto para determinar as outras funções aritméticas que dependem destas (COS, TAN, ASN, ACS, ** SQR).

Estes polinómios são produzidos, para n=1,2,. . ., pela relação recorrente:

$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z),$$

onde $T_n(z)$ é o enésimo polinómio de Chebyshev em z.
Esta série, de facto, produz:

$$T_0,\ 2T_1,\ 2T_2,\ \ldots,\ 2T_{n-1},$$

onde n é 6 no caso de SIN, 8 no de EXP e 12 decimal no caso de LN e ATN.

Os coeficientes das potências de z nestes polinómios podem ser encontrados em *Handbook of Mathematical Functions* de M. Abramowitz e I. A. Stegun (Dover 1965), página 795.

Os programas Basic que mostram a produção de cada uma das quatro funções são apresentados no Apêndice.

Em termos simples, esta subrotina é invocada com o «último valor» no «stack» do calculador, digamos Z, definido como um número que se encontra numa relação simples com o argumento, digamos X, quando a tarefa consiste em avaliar, por exemplo, SIN X. A subrotina que chama esta, fornece igualmente a lista de constantes requeridas (seis constantes no caso de SIN). O GERADOR DE SÉRIES manipula, em seguida, os seus dados e devolve a rotina que invocou um «último valor» que apresenta uma relação simples com a função requerida, por exemplo, SIN X.

Esta subrotina pode ser considerada como tendo quatro partes principais:

1. A definição do contador de ciclo:
   A rotina que invoca, passa os seus parâmetros para o registo A, para uso como contador. Entra-se no calculador em GEN-ENT-1 de tal modo que o contador possa ser definido.

```
3449 série-06   LD    B,A         Passa o parâmetro para B.
     etc.       CALL  335E,GEN-ENT-1 É de facto uma RST 0028
                                  mas define o contador.
```

2. O tratamento do «último valor», Z:
   O ciclo do gerador requer 2•Z em mem-0, zero em mem-2 e um «último valor» igual a zero.

```
                                  «stack» do calculador.
           DEFB  +31,copiar       Z,Z
           DEFB  +0F,somar        2•Z
           DEFB  +C0,st-mem-0     2•Z        mem-0 contém 2•Z
           DEFB  +02,apagar       —
           DEFB  +A0,stk-zero     0
           DEFB  +C2,st-mem-2     0          mem-2 contém 0
```

3. O ciclo principal:
   A série é produzida por um ciclo, usando BREG como contador; as constantes na subrotina que invoca são guardadas separadamente invocando STK-DADOS; reentra-se no calculador em GEN-ENT-2, a fim de não afectar o valor de BREG; e a série é construída com a forma:

$$B(R) = 2 \cdot Z \cdot B(R-1) - B(R-2) + A(R)$$

para R=1,2,...,N, onde A(1),A(2),...,A(N) são as constantes fornecidas pela subrotina que invoca (SIN, ATN, LN e EXP) e B(0)=0=B(−1).

O ciclo de ordem (R+1) começa com B(R) no «stack» e com 2*Z, B(R−2) e B(R−1) em mem-0, mem-1 e mem-2, respectivamente.

```
3453 G-LOOP     DEFB  +31,copiar          B(R), B(R)
                DEFB  +E0,obter-mem-0     B(R), B(R), 2*Z
                DEFB  +04,multiplicar     B(R), 2*B(R)*Z
                DEFB  +E2,obter-mem-2     B(R), 2*B(R)*Z, B(R-1)
                DEFB  +C1,st-mem-1        mem-1 contém B(R-1)
DEFB +38,fim-calc DEFB +03,subtrair       B(R), 2*B(R)*Z-B(R-1)
```

A constante seguinte é colocada no «stack» do calculador.

```
                CALL  33C6,STK-DATA   B(R), 2*B(R)*Z-B(R-1), A(R+1)
```

Reentra-se no calculador sem perturbar BREG.

```
                CALL  3362,GEN-ENT-2
                DEFB  +0F,somar       B(R), 2*B(R)*Z-B(R-1)+A(R+1)
                DEFB  +01,trocar      2*B(R)*Z-B(R-1)+A(R+1), B(R)
                DEFB  +C2,st-mem-2    mem-2 contém B(R).
                DEFB  +02,apagar      2*B(R)*Z-B(R-1)+A(R1) =
                                      B(R11)
                DEFB  +35,dec-jr-nz   B(R+1)
                DEFB  +EE,para 3453,G-LOOP
```

4. A subtracção de B(N-2):
O ciclo acima deixa B(N) no «stack» sendo o resultado requerido por B(N)-B(N-2).

```
                DEFB  +E1,obter-mem-1 B(N), B(N-2)
                DEFB  +03,subtrair    B(N)-B(N-2)
                DEFB  +38,fim-calc
                RET                   Final.
```

A função «Grandeza absoluta»
(Deslocamento 2A: «abs»)

Esta subrotina realiza a sua operação unária garantindo que o bit de sinal de um número em vírgula flutuante é passado a zero.

Os «inteiros pequenos» devem ser tratados separadamente. A maior parte do trabalho é partilhada com a operação «menos unário».

```
346A abs        LD    B,+FF           B passa a FF hex.
                JR    3474,NEG-TEST   Salto para «menos
                                      unário».
```

A operação «Menos unário»
(Deslocamento 1B: «negar»)

Esta subrotina realiza a sua operação unária alterando o sinal de «último valor» no «stack» do calculador.

O zero é devolvido sem alterações. Os números em vírgula flutuante com cinco bytes têm o seu sinal manipulado de tal modo que passa a zero (no caso de «abs» ou é alterado (no caso de «negar»). Os «inteiros pequenos» ficam com o byte de sinal em zero (no caso de «abs») ou alterado (no caso de «negar»).

```
346E NEGATE     CALL  34E9,TEST-ZERO  Se o número é zero, a
                RET   C               subrotina termina man-
                                      tendo 00 00 00 00 00.
                LD    B,+00           B passa a +00 hex para «negar».
```

«ABS» termina aqui.

```
3474 NEG-TEST   LD    A,(HL)          Se o primeiro byte é
                AND   A               zero, salto para tratar
                JR    Z,3483,INT-CASE «inteiro pequeno».
                INC   HL              Apontar para o 2.º byte.
                LD    A,B             Obter +FF para «abs»,
                                      +00 para «negar».
                AND   +80             Agora +80 para «abs»,
                                      +00 para «negar».
                OR    (HL)            Passa a um o bit 7 para «abs»,
                                      mas não altera para «negar».
                RLA                   Altera bit 7 (bit 7 de byte
                CCF                   2 a zero para «abs», e
                RRA                   alterado para «negar»).
                LD    (HL),A          Guarda o novo 2.º byte.
                DEC   HL              HL aponta de novo para
                                      o primeiro byte.
                RET                   Final.
```

No «caso inteiro» realiza-se uma operação semelhante para o byte de sinal.

```
3483 INT-CASE   PUSH  DE              Guarda STKEND em DE.
                PUSH  HL              Guarda indicador do número
                                      em HL.
                CALL  2D7F,INT-FETCH  Obtém o sinal em C, e o
                                      número em DE.
                POP   HL              Restaura o indicador do
                                      número em HL.
                LD    A,B             Obtém +FF para «abs»,
                                      +00 para «negar».
                OR    C               Agora +FF para «abs», sem
                                      alteração para «negar».
                CPL                   Agora +00 para «abs», e byte
                LD    C,A             alterado para «negar»; guarda em C.
                CALL  2D8E,INT-STORE  Guarda resultado no «stack».
                POP   DE              Devolve STKEND a DE.
                RET                   Final.
```

A função «SIGNUM»
(Deslocamento 29: «sgn»)

Esta subrotina trata a função SGN X, produzindo portanto um «último valor» de 1 se X é positivo, zero se X é zero, e -1 se X é negativo.

```
3492 sgn    CALL  34E9,TEST-ZERO   Se X é zero, termina com
            RET   C                zero como «último valor».
            PUSH  DE               Guardar o indicador em STKEND.
            LD    DE,+0001         Guardar 1 em DE.
            INC   HL               Apontar para o 2.º byte de X.
            RL    (HL)             Rodar o bit 7 para a «carry».
            DEC   HL               Apontar de novo para o destino.
            SBC   A,A              Passa C a zero para X positivo
            LD    C,A              e para FF hex se X negativo.
            CALL  2D8E,INT-STORE   Guarda 1 ou -1, conforme o caso.
            POP   DE               Restaura o indicador para
                                   STKEND.
            RET                    Final.
```

A função «IN»
(Deslocamento 2C: «in»)

Esta subrotina trata a função IN X. Aceita uma entrada no processador pelo porto X, carregando X em BC e executando a instrução IN A,(C).

```
34A5 in     CALL  1E99,FIND-INT2   O «último valor», X, é
                                   comprimido em BC.
            IN    A,(C)            O sinal é recebido.
            JR    34B0,IN-PK-STK   Salto para guardar resultado.
```

A função «PEEK»
(Deslocamento 2B: «peek»)

Esta subrotina trata a função PEEK X. O «último valor» é tirado do «stack» invocando FIND-INT2, e substituído pelo valor do conteúdo da posição requerida.

```
34AC peek   CALL  1E99,FIND-INT2   Avaliar «último valor», arre-
                                   dondado para inteiro + próximo;
                                   verificar se é aceitável
                                   e devolve-o em BC.
            LD    A,(BC)           Obter o byte requerido.
34B0 IN-PK-STK JP 2D28,STACK-A     Sair saltando para STACK-A.
```

A função «USR»
(Deslocamento 2D: «usr-n.º»)

Esta rotina («USR número», diferente de «USR cadeia») trata a função USR X, onde X é um número. O valor de X é obtido em BC, no «stack» encontra-se um endereço de retorno, e o código-máquina é executado a partir da posição X.

```
34B3 usr-no CALL  1E99,FIND-INT2   Avaliar o «último valor» arre-
                                   dondado para inteiro + próximo;
                                   verificar se é válido e de-
                                   volve-o em BC.
            LD    HL,+2D28         Obriga o endereço de retorno
            PUSH  HL               a ser o da subrotina STACK-BC.
            PUSH  BC               Executa salto indirecto para a
            RET                    posição adequada.
```


Nota: É interessante que o par de registos IY seja inicializado de novo após o retorno a STACK-BC, enquanto o importante H'L', que contém o indicador do literal seguinte, não é restaurado para o caso de se ter corrompido. Para um retorno com êxito ao Basic, H'L' deve, no final da rotina em código, conter o endereço da instrução «fim-calc» em SCANNING, 2758 hex (10072 decimal).

A função «USR cadeia»
(Deslocamento 19: «usr-$»)

Esta subrotina trata a função USR X$, onde X$ é uma cadeia. A subrotina devolve em BC o endereço do padrão de bits do gráfico definido pelo utilizador («udg»), correspondente a X$. Indica o erro A se X$ não é uma letra única entre «a» e «u», ou um gráfico definido pelo utilizador.

```
34BC usr-$  CALL  2BF1,STK-FETCH   Obter os parâmetros da
                                   cadeia X$.
            DEC   BC               Diminuir o comprimento
                                   de 1 para o verificar.
            LD    A,B              Se o comprimento não era 1,
            OR    C                salto para imprimir o erro A.
            JR    NZ,34E7,REPORT-A
            LD    A,(DE)           Obter o código de
                                   cadeia.
            CALL  2C8D,ALPHA       Indica uma letra?
            JR    C,34D3,USR-RANGE Se sim, salto para obter endereço.
            SUB   +90              Reduzir a gama dos gráficos
                                   definidos pelo utilizador a
                                   0-20 decimal.
            JR    C,34E7,REPORT-A  Dar mensagem A se fora da gama.
            CP    +15              Verificar de novo a gama.
            JR    NC,34E7,REPORT-A Dar mensagem A se fora da gama.
            INC   A                Passar gama de gráficos defi-
                                   nidos pelo utilizador para
                                   1-21 decimal («a» a «u»).
34D3 USR-RANGE DEC A               Passar a gama a 0-20 decimal
                                   em cada caso.
            ADD   A,A              Multiplicar por 8 para obter
            ADD   A,A              deslocamento do endereço.
            ADD   A,A
            CP    +A8              Verificar a gama do deslocamento.
            JR    NC,34E7,REPORT-A Dar mensagem A se fora da gama.
            LD    BC,(UDG)         Obter endereço do primeiro
                                   «udg» em BC.
            ADD   A,C              Somar C ao deslocamento.
            LD    C,A              Guardar o resultado em C.
            JR    NC,34E4,USR-STACK Saltar se não há «carry».
            INC   B                Incrementar B para completar
                                   o endereço.
            JP    2D28,STACK-BC    Saltar para guardar endereço.
```

Mensagem «A — Invalid argument»

```
34E7 REPORT-A RST  0008,ERROR-1    Chamar rotina de
              DEFB +09             tratamento do erro.
```

A subrotina «TEST-ZERO»

Esta subrotina é invocada pelo menos nove vezes a fim de verificar se um dado número em vírgula flutuante é zero. Este teste requer que os primeiros quatro bytes do número sejam zero. A subrotina termina com a flag «carry» em um se o número é de facto zero.

```
34E9 TEST-ZERO  PUSH  HL      Guardar HL no «stack».
                PUSH  BC      Guardar BC no «stack».
                LD    B,A     Guardar o valor de A em B.
                LD    A,(HL)  Obter o primeiro byte.
                INC   HL      Apontar para o 2.º byte.
                OR    (HL)    OR primeiro byte com segundo.
                INC   HL      Apontar para terceiro byte.
                OR    (HL)    OR o resultado com o ter-
                                ceiro byte.
                INC   HL      Apontar para quarto byte.
                OR    (HL)    OR o resultado com o
                                quarto byte.
                LD    A,B     Restaurar o valor original de A.
                POP   BC      E de BC.
                POP   HL      Restaurar o indicador do
                                número em HL.
                RET   NZ      Retorno com «carry» em zero
                                se qualquer dos 4 bytes não é 0.
                SCF           Passa «carry» a um para indicar
                RET             que o número era zero, e
                                retorno.
```

A operação «Maior do que zero»
(Deslocamento 37: «maior-0»)

Esta subrotina produz um «último valor» igual a um se o «último valor» actual é maior do que zero, e igual a zero se este não o for. É ainda usada por outras subrotinas para executarem saltos condicionados («salto se mais»).

```
34F9 GREATER-0  CALL  34E9,TEST-ZERO  O «último valor» é zero?
                RET   C               Se sim, retorno.
                LD    A,+FF           Salto adiante para MENOS DO
                JR    3507,SIGN-TO-C  QUE ZERO, mas sinalizar
                                        operação oposta.
```

A função «NOT»
(Deslocamento 30: «not»)

Esta subrotina produz um «último valor» igual a um se o «último valor» actual é zero, e zero noutro caso. É igualmente usada por outras subrotinas para saltos condicionados («salto se zero»).

```
3501 NOT        CALL  34E9,TEST-ZERO  A flag «carry» será um apenas
                                        se o «último valor» é zero;
                                        isto dá o resultado correcto.
                JR    3508,FP-0/1     Saltar para diante.
```


A operação «Menos do que zero»
(Deslocamento 36: «menos-0»)

Esta subrotina produz um «último valor» igual a um se o «último valor» é menos de zero, e a zero no caso contrário. É igualmente usada por outras subrotinas para «saltar se menos».

```
3506 menos-0    XOR   A       Limpar o registo A.
3507 SIGN-TO-C  INC   HL      Apontar para o byte de sinal.
                XOR   (HL)    A «carry» é zero para um n.º
                DEC   HL        positivo, e um para um n.º
                RLCA            negativo; quando acedida por
                                GREATER-0 o sinal oposto
                                passa à «carry».
```

A subrotina «Zero ou um»

Esta subrotina passa a zero o «último valor» se a flag «carry» está em zero e a um se esta flag é um. Quando invocada por «E-TO-FP», no entanto, cria o zero ou o um em mem-0 e não no «stack».

```
3508 FP-0/1     PUSH  HL      Guardar o indicador do resultado.
                LD    A,+00   Limpar A sem perturbar a
                                «carry».
                LD    (HL),A  Passar o 1.º byte a zero.
                INC   HL      Apontar para o 2.º byte.
                LD    (HL),A  Passar o 2.º byte a zero.
                INC   HL      Apontar para o 3.º byte.
                RLA           Rodar a «carry» para A, passando
                                esta a um se a «carry» era um,
                                e a zero se esta era zero.
                LD    (HL),A  Passar o 3.º byte para um e
                                para zero.
                RRA           Garantir que A seja zero.
                INC   HL      Apontar par o 4.º byte.
                LD    (HL),A  Passar o 4.º byte para zero.
                INC   HL      Apontar para o 5.º byte.
                LD    (HL),A  Passar o 5.º byte para zero.
                POP   HL      Restaurar o indicador de resultado.
                RET           Final.
```

A operação «OR»
(Deslocamento 07: «or»)

Esta subrotina realiza a operação binária «X ou Y», e devolve X se Y é zero e o valor 1, se assim não for.

```
351B or         EX    DE,HL   Apontar HL para Y, o 2.º
                                número.
                CALL  34E9,TEST-ZERO  Verificar se Y é zero.
                EX    DE,HL   Restaurar os indicadores.
                RET   C       Retorno se Y é zero; X é
                                agora o «último valor».
                SCF           Passar a um a flag «carry» e
                JR    3508,FP-0/1  saltar atrás para passar
                                «último valor» também para um.
```

A operação «Número e número»
(Deslocamento 08: «n.ºe-n.º»)

Esta subrotina realiza a operação binária «X AND Y», devolvendo X se Y não é zero, e zero no caso contrário.

```
3524 no-&-no    EX    DE,HL          Apontar HL para Y, DE para X.
                CALL  34E9,TEST-ZERO Verificar se Y é zero.
                EX    DE,HL          Trocar os indicadores.
                RET   NC             Retorno com X como «último
                                     valor» se Y não for zero.
                AND   A              Passa a zero a flag «carry»
                JR    3508,FP-0/1    e salta atrás para passar o
                                     «último valor» a zero.
```

A operação «Cadeia e número»
(Deslocamento 10: «cadeia-e-n.º»)

Esta subrotina realiza a operação binária «X$ AND Y» e devolve X$ se Y não for zero ou uma cadeia nula, no caso contrário.

```
352D cadeia-e-n.º  EX    DE,HL          Apontar HL para Y, DE para X$.
                   CALL  34E9,TEST-ZERO Verificar se Y é zero.
                   EX    DE,HL          Trocar de novo indicadores.
                   RET   NC             Retorno com X$ como «último
                                        valor» se Y não é zero.
                   PUSH  DE             Guardar o indicador do número.
                   DEC   DE             Apontar para o 5.º byte dos
                                        parâmetros da cadeia, ou seja,
                                        para o byte alto do comprimento.
                   XOR   A              Limpar o registo A.
                   LD    (DE),A         O byte alto de comp. passa a 0.
                   DEC   DE             Apontar para byte baixo.
                   LD    (DE),A         O byte baixo passa a zero.
                   POP   DE             Restaurar o indicador.
                   RET                  Retorno com «último valor»
                                        como parâmetro da cadeia.
```

As operações «Comparação»
(Deslocamentos 09 a 0E e 11 a 16: «n.ºmenor-igl», «n.ºmaior-igl», «n.ºs-nigl», «n.ºmaior», «n.ºmenor» e «cadeia-igl»).

Esta subrotina é usada para realizar as doze operações de comparação possíveis. O deslocamento da operação encontra-se no registo B no início da subrotina.

```
3538 n.ºmenor-igl  LD    A,B            O deslocamento passa para
     etc.                               o registo A.
                   SUB   +08            A gama é agora 01-06 e
                                        09-0E.
                   BIT   2,A            Esta gama é mudada para
                   JR    NZ,3543,EX-OR-NOT 00-02, 04-06, 08-0A e
                   DEC   A              0C-0E.
```



```
3543 EX-OR-NOT  RRCA                   Depois reduzida para 00-07
                                       com a «carry» a um se «maior
                                       ou igual a» e «menor que»; as
                                       operações com a «carry» a um
                JR    NC,354E,NU-OR-STR são então tratadas como suas
                PUSH  AF               operações complementares depois
                PUSH  HL               de os valores serem trocados.
                CALL  343C,EXCHANGE
                POP   DE
                EX    DE,HL
                POP   AF
354E NU-OR-STR  BIT   2,A              As comparações numéricas são
                JR    NZ,3559,STRINGS  agora separadas das de cadeia
                                       testando o bit 2.
                RRCA                   As operações numéricas têm
                                       a gama 00-01 com a «carry»
                                       a um para «igual» e «não igual».
                PUSH  AF               Guardar o deslocamento.
                CALL  300F,SUBTRACT    Os números são subtraídos para
                JR    358C,END-TESTS   os testes finais.
3559 STRINGS    RRCA                   As comparações de cadeias têm
                                       agora a gama 02-03 com «carry»
                                       a um para «igual» e «não igual».
                PUSH  AF               Guardar o deslocamento.
                CALL  28F1,STK-FETCH   Os comprimentos e endereços
                PUSH  DE               iniciais das cadeias são
                PUSH  BC               recuperados no «stack» do
                CALL  28F1,STK-FETCH   calculador.
                POP   HL               Comprimento da segunda cadeia.
3564 BYTE-COMP  LD    A,H
                OR    L
                EX    (SP),HL
                LD    A,B
                JR    NZ,3575,SEC-PLUS Salta a menos que a segunda
                OR    C                cadeia seja vazia.
3568 SECND-LOW  POP   BC               Aqui a segunda cadeia é vazia
                                       ou menos do que a primeira.
                JR    Z,3572,BOTH-NULL
                POP   AF
                CCF                    A «carry» é complementada para
                JR    3588,STR-TEST    garantir testes correctos.
3572 BOTH-NULL  POP   AF               A «carry» é usada tal como
                JR    3588,STR-TEST    está.
3575 SEC-PLUS   OR    C
                JR    Z,3585,FRST-LESS A 1.ª cadeia é nula, a segunda
                                       não.
                LD    A,(DE)           Nenhuma cadeia é nula, sendo
                SUB   (HL)             comparados os bytes seguintes.
                JR    C,3585,FRST-LESS O primeiro byte é menos.
                JR    NZ,3568,SECND-LOW O segundo byte é menos.
                DEC   BC               Os bytes são iguais; os
                INC   DE               comprimentos são decrementados
                INC   HL               e é feito um salto para BYTE-
                EX    (SP),HL          -COMP a fim de comparar os
                DEC   HL               bytes seguintes das cadeias.
                JR    3564,BYTE-COMP
3585 FRST-LESS  POP   BC
                POP   AF
                AND   A                A «carry» é limpa aqui para
                                       garantir testes correctos.
```

```
3588 STR-TEST   PUSH  AF              Para os testes de cadeias, põe
                RST   0028,FP-CALC    um zero no «stack» do calculador.
                DEFB  +A0,stk-zero
                DEFB  +38,fim-calc
358C END-TESTS  POP   AF              Estes três testes, feitos quando
                PUSH  AF              necessário, dão os resultados
                CALL  C,3501,NOT      correctos para as 12 comparações.
                POP   AF              A «carry» inicial é um para «não
                PUSH  AF              igual» e «igual», e a «carry»
                CALL  NC,34F9,GREATER-0 final é um para «maior que»,
                POP   AF              «menor que» e «igual».
                RRCA
                CALL  NC,3501,NOT
                RET                   Final.
```

A operação «Concatenação de cadeias»
(Deslocamento 17: «cadeias-soma»)

Esta subrotina realiza a operação binária A$+B$. Os parâmetros destas cadeias são recuperados, determinando-se o comprimento total. É reservado espaço suficiiente na área de trabalho para guardar ambas as cadeias, sendo estas copiadas em seguida. O resultado desta subrotina consiste, portanto, em produzir uma variável temporária A$+B$ que reside na área de trabalho.

```
359C cadeias-soma  CALL  2BF1,STK-FETCH   Obtêm-se e guardam-se os
                   PUSH  DE               parâmetros da 2.ª cadeia.
                   PUSH  BC
                   CALL  2BF1,STK-FETCH   Obtêm-se os parâmetros da
                                          primeira cadeia.
                   POP   HL
                   PUSH  HL               Os comprimentos estão agora
                                          em HL e BC.
                   PUSH  DE               São guardados os parâmetros
                   PUSH  BC               da primeira cadeia.
                   ADD   HL,BC            O comprimento total das 2
                   LD    B,H              cadeias é calculado e
                   LD    C,L              passado a BC.
                   RST   0030,BC-SPACES   É reservado o espaço su-
                                          ficiente.
                   CALL  2AB2,STK-STORE   Os parâmetros da nova cadeia
                                          são passados para o «stack»
                                          do calculador.
                   POP   BC               São obtidos os parâmetros da
                   POP   HL               1.ª cadeia, sendo esta copia-
                   LD    A,B              da para a área de trabalho
                   OR    C                se não é nula.
                   JR    Z,3587,OTHER-STR
                   LDIR
3587 OTHER-STR     POP   BC               Segue-se o mesmo método
                   POP   HL               para a 2.ª cadeia, dando
                   LD    A,B              assim A$+B$.
                   OR    C
                   JR    Z,35BF,STK-PNTRS
                   LDIR
```


A subrotina «STK-PNTRS»

Esta subrotina passa o par de registos HL a zero a fim de apontar para o primeiro byte do «último valor», isto é, STKEND-5, e o par de registos DE aponta para um depois do «último valor», isto é, STKEND.

```
35BF STK-PNTRS  LD    HL,(STKEND)     Obter o valor actual de
                                      STKEND.
                LD    DE,+FFFB        Passa DE a -5, complemento para 2.
                PUSH  HL              Guarda o valor de STKEND.
                ADD   HL,DE           Calcula STKEND-5.
                POP   DE              DE contém STKEND, e HL
                RET                   contém STKEND-5.
```

A função «CHR$»
(Deslocamento 2F: «chrs»)

Esta subrotina trata a função CHR$ X, e cria uma cadeia de caracteres na área de trabalho.

```
35C9 chrs       CALL  2DD5,FP-TO-A    O «último valor» é comprimido
                                      no registo A.
                JR    C,35DC,REPORT-B Dar mensagem de erro se X for
                JR    NZ,35DC,REPORT-B maior que 255 decimal ou X
                PUSH  AF              for um número negativo.
                LD    BC,+0001        Guardar o valor comprimido de X.
                RST   0030,BC-SPACES  Deixar um espaço disponível na
                POP   AF              área de trabalho.
                LD    (DE),A          Obter o valor.
                                      Copiar o valor para a área
                                      de trabalho.
                CALL  2AB2,STK-STORE  Passar os parâmetros da nova
                                      cadeia para o «stack» do calculador.
                EX    DE,HL           Passar indicadores a zero.
                RET                   Final.
```

Mensagem «B — Integer out of range»

```
35DC REPORT-B   RST   0008,ERROR-1    Invocar rotina de
                DEFB  +0A             tratamento de erro.
```

A função «VAL» e «VAL$»
(Deslocamentos 1D: «val» e 18: «val$»)

Esta subrotina trata as funções VAL X$ e VAL$ X$. Quando trata VAL X$, devolve um «último valor» que é o resultado da avaliação da cadeia (sem as suas aspas) como expressão numérica. Quando trata VAL$ X$, avalia X$ (sem as suas aspas) como expressão de cadeia, e devolve os parâmetros dessa expressão em cadeia como «último valor» no «stack» do calculador.

```
35DE val        LD    HL,(CH-ADD)     É preservado o valor
   (também val$)PUSH  HL              da CH-ADD no «stack»-máquina.
                LD    A,B             O «deslocamento» de «val» ou
                                      «val$» deve estar no registo
                                      B; é agora copiado para A.
                ADD   A,+E3           Produz +00 e passa a 1 a
```

			«carry» para «val», +FB e a 0 para «val$».
	SBC	A,A	Produz +FF (bit 6 a um) para «val», mas +00 (bit 6 a zero) para «val$».
	PUSH	AF	Guardar esta «flag» no «stack»-máquina.
	CALL	28F1,STK-FETCH	Os parâmetros da cadeia são
	PUSH	DE	obtidos; guarda-se o endereço
	INC	BC	inicial; soma-se um byte ao
	RST	0030,BC-SPACES	comprimento e reserva-se espaço para a cadeia (+1) na área de trabalho.
	POP	HL	O endereço inicial da cadeia passa a HL como endereço fonte.
	LD	(CH-ADD),DE	O indicador do primeiro es-
	PUSH	DE	paço novo vai para CH-ADD e
			para o «stack»-máquina.
	LDIR		A cadeia é copiada para a área de trabalho, juntamente com um byte extra.
	EX	DE,HL	Comutar os indicadores.
	DEC	HL	O byte extra é substituído
	LD	(HL),+0D	por um caracter «retorno de linha».
	RES	7,(FLAGS)	A flag sintaxe passa a 0 e
	CALL	24FB,SCANNING	a sintaxe da cadeia é verificada.
	RST	0018,GET-CHAR	Obtém o caracter após a cadeia.
	CP	+0D	Verifica se foi atingido o final da expressão.
	JR	NZ,360C,V-RPORT-C	Se não, indica erro.
	POP	HL	Obtém endereço inicial da cadeia.
	POP	AF	A «flag» de «val/val$» é
	XOR	(FLAGS)	obtida, comparando-se com o
	AND	+40	bit 6 do resultado da verificação de sintaxe.
360C V-RPORT-C	JP	NZ,1C8A,REPORT-C	Indicar erro se não concordarem.
	LD	(CH-ADD),HL	De novo, endereço inicial em CH-ADD.
	SET	7,(FLAGS)	A flag passa a um se se executa a linha.
	CALL	24FB,SCANNING	A cadeia é tratada como «expressão seguinte» e produz-se um «último valor».
	POP	HL	O valor original de CH-ADD
	LD	(CH-ADD),HL	é restaurado.
	JR	35BF,STK-PNTRS	A subrotina termina através de STK-PNTRS, que passa os indicadores a zero.

A função «STR$»
(Deslocamento 2E: «str$»)

Esta subrotina trata a função STR$ X e devolve um «último valor» que é um conjunto de parâmetros que definem uma cadeia contendo o que surgiria no visor se X fosse impresso por uma ordem PRINT.

361F str$	LD	BC,+0001	É reservado 1 espaço na área
	RST	0030,BC-SPACES	de trabalho e o seu endereço
	LD	(K-CUR),HL	é copiado para K-CUR, endereço do cursor.
	PUSH	HL	Este endereço é guardado também no «stack».
	LD	HL,(CURCHL)	O endereço do canal actual
	PUSH	HL	é guardado no «stack»-máquina.
	LD	A,+FF	Abre o canal «f», permitindo
	CALL	1601,CHAN-OPEN	a «impressão» da cadeia na área de trabalho.
	CALL	2DE3,PRINT-FP	O «último valor», X, é agora impresso na área de trabalho e esta é aumentada para cada caracter.
	POP	HL	Restaurar CURCHL em HL e res-
	CALL	1615,CHAN-FLAG	taurar as flags que lhe correspondem.
	POP	DE	Restaurar o endereço inicial da cadeia.
	LD	HL,(K-CUR)	O endereço do cursor é agora
	AND	A	um após o final da cadeia, e
	SBC	HL,DE	portanto a diferença equivale ao comprimento.
	LD	B,H	Transferir o comprimento para BC.
	LD	C,L	
	CALL	2AB2,STK-STO-$	Passar os parâmetros da nova cadeia para o «stack» do calculador.
	EX	DE,HL	Passar os indicadores a zero.
	RET		Final.

Nota: Ver PRINT-FP para uma explicação do erro «PRINT 'A'+STR$ 0.1».

A subrotina «READ-IN»
(Deslocamento 1A: «ler»)

Esta subrotina é invocada tendo em conta o deslocamento do calculador através da primeira linha da rotina S-INKEY$ em SCANNING. Parece servir para leitura de dados por «streams» diferentes dos existentes no Spectrum «standard». Tal como INKEY$, a subrotina devolve uma cadeia.

3645 ler	CALL	1E94,FIND-INT1	O parâmetro numérico é comprimido no registo A.
	CP	+10	Menor do que 16 decimal?
	JP	NC,1E9F,REPORT-B	Se não, indicar erro.
	LD	HL,(CURCHL)	O endereço do canal actual
	PUSH	HL	é guardado no «stack»-máquina.
	CALL	1601,CHAN-OPEN	É aberto o canal especificado pelo parâmetro.
	CALL	15E6,INPUT-AD	O sinal é aceite agora, como um «valor-chave».
	LD	BC,+0000	O comprimento à partida da cadeia resultante é zero.
	JR	NC,365F,R-I-STORE	Saltar, se não existe sinal.
	INC	C	Passar o comprimento para 1.
	RST	0030,BC-SPACES	Reservar espaço na área de trabalho.
	LD	(DE),A	Colocar a cadeia nele.

```
365F R-I-STORE   CALL  2AB2,STK-STO-$   Passar os parâmetros da cadeia
                                        no «stack» do calculador.
                 POP   HL               Restaurar CURCHL e as
                 CALL  1615,CHAN-FLAG   flags apropriadas.
                 JP    35BF,STK-PNTRS   Sair, definindo os indicadores.
```

A função «CODE»
(Deslocamento 1C: «code»)

Esta subrotina trata a função CODE A$ e devolve o código Spectrum do primeiro caracter em A$, ou zero se A$ deve ser nula.

```
3669 code        CALL  2BF1,STK-FETCH   Obtém os parâmetros da
                                        cadeia.
                 LD    A,B              Verifica o comprimento e o
                 OR    C                registo A, com zero, é mantido,
                 JR    Z,3671,STK-CODE  dado que A$ é uma cadeia nula.
                 LD    A,(DE)           O código do 1.º caracter é
                                        posto em A no caso contrário.
3671 STK-CODE    JP    2D28,STACK-A     A subrotina sai por STACK-A,
                                        o que dá o «último valor»
                                        correcto.
```

A função «LEN»
(Deslocamento 1E: «len»)

Esta subrotina trata a função LEN A$ e produz um «último valor» que é igual ao comprimento da cadeia.

```
3674 len         CALL  2BF1,STK-FETCH   Obtém os parâmetros da
                                        cadeia.
                 JP    2D2B,STACK-BC    A subrotina sai através
                                        de STACK-BC, dando o «últi-
                                        mo valor» correcto.
```

A subrotina «Diminuir o contador»
(Deslocamento 35: «dec-jr-nz»)

Esta subrotina é apenas invocada pelo GERADOR DE SÉRIES, e de facto, é uma operação «DJNZ», mas o contador é a variável de sistema BREG, e não o registo B.

```
367A dec-jr-nz   EXX                    Passa aos registos alter-
                 PUSH  HL               nativos e guarda o indicador do
                                        literal seguinte no «stack»-
                                        -máquina.
                 LD    HL,+5C67         Fazer HL apontar para BREG.
                 DEC   (HL)             Diminuir BREG.
                 POP   HL               Restaurar indicador do literal seguinte.
                 JR    NZ,3687,JUMP-2   Salto se não-zero.
                 INC   HL               Passa o literal seguinte.
                 EXX                    Retorno aos registos principais.
                 RET                    Final.
```


A subrotina «Salto»
(Deslocamento 33: «jump»)

Esta subrotina executa um salto incondicional quando invocada pelo literal «33». É também usada pelas subrotinas DIMINUIR O CONTADOR e SALTAR SE VERDADE.

```
3686 JUMP        EXX                    Passar aos registos alternativos.
3687 JUMP-2      LD    E,(HL)           O literal seguinte (comprimento
                                        do salto) passa para E'.
                 LD    A,E              Forma em A o número 00 hex
                 RLA                    ou FF hex conforme E' é
                 SBC   A,A              positivo ou negativo,
                 LD    D,A              copiando-o depois para D'.
                 ADD   HL,DE            Os registos H' e L' contêm agora
                 EXX                    o indicador do literal seguinte.
                 RET                    Final.
```

A subrotina «SALTAR SE VERDADEIRO»
(Deslocamento 00: «salto-verdade»)

Esta subrotina executa um salto condicional se o «último valor» no «stack» do calculador, ou mais precisamente, o número actualmente endereçado pelo par de registos DE, é verdadeiro.

```
368F salto-verdade  INC   DE           Apontar para o 3º byte,
                    INC   DE           que é zero ou um.
                    LD    A,(DE)       Recolher este byte no
                                       registo A.
                    DEC   DE           Apontar novamente para
                    DEC   DE           o 1.º byte.
                    AND   A            Verificar o 3º byte: é 0?
                    JR    NZ,3686,JUMP Saltar se não é, ou seja,
                                       se o número não é
                                       falso.
                    EXX                Passar aos registos alternativos.
                    INC   HL           Passar o comprimento do salto.
                    EXX                Voltar aos registos
                                       principais.
                    RET                Final.
```

A subrotina «FIM-CALC»
(Deslocamento 38: «fim-calc»)

Esta subrotina termina uma operação RST 0028.

```
3698 fim-calc    POP   AF               Elimina o endereço de
                                        retorno do calculador
                                        («RE-ENTRY»).
                 EXX                    Coloca no «stack»-máquina
                 EX    (SP),HL          o endereço em H'L', executando
                 EXX                    um salto indirecto para ele.
                                        H'L' conterá agora um endereço
                                        anterior da cadeia de endereços
                                        do calculador.
                 RET                    Final.
```

A subrotina «Módulo»
(Deslocamento 32: «n-mod-m»)

Esta subrotina calcula|MI(mod M), onde M é um inteiro positivo guardado no topo do «stack» do calculador, o «último valor», e N é um inteiro guardado no «stack» abaixo de M.

A subrotina produz o quociente inteiro INT (N/M), colocado no topo do «stack» do calculador como «último valor», e o resto N-INT (N/M) na segunda posição do «stack».

Esta subrotina é invocada durante o cálculo de um número aleatório para reduzir N mod 65537 decimal.

```
36A0 n-mod-m    RST    0028,FP-CALC          N,M
                DEFB   +C0,st-mem-0          N,M          mem-0 contém M
                DEFB   +02,apagar            N
                DEFB   +31,copiar            N, N
                DEFB   +E0,obter-mem-0       N, N, M
                DEFB   +05,dividir           N, N/M
                DEFB   +27,int               N, INT (N/M)
                DEFB   +E0,obter-mem-0       N, INT (N/M),M
                DEFB   +01,trocar            N, M, INT (N/M)
                DEFB   +C0,st-mem-0          N, M, INT (N/M) mem-0 contém
                                             INT (N/M)
                DEFB   +04,multiplicar       N, M*INT (N/M)
                DEFB   +03,subtrair          n-M*INT (N/M)
                DEFB   +E0,obter-mem-0       n-M*INT (N/M), INT (N/M)
                DEFB   +38,fim-calc
                RET                          Final.
```

A função «INT»
(Deslocamento 27: «int»)

Esta subrotina trata a função INT X e produz um «último valor» que é a «parte inteira» do valor fornecido. Assim, INT 2.4 produz 2 dado que a subrotina arredonda sempre o resultado para menos; INT -2.4 dá -3.

A subrotina utiliza a subrotina de TRUNCATURA INTEIRA PARA ZERO, em 3214, para produzir I(X) de tal modo que I(2.4) dá 2 e I(-2.4) dá -2. Assim, INT X é dado por I(X) para valores de X que sejam maiores ou iguais a zero, e por I(X)-1 para valores negativos de X que não sejam já inteiros, quando o resultado seria evidentemente I(X).

```
36AF int        RST    0028,FP-CALC          X
                DEFB   +31,copiar            X, X
                DEFB   +36,menor-0           X, (1/0)
                DEFB   +00,salto-verdade     X
                DEFB   +04,para 36B7,X-NEG.  X
```

Para valores de X que se tenha verificado serem maiores ou iguais a zero não ocorre salto, sendo I(X) determinado imediatamente.

```
                DEFB   +3A,truncar           I (X)
                DEFB   +38,fim-calc
                RET                          Final.
```

Quando X é um inteiro negativo produz I(X), senão, produz I(X)-1.


```
36B7 X-NEG      DEFB   +31,copiar            X, X
                DEFB   +3A,truncar           X, I (X)
                DEFB   +C0,st-mem-0          X, I (X)      mem-0 contém I(X)
                DEFB   +03,subtrair          X-I (X)
                DEFB   +E0,obter-mem-0       X-I (X), I (X)
                DEFB   +01,trocar            I (X), X-I (X)
                DEFB   +30,negar             I (X), (1/0)
                DEFB   +00,saltar-verdade    I (X)
                DEFB   +03,para 36C2, EXIT   I (X)
```

O salto é executado para valores de X que sejam inteiros negativos, senão, não ocorre salto e é calculado I(X)-1.

```
                DEFB   +A1,stk-um            I (X), 1
                DEFB   +03,subtrair          I (X)-1
```

Em qualquer caso, a subrotina termina com:

```
36C2 EXIT       DEFB   +38,fim-calc          I (X) ou I (X)-1
                RET
```

A função «Exponencial»
(Deslocamento 26: «exp»)

Esta subrotina trata a função EXP X, e é a primeira de quatro rotinas que utilizam o GERADOR DE SÉRIES para produzir polinómios de Chebyshev.

O valor aproximado de EXP X é determinado do seguinte modo:

1. X é dividido por LN 2 de modo a dar Y, pelo que o resultado agora pretendido é 2 elevado a Y.
2. É determinado o valor N, tal que N=INT Y.
3. É determinado o valor W, tal que W=Y–N, onde 0<=W<=1, como é necessário para que a série seja convergente.
4. É formado o argumento Z, tal que $Z=2 \cdot W-1$.
5. O GERADOR DE SÉRIES é usado para produzir 2↑W.
6. Finalmente, soma-se N ao expoente, dando 2↑(N+W), que é 2↑Y e, portanto, a resposta requerida a EXP X.

O método é ilustrado usando um programa Basic no Apêndice.

```
36C4 EXP        RST    0028,FP-CALC          X
```

Executar o passo 1.

```
                DEFB   +3D,re-stack          X (em vírgula flutuante)
                DEFB   +34,stk-dado          X, 1/LN 2
                DEFB   +F1,expoente+81
                DEFB   +38,+AA,+38,+29
                DEFB   +04,multiplicar       X/LN 2 = Y
```

Executar o passo 2.

```
DEFB  +31,copiar          Y, Y
DEFB  +27,int,1C46        Y, INT Y = N
DEFB  +C3,st-mem-3        Y, N          mem-3 contém N.
```

Executar o passo 3.

```
DEFB  +03,subtrair        Y-N = W
```

Executar o passo 4.

```
DEFB  +31,copiar          W, W
DEFB  +0F,somar           2*W
DEFB  +A1,stk-um          2*W, 1
DEFB  +03,subtrair        2*W-1 = Z
```

Executar o passo 5, passando ao GERADOR DE SÉRIES o parâmetro «8» e as oito constantes requeridas.

```
      DEFB  +88,série-08          Z
1.    DEFB  +13,expoente+63
      DEFB  +36,(+00,+00,+00)
2.    DEFB  +58,expoente+68
      DEFB  +65,+66,(+00,+00)
3.    DEFB  +9D,expoente+6D
      DEFB  +78,+65,+40,(+00)
4.    DEFB  +A2,expoente+72
      DEFB  +60,+32,+C9,(+00)
5.    DEFB  +E7,expoente+77
      DEFB  +21,+F7,+AF,+24
6.    DEFB  +EB,expoente+7B
      DEFB  +2F,+B0,+B0,+14
7.    DEFB  +EE,expoente+7E
      DEFB  +7E,+BB,+94,+58
8.    DEFB  +F1,expoente+81
      DEFB  +3A,+7E,+F8,+CF
```

No final do último ciclo o «último valor» é 21W.
Executar o passo 6.

```
DEFB  +E3,obter-mem-3     2**W,N
DEFB  +38,fim-calc
CALL  2DD5,FP-TO-A        O valor absoluto de N mod
                          256 decimal é passado ao
                          registo A.
JR    NZ,3705,N-NEGTV     Saltar para diante se N<0.
JR    C,3703,REPORT6      Erro se ABS N maior do que
                          255 decimal.
ADD   A,(HL)              Somar agora ABS N ao ex-
                          poente.
JR    NC,370C,RESULT-OK   Saltar a menos que e maior
                          do que 255 decimal.
```


Mensagem «6 — Number too big»

```
3703  REPORT-6    RST   0008,ERROR-1      Invocar rotina de
                  DEFB  +05                tratamento de erro.
3705  N-NEGTV     JR    C,370E,RSLT-ZERO   O resultado será zero se N é
                                           menos de -255 decimal.
                  SUB   (HL)               Subtrair ABS N do expoente,
                                           porque N era negativo.
                  JR    NC,370E,RSLT-ZERO  Resultado 0 se e<0.
                  NEG                      Menos e passa a e.
370C  RESULT-OK   LD    (HL),A             Insere o expoente e.
                  RET                      Final: o «último valor» é EXP X.
370E  RSLT-ZERO   RST   0028,FP-CALC       Usar o calculador para
                  DEFB  +02,apagar         passar «último valor» a zero.
                  DEFB  +A0,stk-zero
                  DEFB  +38,fim-calc
                  RET                      Final, com EXP X=0.
```

A função «Logaritmo natural»
(Deslocamento 25: «ln»)

Esta subrotina trata a função LN X, e é a segunda das quatro rotinas que utilizam o GERADOR DE SÉRIES para produzir polinómios de Chebyshev.
O valor aproximado de LN X é obtido do seguinte modo:

1. Verifica-se X, dando a mensagem A se X não for positivo.
2. X é então dividido no seu expoente verdadeiro, e', e na sua mantissa X'=X/(2<e'), onde X' é maior ou igual a 0.5, mas ainda inferior a um.
3. Forma-se o valor pretendido de Y1 ou Y2. Se X' é maior do que 0.8, Y1 = e'•LN 2; no caso contrário, Y2 = (e'-1)•LN 2.
4. Se X' é maior do que 0.8, é guardada a quantidade X'-1; se não guarda-se 2•X'-1.
5. Forma-se agora o argumento Z, que será Z = 2.5•X'-3 se X' é maior do que 0.8; senão, Z=5•X'-3. De qualquer modo, -1<=Z<=1, o que é necessário para que a série seja convergente.
6. Usa-se o GERADOR DE SÉRIES para produzir a função requerida.
7. Finalmente, uma multiplicação e uma adição simples levam a devolver LN X como «último valor».

```
3713  ln          RST   0028,FP-CALC       X
```

Executar o passo 1.

```
DEFB  +3D,re-«stack»      X (em vírgula flutuante).
DEFB  +31,copiar          X, X
DEFB  +37,maior-0         X, (1/0)
DEFB  +00,saltar-verdade  X
DEFB  +04,para 371C, VALID X
DEFB  +38,fim-calc        X
```

Mensagem «A — Invalid argument»

```
371A REPORT-A   RST   0008,ERROR-1      Invocar rotina de
                DEFB  +09               tratamento de erro.
```

Executar o passo 2.

```
371C VALID      DEFB  +A0,stk-zero      X,0          O 1 eliminado é
                DEFB  +02,apagar        X            substituído por 0.
                DEFB  +38,fim-calc      X
                LD    A,(HL)            O expoente, e, vai para A.
                LD    (HL),+80          X é reduzido a X'.
                CALL  2D28,STACK-A      O «stack» contém: X', e.
                RST   0028,FP-CALC      X'e
                DEFB  +34,stk-dado      X'e, 128 (decimal)
                DEFB  +38,expoente+88
                DEFB  +00,(+00,+00,+00)
                DEFB  +03,subtrair      X', e'
```

Executar passo 3.

```
                DEFB  +01,trocar        e', X'
                DEFB  +31,copiar        e', X', X'
                DEFB  +34,stk-dado      e', X', X',0.8(decimal)
                DEFB  +F0,expoente+80
                DEFB  +4C,+CC,+CC,+CD
                DEFB  +03,subtrair      e', X', X'-0.8
                DEFB  +37,maior-0       e', X', (1/0)
                DEFB  +00,saltar verdade e', X'
                DEFB  +08,para 373D,GRE.8 e', X'
                DEFB  +01,trocar        X', e'
                DEFB  +A1,stk-um        X', e', 1
                DEFB  +03,subtrair      X', e'-1
                DEFB  +01,trocar        e'-1, X'
                DEFB  +38,fim-calc      e'-1,X'
                INC   (HL)              Duplicar X', obtendo 2•X'
                RST   0028,FP-CALC      e'-1, 2•X
373D GRE.8      DEFB  +01,trocar        X', é          – X' grande.
                                        2*X',e'-1      – X' pequeno.
                DEFB  +34,stk-dado      X',e',LN 2
                DEFB  +F0,expoente+80   2*X',e'-1, LN 2
                DEFB  +31,+72,+17,+F8
                DEFB  +04,multiplicar.  X',e'*LN 2 = Y1
                                        2*X', (e'-1)*LN 2 = Y2
```

Executar o passo 4.

```
                DEFB  +01,trocar        Y1, X'         – X' grande.
                                        Y2, 2*X'       – X' pequeno.
                DEFB  +A2,stk-meio      Y1, X',.5 (decimal)
                                        Y2, 2*X', .5
                DEFB  +03,subtrair      Y1, X'-.5
                                        Y2, 2*X'-.5
                DEFB  +A2,stk-meio      Y1, X'-.5, .5
                                        Y2, 2*X'-.5, .5
                DEFB  +03,subtrair      Y1, X'-1
                                        Y2,2*X'-1
```


Executar o passo 5.

```
                DEFB  +31,copiar        Y, X'-1, X'-1
                                        Y2, 2*X'-1, 2*X'-1
                DEFB  +34,stk-dado      Y1, X'-1, X'-1, 2.5 (decimal)
                                        Y2, 2*X'-1, 2*X'-1, 2.5
                DEFB  +32,expoente+82
                DEFB  +20,(+00,+00,+00)
                DEFB  +04,multiplicar   Y1, X'-1, 2.5*X'-2.5
                                        Y2,2*X'-1, 5*X'-2.5
                DEFB  +A2,stk-meio      Y1, X'-1, 2.5*X-2.5, .5
                                        Y2, 2*X'-1, 5*X'-2.5, .5
                DEFB  +03,subtrair      Y1, X'-1,2.5*X'-3 = Z
                                        Y2, 2*X'-1, 5*X'-3 = Z
```

Executar o passo 6, passando ao GERADOR DE SÉRIES o parâmetro «12» decimal, e as doze constantes requeridas.

```
                DEFB  +8C,série-0C      Y1, X'-1, Z ou Y2, 2*X'-1, Z
1.              DEFB  +11,expoente+61
                DEFB  +AC,(+00,+00,+00)
2.              DEFB  +14,expoente+64
                DEFB  +09,(+00,+00,+00)
3.              DEFB  +56,expoente+66
                DEFB  +DA,+A5,(+00,+00)
4.              DEFB  +59,expoente+69
                DEFB  +30,+C5,(+00,+00)
5.              DEFB  +5C,expoente+6C
                DEFB  +90,+AA,(+00,+00)
6.              DEFB  +9E,expoente+6E
                DEFB  +70,+6F,+61,(+00)
7.              DEFB  +A1,expoente+71
                DEFB  +CB,+DA,+96,(+00)
8.              DEFB  +A4,expoente+74
                DEFB  +31,+9F,+B4,(+00)
9.              DEFB  +E7,expoente+77
                DEFB  +A0,+FE,+5C,+FC
10.             DEFB  +EA,expoente+7A
                DEFB  +1B,+43,+CA,+36
11.             DEFB  +ED,expoente+7D
                DEFB  +A7,+9C,+7E,+5E
12.             DEFB  +F0,expoente+80
                DEFB  +6E,+23,+80,+93
```

No final do último ciclo, o «último valor» é:

```
                LD X'/(X'-1)            para os valores maiores de X'
ou
                LD (2•X')/(2•X'-1)      para os valores menores de X'.
```

Executar o passo 7.

```
                DEFB  +04,multiplicar   Y1=LN (2**e'), LN X'
                                        Y2=LN (2**(e'-1)), LN (2*X')
                DEFB  +0F,somar         LD (2**e')*X')          = LNX
                                        LN(2**(e'-1)*2*X')      = LN X
                DEFB  +38,fim-calc      LN X
                RET                     Final: «último valor» é LN X.
```

A subrotina «Reduzir argumento»
(Deslocamento 39: «obter-argt»)

Esta subrotina transforma o argumento X de SIN X ou COS X num valor V.
A subrotina começa por determinar um valor Y tal que:

$$Y = X/(2 \cdot PI) - INT\ (X/2 \cdot PI) + 0.5),$$

onde Y é maior ou igual a -.5, mas menos do que +.5.
A subrotina produz:

$$V = 4 \cdot Y \text{ se } -1 <= 4 \cdot Y <= 1 \qquad \text{(caso 1)}$$

ou

$$V = 2 - 4 \cdot Y \text{ se } 1 < 4 \cdot Y < 2 \qquad \text{(caso 2)}$$

ou

$$V = -4 \cdot Y - 2 \text{ se } -2 <= 4 \cdot Y < -1 \qquad \text{(caso 3)}$$

Em qualquer dos casos, $-1 <= V <= 1$ e $SIN\ (PI \cdot V/2) = SIN\ X$.

```
3783 obter-argt   RST   0028,FP-CALC          X
                  DEFB  +3D,re-stack          X (em vírgula flutuante)
                  DEFB  +34,stk-dado          X, 1/(2*PI)
                  DEFB  +EE,expoente+7E
                  DEFB  +22,+F9,+83,+6E
                  DEFB  +04,multiplicar       X/(2*PI)
                  DEFB  +31,copiar            X/(2*PI), X/(2*PI)
                  DEFB  +A2,stk-meio          X/(2*PI), X/(2*PI), 0.5
                  DEFB  +0F,somar             X/(2*PI), X/(2*PI)+0.5
                  DEFB  +27,int,1C46          X/(2*PI), INT (X/(2*PI)+0.5)
                  DEFB  +03,subtrair,174C     X/(2*PI)-INT (X/(2*PI)+0.5)=Y
```

Nota: Somando 0.5 e realizando INT arredonda-se para o inteiro mais próximo.

```
                  DEFB  +31,copiar            Y, Y
                  DEFB  +0F,somar             2*Y
                  DEFB  +31,copiar            2*Y, 2*Y
                  DEFB  +0F,somar             4*Y
                  DEFB  +31,copiar            4*Y, 4*Y
                  DEFB  +2A,abs               4*Y, ABS (4*Y)
                  DEFB  +A1,stk-um            4*Y, ABS (4*Y), 1
                  DEFB  +03,subtrair          4*Y, ABS (4*Y)-1 = Z
                  DEFB  +31,copiar            4*Y, Z, Z
                  DEFB  +37,maior-0           4*Y, Z, (1/0)
                  DEFB  +C0,st-mem-0          Mem-0 contém o resultado
                                              do teste.
                  DEFB  +00,saltar-verdade    4*Y, Z
                  DEFB  +04,para 37A1,ZPLUS   4*Y, Z
                  DEFB  +02,apagar            4*Y
                  DEFB  +38,fim-calc          4*Y = V - caso 1.
                  RET                         Final.
```

Se foi executado o salto, continuar.

```
37A1 ZPLUS        DEFB  +A1,stk-um            4*Y, Z, 1
                  DEFB  +03,subtrair          4*Y, Z-1
                  DEFB  +01,trocar            Z-1,4*Y
```



```
                  DEFB  +36,menor-0           Z-1, (1/0)
                  DEFB  +00,saltar-verdade    Z-1
                  DEFB  +02,para 37A8,YNEG    Z-1
                  DEFB  +1B,negar             1-Z
37A8 YNEG         DEFB  +38,fim-calc          1-Z = V - caso 2.
                                              Z-1 = V - caso 3.
                  RET                         Final.
```

A função «CO-SENO»
(Deslocamento 20: «cos»)

Esta subrotina trata a função COS X e produz um «último valor» que constitui uma aproximação de COS X.
A subrotina utiliza a expressão:

$$COS\ X = SIN\ (PI \cdot W/2),$$

onde $-1 <= W <= 1$.
Ao obter W a subrotina utiliza o resultado do teste obtido na subrotina anterior e guardado para este fim em mem-0. Salta em seguida para a subrotina SIN, entrando em C-ENT, a fim de produzir um «último valor» de COS X.

```
37AA cos          RST   0028,FP-CALC          X
                  DEFB  +39,obter-argt        V
                  DEFB  +2A,abs               ABS V
                  DEFB  +A1,stk-um            ABS V, 1
                  DEFB  +03,subtrair          ABS V-1
                  DEFB  +E0,obter-mem-0       ABS V-1, (1/0)
                  DEFB  +00,saltar-verdade    ABS V-1
                  DEFB  +06,para 37B7,C-ENT   ABS V-1 = W
```

Se não houve salto, continuar.

```
                  DEFB  +1B,negar             1-ABS V
                  DEFB  +33,saltar            1-ABS V
                  DEFB  +03,para 37B7,C-ENT   1-ABS V = W
```

A função «SENO»
(Deslocamento 1F: «sin»)

Esta subrotina trata a função SIN X e é a terceira das quatro rotinas que utilizam o GERADOR DE SÉRIES para produzir polinómios de Chebyshev.
O valor aproximado de SIN X é obtico do seguinte modo.

1. O argumento X é reduzido, e neste caso W=V directamente.
   Note-se que $-1 <= W <= 1$, como é necessário para que a série seja convergente.
2. Forma-se o argumento Z, tal que $Z = 2 \cdot W \cdot W - 1$.
3. É usado o GERADOR DE SÉRIES para produzir $(SIN\ (PI \cdot W/2))/W$.
4. Finalmente, uma simples multiplicação produz SIN X.

```
3785 sin          RST   0028 FP-CALC         X
```

Executar o passo 1.

```
        DEFB  +39,obter-argt        W
```

Executar o passo 2. A subrotina é daqui em diante comum a ambas as funções SENO e CO-SENO.

```
3787 C-ENT   DEFB  +31,copiar            W, W
             DEFB  +31,copiar            W, W, W
             DEFB  +04,multiplicar       W, W*W
             DEFB  +31,copiar            W, W*W, W*W
             DEFB  +0F,somar             W, 2*W*W
             DEFB  +A1,stk-um            W, 2*W*W, 1
             DEFB  +03,subtrair          W, 2*W*W-1 = Z
```

Executar o passo 3, passando ao GERADOR DE SÉRIES o parâmetro «6» e as seis constantes requeridas.

```
             DEFB  +86,série-06          W, Z
        1.   DEFB  +14,expoente+64
             DEFB  +E6,(+00,+00,+00)
        2.   DEFB  +5C,expoente +6C
             DEFB  +1F,+0B,(+00,+00)
        3.   DEFB  +A3,expoente+73
             DEFB  +8F,+38,+EE,(+00)
        4.   DEFB  +E9,expoente+79
             DEFB  +15,+63,+BB,+23
        5.   DEFB  +EE,expoente+7E
             DEFB  +92,+0D,+CD,+ED
        6.   DEFB  +F1,expoente+81
             DEFB  +23,+5D,+1B,+EA
```

No final do último ciclo, o «último valor» é (SIN (PI•W/2))/W.
Executar o passo 5.

```
             DEFB  +04,multiplicar       SIN (PI*W/2) = SIN X (or =
                                         COS X)
             DEFB  +38,fim-calc
             RET                         Final: «último valor»=SIN X
                                         ou («último valor»=COS X).
```

A função «TAN»
(Deslocamento 21: «tan»)

Esta subrotina trata a função TAN X. Produz simplesmente SIN X/COS X, com «overflow» aritmético se COS X=0.

```
37DA tan     RST   0028,FP-CALC        X
             DEFB  +31,copiar          X, X
             DEFB  +1F,sin             X, SIN X
             DEFB  +01,trocar          SIN X, X
             DEFB  +20,cos             SIN X,COS X
             DEFB  +05,divisão         SIN X/COS X = TAN X
                                       Mensagem de excesso
                                       aritmético, se necessário.
             DEFB  +38,fim-calc        TAN X
             RET                       Final: «último valor»=TAN X.
```


A função «ARCTAN»
(Deslocamento 24: «atn»)

Esta subrotina trata a função ATN X e é a última das quatro rotinas que utilizam o GERADOR DE SÉRIES para produzir polinómios de Chebyshev. Produz um número real entre -PI/2 e PI/2, que é igual ao valor em radianos do ângulo cuja tangente é X.

O valor aproximado de ATN X é determinado do seguinte modo:

1. Acham-se os valores W e Y para três casos de X, a saber:

   Se -1<X<1, então W=0 e Y=X — caso 1

   Se 1<=X, então W=PI/2 e Y=-1/X — caso 2

   Se X<=-1, então W=-PI/2 e Y=-1/X — caso 3.

   Em todos os casos, -1<=Y<=1, como é necessário para que a série seja convergente.

2. É formado o argumento Z, tal que:

   Se -1<X<1, então Z=2•Y•Y-1=2•X•X-1 — caso 1

   Se 1<X, então Z=2•Y•Y-1=2/(X•X)-1 — caso 2

   Se X<=-1, então Z=2•Y•Y-1=2/(X•X)-1 — caso 3

3. O GERADOR DE SÉRIES é agora usado para produzir a função requerida.

4. Finalmente, uma multiplicação e adição dão ATN X.

Executar fase 1.

```
37E2 atn     CALL  3297,RE-STACK       Usar a forma em vírgula
                                       flutuante de X.
             LD    A,(HL)              obter o expoente de X.
             CP    +81
             JR    C,37F8,SMALL        Saltar para diante no caso 1.
             RST   0028,FP-CALC        X
             DEFB  +A1,stk-um          X, 1
             DEFB  +1B,negar           X,-1
             DEFB  +01,trocar          -1, X
             DEFB  +05,dividir         -1/X
             DEFB  +31,copiar          -1/X, -1/X
             DEFB  +36,menor-0         -1/X, (1/0)
             DEFB  +A3,stk-pi/2        -1/X, (1/0), PI/2
             DEFB  +01,trocar          -1/X, PI/2, (1/0)
             DEFB  +00,saltar-verdade  -1/X, PI/2
             DEFB  +06,para 37FA,CASES Saltar para diante no caso 2:
                                       Y = -1/X  W = PI/2
             DEFB  +1B,negar           -1/X, -PI/2
             DEFB  +33,saltar          -1/X, -PI/2
             DEFB  +03,para 37FA,CASES saltar para diante no caso 3:
                                       Y = -1/X  W = -PI/2
37F8 SMALL   RST   0028,FP-CALC        Y
             DEFB  +A0,stk-zero        Y, 0
                                       Continuar no caso 1: W=0
```

Executar passo 2.

```
37FA CASES    DEFB  +01,trocar          W, Y
              DEFB  +31,copiar          W, Y, Y
              DEFB  +31,copiar          W, Y, Y, Y
              DEFB  +04,multiplicar     W, Y, Y*Y
              DEFB  +31,copiar          W, Y, Y*Y, Y*Y
              DEFB  +0F,somar           W, Y, 2*Y*Y
              DEFB  +A1,stk-um          W, Y, 2*Y*Y, 1
              DEFB  +03,subtrair        W, Y, 2*Y*Y-1 = Z
```

Executar o passo 3, passando ao GERADOR DE SÉRIES o parâmetro «12» decimal, e as doze constantes requeridas.

```
              DEFB  +8C,série-0C        W, Y, Z
1.            DEFB  +10,expoente+60
              DEFB  +B2,(+00,+00,+00)
2.            DEFB  +13,expoente+63
              DEFB  +0E,(+00,+00,+00)
3.            DEFB  +55,expoente+65
              DEFB  +E4,+8D,(+00,+00)
4.            DEFB  +58,expoente+68
              DEFB  +39,+BC,(+00,+00)
5.            DEFB  +5B,expoente+6B
              DEFB  +98,+FD,(+00,+00)
6.            DEFB  +9E,expoente+6E
              DEFB  +00,+36,+75,(+00)
7.            DEFB  +A0,expoente+70
              DEFB  +D8,+E8,+B4,(+00)
8.            DEFB  +63,expoente+73
              DEFB  +42+C4,(+00,+00)
9.            DEFB  +E6,expoente+76
              DEFB  +B5,+09,+36,+BE
10.           DEFB  +E9,expoente+79
              DEFB  +36,+73,+1B,+5D
11.           DEFB  +EC,expoente+7C
              DEFB  +D8,+DE,+63,+BE
12.           DEFB  +F0,expoente+80
              DEFB  +61,+A1,+B3,+0C
```

No fim do último ciclo, o «último valor» é:

```
ATN X/X                 — caso 1
ATN (-1/X)/(-1/X)       — caso 2
ATN (-1/X)/(-1/X)       — caso 3
```

Executar o passo 4.

```
              DEFB  +04,multiplicar     W,ATN X          — caso 1
                                        W,ATN (-1/X)     — caso 2
                                        W,ATN (-1/X)     — caso 3
              DEFB  +0F,somar           ATN X — todos os casos
              DEFB  +38,fim-calc
              RET                       Final: «último valor»=ATN X
```


A função «ARCSIN»
(Deslocamento 22: «asn»)

Esta subrotina trata a função ASN X e produz um número real entre -PI/2 e PI/2 inclusive, que é igual ao valor em radianos do ângulo cujo seno é X. Assim, se Y=ASN X, então X=SIN Y.

Esta subrotina usa a identidade trigonométrica:

TAN (Y/2)=SIN Y/1(1+COS Y)

a fim de obter TAN (Y/2), e daí (usando ATN) Y/2 e finalmente Y.

```
3833 asn      RST   0028,FP-CALC      X
              DEFB  +31,copiar        X, X
              DEFB  +31,copiar        X, X, X
              DEFB  +04,multiplicar   X, X*X
              DEFB  +A1,stk-um        X, X*X, 1
              DEFB  +03,subtrair      X, X*X-1
              DEFB  +1B,negar         X,1-X*X
              DEFB  +28,sqr           X,SQR (1-X*X)
              DEFB  +A1,stk-um        X,SQR (1-X*X), 1
              DEFB  +0F,somar         X, 1+SQR (1-X*X)
              DEFB  +05,dividir       X/(1+SQR (1-X*X)) = TAN
                                      (Y/2)
              DEFB  +24,atn           Y/2
              DEFN  +31,copiar        Y/2, Y/2
              DEFN  +0F,somar         Y = ASN X
              DEFB  +38,fim-calc
              RET                     Final: «último valor»=ASN X.
```

A função «ARCCOS»
(Deslocamento 23: «acs»)

Esta subrotina trata a função ACS X e produz um número real entre zero e PI inclusive, que é igual ao valor em radianos do ângulo cujo coseno é X.

Esta subrotina utiliza a relação:

ACS X=PI/2 — ASN X

```
3843 acs      RST   0028,FP-CALC      X
              DEFB  +22,asn           ASN X
              DEFB  +A3,stk-pi/2      ASN X, PI/2
              DEFN  +03,subtrair      ASN X-PI/2
              DEFB  +1B,negar         PI/2-ASN X = ACS X
              DEFB  +38,fim-calc
              RET                     Final: «último valor»=ACS X
```

A função «Raiz quadrada»
(Deslocamento 28: «sqr»)

Esta subrotina trata a função SQR X e produz a raiz quadrada positiva do número real X se X é positivo, e zero se X é zero. Um valor negativo de X dá origem à mensagem «A — Invalid argument» (através da subrotina EXPONENCIAÇÃO).

A subrotina trata a operação de radiciação como sendo X↑.5, e guarda portanto o valor .5, passando directamente à subrotina EXPONENCIAÇÃO.

```
384A sqr    RST   0028,FP-CALC          X
            DEFB  +31,copiar            X, X
            DEFB  +30,not               X, (1/0)
            DEFB  +00,saltar-verdade    X
            DEFB  +1E,para 386C,LAST    X
```

O salto é realizado no caso X=0, senão, continuar com:

```
            DEFB  +A2,stk-meio          X, .5
            DEFB  +38,fim-calc
```

e determinar depois o resultado de X↑.5.

A operação «EXPONENCIAÇÃO»
(Deslocamento 06:«to-power»)

Esta subrotina realiza a operação binária de elevação do primeiro número, X, à potência equivalente ao segundo número, Y.

A subrotina trata o resultado X↑Y como equivalente a EXP (Y•LN X). Produz este valor desde que X não seja zero, e no caso contrário, produz 1 se Y for igualmente zero (0↑0=1), ou zero se Y for positivo, indicando excesso aritmético se Y é negativo.

```
3851 to-power  RST   0028,FP-CALC          X,Y
               DEFB  +01,trocar            Y, X
               DEFB  +31,copiar            Y, X, X
               DEFB  +30,not               Y, X, (1/0)
               DEFB  +00,saltar-verdade    Y, X
               DEFB  +07,para 385D,XISO    Y, X
```

O salto é realizado se X=0, senão, é formado EXP (Y • LN X).

```
               DEFB  +25,ln                Y, LN X
                                           Dando mensagem A se X<0.
               DEFB  +04,multiplicar       Y*LN X
               DEFB  +38,fim-calc
               JP    36C4,EXP              Saída por EXP para formar
                                           EXP (Y•LN X).
```

O valor de X é zero, pelo que se devem considerar os três casos possíveis.

```
385D XISO      DEFB  +02,apagar            Y
               DEFB  +31,copiar            Y, Y
               DEFB  +30,not               Y, (1/0)
               DEFB  +00,saltar-verdade    Y
               DEFB  +09,para 386A,ONE     Y
```

O salto é realizado se X=0 e Y=0, senão, continua.

```
               DEFB  +A0,stk-zero          Y,0
               DEFB  +01,trocar            0, Y
               DEFB  +37,maior-0           0, (1/0)
               DEFB  +00,saltar-verdade    0
               DEFB  +096,para 386C, LAST  0
```


Saltar se X=0 e Y é positivo, senão, continuar.

```
               DEFB  +A1,stk-um            0, 1
               DEFB  +01,trocar            1, 0
               DEFB  +05,dividir           Sair por «divisão», pois
                                           dividir por zero produz
                                           «excesso aritmético».
```

O resultado será um.

```
386A ONE       DEFB  +02,apagar
               DEFB  +A1,stk-um            1
```

Retorno agora, com o «último valor» em «stack» igual a 0↑Y.

```
386C LAST      DEFB  +38,fim-calc          (1/0)
               RET                         Final: «último valor» é 0 ou 1.
```

386E — 3CFF. Estas posições estão «livres». Contêm +FF.

3D00 - 3FFF. Estas posições contêm o «conjunto de caracteres». Existem representações por oito bytes de todos os códigos desde +20 (espaço) a +7F (©).

Por exemplo, a letra «A» possui a representação 00 3C 42 42 7E 42 42 00, ou seja a forma:

```
00000000
00111100
01000010
01000010
01111110
01000010
01000010
00000000
```

APÊNDICE

Programas Basic para as séries principais

Os programas Basic que se seguem foram aqui incluídos por constituírem uma boa ilustração do modo como os polinómios de Chebyshev são usados para produzir valores aproximados das funções SIN, EXP, LN e ATN.

Gerador de séries

Esta subrotina é invocada por todos os programas «função».

```
500 REM GERADOR DE SÉRIES.
510 REM ENTRAR USANDO CONTADOR
520 REM BREG E ARRAY-A CONTENDO
530 REM AS CONSTANTES.
540 REM PRIMEIRO VALOR EM Z.
550 LET M0=2*Z
560 LET M2=0
570 LET T=0
580 FOR I=BREG TO 1 STEP -1
590 LET M1=M2
600 LET U=T*M0-M2+A(BREG+1-I)
610 LET M2=T
620 LET T=U
630 NEXT I
640 LET T=T-M1
650 RETURN
660 REM ÚLTIMO VALOR EM T.
```

Na subrotina acima as variáveis são:

Z	— o valor de entrada
T	— o valor de saída.
M0	— mem-0
M1	— mem-1
M2	— mem-2
I	— o contador de BREG.
U	— uma variável temporária para T.
A(1) a A(BREG)	— as constantes.
BREG	— o número de constantes a usar.

Para ver como são produzidos os polinómios de Chebyshev, registem-se num papel os valores de U, M1, M2 e T ao longo das linhas 550 a 630, passando, por exemplo, 6 vezes pelo ciclo, e mantendo as expressões algébricas de A(1) a A(6) sem substituir valores numéricos. Registar depois T-M1. Os multiplicadores das constantes A(1) a A(6) constituirão então os polinómios de Chebyshev requeridos. Mais precisamente, o multiplicador de A(1) será $2 \cdot T_5(Z)$, o de A(2) será $2 \cdot T_4(Z)$ e assim por diante até $2 \cdot T_1(Z)$ no caso de A(5), e finalmente $T_0(Z)$ para A(6).

Note-se que $T_0(Z)=1$, $T_1(Z)=Z$ e, para $n>=2$, $T_n(Z)=2 \cdot Z \cdot T_{n-1}(Z)-T_{n-2}(Z)$.

SIN X

```
10 REM DEMONSTRAÇÃO PARA SIN X
20 REM USANDO «GERADOR DE SÉRIES»
30 DIM A(6)
40 LET A(1)=-.000000003
50 LET A(2)=-0.000000592
60 LET A(3)=-.000068294
70 LET A(4)=0.004559008
80 LET A(5)=-.142638785
90 LET A(6)=1.276278962
100 PRINT
110 PRINT «INDIQUE VALOR EM GRAUS»
120 INPUT C
130 CLS
140 LET C=C-10
150 PRINT «PROGRAMA BASIC», «PROGRAMA ROM»
160 PRINT ----------, ----------
170 PRINT
180 FOR J=1 TO 4
190 LET C=C+10
200 LET Y=C/360-INT (C/360+.5)
210 LET W=4*Y
220 IF W>1 THEN LET W=2-W
230 IF W<-1 THEN LET W=-W-2
240 LET Z=2*W*W-1
250 LET BREG=6
260 REM USAR «GERADOR DE SÉRIES»
270 GO SUB 550
280 PRINT TAB 6; «SIN »;C;» GRAUS»
290 PRINT
300 PRINT T*W,SIN (PI*C/180)
310 PRINT
320 NEXT J
330 GO TO 100
```

Notas:

1. Quando é indicado C o programa calcula e imprime SIN C graus, SIN (C+10) graus, SIN (C+20) graus e SIN (C+30) graus. Imprime igualmente os valores obtidos usando o programa ROM. Para obter um exemplo dos resultados, experimente introduzir estes valores em graus: 0; 5; 100; -80; -260; 3600; -7200.
2. As constantes A(1) a A(6) nas linhas 40 a 90 são apresentadas (além de um factor de 1/2) em Abramowitz e Stegun, *Handbook of Mathematical*

Functions (Dover 1965), página 76. Podem ser verificadas integrando (SIN (PI•X/2))/X no intervalo U=0 a PI, depois de multiplicar primeiro por COS (N•U) para cada constante (isto é, N=1, 2, ... , 6) e substituindo COS U=2•X•-1. Cada resultado deverá então ser dividido por PI (esta integração pode ser realizada por métodos aproximados, por exemplo, usando a Regra de Simpson se existe um computador ou máquina de calcular programável à mão).

EXP X

```
10 REM DEMONSTRAÇÃO DE EXP X
20 REM USANDO O «GERADOR DE SÉRIES»
30 LET T=0                                  (Isto passa T a primeira variável)
40 DIM A(8)
50 LET A(1)=0.000000001
60 LET A(2)=0.000000053
70 LET A(3)=0.000001851
80 LET A(4)=0.000053453
90 LET A(5)=0.001235714
100 LET A(6)=0.021446556
110 LET A(7)=0.248762434
120 LET A(8)=1.456999875
130 PRINT
140 PRINT «INDIQUE VALOR INICIAL»
150 INPUT C
160 CLS
170 LET C=C-10
180 PRINT «PROGRAMA BASIC», «PROGRAMA ROM»
190 PRINT ←————————, ←————————→
200 PRINT
210 FOR J=1 TO 4
220 LET C=C+10
230 LET D=C•1.442695041                     (D=C•(1/LN 2); EXP C=2↑D)
240 LET N=INT D
250 LET Z=D-N                               (2↑(N+Z) é requerido agora)
260 LET Z=2•Z-1
270 LET BREG=8
280 REM USAR «GERADOR DE SÉRIES»
290 GO SUB 550
300 LET V=PEEK 23627+256•PEEK 23628+1       (V=(VARS)+1)
310 LET N=N+PEEK V
320 IF N>255 THEN STOP                      (STOP com excesso aritmético)
330 IF N<0 THEN GO TO 360
340 POKE V,N
350 GO TO 370
360 LET T=0
370 PRINT TAB 11;«EXP »;C
380 PRINT
390 PRINT T, EXP C
400 PRINT
410 NEXT J
420 GO TO 130
```


Notas:

1. Quando se indica C este programa calcula e imprime EXP C, EXP (C+10), EXP (C+20) e EXP (C+30). Imprime igualmente os valores obtidos usando o programa ROM. Para um exemplo dos resultados, experimente introduzir os seguintes valores: 0; 15; 65 (com *overflow* no final); -100; -40.
2. Verifica-se o expoente, para o caso de excesso («overflow») ou de resultado zero, nas linhas 320 e 330. Estes testes são mais simples em Basic do que em código-máquina, dado que a variável N, ao contrário do registo A, não está limitada a um único byte.
3. As constantes A(1) a A(8) nas linhas 50 a 120 podem ser obtidas integrando 2↑X ao longo do intervalo U=0 a PI, depois de multiplicar COS (N•U) para cada constante (isto é, para N=1, 2, . . . , 8) e substituir COS U=2•X-1. Cada resultado deve então ser dividido por PI.

LN X:

```
10 REM DEMONSTRAÇÃO DE LN X
20 REM USANDO O «GERADOR DE SÉRIES»
30 LET D=0                                  (Isto passa D a primeira variável)
40 DIM A(12)
50 LET A(1)=-.0000000003
60 LET A(2)=0.0000000020
70 LET A(3)=-.0000000127
80 LET A(4)=0.0000000823
90 LET A(5)=-.0000005389
100 LET A(6)=0.0000035828
110 LET A(7)=-.0000243013
120 LET A(8)=0.0001693953
130 LET A(9)=-.0012282837
140 LET A(10)=0.0094766116
150 LET A(11)=-.0818414567
160 LET A(12)=0.9302292213
170 PRINT
180 PRINT «INDIQUE VALOR INICIAL»
190 INPUT C
200 CLS
210 PRINT «PROGRAMA BASIC», «PROGRAMA ROM»
220 PRINT ←————————, ←————————→
230 PRINT
240 LET C=SQR C
250 FOR J=1 TO 4
260 LET C=C•C
270 IF C=0 THEN STOP                        (STOP com «Invalid argument»)
280 LET D=C
290 LET V=PEEK 23627+256•PEEK 23628+1
300 LET N=PEEK V-128                        (N contém e')
310 POKE V,128
320 IF D<=0.8 THEN GO TO 360                (D contém X')
330 LET S=D-1
340 LET Z=2.5•D-3
350 GO TO 390
```

```
360 LET N=N-1
370 LET S=2*D-1
380 LET Z=5*D-3
390 LET R =N*0.6931471806                (R contém N*LN 2)
400 LET BREG=12
410 REM USAR «GERADOR DE SÉRIES»
420 GO SUB 550
430 PRINT TAB 8;«LN »;C
440 PRINT
450 PRINT S*T+R,LN C
460 PRINT
470 NEXT J
480 GO TO 170
```

Notas:

1. Quando se indica C, o programa calcula e imprime LN C, LN (C↑2), LN (C↑4) e LN (C↑8). Imprime igualmente os valores obtidos usando o programa em ROM.
   Para um exemplo dos resultados, experimente os seguintes valores: 1.1; 0.9; 300; 0.004; 1E5 (para overflow) e 1E-5 (STOP por «Invalid argument»).

2. As constantes A(1) a A(12) nas linhas 50 a 160 podem ser obtidas integrando 5*LN (4*(X+1)/5)/(4*X-1) no intervalo U=0 até PI, depois de multiplicar por COS (N*U) para cada constante (isto é, para N=1, 2, ..., 12) e de substituir COS U=2*X-1. Cada resultado deve então ser dividido por PI.

ATN X

```
10 REM DEMONSTRAÇÃO DE ATN X
20 REM USANDO O «GERADOR DE SÉRIES»
30 DIM A(12)
40 LET A(1)=-.0000000002
50 LET A(2)=0.0000000010
60 LET A(3)=-.0000000066
70 LET A(4)=0.0000000432
80 LET A(5)=-.0000002850
90 LET A(6)=0.0000019105
100 LET A(7)=-.0000131076
110 LET A(8)=0.0000928715
120 LET A(9)=-.0006985975
130 LET A(10)=0.0055679210
140 LET A(11)=-.0529464623
150 LET A(12)=0.8813735870
160 PRINT
170 PRINT «INDIQUE VALOR INICIAL»
180 INPUT C
190 CLS
200 PRINT «PROGRAMA BASIC», «PROGRAMA ROM»
210 PRINT ---------------, ---------------
220 PRINT
230 FOR J=1 TO 4
240 LET B=J*C
250 LET D=B
```



```
260 IF ABS B>=1 THEN LET D=-1/B
270 LET Z=2*D*D-1
280 LET BREG=12
290 REM USAR «GERADOR DE SÉRIES»
300 GO SUB 550
310 LET T=D*T
320 IF B>= 1 THEN LET T=T+PI/2
330 IF B<=-1 THEN LET T=T-PI/2
340 PRINT TAB 8;«ATN »;B
350 PRINT
360 PRINT T,ATN B                (ou PRINT T*180/PI, ATN B*180/PI
370 PRINT                        para obter a resposta em graus)
380 NEXT J
390 GO TO 160
```

Notas:

1. Quando C é introduzido, este programa calcula e imprime ATN C, ATN (C*2), ATN (C*3) e ATN (C*4).
   Para um exemplo dos resultados, experimente introduzir os valores seguintes: 0.2; -1; 10 e -100. Os resultados poderão ser mais interessantes se forem convertidos em graus multiplicando as respostas na linha 360 por 180/PI.

2. As constantes A(1) a A(12) nas linhas 40 a 150 são dadas (além de um factor de 1/2) em Abramowitz e Stegun, *Handbook of Mathematical Functions* (Dover 1965), página 82. Podem ser verificadas integrando ATN X/X no intervalo U=0 a PI, depois de multiplicar por COS (N*U) para cada parâmetro (isto é, para n=1, 2, . . . , 12) e substituindo COS U=2*X*X-1. Cada resultado deve então ser dividido por PI.

Uma subrotina alternativa para SIN X:
É simples obter o desenvolvimento completo dos polinómios de Chebyshev, e isto pode ser escrito em Basic do seguinte modo:

```
550 LET T=(32*Z*Z*Z*Z*Z-48*Z*Z*Z+18*Z)*A(1)
         +(16*Z*Z*Z*Z-16*Z*Z+2)*A(2)
         +(8*Z*Z*Z-8*Z)*A(3)
         +(4*Z*Z-2)*A(4)
         +2*Z *A(5)
         +A(6)
560 RETURN
```

Esta subrotina é invocada em vez do GERADOR DE SÉRIES, e como se pode verificar tem um rigor semelhante.
Uma subrotina alternativa para EXP X:
O desenvolvimento completo de EXP X é:

```
550 LET T=(128*Z*Z*Z*Z*Z*Z*Z-224*Z*Z*Z*Z*Z+112*Z*Z*Z-14*Z)*A(1)
         +(64*Z*Z*Z*Z*Z*Z-96*Z*Z*Z*Z+36*Z*Z-2)*A(2)
         +(32*Z*Z*Z*Z*Z-48*Z*Z*Z+18*Z)*A(3)
         +(16*Z*Z*Z*Z-16*Z*Z+2)*A(4)
         +(8*Z*Z*Z-8*Z)*A(5)
         +(4*Z*Z-2)*A(6)
         +2*Z *A(7)
         +A(8)
560 RETURN
```

O desenvolvimento completo de LN X e ATN X, dado algebricamente, será:

```
(2048z^11-5632z^9+5632z^7-2464z^5+440z^3-22z) * A(1)
+ (1024z^10-2560z^8+2240z^6-800z^4+100z^2-2) * A(2)
+ (512z^9-1152z^7+864z^5-240z^3+18z) * A(3)
+ (256z^8-512z^6+320z^4-64z^2+2) * A(4)
+ (128z^7-224z^5+112z^3-14z) * A(5)
+ (64z^6-96z^4+36z^2-2) * A(6)
+ (32z^5-40z^3+10z) * A(7)
+ (16z^4-16z^2+2) * A(8)
+ (8z^3-6z) * A(9)
+ (4z^2-2) * A(10)
+ (2z) * A(11)
+ A(12)
```

O algoritmo «DRAW»

O programa Basic que se segue ilustra as partes essenciais da operação DRAW quando usada para produzir uma linha recta. O programa, na sua forma actual, só permite a execução de linhas em que X>Y.

```
10 REM PROGRAMA DRAW 255,175
20 REM DEFINIR ORIGEM
30 LET PLOTx=0: LET PLOTy=0
40 REM DEFINIR LIMITES
50 LET X=255: LET Y=175
60 REM DEFINIR INCREMENTO, i
70 LET i=X/2
80 REM REALIZAR CICLO
90 FOR B=X TO 1 STEP -1
100 LET A=Y+i
110 IF X>A THEN GO TO 160
120 REM SOBE UM PIXEL
130 LET A=A-X
140 LET PLOTy=PLOTy+1
150 REM REDEFINE INCREMENTO, i
160 LET i=A
170 REM SEMPRE MAIS UM PIXEL
180 LET PLOTx=PLOTx+1
190 REM EXECUTAR «PLOT»
200 PLOT PLOTx,PLOTy
210 NEXT B
```

Podemos encontrar um algoritmo completo no programa que se segue, omo subrotina que desenhará uma linha a partir da última posição de X,Y.

Algoritmo «CIRCLE»

O programa Basic que se segue ilustra a forma como a ordem CIRCLE executada.

Inicialmente, é calculado o número de arcos requeridos. Depois, é preparado um conjunto de parâmetros na «área de memória» e no «stack do calculador».

Os arcos são então desenhados por chamadas repetitivas à subrotina de desenho de linhas, que, em cada caso, desenha uma linha desde a «última posição» até à posição «X,Y».

Nota: No programa em ROM existe uma linha «de fecho» final, mas esta não é incluída aqui.

```
10 REM PROGRAMA DE CÍRCULOS
20 LET X=127: LET Y=87: LET Z=87
30 REM QUANTOS ARCOS?
40 LET ARCOS=4*INT (INT (ABS (PI*SQR Z)+0.5)/4)+4
50 REM DEFINIR ÁREA DE MEMÓRIA; M0 a M5
60 LET M0=X+Z
70 LET M1=0
80 LET M2=2*Z*SIN (PI/ARCOS)
90 LET M3=1-2*(SIN (PI/ARCOS))12
100 LET M4=SIN (2*PI/ARCOS)
110 LET M5=2*PI
120 REM DEFINIR STACK; SA-SD
130 LET SA=X+Z
140 LET SB=Y-Z*SIN (PI/ARCOS)
150 LET SC=SA
160 LET SD=SB
170 REM INICIALIZAR COORDS
180 POKE 23677,SA: POKE 23678,SB
190 LET M0=SD
200 REM DESENHAR OS ARCOS
210 LET M0=M0+M2
220 LET SC=SC+M1
230 LET X=SC-PEEK 23677
240 LET Y=M0-PEEK 23678
250 GO SUB 510
260 LET ARCOS=ARCOS-1: IF ARCOS=0 THEN STOP
270 LET MM1=M1
280 LET M1=M1*M3-M2*M4
290 LET M2=MM1*M4+M2*M3
300 GO TO 210

500 REM DESENHAR LINHA desde última posição até X,Y
510 LET PLOTx=PEEK 23677: LET PLOTy=PEEK 23678
520 LET dx=SGN X: LET dy=SGN Y
530 LET X=ABS X: LET Y=ABS Y
540 IF X>=Y THEN GO TO 580
550 LET L=X: LET B=Y
560 LET DDX=0. LET DDY=DY
570 GO TO 610
580 IF X+Y=0 THEN STOP
590 LET L=Y: LET B=X
600 LET DDX=DX: LET DDY=0
610 LET H=B
620 LET I=INT (B/2)
630 FOR N=B TO 1 STEP -1
640 LET I=I+L
650 IF I<H THEN GO TO 690
660 LET I=I-H
```

```
670 LET IX=DX: LET IY=DY
680 GO TO 700
690 LET IX=DDX: LET IY=DDY
700 LET PLOTy=PLOTy+IY
710 IF PLOTy<0 OR PLOTy>175 THEN STOP
720 LET PLOTx=PLOTx+IX
730 IF PLOT x<0 OR PLOTx>255 THEN STOP
740 PLOT PLOTx,PLOTy
750 NEXT N
760 RETURN
```

Nota sobre inteiros pequenos e -65536.

1. Os pequenos inteiros N são aqueles para os quais -65536 é menor ou igual a N, menor ou igual a 65535. A forma em que são guardados é descrita em «STACK-BC». Note-se que o manual não é rigoroso quando afirma que o terceiro e quarto bytes contêm N mais 131072 se N é negativo. Dado que a gama de N é então -1 a -65535, os dois bytes podem apenas guardar N mais 131072 se for considerado o módulo 65536; isto é, contêm N mais 65536. O manual confunde a questão. O facto é que este não é um verdadeiro complemento para dois (como a forma N mais 131072, noutras circunstâncias, poderia ser). Aqui o mesmo número pode representar dois números diferentes conforme o byte de sinal; ou seja, 00 01 indica 1 se o byte de sinal for 00, e -65535 se o byte de sinal for FF. Do mesmo modo, FF FF indica 65535 se o byte de sinal for 00, e -1 se o byte de sinal for FF.

2. Aceitando que os números negativos recebem uma forma especial em «complemento para dois», a principal característica deste método de guardar números é que estes se encontram preparados para uma «adição curta» sem necessitar de uma complementação para dois. São simplesmente obtidos e guardados directamente pela rotina de adição. Mas para multiplicação devem ser obtidos por INT-FETCH e guardados depois por INT-STORE. Estas subrotinas complementam o número para dois quando o obtêm ou guardam. As chamadas a INT-STORE são realizadas por «multiplicar» (após «multiplicação curta»), por «truncar» (depois de produzir um «inteiro pequeno» entre -65535 e 65535 inclusive), por «negar/abs» para o caso «inteiro», e por «sgn» para guardar 1 ou -1. As chamadas a INT-FETCH são executadas por PRINT-FP para obter a parte inteira do número quando é «pequeno», por «multiplicar», duas vezes, para obter dois «inteiros pequenos», por RE-STACK para obter um «inteiro pequeno» para guardar no «stack», por «negar/abs» para obter um «inteiro pequeno» para manipulação e por FP-TO-BC para obter o inteiro para transferência para BC.

O número -65536

3. O número -65536 pode ser representado no formato «inteiro pequeno» sob a forma 00 FF 00 00 00. É então o «número limite», aquele que produz um excesso («overflow») quando é complementado para dois (notar 80 hex num simples byte ou sistema de 7 bits, ou seja -128 decimal, que quando complementado para dois produz ainda 80 hex., ou seja, -128 decimal, porque o número positivo 128 decimal não cabe no sistema).

4. Deverá ter sido a consciência deste facto que deu origem à tentativa de criar 00 FF 00 00 00 em «truncar». Foi abortada porque nem sequer sobrevive à rotina INT de que «truncar» é uma parte. Conduz simplesmente ao erro INT (-65536) igual a -1.

5. Mas o principal erro é que foi permitido que este número resultasse da «adição curta» de dois inteiros negativos mais pequenos, sendo, em seguida, colocado simplesmente no «stack» sob a forma 00 FF 00 00 00. O sistema não pode tratar este número. A solução proposta em «somar» consiste em produzir imediatamente a forma equivalente em vírgula flutuante, ou seja, em comparar primeiro o número, por volta do byte 3032, do seguinte modo:

```
3032             PUSH  AF              Guardar byte de sinal em A.
3033             INC   A               Passar qualquer FF de A para 00.
3034             OR    E               Verificar os 3 bytes (07).
3035             OR    D
3036             JR    NZ,3040,ADD-STORE Saltar se não é -65536.
3038             POP   AF              Limpar o «stack».
3039             LD    (HL),+80        Passar 80 hex para o 2.º byte.
303B             DEC   HL              Apontar para o 1.º byte.
303C             LD    (HL),+91        Passar 91 hex para o 1.º byte.
303E             JR    3049,ADD-RSTOR  Saltar para definir indicador; sair.
3040 ADD-STORE   POP   AF              Restaurar em A o byte de sinal.
3041             LD    (HL),A          Guardá-lo no «stack».
3042             INC   HL              Apontar para a posição seguinte.
3043             LD    (HL),E          Guardar byte baixo do resultado.
3044             INC   HL              Apontar para a posição seguinte.
3045             LD    (HL),D          Guardar o byte alto do resultado.
3046             DEC   HL              Levar o indicador a endereçar
3047             DEC   HL              de novo o primeiro byte do
3048             DEC   HL              resultado.
3049 ADD-RSTOR  POP   DE              Restaurar STKEND em DE.
304A             RET                   Final.
```

6. A emenda acima apresentada (num total de 15 bytes extra), juntamente com a omissão dos bytes 3223 a 323E de «truncar», deveria resolver o problema. Seria agradável poder verificar esta solução. As chamadas a INT-STORE não deveriam conduzir ao armazenamento da forma 00 FF 00 00 00 no «stack». Em «multiplicar», o número conduzirá a um excesso («overflow») se ocorrer, dado que 65536 passará ao valor um a flag «carry»; será portanto necessário recorrer à «multiplicação longa». Como se fez notar em 30E5, os cinco bytes a partir daqui poderão provavelmente ser omitidos se for introduzida a emenda proposta. «Negar» evita o armazenamento de 00 FF 00 00 00 tratando o zero separadamente e devolvendo-o sem alterações. «Truncar» trata separadamente -65536, como se notou acima. SGN guarda apenas 1 e -1.

ÍNDICE