

Stimate cumpărător,

The COMPLETE SPECTUM ROM DISASSEMBLY (ROM-ul complet dez asamblat) este o carte unanima acceptată ca fiind manualul de referință pentru toti utilizatorii serioși ai calculatoarelor Spectrum și compatibile. Cartea lui Dr. Ian Logan & Dr. Frank O'Hara, scrisă la editura Melbourne House Publishers este tot ceea ce poate visa un utilizator de Spectrum & compatibile pentru a obține cât mai mult din sistemul său. La această lucrare firma noastră nu urmărește decât popularizarea unei lucrări absolut remarcabile (am listat cu 70 de caractere pe rând peste 300 de pagini, compact, la cel mai scăzut pret posibil) pentru a ajuta la înțelegerea cât mai detaliată a acestei minunate unelte care este calculatorul.

"ALPHA Ltd."

May 12, 1991
6:35 PM

	CUPRINS	Pag.
Prefata		
Introducere		4
ASAMBLORUL		6
- Rutinele de restart si tabele		6
- Rutinele tastaturii		11
- Rutinele de difuzor		19
- Rutinele de tratare a casetei		24
- Rutinele de tratare a ecranului si imprimantei		49
- Rutinele executivului		64
- Programul BASIC si interpretarea comenzilor		116
- Evaluarea expresiei		172
- Rutinele aritmetice		222
- Dispozitivul de calcul in virgulă mobilă		259
Anexa		
- Programele BASIC pentru seriile principale (SIN X, EXP X, LN X & ATN X)		302
- Algoritmul 'DRAW' (desenare)		308
- Algoritmul 'CIRCLE' (cerc)		308
- Notă asupra întregilor mici si asupra lui -65536		310
Indexul rutinelor		312

INTRODUCERE

PREFATA

Spectrum ZX Sinclair este un succesos merituos al lui ZX 81 care la rindul inlocuieste ZX 80.

Spectrum are un program monitor de 16 kocteti. Acest program a fost dezvoltat direct din programul monitor de 4 kocteti al lui ZX 80, desi aici snt atft de multe trăsături noi încît diferentele prevalează asupra asemănarilor.

Am avut o deosebită plăcere elaborînd această carte. Am învățat multe lucrînd asupra tehnicilor de programare în cod masină Z80 și acum simțim că am reușit să vămură "secretele Spectrum".

Am vrea să multumim:

- Familiilor noastre
- Lui Alfred Milgram, editorul nostru care ne-a fost de mare ajutor
- Lui Philip Mitchell, ale cărui notări asupra formării casei tei ne-au adus multe informații
- Lui Crive Sinclair și echipei lui și lui Sinclair Research Ltd., care au realizat o masină atît de folositoare și încitantă.

Januarie 1983.

INTRODUCERE

Programul monitor de 16 K al lui Spectrum este un program scris în cod masină Z80 complex. Structura sa generală este foarte clară în împărțirea sa în trei mari părți:

- a. Rutine de intrare/iesire
- b. Interpretorul BASIC
- c. Tratarea expresiei

Totuși aceste blocuri sînt prea întinse pentru a fi descurcate ușor și de aceea în această carte programul monitor este discutat în zece părți.

Fiecare din aceste părți va fi acum 'echitată'.

Rutinele de restart și tabele

La începutul programului monitor sînt diferitele rutine de 'restart' care sînt apelate cu ajutorul instrucțiunilor pe un singur octet 'RST'. Toate 'reporțiile' sînt folosite. De exemplu, 'restart 0000' este folosit pentru raportările sintaxei sau a erorilor de execuție.

Tabelele din această parte a programului monitor conțin formate extinse ale simbolurilor și 'codurile tastelor'.

Rutinele tastaturii

Tastatura este baleiată la fiecare a 1/50-a parte dintr-o secundă (model european) și rutinele tastaturii redau codul caracterului cerut. Toate tastele claviaturii autorepetă dacă sînt apăsată și acest lucru este luat în considerare de către rutinele tastaturii.

Rutinele de difuzor

Spectrum are încorporat un singur difuzor și o notă este produsă prin folosirea repetată a instrucțiunii corespunzătoare 'OUT'. În rutina controloare se are mare grijă să se asigure că nota este păstrată la o înălțime dată pe toată 'durata' sa.

Rutinele de tratare a casetei

A fost o foarte nefericită trăsătură a lui ZX 81 că așa de puțin din programul monitor pentru această mașină a fost consacrat tratării casetei.

În orice caz în Spectrum există un bloc extins al codului și acum standardul înalt de tratare a casetei este una din cele mai reușite trăsături ale mașinii.

Programele BASIC sau blocurile de informații lucrează eficient în același fel, avînd un bloc 'header' (17 octeți) care este salvat mai întâi. Acest 'header' descrie 'blocul de informații' care este salvat după el.

Un dezavantaj al acestui sistem este faptul că nu este posibilă realizarea de programe cu o oarecare 'siguranță'.

Rutinele de tratare a ecranului și imprimantei

Toate rutinele de intrare/iesire rămase ale lui Spectrum sînt 'orientate' direct în 'canalul' și zona sirului de informații.

În standardul Spectrum 'intrarea' este posibilă doar de la tastatură, dar 'iesirea' poate fi direcționată la imprimantă, în partea superioară a ecranului TV sau în partea inferioară a ecranului TV.

Principala rutină 'de iesire' din această parte a programului monitor este EDITOR, care permite utilizatorului introducerea de caractere în partea inferioară a ecranului TV.

Rutina PRINT-OUT este o rutină destul de înceată care este folosită pentru 'toate posibilitățile'. De exemplu: adăugarea unui singur octet în 'spatiul ecranului' implică luarea în considerare a stării prezente a lui OVER și INVERSE în fiecare caz.

Rutinele executivului

În această parte a programului monitor trebuie găsită procedura INITIALISATION și 'bucla principală de execuție' a interpretorului BASIC.

În Spectrum linia BASIC returnată de către EDITOR este verificată

din punct de vedere al corectitudinii sintaxei si apoi salvată în spatiul programului, dacă a fost o linie care a început cu un număr de linie.

Linia BASIC și interpretarea comenzii

Această parte a programului monitor consideră linia BASIC ca un set de expresii, fiecare expresie la rândul ei începând cu o comandă particulară. Pentru fiecare comandă există o 'rutina de comandă', iar interpretarea este efectuată de execuția codului mașină în 'rutina de comandă' potrivită.

Evaluarea expresiei

Spectrum are cea mai cuprinzătoare evaluare de expresie, disponibilă pentru o gamă largă de tipuri de variabile, funcții, operații. Tratarea sirurilor este, în mod particular, bine rezolvată. Toate sirurile simple sînt rezolvate 'dinamic', iar rezultatele cunoscute sînt 'recuperate' odată ce sînt redundante. Aceasta înseamnă că nu este de făcut nici o 'curățenie'.

Rutinele aritmetice

Spectrum are două forme pentru numere. Valorile din domeniul -65535 + $+65535$ sînt în formă 'integrală' sau 'scurtă' pe cînd toate celelalte numere sînt în formă de 5 biți cu virgulă mobilă.

Versiunea de față a monitorului are în această privință, din păcate, două inconveniente:

i. Există un inconvenient la împărțire prin faptul că se pierde al 34-lea bit.

ii. Valoarea lui -65536 este cîteodată pusă în formă 'scurtă' iar în alte dăți în 'virgulă mobilă', ceea ce conduce la necazuri.

Calcularea virgulei mobile

CALCULATORUL din Spectrum tratează numerele și sirurile, iar operațiile lui sînt specificate prin 'litere'. De aceea se poate considera că există un limbaj de operare intern în CALCULATOR.

Această parte a programului monitor conține rutine pentru toate funcțiile matematice. Aproximațiile la $\sin X$, $\exp X$, $\ln X$ și respectiv $\arctan X$ sînt obținute dezvoltînd polinoamele Chebyshev, mai multe detalii fiind date în appendix.

În totalitate, programul monitor de 16K, oferă o gamă extrem de largă de diferite comenzi și funcții BASIC. Cu toate acestea, programatorii întotdeauna duc lipsă de 'posibilitate' și din acest motiv programul este scris mai degrabă pentru 'scurttime decît pentru 'viteză'.

Rutinele dezasamblate

RUTINELE RESTART si TABELELE

THE START (START)

Intreruperea mascabila este dezactivata si registrul pereche DE este fixat sa pastreze 'virful RAM-ului posibil'.

0000 START	JF		Dezactiveaza 'Intrerupere
	KOR	A,+00	claviatură' pentru 'start'
	LD	DE,+FFFF	(dar +FF pentru 'NEW').
	JP	11CB,START/NEW	Virful RAM-ului posibil.
			Salt in fata.

THE 'ERROR' RESTART

Pointerul de eroare este facut sa indice pozitia erorii.

0008 ERROR-1	LD	HL,(CH-ADD)	Adresa ajunsa de interpretor
	LD	(X-PTR),HL	este copiată la pointerul de
	JR	0053,ERROR-2	eroare înainte de a continua.

THE 'PRINT A CHARACTER' RESTART ('TIPARESTE UN CHARACTER')

Registrul A pastreaza codul caracterului care este tipărit.

0010 PRINT-A-1	JF	5F2,PRINT-A-2	Salt in fata imediat.
	DEFB	FF,+FF,+FF,+FF,+FF	Locatii nefolosite.

THE 'COLLECT CHARACTER' RESTART ('DULEGE CHARACTER')

Sint scoase continuturile locatiilor adresate curent de CH-ADD. Se face o intoarcere daca valoarea reprezinta un caracter ce se poate tipări, in caz contrar CH-ADD este incrementat si testele sint repetate.

0018 GET-CHAR	LD	HL,(CH-ADD)	Scoate valoarea care este
	LD	A,(HL)	adresata de CH-ADD.
0010 TEST-CHAR	CALL	007D,SKIP-OVER	Constata daca caracterul se
	RET		poate tipări. Daca este
			asa, intoarcere.

THE 'COLLECT NEXT CHARACTER' RESTART ('DULEGE URMATORUL CHARACTER')

Cum o linie BASIC este interpretabila, aceasta rutina este chemata frecvent de-a lungul liniei.

0020 NEXT-CHAR	CALL	0074,CH-ADD+1	CH-ADD trebuie sa fie
	JR	001C,TEST-CHAR	incrementat.
	DEFB	+FF,+FF,+FF	Salt inapoi pentru a testa
			noua valoare.
			Locatii nefolosite.

THE 'CALCULATOR' RESTART (RESTART 'CALCULATOR')

Virgula mobila este introdusa la 335B.

0028 FP-CALC	JP	335B,CALCULATE	Salt neconditionat in fata.
	DEFB	+FF,+FF,+FF,+FF,+FF	Locatii nefolosite.

THE 'MAKE 30 SPACES' RESTART ('CREEAZA SPATII 30')

Aceasta rutina creeaza locatii libere in spatiul de lucru. Numarul locatiilor este determinat de continuturile curente ale registrului pereche BC.

0030 BC-SPACES	PUSH	BC	Salveaza 'numarul'.
	LD	HL,(WORKSP)	Scoate adresa prezenta a
	PUSH	HL	inceputului spatiului de
	JP	169E,RESERVE	lucru si salveaz-o inainte de
			a merge mai departe.

THE 'MASKABLE INTERRUPT' ROUTINE ('INTRERUPERE MASCABILA')

Ceasul de timp real este incrementat, iar testatura cercetata pentru a sesiza cind apare o intrerupere mascabila.

0039 MASK-INT	PUSH	AF	Salveaza valorile curente
	PUSH	HL	mentinute in aceste registre
	LD	HL,(FRAMES)	Cei mai putin semnificativi 2
	LD	(FRAMES),HL	biti ai fiecărei 20 ms (U.K.)
	LD	A,H	Cel mai semnificativ bit al
	OR	L	numărătorului de cadre este
	JR	NZ,004B,KEY-INT	incrementat numai cind
	INC	(FRAMES-3)	valoarea celor doi biti mai

0043 KEY-INT	PUSH BC	putin semnificativi este zero.
	PUSH DE	Salvează valorile curente păstrate în aceste registre
	CALL 02BF,KEYBOARD	Acua cercetează tastatura.
	POP DE	Se refac valorile.
	POP BC	
	POP HL	
	POP AF	
	EI	Intreruperea mascabilă este permisă înainte de întoarcere.
	RET	

THE 'ERROR-2' ROUTINE

Adresa de întoarcere la interpretor indică la 'DEFB' că, semnifică ce eroare a apărut. Acest 'DEFB' este scos și transferat la ERR-NR. Stiva masinii este stearsă înainte de a sări mai departe pentru a șterge stiva calculatorului.

0053 ERROR-2	POP HL	Adresa din stivă semnifică codul erorii.
	LD L,(HL)	Ei este transferat la ERR-NR.
0055 ERROR-3	LD (ERR-NR),L	Masina este stearsă înainte de a iesi prin SET-STK.
	LD SP,(ERR-SP)	
	JP 16C5,SET-STK	
	DEFB +FF,+FF,+FF,+FF	Locatii nefolosite.
	DEFB +FF,+FF,+FF	

THE 'NON-MASKABLE INTERRUPT' ROUTINE (RUTINA 'INTRERUPERE NEHASCABILA')

Această rutină nu este folosită în Spectrum-ul standard, dar urmărind activarea liniei NMI codul permite apariția unui reset sistem. Variabila sistem la SC80, numită aici NMIADD, trebuie să aibă valoarea zero pentru a apare un reset.

0066 RESET	PUSH AF	Salvează valorile curente păstrate în aceste registre.
	PUSH HL	Cei doi biti ai NMIADD trebuie să fie ambii zero pentru ca reset-ul să apară.
	LD HL,(NMIADD)	Notă: Acesta ar fi trebuit să fie 'JR Z'!
	LD A,H	Salt la START.
	OR L	Se refac valorile curente la aceste registre și întoarcere.
	JR NZ,0079,NO-RESET	
	JP (HL)	
0070 NO-RESET	POP HL	
	POP AF	
	RET H	

THE 'CH-ADD+1' SUBROUTINE

Adresa păstrată în CH-ADD este scoasă, incrementată și reintrodusă. Este scos conținutul locației adresată acua de CH-ADD. Punctele de intrare TEMP-PTR1 și TEMP-PTR2 sînt folosite pentru a fixa CH-ADD pentru o perioadă temporară.

0074 CH-ADD+1	LD HL,(CH-ADD)	Scoate adresa.
0777 TEMP-PTR1	INC HL	Se incrementează pointerul.
0078 TEMP-PTR2	LD (CH-ADD),HL	Fixează CH-ADD.
	LD A,(HL)	Scoate valoarea adresată și apoi întoarcere.
	RET	

THE 'SKIP-OVER' SUBROUTINE (SUBROUTINA 'SARE PESTE')

Valoarea adusă subrutinei în registrul A este testată pentru a vedea dacă se poate tipări. Diferite coduri speciale conduc ca HL să fie incrementat o dată, sau de două ori, și CH-ADD corectat corespunzător.

007D SKIP-OVER	CP +21	Întoarcere cu indicatorul de stare carry resetat.
	RET NC	Întoarcere dacă s-a ajuns la capătul liniei.
	CP +0D	Întoarcere cu codurile +00 la +0F dar cu indicatorul de stare fixat.
	RET Z	
	CP +10	Întoarcere cu codurile +18 la +20 din nou cu indicatorul carry fixat.
	RET C	Incrementează încă o dată Salt în față cu codurile +10 la +15 (de la INK la OVER).
	CP +18	Incrementează încă o dată (AT&TAB).
	CCF	Întoarcere cu indicatorul carry fixat și cu CH-ADD păstrînd adresa potrivită.
	RET C	
	INC HL	
	CP +16	
	JR C,0090,SKIPS	
	INC HL	
0090 SKIPS	SCF	
	LD (CH-ADD),HL	
	RET	

THE TOKEN TABLE (TABELUL DE SIMBOLURI)

Toate simbolurile folosite de Spectrum sînt dezvoltări avînd ca referință acest tabel. Ultimul cod al fiecărui simbol 'inversat' are bitul său 7 fixat.

0095	BF	52	4E	C4	49	4E	4B	45	?	R	N	D	I	N	K	E
009B	59	A4	50	C9	46	CE	50	4F	Y	J	P	F	N	P	A	O
00A5	49	4E	D4	B3	43	52	45	45	I	N	T	A	R	R	E	E
00AB	4E	A4	41	54	54	D2	41	D4	I	N	T	A	R	R	E	E
00B5	54	41	C2	56	41	4C	A5	43	T	A	B	V	A	L	S	C
00B9	4F	44	C5	56	41	CC	4C	43	Ø	S	N	I	A	C	N	O
00C5	CE	53	49	CE	43	4F	D3	54	N	A	S	I	N	L	O	A
00CD	01	CE	41	53	CE	41	43	D3	A	I	N	S	L	O	A	E
00D5	41	54	CE	4C	CE	45	50	D0	A	I	N	S	L	O	A	E
00DD	49	4E	D4	53	51	D2	53	47	I	N	S	L	O	A	E	R
00E5	CE	41	42	D3	50	45	45	C3	I	N	S	L	O	A	E	R
00EB	49	CE	55	53	D2	53	50	52	I	N	S	L	O	A	E	R
00F5	4F	43	48	B2	A4	4E	4F	D4	S	C	I	N	S	L	O	A
00FB	42	49	CE	4F	D2	41	4E	C4	S	C	I	N	S	L	O	A
0105	3C	BD	5E	FD	3C	DE	4C	49	<	-	>	-	>	-	>	-
010D	4E	C5	54	08	43	CE	54	CF	N	S	L	O	A	E	R	E
0115	53	54	45	D0	44	45	46	20	S	F	T	E	C	T	O	R
011B	46	0E	43	41	D4	46	4F	52	F	M	A	T	O	R	E	N
0125	4B	41	D4	4D	4F	56	C5	45	R	A	T	O	R	E	N	E
012B	52	41	53	C5	4F	50	45	4E	R	A	T	O	R	E	N	E
0135	20	A3	43	AC	4F	53	45	20	A	T	O	R	E	N	E	E
013B	A3	4B	45	52	47	C5	54	45	A	T	O	R	E	N	E	E
0145	52	49	46	D9	42	45	45	D0	R	C	I	P	R	A	H	N
014D	43	49	52	43	4C	C5	49	4E	R	C	I	P	R	A	H	N
0155	CE	50	41	50	45	B2	46	4C	C	A	T	O	R	E	N	E
015D	41	53	C8	42	52	49	47	48	A	T	O	R	E	N	E	E
0165	D4	49	4E	56	45	52	53	C5	T	O	R	E	N	E	E	E
016B	4F	56	45	D2	4F	55	D1	4C	O	V	E	R	E	N	E	E

0175	50	52	49	4E	D4	4C	4C	49	P	R	I	N	T	L	L	I
017B	53	54	53	54	4F	50	52	45	S	A	T	A	T	A	R	E
0185	41	C4	44	41	54	C1	52	45	A	S	T	A	T	A	R	E
018B	53	54	4F	52	C5	4E	45	57	S	A	T	A	T	A	R	E
0195	42	4F	52	44	45	B2	43	4F	S	A	T	A	T	A	R	E
019B	4E	54	49	4E	55	C5	44	49	S	A	T	A	T	A	R	E
01A5	CD	52	45	CF	46	4F	D2	47	N	O	N	T	O	P	I	O
01AB	4F	20	54	CF	47	4F	20	53	O	N	T	O	P	I	O	P
01B5	55	C2	49	4E	50	55	D4	4C	O	N	T	O	P	I	O	P
01BB	4F	41	C4	4C	49	53	D4	4C	O	N	T	O	P	I	O	P
01C5	45	D4	50	41	55	53	C5	4E	E	R	R	A	N	S	A	C
01CB	45	58	D4	50	4F	4B	C5	50	E	R	R	A	N	S	A	C
01D5	52	49	4E	D4	50	4C	4F	D4	R	A	N	S	A	C	R	O
01DD	52	55	CE	53	41	56	C5	52	R	A	N	S	A	C	R	O
01E5	41	4E	44	4F	49	49	5A	C5	A	N	S	A	C	R	O	P
01EB	49	C6	43	4C	D3	44	52	41	I	N	S	A	C	R	O	P
01F5	B7	43	4C	45	41	B2	52	45	M	T	C	L	E	N	A	C
01FB	54	55	52	CE	43	4F	50	D9	T	C	L	E	N	A	C	R

THE KEY TABLES (TABELE REFERITOARE LA TASTE)

Există șase tabele diferite. Codul final al caracterului obținut depinde de tasta care a fost apăsată și de 'modul' folosit.

(*) Tabelul principal de taste - Modul L și CAPS SHIFT

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

() Modul extins. Taste literale si nesifiate.

0220	E3 C4 E0 E4	READ	BIN	LPRINT	DATA
0230	B4 BC BD BE	TAN	SON	ABS	SCR
0234	AF B0 B1 C0	CODE	VAL	LEN	USR
0238	A7 A6 BE AD	PI	INKEYS	PEEK	TAB
023C	B2 BA EB A5	BIN	INT	RESTORE	RND
0240	C2 E1 B3 B9	CHRS	LIST	COS	EXP
0244	D1 B8	STR	LN		

() Modul extins. Taste literale si sifiate.

0246	7E DC DA 5C	/\	BRIGHT	PAPER	\
024A	B7 7B 7D DB	ATN	()	CIRCLE
024E	BF AE AA AB	IN	VAL*	SCREEN*	ATTR
0252	BD BE DF 7F	INVERSE	OVER	OUT	@
0256	B5 B6 7C 95	ASN	VERIFY		MERGE
025A	5D DB B6 D9	J	FLASH	ACS	INK
025E	3B B7	(BEEP		

() Coduri de control. Taste digitale si CAPS SHIFT.

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

() Coduri simbol. Taste literale si simbolul shift.

026A	E2 2A 3F 6B	STOP	*	?	STEP
026E	C8 CC CB 5E	>=	TO	THEN	
0272	AC 2B 2B 3D	AT	-	+	=
0276	2E 2C 3B 22	.	{		="
027A	C7 3C C3 3E	<=	}	NOT	>
027E	C5 2F C9 60	OR	/	<>	E
0282	C6 3A	AND			

() Modul extins. Taste digitale si simbolul shift.

0284	B0 CE A8 CA	FORMAT	DEF FN	FN	LINE
0288	B3 D4 D1 D2	OPEN	CLOSE	MOVE	ERASE
028C	A9 CF	POINT	CAT		

RUTINELE TASTATURII

THE 'KEYBOARD SCANNING' SUBROUTINE (SUBROUTINA 'CERCETARE TASTATURA')
 Această subrutină foarte importantă este apelată atât de principala subrutină a tastaturii cât și de rutina INKEY\$ (în SCANNING).
 În toate cazurile registrul E este reschimbăat cu o valoare în domeniul de la +00 la +27, valoarea fiind diferită pentru fiecare dintre cele 40 de chei ale tastaturii, sau valoarea +FF, pentru nici o cheie.

Registrul D este neschimbăat cu o valoare ce indică care cheie shift singulară este apăsată. Dacă abele chei shift sînt apăstate atunci registrele D și E sînt neschimbăate cu valorile pentru CAPS SHIFT și respectiv pentru cheia SYMBOL SHIFT.
 Dacă nu s-a apăsat nici o tastă, atunci registrul pereche DE va păstra neschimbăat +FFFF.
 Indicatorul zero este resetat dacă mai mult de două taste sînt apăstate sau dacă nici una din tastele unei perechi de taste nu este o tastă shift.

028E KEY-SCAN	LD	L,+2F	Valoarea inițială a cheii pentru fiecare linie va fi +2F, +2E..., +28 (8 linii).
	LD	DE,+FFFF	Inițializează DE pentru 'nici o tastă'.
	LD	BC,+FEFE	C = port de adrese, B = numărator

Acum intră într-o buclă. Se fac opt treceri, la fiecare pas tastă avînd o valoare inițială diferită și se va cerceta o linie diferită de 5 taste. (Prima linie este CAPS SHIFT, Z, X, C, V).

0296 KEY-LINE	IN	A,(C)	Citeste de la portul specificat.
	CPL		O tastă apăsată în linie va fixa bitul său respectiv (de la bit 0 - tastă exterioară, la bit 4 - tastă internă).
	AND	+1F	Salt în față dacă nici una din cele 5 taste nu a fost apăsată.
	JR	Z,02A8,KEY-DONE	Bitii tastei merg în registrul H în timp ce este adusă valoarea inițială a tastei.
	LD	H,A	
	LD	A,L	
029F KEY-3KEYS	INC	D	Dacă sînt apăstate trei taste pe tastatură, atunci registrul D nu va mai mentine +FF; dacă aceasta se întimplă, întoarcere.
	RET	NZ	Frecvență se scade '8' din valoarea prezentă a tastei pînă se găsește un key-bit. Copiază oricare primă valoare de tastă în registrul D. Pasează nouă valoare a tastei în registrul E.
02A1 KEY-BIT	SUB	+08	Dacă există în această linie o a 2-a sau o a 3-a posibilă tastă apăsată, atunci salt înapoi.
	SRL	H	Linia a fost cercetată astfel că valoarea inițială a cheii este redusă pentru pasul următor.
	JR	NC,02A1,KEY-BITS	Număratorului este shiftat și săritura are loc dacă mai sînt linii de cercetat.
	LD	D,E	
	LD	E,A	
	JR	NZ,029F,KEY.3KEYS	
02A8 KEY-DONE	DEC	L	
	RLC	B	
	JR	C,0296,KEY-LINE	

Acum se mai fac patru taste.

LD	A,D	Acceptă orice valoare de tastă care totuși mentine în registrul D, +FF. Adică o singură tastă apăsată sau 'nici o tastă'.
INC	A	
RET	Z	
CP	+28	Acceptă valoarea tastei pentru o pereche de taste
RET	Z	

```

CP      +19
RET     Z

LD      A,E
LD      E,D
LD      D,A
CP      +18

RET

```

dacă tasta 'D' este CAPS SHIFT. Acceptă valoarea tastei pentru o pereche de taste dacă tasta 'D' este SYMBOL SHIFT. Este totuși posibil pentru tasta 'E' a unei perechi să fie SYMBOL SHIFT - astfel aceasta trebuie luată în considerare. Întoarcere cu indicatorul zero fixat dacă a fost SYMBOL SHIFT și 'o altă tastă'; altfel este reset.

THE 'KEYBOARD' SUBROUTINE (TASTATURA)

Această subrutină este chemată în fiecare ocazie în care apare o întrerupere mascabilă. În operații normale aceasta se întâmplă o dată la fiecare 20 ms. Scopul acestei subrutine este de a cerceta tastatura și de a codifica valoarea tastei. Acest cod produs - dacă starea de 'repetare' o permite - va fi trecut la variabila sistem LAST-K. Când un cod este pus în această variabilă sistem, bitul 5 al-FLAG8 este fixat să arate că o 'nouă' tastă a fost apăsată.

```

02BF KEYBOARD      CALL 028E,KEY-SCAN
RET                NZ

```

Se aduce o valoare de tastă în registrul pereche DE cu întoarcere imediată dacă indicatorul zero este resetat.

De aici încolo se va folosi un sistem dublu de 'variabile sistem KSTATE' (KSTATE0 - KSTATE3 și KSTATE4 - KSTATE7).

Cele două seturi tin seama de detecția unei noi taste care este apăsată (folosind un singur set) încă în timpul 'perioadei de repetare' a tastei ce a fost anterior apăsată (detalii în al 2-lea set).

Un set va deveni liber să testeze o nouă tastă numai dacă este apăsată aproximativ 1/10 părți dintr-o secundă, adică 5 cereri la tastatură.

```

02C6 K-ST-LOOP     LD      HL,KSTATE0
BIT          7,(HL)
JR          NZ,02D1,K-CH-SET

INC          HL
DEC          (HL)
DEC          HL
JR          NZ,02D1,K-CH-SET
LD          (HL),+FF

```

Începe cu KSTATE0. Salt mai departe dacă un set este gol, adică KSTATE0/4 păstrează +FF. Totuși, dacă setul nu este gol, decrementează al său 'numărător de 5 cereri' și când ajunge zero, se semnalizează setul ca fiind gol.

După ce s-a luat în considerare primul set, se schimbă indicatorul și se ia în considerare al 2-lea set.

```

02D1 K-CH-SET     LD      A,L
LD          HL,+KSTATE4
CP          L
JR          NZ,02C6,K-ST-LOOP

```

Scoate bitul cel mai puțin semnificativ al adresei și salt înapoi dacă al 2-lea set este încă luat în considerare.

Întoarcere acum, dacă valoarea tastei indică 'nici o tastă' sau numai o tastă shift.

```

CALL 031E,K-TEST
RET  NC

```

Se fac testele necesare și se întoarcere dacă este nevoie. Se schimbă de asemenea valoarea tastei cu un 'cod principal'.

O lovitură de tastă ce se repetă (menținută apăsată) este separată de o nouă lovitură de tastă.

```

LD          HL,+KSTATE0
CP          (HL)
JR          Z,0310,K-REPEAT

EX          DE,HL

LD          HL,+KSTATE4
CP          (HL)
JR          Z,0310,K-REPEAT

```

Priveste întâi la KSTATE0. Salt mai departe dacă codurile care se întrec indică o repetare. Salvează adresa lui KSTATE0. Acum se verifică KSTATE4. Salt mai departe dacă codurile ce se întrec indică

o repetare.

Bar o nouă tastă nu va fi acceptată decât dacă unul din seturile de variabile sistem KSTATE este 'gol'.

BIT	7, (HL)	Se consideră al 2-lea set.
JR	NZ, 02F1, K-NEW	Salt mai departe dacă este 'gol'.
EX	DE, HL	Acum se consideră primul set.
BIT	7, (HL)	Continuă dacă setul este 'gol', dacă nu părăsește
RET	Z	subrutina KEYBOARD (tastatură).

Noua tastă este acceptată. Înainte ca variabila sistem LAST-K să poată fi umplută, variabilele sistem KSTATE, ale setului folosit, trebuie să fie initializate pentru a trata orice repetări, iar codul tastei trebuie decodat.

02F1 K-NEW	LD	E, A	Codul este pasat la registrul E și la KSTATE0/4.
	LD	(HL), A	
	INC	HL	'Numărătorul' de 5 cereri pentru acest set este resetat la '5'.
	LD	(HL), +05	
	INC	HL	A treia variabilă sistem a setului menține valoarea REPDEL (normal 0,7 sec).
	LD	A, (REPDEL)	Indică la KSTATE3/7.
	LD	(HL), A	
	INC	HL	

Decodarea unui 'cod principal' depinde de starea prezentă a lui MODE, bitul 3 de la FLAG și de 'bitul de shift'.

LD	C, (MODE)	Scoate MODE.
LD	D, (FLAG)	Scoate FLAG.
PUSH	HL	Salvează pointerul în timp ce 'codul principal' este decodat.
CALL	0333, K-DECODE	Valoarea finală a codului este salvată în KSTATE3/7, de unde el este cules în cazul unei repetări.
POP	HL	
LD	(HL), A	

Următoarele trei linii de instrucții sînt comune pentru tratarea atât a 'tastelor noi' cît și a 'tastelor repetabile'.

0308 K-END	LD	(LAST-K), A	Se introduce valoarea finală a codului în LAST-K.
	SET	5, (FLAG)	Întoarcere finală.
	RET		

THE 'REPEATING KEY' SUBROUTINE (REPETARE TASTA)
O tastă se va 'repetă' într-o primă fază după perioada de ștergere - REPDEL (normal 0,7 sec.) iar în fazele următoare după perioada de ștergere - REPER (normal 0,1 sec.).

0310 K-REPEAT	INC	HL	Se indică la numărătorul de '5 cereri' setul care se va folosi și se resetează la '5'.
	LD	(HL), +05	
	INC	HL	Se indică la a 3-a variabilă de sistem valoarea REPDEL/REPER și se decrementează.
	DEC	(HL)	
	RET	NZ	Se iese din rutina KEYBOARD (tastatură) dacă nu a trecut perioada de ștergere.
	LD	A, (REPER)	Totuși, o dată ce a trecut perioada de ștergere, pentru următoarea repetare este REPER.
	LD	(HL), A	Repetarea a fost acceptată astfel că valoarea finală a codului este scoasă de KSTATE3/7 și pasată la K-END.
	INC	HL	
	LD	A, (HL)	
	JR	0308, K-END	

THE 'K-TEST' SUBROUTINE
Se testează valoarea tastei și se face o întoarcere dacă este 'nici o tastă' sau 'numai shift'; în caz contrar este găsit 'codul principal' al acestei taste.

031E K-TEST	LD	B, D	Copiază bitul shift.
	LD	D, +00	Sterge registrul D pentru mai târziu.
	LD	A, E	Transferă numărul tastei.
	CP	+27	Acum se execută întoarcere dacă tasta a fost numai 'CAPS SHIFT' sau 'nici o tastă'.
	RET	NC	Salt mai departe în afară de cazul că tasta 'E' a fost SYMBOL SHIFT și o altă tastă.
	CP	+18	Totusi se acceptă SYMBOL SHIFT și o altă tastă; întoarcere numai cu SYMBOL SHIFT.
	JR	NZ, 032C, K-MAIN	
	BIT	7, B	
	RET	NZ	

'Codul principal' este găsit prin indexarea tabelului principal de taste.

032C K-MAIN	LD	HL, 0205	Adresa de bază a tabelului.
	ADD	HL, DE	Se indexează în tabel și se aduce 'codul principal'.
	LD	A, (HL)	Seanalizează 'lovitură tastă validă' înainte de întoarcere.
	BCF		
	RET		

THE KEYBOARD DECODING SUBROUTINE (DECODARE TASTATURA)

Această subrutină este începută cu 'codul principal' în registrul E, valoarea lui FLAGS în registrul D, valoarea lui MODE în registrul C și 'bitul shift' în registrul B.

Luând în considerare aceste patru valori și referindu-ne, dacă este necesar, la cele 6 tabele de taste, se va produce 'codul final'. Acesta este întors în registrul A.

0333 K-DECODE	LD	A, E	Copiază 'codul principal'.
	CP	+3A	Salt mai departe dacă este considerată o tastă digitală; de asemenea SPACE, ENTER și ambele shiftate.
	JR	C, 0367, K-DIGIT	Decrementează valoarea MODE.
	DEC	C	Salt mai departe pentru modurile 'K', 'L', 'C' și 'E'.
	JP	M, 034F, K-KLC-LET	
	JR	Z, 0341, K-E-LET	

Rămâne doar modul 'grafic' iar 'codul final' pentru tastele literale în modul grafic este calculat din 'codul principal'.

ADD	A, +4F	Adună offset-ul.
RET		Întoarcere cu 'codul final'.

În continuare sînt luate în considerare tastele literale în modul extins.

0341 K-E-LET	LD	HL, +01E9	Adresa de bază pentru tabelul 'b'.
	INC	B	Dacă nici o tastă shift nu a fost apăsată, salt mai departe pentru a folosi acest tabel.
	JR	Z, 034A, K-LOOK-UP	În caz contrar se folosește adresa de bază pentru tabelul 'c'.
	LD	HL, 0205	

Tabelele de taste 'b-f' sînt toate aservite de următoarea rutină de urmărire. În toate cazurile este servit și întors un 'cod final'.

034A L-LOOK-UP	LD	B, +00	Se șterge registrul B.
	ADD	LD, DE	Se indexează tabelul cerut și se aduce 'codul final'.
	LD	A, (HL)	Apoi întoarcere.
	RET		

Se vor considera acum tastele literale în modurile 'K', 'L' sau 'C'. Dar mai întâi trebuie tratate cu codurile speciale SYMBOL SHIFT.

034F K-KLC-LET	LD	HL, +0229	Adresa de bază pentru tabelul 'e'.
	BIT	0, B	Salt înapoi dacă se folosește tasta SYMBOL SHIFT și o tastă literală.
	JR	Z, 034A, K-LOOK-UP	Salt mai departe dacă esti curent în modul 'K'.
	BIT	3, B	Dacă este fixat CAPS LOCK,
	JR	Z, 0364, K-TOKENS	
	BIT	3, (FLAG62)	

RET	NZ	atunci întoarcere cu 'codul principal'.
INC	B	Întoarcere în aceeași manieră
RET	NZ	dacă CAPS SHIFT a fost apăsat.
ADD	A,+20	Dacă sînt cerute coduri mici
RET		atunci trebuie să se adune +20 la 'codul principal' pentru a da 'codul final' corect.

Valorile de 'cod final' pentru simboluri sînt găsite adăugîndu +A5 la 'codul principal'.

0364 K-TOKENS	ADD	A,+A5	Se adună offset-ul cerut și
	RET		întoarcere.

În continuare se consideră tastele digitale și SPACE, ENTER și amîndouă shiftate.

0367 K-D181T	CP	+30	Continuă numai cu tastele digitale,
	RET	C	adică întoarcere cu SPACE (+20), ENTER (+0D) și amîndouă shiftate (+0E).
	DEC	C	Acum separă tastele digitale în trei grupe, în corespondență cu modul de lucru.
	JP	N,039D,K-KLC-DGT	Sari cu modurile 'K', 'L' și 'C'
	JR	NZ,0389,K-GRA-DGT	și cu modul 'G' de asemenea. Continuă cu modul 'E'.
	LD	HL,+0254	Adresa de bază a tabelului 'f'.
	BIT	S,B	Folosește acest tabel pentru SYMBOL SHIFT și o tastă digitală în modul extins.
	JR	Z,034A,K-LOOK-UP	Sari mai departe cu tastele digitale '8' și '9'.
	CP	+38	
	JR	NC,0382,K-8-&-9	

Tastele digitale de la '0' la '7' în modul extins sînt pentru a da ori un 'cod culoare hîrtie' ori un 'cod culoare cerneală', depinzînd de modul de folosire a lui CAPS SHIFT.

	SUB	+20	Se reduce domeniul de la +30 la +37, oferind un domeniu de la +10 la +17.
	INC	B	Întoarcere cu acest 'cod de culoare hîrtie' dacă CAPS SHIFT nu a fost folosit.
	RET	Z	Dar dacă el este, domeniul în schimb este de la +18 la +1f - indicînd un 'cod culoare cerneală'.
	ADD	A,+08	
	RET		

Tastele digitale '8' și '9' sînt pentru a da codurile 'BRIGHT' (LUMINOS) și respectiv 'FLASH'.

0382 K-8-&-9	SUB	+36	+38 și +39 ajung +02 și respectiv +03.
	INC	B	Întoarcere cu aceste coduri dacă CAPS SHIFT nu a fost folosit. (Acestea sînt codurile 'BRIGHT').
	RET	Z	Scădere 'Z' dacă CAPS SHIFT este folosit, obținînd +00 și respectiv +01 (ca și coduri 'FLASH').
	ADD	A,+FE	
	RET		

Tastele digitale în modul grafic sînt pentru a furniza blocul de caractere grafice (+80 și +8F), codul GRAFICS (+0F) și codul DELETE (+0C) - STERGERE.

0389 K-GRA-DGT	LD	HL,+0230	Adresa de bază a tabelului 'd'.
	CP	+39	Folosește acest tabel direct pentru ambele taste digitale '9' pentru a GRAFICS și tastă digitală
	JR	Z,034A,K-LOOK-UP	
	CP	+30	
	JP	Z,034A,K-LOOK-UP	

AND	+07	'0' pentru a furniza DELETE.
ADD	A,+80	Pentru tastele de la '1' la '8' se face domeniul +80 +87.
INC	8	Intoarce cu o valoare din acest domeniu dacă nici una din tastele SHIFT nu a fost apăsată.
RET	Z	Dar dacă s-a shiftat, fă domeniul între +88 și +8F.
XOR	+0F	
RET		

In final, considerăm tastele digitale în codurile 'K', 'L' și respectiv 'C'.

039D K-KLC-DBT	INC	B	Intoarce directă dacă nu s-a folosit nici o tastă SHIFT (codurile finale de la +30 la +39).
	RET	Z	Folosește tabelul 'd' dacă tasta CAPS SHIFT a fost de asemenea apăsată.
	BIT	5,B	
	LD	HL,+0230	
	JR	NZ,034A,K-LOOK-UP	

Acum pot fi găsite codurile pentru diferite taste digitale și SYMBOL SHIFT.

	SUB	+10	Redu domeniul pentru a furniza +20 și +29.
	CP	+22	Separă caracterul '@' de celelalte.
	JR	Z,03B2,K-@-CHAR	Caracterul '-' trebuie de asemenea separat.
	CP	+20	Acum întoarce cu codurile finale +21, +23 și +29.
	RET	NZ	Dă caracterului '-' un cod de +5F.
	LD	A,+5F	Dă caracterului '@' un cod de +40.
03B2 K-@-CHAR	RET		
	LD	A,+40	
	RET		

RUTINELE DE DIFUZOR

Cele două subrutine care se prezintă în această secțiune sînt subrutina BEEPER, care controlează difuzorul, și rutina de comandă BEEP. Difuzorul este activat prin D4 pe nivel '0' logic în timpul unei instrucții OUT folosind portul '254'. Într-o situație similară, cînd D4 este pe nivel '1' logic, difuzorul este dezactivat. De aceea un 'beep' se va produce prin modificarea regulată a nivelului lui D4.

Acum 'nota de mijloc C' care are frecvența 261,63 Hz. Pentru a produce această notă, difuzorul va fi activat și dezactivat alternativ la fiecare a 1/523,26-a parte dintr-o secundă. În SPECTRUM sistemul clock funcționează la 3,5 MHz, iar 'nota de mijloc C' va necesita ca instrucția OUT necesară să fie executată pe cît posibil la fiecare din stările 6,689 T. Această ultimă valoare, redusă ușor la depășiri inevitabile, reprezintă în subrutina BEEPER 'lungimea buclei de timp'.

THE 'BEEPER' SUBROUTINE (SUBROUTINA BEEPER)

Subrutina începe cu registrul pereche DE conținînd valoarea 'ft', unde o notă de frecvență 'f' dată va avea o durată de 't' secunde și cu registrul pereche HL conținînd o valoare egală cu numărul de stări T din 'bucle de timp' împărțit cu '4'.

Adică, pentru ca să fie produsă 'nota de mijloc C', DE va conține pentru o secundă +0105 (INT (261.63*1)) și HL va conține +066A (derivat de la 6.689/4 - 30.125).

03B5 BEEPER	DI		Dezactivează întreruperea pe durata unui 'beep'.
	LD	A,L	Salvează temporar L.
	SRL	L	Fiecare '1' din registrul L este pentru a număra '4' stări T dar, în schimb acceptă INT (L/4) și numără '16' stări T.
	SRL	L	
	CPL		Întoarce la valoarea originală din L și află cîți de '1' s-au pierdut prin acceptarea lui INT (L/4).
	AND	+03	Adresa de bază a buclei de timp.
	LD	C,A	Modifică lungimea buclei de timp. Folosește un punct de pornire prin fiecare '1' pierdut prin acceptarea lui INT (L/4).
	LD	B,+00	Scotește culoarea de margine prezentă și mut-o la bitii 2, 1 și 0 ai registrului A.
	LD	IX,+03D1	
	ADD	IX,BC	Asigură-te că ieșirea MID este 'off'.
	LD	A,(BORDER)	
	AND	+39	
	RRCA		
	RRCA		
	OR	+08	

Acum se intră în bucle de generare a sunetului. Se fac treceri complete prin 'DE', adică un pas pentru fiecare ciclu al notei.

Registrul HL păstrează lungimea buclei de timp pentru fiecare '1' din registrul L fiind folosite '16' stări T și '1,024' stări T pentru fiecare '1' din registrul H.

03D1 BE-IX+3	NOP		Adună '4' stări T pentru fiecare punct prim de intrare folosit.
03D2 BE-IX+2	NOP		Valorile din registrele B&C vin din registrele H&L - după cum se vede mai jos.
03D3 BE-IX+1	NOP		Bucle de timp.
03D4 BE-IX+0	INC	B	Adică 'BC' * '4' stări T.
	INC	C	(Dar notează că în punctul de jumătate ciclu - C va fi egal cu 'L+1').
03D6 BE-H&L-LP	DEC	C	
	JR	NZ,03D6,BE-H&L-LP	
	LD	C,+3F	
	DEC	B	
	JP	NZ,03D6,BE-H&L-LP	

În continuare difuzorul va fi activat și dezactivat alternativ.

XOR	+10	Înșeură bitul 4.
OUT	(+FE),A	Execută operația OUT lăsînd marginea neschimbată.
LD	B,H	Resetează registrul B.
LD	C,A	Salvează registrul A.
BIT	4,A	Salt dacă ești la punctul ce

JR NZ,03F2,BE-AGAIN indică jumătate de ciclu.

După un ciclu complet este testat registrul perche DE.

LD	A,B	Salt în față dacă ultimul pas complet a fost deja făcut.
OR	E	
JR	I,03F6,BE-END	
LD	A,C	Se aduce valoarea salvată.
LD	C,L	Resetare registrul C.
DEC	DE	Se decrementează numărătorul de pași.
JP	(IX)	Salt înapoi la locația de început cerută a buclei.

Sînt stabiliți parametrii pentru a doua jumătate de ciclu.

03F2 BE-AGAIN	LD	C,L	Resetare registrul C.
	INC	C	Cum acest pas este mai scurt, se adună '16' stări T.
	JP	(IX)	Salt înapoi.

Cu privire la completarea 'beep'-ului, întreruperea mascabilă trebuie să fie validată.

03F6 BE-END	EI	Intrerupere validată.
	RET	Intrerupere finală.

THE 'BEEP' COMMAND ROUTINE (RUTINA DE COMANDA BEEP)

Această subrutină începe cu două numere în stiva calculatorului. Numărul cel mai de sus reprezintă 'nota', iar numărul imediat sub el reprezintă 'durata'.

03F8 BEEP	RST	0028,FP-CALC	Virgula mobilă este folosită pentru a manipula cele două valori - t & P.
	DEFB	+31,duplicate	t, P, P
	DEFB	+27,int	t, P, i (unde i = INT P)
	DEFB	+C0,at-see-0	t, P, i (see-0 păstrează i)
	DEFB	+03,subtract	t, p (unde p este partea fracționară a lui P)
	DEFB	+34,stk-data	Stochează valoarea zecimală a lui 'K'.
	DEFB	+EC,exponent+7C	0.0577622606 (care este puțin sub 12 V2 - 1)
	DEFB	+6C,+98,+IF,+FS	t, pK
	DEFB	+04,multiply	t, pK, i
	DEFB	+A1,stk-one	t, pK+1
	DEFB	+0F,addition	
	DEFB	+38,end-zalc	

Acum se vor efectua mai multe teste asupra lui i, partea 'întreagă a intensității'.

LD	HL,+5C92	Acesta este primul 'see-0' (HENSOT).
LD	A,(HL)	Adu exponentul lui i.
AND	A	Dacă i nu este în forma integrală (scurtă) atunci se dă o eroare.
JR	NZ,046C,REPORT-B	Copiază bitul semn în registrul C.
INC	HL	Copiază bitul cel mai puțin semnificativ în registrul B și în registrul A.
LD	C,(HL)	
LD	B,(HL)	
LD	A,B	Se dă iar un report B dacă i nu satisface testul:
RLA	A,A	-128<=i<=+127
SBC	A,A	
CP	C	
JR	NZ,046C,REPORT-B	
INC	HL	
CP	(HL)	
JR	NZ,046C,REPORT-B	
LD	A,B	Adu bitul mai puțin semnificativ și testează-l mai departe.
ADD	A,+3C	
JP	P,0425,BE-i-OK	Acceptă -60<=i<=67
JP	P0,046C,REPORT-B	Respinge de la -128 la -61.

Notă: Domeniul de la +70 la +127 se va respinge mai târziu.

Acum se va putea găsi frecvența corectă pentru 'intensitatea' i.

0425 BE-i-OK	LD	B,+FA	Se începe cu 6 octave sub nota de mijloc C.
0427 BE-OCTAVE	INC	B	Pentru a afla octava corectă se va reduce regulat i.
	SUB	+0C	
	JR	NC,0427,BE-OCTAVE	
	ADD	A,+0C	Adună înapoi ultima scădere făcută.
	PUSH	BC	Salvează numărul octavei.
	LD	HL,+046E	Adresa de bază a 'tabelii de semitonuri'.
	CALL	3406,LOC-MEM	Se ia în considerare tabelul și se merge la stiva calculatorului la a 'A'-a valoare. (Numeste-o C).
	CALL	33B4,STACK-NUM	

Acum se poate lua în considerare partea fracțională a 'intensității' i.

RST	0028,FP-CALC	t, pK+1, C
DEFB	+04,multiply	t, C(pK+1)
DEFB	+38,end-calc	

Prin modificarea 'ultimei valori', în concordanță cu numărul octavei, se găsește frecvența finală f.

POP	AF	Se scoate numărul octavei.
ADD	A,(HL)	Se multiplică 'ultima valoare' cu '2' la puterea numărului octavei.
LD	(HL),A	
RST	0028,FP-CALC	t, f
DEFB	+C0,st-mem-0	Pentru moment frecvența se pune deoparte în mem-0.
DEFB	+02,delete	

Atenție în noua turură la 'durată'.

DEFB	+31,duplicate	t, t
DEFB	+38,end-calc	
CALL	1E94,FIND-INT1	Noua valoare 'INT t' trebuie să fie în domniul +00 - +0A.
CP	+0B	
JR	NC,046C,REPORT-1	

Numărul complet de cicluri din 'beep' este dat de 'f*t'. Astfel acum se poate găsi noua valoare.

CST	0028,FP-CALC	t
DEFB	+E0,get-mem-0	t, f
DEFB	+04,multiply	f*t

Rezultatul este lăsat în stiva calculatorului în timp ce este calculată lungimea 'buclei de timp' cerută de 'beep'.

DEFB	+E0,get-mem-0	f*t, f
DEFB	+34,stk-data	Valoarea '3.5*10 ⁶ /8' este formată în vârful stivei calculatorului.
DEFB	+80,four bytes	
DEFB	+43,exponent +93	f*t, f, 437,500 (dec.)
DEFB	+55,+9F,+80,(+00)	f*t, 437,500, f
DEFB	+01,exchange	f*t, 437,500/f
DEFB	+05,division	
DEFB	+34,stk-data	
DEFB	+35,exponent +85	f*t, 437,500/4, 30,125 (dec)
DEFB	+71,(+00,+00,+00)	f*t, 437,500/f - 30,125
DEFB	+03,subtract	
DEFB	+38,end-calc	

Notă: Valoarea '437,500/f' dă lungimea 'jumătății de ciclu' a notei și reducând-o cu '30,125' se ia în considerare stările 1 '120,5' în care se produce nota și se ajustează contoarele etc.
Acum se vor putea transfera valorile în registrele cerute.

CALL	1E99,FIND-INT2	Valoarea 'buclei de timp' este în registrul pereche BC;
PUSH	BC	este salvată în stivă.

Notă: Dacă bucla de timp este prea mare, atunci va apărea o eroare (reîntorcându-se prin ERROR-1); de aceea se exclud valorile 'intensității' din domeniul '+70 - +127'.

CALL	1E99,FIND-INT2	Valoarea f*t este comprimată în registrul pereche BC.
------	----------------	---

POP	HL	Se mută valoarea 'buclei de timp' în registrul pereche HL.
LD	D,B	Se mută valoarea fet în registrul pereche DE.
LD	E,C	

Totusi, înainte de a face 'beep'-ul, se testează valoarea fet.

LD	A,E	Dacă fet a dat rezultatul că nu e necesar nici un ciclu, atunci reîntoarcere.
OR	E	Se descrește numărul de ciclu și se sare la rutina BEEPER (făcându-se în afărit un pas).
RET	Z	
DEC	DE	
JP	03B5,BEEPER	

Prezentarea 3 - INTEGER în afara domeniului

046C REPORT-1	RST	0008,ERROR-1	Cheamă rutina de tratare a erorii.
	DEFB	+0A	

THE 'SEMI-TONE' TABLE (TABELUL DE 'SEMI-TONE')

Acest tabel conține frecvențele a 12 semitonuri dintr-o octavă.

	frecvența Hz	nota
046E DEFB +89,+02,+D0,+12,+86	261.63	C
DEFB +89,+0a,+97,+60,+75	277.18	C#
DEFB +89,+12,+95,+17,+1F	293.66	D
DEFB +89,+1B,+90,+41,+02	311.13	D#
DEFB +89,+24,+90,+53,+CA	329.63	E
DEFB +89,+2E,+9D,+36,+E1	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

THE 'PROGRAM NAME' SUBROUTINE (ZX81)

Subrutina următoare se aplică la ZX81 și nu a fost scoasă când programul a fost rescris pentru SPECTRUM.

04AA DEFB +CB,+F1,+24,+3A
DEFB +38,+5C,+87,+FA
DEFB +8A,+1C,+E1,+D0
DEFB +E5,+CD,+F1,+2B
DEFB +62,+6B,+0B,+F8
DEFB +09,+CB,+FE,+C9

THE CASSETTE HANDLING ROUTINES (RUTINELE DE TRATARE A CASETEI)

Programul monitor de 16K are un set extins de rutine pentru tratarea interfetei cu caseta. In consecință, aceste rutine formează rutinele de comanda SAVE, LOAD, VERIFY și MERGE.

Punctul de intrare al rutinelor este la SAVE-ETC (0605). Cu toate acestea, înaintea acestui punct se găsește subrutinele referitoare la salvarea și încărcarea (sau verificarea) bitilor.

În toate cazurile, bitii ce se vor trata cu aceste subrutine sînt descriși de registrul pereche DE ce conține 'lungimea blocului', de registrul pereche IX ce conține 'adresa de bază' și de registrul A ce conține +00 pentru blocul de vîrf sau +FF pentru un bloc program/data.

THE 'SA-BYTES' SUBROUTINE (SUBROUTINA 'SA-BYTES')

Această subrutină este chemată să salveze informația de vîrf (de la 098A) și mai tîrziu blocul actual program/data (de la 099E).

04C2 SA-BYTES	LD HL,+053F	Preîncărcă stiva msinii cu adresa - SA/LD-RET.
	PUSH HL	Această constantă va da un rol conducător unei informații de vîrf pentru aprox.5 sec.
	LD HL,+1F80	Sari mai departe dacă se salvează un header.
		Această constantă va da un rol conducător blocului program/data pentru aprox.2 secunde.
	BIT 7,A	Indicatorul este salvat.
	JR Z,04D0,SA-FLAG	'Lungimea' este incrementată și 'adresa de bază' este redusă tinînd seama de indicator.
	LD HL,+0C98	Înteruperea mascabilă este dezactivată în timpul lui SAVE.
04D0 SA-FLAG	EX AF,A'F'	Seamnalul 'NIC on' și marginea să fie RED.
	INC DE	Se dă o valoare lui B.
	DEC IX	
	B)	
	LD A,+02	
	LD B,A	

Se intră acum într-o buclă pentru a crea impulsurile conducătorului. Atît impulsurile 'NIC on' cît și cele 'NIC off' sînt de lungime 2,168 stări T. Culoarea de margine se va schimba pe fiecare 'front' de la RED la CYAN.

Notă: Se consideră un 'front' ca fiind o tranziție atît de la 'on' la 'off', cît și de la 'off' la 'on'.

04D8 SA-LEADER	DJNZ 04D8,SA-LEADER	Perioada de timp principală.
	OUT (+FE),A	La fiecare pas, NIC on/off, margine RED/CYAN.
	XOR +OF	Constanta de timp principală.
	LD B,+A4	Se decrementează numărătorul cel mai puțin semnificativ.
	DEC L	Salt înapoi pentru un alt impuls.
	JR NZ,04D8,SA-LEADER	Se ține seamă de un pas mai lung (-reduc cu 13 stări T).
	DEC B	Se decrementează numărătorul cel mai semnificativ.
	DEC H	Salt înapoi pentru alt impuls pînă la stabilirea conducătorului.
	JP P,04D8,SA-LEADER	

Se trimite acum un impuls de sincronizare.

04EA SA-SYNC-1	LD B,+2F	
	DJNZ 04EA,SA-SYNC-1	NIC off pentru 667 stări T de la 'OUT to OUT'.
	OUT (+FE),A	NIC on și RED.
	LD A,0B	Seamnal 'NIC off & CYAN'.
	LD B,+37	NIC on pentru 735 stări de la 'OUT to OUT'.
04F2 SA-SYNC-2	DJNZ 04F2,SA-SYNC-2	Acum NIC off & margine CYAN.
	OUT (+FE),A	+3E este o constantă de timp.
	LD BC,+310E	+0E seamnal 'NIC off & YELLOW'.
	EX AF,A'F'	Încarcă fanionul și îl treci în registrul L pentru
	LD L,A	

JP 0507,SA-START 'trimitere'.
Intoarcere în bucla SAVE
(SALVARE).

Octetul SAVE (SALVARE) este acum introdus. Primul octet ce urmează a fi salvat este indicatorul; acesta este urmat de octetul informație actuală, iar ultimul octet construit este octetul paritate, care se formează prin evaluarea bitilor de mai înainte.

04FE SA-LOOP	LD A,D	Numărătorul lungime este testat și saltul se face dacă a ajuns pe zero.
	OR E	Incarcă următorul octet care urmează să fie salvat.
	JR Z,050E,SA-PARITY	Incarcă 'paritatea' curentă.
	LD L,(IX+00)	Include octetul actual.
0505 SA-LOOP-P	LD A,H	Rememorează 'paritatea'.
	XOR L	Atenție: pe această ramură valoarea indicatorului inițializează 'paritatea'.
0507 SA-START	LD H,A	Senal 'MIC on & BLUE'.
		Setează fanionul de transport. Va funcționa ca un 'marcător' pentru cei 8 biti ai octetului.
	LD A,+01	Salt înainte.
	SCF	
	JP 0525,SA-8-BITES	

Cînd trebuie trimis octetul 'paritate', atunci el este transferat în registrul L pentru a fi salvat.

050E SA-PARITY	LD L,H	În valoarea 'paritate finală'
	JR 0505,SA-LOOP-P	Salt înapoi.

Urmașoarea buclă internă produce actualul tact. Accesul în buclă se face la SA-BIT-1 cu tipul bitului de salvat indicat de fanionul de transport. Pentru fiecare bit se fac două treceri prin buclă, realizînd astfel un 'tact off' și nu un 'tact on'.

0511 SA-BIT-2	LD A,C	Adus la a doua trecere și se încarcă 'MIC off & YELLOW'.
	BIT 7,B	Pune fanionul zero să indice 'a doua trecere'.
0514 SA-BIT-1	DJNZ 0514,SA-BIT-1	Aceeași buclă de sincronizare. Totdeauna 801 stări T la a doua trecere.
	JR NC,051C,SA-OUT	Salt, pe calea cea mai scurtă dacă se salvează '0'.
	LD B,+42	Oricum, dacă se salvează '1', atunci se adună 855 stări T.
051A SA-SET	DJNZ 051A,SA-SET	La prima trecere este 'MIC on & BLUE' iar la a doua trecere este 'MIC off & YELLOW'.
051C SA-OUT	OUT (+FE),A	Incarcă constanta de sincronizare pentru a doua trecere.
	LD B,+3E	Salt înapoi la sfîrșitul primei treceri; altfel se refac 13 stări T.
	JR NZ,0511,SA-BIT-2	Sterge fanionul de transport și pune în A +01 (MIC on & BLUE) înainte de a continua 'bucla celor 8 biti'.
	DEC B	
	XOR A	
	INC A	

'Bucla celor 8 biti' este accesată inițial cu întregul octet în registrul L și indicatorul de transport este 1. Oricum, este reentrantă după fiecare bit ce a fost salvat pînă cînd s-a ajuns la capăt, cînd 'marcătorul' trece în indicatorul de transport, lăsînd registrul L plin.

0525 SA-8-BITS	RL L	Pune bitul 7 în indicatorul de transport și deplasează la stînga 'marcătorul'.
	JP NZ,0514,SA-BIT-1	Salvare bit pînă la terminare octet.
	DEC B,E	Decrementare 'contor'.
	INC IX	Avans 'adresa bază'.
	LD B,+31	Setare constantă de sincronizare pentru primul bit al următorului octet.
	LD A,+7F	Intoarcere (la SA/LD-RET)

	IN	A, (+FE)	dacă taste BREAK (PAUZA) a fost apăsată.
	RRA	NC	
	RET	NC	
	LD	A, D	Testare 'contor' și salt înapoi chiar dacă este zero (pentru a trimite octetul 'paritate').
	INC	A	
	JP	NZ, 04FE, SA-LOOP	Iesire când 'contorul' a ajuns la +FFFF. Dar înainte se dă o scurtă întârziere.
053C SA-DELAY	LD	B, +3B	
	DJNZ	053C, SA-DELAY	
	RET		

Notă: O resetare va da pentru 'MIC off' tact pentru 855 stări T urmate de un 'MIC on' tact pentru 855 stări T. Se ia în considerare că setarea unui bit va da tacturi exact de două ori mai lungi. De asemenea nu există intervale nici între tactul sincronizare și primul bit al indicatorului, nici între octeți.

THE 'SA/LD-RET' SUBROUTINE

Această subrutină este comună pentru SALVARE și INCARCARE. Marginea ecranului este pusă pe culoarea originală, iar taste BREAK testată pentru ultima dată.

053F SA/LD-RET	PUSH	AF	Salvare indicator de transport. (Este resetat după o încărcare eronată). Se aduce culoarea originală a marginii din variabila sa sistem.
	LD	A, (BORDCA)	Mută culoarea marginii în bitii 2, 1 și 0.
	AND	+3B	
	RRCA		
	RRCA		
	RRCA		
	OUT	(+FE), A	Pune marginea pe culoarea originală.
	LD	A, +7F	Citeste taste BREAK pentru ultima oară.
	IN	A, (+FE)	
	RRA		
	EI		Autorizare întrerupere mascabilă.
	JR	C, 0554, SA/LD-END	Salt numai dacă urmează o pauză.

Prezentarea D-BREAK-CONT se repetă.

0552 REPORT-D	RST	0008, ERROR-1	Apel rutină tratare eroare.
	DEFB	+0C	

Se continuă aici.

0554 SA/LD-END	POP	AF	Regăsire fanion de transport.
	RET		Întoarcere în rutina apelantă.

THE 'LD-BYTES' SUBROUTINE

Această subrutină este apelată pentru încărcarea (LOAD) header-ului informației (de la 078E) și încărcarea întârziată (LOAD) sau verificarea (VERIFY) unui set actual de informații (de la 0802).

0556 LD-BYTES	INC	D	Resetează fanionul zero (în D nu poate fi +FF).
	EX	AF, A'F'	Registrul A conține +00 pentru header și +FF pentru un set de informații.
	DEC	D	Fanionul de transport este resetat pentru verificare (VERIFY) și setat pentru încărcare (LOAD).
	DI		Refă D la valoarea de la care a plecat.
	LD	A, +0F	Intreruperea mascabilă este acum dezautorizată.
	OUT	(+FE), A	Marginea se face acum WHITE (albă).
	LD	HL, +053F	
	PUSH	HL	Încarcă prima dată în stiva calculatorului adresa SA/LD-RET.
	IN	A, (+FE)	Se face o citire inițială a portului '254'.
	RRA		Se rotește octetul obținut,
	AND	+20	dar se păstrează numai bitul

OR +02
LD C,A

CP A

EAR.
Semnal margine 'RED' (rosu).
Se păstrează valoarea în
registrul C (+22 pentru 'off'
și +02 pentru 'on' - starea
prezentă pentru EAR).
Se setează fanionul zero.

Primul stadiu al citirii benzii implică demonstrarea faptului că semnalul
pulsat actual există (de exemplu, tranzițiile 'on/off' sau 'off/on').

056B LD-BREAK RET NZ
056C LD-START CALL 05E7,LD-EDGE-1
JR NC,056B,LD-BREAK

Revenire dacă taste BREAK a
fost apăsată.
Întoarcere cu fanionul de
transport pe zero dacă nu
este 'tranziție' între
aproximativ 14,000 stări T.
Dar dacă se găsește o
'tranziție', marginea
ecranului va fi CYAN (azuriu)

Următorul stadiu implică o așteptare și apoi demonstrarea că semnalul
pulsează.

0574 LD-WAIT LD HL,+0415
DJNZ 0574,LD-WAIT
DEC HL
LD A,H
OR L
JR NZ,0574,LD-WAIT
CALL 05E3,LD-EDGE-2
JR NC,056B,LD-BREAK

Lungimea acestei perioade de
așteptare va fi de circa o
secundă.

Se continuă numai dacă sînt
găsite două 'tranziții' în
perioada de timp acordată.

Acum se acceptă un semnal 'conducător'.

0580 LD-LEADER LD B,+9C
CALL 05E3,LD-EDGE-2
JR NC,056B,LD-BREAK

LD A,+C6
CP B
JR NC,056C,LD-START
INC H
JR NZ,0580,LD-LEADER

Constanta de sincronizare.
Continuă numai dacă s-au
găsit două 'tranziții' în
perioada de timp acordată.
Totuși tranzițiile trebuie să
fie găsite în aproximativ
3,000 stări T pentru fiecare.
Numără perechile de tranziții
în registrul H pînă găsește
'256' perechi.

Apoi, tactul conducător este 'off' și 'on' este parte a tactului de
sincronizare.

058F LD-SYNC LD B,+C9
CALL 05E7,LD-EDGE-1
JR NC,056B,LD-BREAK
LD A,B
CP +D4
JR NC,058F,LD-SYNC

CALL 05E7,LD-EDGE-1
RET NC

Constanta de sincronizare.
Fiecare tranziție este
considerată pînă cînd sînt
găsite două tranziții
apropiate împreună - acesta
va fi începutul și sfîrșitul
tranziției impulsului de
sincronizare 'off'.
Tranziția de sfîrșit a
impulsului 'on' trebuie să
existe.
(Revenire dacă fanionul de
transport este resetat.)

Octetul header-ului sau programul/setul de informații poate fi acum fi acum
încărcat (LOAD) sau verificat (VERIFY). Dar primul octet este tipul
fanionului.

LD A,C
XOR +03

LD C,A
LD H,+00

LD B,+B0

JR 05C8,LD-MARKER

Culoarea marginii ecranului
de acum înainte va fi BLUE &
YELLOW (albastru & galben).

Initializare octet 'egalitate
paritate' pe zero.
Setare constantă de
initializare pentru octetul
indicator.
Salt înainte în bucla
LOADING (încărcare) octet.

Bucia **LOADING** (încărcare) octet este folosită pentru aducerea unui octet la un moment dat. Primul octet este al fanioanelor. Acesta este urmat de octetii de informații, iar ultimul este octetul 'paritate'.

05A9 LD-LOOP	EX	AF,A'F'	Se aduc fanioanele.
	JR	NZ,05B3,LD-FLAG	Salt înainte numai dacă se tratează primul octet.
	JR	NC,05B0,LD-VERIFY	Salt înainte dacă se verifică o bandă.
	LD	(IX+00),L	Actuala încărcare se face la cerere.
05B3 LD-FLAG	JR	05C2,LD-NEXT	Se face salt înainte pentru încărcarea octetului următor.
	RL	C	Se păstrează temporar fanionul de transport într-un loc sigur.
	XOR	L	Revenire dacă tipul fanionului nu coincide cu primul octet al benzii.
	RET	NZ	(Fanionul de transport resetat.)
	LD	A,C	Se readuce fanionul de transport acum.
	RRA		
	LD	C,A	
	INC	DE	
	JR	05C4,LD-DEC	Se incrementează contorul pentru a compensa decrementarea anterioară saltului.

Dacă un set de informații a fost verificat, atunci octetul actual încărcat este testat în raport cu octetul inițial.

05B0 LD-VERIFY	LD	A,(IX+00)	Se aduce octetul inițial.
	XOR	L	Se compară octetul nou cu octetul inițial.
	RET	NZ	Revenire dacă este 'neegalitate'. (Fanionul de transport resetat.)

Un nou octet poate fi acum colectat de pe bandă.

05C2 LD-NEXT	INC	IX	Incrementare 'destinație'.
	DEC	DE	Se decrementează 'contorul'.
	EX	AF,A'F'	Se salvează fanioanele.
	LD	B,+B2	Încărcare constantă de sincronizare.
05C8 LD-MARKER	LD	L,+01	Se sterge registrul 'obiect' separat printr-un bit 'marcător'.

Bucia '**LD-8-BITS**' este folosită pentru a construi în registrul L un octet.

-8 05CA LD-BITS	CALL	05E3,LD-ERBE-2	Gaseste lungimea impulsului 'off' si 'on' al urmatorului bit.
	RET	NC	Revenire dacă perioada de timp a trecut. (Fanionul de transport este resetat.)
	LD	A,+CE	Se compară lungimea cu aproximativ 2,400 stări T ₁
	CP	R	fanionul de transport se setează pentru un '0' si se resetează pentru un '1'.
	RL	L	Se include noul bit în registrul L.
	LD	B,+B0	Se pune constanta de sincronizare pentru bitul următor.
	JP	NC,05CA,LD-8-BITS	Salt înapoi dacă mai sînt biti de adus.

Octetul 'egalitate paritate' trebuie să fie actualizat pentru fiecare octet.

LD	A,H	Se încarcă octetul 'egalitate paritate' si se include noul octet.
XOR	L	
LD	H,A	Se salvează încă odată.

Trecerile prin buclă se fac pînă cînd 'contorul' ajunge pe zero. În acest moment octetul 'egalitate paritate' poate fi ținut pe zero.

LD	A, B	Dacă registrul pereche DE nu este pe zero, se trece mai departe.
OR	E	
JR	NZ, 05A9, LD-LOOP	
LD	A, H	Se aduce octetul 'egalitate paritate'.
CP	+01	Revenire cu fanionul de transport setat dacă valoarea este zero.
RET		(Fanionul de transport este resetat dacă este eroare.)

THE 'LD-EDGE-2' AND 'LD-EDGE-1' SUBROUTINES

Aceste două subrutine formează cea mai importantă parte a operației de LOAD/VERIFY (încărcare/verificare).

Subrutinele sînt introduse cu o constantă de sincronizare în registrul B, iar culoarea anterioară a marginii și 'tipul tranziției' în registrul C.

Subrutinele revin cu fanionul de transport setat, dacă numărul de 'tranziții' cerut a fost găsit în timpul permis; și schimbarea valorii din registrul B arată doar ce perioadă a fost necesară pentru găsirea 'tranziției'.

Fanionul de transport va fi resetat dacă este o eroare. Fanionul zero va fi resetat dacă semnalul 'BREAK' este tastat, sau va fi setat pentru 'timp expirat'.

Punctul de intrare LD-EDGE-2 este folosit cînd este cerută lungimea unui impuls complet, iar LD-EDGE-1 este folosit pentru a găsi timpul dinaintea 'tranziției' următoare.

05E3 LD-EDGE-2	CALL	05E7, LD-EDGE-1	Prin urmare, apelează LD-EDGE-1 de două ori; revenire dacă este eroare.
	RET	NC	Așteaptă 358 stări T înaintea intrării în buclă esanțion.
05E7 LD-EDGE-1	LD	A, +16	
05E9 LD-DELAY	DEC	A	
	JR	NZ, 05E9, LD-DELAY	
	AND	A	

Intrarea în buclă esanțion s-a făcut. Valoarea din registrul B se incrementează pentru fiecare trecere; 'timpul expirat' este dat cînd B ajunge pe zero.

05EB LD-SAMPLE	INC	B	Numără fiecare trecere.
	RET	Z	Revenire cu fanionul de transport resetat și cu fanionul zero setat dacă este 'timpul expirat'.
	LD	A, +7F	Se citește din portul +7FFE.
	IN	A, (+FE)	Acesta este BREAK & EAR.
	RRR		Roteste octetul.
	RET	NC	Revenire cu fanionul de transport resetat și cu fanionul zero resetat dacă s-a tastat 'BREAK'.
	XOR	C	Acum se testează octetul față de 'tipul ultimei tranziții'; salt înapoi numai dacă s-a schimbat.
	AND	+20	
	JR	Z, 05EB, LD-SAMPLE	

O nouă 'tranziție' a fost găsită în perioada de timp alocată căutării. Deci se schimbă culoarea marginii și se setează fanionul de transport.

LD	A, C	Se schimbă 'tipul ultimei tranziții' și culoarea marginii ecranului.
CPL		
LD	C, A	
AND	+07	Se păstrează numai culoarea marginii.
OR	+08	Se resetează fanionul de transport.
OUT	(+FE), A	Se schimbă culoarea marginii (RED/CYAN (rosu/azuriu) sau BLUE/YELLOW (albastru / galben)).
SCF		Se semnalizează reușita căutării înaintea revenirii.
RET		

Notă: Subrutina LD-EDGE-1 ia 465 stări T, plu 58 stări T pentru fiecare trecere nereușită prin buclă esanțion.

De exemplu, de aceea, cînd se așteaptă impulsul sincron (vezi LD-SYNC la 058F) se incrementează zece treceri adiționale prin bucla esanțion. Căutarea este astfel pentru ca următoarea tranziție să fie găsită, grosolan, în 1,100 stări T (465 + 10 + 58 + deasupra). Aceasta va dovedi realizarea impulsului sincron 'off' care vine după lungul 'impuls conducător'.

THE 'SAVE, LOAD, VERIFY & MERGE' COMMAND ROUTINES (RUTINELE DE SALVARE, INCARCARE, VERIFICARE SI UNIRE)

Punctul de intrare SAVE-ETC este folosit pentru toate cele patru comenzi. Valoarea continuată în T-ADDR oricum va fi distinctă pentru cele patru comenzi. Prima parte a rutinei următoare se ocupă de construirea unui 'header informație' în spațiul de lucru.

0605 SAVE-ETC POP AF
 LD A,(T-ADDR-10)
 SUB +E0
 LD (T-ADDR-10),A
 CALL 1C8C,EXPT-EXP
 CALL 2530,SYNTAX-Z
 JR Z,0652,SA-DATA
 LD BC,+0011
 LD A,(T-ADDR-10)
 AND A
 JR Z,0621,SA-SPACE
 LD C,+22

Se trimite adresa SCAN-LOOP.
 Se scade T-ADDR-10 cu +E0,
 se dă +00 pentru SAVE, +01
 pentru LOAD, +02 pentru
 VERIFY și +03 pentru MERGE.
 Se trec parametrii 'nume' în
 stiva calculatorului.
 Salt înainte dacă se caută
 sintaxa.
 Se cedează 17 locații pentru
 header-ul SAVE, dar 34 pentru
 celelalte comenzi.

0621 SA-SPACE RST 0030,BC-SPACES

Spațiul total este cerut din
 spațiul de lucru.

 PUSH BE
 POP IX
 LD I,+01
 LD A,+20
 LD (BE),A
 INC DE
 DJNZ 0629,SA-BLANK
 LD (IX+01),+FF

Se copiază adresa de start
 din registrul pereche IX.
 Numele unui program poate
 avea pînă la zece caractere,
 dar prima dată se introduc 11
 caractere spațiu în zona
 rezervată.

0629 SA-BLANK

 CALL 2BF1,STK-FETCH

Un nume invalidat este numai
 +FF.

 LD HL,+FFF6
 DEC BC
 ADD HL,BC
 INC BC
 JR NC,0643,SA-NAME
 LD A,(T-ADDR-10)
 AND A
 JR NZ,0644,SA-NULL

Parametrii numelui sînt
 încărcati și lungimea este
 testată.
 Este '-10'.

Da, salt înainte dacă
 lungimea numelui nu este prea
 lungă (nu mai mult de 10
 caractere).
 Dar aprobă încărcarea (LOAD),
 verificarea (VERIFY) și
 unirea (MERGE) programelor cu
 nume 'invalidat' sau foarte
 lung.

Prezentarea 1 - numele invalidat al fișierului.

0642 REPORT-F RST 000B,ERRDR-1
 DEFB +0E

Ape! rutină pentru tratarea
 erorii.

Se continuă tratarea numelui programului.

0644 SA-NULL LD A,B
 OR C
 JR Z,0652,SA-DATA
 LD BC,+000A

Salt înainte dacă numele are
 lungimea invalidată.

Dar se trunchiază numele prea
 lungi.

Numele va fi transferat acum în spațiul de lucru (a doua locație în
 continuare).

0643 SA-NAME PUSH IX
 POP HL
 INC HL
 EX DE,HL
 LDIR

Se aduce adresa de început în
 registrul pereche HL.
 Pas spre a doua locație.
 Se schimbă indicatorul și se
 copiază numele.

Se prezintă în continuare cazul în care parametrii multi și diferiți urmează o
 comandă.
 Se începe cu tratarea 'xxx' 'name' 'DATA'.

0652 SA-DATA RST 0018,GET-CHAR Codul prezent este simbolul
CP +E4 'DATA'?
JR NZ,06A0,SA-SCR\$ Dacă nu este, salt.
LB A,(T-ADDR-10) Oricum, nu este posibil să
CP +03 fie 'MERGE name DATA'.
JP Z,10BA,REPORT-C Se avansează CH-ADD.
RST 0020,NEXT-CHAR Se face controlul dispunerii
CALL 28B2,LOOK-VARS în spațiul variabilelor.
SET 7,C Setare bit 7 în dispunerea
numelui.
JR NC,0672,SA-V-OLD Salt dacă se tratează o
LD HL,+0000 dispunere existentă.
LD A,(T-ADDR-10) Semnal 'utilizare dispunere
DEC A nouă'.
JR Z,0685,SA-V-NEW Se încarcă valoarea T-ADDR și
este eroare dacă se încearcă
sau verificare a unei noi
dispuneri.

Prezentarea 2 - Variabilă negăsită.

0670 REPORT-2 RST 0008,ERRDR-1 Apel rutină tratare eroare.
DEFB +01

Se continuă cu tratarea unei dispuneri existente.

0672 SA-V-OLD JP NZ,10BA,REPORT-C Notă: Acesta nu reușește să
CALL 2530,SYNTAX-Z excludă sirurile simple.
JR Z,0692,SA-DATA-1 Salt înainte dacă se verifică
INC HL sintaxa.
LD A,(HL) Incrementare 'lungime mică' a
LD LD (IX+0B),A variabilei.
INC HL Octetul lungimii mici trece
în spațiul de lucru; el este
LD A,(HL) urmat de octetul lungime
LD LD (IX+0C),A mare.
INC HL Pas peste lungimea octetului.

Următoarea parte este comună și pentru dispunerea 'veche' și pentru dispunerea
'nouă'. Notă: Cale eronată sintactic.

0685 SA-V-NEW LD (IX+0E),C Incărcare nume dispunere.
LD A,+01 Initializare dispunere de
numere.
BIT 6,C Salt dacă este așa.

068F SA-V-TYPE JR Z,068F,SA-V-TYPE Este o dispunere de
INC A caracter.
LD (IX+00),A Salvare 'tip' în prima
locatie a spațiului de
header.

Ultimă parte a afirmației este examinată înaintea celorlalte căi.

0692 SA-DATA-1 EX DE,HL Salvare indicator în DE.
RST 0020,NEXT-CHAR Următorul caracter este')' ?
CP +29
JR NZ,0672,SA-V-OLD Se prezintă registrul C dacă
nu este.
RST 0020,NEXT-CHAR Avansează CH-ADD.
CALL 1BEE,CHECK-END Inscribe următoarea formulare
dacă verifică sintaxa.
EX DE,HL Readucere indicator în
JP 075A,SA-ALL registrul pereche HL înainte
de saltul în față.
(Indicatorul arată începutul
unei dispuneri cu conținut
existent.)

Acum se va considera 'SCREEN\$'.

06A0 SA-SCR\$ CP +AA Codul prezent este 'SCREEN\$'.
JR NZ,06C3,SA-CODE Salt dacă nu este.
LD A,(T-ADDR-10) Oricum nu este posibil să fie
CP +03 'MERGE name SCREEN\$'.

JP	Z, 1C8A, REPORT-C	Avansează CH-ADD.
RST	0020, NEXT-CHAR	Se înscrie următoarea
CALL	1BEE, CHECK-END	formulare dacă verifică
		sintaxa.
LD	(IX+0B), +00	Spatiul pentru înscriere și
LD	(IX+0C), +1B	spatiul atributelor ocupă
		+1B00 locații, care încep la
LD	HL, +4000	+4000; aceste detalii sînt
LD	(IX+0D), L	trecute în spatiul headerului
LD	(IX+0E), H	din spatiul de lucru.
JR	0710, SA-TYPE-3	Salt înainte.

Acum se va considera 'CODE'.

06C3 SA-CODE	CP	+AF	Este codul prezentat simbolul
			'CODE' ?
JR	NZ, 0716, SA-LINE		Dacă nu este, atunci salt.
LD	A, (T-ADDR-1e)		Oricum nu este posibil să fie
CP	+03		'MERGE name CODE'.
JP	Z, 1C8A, REPORT-C	Avansează CH-ADD.	
RST	0020, NEXT-CHAR	Salt înainte dacă formularea	
CALL	2048, PR-ST-END	nu a fost terminată.	
JR	NZ, 06E1, SA-CODE-1	Oricum nu este posibil să fie	
LD	A, (T-ADDR-1e)	'SAVE name CODE' cu el	
AND	A	însusi.	
JP	A, 1C8A, REPORT-C	Se pune un zero în stiva	
CALL	1CE6, USE-ZERO	calculatorului pentru 'start'	
		Salt înainte.	
JR	06F0, SA-CODE-2		

Prezentare 'adresă de început.

06E1 SA-CODE-1	CALL	1C82, EXPT-INUM	Se aduce primul număr.
	RST	0018, GET-CHAR	Este caracterul adus ' ' ?
	CP	+2C	
	JR	Z, 06F5, SA-CODE-3	Dacă este, salt înainte -
			numărul a fost 'adresa de
			început'.
	LD	A, (T-ADDR-1e)	Oricum se refuză 'SAVE name
	AND	A	CODE', care nu are un
	JP	Z, 1C8A, REPORT-C	'început' și o 'lungime'.
06F0 SA-CODE-2	CALL	1CE6, USE-ZERO	Se pune un zero în stiva
			calculatorului pentru
			'lungime'.
	JR	06F9, SA-CODE-4	Salt înainte.

Se încarcă 'lungimea' așa cum a fost specificat.

06F5 SA-CODE-3	RST	0020, NEXT-CHAR	Avansează CH-ADD.
	CALL	1C82, EXPT-INUM	Se încarcă 'lungimea'.

Parametrii sînt acum păstrați în spatiul header-ului din spatiul de lucru.

06F9 SA-CODE-4	CALL	1BEE, CHECK-END	Deplasare în formularea
			următoare care verifică acum
			sintaxa.
	CALL	1E99, FIND-INT2	Se condensează 'lungimea' în
	LD	(IX+0B), C	registru pereche IC și se
	LD	(IX+0C), B	stochează.
	CALL	1E99, FIND-INT2	Se condensează 'adresa de
	LD	(IX+0B), C	început' în registrul pereche
	LD	(IX+0E), B	IC și se stochează.
	LD	H, B	Se transferă 'indicatorul' în
	LD	L, C	registru pereche HL, ca de
			obicei.

'SCREEN\$' și 'CODE' sînt ambele de tipul 3.

0710 SA-TYPE-3	LD	(IX+00), +03	Se încarcă tipul numărului.
	JR	075A, SA-ALL	Se revine la alte cai.

Acum se vor prezenta 'LINE' și 'nu mai multi parametrii'.

0716 SA-LINE	CP	+CA	Codul prezent este 'LINE' ?
	JR	Z, 0723, SA-LINE	Dacă este, atunci se execută
			salt.
	CALL	1BEE, CHECK-END	Dacă se verifică sintaxa,
			atunci salt la formularea

LD	(IX+0E),+80	urătoare.
JR	073A,SA-TYPE-0	Dacă nu sînt mai multi parametri se introduce +80. Salt înainte.

Incãrcare 'numãr linie', ce trebuie sã urmeze 'LINE'.

0723 SA-LINE-1	LD A,(T-ADDR-1e)	Este permis doar 'SAVE name LINE number'.
	AND A	
	JP NZ,IC8A,REPORT-C	Avanseazã CH-ADD.
	RST 0020,NEXT-CHAR	Se trece numãrul în stiva calculatorului.
	CALL IC82,EXPT-INUM	Dacã se verificã sintaxa, salt în formularea urãtoare.
	CALL IBEE,CHECK-END	Se comprimã 'numãr linie' în registrul pereche BC si se stocheazã.
	CALL IE99,FIND-INT2	
	LD (IX+0D),C	
	LD (IX+0E),B	

'LINE si 'nu mai multi parametri' sînt de tip 0.

073A SA-TYPE-0	LD (IX+00),+00	Se încarcã 'tipul' numãrului.
----------------	----------------	-------------------------------

Parametrii care descriu programul si variabilele lui sînt pãstrate si stocate în spatiul header-ului din spatiul de lucru.

LD	HL,(E-LINE)	Indicatorul pentru sfîrșitul spatiului variabilelor.
LD	DE,(PRUG)	Indicatorul pentru începutul programului BASIC.
SCF		Acum se executã scãderi pentru a gãsi lungimea program + variabile.
SBC	HL,DE	rezultatul se stocheazã.
LD	(IX+0B),L	Operatiile se repetã, dar de data aceasta se stocheazã doar lungimea 'program'.
LD	(IX+0C),H	
LD	HL,(VARB)	
SBC	HL,DE	
LD	(IX+0E),L	
LD	(IX+10),H	
EX	DE,HL	Se transferã 'indicatorul' în registrul pereche HL, ca de obicei.

In toate cazurile header-ul de informatie a fost pregãtit.

Locatia 'IX+00' contine numãrul tipului.
 Locatiile 'IX+01' pînã la IX+0A' contin numele (+FF în 'IX+01' dacã este invalidat.)
 Locatiile 'IX+0B & IX+0C' contin numãrul de octeti care trebuie sã fie gãsiți în 'blocul de informatii'.
 Locatiile 'IX+0D' pînã la IX+10' contin o varietate de parametrii a cãror interpretare depinde de 'tip'.

Rutina continuã cu primul task realizat astfel încît sã separe SAVE de LOAD, VERIFY si MERGE.

075A SA-ALL	LD A,(T-ADDR-1e)	Salt înainte dacã se trateazã o comandã SAVE (salvare).
	AND A	
	JP Z,0970,SA-CONTRL	

In cazul unei comenzi LOAD, VERIFY sau MERGE, primii 17 octeti ai spatiului header-ului din spatiul de lucru contin informatia pregãtitã, descriã mai sus; acum este momentul sã se aducã 'header-ul' de pe bandã.

PUSH	HL	Salvare indicator 'destinatie'.
LD	BC,+0011	Din registrul dublu IX se aduce adresa de bazã pentru 'al doilea spatiu al header-ului'.
ADD	IX,BC	

Acum intervine o buclã; se pãstreazã bucla numai cînd 'header-ul' a fost încãrcat.

0767 LD-LOOK-H	PUSH IX	Se face o copie a adresei de bazã.
	LD DE,+0011	Se încarcã 17 octeti.
	XOR A	Se șterge 'header'.
	SCF	Se șterge 'LOAD'.
	CALL 0556,LD-BYTES	Acum cautã header.

POP IX Se regăsește adresa de bază.
 JR NC,0767,LD-LOOK-H Ciclează bucla pînă se cînd
 se termină încărcarea.

Noul 'header' este afișat pe ecran, dar rutina se va executa numai dacă 'noul' header coincide cu 'vechiul' header.

LD A,+FE Se asigură că este deschis
 CALL 1601,CHAN-OPEN canalul 'B'.
 LD (SER-ET),+03 Setare numărator defilare.
 LD C,+80 Semnal 'numele nu coincid'.
 LD A,(IX+00) Se compară 'noul' tip cu
 CP (IX-11) 'vechiul' tip.
 JR NZ,078A,LD-TYPE Salt dacă tipurile nu coincid
 LD C,+F6 Dacă ele coincid, va apărea
 semnal 'zece caractere
 coincide'.
 078A LD-TYPE CP +04 Curățarea 'header-ului' nu
 JR NC,0767,LD-LOOK-H are sens dacă este 'tip 4 sau
 mai mult'.

Va fi tipărit mesajul potrivit - 'Program', 'Numele spațiului', 'Spațiu
 caracteri' sau 'Octet'.

Adresa de bază a blocului de
 mesaj.
 Se salvează registrul C pînă
 sînd mesajul potrivit va fi
 tipărit.

'Nume nou' este tipărit și așa cum sînt date, 'nou' și 'vechi' nume sînt
 comparate.

PUSH IX
 LD DE,+09C0
 POP DE
 LD HL,+FFF0
 PUSH BC
 ADD HL,DE
 CALL 0C0A,PO-MSS
 LD B,+0A
 POP BC
 LD A,(HL)
 INC A
 JR NZ,07A6,LD-NAME Salt înainte dacă nu este
 coincidentă cu noul nume.
 LD A,C
 ADD A,B
 LD C,A Dacă 'nume vechi' este
 invalidat, este trimis
 semnalul 'zece caractere
 deja coincide'.

Este introdusă o buclă pentru tipărirea caracterelor de la 'nume nou'. Numele
 va fi acceptat dacă 'contorul' ajunge pe zero, pînă la urmă.

07A6 LD-NAME INC DE
 LD A,(DE) Se consideră că fiecare
 caracter de la 'nume nou'
 este unul după altul.
 CP (HL) Se compară cu caracterul
 corespunzător al lui 'nume
 vechi'.
 INC HL
 JR NZ,07AD,LD-CH-PR Nu se numără dacă nu coincid.
 INC C
 07AD LD-CH-PR RST 0010,PRINT-A-1 Se tipărește 'noul' caracter.
 DJNZ 07A6,LD-NAME Se buclează pentru zece
 caractere.
 BIT 7,C Se acceptă numele numai dacă
 contorul a ajuns la zero.
 JR NZ,0767,LD-LOOK-H Urmează 'nume nou' cu
 LD A,+0D revenire la începutul
 RST 0010,PRINT-A-1 rîndului (CR).

Headerul corect a fost găsit și a sosit momentul pentru a fi prezentate
 separat cele trei comenzi LOAD (încărcare), VERIFY (verificare) și MERGE
 (unire).

POP HL
 LD A,(IX+00) Încarcă indicatorul.
 CP +03 'SCREEN\$ și CODE' sînt
 JR Z,07CB,VR-CONTRL comparate cu VERIFY.
 LD A,(T-ADDR-10) Salt înainte dacă este

DEC	A	folosită o comandă LOAD.
JP	Z,0808,LD-CONTRL	
CP	+02	Salt înainte dacă este
JP	Z,0816,ME-CONTRL	folosită o comandă MERGE; se continuă cu o comandă VERIFY.

THE 'VERIFY' ROUTINE (RUTINA DE VERIFICARE)

Procesul de verificare implică încărcarea (LOAD) unui bloc de informații, un octet la un moment dat, dar octetii nu sînt reținuți - ci doar verificați. Această rutină este folosită de asemenea pentru încărcarea (LOAD) blocurilor de informații care au fost descrise cu 'SCREEN# & CODE'.

0703 VR-CONTRL	PUSH HL	Salvare 'indicator'.
	LD L,(IX-06)	Se aduce 'număr octeti' cum
	LD H,(IX-05)	s-a descris în 'vechiul'
		header.
	LD E,(IX+0B)	Se aduce și numărul din
	LD D,(IX+0C)	'noul' header
	LD A,H	Salt înainte dacă 'lungimea'
	OR L	nu este specificată.
	JR Z,07E9,VR-CONT-1	De exemplu, doar 'LOAD' nume
		'CODE'.
	SRC HL,DE	Se dă prezentarea R dacă se
	JR C,0806,REPORT-R	încearcă încărcarea (LOAD)
		unui bloc mai mare decît a
		fost cerut.
	JR Z,07E9,VR-CONT-1	Acceptă 'lungimi' egale.
	LD A,(IX+00)	De asemenea, se dă
	CP +03	prezentarea R dacă se
	JR NZ,0806,REPORT-R	încearcă verificarea
		(VERIFY) blocurilor de
		mărimi diferite. ('Lungime
		veche' este mai mare decît
		'lungime nouă'.)

Rutina continuă, luînd în considerare 'indicatorul destinație'.

07E9 VR-CONT-1	POP HL	Se încarcă 'indicatorul';
		acesta este 'start'.
	LD A,H	Acest 'indicator' va fi
	OR L	folosit numai în cazul în
	JR NZ,07F4,VR-CONT-2	care zero, caz în care
	LD L,(IX+0D)	'start' găsit în 'noul'
	LD H,(IX+0E)	header va fi folosit pe loc.

Indicatorul pentru VERIFY/LOAD (verificare/încărcare) este luat acum în considerare și încărcarea actuală făcută.

07F4 VR-CONT-2	PUSH HL	Se mută 'indicatorul' în
	POP IX	registru pereche HL.
	LD A,(T-ADDR-10)	Salt înainte numai dacă se
	CP +02	folosește comanda VERIFY;
	SCF	fanionul de transport va
	JR NZ,0800,VR-CONT-3	indica 'LOAD' (încărcare).
	AND A	Semnalare 'VERIFY'.
0800 VR-CONT-3	LD A,+FF	Se semnalizează 'acceptare
		numai bloc de informații'
		înaintea încărcării (LOAD)
		blocului.

THE 'LOAD DATA BLOCK' SUBROUTINE (SUBRUTINA 'ÎNCARCARE BLOC DE INFORMATII')

Subrutina este comună pentru toate rutinele de încărcare (LOAD). În cazul LOAD & VERIFY (încărcare și verificare) subrutina acționează ca o revenire completă de pe caseta de tratare a rutinei, dar în cazul MERGE (unire) blocul de informații trebuie să fie 'contopit'.

0802 LD-BLOCK	CALL 0556,LD-BYTES	Încărcarea și verificarea
	RET C	unui bloc de informații.
		Revenire numai în caz de
		eroare.

Prezentare R - înregistrează încărcare eronată.

0806 REPORT-R	RST 0008,ERROR-1	Apel rutină tratare eroare.
	DEFB +1A	

THE 'LOAD' CONTROL ROUTINE (RUTINA DE CONTROL ÎNCARCARE)

Această rutină controlează încărcarea unui program BASIC, și a variabilelor

lui, sau a unui spatiu.

0808 LD-CONTRL	LD	E, (IX+0B)	Încărcare număr octeți' așa cum sînt dati în 'noul header'.
	LD	D, (IX+0C)	
	PUSH	HL	Se salvează 'indicator destinație'.
	LD	A, H	Salt înainte numai dacă se încearcă încărcarea unei porțiuni anterioare nedecarate.
	OR	L	
	JR	NZ, 0819, LD-CONT-1	Se adună trei biți la lungime - pentru nume, lungimea minimă și lungimea maximă a unei variabile.
	INC	DE	Salt înainte.
	INC	DE	
	INC	DE	
	EX	DE, HL	
	JR	0825, LD-CONT-2	

Acum se evaluează dacă sînt suficiente locații de memorie pentru noul bloc de informații.

0819 LD-CONT-1	LD	L, (IX-06)	Se încarcă mărimea 'program + variabile sau spațiu' existente.
	LD	M, (IX-05)	
	EX	DE, HL	
	BCF		Salt înainte dacă nu se cer prea multe locații, ținînd cont de gradul prezent de folosire al memoriei.
	SBC	HL, DE	
	JR	C, 082E, LD-DATA	

Se execută actualul test pentru locații.

0825 LD-CONT-2	LD	DE, +0005	Se acceptă deasupra un spațiu de cinci octeți.
	ADD	HL, DE	Se mută rezultatul în registrul pereche BC și se execută testul.
	LD	B, R	
	LD	C, L	
	CALL	1F05, TEST-ROOM	

Acum se prezintă partea cu încărcarea unei porțiuni.

082E LD-DATA	POP	HL	Se readuce 'indicatorul'.
	LD	A, (IX+00)	Salt înainte dacă se încarcă un program BASIC.
	AND	A	
	JR	Z, 0873, LD-PROG	
	LD	A, H	Salt înainte dacă se încarcă o porțiune nouă
	OR	L	
	JR	Z, 084C, LD-DATA-1	
	DEC	HL	Se încarcă 'lungimea' unei porțiuni existente prin adunarea octeților din spațiul variabilelor.
	LD	B, (HL)	Se indică vechiul nume.
	DEC	HL	Se adună trei biți la lungime - unul pentru nume și doi pentru 'lungime'.
	LD	C, (HL)	Se salvează registrul pereche IX pînă cînd se cere 'vechea' porțiune.
	DEC	HL	
	INC	BC	
	INC	BC	
	INC	BC	
	LD	(X-PT), IX	
	CALL	19E8, RECLAIM-2	
	LD	IX, (X-PTR)	

Spațiul este acum accesibil pentru noua porțiune - la sfîrșitul spațiului variabilelor prezente.

084C LD-DATA-1	LD	HL, (E-LINE)	Găsește indicatorul la sfîrșitul ariei variabilelor - al '80-lea bit'.
	DEC	HL	
	LD	C, (IX+0B)	Se încarcă 'lungimea' noii porțiuni.
	LD	B, (IX+0C)	Salvarea acestei 'lungimi'.
	PUSH	BC	Se adaugă trei biți - unul pentru nume și doi pentru 'lungime'.
	INC	BC	'IX+0E' din vechiul header dă numele porțiunii.
	INC	BC	Numele este salvat pînă cînd numărul potrivit de locații este făcut accesibil. De fapt 'BC' spații înaintea 'noilor 80 octeți'.
	INC	BC	Numele este introdus.
	LD	A, (IX-03)	'Lungimea' este adusă și cei
	PUSH	AF	
	CALL	1655, MAKE-ROOM	
	INC	HL	
	POP	AF	
	LD	(HL), A	
	POP	DE	

INC	HL	doi octeti sînt de asemenea
LD	(HL),E	introdusi.
INC	HL	
LD	(HL),D	
INC	HL	
PUSH	HL	HL indică acum prima locație
POP	IX	care urmează a fi înlocuită
SCF		cu informații de pe bandă.
LD	A,+FF	Această adresă este mutată în
JP	0802,LD-BLOCK	registru pereche IX;
		fanionul de transport este
		setat; este semnalat 'bloc de
		informații'; și blocul este
		încărcat.

Acum se prezintă partea cu încărcarea unui program BASIC cu variabilele lui.

0873 LD-PROG	EX DE,HL	Se salvează 'indicatorul
	LD HL,(E-LINE)	destinație'.
	DEC HL	Se setează adresa indicatorului
		de sfîrșit a porțiunii
		curente de variabile - cei
		'80-octeti'.
	LD (X-PTR),IX	Salvare temporară a
		registruului IX.
	LD C,(IX+0B)	Se aduce 'lungimea' noului
	LD B,(IX+0C)	bloc de informații.
	PUSH BC	Se păstrează o copie a
	CALL 19E5,RECLAIM-1	'lungimii' pînă cînd
	POP BC	programul prezent și aria
		variabilelor sînt cerute.
	PUSH HL	Se salvează indicatorul în
	PUSH BC	spațiul programului și
		lungimea noului bloc de
		informații.
	CALL 1655,MAKE-ROOM	Se accesează un număr
		suficient de locații pentru
		noul program și variabilele
		lui.
	LD IX,(X-PTR)	Se readuce conținutul
		registruului pereche IX.
	INC HL	De asemenea trebuie pregătit
	LD C,(IX+0F)	sistemul de variabile VARS
	LD B,(IX+10)	pentru noul program.
	ADD HL,BC	
	LD (VARS),HL	
	LD H,(IX+0E)	Dacă un număr de linie a fost
	LD A,H	specificat, atunci și acesta
	AND +C0	trebuie luat în considerare.
	JR NZ,08AD,LD-PROG-1	Salt dacă 'nici un număr';
	LD L,(IX+0D)	altfel se setează NENPPC &
	LD (NENPPC),HL	NSPPC.
	LD (NSPPC),+00	

Acum blocul de informații poate fi încărcat.

08AD LD-PROG-1	POP DE	Se aduce 'lungimea'.
	POP IX	Se aduce 'începutul'.
	SCF	Se setează 'LOAD' (încărcare).
	LD A,+FF	Se setează 'numai bloc de
		informații'.
	JP 0802,LD-BLOCK	Acum se încarcă (LOAD).

THE 'MERGE' CONTROL ROUTINE (RUTINA DE CONTROL UNIRE)

Sînt trei părți principale în această rutină.

- i. LOAD (încărcare) bloc de informații în spațiul de lucru.
- ii. MERGE (contopirea) liniilor noului program cu vechiul program.
- iii. MERGE (contopirea) noilor variabile cu cele vechi.

De aceea se începe cu încărcarea blocului de informații.

08B6 ME-CONTROL	LD C,(IX+0B)	Se aduce 'lungimea' blocului
	LD B,(IX+0C)	de informații.
	PUSH BC	Se salvează o copie a
		'lungimii'.
	INC BC	Acum se face 'lungime + 1'
	RST 0030,BC-SPACES	locații accesibile în spațiul
		de lucru.
	LD (HL),+80	Se pune un indicator de

EX	DE,HL	sfîrsit la extrema locatiei. Se mută indicatorul 'start' în registrul pereche HL.
POP	DE	Se aduce 'lungimea' inițială.
PUSH	HL	Se salvează o copie a lui 'start'.
PUSH	HL	Se setează registrul pereche
POP	IX	IX pentru încărcarea actuală.
SCF		Semnal 'LOAD' (încărcare).
LD	A,+FF	Semnal 'numai bloc de informatii'.
CALL	0802,LD-BLOCK	Se încarcă blocul de informatii.

Liniiile noului program sînt unite (MERGE) cu liniile vechiului program.

POP	HL	Se aduce 'începutul' noului program.
LD	DE,(PROG)	Se initializează DE cu 'începutul' ('start') vechiului program.

Se intră într-o buclă pentru partea cu liniile noului program.

08D2 ME-NEW-LP	LD	A,(HL)	Se aduce un număr de linie si
	AND	+C0	se testează.
	JR	NZ,08F0,ME-VAR-LP	Salt cînd s-au terminat toate liniile.

Intrare în interiorul buclei pentru partea cu liniile vechiului program.

08D7 ME-OLD-LP	LD	A,(DE)	Se încarcă octetul ce contine
	INC	DE	numărul maxim de linii si se
	CP	(HL)	compară.
	INC	HL	Salt înainte dacă nu este
	JR	NZ,08DF,ME-OLD-L1	coincidentă, dar în orice caz
	LD	A,(DE)	vor avansa ambele indicatoare
	CP	(HL)	Se repetă comparatia pentru
08DF ME-OLD-L1	DEC	DE	octetul care contine numărul
	DEC	HL	minim de linii.
	JR	NC,08EB,ME-NEW-L2	Acum se decrementează
			indicatorul.
			Salt înainte dacă s-a găsit
			locul corect pentru linia
			noului program.
	PUSH	HL	Altfel trebuie găsită adresa
	EX	DE,HL	începutului unei alte linii
	CALL	1928,NEXT-ONE	vechi.
	POP	HL	
	JR	08D7,ME-OLD-LP	Se ciclează bucla pentru
08EB ME-NEW-L2	CALL	092C,ME-ENTER	fiecare 'linie veche'.
	JR	08D2,ME-NEW-LP	Se introduce 'linia nouă' si
			se ciclează din nou bucla
			exterioară.

In mod analog, variabilele noului program sînt unite (MERGE) cu variabilele vechiului program.

O buclă este introdusă pentru rezolvarea fiecărei noi variabile, una cîte una.

08F0 ME-VAR-LP	LD	A,(HL)	Se aduce fiecare nume de
	LD	C,A	variabilă unul cîte unul si
			se testează.
	CP	+80	Revenire dacă au fost testate
	RET	Z	toate variabilele.
	PUSH	HL	Se salvează valoarea actuală
			a indicatorului curent.
	LD	HL,(VARS)	Se aduce VARS (pentru vechiul program).

Acum se intră într-o buclă internă pentru căutarea spatiului variabilelor existente.

08F9 ME-OLD-VP	LD	A,(HL)	Se aduce fiecare nume de
	CP	+80	variabilă si se testează.
	JR	Z,0923,ME-VAR-L2	Salt înainte îndată ce s-a
			găsit indicatorul de sfîrsit.
			(Face o adăugare).
	CP	C	Se compară numele (primul octet).

	JR	Z,0909,ME-OLD-V2	Salt înainte pentru a-l lua în considerare mai târziu; aici se va face revenire dacă se dovedește că numele nu coincid în totalitate.
0901 ME-OLD-V1	PUSH	BC	Se salvează noul nume al variabilei pînă cînd următoarea 'variabilă veche' este locată.
	CALL	192B,NEXT-ONE	Se readuce indicatorul în registrul pereche DE și se ciclază din nou bucla.
	POP	BC	
	EX	DE,HL	
	JR	0BF9,ME-OLD-VP	

Variabilele vechi și noi coincid prin respectarea primului octet, dar variabilele cu nume lung necesită o totală coincidență.

0909 ME-OLD-V2	AND	+EO	Se consideră doar bitii 7, 6 și 5.
	CP	+A0	Se acceptă toate tipurile de variabile, exceptînd 'variabilele cu nume lung'.
	JR	NZ,0921,ME-VAR-L1	Se face DE purtătorul primului caracter al 'noului nume'.
	POP	DE	Se salvează indicatorul 'vechiului nume'.
	PUSH	DE	
	PUSH	HL	

Se intră într-o buclă de comparare a literelor numelor lungi.

0912 ME-OLD-V3	INC	HL	Se actualizează indicatorii 'vechi' și 'nou'.
	INC	DE	Se compară cele două litere.
	LD	A,(DE)	
	CP	(HL)	
	JR	NZ,091E,ME-OLD-V4	Salt înainte dacă literele nu coincid.
	RLA		Se ciclază bucla pînă cînd se găsește 'ultimul caracter'.
	JR	NC,0912,ME-OLD-V3	Se aduce indicatorul la începutul 'vechiului' nume și salt înainte - reușit.
	POP	HL	Se aduce indicatorul și salt înapoi - nereușit.
	JR	0921,ME-VAR-L1	
091E ME-OLD-V4	POP	HL	
	JR	0901,ME-OLD-V1	

Se vine aici dacă s-a găsit coincidență.

0921 ME-VAR-L1	LR	A,+FF	Se reînlocuiește 'înlocuire' variabilă.
----------------	----	-------	---

Si se vine aici dacă nu s-a găsit coincidență. (O reținere +80 - variabilă ce trebuie 'adăugată').

0923 ME-VAR-L2	POP	DE	Se aduce indicatorul 'noului' nume.
	EX	DE,HL	Schimbare între registre.
	INC	A	Fanionul zero va fi setat dacă va fi o 'înlocuire'; va fi resetat în cazul unei 'adăugări'.
	SCF		Semnalare 'tratare variabile'.
	CALL	092C,ME-ENTER	Acum se face intrarea.
	JR	0BF0,ME-VAR-LP	Se ciclază bucla pentru a considera următoarea nouă variabilă.

THE 'MERGE A LINE OR A VARIABLE' SUBROUTINE (SUBROUTINA 'UNIRE LINIE SAU VARIABILA')

Această subrutină este introdusă cu următorii parametri:

Fanionul de transport	resetat	- unire linie BASIC
	setat	- unire variabilă
Fanionul zero	resetat	- va fi o 'adăugare'
	setat	- va fi o 'înlocuire'
Registrul pereche HL		- punctul de început al unei noi intrări
Registrul pereche DE		- punctul în care se va face unirea

092C NE-ENTER	JR	NZ,093E,NE-ENT-1	Salt dacă se tratează o 'adaugare'.
	EX	AF,A'F'	Se salvează fanioanele.
	LD	(X-PTR),HL	Se salvează indicatorul 'nou' pînă cînd vechea linie sau variabilă se reface.
	EX	DE,HL	
	CALL	1988,NEXT-ONE	
	CALL	19E8,RECLAIM-2	
	EX	DE,HL	
	LD	HL,(X-PTR)	
	EX	AF,A'F'	Reducere fanioane.
Acum se poate face o nouă intrare.			
093E NE-ENT-1	EX	AF,A'F'	Se salvează fanioanele.
	PUSH	DE	Se face o copie a indicatorului 'destinație'.
	CALL	1988,NEXT-ONE	Găsește lungimea 'noul' variabile/linie.
	LD	(X-PTR),HL	Se salvează indicatorul 'noul' variabile/linie.
	LD	HL,(PROG)	Se aduce PROG - pentru a evita deteriorarea.
	EX	(SP),HL	Se salvează PROG în stivă și se aduce 'noul' indicator.
	PUSH	BC	Se salvează lungimea.
	EX	AF,A'F'	Se refac fanioanele.
	JR	C,0955,NE-ENT-2	Salt înainte dacă se adaugă o nouă variabilă.
	DEC	HL	Se adaugă o nouă linie înaintea locației 'destinație'.
	CALL	1655,MAKE-ROOM	Se pregătește locația pentru noua linie.
	INC	HL	
0955 NE-ENT-2	JR	0958,NE-ENT-3	Salt înainte.
	CALL	1655,MAKE-ROOM	Se pregătește locația pentru noua variabilă.
0958 NE-ENT-3	INC	HL	Se adresează prima locație.
	POP	BC	Se reface lungimea.
	POP	DE	Regăsește PROG și îl păstrează la locul adecvat.
	LD	(PROG),DE	De asemenea, se aduce 'noul' indicator.
	LD	DE,(X-PTR)	
	PUSH	BC	Se salvează din nou lungimea și 'noul' pointer.
	PUSH	DE	Se schimbă indicatorii și se copiază 'noua' variabilă / linie în locația rezervată.
	EX	DE,HL	
	LDIR		

'Noua' variabilă/linie trebuie acum mutată din spațiul de lucru.

	POP	HL	Se aduce indicatorul 'nou'.
	POP	BC	Se aduce lungimea.
	PUSH	DE	Se salvează indicatorul 'vechi'. (Se adresează locația după 'adunarea' variabilei/liniei.)
	CALL	19E8,RECLAIM-2	Se mută variabilă/linie din spațiul de lucru.
	POP	DE	Revenire cu indicatorul 'vechi' în registrul pereche
	RET		DE.

THE 'SAVE' CONTROL ROUTINE (RUTINA DE CONTROL SALVARE)

Operația de salvare (SAVE) a programului sau a unui bloc de informații este foarte utilizată.

0970 SA-CONTRL	PUSH	HL	Se salvează 'indicatorul'.
	LD	A,+FB	Se asigură că este deschis canalul K.
	CALL	1601,CHAN-OPEN	Se semnalează 'primul mesaj'.
	XOR	A	Se tipărește mesajul - Se porneste banda, apoi se apasă oricare tastă.
	LD	DE,+09A1	Semnalizare 'ecranul va cere să fie sters'.
	CALL	0C0A,PO-NSG	Se așteaptă apăsarea unei taste.
	SET	S,(TV-FLAG)	
	CALL	1SD4,WAIT-KEY	

Atita timp cât o tastă este apăsată, 'header'-ul este salvat.

<pre>PUSH IX LD B, +0011 XOR A CALL 04C2, SA-BYTES</pre>	<p>Se salvează adresa de bază a 'header'-ului în stiva calculatorului. Urmează să fie salvati 17 octeți. Se semnalează 'este un header'. Se trimite 'header'-ul; el este precedat de un octet 'tip' și urmat de un octet 'paritate'.</p>
--	--

Acum urmează o scurtă întârziere înaintea salvării programului / blocului de informații.

<pre>0991 SA-1-SEC POP IX LD B, +32 HALT DNZ 0991, SA-1-SEC LD E, (IX+08) LD B, (IX+0C) LD A, +FF POP IX JP 04C2, SA-BYTES</pre>	<p>Revenire indicator în 'header'. Întârzierea este pentru 50 de întreruperi, aceasta reprezentând o secundă. Se aduce lungimea blocului de informații care trebuie salvat. Se semnalează 'bloc de informații'. Se aduce 'indicatorul de început al blocului' și se salvează blocul.</p>
--	--

THE CASSETTE MESSAGES (MESAJELE CASETEI)

Fiecare mesaj este dat cu ultimul caracter invertit (+80 hex.).

<pre>09A1 DEFB +80 09A2 DEFM 09C1 DEFM 09C8 DEFM 09DA DEFM 09EC DEFM</pre>	<ul style="list-style-type: none"> - Octetul initial este sărit. - Pornire înregistrare, urmată de apăsarea oricărei taste. - 'carriage return' - Program; - 'carriage return' - Zonă număr; - 'carriage return' - Zonă caracter; - 'carriage return' - Octeți.
--	---

THE SCREEN PRINTER HANDLING ROUTINES (TRATAREA RUTINELOR DE AFISARE PE ECRAN)

THE 'PRINT - OUT' ROUTINES

Toate afisările din partea principală a ecranului, din partea de jos a ecranului și afisarea sint tratate de acest set de rutine.

Rutina PRINT-OUT este introdusă cu registrul A, conținând codul pentru controlul caracterului, caracterul bun de afisat sau un simbol.

09F4 PRINT-OUT	CALL	0B03,PO-FETC	Pozitia actuală de afisat.
	OP	+20	Salt dacă codul reprezintă un caracter afisabil.
	JP	NC,0AD9,PO-ABLE	Se afisează semn de întrebare pentru coduri din intervalul +00 - +05.
	CP	+06	Si de asemenea pentru coduri din intervalul +18 - +1F.
	JR	C,0A69,PO-QUEST	
	CP	+18	
	JR	NC,0A69,PO-QUEST	
	LD	HL,+0A08	Baza tabelului 'control'.
	LD	E,A	Mutare cod în registrul pereche DE.
	LD	D,+00	Indexare în tabel și se aduce deplasamentul.
	ADD	HL,DE	Adunare deplasament și salt indirect la subrutina adecvată.
	LD	E,(HL)	
	ADD	HL,DE	
	PUSH	HL	
	JP	0B03,PO-FETCH	

THE 'CONTROL CHARACTER' TABLE (TABELUL 'CARACTERELOR DE CONTROL')

adresa	deplasament	caracter	adresa	deplasament	caracter
0A11	4E	afisare virgulă	0A1A	4F	nefolosit
0A12	57	EDIT	0A1B	5F	control cerneală
0A13	10	cursor stînga	0A1C	5E	control hîrtie
0A14	29	cursor dreapta	0A1D	5D	control lumină
0A15	54	cursor jos	0A1E	5C	control strălucire
0A16	53	cursor sus	0A1F	5B	control invers
0A17	52	DELETE (sterge)	0A20	5A	supracontrol
0A18	37	ENTER	0A21	54	control AT
0A19	50	nefolosit	0A22	53	control TAB

THE 'CURSOR LEFT' SUBROUTINE (SUBRUTINA 'CURSOR STINGA')

Subrutina este introdusă prin registrul B, care conține numărul liniei actuale, și prin registrul C, care conține numărul coloanei următoare.

0A23 PO-BACK-1	INC	C	Mutare la stînga cu o coloană.
	LD	A,+22	Se acceptă schimbarea numai în partea stîngă.
	CP	C	
	JR	NZ,0A3A,PO-BACK-3	Dacă folosește afisarea, atunci salt înainte.
	BIT	1,(FLAG)	Ridicare cu o linie.
	JR	NZ,0A38,PO-BACK-2	Setare valoare coloană.
	INC	B	Testare față de linia maximă.
	LD	C,+02	Notă: Aceasta trebuie să fie +19.
	LD	A,+18	Se acceptă schimbul doar în partea de sus a ecranului.
	CP	B	Nu s-a acceptat, se coboară o linie.
	JR	NZ,0A3A,PO-BACK-3	Setare coloană în partea stîngă.
	DEC	B	Revenire indirectă prin CL-SET & PO-STORE.
0A38 PO-BACK-2	LD	C,+21	
0A3A PO-BACK-3	JP	0DD9,CL-SET	

THE 'CURSOR RIGHT' SUBROUTINE (SUBRUTINA 'CURSOR DREAPTA')

Această subrutină execută o operație identică formulării BASIC - PRINT OVER 1;CHR# 32;-.

0A3D PO-RIGHT	LD	A,(P-FLAG)	Se aduce P-FLAG și se salvează în stivă.
	PUSH	AF	Setare P-FLAG la OVER 1.
	LD	(P-FLAG),+01	În A se încarcă 'spatiu'.
	LD	A,+20	Afisare caracter.
	CALL	0165,PO-CHAR	Se aduce vechea valoare a lui P-FLAG.
	POP	AF	Terminare.
	LD	(P-FLAG),A	Notă: Programatorul a uitat
	RET		

să iasă prin PO-STORE.

THE 'CARRIAGE RETURN' SUBROUTINE (SUBRUTINA 'CARRIAGE RETURN')

Dacă tipărirea se face pe ecran (afisare), atunci se face un test pentru defilare pe ecran ('scroll?') înainte de a decremента numărul liniei.

0A4F PO-ENTER	BIT	1, (FLAG)	Salt înainte dacă se tratează
	JP	NZ, OECD, COPY-BUFF	tipărire.
	LD	C, +21	Setare coloană în partea
			stingă.
	CALL	0C55, PO-SCR	Defilare, dacă este necesar.
	DEC	B	Acum se coboară o linie.
	JP	0B59, CL-SET	Se execută acum o revenire
			indirectă, prin CL-SET &
			PO-STORE.

THE 'PRINT COMMA' SUBROUTINE (SUBRUTINA 'AFISARE VIRGULA')

Valoarea coloanei actuale este schimbată și registrul A este setat cu +00 (pentru TAB 0) sau +10 (pentru TAB 16).

0A5F PO-COMMA	CALL	0B03, PO-FETCH	De ce se reia?
	LD	A, C	Numărul coloanei actuale.
	DEC	A	Se mută în partea dreaptă cu
	DEC	A	două coloane și apoi se
			testează.
	AND	+10	Registrul A va conține +00
			sau +10.
	JR	0AC3, PO-FILL	Se iasă prin PO-FILL.

THE 'PRINT A QUESTION MARK' SUBROUTINE (SUBRUTINA 'AFISAREA UNUI SEMN DE INTREBARE')

Un semn de întrebare se afișează întotdeauna când se face o încercare de afișare a unui cod eronat.

0A69 PO-QUEST	LD	A, +3F	Caracterul '?'. .
	JR	0AD9, PO-ABLE	Se afișează acest caracter în
			locul celui nevalidat pentru
			afișare.

THE 'CONTROL CHARACTERS WITH OPERANDS' ROUTINE (RUTINA 'CONTROLUL CARACTERELOR CU OPERANZI')

Controlul caracterelor de la INK (cerneală) la OVER necesită un singur operand, în timp ce controlul caracterelor AT & TAB necesită să fie urmate de doi operanzi.

Această rutină duce la salvarea codului de control și caracterului în TVDATA-lo, primul operand în TVDATA-hi sau în registrul A dacă este cerut un singur operand, și al doilea operand în registrul A.

0A6D PO-TV-2	LD	DE, +0A87	Salvarea primului operand în
	LD	(TVDATA-hi), A	TVDATA-hi și schimbarea
	JR	0A80, PO-CHANGE	adresei rutinei de 'iesire'
			în PO-CONT (+0A87).

Aici se va intra când se tratează caracterele AT & TAB.

0A75 PO-2-OPER	LD	DE, +0A6D	Codul caracterului se
	JR	0A7D, PO-TV-1	salvează în TVDATA-lo și
			adresa rutinei de 'iesire'
			se schimbă în PO-TV-2
			(+0A6D).

Aici se va intra când se tratează fiecare culoare - INK la OVER.

0A7A PO-1-OPER	LD	DE, +0A87	Rutina 'iesire' va fi
			schimbată în PO-CONT (+0A87).
0A7D PO-TV-1	LD	(TVDATA-lo), A	Salvare cod caracter control.

Adresa rutinei curente de 'iesire' este schimbată pentru un timp.

0A80 PO-CHANGE	LD	HL, (CURCHL)	Registrul HL va conține
			adresa rutinei 'iesire'.
	LD	(HL), E	Se introduce noua adresa a
	INC	HL	rutinei 'iesire', forțind
	LD	(HL), D	astfel ca următorul cod de
	RET		caracter să fie considerat
			operand.

Odată ce operanzii au fost adunați (strinși) rutina continuă.

0A87 PO-CONT	LD	DE,+09F4	Se rememorează adresa originală pentru PRINT-OUT (+09F4).
	CALL	0A80,PO-CHANGE	
	LD	HL,(TVDATA)	Se aduce codul de control si primul operand dacă sînt doi operanzi.
	LD	D,A	'Ultimul' operand si codul de control sînt mutate.
	LD	A,L	Salt înainte dacă se tratează
	CP	+16	INK to OVER.
	JP	C,2211,CO-TEMPS	Salt înainte dacă se tratează
	JR	NZ,0AC2,PO-TAB	TAB.

Acum se lucrează cu caracterul de control AT.

	LD	B,H	Numărul liniei.
	LD	C,D	Numărul coloanei.
	LD	A,+1F	Se schimbă numărul coloanei, astfel încît +00 - +1F devine +1F - +00.
	SUB	C	Trebuie să fie în ordine.
	JR	C,0AAC,PO-AT-ERR	Se adună la deplasament
	ADD	A,+02	pentru a pune în C +21 - +22.
	LD	C,A	Salt înainte dacă se tratează
	BIT	1,(FLAGS)	afisare.
	JR	NZ,0ABF,PO-AT-SET	Se schimbă numărul liniei, astfel încît +00 - +15 devine +16 - +01.
	LD	A,+16	
	SUB	B	
0AAC PO-AT-ERR	JP	C,1E9F,REPORT-3	Dacă condiția este îndeplinită, salt înainte.
	INC	A	Sirul +16 - +01 devine +17 - +02.
	LD	B,A	Si acum +18 - +03.
	INC	B	Dacă afisarea se face în
	BIT	0,(TV-FLAG)	partea de jos a ecranului se
	JP	NZ,0C55,PO-SCR	consideră dacă defilarea este
	CP	(DF-SZ)	necesară.
	JP	C,0C86,REPORT-5	Se dă prezentarea 5 - afară
0ABF PO-AT-SET	JP	0DD9,CL-SET	din ecran, dacă se cere.
			Revenire prin CL-SET&PO-STORE

Si cu caracterul de control TAB.

0AC2 PO-TAB	LD	A,H	Se aduce primul operand.
0AC3 PO-FILL	CALL	0B03,PO-FETCH	Pozitia de afisare actuală.
	ADD	A,C	Se adună valoarea coloanei
	DEC	A	curente.
	AND	+1F	Șasește cîte spații modulo
	RET	Z	32 sînt cerute si revenire
	LD	D,A	dacă rezultatul este zero.
	SET	0,(FLAGS)	Se folosește D ca si contor.
0AD0 PO-SPACE	LD	A,+20	Se 'suprapun' spațiile de
	CALL	0C3B,PO-SAVE	fond.
	DEC	D	Se afisează 'numărul D' de
	JR	NZ,0AD0,PO-SPACE	spații.
	RET		

Acum se termină.

PRINT THE CHARACTER CODES (AFISAREA CODULUI CARACTERULUI)

Caracterul (sau caracterele) cerute sînt afisate prin apelarea PO-ANY urmat de PO-STORE.

0AD9 PO-ABLE	CALL	0B24,PO-ANY	Se tipăreste caracterul (caracterele) si se continuă în PO-STORE.
--------------	------	-------------	---

THE 'POSITION STORE' SUBROUTINE (SUBRUTINA 'REZERVARE POZITIE')

Noua valoare a pozitiei 'linie & coloană' si adresa 'pixelului' sînt stocate în sistemul propriu de variabile.

0ADC PO-STORE	BIT	1,(FLAGS)	Salt înainte pentru tratare
	JR	NZ,0AFC,PO-ST-PR	afisare.
	BIT	0,(TV-FLAG)	Salt înainte dacă se tratează
	JR	NZ,0AFO,PO-ST-E	partea de jos a ecranului.
	LD	(S-POSN),BC	Se salvează valorile care
	LD	(DF-CC),HL	descriu partea principală a
	RET		ecranului. Apoi revenire.
0AFO PO-ST-E	LD	(S-POSNL),BC	Se salvează valorile care

	LD	(ECHO-E),BC	descriu partea de jos a
	LD	(DF-CCL),HL	ecranului.
	RET		Apoi revenire.
0AFC PO-ST-PR	LD	(P-POSN),C	Se salvează valorile care
	LD	(PR-CC),HL	descriu bufferul de afisare.
	RET		Apoi revenire.

THE 'POSITION FETCH' SUBROUTINE (SUBRUTINA 'ADUCERE POZITIE')

Parametrii actuali ai pozitiei sînt adusi din sistemul propriu de variabile.

0B03 PO-FETCH	BIT	1,(FLAGS)	Salt înainte dacă se tratează
	JR	NZ,0B1D,PO-F-PR	afisare.
	LD	BC,(S-POSN)	Se aduc valorile ce descriu
	LD	HL,(DF-CC)	partea principală a ecranului
	BIT	0,(TV-FLAG)	si revenire dacă aceasta a
	RET	Z	fost intentia.
	LD	BC,(S-POSNL)	Altfel, se aduc valori care
	LD	HL,(PR-CC)	descriu partea de jos a
	RET		ecranului.
0B1D PO-F-PR	LD	C,(P-POSN)	Se aduc valorile care descriu
	LD	HL,(PR-CC)	bufferul de afisare.
	RET		

THE 'PRINT ANY CHARACTER(S)' SUBROUTINE (SUBRUTINA 'AFISARE ORICARUI CARACTER (CARACTERE)')

Codurile caracterelor actuale, codurile simbolurilor, codurile grafice definite de utilizator si codurile grafice sînt prelucrate separat.

0B24 PO-ANY	CP	+80	Salt înainte pentru cod
	JR	C,0B65,PO-CHAR	caracter actual.
	CP	+90	Salt înainte pentru cod semn
	JR	NC,0B52,PO-T&UDG	sau cod semn grafic definit
			de utilizator (UDG).
	LD	B,A	Transfer cod grafic.
	CALL	0B38,PO-GR-1	Formare formă grafică.
	CALL	0B03,PO-FETCH	HL a fost modificat, deci se
			'reface'.
	LD	DE,+5C92	Se încarcă în DE începutul
			forme grafice; acesta este
			MENBOT.
	JR	0B7F,PO-ALL	Salt înainte pentru tipărire
			cod grafic.

Caracterele grafice sînt construite într-o manieră Ad Hoc în spațiul de memorie al calculatorului; acestea sînt MEN-0 & MEN-1.

0B38 PO-GR-1	LD	HL,+5C92	Incarcă MENBOT.
	CALL	0B3E,PO-GR-2	Ca urmare, se apelează
			subrutina următoare de două
			ori.
0B3E PO-GR-2	RR	B	Se determină bitul 0 (si mai
	SBC	A,A	tîrziu bitul 2) al codului
			grafic.
	AND	+0F	Registrul A va contine +00
			sau +0F, în functie de
			valoarea bitului în cadrul
			codului.
	LD	C,A	Se salvează rezultatul în C.
	RR	B	Se determină bitul 1 (si mai
	SBC	A,A	tîrziu bitul 3) al codului
			grafic.
	AND	+F0	Registrul A va contine +00
			sau +F0.
	OR	C	Cele două rezultate sînt
			combinat.
	LD	C,+04	Registrul A va contine o
0B4C PO-GR-3	LD	(HL),A	jumătate a caracterului
	INC	HL	format, ce va fi utilizat de
			patru ori.
	DEC	C	Aceasta se face pentru partea
	JR	NZ,0B4C,PO-GR-3	superioară a caracterului si
	RET		apoi pentru partea sa
			inferioară.

Codurile semnelor si codurile semnelor grafice definite de către utilizator sînt acum separate.

0B52 PO-T&UDG	SUB	+A5	Salt înainte cu codul grafic.
---------------	-----	-----	-------------------------------

	JR	NC,0B5F,PO-T	
	ADD	A,+15	Codurile UDG sînt acum +00 - +0F.
	PUSH	BC	Salvarea valorii pozitiei actuale în stiva calculatorului.
	LD	BC,(UDG)	Se aduce adresa de bază a zonei UDG si salt înainte.
0B5F PO-T	JR	0B6A,PO-CHAR-2	Acum se face afisare semn si revenire prin PO-FETCH.
	CALL	0C10,PO-TOKENS	
	JP	0B03,PO-FETCH	

Este identificată forma caracterului cerut.

0B65 PO-CHAR	PUSH	BC	Este salvată pozitia curentă.
	LD	BC,(CHARS)	Se aduce adresa de bază a spatiului caracterului
0B6A PO-CHAR-2	EX	DE,HL	Este salvată adresa la care se afisează.
	LD	HL,+5C3B	Aceasta este FLAGS.
	RES	0,(HL)	Permite spatii de fond.
	CP	+20	Salt înainte dacă caracterul nu este 'spatiu'.
	JR	NZ,0B76,PO-CHAR-3	Dar daca este 'se suprapune'.
0B76 PO-CHAR-3	SET	0,(HL)	Acum se trece codul caracterului în registrul pereche HL.
	LD	H,+00	De fapt, codul caracterului este inmultit cu 8.
	LD	L,A	
	ADD	HL,HL	Adresa de bază a formei caracterului este găsită.
	ADD	HL,HL	Se aduce pozitia actuală si adresa de bază se trece în registrul pereche HL.
	ADD	HL,HL	
	ADD	HL,BC	
	POP	BC	
	EX	DE,HL	

THE 'PRINT ALL CHARACTERS' SUBROUTINE (SUBROUTINA 'TIPARIREA TUTUROR CARACTERELOR')

Această subrutină este folosită pentru afisarea tuturor '8*8' biti caracter. La intrare, registrul pereche DE contine adresa bazei caracterului format, registrul pereche HL contine adresa destinatiei si registrul pereche BC contine valoarea curentă 'linie & coloană'.

0B7F PR-ALL	LD	A,C	Se aduce numărul coloanei.
	DEC	A	Mutare cu o coloană la dreapta.
	LD	A,+21	Salt înainte doar dacă se indică o nouă linie.
	JR	NZ,0B93,PR-ALL-1	Coverire cu o linie.
	DEC	B	Numărul coloanei este +21.
	LD	C,A	Salt înainte dacă se tratează ecranul.
	BIT	1,(FLAGS)	Salvarea adresei de bază pînă cînd se golește bufferul de afisare.
	JR	Z,0B93,PR-ALL-1	Se copiază noul număr al coloanei.
	PUSH	DE	Testare dacă o linie nouă urmează să fie folosită. Dacă da, se cercetează dacă display-ul cere defilare.
	CALL	0ECD,COPY-BUFF	
	POP	DE	
	LD	A,C	
0B93 PR-ALL-1	CP	C	
	PUSH	DE	
	CALL	Z,0C55,PO-SCR	
	POP	DE	

Se prezintă acum starea 'INVERSE & OVER'.

	PUSH	BC	Se salvează valoarea pozitiei si a adresei destinatie în stiva calculatorului.
	PUSH	HL	Se aduce P-FLAG si se citește bitul 0.
	LD	A,(P-FLAG)	Se pregătește 'OVER-mask' în registrul B; aceasta este OVER 0 = +00 & OVER 1 = +FF.
	LD	B,+FF	
	RRA		
	JR	C,0BA4,PR-ALL-2	
	INC	B	
0BA4 PR-ALL-2	RRA		Se citește bitul 2 din P-FLAG si se pregătește 'INVERSE-mask' în registrul C; aceasta este INVERSE 0 = +00 & INVERSE 1 = +FF.
	RRA		
	SBC	A,A	
	LD	D,A	Se pune registrul A în așteptarea contorului 'pixel'
	LD	A,+08	
	AND	A	

	BIT	1, (FLAGS)	linie' si se sterge fanionul de transport.
	JR	Z, 0BB6, PR-ALL-3	Salt înainte dacă se trateaza ecranul.
	SET	1, (FLAG82)	Se semnalează 'bufferul de afisare nu mai este gol'.
	SCF		Se pune fanionul de transport să arate că s-a utilizat printer-ul.
0BB6 PR-ALL-3	EX	DE, HL	Se schimbă adresa de destinatie cu adresa de bază înainte de a intra în buclă.

Caracterul poate fi acua tipărit. Se fac opt treceri prin buclă - una pentru fiecare 'linie-pixel'.

0BB7 PR-ALL-4	EX	AF, A'F'	Fanionul de transport este setat cînd se utilizează afisarea. Acest flag se salvează în F'.
	LD	A, (DE)	Se aduce 'linie - pixel' existentă.
	AND	B	Se foloseste 'OVER-mask' si se compară rezultatul cu 'linie-pixel' a caracterului format.
	XOR	(HL)	In final se consideră 'INVERS-mask'.
	XOR	C	Se introduce rezultatul.
	LD	(DE), A	Se aduce fanionul de imprimare si salt înainte, dacă este cerut.
	EX	AF, A'F'	Se incrementează adresa destinatie.
	JR	C, 0BB3, PR-ALL-6	Se incrementează 'linie - pixel' a caracterului format.
	INC	D	Se decrementează contorul si se buclează înapoi numai dacă este zero.
0BC1 PR-ALL-5	INC	HL	
	DEC	A	
	JR	NZ, 0BB7, PR-ALL-4	

Odată ce s-a afisat caracterul, octetul atribut se setează la cerere.

	EX	DE, HL	Registrul H va contine adresa superioară corectă pentru zona caracterului.
	DEC	H	Se setează octetul atribut numai dacă se tratează ecranul.
	BIT	1, (FLAGS)	Se reface adresa destinatie initială si valorile pozitiei.
	CALL	Z, 0BDB, PO-ATTR	Se decrementează numărul coloanei si se incrementează adresa destinatie înainte de revenire.
	POP	HL	
	POP	BC	
	DEC	C	
	INC	HL	
	RET		

Cînd se foloseste imprimanta, adresa destinatie trebuie ridicată prin incrementare cu +20.

0BB3 PO-ALL-6	EX	AF, A'F'	Salvare fanion afisare.
	LD	A, +20	Se încarcă valoarea cerută pentru incrementare.
	ADD	A, E	Se adună valoarea si rezultatul se pune înapoi în registrul E.
	LD	E, A	Se aduce fanionul.
	EX	AF, A'F'	Salt înapoi în buclă.
	JR	0BC1, PR-ALL-5	

THE 'SET ATTRIBUTE BYTE' SUBROUTINE (SUBROUTINA 'SETARE OCTET ATRIBUT')
Octetul atribut adecvat este identificat si încărcat. Noua valoare este formată prin modificarea valorii vechi, ATTR-T, MASK-T si P-FLAG. In final noua valoare este copiată în spatiul atribuit.

0BDE PO-ATTR	LD	A, H	Octetul superior al adresei destinatie este divizat cu 8 si adunat cu +03 pentru a determina care treime de ecran este adresată; aceasta
	RRCA		
	RRCA		
	RRCA		
	AND	+03	

	OR	+58	este 00, 01 sau 02.
	LD	H,A	Octetul superior pentru
	LD	DE, (ATTR-T)	spatiul atribuit este acum
	LD	A, (HL)	format.
	XOR	E	D contine ATTR-T si E contin
	AND	D	MASK-T.
	XOR	E	Vechea valoare
	BIT	6, (P-FLAG)	caracteristică.
	JR	Z, OBFA, PO-ATTR-1	Valorile lui ATTR-T si MASK-
	AND	+C7	sint luate in calcul.
	BIT	2, A	Salt inainte doar dacă se
	JR	NZ, OBFA, PO-ATTR-1	lucrează cu PAPER 9.
	XOR	+38	Vechea culoare a hîrtiei este
OBFA PO-ATTR-1	BIT	4, (P-FLAG)	ignorată, si după cum
	JR	Z, OC08, PO-ATTR-2	culoarea cernelii este
	AND	H8	luminoasă sau întunecată,
	BIT	5, A	noua culoare a hîrtiei va fi
	JR	NZ, OC08, PO-ATTR-2	negru (000) sau alb (111).
	XOR	+07	Salt inainte numai dacă se
OC08 PO-ATTR-2	LD	(HL), A	lucrează cu INK 9.
	RET		Vechea culoare de cerneală

THE 'MESSAGE PRINTING' SUBROUTINE (SUBROUTINA 'TIPARIRE MESAJ')

Această subrutină este folosită pentru tipărirea mesajelor și a semnelor. Registrul A contine 'numărul intrării' mesajului sau semnului într-un tabel. Registrul pereche DE contine adresa de bază a tabelului.

OC0A PO-MSG	PUSH	HL	Octetul superior al ultimei
	LD	H, +00	intrări în stiva
	EX	(SP), HL	calculatorului este pus pe
	JR	OC14, PO-TABLE	zero, astfel încît să
			suprapună spații (vezi mai
			jos).
			Salt înainte.

Aici se intră dacă se însiră un cod de semn.

OC10 PO-TOKENS	LD	DE, +0095	Adresa de bază a tabelului
	PUSH	AF	de semn.
			Se salvează codul în stivă.
			(Ordinea +00 - +5A; RND-COPY).

Se cercetează tabelul și intrarea corectă este afisată.

OC14 PO-TABLE	CALL	OC41, PO-SEARCH	Localizare intrare cerută.
	JR	C, OC22, PO-EACH	Afisare mesaj/semn.
	LD	A, +20	Se va afisa un 'spatiu'
	BIT	0, (FLAGS)	inaintea mesajului/semnului
	CALL	Z, OC3B, PO-SAVE	cerut.

Caracterele mesajului/semnului sînt afisate unul după altul.

OC22 PO-EACH	LD	A, (DE)	Încărcare cod.
	AND	+7F	Se anulează orice 'bit
	CALL	OC3B, PO-SAVE	invertit'.
	LD	A, (DE)	Afisare caracter.
	INC	DE	Se încarcă din nou codul.
	ADD	A, A	Avans indicator.
	JR	NZ, OC22, PO-EACH	'Bitul invertit' se încarcă
			în fanionul de transport și
			semnalează sfîrsitul
			mesajului/semnului; în caz
			contrar, salt înapoi.

Acum se consideră dacă este necesar un 'spatiu următor'.

	POP	DE	Pentru mesaje- D contine +00;
			pentru semne- D contine +00
			- +5A.
	CP	+48	Salt inainte dacă ultimul
	JR	Z, OC35, PO-TRSP	caracter a fost '\$'.

	CP	+82	Revenire dacă ultimul
	RET	C	caracter a fost oricare altul
			înainte de 'A'.
0C35 PO-TRSP	LD	A,D	Se examinează valoarea din D
	CP	+03	și revenire dacă se indică un
	RET	C	mesaj, RND, INKEY\$ sau PI.
	LD	A,+20	Toate celelalte cazuri vor
			cere un 'spatiu următor'.

THE 'PO-SAVE' SUBROUTINE (SUBROUTINA 'PO-SAVE')

Această subrutină permite scrierea 'recursivă' a caracterelor. Registrii necesari sînt salvati pînă cînd se apelează 'PRINT-OUT'.

0C3B PO-SAVE	PUSH	DE	Se salvează registrul
			pereche DE.
	EXX		Se salvează HL & BC.
	RST	0010,PRINT-A-1	Afisare caracter singular.
	EXX		Refacere HL & BC.
	POP	DE	Refacere DE.
	RET		Terminare.

THE 'TABLE SEARCH' SUBROUTINE (SUBROUTINA 'CAUTARE TABEL')

Subrutina revine cu registrul pereche DE indicînd caracterul initial al intrării cerute și fanionul de transport este resetat dacă trebuie considerat un 'spatiu de fond'.

0C41 PO-SEARCH	PUSH	AF	Se salvează 'număr intrare'.
	EX	DE,HL	HL contine acum adresa de
			bază.
0C44 PO-STEP	INC	A	Se face ordinul +01 ?
	BIT	7, (HL)	Se așteaptă un 'caracter'
	INC	HL	invers.
	JR	Z,0C44,PO-STEP	
	DEC	A	Se caută intrările pînă se
	JR	NZ,0C44,PO-STEP	găsește cea corectă.
	EX	DE,HL	DE indică caracterul initial.
	POP	AF	Incercare 'număr intrare' și
	CP	+20	revenire cu transportul setat
	RET	C	pentru primele 32 de intrări.
	LD	A, (DE)	Oricum, dacă caracterul
	SUB	+41	initial este o literă ar
	RET		ar putea fi nevoie de un
			spatiu de fond.

THE 'TEST FOR SCROLL' SUBROUTINE (SUBROUTINA 'TEST PENTRU DEFILARE')

Această subrutină este apelată cînd este necesară defilarea ecranului. Aceasta se poate întîmpla în trei cazuri i. cînd se tratează un caracter 'carriage return'; ii. cînd se folosește AT într-o linie INPUT; iii. cînd linia curentă este plină și trebuie folosită linia următoare.

La intrare registrul Z va contine numărul liniei după test.

0C55 PO-SCR	BIT	1, (FLAGS)	Revenire imediată dacă
	RET	NZ	afisarea a fost utilizată.
	LD	DE,+0DD9	Profundărcarea stivei
	PUSH	DE	calculatorului cu adresa lui
			'CL-SET'.
	LD	A,Z	Se transferă numărul liniei.
	BIT	0, (TV-FLAG)	Salt înainte dacă se
	JP	NZ,0B02,PO-SCR-4	consideră 'INPUT ... AT ...'.
	CP	(DF-SZ)	revenire, prin CL-SET, dacă
	JR	C,0C86,REPORT-5	numărul liniei este mai mare
	RET	NZ	decît valoarea lui DF-SZ; se
			dă prezentarea 5 dacă este
			mai mic; altfel se continuă.
	BIT	4, (TV-FLAG)	Salt înainte numai dacă se
	JR	Z,0C88,PO-SCR-2	lucrează cu 'listare
			automată'.
	LD	E, (BREG)	Se aduce numărătorul de
			linii.
	DEC	E	Se decrementează numărătorul.
	JR	Z,0C92,PO-SCR-3	Salt înainte dacă listarea
			trebuie rulată.
	LD	A,+00	Altfel se deschide canalul
	CALL	1601,CHAN-OPEN	'K', se refac indicatorul de
	LD	SP, (LIST-SP)	stivă, se repositionează
	REG	4, (TV-FLAG)	fanionul dacă s-a terminat
	RET		listarea automată și se
			revine prin CL-SET.

Prezentarea 5 - afară din ecran.

OC86 REPORT-5 RST 0008,ERROR-1 Se apelează rutina de tratare
DEFB +04 erori.

Acum se ia în considerare dacă s-a cerut prompt 'scroll?' ('defilare?').

OC88 PD-SCR-2 DEC (SCR-GT) Se decrementează contorul de
JR NZ,OC82,PD-SCR-3 rulare și se trece la redare
numai dacă a ajuns pe zero.

Se trece la redarea unui mesaj.

LD	A,+18	Contorul este resetat.
SUB	B	
LD	(SCR-GT),A	
LD	HL,(ATTR-T)	Valorile curente ale lui
PUSH	HL	ATTR-T și MASK-T sînt
		salvate.
LD	A,(P-FLAG)	Se salvează valoarea curentă
PUSH	AF	a lui P-FLAG.
LD	A,+FD	Se deschide canalul 'K'.
CALL	0601,CHAN-OPEN	
XOR	A	Mesajul 'scroll?' este mesaj
LD	DE,+0CF8	'0'. Acest mesaj este acum
CALL	0C0A,PD-M88	afisat.
SET	S,(TV-FLAG)	Semnal 'sterge partea de jos
		a ecranului după o apăsare de
		tastă'.
LD	HL,+5C3E	Acesta este FLABS.
SET	3,(HL)	Semnal 'mod L'.
RES	5,(HL)	Semnal 'încă nu este nici o
		tastă'.
EXX		Notă: Si DE trebuie salvat în
		stivă.
CALL	15D4,WAIT-KEY	Se încarcă codul unei singure
		taste.
EXX		Se refac registrii.
CP	+20	Se face un salt înainte la
JR	Z,0D00,REPORT-D	REPORT-D - BREAK-CONT
CP	+E2	repetare' - dacă s-a apăsât
JR	Z,0D00,REPORT-D	tasta 'BREAK', 'STOP', 'N'
OR	+20	sau 'n' altfel, se acceptă
CP	+6E	că apăsarea tastei a indicat
JR	Z,0D00,REPORT-D	necesitatea rulării ecranului
LD	A,+FE	Se deschide canalul 'S'.
CALL	1601,CHAN-OPEN	
POP	AF	Se reface valoarea din
LD	(P-FLAG),A	P-FLAG.
POP	HL	Se refac valorile lui ATTR-T
LD	(ATTR-T),HL	și MASK-T.

Ecranul este acum rulat.

OC82 PD-SCR-3	CALL	0DFE,CL-SC-ALL	Intregul ecran este rulat.
	LD	B,(DF-SZ)	Numererele de linie și coloană
	INC	B	pentru începutul liniei
	LD	C,+21	dinaintea părții de jos a
	PUSH	BC	ecranului sînt găsite și
			salvate.
	CALL	0E9B,CL-ADDR	Octetul atribut corespunzător
	LD	A,H	pentru această porțiune de
	RRCA		caracter este găsit.
	RRCA		Registrul pereche HL conține
	RRCA		adresa octetului.
	AND	+03	
	OR	+58	
	LD	H,A	

Linia în discuție va avea valoarea caracteristică a părții inferioare și noua linie din partea de jos a ecranului poate avea valoarea 'ATTR-P', așa că valorile caracteristice sînt schimbate.

LD	DE,+5AE0	DE conține primul octet
		atribut al liniei inferioare.
LD	A,(DE)	Valoarea este încărcată.
LD	C,(HL)	Valoarea 'partea inferioară'.
LD	B,+20	Acolo sînt 32 de octeți.
EX	DE,HL	Schimbare indicatori.

OCFO PD-SCR-3A	LD (DE),A	Se face prima schimbare si
	LD (HL),C	apoi se trece la utilizarea
	INC DE	aceleasi valori pentru cei 32
	INC HL	de octeti atribut pentru
	DJNZ OCFO,PD-SCR-3A	tratata a doua linie.
	POP BC	Se incarca numerele de linie
		si coloana ale liniei
		inferioare a 'partii de sus'
		inainte de a se reveni.
	RET	

Mesajul 'rulare'.

OCF8	DEFB +80	Marcatorul initial - pas
		inainte.
	DEFB +73,+63,+72,+6F	s - c - r - o
	DEFB +6C,+6C,+BF	1 - 1 - 2 (convertite).

Prezentarea D- BREAK - CONT se repetă.

OD00 REPORT-D	RST 0008,ERROR-1	Se apelează rutina de tratare
	DEFB +0C	a erorilor.

Partea inferioară a ecranului este prezentată în continuare:

OD02 PD-SCR-4	CP +02	Eroarea 'iesire din ecran'
	JR C,OC86,REPORT-5	se da dacă partea inferioară
	ADD A,(DF-SZ)	urmează să fie 'prea mare' si
	SUB +19	se revine dacă rularea nu
	RET NC	este necesară.
	NEG	Registrul A va contine acum
		'numărul rulărilor ce trebuie
		făcute'.
	PUSH BC	Numerele de linie si coloana
		sint acum salvate.
	LD B,A	Acum se salvează 'numărul
	LD HL,(ATTR-T)	rulărilor', ATTR-T, MASK-T &
	PUSH HL	P-FLAG.
	LD HL,(P-FLAG)	
	PUSH HL	
	CALL OD4D,TEMPS	Trebuie folosite numerele
		culorilor 'permanente'.
	LD A,B	Incărcare 'număr rulări'.

Partea inferioară a ecranului este acum rulată de un număr 'A' de ori.

OD1C PD-SCR-4A	PUSH AF	Salvare 'număr'.
	LD HL,+5C6B	Acesta este DF-SZ.
	LD B,(HL)	Valoarea din DF-SZ este
	LD A,B	incrementată; registrul B va
	INC A	contine valoarea formată iar
	LD (HL),A	registrul A noua valoare.
	LD HL,+5C89	Incărcare S-POSN-hi.
	CP (HL)	Saltul are loc numai dacă
	JR C,OD2D,PD-SCR-4B	partea inferioară a ecranului
		urmează a fi rulată (B=vechea
		DF-SZ).
	INC (HL)	Altfel, S-POSN-hi este
	LD B,+18	incrementat si întregul ecran
		este rulat (B+18).
		Rulare 'B' linii.
		Se aduce si se decrementează
		'numărul de rulare'.
		Salt înapoi până la terminare
		Se refacă valoarea lui P-FLAG
OD2D PD-SCR-4B	CALL OE00,CL-SCROLL	Se refac valorile lui ATTR-T
	POP AF	si MASK-T.
	DEC A	Dacă S-POSN a fost schimbat,
	JR NZ,OD1C,PD-SCR-4A	este apelat CL-SET pentru a-i
	POP HL	egala valoarea cu B-CD.
	LD (P-FLAG),L	Se resetează flagul pentru
	POP HL	ca să indice că s-a tratat
	LD (ATTR-T),HL	partea inferioară a ecranului
	LD BC,(S-POSN)	se aduc numerele liniei si
	RES 0,(TV-FLAG)	coloanei si se revine.
	CALL OD09,CL-SET	
	SET 0,(TV-FLAG)	
	POP BC	
	RET	

THE 'TEMPORARY COLOUR ITEMS' SUBROUTINE (SUBROUTINA 'NUMERILE DE CULORI TEMPORARE').

Aceasta este o rutină foarte importantă. Ea este utilizată întotdeauna când

trebuie copiate detaliile 'permanente' în sistemul 'temporar' de variabile.
Se consideră mai întâi ATTR-T & MASK-T.

0D4D TEMPS	XOR A	Registrul A se setează (+00).
	LD HL,(ATTR-P)	Se încarcă valorile curente
	BIT 0,(TV-FLAG)	ale lui ATTR-T și MASK-P.
	JP Z,0D5B,TEMPS-1	Salt înainte dacă se tratează
		partea principală a
		ecranului.
	LD H,A	Altfel se utilizează +00 și
0D5B TEMPS-1	LD L,(BORDCR)	valoarea din BORDCR n loc.
	LD (ATTR-T),HL	Setare ATTR-T & MASK-T.

În continuare se consideră P-FLAG.

	LD HL,+5C91	Acesta este P-FLAG.
	JR NZ,0D65,TEMPS-2	Salt înainte dacă se lucrează
		cu partea inferioară a
		ecranului (A = +00).
	LD A,(HL)	Altfel se încarcă valoarea
	RRCA	P-FLAG și se mută bitii
		impari în locul celor pari.
0D65 TEMPS-2	XOR (HL)	Se trece la copierea bitilor
	AND +55	pari din A în P-Flag.
	XOR (HL)	
	LD (HL),A	
	RET	

THE 'CLS COMMAND' ROUTINE (RUTINA 'COMANDA CLS')

Prima dată se 'sterge' tot ecranul - toți 'pixelii' se resetează și octetii atribut sînt setați pentru egalitatea cu valoarea ATTR-P - apoi partea inferioară a ecranului este reformatată.

0D6B CLS	CALL 0DAF,CL-ALL	Se 'sterge' tot ecranul.
0D6E CLS-LOWER	LD HL,+5C3C	Se încarcă TV-FLAG.
	REG S,(HL)	Semnal 'nu sterge partea
		inferioară a ecranului după
		apăsare tastă'.
	SET 0,(HL)	Semnal 'apre inferioară'.
	CALL 0D4D,TEMPS	Se folosesc valorile
		permanente, adică se copiază
		ATTR-T din BORDCR.
	LD B,(DF-SZ)	Partea inferioară a ecranului
	CALL 0E44,CL-LINE	este acum 'curățată' cu
		această valoare.

Cu excepția octetilor atribut ai liniilor '22' & '23', octetii atribut ai liniilor din partea inferioară a ecranului trebuie să fie făcuți egali cu ATTR-P.

	LD HL,+5AC0	Octetul atribut la începutul
	LD A,(ATTR-P)	liniei '22'.
	DEC B	Se încarcă ATTR-P.
	JR 0D8E,CLS-3	Contorul de linie.
0D87 CLS-1	LD C,+20	Salt înainte în buclă.
0D89 CLS-2	DEC HL	+20 de caractere pe linie.
	LD (HL),A	Întoarcere de-a lungul liniei
		care are setat octetul
		atribut..
	DEC C	
0D8E CLS-3	JR NZ,0D89,CLS-2	
	DJNZ 0D87,CLS-1	Se ciclează pînă la terminare

Se poate fixa mărimea părții inferioare a ecranului.

LD (DF-SZ),+02	Vor fi două linii ca mărime.
----------------	------------------------------

Acum rămîne ca celelalte taskuri de 'administrare' să lucreze.

0D94 CL-CHAN	LD A,+FD	Se deschide canalul 'K'.
	CALL 1801,CHAN-OPEN	
	LD HL,(CURCHL)	Se aduce adresa canalului
	LD DE,+09F4	actual și adresa de ieșire
	AND A	este +09F4 (=PRINT-OUT) iar
0DA0 CL-CHAN-A	LD (HL),E	adresa de intrare este +10A8
	INC HL	(=KEY-INPUT).
	LD (HL),D	
	INC HL	
	LD DE,+10A8	

CCF		Prima dată adresa de ieşire,
JR	C, ODA0, CL-CHAN-A	apoi adresa de intrare.
LD	BC, +1721	Cum partea inferioară a
		ecranului a fost tratată,
		'linia inferioară de afişare'
		va fi linia '23'.
JR	ODD9, CL-SET	Revenire prin CL-SET.

THE 'CLEARING THE WHOLE DISPLAY AREA' SUBROUTINE (SUBROUTINA 'STERGEREA INTREGULUI SPATIU DE AFISARE')

Această subrutină este apelată: i. de către rutina de comandă CLS, ii. de către execuția principală a rutinei, iii. de către rutina de listare automată.

ODAF CL-ALL	LD HL, +0000	Sistemul COORDS este resetat pe zero.
	LD (COORDS), HL	Seamnal 'ecranul este sters'.
	RES 0, (FLAGS2)	Prelucrare taskuri 'administrare'.
	CALL OD94, CL-CHAN	Se deschide canalul 'S'.
	LD A, +FE	
	CALL 1601, CHAN-OPEN	Utilizare valori 'permanente'
	CALL OD4D, TEMPS	Acele se 'sterg' 24 de linii de ecran.
	LD B, +10	Se asigură că adresa curentă de ieşire este +09F4 (PRINT-OUT).
	CALL OE48, CL-LTNE	
	LD HL, (CURCHL)	
	LD DE, +09F4	
	LD (HL), E	
	INC HL	
	LD (HL), D	Resetare contor rulare.
	LD (SCR-CT), +01	Cum partea superioară a ecranului a fost tratată, 'linia de imprimare superioară' va fi linia '0'.
	LD BC, +1821	Se continuă în CL-SET.

THE 'CL-SET' SUBROUTINE (SUBROUTINA 'CL-SET')

Această subrutină este introdusă prin registrul pereche BC, care conține numerele de linie și coloană pentru zona caracterului, sau prin registrul C, care conține numărul coloanei din bufferul printer-ului. Adresa potrivită primului bit de caracter este găsită. Subrutina revine prin PO-STORE pentru a stoca toate valorile în sistemul de variabile cerut.

ODD9 CL-SET	LD HL, +5800	Inceputul buffer-ului printer-ului.
	BIT A, (FLAGS)	Salt înainte dacă se tratează buffer printer.
	JR NZ, ODF4, CL-SET-2	Se transferă numărul liniei.
	LD A, 8	Salt înainte dacă se tratează
	BIT 0, (TV-FLAG)	partea principală a ecranului
	JR 7, ODEE, CL-SET-1	Linia de sus a părții
	ADD A, (DF-SZ)	superioară a ecranului este
	SUB #18	numită 'linia +18' și ea
		trebuie convertită.
ODEE CL-SET-1	PUSH BC	Se salvează numerele liniei și coloanei.
	LD B, A	Se încarcă numărul liniei.
	CALL OE9B, CL-ADDR	Adresa de început a liniei este formată în HL.
	POP BC	Numerele liniei și coloanei sunt aduse înapoi.
ODF4 CL-SET-2	LD A, +21	Numărul coloanei este acum
	SUB C	inversat și transferat în
	LD E, A	registrul pereche DE.
	LD D, +00	
	ADD HL, DE	Adresa cerută este formată,
	JP OADC, PO-STORE	și adresa și numerele liniei și coloanei sunt stocate prin salt la PO-STORE.

THE 'SCROLLING' SUBROUTINE (SUBROUTINA 'RULARE' ('DEFILARE'))

Numărul de linii ale ecranului ce trebuie rulate trebuie aduse, la intrarea în subrutina principală, în registrul B.

ODFE CL-SC-ALL	LD B, +17	Punctul de intrare după 'rulare'?
----------------	-----------	-----------------------------------

Punctul de intrare principal - de mai sus fiind rularea - se face pentru INPUT..AT.

0E00 CL-SCROLL CALL 0E9B,CL-ADDR Găsește adresa de început a liniei.
 LD C,+08 Sînt opt linii pixel pentru o linie completă.

Acum se intră în bucla principală de rulare. Registrul B conține numărul liniei superioare ce trebuie rulată, registrul pereche HL conține adresa de început a acestei linii din spațiul ecranului și registrul C conține pixelul de linie pentru culoare.

0E05 CL-SCR-1 PUSH BC Se salvează ambele contoare.
 PUSH HL Se salvează adresa de început.
 LD A,B Salt înainte numai dacă se lucrează cu o 'treime' din ecran.
 AND +07
 LD A,B
 JR NZ,0E19,CL-SCR-3

Pixelul de linie pentru linia de sus a 'treimii' ecranului trebuie mutată peste 2K limite. (Fiecare treime 2K.)

0E0B CL-SCR-2 EX DE,HL Rezultatul acestei mutări lasă HL neschimbat iar DE va indica destinația cerută.
 LD HL,+FBEO
 ADD HL,DE
 EX DE,HL
 LD BC,+0020 Sînt +20 de caractere.
 DEC A Se decrementează contorul astfel încît o linie să fie tratată ca atare.
 LDIR Acum se mută cei 32 de octeți

În continuare se prezintă pixelul liniilor prin care 'treimile' pot fi rulate. Registrul A conține, la primul pas, +01 - +07, +09 - +0F sau +11 - +17.

0E19 CL-SCR-3 EX DE,HL Se face ca DE să indice destinația cerută.
 LD HL,+FFE0 De această dată doar 32 de locații avansează.
 ADD HL,DE Se salvează numărul liniei în registrul B.
 EX DE,HL
 LD B,A Acum se determină cîte caractere rămîn în 'treime'.
 AND +07
 RRCA
 RRCA
 RRCA
 LD C,A Se aduce 'caracterul total' în registrul C.
 LD A,B Se încarcă numărul liniei.
 LD B,+00 BC conține 'caracterul total' și un pixel linie din fiecare din caracterele 'rulate'.
 LDIR Acum se pregătește pentru incrementare adresa de salt la a linia 'treimii'.
 LD B,+07 Se incrementează HL cu +0700.
 ADD HL,BC Salt înapoi dacă sîi există o 'treime de luat în considerare'.
 AND +F8
 JR NZ,0E0B,CL-SCR-2

Acum se determină dacă bucla a fost parcursă de opt ori - o dată pentru fiecare pixel de linie.

POP HL Se încarcă adresa originală.
 INC H Se adresează următorul pixel linie.
 POP BC Se aduc contorii.
 DEC C Decrementare contor pixel linie și salt înapoi numai dacă s-au mutat opt linii.
 JR NZ,0E05,CL-SR-1

Urmează rularea octetilor atribut . De notat că registrul B conține încă numărul liniilor ce trebuie rulate și registrul C conține zero.

CALL 0E88,CL-ATTR Sînt găsite adresele cerute în spațiul distribuit și numerele caracterelor în linia 'B'.
 LD HL,+FFE0 Deplasamentul pentru tot octetii atribut este avansat cu 32 de locații.
 ADD HL,DE
 EX DE,HL

LDIR	Octetii	atribut	sint
	'rulai'.		

A rămas acum de curătat doar linia de jos ale ecranului.

LD	B,+01	Registrul B este încărcat cu +01 și CL-LINE este introdusă.
----	-------	---

THE 'CLEAR LINES' SUBROUTINE (SUBROUTINA 'STERGERE LINII')
Această subrutină va șterge 'B' linii de jos ale ecranului.

0E44 CL-LINE	PUSH BC	Se salvează numărul liniei pe durata acestei subrutine.
	CALL 0E9B,CL-ADDR	Adresa de început a liniei este formată în HL.
	LD C,+08	Din nou trebuie considerate opt linii pixel.

Acum urmează o buclă de ștergere a tuturor pixelilor linie.

0E4A CL-LINE-1	PUSH BC	Salvare număr liniei și contor pixel linie.
	PUSH HL	Salvare adresă.
0E4D CL-LINE-2	LD A,E	Salvare număr linie în A.
	AND +07	Se determină acum câte câte caractere sînt implicate în 'B mod 8' linii.
	RRCA	Se trece rezultatul în registrul C. (C va conține +00, acesta este 256 zec., pentru o 'treime'.)
	RRCA	Se încarcă numărul liniei.
	RRCA	Se face ca registrul pereche BC să conțină 'cu unul mai puțin' decît numărul caracterelor.
	LD C,A	Se face ca DE să indice primul caracter.
	LD A,B	Se șterge octetul-pixel al primului caracter.
	LD B,+00	Se face ca DE să indice al doilea caracter și se șterg octetii-pixel ale tuturor celorlalte caractere.
	DEC C	Pentru fiecare 'treime' a ecranului HL se va incrementa cu +0701.
	LD D,H	Acum se decrementează numărul liniei.
	LD E,L	Se înălătură orice linie superioară și se trece contorul 'treimii' în B.
	LD (HL),+00	Salt înapoi dacă mai există treime cu care să se lucreze.
	INC DE	
	LDIR	
	LD DE,+0701	
	ADD HL,DE	
	DEC A	
	AND +FS	
	LD B,A	
	JR NZ,0E4D,CL-LINE-2	

Acum se determină dacă bucla a fost folosită de opt ori.

POP HL	Se ridică adresa pentru fiecare pixel linie.
INC H	Încărcare contori.
POP BC	Decrementare contor pixel linie și salt înapoi pînă se termină.
DEC C	
JR NZ,0E4A,CL-LINE-1	

În continuare se setează octetii atribut, așa cum se cere. Valoarea din ATTR-P va fi folosită cînd se tratează partea principală a ecranului iar valoarea din BORDCR va fi folosită cînd se tratează partea inferioară.

CALL 0E88,CL-ATTR	Se determină adresa primului octet atribut și numărul octetilor.
LD H,D	HL va indica primul octet atribut iar DE și va indica pe cel de al doilea.
LD L,E	Se aduce valoarea din ATTR-P.
INC DE	Salt înainte dacă se tratează partea principală a ecranului.
LD A,(ATTR-P)	
BIT 0,(TV-FLAG)	
JR 2,0E80,CL-LINE-3	

	LD	A, (BORDCR)	Altfel, în loc se folosește BORDCR.
0E80 CL-LINE-3	LD	(HL), A	Setare octet atribut.
	DEC	BC	S-a făcut un octet.
	LDIR		Acum se copiază valoarea în toți octetii atribut.
	POP	BC	Se reface numărul liniei.
	LD	C, +21	Setare număr coloană la stînga și revenire.
	RET		
	RET		

THE 'CL-ATTR' SUBROUTINE (SUBRUTINA 'CL-ATTR')
 Această subrutină îndeplinește două funcții diferite.

i. Pentru o adresă dată a zonei ecranului, adresa proprie a atributului este returnată în registrul pereche DE. Este de notat faptul că valoarea indicatorului de intrare este pe a 'noua' liniei a a caracterului.

ii. Pentru un număr de linie dat, în registrul B, numărul zonelor caracterului de pe ecran de la începutul acestei linii în jos este readusă în registrul pereche BC.

0E88 CL-ATTR	LD	A, H	Se aduce bitul superior.
	RRCA		Se multiplică această valoare de 32 de ori.
	RRCA		
	RRCA		
	DEC	A	Întoarcere la linia a 'opta'.
	OR	+50	Adresa zonei atribut.
	LD	H, A	Se readuce octetul superior și se transferă adresa în DE.
	EX	DE, HL	Este întotdeauna zero.
	LD	H, C	Numărul liniei.
	LD	L, B	Se multiplică de 32 de ori.
	ADD	HL, HL	
	ADD	HL, HL	
	ADD	HL, HL	
	ADD	HL, HL	
	ADD	HL, HL	
	LD	B, H	Se mută rezultatul în registrul pereche BC înainte de revenire.
	LD	C, L	
	RET		

THE 'CL-ADDR' SUBROUTINE (SUBRUTINA 'CL-ADDR')
 Pentru un număr de linie dat, în registrul B, adresa câmpului caracteristic ecranului este format în registrul pereche HL.

0E98 CL-ADDR	LD	A, +18	Numărul liniei trebuie inversat.
	SUB	B	Rezultatul este salvat în B.
	LD	D, A	Ca efect '(A mod 8) * 32'.
	RRCA		În o 'treime' a ecranului octetul inferior pentru:
	RRCA		linia 1 = +00
	RRCA		linia 2 = +20, etc.
	AND	+E0	Octetul inferior se transferă în L.
	LD	L, A	Se încarcă numărul real al liniei.
	LD	A, D	Ca efect '64 + 8 * INT (A/8)'
	AND	+18	Pentru 'treimea' de sus a ecranului octetul superior = +40, 'treimea' mijlocie = +48, iar pentru 'treimea' de jos = +50.
	OR	+40	Octetul superior este transferat în H.
	LD	H, A	Terminare.
	RET		

THE 'COPY' COMMAND ROUTINE (RUTINA DE COMANDA 'COPY')
 176 de pixeli linie ai ecranului sînt prelucrați rînd pe rînd.

0EAC COPY	DI		Înteruperea mascabilă este întreruptă atîta timp cît durează COPY.
	LD	B, +B0	'176' linii.
	LD	HL, +4000	Adresa de bază a ecranului.

Acum se introduce următoarea buclă.

0E12 COPY-1	PUSH HL	Se salvează adresa de bază și numărul liniei.
	PUSH BC	Se apelează de '176' ori.
	CALL 0EF4,COPY-LINE	Se aduce numărul liniei și adresa de bază.
	POP BC	Se mărește adresa de bază cu '256' locații pentru fiecare linie de pixeli.
	POP HL	Salt înainte și rotire prin buclă pentru opt linii pixel pentru o linie caracter.
	INC H	
	LD A,H	
	AND +07	
	JR NZ,0EC9,COPY-2	

Pentru fiecare linie de caracter nouă adresa de bază trebuie mărită.

	LD A,L	Se aduce octetul inferior.
	ADD A,+20	Se mărește cu +20 octeți.
	LD L,A	Fanionul de transport va fi resetat când se lucrează 'în cadrul treimii'.
	CCF	Se schimbă fanionul de transport.
	SBC A,A	Registrul A va conține +F8 când se lucrează în cadrul 'treimii' și +00 când o nouă 'treimă' este atinsă.
	AND +F8	Se mărește acum octetul superior al adresei.
0EC9 COPY-2	ADD A,H	Salt înapoi înă când vor fi afisate '176' de linii.
	LD H,A	Salt înainte la sfîrsitul rutinei.
	DJNZ 0E12,COPY-1	
	JR 0EDA,COPY-END	

THE 'COPY-BUFF' SUBROUTINE (SUBROUTINA 'COPY-BUFF')

Această subrutină este apelată întotdeauna când bufferul de imprimare trebuie să-si transfere continutul la afisaj.

0ECD COPY-BUFF	DI	Dezactivare întrerupere mascabilă.
	LD HL,+5800	Adresa de bază a bufferului de imprimare.
0ED3 COPY-3	LD B,+08	Sînt opt linii pixel.
	PUSH BC	Se salvează numărul liniei.
	CALL 0EF4,COPY-LINE	Se cheamă de '8' ori.
	POP BC	Se aduce numărul liniei.
	DJNZ 0ED3,COPY-3	Salt înapoi pînă cînd '8' linii au fost imprimate.

Continuare la rutina COPY-END.

0EDA COPY-END	LD A,+04	Oprire imprimantă.
	OUT (+F8),A	
	EI	Se activează întreruperea mascabilă și se continuă la CLEAR-PR3.

THE 'CLEAR PRINTER BUFFER' SUBROUTINE (SUBROUTINA 'STERGERE BUFFER INPRIMARE')

Bufferul de imprimare este sters prin apelarea acestei subrutine.

0EDF CLEAR-PR3	LD HL,+5800	Adresa de bază a bufferului de imprimare.
	LD (PR-CC),L	Resetare 'coloană' imprimator
	XOR A	Se șterge registrul A.
	LD B,A	De asemenea se șterge registrul B (de fapt B conține 256 zecimal).
0EE7 PR3-BYTES	LD (HL),A	Toti cei '256' de octeți ai bufferului de imprimare sînt stersi pe rînd.
	INC HL	Seamnal 'bufferul este gol'.
	DJNZ 0EE7,PR3-BYTES	Se setează poziția imprimatorului și se revine prin CL-SET & PO-STORE.
	RES 1,(FLAGS2)	
	LD C,+21	
	JP 0DD9,CL-SET	

THE 'COPY-LINE' SUBROUTINE (SUBROUTINA 'COPY-LINE')

La intrarea în subrutină registrul pereche HL conține adresa de bază pentru cei 32 octeți care formează linia de pixeli și registrul B conține numărul pixelilor linie.

0EF4 COPY-LINE	LD	A, B	Se copiază numărul pixel-linie.
	CP	+03	Registrul A va conține +00
	SBC	A, A	până când ultimele două linii vor fi tratate.
	AND	+02	Pornire motor doar pentru ultimele două linii superioare.
	OUT	(+FB), A	Registrul A va conține sau +00 sau +02.
	LD	D, A	

Mai trebuie făcute trei teste înaintea oricărei 'imprimări'.

0EFD COPY-L-1	CALL	1F54, BREAK-KEY	Salt înainte doar dacă nu s-a apăsât tasta BREAK.
	JR	C, 0F0C, COPY-L-2	Dar dacă s-a apăsât tasta, se oprește motorul, se activează întreruperile mascabile, se șterge bufferul de imprimare și se iese prin rutina de tratare a erorilor - 'BREAK-CONT se repetă'.
	LD	A, +04	Se aduce starea imprimantei.
	OUT	(+FB), A	
	EI		
	CALL	0EDF, CLEAR-PRB	Dacă nu este prezentă imprimanta, se face imediat o revenire.
	RST	0008, ERROR-1	Așteptare mod.
	DEFB	+0C	Sînt 32 de octeți.
0F0C COPY-L-2	IN	A, (+FB)	
	ADD	A, A	
	RET	M	
	JR	NC, 0EFD, COPY-L-1	
	LD	C, +20	

Acum urmează o buclă care tratează acești octeți.

0F14 COPY-L-3	LD	E, (HL)	Se aduce un octet.
	INC	HL	Incrementare indicator.
	LD	B, +08	Opt biți într-un octet.
0F18 COPY-L-4	RL	D	Se mută conținutul lui D la stînga.
	RL	E	Se mută fiecare bit în carry.
	RR	D	Se mută conținutul lui D înapoi, aducînd bitul de transport din E.
0F1E COPY-L-5	IN	A, (+FB)	Se aduce din nou starea imprimantei și se așteaptă semnal de la decodor.
	RRA		Acum se merge mai departe și se trece 'bitul' la imprimantă.
	JR	NC, 0F1E, COPY-L-5	Notă: bit 2 - pornirea imprimantei, bit 1 - creșterea cu viteză mică a imprimării și bitul 7 este pentru ridicarea vitezei la actuala imprimare.
	LD	A, B	Se 'imprimă' fiecare bit.
	OUT	(+FB), A	Se decrementează contorul de octeți.
	DJNZ	0F18, COPY-L-4	Salt înapoi atîta vreme cît mai există octeți; altfel revenire.
	DEC	C	
	JR	NZ, 0F14, COPY-L-3	
	RET		

THE 'EDITOR' ROUTINES (RUTINA 'EDITOR')

Editorul este apelat în două cazuri:

i. Din rutina de principală de execuție astfel încît utilizatorul poate introduce o linie BASIC în sistem.

ii. Din rutina de comandă INPUT.

Prima dată se salvează 'eroare indicator stivă' iar apoi se prevede o adresă alternativă.

0F2C EDITOR	LD	HL, (ERR-SP)	Valoarea curentă este salvată în stiva calculatorului.
	PUSH	HL	Acesta este ED-ERROR.
0F30 ED-AGAIN	LD	HL, +107F	Orice eveniment care conduce la rutina de tratare a erorii ce a fost folosită, va reveni la ED-ERROR.
	PUSH	HL	
	LD	(ERR-SP), SP	

Acum se introduce o buclă ce va trata apăsarea fiecărei taste.

0F38 ED-LOOP	CALL 15D4, WAIT-KEY	Revenire dacă s-a apăsat o tastă.
	PUSH AF	Se salvează temporar codul.
	LD D, +00	Se aduce durata cliicului tastei.
	LD E, (PIP)	Si a căderii.
	LD HL, +00C8	Acum se realizează 'beep'-ul.
	CALL 03B5, BEEPER	Refacerea codului.
	POP AF	Adresa ED-LOOP este încărcată în stiva calculatorului.
	LD HL, +0F38	
	PUSH HL	

Acum se face analiza codului obtinut.

CP +18	Se acceptă toate codurile caracterelor, caracterelor grafice și ale semnelor.
JR NC, 0F81, ADD-CHAR	De asemenea se acceptă '.,'.
CP +07	
JR C, 0F81, ADD-CHAR	
CP +10	Salt înainte dacă codul reprezintă o tastă de editare
JR C, 0F92, ED-KEYS	

Acum se iau în considerare tastele de control - de la INK la TAB.

LD BC, +0002	INK & PAPER necesită două locatii.
LD D, A	Se copiază codul în D.
CP +14	Salt înainte cu 'INK&PAPER'.
JR C, 0F6C, ED-CONTR	

AT & TAB se tratează după cum urmează:

INC BC	Se cer trei locatii.
BIT 7, (FLAGX)	Salt înainte dacă nu se lucrează cu INPUT LINE...
JP Z, 101E, ED-IGNORE	Se aduce cel de al doilea cod și se încarcă în E.
CALL 15D4, WAIT-KEY	
LD E, A	

Acum se încarcă ceilalți octeți pentru controlul caracterelor.

0F6C ED-CONTR	CALL 15D4, WAIT-KEY	Se aduce alt cod.
	PUSH DE	Se salvează codul anterior.
	LD HL, (K-CUR)	Se aduce K-CUR.
	RES 0, (MODE)	Seamnal 'mod K'.
	CALL 1655, MAKE-ROOM	Se realizează două sau trei spații.
	POP DE	Se readuce codul anterior.
	INC HL	Se indică prima locație.
	LD (HL), D	Se introduce primul cod.
	INC HL	Apoi se introduce al doilea cod care va fi scris deapoi dacă sînt numai două coduri - aceasta cu INK&PAPER
	LD (HL), C	Salt înainte.
	JP 0F8B, ADD-CH-1	

THE 'ADD-CHAR' SUBROUTINE (SUBROUTINA 'ADD-CHAR')

Această subrutină de obicei adună un cod la linia curentă EDIT (editare) sau INPUT (introducere).

0F81 ADD-CHAR	RES 0, (MODE)	Seamnal 'mod K'.
	LD HL, (K-CUR)	Se aduce poziția cursorului.
	CALL 1652, ONE-SPACE	Se realizează un singur spațiu.
0F8B ADD-CH-1	LD (DE), A	Se introduce codul în spațiu și se semnalează că cursorul se poziționează cu o locație după. Apoi se revine indirect în ED-LOOP.
	INC DE	
	LD (K-CUR), DE	
	RET	

Tastele de comandă sînt tratate ca mai jos:

0F92 ED-KEYS	LD E, A	Codul este transferat în registrul pereche DE.
	LD D, +00	Adresa de bază a tabelului tastei de editare.
	LD HL, +0F99	Intrarea este adresată și apoi este introdusă în DE.
	ADD HL, DE	
	LD E, (HL)	

ADD HL,DE	Adresa rutinei de tratare este salvată în stiva calculatorului.
PUSH HL	Registrul pereche HL este setat și se face un salt indirect la rutina cerută.
LD HL,(K-CUR)	
RET	

THE 'EDITING KEYS' TABLE (TABELUL 'TASTELOR DE EDITARE')

adresă deplasament	caracter	adresă deplasament	caracter
OFA0 09	EDIT	OFA5 70	DELETE
OFA1 66	cursor stînga	OFA6 7E	ENTER
OFA2 6A	cursor dreapta	OFA7 CF	SYMBOL SHIFT
OFA3 50	cursor jos	OFA8 D4	GRAPHICS
OFA4 85	cursor sus		

THE 'EDIT KEY' SUBROUTINE (SUBROUTINA 'EDITARE TASTA')

Cînd în 'mod editare' se apasă tasta EDIT se va aduce jos 'linia BASIC actuală'.
 În 'mod INPUT' acțiunea tastei EDIT este de a șterge răspunsul curent și de a scrie linia nouă.

OFA7 ED-EDIT	LD HL,(E-PPC)	Se aduce numărul liniei curente.
	BIT 5,(FLAGX)	Salt înainte dacă este în 'mod INPUT'.
	JP NI,1097,CLEAR-SP	Găsește adresa de început a liniei curente și avansează numărul ei.
	CALL 196E,LINE-ADDR	Dacă numărul liniei returnate este zero, atunci pur și simplu se șterge zona de editare.
	CALL 1695,LINE-NO	Se salvează adresa liniei.
	LD A,D	Se păstrează lungimea liniei.
	OR E	
	JP Z,1097,CLEAR-SP	
	PUSH HL	
	INC HL	
	LD C,(HL)	
	INC HL	
	LD B,(HL)	
	LD HL,+000A	Se adună +0A la lungime și se testează pentru a determina dacă sînt suficiente locații pentru o copie a liniei.
	ADD HL,BC	
	LD B,H	
	LD C,L	
	CALL 1F05,TEST-ROOM	Acum se șterge spațiul de editare.
	CALL 1097,CLEAR-SP	Se aduce adresa canalului curent și se schiabă cu adresa liniei.
	LD HL,(CURCHL)	Salvare temporară a acesteia.
	EX (SP),HL	Se deschide canalul 'R' așa că linia va fi copiată în zona de editare.
	PUSH HL	Se aduce adresa liniei.
	LD A,+FF	Se trece înainte de linie.
	CALL 1801,CHAN-OPEN	Se decrementează numărul liniei curente astfel încît să se evite tăierea peste cursor.
	POP HL	Se tipărește linia BASIC.
	DEC HL	Se incrementează numărul liniei curente.
	DEC (E-PPC-10)	Notă: Decrementarea numărului liniei nu evită întotdeauna supraimprimarea cursorului.
	CALL 1855,OUT-LINE	Se aduce începutul liniei în spațiul de editare și se trece dincolo de numărul liniei și de lungime pentru a găsi adresa lui K-CUR.
	INC (E-PPC-10)	
	LD HL,(E-LINE)	
	INC HL	
	INC HL	
	INC HL	
	INC HL	
	LD (K-CUR),HL	
	POP HL	
	CALL 1615,CHAN-FLAG	Se aduce adresa canalului anterior și se setează fanioanele corespunzătoare înainte de a reveni în ED-LOOP.
	RET	

THE 'CURSOR DOWN EDITING' SUBROUTINE (SUBRUTINA 'EDITARE CURSOR JOS')

OFF3 ED-DOWN	BIT	5, (FLAGX)	Salt înainte dacă este 'mod INPUT'.
	JR	NZ, 1001, ED-STOP	Acesta este E-PPC.
	LD	HL, +5C49	S-a găsit numărul liniei următoare și se realizează o nouă histare automată.
	CALL	190F, LN-FETCH	Prezentarea 'STOP în INPUT'.
	JR	106E, ED-LIST	Salt înainte.
1001 ED-STOP	LD	(ERR-NR), +10	
	JR	1024, ED-ENTER	

THE 'CURSOR LEFT EDITING' SUBROUTINE (SUBRUTINA 'EDITARE CURSOR STINGA')

1007 ED-LEFT	CALL	1031, ED-EDGE	Cursorul este mutat.
	JR	1011, ED-CUR	Salt înainte.

THE 'CURSOR RIGHT EDITING' SUBROUTINE (SUBRUTINA 'EDITARE CURSOR DREAPTA')

100C ED-RIGHT	LD	A, (HL)	Caracterul curent este testat și se execută revenire dacă este 'acriage return'.
	CP	+0D	In caz contrar se determină poziționarea cursorului după caracter.
	RET	Z	Se setează variabilă sistem K-CUR.
	INC	HL	
1011 ED-CUR	LD	(K-CUR), HL	
	RET		

THE 'DELETE EDITING' SUBROUTINE (SUBRUTINA 'EDITARE STERGERE')

1015 ED-DELETE	CALL	1031, ED-EDGE	Se mută cursorul la stînga.
	LD	BC, +0001	Cerere caracter curent.
	JP	19E8, RECLAIM-2	

THE 'ED-IGNORE' SUBROUTINE (SUBRUTINA 'ED-IGNORE' (ignorare editare))

101E ED-IGNORE	CALL	15D4, WAIT-KEY	Următoarele două coduri ale rutinei de introducere tastă se ignoră.
	CALL	15D4, WAIT-KEY	

THE 'ENTER EDITING' SUBROUTINE (SUBRUTINA 'INTRODUCERE EDITARE')

1024 ED-ENTER	POP	HL	Se înlătură adresele de la ED-LOOP și ED-ERROR.
	POP	HL	Se reface vechea valoare a lui ERR-SP.
1026 ED-END	LD	(ERR-SP), HL	Revenire dacă nu au fost erori.
	BIT	7, (ERR-NR)	Altfel se execută un salt indirect la rutina de eroare.
	RET	NZ	
	LD	SP, HL	
	RET		

THE 'ED-EDGE' SUBROUTINE (SUBRUTINA 'ED-EDGE')

Adresa cursorului se află în registrul pereche HL și va fi decrementată dacă cursorul nu se află la începutul liniei. Trebuie avut grijă ca să nu se pună cursorul între caracterele de control și parametrii lor.

1031 ED-EDGE	SCF		DE va conține fie E-LINE (pentru editare) fie WORKSP (pentru introducere).
	CALL	1195, SET-DE	Fanionul de transport va fi setat dacă cursorul este deja la începutul liniei.
	SBC	HL, DE	Corectare pentru scădere.
	ADD	HL, DE	Se trimite adresa de revenire.
	INC	HL	Revenire prin ED-LOOP dacă fanionul de transport este setat.
	POP	BC	Se reface adresa de revenire.
	RET	C	Se mută adresa curentă a cursorului în BC.
	PUSH	BC	
	LD	R, H	
	LD	C, L	

Acum se introduce o buclă care să verifice dacă caracterele de control nu sînt despărțite de parametrii lor.

103E ED-EDGE-1	LD	H, D	HL va indica caracterul în linie după cea adresată de DE
	LD	L, E	
	INC	HL	

LD	A, (DE)	Se aduce codul caracterului.
AND	+F0	Salt înainte dacă caracterul nu reprezintă INK to TAB.
CP	+10	
JR	NZ, 1051, ED-EDGE-2	
INC	HL	Este permis un parametru.
LD	A, (DE)	Se aduce iarasi codul.
SUB	+17	Carry este resetat pentru TAB
ADC	A, +00	Notă: Aceasta desparte AT&TAB dar AT&TAB în această formă oricum nu sînt implementate, așa că nu implică nici o diferențiere.
JR	NZ, 1051, ED-EDGE-2	Salt înainte dacă nu se lucrează cu AT&TAB care vor avea doi parametri, cînd se folosesc.
INC	HL	Pregătire pentru scădere.
1051 ED-EDGE-2	AND A	Fanionul de transport va fi resetat cînd indicatorul actualizat ajunge pe K-CUR.
	SBC HL, BC	Pentru următoarea buclă se folosește indicatorul actualizat, dar dacă există se folosește indicatorul prezent pentru K-CUR.
	ADD HL, BC	Notă: Se șterge caracterul de control cînd este folosit DELETE.
	EX DE, HL	
	JR C, 103E, ED-EDGE-1	
	RET	

THE 'CURSOR UP EDITING' SUBROUTINE (SUBROUTINA 'EDITARE CURSOR SUS')

1059 ED-UP	BIT 5, (FLAGX)	Revenire dacă este în 'mod INPUT'.
	RET NZ	
	LD HL, (E-PPC)	Se aduce numărul liniei curente și adresa sa de început.
	CALL 196E, LINE-ADDR	HL indică acum linia anterioară.
	EX DE, HL	Se aduce acest număr de linie.
	CALL 1695, LINE-NO	Este E-PPC-hi.
	LD HL, +5C4A	Se stochează numărul liniei.
106E ED-LIST	CALL 191C, LN-STORE	Se execută o nouă listare automată și canalul 'K' este redeschis înainte de a se reveni la ED-LOOP.
	CALL 1795, AUTO-LIST	
	LD A, +00	
	JP 1601, CHAN-OPEN	

THE 'ED-SYMBOL' SUBROUTINE (SUBROUTINA 'ED-SYMBOL' (editare simbol))
 Dacă se vor folosi codurile SYMBOL & GRAPHICS, ele vor fi tratate cum sînt prezentate mai jos:

1076 ED-SYMBOL	BIT 7, (FLAGX)	Salt înapoi dacă nu se lucrează cu INPUT .. LINE.
	JR Z, 1024, ED-ENTER	Salt înapoi.
107C ED-GRAPH	JP 0F81, ADD-CHAR	

THE 'ED-ERROR' SUBROUTINE (SUBROUTINA 'ED-ERROR' (editare eroare))
 Aici se va executa saltul în cazul în care au apărut diferite tipuri de erori.

107F ED-ERROR	BIT 4, (FLAGB2)	Salt înapoi dacă se folosește alt canal decît canalul 'K'.
	JR Z, 1026, ED-END	Se anulează numărul erorii și se dă o semnalizare sonoră înainte de a intra iar în editor.
	LD (ERR-NR), +FF	
	LD D, +00	
	LD E, (RASP)	
	LD HL, +1A90	
	CALL 0385, BEEPER	
	JP 0F30, ED-AGAIN	

THE 'CLEAR-SP' SUBROUTINE (SUBROUTINA 'CLEAR-SP' (ștergere spațiu))
 Zona de editare sau spațiul de lucru se șterge după cum se indică.

1097 CLEAR-SP	PUSH HL	Salvare indicator de spațiu.
	CALL 1190, SET-HL	DE va indica primul caracter și HL pe ultimul.
	DEC HL	Acum se cere valoarea corectă.
	CALL 19E5, RECLAIM-1	Variabilele sistem K-CUR și
	LD (K-CUR), HL	MODE ('mod K') sînt
	LD (MODE), +00	initializate înainte de a
	POP HL	se aduce indicatorul, apoi
	RET	

revenire.

THE 'KEYBOARD INPUT' SUBROUTINE (SUBROUTINA 'INTRODUCERE TASTATURA')

Această importantă subrutină readuce codul ultimei taste care a fost apăsată, și este de notat faptul că CAPS LOCK, schimbarea modului și parametrii de control ai culorii sînt prezentați în cadrul subrutinei.

10A8 KEY-INPUT	BIT	3, (TV-FLAG)	Copiere linie-editare sau linie-introducere pe ecran dacă modul a fost schimbat.
	CALL	NZ,111D,ED-COPY	Revenire cu fanionul de transport și de zero resetate dacă nici o tastă nu a fost apăsată.
	AND	A	Altfel se încarcă codul și se semnalează că a fost luat.
	BIT	5, (FLAGS)	Salvare temporară a codului.
	RET	Z	Dacă este necesar se șterge partea inferioară a ecranului cu 'după defilare?'
	LD	A, (LAST-K)	Se aduce codul.
	RES	5, (FLAGS)	Se acceptă toate codurile de caractere și semne.
	PUSH	AF	Salt înainte cu majoritatea codurilor caracterelor de control.
	BIT	5, (TV-FLAG)	Salt înainte cu codurile 'mod' și cu codul CAPS LOCK.
	CALL	NZ,10B6,CLS-LOWER	
	POP	AF	
	CP	+20	
	JR	NC,111B,KEY-DONE	
	CP	+10	
	JR	NC,10FA,KEY-CONTR	
	CP	+06	
	JR	NC,10DB,KEY-M&CL	

Acum se lucrează cu codurile FLASH, BRIGHT & INVERSE.

LD	B,A	Se salvează codul.
AND	+01	Se păstrează doar bitul 0.
LD	C,A	C conține +00 (= OFF) sau C conține +01 (= ON).
LD	A,B	Se aduce codul.
RRA		Se rotește o dată (se pierde bitul 0).
ADD	A,+12	Se incrementează cu +12 rezultînd FLASH- +12, BRIGHT- +13 și INVERSE - +14.
JR	1105,KEY-DATA	

Codul CAPS LOCK și codurile modului sînt lucrate cu 'limite'.

10DB KEY-M&CL	JR	NZ,10E6,KEY-MODE	Salt înainte cu codurile 'modului'.
	LD	HL,+5C6A	Acesta este FLASS2.
	LD	A,+08	Se schimbă bitul 3 din FLAGS.
	XOR	(HL)	Acesta este fanionul pentru CAPS LOCK.
	LD	(HL),A	
10E6 KEY-MODE	JR	10F4,KEY-FLAG	Salt înainte.
	CP	+0E	Se verifică limita inferioară
	RET	C	
	SUB	+0D	Se reduce rangul.
	LD	HL,+5C41	Încărcare MODÉ.
	CP	(HL)	A fost schimbat?
	LD	(HL),A	Se introduce cod 'mod' nou.
10F4 KEY-FLAG	JR	NZ,10F4,KEY-FLAG	Salt dacă a fost schimbat;
	LD	(HL),+00	altfel se schimbă în 'mod L'.
	SET	3, (TV-FLAG)	Semnal 'modul poate fi schimbat'.
	CP	A	Resetare fanion de transport și revenire.
	RET		

Se prelucrează codurile de control tastă (diferite de FLASH, BRIGHT & INVERSE)

10FA KEY-CONTR	LD	B,A	Se salvează codul.
	AND	+07	Se face ca registrul C să conțină parametrul (+00 la +07).
	LD	C,A	Registrul A conține acum codul INK.
	LD	A,+10	Dar dacă codul a fost unul 'nedepășat', atunci se face astfel ca A să conțină codul PAPER.
	BIT	3,B	
	JR	NZ,1105,KEY-DATA	
	INC	A	

Parametrul este salvat în K-DATA și adresa canalului schimbată de la KEY-INPUT

1a KEY-NEXT.

1105 KEY-DATA	LD	(K-DATA),C	Salvare parametru.
	LD	DE,+110D	Se încarcă adresa KEY-NEXT.
	JR	1113,KEY-CHAN	Salt înainte.

Notă: La primul pas la intrarea în KEY-INPUT, registrul A este returnat conținând un 'cod de control' și apoi, la următorul pas, la intrarea în KEY-NEXT, acesta este parametru care este returnat.

110D KEY-NEXT	LD	A,(K-DATA)	Se aduce parametru.
	LD	DE,+10A8	Se încarcă adresa KEY-INPUT.

Se setază adresa de intrare în prima zonă canal.

1113 KEY-CHAN	LD	HL,(CHANS)	Se aduce adresa canalului.
	INC	HL	
	INC	HL	
	LD	(HL),E	Acum se setează adresa de intrare.
	INC	HL	
	LD	(HL),D	

În final se iese cu codul cerut în registrul A.

111B KEY-DONE	SCF		Se arată că un cod a fost găsit și se revine.
	RET		

THE 'LOWER SCREEN COPYING' SUBROUTINE (SUBROUTINA 'COPIERE PARTEA INFERIOARA A ECRANULUI')

Această subrutină este apelată întotdeauna când linia din zona de editare sau din spațiul INPUT urmează să fie tipărită în partea de jos a ecranului.

111D ED-COPY	CALL	0D4D,TEMPS	Se utilizează culorile permanente.
	RES	3,(TV-FLAG)	Se semnalizează că 'modul trebuie considerat neschimbat' și 'partea de jos a ecranului nu trebuie stersă'.
	RES	5,(TV-FLAG)	
	LD	HL,(S-POSNL)	Se salvează valoarea curentă a lui S-POSNL.
	PUSH	HL	
	LD	HL,(ERR-SP)	Se păstrează valoarea curentă a lui ERR-SP.
	PUSH	HL	
	LD	HL,+1167	Adresa ED-FULL.
	PUSH	HL	Se salvează această adresă în stiva calculatorului pentru a face ???
	LD	(ERR-SP),SPS	
	LD	HL,(ECHO-E)	Se încarcă stiva cu valoarea lui ECHO-E.
	PUSH	HL	
	SCF		Se face ca HL să indice începutul spațiului iar DE sfârșitul lui.
	CALL	1195,SET-HL	
	EX	DE,HL	Acum se tipărește linia.
	CALL	187D,OUT-LINE2	Se schimbă indicatorii și se afișează cursorul.
	EX	DE,HL	
	CALL	18E1,OUT-CURS	
	LD	HL,(S-POSNL)	Apoi se aduce valoarea curentă a lui S-POSNL și se schimbă cu ECHO-E.
	EX	(SP),HL	
	EX	DE,HL	Se trece CEHO-E în DE.
	CALL	0D4D,TEMPS	Se aduc din nou culorile permanente.

Restul oricărei linii care a fost începută este acum completată cu tipărirea spațiilor cu culoarea permanentă a hîrtiei (PAPER).

1150 ED-BLANK	LD	A,(S-POSNL-hi)	Se aduce numărul liniei curente și se scade numărul liniei vechi.
	SUB	D	
	JR	C,117C,ED-C-DONE	Salt înainte dacă nu s-a cerut nici un 'spatiu' de linie.
	JR	NZ,115E,ED-SPACES	Salt înainte dacă nu este pe aceeași linie.
	LD	A,E	Se aduce numărul coloanei vechi și se scade numărul coloanei noi.
	SUB	(S-POSNL-lo)	
	JR	NC,117C,ED-C-DONE	Salt dacă nu este nici o cerere de spatiu.

115E ED-SPACES	LD	A,+20	Se încarcă un 'spatiu'.
	PUSH	DE	Se salvează vechile valori.
	CALL	09F4,PRINT-OUT	Se tipăresc.
	POP	DE	Se aduc vechile valori.
	JR	1150,ED-BLANK	Din nou înapoi.

Acum se lucrează cu orice eroari.

1167 ED-FULL	LD	D,+00	Seanalizare sonoră.
	LD	E,(RASP)	
	LD	HL,+1A90	
	CALL	0315,BEEPER	
	LD	(ERR-NR),+FF	Anulare număr eroare.
	LD	DE,(S-POSNL)	Se aduce valoarea curentă a lui S-POSNL si se execută salt înainte.
	JR	117E,ED-C-END	Valoarea noii pozitii.
117C ED-C-DONE	POP	DE	'Adresa erorii'.
	POP	HL	

Aici se vine după o eroare.

117E ED-C-END	POP	HL	Se reface vechea valoare a lui ERR-SP.
	LD	(ERR-SP),HL	Se aduce vechea valoare a lui S-POSNL.
	POP	IC	Se salvează valorile pozitiei noi.
	PUSH	DE	Se setează variabilele sistemului.
	CALL	0DB9,CL-SET	Vechea valoarea a lui S-POSNL se încarcă în ECHO-E.
	POP	HL	X-PTR se șterge în mai multe feluri si se face revenire.
	LD	(ECHO-E),HL	
	LD	(X-PTR-hj),+00	
	RET		

THE 'SET-HL' AND 'SET-DE' SUBROUTINES

Aceste subrutine revin cu HL indicînd prima locatie si DE 'ultima' locatie pentru fiecare spatiu de editare sau zonă de lucru.

1190 SET-HL	LD	HL,(WORKSP)	Este indicată ultima locatie a spatiului de editare.
	DEC	HL	Se șterge fanionul de transport.
	AND	A	Se indică începutul spatiului de editare si se revine dacă este 'mod editare'.
1195 SET-DE	LD	DE,(E-LINE)	Altfel este schimbat DE.
	BIT	5,(FLAGX)	Se revine, dacă se intentionează.
	RET	Z	Se aduce STKBOT si se revine.
	LD	DE,(WORKSP)	
	RET	C	
	LD	HL,(STKBOT)	
	RET		

THE 'REMOVE-FP' SUBROUTINE

Această subrutină reamută punctul flotant ascuns format într-o linie BASIC.

11A7 REMOVE-FP	LD	A,(HL)	Se examinează pe rînd fiecare caracter.
	CP	+0E	Este semnul unui număr?
	LD	IC,+0006	Va ocupa șase locatii.
	CALL	Z,19E8,RECLAIM-2	Este cerut numărul F-P.
	LD	A,(HL)	Se aduce din nou codul.
	INC	HL	Se mărește indicatorul.
	CP	+0D	'Carriage return'?
	JR	NZ,11A7,REMOVE-FP	Înapoi dacă nu. Dar dacă este se face o întoarcere simplă.
	RET		

THE EXECUTIVE ROUTINES (RUTINELE EXECUTIVULUI)

THE 'INITIALISATION' ROUTINE (RUTINA 'INITIALIZARE')

Punctul principal de intrare în această rutină este la START/NEW (11CB). Cînd intrarea se face prin START (0000), ca și cînd atunci se pune sub tensiune sistemul, registrul A conține zero și registrul pereche DE conține valoarea +FFFF. Oricum, punctul principal de intrare poate fi atins și urmînd execuția rutinei de comandă NEW.

THE 'NEW COMMAND' ROUTINE (RUTINA 'COMANDA NOUA')

11B7 NEW	DI		Dezactivare întrerupere mascabilă.
	LD	A,+FF	Fanionul NEW.
	LD	DE,(RANTOP)	Se păstrează valoarea existentă a lui RANTOP.
	EXX		Se încarcă registrii alternativi cu următoarele variabile sistem. Si acestea vor si păstrate.
	LD	BC,(P-RANT)	
	LD	DE,(RASP/PIP)	
	LD	HL,(UDG)	

Punctul principal de intrare.

11CB START/NEW	LD	BC	Se salvează fanionul pentru mai tîrziu.
	LD	A,+07	Se colorează marginea albă.
	OUT	(+FE),A	
	LD	A,+3F	Se încarcă registrul I cu valoarea +3F.
	LD	I,A	Se așteaptă 24 de stări I.
	DEFB	+00,+00,+00	
	DEFB	+00,+00,+00	

Acum se verifică memoria.

11DA RAM-CHEK	LD	H,D	Se transferă valoarea în DE.
	LD	L,E	(START = +FFFF, NEW = RANTOP)
11DC RAM-FILL	LD	(HL),+02	Se introduce valoarea +02 în fiecare locație pînă la +3FFF.
	DEC	HL	
	CP	H	
	JR	NZ,11DC,RAM-FILL	
11E2 RAM-READ	AND	A	Pregătire pentru scîdere.
	SBC	HL,DE	Fanionul de transport va fi resetat cînd se atinge vîrf.
	ADD	HL,DE	Se mărește indicatorul.
	INC	HL	Salt cînd este în vîrf.
	JR	NC,11EF,RAM-DONE	+02 ajunge +01.
	DEC	(HL)	Dacă ajunge zero, atunci RAM gresit.
	JR	Z,11EF,RAM-DONE	Se folosește ca vîrf HL actual.
	DEC	(HL)	+01 ajunge +00.
	JR	Z,11E2,RAM-READ	Se trece la următorul test dacă nu este eroare.
11EF RAM-DONE	DEC	HL	HL indică ultima locație curentă în ordinea lucrului.

Urmează refacerea sistemului de variabile 'păstrate'. (Ele sînt lipsite de sens dacă vin de la START.)

EXX			Se schimbă registrii.
LD	(P-RANT),BC		Se refac P-RANT, RASP/PIP & UDG.
LD	(RASP/PIP),DE		
LD	(UDG),HL		
EXX			
INC	B		Se testează fanionul START/NEW.
JR	Z,1219,RAM-SET		Salt înainte dacă se vine de la rutina de comandă NEW.

Se redefinesc variabilele sistemului cînd se vine de la START și se initializează spațiul grafic definit de utilizator.

LD	(P-RANT),HL	Vîrf RAM fizic.
LD	DE,+3EAF	Ultimul octet al lui 'U' în caracterul setat.
LD	BC,+00A8	Există acest număr de octeti în 21 de litere.
EX	DE,HL	Schimbare indicatori.
LDDR		Acum se copiază forma

EX	DE,HL	caracterelor de litere de la 'A' la 'U'.
INC	HL	Se schimbă indicatorii înainte.
LD	(UDG),HL	Se indică primul octet.
DEC	HL	Acum se setează UDG.
LD	BC,+0040	Se scade o locație.
LD	(RASP/PIP),BC	Se setează variabilele sistem RASP & PIP.

Restul rutinei este comun și pentru operațiile START și pentru operațiile NEW.

1219 RAM-SET	LD (RAMTOP),HL	Se setează RAMTOP.
	LD HL,+3C00	Se inițializează variabila sistem CHARS.
	LD (CHARS),HL	

În continuare se setează stiva calculatorului.

LD	HL,(RAMTOP)	Locația din vîrf va conține +3E.
LD	(HL),+3E	Următoarea locație este făcută să contină zero.
DEC	HL	Aceste două locații reprezintă 'ultima intrare'.
LD	SP,HL	Se coboară două locații pentru a afla valoarea corectă pentru ERR-SP.
DEC	HL	
DEC	HL	
LD	(ERR-SP),HL	

Inițializarea rutinei continuă cu:

IM	1	Se folosește modul 1 de întrerupere.
LD	IY,+5C3A	IY conține întotdeauna +ERR-NR.
EI		Se poate activa acum întreruperea mascabilă.
		Ceasul de timp real va fi mărit și tastatura scanată la fiecare a 1/50 partea dintr-o secundă.
LD	HL,+5CB6	Adresa de bază a zonei canalului de informații.
LD	(CHANS),HL	Canalul de informații inițial este mutat din tabel (15AF) în zona canalului de informații.
LD	DE,15AF	Variabila sistem DATADD este făcută să arate ultima locație a canalului de informații.
LD	BC,+0015	Iar PROC & VARS va indica locația următoare.
EX	DE,HL	Indicatorul de sfîrșit al spațiului variabilelor.
LDIR		Se incrementează cu o locație pentru a găsi valoarea pentru E-LINE.
EX	DE,HL	Se face ca linia editată să contină un singur 'carriage return'.
DEC	HL	Acum se introduce un marcator de sfîrșit.
LD	(DATADD),HL	Se mută cu o locație pentru a se găsi valoarea pentru WORKSP, STKBOT & STKEND.
INC	HL	
LD	(PROC),HL	
LD	(VARS),HL	
LD	(HL),+80	
INC	HL	
LD	(E-LINE),HL	
LF	(HL),+0D	
INC	HL	
LD	(HL),+80	
INC	HL	
LD	(WORKSP),HL	
LD	(STKBOT),HL	
LD	(STKEND),HL	
LD	A,+38	
LD	(ATTR-P),A	Inițializare variabile sistem pentru culoare pe: FLASH 0, BRIGHT 0, PAPER 7, & INK 0.
LD	(ATTR-T),A	
LD	(BORDCR),A	
LD	HL,+0523	Se inițializează variabilele sistem REPDEL & REPPER.
LD	(REPDEL),HL	KSTATE-0 va conține +FF.
DEC	(KSTATE-0)	KSTATE-4 va conține +FF.
DEC	(KSTATE-4)	În continuare se mută sirul inițial de informații din tabelul său în spațiul sirurilor.
LD	HL,+15C6	Se mută 'folosire imprimantă'.
LD	DE,+5C10	
LD	BC,+000E	
LDIR		
SET	1,(FLAGS)	

CALL	OEDF,CLEAR-PRB	si se sterge bufferul imprimantei.
LD	(DF-SZ),+02	Se setează mărimea părții inferioare a ecranului si se sterge tot ecranul.
CALL	OD6B,CLS	Acum se tipărește mesajul '© 1982 Sinclair Research Ltd' pe ultima linie.
XDR	A	Seenal 'se cere stergerea părții inferioare'.
LD	DE,+1538	Salt înainte în bucla principală de executie.
CALL	OC0A,PD-NSB	
SET	5,(TV-FLAG)	
JR	12A9,MAIN-1	

THE 'MAIN EXECUTION' LOOP (BUCLA 'EXECUTIE PRINCIPALA')

Bucla executie principală se întinde de la locația 12A2 pînă la locația 15AE si ea controlează 'modul de editare', executia comenzilor directe si prezentarea activității.

12A2 MAIN-EXEC	LD	(DF-SZ),+02	Mărimea părții inferioare a ecranului este de două linii.
	CALL	1795,AUTO-LIST	Se execută o listare automată.
12A9 MAIN-1	CALL	16B0,SET-MIN	Toate zonele de la E-LINE în continuare au minimul configurației dat.
12AC MAIN-2	LD	A,+00	Se deschide canalul 'K' înainte de a se apela EDITOR.
	CALL	1601,CHAN-OPEN	EDITOR este apelat pentru a permite utilizatorului să construiască o linie BASIC.
	CALL	0F2C,EDITOR	Linia curentă este scanată pentru corectarea sintaxei.
	CALL	1B17,LINE-SCAN	Salt înainte dacă sintaxa este corectă.
	BIT	7,(ERR-NR)	Salt înainte dacă s-a folosit alt canal decît canalul 'K'.
	JR	NZ,12CF,MAIN-3	Se indică începutul unei linii cu eroare.
	BIT	4,(FLAGS2)	Se mută din nou forța punctului de lansare din această linie.
	JR	Z,1303,MAIN-4	Se resetează ERR-NR si salt înapoi la MAIN-2 lăsînd listarea neschimbată.
	LD	HL,(E-LINE)	
	CALL	11A7,REMOVE-EP	
	LD	(ERR-NR),+FF	
	JR	12AC,MAIN-2	

'Linia editată' a trecut partea de sintaxă si cele trei tipuri de linii care sînt posibile vor trebui să fie deosebite una de cealaltă.

12CF MAIN-3	LD	HL,(E-LINE)	Se indică începutul liniei.
	LD	(CH-ADD),HL	Se setează începutul lui CH-ADD.
	CALL	19FB,E-LINE-NO	Se aduce numărul oricărui linii în BC. Este numărul liniei unul valid?
	LD	A,B	Salt dacă este, si se adaugă noua linie programului existent.
	OR	C	Se aduce primul caracter al liniei si se testează dacă linia este doar 'carriage return'. Dacă este, salt înapoi.
	JR	NZ,155D,MAIN-ADD	
	RST	0018	
	CP	+0D	
	JR	Z,12A2,MAIN-EXEC	

'Linia editată' trebuie să înceapă cu o comandă BASIC directă si această linie va fi prima linie ce trebuie interpretată.

	BIT	0,(FLAGS2)	Se sterge întregul ecran dacă nu fanioanele nu arată că nu este necesar.
	CALL	NZ,ODAF,CL-ALL	Oricum se sterge partea de jos a ecranului.
	CALL	OD6E,CLS-LOWER	Se încarcă valoarea potrivită pentru contor defilare.
	LD	A,+19	Seenal 'executie linie'.
	SUB	(S-POSN-hi)	Asigurare că ERR-NR este corect.
	LD	(SCR-CT),A	Se prelucrează prima instrucție din linie.
	SET	7,(FLAGS)	
	LD	(ERR-NR),+FF	
	LD	(NSPPC),+01	

CALL 138A, PROG-RUN

Acum se interpretează linia.
Notă: Adresa 1303 este
încărcată în stiva
calculatorului și este
adresată cu ERR-SP.

După ce linia a fost interpretată și toate acțiunile consecutive au fost executate se execută o revenire în MAIN-4, așa că se poate face o prezentare.

1303 MAIN-4	HALT	Se activează întreruperea mascabilă.
	RES 5, (FLAGS)	Seamnal 'gata pentru o tastă nouă'.
	BIT 1, (FLAGS2)	Se golește bufferul imprimantei dacă a fost folosit.
	CALL NZ, OECB, COPY-BUFF	Se aduce numărul de eroare și se incrementează.
1313 MAIN-6	LD A, (ERR-NR)	Se salvează noua valoare.
	INC A	Variabilele sistem FLAGX, X-PTR-hi & DEFADD sînt puse toate zero.
	PUSH AF	
	LD HL, +0000	
	LD (FLAGX), H	
	LD (X-PTR-hi), H	
	LD (DEFADD), HL	
	LD HL, +0001	Se asigură că sirul +00 indică canalul 'K'.
	LD (STRMS-6), HL	Se șterg toate spațiile de lucru și stiva calculatorului.
	CALL 1680, SET-MIN	Seamnal 'mod editare'.
	RES 5, (FLAGX)	Se șterge partea inferioară a ecranului.
	CALL 006E, CLS-LOWER	Seamnal 'partea inferioară a ecranului cere ștergere'.
	SET 5, (TV-FLAG)	Se aduce valoarea reportată.
	POP AF	Se face o copie în B.
	LD B, A	Salt înainte cu prezentarea numerelor '0 la 9'.
	CP +0A	Se adună valorile ASCII ale literelor deplasabile.
	JR C, 1330, MAIN-5	Se tipărește codul prezentat urmat de un 'spatiu'.
	ADD A, +07	
1330 MAIN-5	CALL 15EF, OUT-CODE	Se aduce valoarea prezentată și se folosește pentru a identifica mesajul prezentat cerut.
	LD A, +20	Se tipărește mesajul care este urmat de o 'virgulă' și de un 'spatiu'.
	RST 0010, PRINT-A-1	Acum se aduce numărul liniei curente și se tipărește.
	LD A, B	El va fi urmat de ' '.
	LD DE, +1391	
	CALL 0C0A, PD-MSG	Se aduce numărul instrucției curente în registrul pereche BC și se tipărește.
	XOR A	Se șterge spațiul de editare.
	LD DE, +1536	Se aduce din nou numărul de eroare.
	CALL 0C0A, PD-MSG	Se incrementează ca de obicei
	LD BC, (PPC)	Dacă programul a reușit nu mai există 'continuare', deci se execută salt.
	CALL 1A1B, OUT-NUM1	Dacă programul se oprește cu 'instrucția STOP' sau 'BREAK în program', continuarea se face de la următoarea instrucțiune; altfel SUBPPC este neschimbat.
	LD A, +3A	Variabilele sistem OLBPCC & OSPPC sînt făcute să continue linia de continuare și numerele instrucției.
	RST 0010, PRINT-A-1	Valorile folosite sînt acelea care în PPC & SUBPPC, dar nu în NSPPC, arată că 'break' este necesar înainte de 'jump' (salt).
	LD C, (SUBPPC)	(de exemplu, după o
	LD B, +00	
	CALL 1A1B, OUT-NUM1	
	CALL 1097, CLEAR-SP	
	LD A, (ERR-NR)	
	INC A	
	JR Z, 1386, MAIN-9	
	CP +09	
	JR Z, 1373, MAIN-6	
	CP +15	
1373 MAIN-6	JR NZ, 1376, MAIN-7	
	INC (SUBPPC)	
1376 MAIN-7	LD BC, +0003	
	LD DE, +5C70	
	LD HL, +5C44	
	BIT 7, (NSPPC)	
	JR Z, 1384, MAIN-8	
	ADD HL, BC	
1384 MAIN-8	LDDR	

1386 MAIN-9	LD	(NSPPC),+FF	instructie 80 TD, etc.) Se resetează NSPPC pentru a indica 'nu este salt'.
	RES	3,(FLAGS)	Se selectează 'modul K'.
	JP	12AC,MAIN-2	In final se execută salt înapoi, dar nu apare listarea programului pînă ce nu este cerută.

THE REPORT MESSAGES (PREZENTAREA MESAJELOR)

Fiecare mesaj este dat cu ultimul caracter invertit (+80 hexa).

1391 DEFB +80	-	trecere peste octetul initial
1392 Report 0	-	'OK'
1394 Report 1	-	'NEXT fără FOR'
13A4 Report 2	-	'Nu se găsește variabila'
13B6 Report 3	-	'Subscriere gresită'
13C6 Report 4	-	'Afară din memorie'
13D2 Report 5	-	'Afară din ecran'
13DF Report 6	-	'Număr prea mare'
13ED Report 7	-	'RETURN fără GOSUB'
1401 Report 8	-	'Sfîrsitul fisierului'
140C Report 9	-	'Instrucțiunea STOP'
141A Report A	-	'Argument invalid'
142A Report B	-	'Intreg afară din rang'
143E Report C	-	'Fără sens în BASIC'
144F Report D	-	'Repetare BREAK-CONT'
1463 Report E	-	'Afară din DATA'
146E Report F	-	'Nume fisier invalid'
147F Report 6	-	'Nu mai sînt locuri pentru linie'
148F Report H	-	'STOP în INPUT'
149C Report J	-	'FOR fără NEXT'
14AC Report J	-	'Unitate I/O invalidată'
14BE Report K	-	'Culoare invalidată'
14CC Report L	-	'BREAK în program'
14DE Report M	-	'Nu este bun RANTOP'
14EC Report N	-	'Instrucție pierdută'
14FA Report O	-	'Sir invalidat'
1508 Report P	-	'FN fără DEF'
1516 Report Q	-	'Eroare parametru'
1525 Report R	-	'Eroare încărcare bandă'

Mai există următoarele două mesaje.

1537	.	.	-	ca o 'virgulă' și un 'spatiu'
1539	'	©	1982	Sinclair Research Ltd'

Prezentarea G - nu mai sînt locuri pentru linie.

1555 REPORT-G	LD	A,+10	'G' are codul '10+07+30'.
	LD	BC,+0000	Se șterge BC.
	JP	1313,MAIN-6	Salt înapoi pentru prezentare

THE 'MAIN-ADD' SUBROUTINE

Această subrutină permite unei noi linii BASIC să fie adăugată unui program BASIC existent, în spațiul programului. Dacă o linie are atît o versiune veche cît și una nouă, atunci cea veche se 'reface'. O linie nouă care conține doar un număr de linie nu va trimisi în zona programului.

155D MAIN-ADD	LD	(E-PPC),XC	Se face noul număr de linie linie 'curentă'.
	LD	HL,(CH-ADD)	Se aduce CH-ADD și se
	EX	DE,HL	salvează în DE.
	LD	HL,+1555	Se salvează adresa lui
	PUSH	HL	REPORT-G în stiva
			calculatorului.
	LD	HL,(WORKSP)	Se aduce WORKSP.
	SCF		Se determină lungimea liniei
	SBC	HL,DE	de după numărul liniei care
			a avut 'carriage return'.
	PUSH	HL	Se salvează lungimea.
	LD	H,B	Se mută numărul liniei în
	LD	L,C	registru pereche HL.
	CALL	196E,LINE-ADDR	Există o linie care are acest
			număr?
	JR	NZ,157D,MAIN-ADD1	Salt dacă nu există.
	CALL	1988,NEXT-ONE	Se caută lungimea liniei
	CALL	19E8,RECLAIM-2	'vechi' și se reface.
157D MAIN-ADD1	POP	BC	Se aduce lungimea liniei

LD	A,C	'noi' si salt înainte dacă este numai 'număr de linie si carriage return'.
DEC	A	
OR	I	
JR	15AB,MAIN-ADD2	
PUSH	BC	Se salvează lungimea.
INC	BC	Sînt necesare patru locatii suplimentare.
INC	BC	De exemplu, două pentru număr & două pentru lungime.
INC	BC	HL va indica locatia dinainte de 'destinatia'.
INC	BC	Se salvează valoarea curentă a lui PROG pentru a preveni deteriorarea cînd se adaugă prima linie.
DEC	HL	Se creează spatiu pentru noua linie.
LD	DE,(PROG)	Se aduce vechea valoare a lui PROG si se reface.
PUSH	DE	Se face o copie a lungimii liniei (fără parametrul).
CALL	1655,MAKE-ROOM	DE va indica locatia finală a noului spatiu si HL va indica 'carriage return' al unei linii noi din spatiul de editare.
POP	HL	Acum se copiază linia.
LD	(PROG),HL	Se aduce numărul liniei.
POP	BC	Destinatia se găsește în HL & numărul în DE.
PUSH	BC	Se aduce noua lungime a liniei.
INC	DE	Octetul superior al lungimii.
LD	HL,(WORKSP)	Octetul inferior al lungimii.
DEC	HL	Octetul inferior al numărului liniei.
DEC	HL	
LDDR		
LD	HL,(E-PPC)	Octetul superior al numărului liniei.
EX	DE,HL	Salvare adresă REPORT-G.
POP	BC	Salt înapoi si de data asta se execută o listare automată.
LD	(HL),B	
DEC	HL	
LD	(HL),C	
DEC	HL	
LD	(HL),E	
DEC	HL	
LD	(HL),D	
15AB MAIN-ADD2	POP AF	
	JP 12A2,MAIN-EXEC	

THE 'INITIAL CHANNEL INFORMATION'

Initial sînt patru canale - 'K', 'S', 'R' & 'P' - pentru a comunica cu 'tastatura', 'ecranul', 'spatiul de lucru' si 'imprimanta'. Pentru fiecare canal adresa rutinei de iesire vine după adresa rutinei de intrare si codul canalului.

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		-	Indicator sfîrsit.

Prezentare J - unitate I/O invalidată.

15C4	REPORT-J	RST	0008,ERROR-1	Se apelează rutina de tratare eroare.
		DEFB	+12	

THE 'INITIAL STREAM DATA' ('SIRUL INITIAL DE INFORMATII')

15C6	DEFB	01	00	-	sirul +FD conduce la canalul 'K'
15C8	DEFB	06	00	-	sirul +FE conduce la canalul 'S'
15CA	DEFB	0B	00	-	sirul +FF conduce la canalul 'R'
15CC	DEFB	01	00	-	sirul +00 conduce la canalul 'K'
15CE	DEFB	01	00	-	sirul +01 conduce la canalul 'K'
15D0	DEFB	06	00	-	sirul +02 conduce la canalul 'S'

15D2 DEFB 10 00 - sirul +03 conduce la canalul 'P'

THE 'WAIT-KEY' SUBROUTINE (SUBROUTINA 'ASTEPTARE TASTA')

Această subrutină este o subrutină de control pentru apelarea subrutinei curente de intrare.

15D4 WAIT-KEY	JIT	5, (TV-FLAG)	Salt înainte dacă fanioanele indică faptul că partea de jos a ecranului nu a cerut stergere.
	JR	NZ, 15DE, WAIT-KEY1	
	SET	3, (TV-FLAG)	Altfel se semnaleză 'se consideră că s-a schimbat modul'.
15DE WAIT-KEY1	CALL	15E6, INPUT-AD	Se apelează indirect rutina de intrare, prin INPUT-AD.
	RET	C	Revenire.
	JR	Z, 15DE, WAIT-KEY1	'Dacă nici o tastă nu a fost apăsată' se resetează fanioanele de transport și de zero; altfel se semnaleză eroare.

Prezentarea 8 - Sfirsitul fisierului

15E4 REPORT-8	RST	0008, ERROR-1	Se apelează rutina de tratare eroare.
	DEFB	+07	

THE 'INPUT-AD' SUBROUTINE (SUBROUTINA 'INTRODUCERE ADRESA')

Registrii sînt salvati și HL va indica adresa de intrare.

15E6 INPUT-AD	EXX	
	PUSH	HL
	LD	HL, (CURCHL)
	INC	HL
	INC	HL
	JR	15F7, CALL-SUB

THE 'MAIN PRINTING' SUBROUTINE

Această subrutină este apelată sau printr-o valoare absolută sau prin codul propriu de caracter în registrul A.

15EF OUT-CODE	LD	E, +30	Se crește valoarea din registrul A cu +30.
	ADD	A, E	
15F2 PRINT-A-2	EXX		Se salvează din nou registrii
	PUSH	HL	
	LD	HL, (CURCHL)	Se aduce adresa de bază a canalului curent. Aceasta va indica o adresa de ieșire.

Acum se apelează subrutina actuală. HL va indica adresa de ieșire sau de intrare, după cum se indică.

15F7 CALL-SUB	LD	E, (HL)	Se aduce octetul inferior.
	INC	HL	
	LD	D, (HL)	Se aduce octetul superior.
	EX	DE, HL	
	CALL	162C, CALL-JUMP	Se mută adresa în registrul pereche HL.
	POP	HL	Se cheamă subrutina actuală.
	EXX		Se refac registrii.
	RET		Se revine dacă nu a intervenit o eroare.

THE 'CHAN-OPEN' SUBROUTINE (SUBROUTINA 'DESCHIDERE CANAL')

Subrutina se apelează cu registrul A conținând un număr valid de sir - în mod normal de la +FD la +03. Apoi depinde de sirul de informații dacă un canal anumit va fi făcut canal curent.

1601 CHAN-OPEN	ADD	A, A	Valoarea din registrul A este dublată și apoi mărită cu +16. Rezultatul se mută în L. Adresa 5C16 este adresa de bază a sirului +00.
	ADD	A, +16	
	LD	L, A	
	LD	H, +5C	
	LD	E, (HL)	Se aduce primul octet al sirului de informații cerut; apoi al doilea bit. Va apare eroare dacă ambii
	INC	HL	
	LD	D, (HL)	
	LD	A, D	

OR E octeti sînt zero; altfel
JR NZ,1610,CHAN-OP-1 salt înainte.

Prezentare 0 - Sir invalid

160E REPORT-0 RST 0008,ERROR-1 Se apelează rutina de tratare
DEFB +17 eroare.

Folosind acum sirul de informatii se găsește adresa de bază a canalului de
informatii asociat acestui sir.

1610 CHAN-OP-1 DEC DE Se reduce sirul de informatii
LD HL,(CHANS) Adresa de bază pentru toată
ADD HL,DE Se formează adresa cerută în
acest spatiu.

THE 'CHAN-FLAG' SUBROUTINE

Fanioanele corespunzătoare pentru diferitele canale sînt setate de această
subrutină.

1615 CHAN-FLAG LD (CURCHL),HL Registrul pereche HL contine
adresa de bază a unui anumit
canal.
RES 4,(FLAGS2) Semnal 'se folosește alt
canal decît canalul K'.
INC HL Se trece peste adresele de
INC HL de iesire și intrare și HL va
INC HL indica codul canalului.
LD C,(HL) Se aduce codul.
LD HL,+162D Adresa de bază a 'tabelului
de trecere cod canal'.
CALL 16DC,INDEXER Se indexează în acest tabel
și se localizează deplasamentu
RET NC cerut; se revine dacă nu s-
LD D,+00 găsit un cod canal identic.
LD E,(HL) Se trece deplasamentul în
ADD HL,DE registrul pereche DE.
162C CALL-JUMP JP (HL) Salt înainte la rutina
corespunzătoare pentru
setarea fanioanelor.

THE 'CHANNEL CODE LOOK-UP' TABLE

162D	DEFB	4B	06	- canal 'K'	deplasament +06	adresa 1634
162F	DEFB	53	12	- canal 'S'	deplasament +12	adresa 1642
1631	DEFB	50	1B	- canal 'P'	deplasament +1B	adresa 164D
1633	DEFB	00		- marcator sfîrsit		

THE 'CHANNEL 'K' FLAG' SUBROUTINE (SUBRUTINA 'FANION CANAL 'K'')

1634 CHAN-K SET 0,(TV-FLAG) Semnal 'se folosește partea
inferioară a ecranului'.
RES 5,(FLAGS) Semnal 'pregătit pentru o
tastă'.
SET 4,(FLAGS2) Semnal 'se folosește canalul
'K'.
JR 1646,CHAN-S-1 Salt înainte.

THE 'CHANNEL 'S' FLAG' SUBROUTINE (SUBRUTINA 'FANION CANAL 'S'')

1642 CHAN-S RES 0,(TV-FLAG) Semnal 'se folosește partea
principală a ecranului'.
1646 CHAN-S-1 RES 1,(FLAGS) Semnal 'imprimanta nu a fost
folosită'.
JP 0D4D,TEMPS Iesire prin TEMPS așa încît
să se seteze variabilele
sistem culoare.

THE 'CHANNEL 'P' FLAG' SUBROUTINE (SUBRUTINA 'FANION CANAL 'P'')

164D CHAN-P SET 1,(FLAGS) Semnal 'imprimanta în
RET funcțiune'.

THE 'MAKE-ROOM' SUBROUTINE (SUBRUTINA 'FACE LOC')

Aceasta este o subrutină foarte importantă. Ea este apelată în multe ocazii

pentru a 'lărgi' o zonă. În toate cazurile registrul pereche HL indică locația de după locul unde 'camera' este cerută și registrul pereche BC conținea lungimea 'camerei' de care este nevoie.

Când se cere un singur spațiu, subrutina este introdusă prin ONE-SPACE.

1652 ONE-SPACE	LD	BC,+0001	Se cere doar o singură locație aditională.
1655 MAKE-ROOM	PUSH HL		Salvare indicator.
	CALL	1F05,TEST-ROOM	Se asigură că este accesibilă suficientă memorie pentru ca taskul să fi preluat.
	POP HL		Refacere indicator.
	CALL	1664,POINTERS	Se schimbă toți indicatorii înainte de a realiza 'camera' (locația).
	LD	HL,(STKEND)	HL conține noul STKEND.
	EX	DE,HL	Schimbare 'vechi' și 'nou'.
	LDBR		Acum se realizează 'locația' și revenire.
	RET		

Notă: Această subrutină revine cu registrul pereche HL indicând locația dinaintea noii 'camere' și registrul pereche DE indicând sfârșitul noii locații. De aceea, noua 'cameră' are descrierea de la '(HL)+1' la '(DE)+1' inclusiv.

După cum 'noile locații' retin 'vechile lor valori' este de asemenea posibil să se considere noua 'cameră' ca fiind făcută după locația originală '(HL)' și astfel va avea descrierea de la '(HL)+2' la '(DE)+1'.

De fapt programatorul pare să aibe o preferință pentru a 'două descriere' și acest fapt poate duce la confuzii.

THE 'POINTERS' SUBROUTINE (SUBROUTINA 'INDICATORI')

Intotdeauna când o zonă trebuie 'făcută' sau 'refăcută' variabilele sistem care adresează locația dincolo de 'poziția' schimbării trebuie modificată după cum se cere. La intrare, în registrul pereche BC se află numărul octetilor indusi, iar în registrul pereche HL se află adresele locației dinaintea 'poziției'.

1664 POINTERS	PUSH AF		Se salvează registrii.
	PUSH HL		Se copiază adresa 'poziției'.
	LD	HL,+5C4B	Aceasta este adresa lui VARS,
	LD	A,+0E	primul dintre cei 14 indicatori sistem.

Acum se intră într-o buclă în care se va lua în considerare fiecare indicator pe rând. Se vor schimba doar acei indicatori care arată dincolo de 'poziție'.

166B PTR-NEXT	LD	E,(HL)	Se aduc cei doi octeti ai indicatorului curent.
	INC	HL	
	LD	D,(HL)	
	EX	(SP),HL	Se schimbă variabila sistem cu adresa 'poziției'.
	AND	A	Fanionul de transport va fi setat dacă adresa variabilei sistemului trebuie mărită.
	SBC	HL,DE	Se reface 'poziția'.
	ADD	HL,DE	Salt înainte dacă indicatorul trebuie să fie la stînga; altfel se schimbă.
	EX	(SP),HL	Se salvează valoarea veche.
	JR	NC,167F,PTR-DONE	Acum se adună la valoarea din BC vechea valoare.
	PUSH DE		
	EX	DE,HL	Se introduce noua valoare în variabila sistem - octetul superior înaintea octetului inferior.
	ADD	HL,BC	Se indică octetul superior.
	EX	DE,HL	Se aduce vechea valoare.
	LD	(HL),D	Se indică următoarea variabilă sistem și salt înapoi pînă cînd toți cei 14 indicatori au fost considerați.
	DEC	HL	
	LD	(HL),E	
167F PTR-DONE	INC	HL	
	POP	DE	
	INC	HL	
	DEC	A	
	JR	NZ,166B,PTR-NEXT	

Acum se află marimea blocului ce trebuie mutat.

EX	DE,HL	Se pune vechea valoare a lui STKEND în HL și se refac ceilalți regiștri.
POP	DE	
POP	AF	

AND	A	Acum se determină diferența
SBC	HL, DE	între vechea valoare a lui
LD	B, H	STKEND și 'poziție'.
LD	C, L	Rezultatul se transferă în BC
INC	BC	și se adună '1' pentru octet
		inclusiv.
ADD	HL, DE	Reformarea vechii valori a
EX	DE, HL	lui STKEND și trecerea ei în
RET		DE înainte de revenire.

THE 'COLLECT A LINE NUMBER' SUBROUTINE (SUBROUTINA 'COLECTARE NUMAR DE LINIE')

La intrare, registrul pereche HL va indica locația luată în considerare. Dacă locația conține o valoare care reprezintă partea superioară a unui octet care convine pentru un număr de linie, atunci numărul de linie este trecut în DE. După cum, dacă nu este așa, locația adresată prin DE este testată pe loc; și dacă și acest test este nereușit, atunci se reface numărul de linie zero.

168F LINE-ZERO	DEFB	+00	Număr de linie zero.
	DEFB	+00	
1691 LINE-NO-A	EX	DE, HL	Se consideră celălalt
			indicator.
	LD	DE, +168F	Se folosește numărul de
			linie zero.

În mod normal, punctul de intrare este LINE-NO.

1695 LINE-NO	LD	A, (HL)	Se aduce octetul superior și
	AND	+C0	se testează.
	JR	NZ, 1691, LINE-NO-A	Salt înapoi dacă nu este
			convenabil.
	LD	D, (HL)	Se aduce octetul inferior și
	INC	HL	se revine.
	LD	E, (HL)	
	RET		

THE 'RESERVE' SUBROUTINE (SUBROUTINA 'REZERVARE')

În mod normal, această subrutină este apelată folosind RST 0030, BC-SPACES.

La intrare, ultima valoare din stiva calculatorului este WORKSP și valoarea de după ea este numărul spațiilor care trebuie rezervate.

Această subrutină întotdeauna realizează 'camere' între spațiul de lucru existent și stiva calculatorului.

169E RESERVE	LD	HL, (STKBOT)	Se aduce valoarea curentă a
	DEC	HL	lui STKBOT și se
			decrementează pentru a
			obține ultima locație a
			spațiului de lucru.
	CALL	1655, MAKE-ROOM	Se fac 'BC spații'.
	INC	HL	Se indică primul spațiu nou
	INC	HL	și apoi al doilea.
	POP	BC	Se aduce vechea valoare a
	LD	(WORKSP), BC	lui WORKSP și se reface.
	POP	BC	Se reface BC - numărul de
			spații.
	EX	DE, HL	Schimbare indicatori.
	INC	HL	HL indică primul octet
			înlocuit.
	RET		Revenire.

Notă: Se poate lua de asemenea în considerare că subrutina execută revenirea cu registrul pereche DE indicând 'primul octet adițional' și cu registrul pereche HL indicând 'ultimul octet adițional', acești octeți adiționali fiind adăugați după locația originară '(HL)+1'.

THE 'SET-MIN' SUBROUTINE

Această subrutină resetează zona de editare și zonele de după acesta la mărimea lor minimă. De fapt, ea 'șterge' zonele.

16B0 SET-MIN	LD	HL, (E-LINE)	Se aduce conținutul E-LINE.
	LD	(HL), +0D	Zona de editare conține doar
	LD	(K-CUR), HL	caracterul 'carriage return'
	INC	HL	și marcatorul de sfârșit.
	LD	(HL), +80	
	INC	HL	Avans pentru curățirea
	LD	(WORKSP), HL	spațiului de lucru.

Această intrare 'va curăța' spațiul de lucru și stiva calculatorului.

16BF SET-WORK	LD	HL, (WORKSP)	Se aduce conținutul lui
---------------	----	--------------	-------------------------

LD (STKBOT),HL WORKSP.
Aceasta curată stiva.

Această intrare 'va curăta' numai stiva calculatorului.

16C5 SET-STK LD HL,(STKBOT) Se aduce continutul lui STKBOT.
LD (STKEND),HL Aceasta curată stiva.

In toate aceste cazuri MEM adresează zona de memorie a calculatorului.

PUSH HL Se salvează STKEND.
LD HL,+5C92 Baza zonei de memorie.
LD (MEM),HL Se setează MEM de la această adresă.
POP HL Se reface STKEND în registrul pereche HL înainte de a se reveni.
RET

THE 'RECLAIM THE EDIT-LINE' SUBROUTINE (SUBROUTINA 'REFACERE LINIE EDITARE')

16D4 REC-EDIT LD DE,(E-LINE) Se aduce continutul lui E-LINE.
JP 19E5,RECLAIM-1 Refacere memorie.

THE 'INDEXER' SUBROUTINE (SUBROUTINA 'INDEXARE')

Această subrutină este folosită de mai multe ori pentru a căuta prin tabele. Intrarea este la INDEXER.

16DB INDEXER-1 INC HL Avans pentru considerarea următoarelor perechi de intrări.
16DC INDEXER LD A,(HL) Se aduce prima dintr-o pereche de intrări, dar se revine dacă ea este zero - marculatorul de sfîrsit.
AND A Se compară cu codul înlocuit.
RET Z Se indică a doua intrare.
CP C Salt înapoi dacă nu s-a găsit intrarea corectă.
INC HL Fanionul de transport este setat după o căutare reușită.
JR NZ,16DB,INDEXER-1
SCF
RET

THE 'CLOSE#' COMMAND ROUTINE (RUTINA DE COMANDA 'CLOSE#' (închiderea#))

Această comandă permite utilizatorului să închidă (CLOSE) siruri. Pentru siruri +00 la +03 informația sir 'ințială' este refăcută și aceste siruri nu pot fi închise.

16E5 CLOSE CALL 171E,STR-DATA Se aduce informația existentă pentru sir.
CALL 1701,CLOSE-2 Se verifică codul în acest canal al sirului.
LD BC,+0000 Pregătire pentru punerea informației sirului pe zero.
LD DE,+A3E2 Pregătire pentru a identifica utilizarea sirurilor de la +00 la +03.
EX DE,HL Fanionul de transport se setează pentru siruri de la +04 la +0F.
ADD HL,DE Salt înainte cu aceste siruri în caz contrar se află intrarea corectă în tabelul 'informație sir ințială'.
JR C,16FC,CLOSE-1 Se aduce informația ințială pentru sirurile de la +00 la +03.
LD BC,+15D4 Acum se introduce informația; sau zero & zero sau valoarea ințială.
ADD HL,BC
LD C,(HL)
INC HL
LD B,(HL)
EX DE,HL
LD (HL),C
INC HL
LD (HL),B
RET

THE 'CLOSE-2' SUBROUTINE

Codul canalului asociat sirului care a fost închis trebuie să fie 'K', 'S' sau 'P'.

1701 CLOSE-2 PUSH HL Se salvează adresa informației sir.

LD	HL,(CHANS)	Se aduce adresa de bază a
ADD	HL,BC	zonei canalului de informație
		și se caută informația de
		canal pentru sirul care a
		fost închis.
INC	HL	Se trece peste adresele
INC	HL	subrutinei și se selectează
INC	HL	codul pentru acel canal.
LD	C,(HL)	
EX	DE,HL	Salvare indicator.
LD	HL,+1716	Adresa de bază pentru tabelul
		'trecere prin sir închis'.
CALL	16DC,INDEXER	Se indexează în acest tabel
		și se localizează deplasamentul
		cerut.
LD	C,(HL)	Se trece deplasamentul în
LD	B,+00	registru pereche BC.
ADD	HL,BC	Salt înainte la rutina
JP	(HL)	corespunzătoare.

THE 'CLOSE STREAM LOOK-UP' TABLE (TABELUL 'TRECERE PRIN SIR INCHIS')

1716	DEFB	48	05	-	canal 'K'	deplasament +05,	adresa 171C
1718	DEFB	53	03	-	canal 'S'	deplasament +03,	adresa 171C
171A	DEFB	50	01	-	canal 'P'	deplasament +01,	adresa 171C

Notă: Nu este nici un marcator de sfârșit la sfârșitul acestui tabel.

THE 'CLOSE STREAM' SUBROUTINE (SUBROUTINA 'INCHIDE SIR')

171C	CLOSE-STR	POP	HL	Se aduce	indicatorul
		RET		informației	canal și se
				revine.	

THE 'STREAM DATA' SUBROUTINE (SUBROUTINA 'INFORMATII SIR')

Această subrutină returnează în registrul pereche HL informația sir pentru un sir dat.

171E	STR-DATA	CALL	1E94,STK-TD-A	Numărul sirului dat este scos
		CP	+10	din stiva calculatorului.
		JR	C,1727,STR-DATA1	Va apărea eroare dacă numărul
				sirului este mai mare decât
				+0F.

Prezentarea 0 - Sir invalid

1725	REPORT-0	DEFB	0008,ERROR-1	Se apelează rutina de tratare
			+17	eroare.

Se continuă cu numere de sir valide.

1727	STR-DATA1	ADD	A,+03	Acum se aranjează de la +03
		RLCA		la +12, apoi de la +06 la
		LD	HL,+5C10	+24.
		LD	C,A	Adresa de bază a spațiului
		LD	B,+00	informațiilor sir.
		ADD	HL,BC	Se mută informația sir în
		LD	C,(HL)	registru pereche BC.
		INC	HL	Se indexează în spațiul
		LD	B,(HL)	informațiilor și se aduc doi
		DEC	HL	octeți de informații în
		RET		registru pereche BC.
				Indicatorul adresează primul
				octet de informații înaintea
				revenirii.

THE 'OPEN# COMMAND ROUTINE (RUTINA DE COMANDA 'OPEN#' (deschidere))

Această comandă permite utilizatorului să deschidă siruri. Un cod canal trebuie înlocuit și el trebuie să fie 'K', 'k', 'S', 's', 'P' sau 'p'.
De notat că nu se face nici o încercare pentru a da sirurilor +00 la +03 informațiile lor initiale.

1736	OPEN	RST	0028,FP-CALC	Utilizare CALCULATOR.
		DEFB	+01,exchange	Se schimbă numărul de sir și
		DEFB	+38,end-calc	codul canalului.
		CALL	171E,STR-DATA	Se aduc informații pentru
				sir.
		LD	A,B	Salt înainte dacă ambii
		OR	C	octeți de informații sînt

	JR	Z,1756,OPEN-1	zero, adică sirul a fost într-o stare închisă.
	EX	DE,HL	Se salvează DE.
	LD	HL,(CHANS)	Se aduce CHANS - adresa de bază a canalului de
	ADD	HL,BC	informații și se găsește
	INC	HL	codul canalului asociat cu
	INC	HL	sirul care a fost deschis.
	LD	A,(HL)	
	EX	DE,HL	Revenire DE.
	CP	+4B	Codul adus din zona canalului
	JR	Z,1756,OPEN-1	de informații trebuie să fie
	CP	+53	'K', 'S' sau 'P'; dacă nu
	JR	Z,1756,OPEN-1	este, va apare eroare.
	CP	+50	
1756 OPEN-1	JR	NZ,1725,REPORT-0	
	CALL	175D,OPEN-2	Se adună în DE informațiile
	LD	(HL),E	corespunzătoare.
	INC	HL	Informațiile se introduc în
	LD	(HL),D	doi octeți în zona de
	RET		informații a canalului.
			In final, revenire.

THE 'OPEN-2' SUBROUTINE

Octeții corespunzători ai sirului de informații pentru canalul asociat cu sirul care a fost deschis sînt găsiți.

175D OPEN-2	PUSH	HL	Salvare HL.
	CALL	2BF1,STK-FETCH	Se aduc parametrii din
	LD	A,B	canalul de cod.
	OR	C	Va apare eroare dacă expresia
	JR	NZ,1767,OPEN-3	înlocuită este una invalidă;
			aceasta este OPEN #5, ''''.

Prezentarea F - Nume fisier invalid

1765 REPORT-F	RST	0008,ERROR-1	Se apelează rutina de tratare
	DEFB	+0E	eroare.

Se continuă dacă nu a intervenit nici o eroare.

1767 OPEN-3	PUSH	BC	Salvare lungime expresie.
	LD	A,(DE)	Se aduce primul caracter.
	AND	+DF	Se face conversia codurilor
			pentru literele mici în
			codurile literelor mari.
	LD	C,A	Se mută codul în registrul C.
	LD	HL,+177A	Adresa de bază a tabelului
			'trecere prin sir deschis'.
	CALL	16DC,INDEXER	Se indexează în acest tabel
			și se localizează deplasamentul
			cerut.
	JR	NC,1765,REPORT-F	Salt înapoi dacă nu a fost
			găsit.
	LD	C,(HL)	Se trece deplasamentul în
	LD	B,+00	registrul pereche BC.
	ADD	HL,BC	HL va indica începutul
			subrutinei corespunzătoare.
	POP	BC	Se aduce lungimea expresiei
	JP	(HL)	înainte de a efectua saltul
			din subrutină.

THE 'OPEN STREAM LOOK-UP' TABLE (TABELUL 'TRECERE PRIN SIR DESCHIS')

177A	DEFB	4B	06	- canal 'K',	deplasament +06,	adresa 1781
177C	DEFB	53	08	- canal 'S',	deplasament +08,	adresa 1785
177E	DEFB	50	0A	- canal 'P',	deplasament +0A,	adresa 1789
1780	DEFB	00		- marcator de sfîrsit.		

THE 'OPEN-K' SUBROUTINE (SUBRUTINA 'DESCHIDE K')

1781 OPEN-K	LD	E,+01	Octeții de informații vor fi
	JR	1781,OPEN-END	+06 & +00.

THE 'OPEN-S' SUBROUTINE (SUBRUTINA 'DESCHIDE S')

1785 OPEN-S	LD	E,+06	Octeții de informații vor fi
	JR	178B,OPEN-END	+06 & +00.

THE 'OPEN-P' SUBROUTINE (SUBROUTINA 'DESCHIDE P')

1789 OPEN-P	LD	E,+10	Octetii de informatii vor fi +10 & +00.
1788 OPEN-END	DEC	BC	Se decrementează lungimea expresiei si va apare eroare dacă nu a fost un singur caracter; altfel se sterge registrul D, se reface HL si se execută revenire.
	LD	A,B	
	OR	C	
	JR	NZ,1765,REPORT-F	
	LD	D,A	
	POP	HL	
	RET		

THE 'CAT, ERASE, FORMAT & MOVE' COMMAND ROUTINES (RUTINELE DE COMANDA 'CAT' (ridicare), ERASE (stergere), FORMAT (formatare), MOVE (mutare))
In sistemul standard SPECTRUM utilizarea acestor comenzi duce la executia prezentării D - Sir invalid.

1793 CAT-ETC	JR	1725,REPORT-D	Se dă această prezentare.
--------------	----	---------------	---------------------------

THE 'LIST & LLIST' COMMAND ROUTINES (RUTINELE DE COMANDA 'LIST & LLIST')
Rutinele din această parte de 16 K program sînt folosite pentru executarea listării programului programului BASIC curent. Fiecare linie trebuie să aibe numărul propriu de linie evaluat, dezvoltarea semnelor ei si cursorul corespunzător poziționate.

Punctul de intrare AUTO-LIST este utilizat atît de rutina 'MAIN EXECUTION' (executie principală) cît si de editor, pentru executarea unei singure pagini de listing.

1795 AUTO-LIST	LD	(LIST-SP),SP	Indicatorul de stivă este salvat pentru a permite stivei calculatorului să fie resetată cînd se termină listarea (vezi PD-SCR, DC55).
	LD	(TV-FLAG),+10	Se resetează cînd se termină listarea (vezi PD-SCR, DC55).
	CALL	ODAF,CL-ALL	Semnal 'listare automată în partea principală a ecranului'.
	SET	0,(FLAGS)	Se sterge această parte a ecranului.
	LD	B,(DF-SZ)	Schimbare în zona de editare.
	CALL	OE44,CL-LINE	Acum se sterge partea inferioară a ecranului.
	RES	0,(FLAGS)	Se schimbă la loc.
	SET	0,(FLAGS2)	Semnal 'ecran curat'.
	LD	HL,(E-PPC)	Acum se aduce numărul liniei 'curente' si numărul liniei 'automatic'.
	LD	DE,(S-TOP)	Dacă numărul 'curent' este mai mic decît numărul 'automatic' atunci se execută salt înainte pentru mărirea numărului 'automatic'.
	AND	A	
	SBC	HL,DE	
	ADD	HL,DE	
	JR	C,17E1,AUTO-L-2	

Numărul 'automatic' trebuie modificat pentru a da o listare cu linia 'curentă' aparînd lîngă linia cea mai de jos a ecranului.

PUSH	DE	Se salvează numărul 'automatic'.
CALL	196E,LINE-ADDR	Se caută adresa inceputului liniei 'curente'.
LD	DE,+02C0	
EX	DE,HL	
SBC	HL,DE	Se salvează rezultatul în stiva calculatorului pînă cînd adresa liniei 'automatic' va fi si ea găsită.
EX	(SP),HL	
CALL	196E,LINE-ADDR	Rezultatul se trece în registrul pereche BC.
POP	BC	

Acum se introduce o buclă. Numărul liniei 'automatic' este incrementat la fiecare trecere pînă cînd este potrivit cu cel al liniei 'curente' ce va apărea la listare.

17CE AUTO-L-1	PUSH	BC	Se salvează 'rezultatul'.
	CALL	19B8,NEXT-ONE	Se caută adresa inceputului liniei după linia 'automatic' prezentă (în DE).
	POP	BC	Se reface 'rezultatul'.
	ADD	HL,BC	Se execută calculul si se execută salt înainte cînd este gata.
	JR	C,17E4,AUTO-L-3	

EX DE, HL
LD B, (HL)
INC HL
LD E, (HL)
DEC HL
LD (S-TOP), DE
JR 17CE, AUTO-L-1

Se mută adresa următoarei linii în registrul pereche HL și se adună numărul ei de linie.

Acum S-TOP se poate mări și se repetă testul cu noua linie.

Acum se poate executa listarea 'automatic'.

17E1 AUTO-L-2 LD (S-TOP), HL
17E4 AUTO-L-3 LD HL, (S-TOP)
CALL 196E, LINE-ADDR
JR Z, 17ED, AUTO-L-4
EX DE, HL
17ED AUTO-L-4 CALL 1833, LIST-ALL
RES 4, (TV-FLAG)
RET

Cînd E-PPC este mai mic decît S-TOP.

Se aduce numărul liniei din vîrf și deci adresa ei. Dacă linia nu poate fi găsită se folosește imediat DE. S-a executat listarea. Revenirea se va face aici dacă nu este necesară defilarea pentru a arăta linia curentă.

THE 'LLIST' ENTRY POINT (PUNCTUL DE INTRARE 'LLIST')
Trebuie deschis canalul imprimantei.

17F5 LLIST LD A, +03
JR 17FB, LIST-1

Se folosește sirul +03.
Salt înainte.

THE 'LIST' ENTRY POINT (PUNCTUL DE INTRARE 'LIST')
Trebuie deschis canalul 'ecranului principal'.

17F9 LIST LD A, +02
17FB LIST-1 LD (TV-FLAG), +00
CALL 2530, SYNTAX-Z
CALL NZ, 1601, CHAN-OPEN
RST 0018, GET-CHAR
CALL 2070, STR-ALTER
JR C, 181F, LIST-4
RST 0018, GET-CHAR
CP +3B
JR Z, 1814, LIST-2
CP +2C
JR NZ, 181A, LIST-3
1814 LIST-2 RST 0020, NEXT-CHAR
CALL 1C82, EXPT-1NUM
JR 1822, LIST-5
181A LIST-3 CALL 1CE6, USE-ZERO
JR 1822, LIST-5

Se folosește sirul +02.
Se semnalează o listare obisnuită în partea de jos a ecranului.
Se deschide canalul dacă nu este verificată sintaxa.
Cu prezentul caracter în registrul A se verifică dacă sirul este schimbat.
Salt înainte dacă nu s-a schimbat.
Este caracterul prezent ';' ?

Dacă este, salt.
Este ' '?
Dacă nu este, salt.
Urmează o expresie numerică, de exemplu LIST #5.
Salt înainte cu acesta.
Altfel se folosește zero și salt înainte.

Se ajunge în acest punct dacă sirul a fost neschimbat.

181F LIST-4 CALL 1CDE, FETCH-NUM
1822 LIST-5 CALL 1BEE, CHECK-END
CALL 1E99, FIND-INT
LD A, B
AND +3F
LD H, A
LD L, C
LD (E-PPC), HL
CALL 196E, LINE-ADDR
LD E, +01

Se aduce orice linie sau se folosește zero dacă nu sînt înlocuiri.
Dacă s-a verificat sintaxa liniei de editare se face o mutare la următoarea instrucțiune.
Numărul liniei în BC.
Octetul superior în A.
Se limitează octetul superior la ordinul corect și se trece în regulă număr al liniei în HL.
Se setează E-PPC și se găsește adresa de început a acestei linii sau prima linie după ea actuala linie nu există.
Fanion 'înaintea liniei curente'.

Acum este introdusă bucla pentru controlul tipăririi unei serii de linii.

1835 LIST-ALL	CALL	1855,OUT-LINE	Se tipăreste întreaga linie BASIC.
	RST	0010,PRINT-A-1	Acesta va fi un 'carriage return'.
	BIT	4, (TV-FLAG)	Salt înapoi dacă nu s-a lucrat cu listare automată.
	JR	Z,1835,LIST-ALL	De asemenea salt înapoi dacă mai există o parte din ecranul principal care poate fi folosită.
	LD	A, (DF-SZ)	In acest punct se poate face o revenire dacă ecranul este plin și linia curentă a fost tipărită (E = +00).
	SUB	(S-POSN-hi)	Oricum, dacă linia curentă lipsește din listare, atunci S-TOP trebuie mărit și o linie următoare trebuie tipărită (folosind defilarea)
	JR	NZ,1835,LIST-ALL	
	XOR	E	
	RET	Z	
	PUSH	HL	
	PUSH	DE	
	LD	HL, +5C6C	
	CALL	190F, LN-FETCH	
	POP	DE	
	POP	HL	
	JR	1835, LIST-ALL	

THE 'PRINT A WHOLE BASIC LINE' SUBROUTINE (SUBROUTINA 'TIPARIRE A UNEI INTREGI LINII BASIC')

Registrul pereche HL indică începutul liniei - locația care conține octetul superior al numărului liniei. Înainte de a se tipări numărul liniei el este testat pentru a se determina dacă vine înaintea liniei 'curente', este linia 'curentă' sau vine după linia 'curentă'.

1855 OUT-LINE	LD	BC, (E-PPC)	Se aduce numărul liniei 'curente' și se compară.
	CALL	1980, CP-LINE	Se încarcă registrul D cu cursorul liniei curente.
	LD	D, +3E	Salt înainte dacă se tipăreste linia 'curentă'.
	JR	Z, 1865, OUT-LINE1	Se încarcă registrul D cu zero (nu este cursorul) și se setează registrul E încât să continue +01 dacă linia este înainte de linia 'curentă' și +00 dacă este după ea.
	LD	DE, +0000	(Fanionul de transport vine din CP-LINES).
	RL	E	Salvare marcator de linie.
			Se aduce octetul superior al numărului liniei și se face o revenire completă dacă s-a terminat listarea.
1865 OUT-LINE	LD	(BREG), E	Acum se poate lista numărul liniei - cu spații de fond.
	LD	A, (HL)	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	CP	+40	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	POP	BC	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	RET	NC	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	PUSH	BC	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	CALL	1A28, OUT-NUM-2	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	INC	HL	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	INC	HL	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	INC	HL	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	RES	0, (FLAGS)	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	LD	A, D	Se aduce codul cursorului și salt înainte dacă cursorul nu trebuie listat.
	AND	A	Acum se tipăreste cursorul.
	JR	Z, 1881, OUT-LINE3	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
	RST	0010, PRINT-A-1	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
1870 OUT-LINE2	SET	0, (FLAGS)	Se mută indicatorul ca să adreseze primul cod de comandă în linie.
1881 OUT-LINE3	PUSH	DE	Se salvează registrul DE.
	EX	DE, HL	Se mută indicatorul în DE.
	RES	2, (FLAGS2)	Se mută indicatorul în DE.
	LD	HL, +5C3B	Se mută indicatorul în DE.
	RES	2, (HL)	Se mută indicatorul în DE.
	BIT	5, (FLAGX)	Se mută indicatorul în DE.
	JR	Z, 1894, OUT-LINE4	Se mută indicatorul în DE.
	SET	2, (HL)	Se mută indicatorul în DE.

Acum este introdusă o buclă care să tipărească toate codurile din restul liniei BASIC - sărind peste formele punctului flotant după caz.

1894 OUT-LINE4	LD	HL, (X-PTR)	Se aduce indicatorul erorii de sintaxă și salt înainte dacă nu este momentul pentru tipărirea marcatorului de
	AND	A	
	SBC	HL, DE	
	JR	NZ, 18A1, OUT-LINES	

	LD	A,+3F	eroare.
18A1 OUT-LINES	CALL	18C1,OUT-FLASH	Acum se tipăreste marcatorul de eroare.
	CALL	18E1,OUT-CURS	Pilpîie '?'. Se consideră dacă este timpul să se tipărească cursorul.
	EX	DE,HL	Acum se mută indicatorul în HL.
	LD	A,(HL)	Se aduce fiecare caracter pe rînd.
	CALL	18B6,NUMBER	Dacă caracterul este un 'număr marcator' atunci forma ascunsă a formei punctului flotant nu trebuie tipărită.
	INC	HL	Se incrementează indicatorul pentru următoarea trecere.
	CP	+0D	Este caracterul un 'carriage return'?
	JR	Z,18B4,OUT-LINE6	Salt dacă este.
	EX	DE,HL	Schimbare indicator în DE.
	CALL	1937,OUT-CHAR	Se tipăreste caracterul.
JR	1894,OUT-LINE4	Ciclare buclă pînă la sfîrsit cu cîte o nouă trecere.	

Acum linia a fost tipărită.

18B4 OUT-LINE6	POP	DE	Se reface registrul pereche
	RET		DE si revenire.

THE 'NUMBER' SUBROUTINE (SUBRUTINA 'NUMAR')

Dacă registrul A contine 'număr marcator' atunci registrul pereche HL este avansat peste forma punctului flotant.

18B6 NUMBER	CP	+0E	Este caracterul un 'număr marcator'? Dacă nu - revenire
	RET	NZ	Se avansează indicatorul de sase ori astfel încît să treacă peste 'numărul marcator' si cele cinci locatii să contină forma punctului flotant.
	INC	HL	
	INC	HL	
	INC	HL	
	INC	HL	
	INC	HL	
	INC	HL	
	LD	A,(HL)	Se aduce codul curent înainte de revenire.
	RET		

THE 'PRINT A FEATCHING CHARACTER' SUBROUTINE (SUBRUTINA 'TIPARIREA UNUI CARACTER PILPIITOR')

'Eroarea cursor' si 'modul cursor' se tipăresc folosind această subrutină.

18C1 OUT-FLASH	EXX		Salvare registrii curenti.
	LD	HL,(ATTR-T)	Se salvează ATTR-T & MASK-T în stiva calculatorului.
	PUSH	HL	
	RES	7,H	Se asigură că FLASH este activ.
	SET	7,L	
	LD	(ATTR-T),HL	Se folosesc aceste valori modificate ale lui ATTR-T & MASK-T.
	LD	HL,+5C91	Acesta este P-FLAG.
	LD	D,(HL)	Si P-FLAG se salvează în stiva calculatorului.
	PUSH	DE	Se asigură că este INVERSE 0, OVER 0 si nu PAPER 9, nici INK 9.
	LD	(HL),+00	Caracterul este tipărit.
	CALL	09F4,PRINT-OUT	Se reface valoarea anterioară a lui P-FLAG.
	POP	HL	De asemenea se refac vechile valori ale lui ATTR-T & MASK-T înainte de revenire.
	LD	(P-FLAG),H	
	POP	HL	
	LD	(ATTR-T),HL	
	EXX		
	RET		

THE 'PRINT THE CURSOR' SUBROUTINE (SUBRUTINA 'TIPARIRE CURSOR')

Se face o revenire dacă nu este locul potrivit pentru tipărirea cursorului, dar dacă este, atunci se va tipări unul dintre 'C', 'E', 'B', 'K' sau 'L'.

18E1 OUT-CURS	LD	HL,(K-CUR)	Se aduce adresa cursorului dar se execută revenire dacă nu s-a considerat locul corect.
	AND	A	
	SBC	HL,DE	Se aduce valoarea curentă a
	RET	NZ	
	LD	A,(MODE)	

	RLC	A	lui MODE si se dublează.
	JR	Z,18F3,OUT-C-1	Salt înainte dacă nu se
	ADD	A,+43	lucrează cu modul Extins sau
	JR	1909,OUT-C-2	Grafic.
18F3 OUT-C-1	LD	HL,+5C3E	Salt înainte pentru a-l
	RES	3,(HL)	tipări.
	LD	A,+4B	Acesta este FLAGS.
	BIT	2,(HL)	Semnal 'mod K'.
	JR	Z,1909,OUT-C-2	Caracterul 'K'.
	SET	3,(HL)	Salt înainte pentru a tipări
	INC	A	'K' dacă 'tipărirea se va
	BIT	3,(FLAGS2)	face în mod K'.
	JR	Z,1909,OUT-C-2	'Tipărirea se va face în mod
	LD	A,+43	L', așa că semnalizare 'mod
1909 OUT-C-2	PUSH	DE	L'.
	CALL	18C1,OUT-FLASH	Formare caracter 'L'.
	POP	DE	Salt înainte dacă nu este în
	RET		'mod C'.
			Caracterul 'C'.
			Se salvează registrul pereche
			DE pînă cînd cursorul este
			tipărit - pîlpînd.
			Revenire odată ce s-a făcut.

Notă: Aceasta este acțiunea de considerare a cursorului - literă de tipărit care determină modul 'K' sau 'L/C'.

THE 'LN-FETCH' SUBROUTINE (SUBROUTINA 'LN-FETCH')

Această subrutină este introdusă cu registrul pereche HL adresînd o variabilă sistem - S-TOP sau E-PPC.

Subrutina revine cu variabila sistem conținînd numărul de linie al liniei următoare.

190F LN-FETCH	LD	E,(HL)	Se colectează numărul de
	INC	HL	linie conținut de variabila
	LD	D,(HL)	sistem.
	PUSH	HL	Indicatorul este salvat.
	EX	DE,HL	Numărul de linie este mutat
	INC	HL	în registrul pereche HL și
	CALL	196E,LINE-ADDR	incrementat.
	CALL	1695,LINE-NO	S-a găsit adresa de început a
	POP	HL	acestei linii, sau a liniei
			următoare, dacă actualul
			număr de linie nu a fost
			folosit.
			Acel număr de linie este adus
			Se reface indicatorul
			variabilei sistem.

Punctul de intrare LN-STORE este folosit de EDITOR.

191C LN-STORE	BIT	5,(FLAGX)	Revenire dacă este 'mod
	RET	NZ	INPUT'; altfel se procedează
	LD	(HL),D	la introducerea numărului de
	DEC	HL	linie în două locații ale
	LD	(HL),E	variabilei sistem.
	RET		Revenire.

THE 'PRINTING CHARACTERS IN A BASIC LINE' SUBROUTINE (SUBROUTINA 'TIPĂRIRE CARACTERE INTR-O LINIE BASIC')

Toate codurile de caractere/semne dintr-o linie BASIC sînt tipărite prin apelări repetate ale acestei subrutine.

Punctul de intrare OUT-SP-NO este utilizat cînd se tipăresc numere de linii care cer spații de fond.

1925 OUT-SP-2	LD	A,E	Registrul A va conține +20
	AND	A	pentru un spațiu și +FF
	RET	M	pentru nici un spațiu.
	JR	1937,OUT-CHAR	Se testează valoarea și se
			revine dacă nu este spațiu.
			Salt înainte pentru tipărire
			spațiu.
192A OUT-SP-NO	XOR	A	Se șterge registrul A.

Registrul pereche HL conține numărul liniei și registrul BC conține valoarea pentru 'scădere repetată'. (BC conține '-1000, -100 sau -10'.)

192B OUT-SP-1	ADD	HL,BC	'Incertare scădere'.
---------------	-----	-------	----------------------

INC	A	Se numără fiecare 'încercare'
JR	C, 192B, OUT-SP-1	Salt înapoi pînă la epuizare.
SBC	HL, BC	Se reface ultima 'scădere' și
DEC	A	se decrementează.
JR	Z, 1925, OUT-SP-2	Dacă nu este posibilă nici o
		'scădere', salt înapoi pentru
		a vedea dacă nu trebuie
		tipărit un spațiu.
JP	15EF, OUT-CODE	În caz contrar, tipărire
		digit.

Punctul de intrare OUT-CHAR este folosit pentru toate caracterele, semnele și caracterele de control.

1937 OUT-CHAR	CALL	2D1B, NUMERIC	Readucerea transportului
			resetat dacă se tratează cod
			digit.
	JR	NC, 196C, OUT-CH-3	Salt înainte pentru tipărire
			digit.
	CP	+21	De asemenea se tipăresc
	JR	C, 196C, OUT-CH-3	caracterele de control și
			'spațiu'.
	RES	2, (FLAGS)	Semnal 'tiărare în mod K'.
	CP	+CB	Salt înainte dacă se lucrează
	JR	Z, 196C, OUT-CH-3	cu simbolul 'THEN'.
	CP	+3A	Salt înainte dacă nu se
	JR	NZ, 195A, OUT-CH-1	lucrează cu ':'.
	BIT	5, (FLAGX)	Salt înainte pentru a tipări
	JR	NZ, 1968, OUT-CH-2	';' dacă este 'mod INPUT'.
	BIT	2, (FLAGS2)	Salt înainte dacă ':' nu
	JR	Z, 196C, OUT-CH-3	este în limite', adică un
			marcator inter-instrucțiuni.
	JR	1968, OUT-CH-2	Semnul ';' este între limite
			și acum poate fi tipărit.
195A OUT-CH-1	CP	+22	Se acceptă la tipărire toate
	JR	NZ, 1968, OUT-CH-2	caracterele, mai puțin '...'
	PUSH	AF	Se salvează codul
			caracterului pînă cînd se
			schimbă 'mod limite'.
	LD	A, (FLAGS2)	Se aduce FLAG2 și se schimbă
	XDR	+04	bitul 2.
	LD	(FLAGS2), A	Se introduce valoarea
	POP	AF	modificată și se reface codul
			caracterului.
1968 OUT-CH-2	SET	2, (FLAGS)	Semnal 'următorul caracter
			trebuie tipărit în mod L'.
196D OUT-CH-3	RST	0010, PRINT-A-1	Prezentul caracter este
	RET		tipărit înainte de revenire.

Notă: Este o consecință a testului asupra caracterului prezent care determină dacă următorul caracter 'se va tipări în mod 'K' sau 'L'.

De asemenea, de notat cum programul nu furnizează ':' în instrucțiunea REM.

THE 'LINE-ADDR' SUBROUTINE (SUBROUTINA 'LINE-ADDR')

Pentru a da un număr de linie, în registrul pereche HL, această subrutină returnează începutul adresei acelei linii sau 'prima linie de după' și începutului liniei anterioare în registrul pereche DE.

Fanionul zero este setat dacă numărul liniei a fost folosit. Oricum, dacă 'prima linie de după' este substituită atunci fanionul zero este returnat resetat.

196E LINE-ADDR	PUSH	HL	Se salvează numărul liniei
			dat.
	LD	HL, (PROG)	Se aduce variabila sistem
	LD	D, H	PROG și se transferă adresa
	LD	E, L	în registrul pereche DE.

Acum se introduce o buclă pentru a testa numărul liniei al fiecărei linii a programului cu numărul liniei date pînă cînd numărul liniei este egal sau depășit.

1974 LINE-AD-1	POP	BC	Numărul liniei dat.
	CALL	1980, CP-LINES	Se compară numărul de linie
	RET	NC	dat cu numărul liniei
			adresate.
	PUSH	BC	Revenire dacă transportul
	CALL	1988, NEXT-ONE	este resetat; altfel se
			adresează numărul liniei
			următoare.

EX	DE, HL	Se schimbă indicatorii și
JR	1974, LINE-AD-1	salt înapoi pentru a lua în
		considerare următoarea linie
		a programului.

THE 'COMPARE LINE NUMBERS' SUBROUTINE (SUBROUTINA 'COMPARARE NUMERE LINIE')
Numărul de linie dat în registrul pereche BC este comparat cu numărul liniei adresate.

1980 CP-LINES	LD	A, (HL)	Se aduce octetul superior al
	CP	B	numărului de linie adresat și
	RET	NZ	se compară. Salt dacă nu
			coincide.
	INC	HL	Apoi se compară octetii
			inferiori.
	LD	A, (HL)	Revenire cu fanionul de
	DEC	HL	transport setat dacă numărul
	CP	C	liniei adresate trebuie încă
	RET		să ajungă la numărul liniei
			date.

THE 'FIND EACH STATEMENT' SUBROUTINE (SUBROUTINA 'GASIREA FIECAREI INSTRUCȚII')
Această subrutină are două funcții distincte.

i. Poate fi folosită pentru a afla 'D'-a instrucțiune dintr-o linie BASIC - revenind cu registrul pereche HL adresând locația dinaintea de începutul instrucției și cu fanionul zero setat.

ii. De asemenea, subrutina se poate folosi pentru a afla o instrucțiune, dacă există, care începe cu codul unui simbol (în registrul E).

	INC	HL	Nu este utilizat.
	INC	HL	
	INC	HL	
EACH-STMT	LD	(CH-ADD), HL	Se setează CH-ADD pe octetul
			curent.
	LD	C, +00	Setare fanion 'limite off'.

Se introduce o buclă pentru tratarea fiecărei instrucțiuni dintr-o linie BASIC.

1990 EACH-S-1	DEC	D	Se decrementează 'D' și se
	RET	Z	revine dacă instrucțiunea
			cerută nu a fost găsită.
	RST	0020, NEXT-CHAR	Se aduce codul următorului
	CP	E	caracter și salt dacă nu
	JR	NZ, 199A, EACH-S-3	coincide cu codul simbolului
			dat.
	AND	A	Dacă coincide, atunci se
	RET		revine cu fanioanele zero și
			de transport resetate.

Acum se introduce o altă buclă pentru a considera fiecare caracter din linie, pentru a determina unde se termină instrucțiunea.

1998 EACH-S-2	INC	HL	Se mărește indicatorul și se
	LD	A, (HL)	aduce noul cod.
199A EACH-S-3	CALL	1986, NUMBER	Salt peste orice număr.
	LD	(CH-ADD), HL	Se mărește CH-ADD.
	CP	+22	Salt înainte în cazul în care
	JR	NZ, 19A5, EACH-S-4	caracterul nu este " ".
	DEC	C	În caz contrar se setează
			'fanionul limite'.
19A5 EACH-S-4	CP	+3A	Salt înainte în caz în care
	JR	Z, 19AD, EACH-S-5	caracterul este ' ; ' -
	CP	+CB	Salt înainte dacă nu este
	JR	NZ, 19B1, EACH-S-6	simbolul 'THEN'.
19AD EACH-S-5	BIT	0, C	Se citește 'fanionul cote' și
	JR	Z, 1990, EACH-S-1	salt înapoi la sfârșitul
			fiecărei instrucții (începând
			după 'THEN').
19B1 EACH-S-6	CP	+0B	Salt înapoi dacă nu este
	JR	NZ, 1998, EACH-S-2	sfârșitul unei linii BASIC.
	DEC	D	Se decrementează contorul de
	SCF		instrucțiuni și se setează
	RET		fanionul de transport înainte
			de a se reveni.

THE 'NEXT-ONE' SUBROUTINE (SUBROUTINA 'URMATOAREA')

Această subrutină se poate folosi pentru a afla 'linia următoare' în spațiul programului sau 'variabila următoare' în spațiul variabilelor. Subrutina este furnizorul pentru șase tipuri diferite de variabile care sînt utilizate în sistemul SPECTRUM.

19B8 NEXT-ONE	PUSH HL	Se salvează adresa liniei curente sau a variabilei.
	LD A, (HL)	Se aduce primul octet.
	CP +40	Salt înainte dacă se caută 'linia următoare'.
	JR C, 19D5, NEXT-0-3	Salt înainte dacă se caută următorul sir.
	BIT 5, A	Salt înainte cu un numeric simplu și variabilele FOR-NEXT.
	JR Z, 19D6, NEXT-0-4	Numai pentru variabile numerice cu nume lung.
	ADD A, A	O variabilă numerică va ocupa cinci locații dar variabila de control FOR-NEXT va avea nevoie de nouă locații.
	JP M, 19C7, NEXT-0-1	Fanionul de transport va fi resetat doar pentru variabilele cu nume lung; pînă se găsește caracterul final al numelui lung.
	CCF	Se incrementează indicatorul și se aduce noul cod.
19C7 NEXT-0-1	LD BC, +0005	Salt înapoi dacă codul anterior nu a fost ultimul cod al numelui variabilei.
	JR NC, 19CE, NEXT-0-2	Acum salt înainte (BC = +0005 sau +0012).
	LD C, +12	Se trece peste octetul inferior al numărului liniei. Acum se indică octetul inferior al lungimii.
19CE NEXT-0-2	RLA	Se aduce lungimea în registrul pereche BC.
	INC HL	Se dă permisiunea octetului cuprinzător.
	LD A, (HL)	
	JR NC, 19CE, NEXT-0-2	
	JR 19DB, NEXT-0-5	
19D5 NEXT-0-3	INC HL	
19D6 NEXT-0-4	INC HL	
	LD C, (HL)	
	INC HL	
	LD B, (HL)	
	INC HL	

În toate cazurile s-a găsit adresa liniei 'următoare' sau a variabilei.

19DB NEXT-0-5	ADD HL, BC	Se indică primul octet al liniei 'următoare' sau al variabilei.
	POP DE	Se aduce adresa celei anterioare și se continuă în subrutina de 'diferențiere'.

THE 'DIFFERENCE' SUBROUTINE (SUBROUTINA 'DIFERENȚIERE')

'Lungimea' dintre două 'începuturi' este formată în registrul pereche BC. Indicatorii sînt refăcuți dar ei sînt returnați schimbați.

19DD DIFFER	AND A	Pregătire pentru scădere.
	SBC HL, DE	Se găsește lungimea de la un 'început' la următorul și ea se trece în registrul pereche BC.
	LD B, H	
	LD C, L	
	ADD HL, DE	Se reformează adresa și se schimbă înainte de revenire.
	EX DE, HL	
	RET	

THE 'RECLAIMING' SUBROUTINE (SUBROUTINA 'REFACERE')

Punctul de intrare RECLAIM-1 este utilizat cînd adresa primei locații ce trebuie refăcută este în registrul pereche DE și adresa primei locații lăsată izolată este în registrul pereche HL. Punctul de intrare RECLAIM-2 este utilizat cînd registrul pereche indică prima locație ce trebuie refăcută și registrul pereche BC conține numărul octetilor care trebuie refăcuți.

19E5 RECLAIM-1	CALL 19DD, DIFFER	Se folosește rutina de 'diferențiere' pentru a realiza valoarea corespunzătoare.
19E8 RECLAIM-2	PUSH BC	Se salvează numărul octetilor care trebuie refăcuți.
	LD A, B	Indicatorii tuturor

CPL		variabilelor sistem deasupra
LD	B,A	zonei trebuie redusi cu 'BC'
LD	A,C	asa ca acest numar este
CPL		complementat de 2 ori inainte
LD	C,A	ca indicatorii sa fie
INC	BC	schimbati.
CALL	1664,POINTERS	
EX	DE,HL	Adresa 'primei locatii'
POP	HL	revine in registrul pereche
ADD	HL,DE	DE si se reformeaza adresa
		primei locatii izolate.
PUSH	DE	Se salveaza 'prima locatie'
LDIR		pina cand intervine actuala
POP	HL	refacere.
RET		Acum se revine.

THE 'E-LINE-NO' SUBROUTINE (SUBROUTINA 'E-LINE-NO')

Aceasta subrutina este folosita pentru a citi numarul de linie al liniei din zona de editare. Daca nu este nici un numar de linie, adica o linie Basic directa, atunci se considera ca numarul liniei este zero.

In toate cazurile, numarul liniei este returnat in registrul pereche BC.

19FB E-LINE-NO	LD	HL,(E-LINE)	Punctarea indicatorilor pe linia de editare.
	DEC	HL	Se seteaza CH-ADD sa indice
	LD	(CH-ADD),HL	locatia dinaintea oricarui
	RST	0020,NEXT-CHAR	numar.
	LD	HL,+5C92	Se trece primul cod in
	LD	(STKEND),HL	registrul A.
	CALL	2D3B,INT-TO-FP	Inainte de a considera codul
			se realizeaza in zona
			memoriei calculatorului o
			zona de stiva temporara.
			Acum se citesc digitii
			numarului liniei. Revenire
			zero daca nu este nici un
			numar.
	CALL	2DA2,FP-TO-BC	Se comprima numarul liniei in
			registrul pereche BC.
	JR	C,1A15,E-L-1	Salt inainte daca numarul
			depaseste '65,536'.
	LD	HL,+DBF0	Altfel se testeaza fata de
	ADD	HL,BC	'10,000'.
	JP	C,1C8A,REPORT-C	Se da prezentarea C daca este
			peste '9,999'.
	JP	16C5,SET-STK	Revenire prin SET-STK, care
			reface stiva calculatorului
			in locul cuvenit.

THE 'REPORT AND LINE NUMBER PRINTING' SUBROUTINE (SUBROUTINA 'TIPARIRE PREZENTARE SI NUMAR DE LINIE')

Punctul de intrare OUT-NUM-1 va conduce ca numarul din registrul pereche BC sa fie tiparit. Orice valoare peste '9,999' nu va fi tiparita corect.

Punctul de intrare OUT-NUM-2 va conduce ca numarul indirect adresat de registrul pereche HL sa fie tiparit. In acest moment orice spatiu de fond este necesar. Si aici numarul limita pentru tiparirea corecta este '9,999'.

1A1B OUT-NUM-1	PUSH	DE	Se salveaza registrul
	PUSH	HL	implicati in subrutina.
	XOR	A	Se sterge registrul A.
	BIT	7,B	Salt inainte pentru a tipari
	JR	NZ,1A42,OUT-NUM-4	un zero mai degrabă decât un
			'-2' când se prezintă într-o
			linie de editare.
	LD	H,B	Se muta numarul in registrul
	LD	L,C	pereche HL.
	LD	E,+FF	Fanion 'nici un spatiu de
			fond'.
	JR	1A30,OUT-NUM-3	Salt inainte pentru a tipari
			numarul.
1A2B OUT-NUM-2	PUSH	DE	Se salveaza registrul pereche
			DE.
	LD	D,(HL)	Se aduce numarul in registrul
	INC	HL	pereche DE si se salveaza
	LD	E,(HL)	indicatorul (marit).
	PUSH	HL	
	EX	DE,HL	
	LD	E,+20	Se muta numarul in registrul
			pereche HL si se semnaleaza

1A30 OUT-NUM-3	LD BC,+FC18 CALL 192A,OUT-SP-NO LD BC,+FF9C CALL 192A,OUT-SP-NO LD C,+F5 CALL 192A,OUT-SP-NO LD A,L	'spatiile de fond trebuie tipărite'. Acesta este '-1.000'. Se tipăreste primul digit. Acesta este '-100'. Se tipăreste al doilea digit. Acesta este '-10'. Se tipăreste al treilea digit Mutarea oricărei părți rămase a numărului în registrul A. Se tipăreste digitul. Se refac registrii înainte de revenire.
1A42 OUT-NUM-4	CALL 15EF,OUT-CODE POP HL POP DE RET	

BASIC LINE AND COMMAND INTERPRETATION (INTERPRETAREA LINIEI BASIC SI A COMENZILOR)

THE SYNTAX TABLES (TABELELE DE SINTAXA)

i. The offset table (tabelul de deplasament)

Pentru fiecare 50 de comenzi BASIC există o valoare a deplasamentului.

	comanda	adresa		comanda	adresa
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 1B0A	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 1AEB	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 1AFO	1A71	DEFB +3A	RUN 1AAB
1A59	DEFB +98	OUT 1AF1	1A72	DEFB +6D	SAVE 1ADF
1A5A	DEFB +7F	LPRINT 1AD9	1A73	DEFB +42	RANDOMIZE 1AB5
1A5B	DEFB +81	LLIST 1ADC	1A74	DEFB +0D	IF 1A81
1A5C	DEFB +2E	STOP 1ABA	1A75	DEFB +49	CLS 1ABE
1A5D	DEFB +6C	READ 1AC9	1A76	DEFB +5C	DRAW 1AD2
1A5E	DEFB +6E	DATA 1ACC	1A77	DEFB +44	CLEAR 1ABD
1A5F	DEFB +70	RESTORE 1ACF	1A78	DEFB +15	RETURN 1A8D
1A60	DEFB +48	NEW 1AA8	1A79	DEFB +5D	COPY 1AD6

ii. The parameter table (Tabelul parametru)

Pentru fiecare 50 de comenzi BASIC există pînă la opt intrări în tabelul parametru. Aceste intrări cuprind detaliile clasei de comenzi, solicită separatori și adresele rutinelor comenzilor.

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
		DEFB +06	CLASS-06
		DEFB +CB	'THEN'
		DEFB +05	CLASS-05
		DEFB +F0,+1C	IF,ICF0
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-0
		DEFB +23,+1F	RETURN,1F23
		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

06821

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	+4FF,+1E	RANDOMIZE,1E4F
1AB8 P-CONT	DEFB	+00	CLASS-00
	DEFB	+5F,+1E	CONTINUE,1E5F
1AB3 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-LIST	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
1AE3 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

Notă: Cererile pentru diferitele categorii de comenzi sînt următoarele:

- CLASS-00 - Nu mai sînt operatori în continuare.
 CLASS-01 - Se folosește în LET. Cerere de variabilă.
 CLASS-02 - Se folosește în LET. Trebuie să urmeze o expresie numerică sau un sir.
 CLASS-03 - Trebuie să urmeze o expresie numerică. În caz de eroare se folosește zero.
 CLASS-04 - Trebuie să urmeze un singur caracter variabil.
 CLASS-05 - Trebuie dat un set de numere.
 CLASS-06 - Trebuie să urmeze o expresie numerică.
 CLASS-07 - Se tratează numerele de culori.
 CLASS-08 - Trebuie să urmeze două expresii numerice, separate prin virgulă.
 CLASS-09 - În privința CLASS-08, doar numerele de culori pot preceda expresii.
 CLASS-0A - Trebuie să urmeze o expresie sir.
 CLASS-0B - Se tratează rutinele legate de casetă.

THE 'MAIN PARSER' OF THE BASIC INTERPRETER ('ANALIZA PRINCIPALA' A INTERPRETORULUI BASIC)

Rutina de analizare a interpretorului BASIC intră pe la LINE-SCAN cînd sintaxa a fost verificată, și pe la LINE-RUN cînd se execută un program BASIC care are una sau mai multe instrucții.

Se consideră pe rînd fiecare instrucțiune și se folosește variabila sistem CH-ADD pentru a indica fiecare cod al instrucțiunii ca și cum s-ar afla în spațiul programului sau în spațiul editat.

1B17 LINE-SCAN	RES 7, (FLAGS)	Signal 'verificare sintaxă'.
	CALL 19FB, E-LINE-NO	CH-ADD va indica primul cod după oricare număr de linie.
	XOR A	Variabila sistem SURPPC se
	LD (SUBPPC), A	initializează cu +00 și
	DEC A	ERR-NR cu +FF.
	LD (ERR-NR), A	
	JR 1B29, STMT-L-1	Salt înainte pentru a lua în considerare prima instrucțiune a liniei.

THE STATEMENT LOOP (INSTRUCȚIUNEA DE CICLARE)

Se consideră pe rînd fiecare instrucțiune pînă cînd se atinge sfîrșitul liniei.

1B28 STMT-LOOP	RST 0020, NEXT-CHAR	Avans CH-ADD de-a lungul liniei.
1B29 STMT-L-1	CALL 16BF, SET-WORK	Spațiul de lucru este sters.
	INC (SUBPPC)	Se incrementează SURPPC la fiecare trecere prin buclă.
	JP M, 1C8A, REPORT-C	Sînt permise doar '127' de instrucțiuni într-o singură linie.
	RST 0018, GET-CHAR	Se aduce un caracter.
	LD R, +00	Se șterge registrul pentru mai tîrziu.
	CP +0B	Salt în cazul în care caracterul este un 'carriage return'.
	JR Z, 1B33, LINE-END	Reîntoarcere în buclă dacă este ','.
	CP +3A	
	JR Z, 1B28, STMT-LOOP	

A fost identificată o instrucțiune, așa că, prima dată se consideră comanda sa inițială.

LD HL, +1B76	Se încarcă stiva masinii cu adresa de revenire - STMT-RET
PUSH HL	
LD C, A	Se salvează temporar comanda în registrul C pînă CH-ADD a avansat iar.
RST 0020, NEXT-CHAR	Se reduce codul comenzii cu +CE; se dă ordinul +00 pînă la +31 pentru 50 de comenzi.
LD A, C	Se comunică eroarea corespunzătoare, dacă nu este un cod de comandă.
SUB +CE	Se mută codul comenzii în registrul pereche BC (B conține +00).
JP C, 1B8A, REPORT-C	Adresa de bază a tabelului
LD C, A	
LD HL, +1A48	

ADD	HL,BC	sintaxei deplasamentului. Deplasamentul cerut este trecut în registrul C și se folosește la calculul adresei de bază pentru intrările comenzii în tabelul parametru.
LD	C,(HL)	
ADD	HL,BC	
JR	1B55,GET-PARAM	Salt înainte cu această adresă în bucla de scanare.

Fiecare din rutinele categoriei de comenzi aplicabile prezentei comenzi se execută pe rând. De asemenea, se consideră orice cerere de separatori.

1B52 SCAN-LOOP	LD	HL,(T-ADDR)	Se încarcă în tabelul parametru indicatorul temporar al intrărilor.
1B55 GET-PARAM	LD	A,(HL)	Se încarcă pe rând fiecare intrare.
	INC	HL	Se incrementează indicatorul intrărilor pentru următorul pas.
	LD	(T-ADDR),HL	Se încarcă stiva mașinii cu adresa de revenire SCAN-LOOP.
	LD	BC,+1B52	Se copiază intrarea în registrul C pentru mai târziu.
	PUSH	BC	Salt înainte dacă intrarea este un 'separator'.
	LD	C,A	Se încarcă adresa de bază a tabelului 'clasei de comenzi'.
	CP	+20	Se șterge registrul B și se indexează în tabel.
	JR	NC,1B6F,SEPARATOR	Se aduce deplasamentul și se calculează adresa de început a rutinei clasei de comenzi cerute.
	LD	HL,+1G01	Se încarcă această adresă în stiva calculatorului.
	LD	B,+00	Înainte de a executa un salt indirect la rutina clasei comenzii, se trece codul comenzii în registrul A iar registrul B se încarcă cu +FF.
	ADD	HL,BC	
	LD	C,(HL)	
	ADD	HL,BC	
	PUSH	HL	
	RST	001B,GET-CHAR	
	DEC	B	
	RET		

THE 'SEPARATOR' SUBROUTINE (SUBROUTINA 'SEPARATOR')

Prezentarea - 'Nonsens în BASIC' este dată dacă nu există cererea de separator. De notat că atunci când sintaxa a fost verificată, această prezentare nu apare pe ecran - doar 'error marker' (marcător de eroare).

1B6F SEPARATOR	RST	001B,GET-CHAR	Caracterul curent este adus și comparat cu intrarea în tabelul parametru.
	CP	C	Se comunică eroare dacă nu coincid.
	JP	NZ,1C8A,REPORT-C	Se trece un caracter corect și revenire.
	RST	0020,NEXT-CHAR	
	RET		

THE 'STMT-RET' SUBROUTINE (SUBROUTINA 'STMT-RET')

După interpretarea corectă a instrucțiunii se revine la acest punct de intrare.

1B76 STMT-RET	CALL	1F54,BREAK-KEY	Se apasă tasta BREAK după fiecare instrucțiune.
	JR	C,1B7D,STMT-R-1	Salt înainte dacă nu a fost apăsată.

Prezentarea L - 'BREAK în program'

1B7B REPORT-L	RST	000B,ERROR-1	Se apelează rutina de tratare eroare.
	DEFB	+14	

Aici se continuă dacă tasta BREAK nu a fost apăsată.

1B7D STMT-R-1	BIT	7,(NSPPC)	Salt înainte dacă nu trebuie făcut nici un 'salt'.
	JR	NZ,1B74,STMT-NEXT	Se aduce numărul 'liniei noi' dacă nu se lucrează cu o
	LD	HL,(NEWPPC)	
	BIT	7,H	

JR Z,1B9E,LINE-NEW instructiune nouă în spatiul de editare.

THE 'LINE-RUN' ENTRY POINT (PUNCTUL DE INTRARE 'EXECUTIE LINIEI')

Acest punct de intrare este utilizat întotdeauna când o linie din zona de editare trebuie 'executată'. Într-un asemenea caz fanionul sintaxă/executie (bitul 7 din FLAGS) trebuie setat.

Punctul de intrare este de asemenea utilizat la testarea sintaxei unei linii din zona de editare care are mai mult de o instructiune (bitul 7 din FLAGS va fi resetat).

1B8A LINE-RUN	LD	HL,+FFFE	D linie din zona de editare este considerată ca linia '-2'.
	LD	(PPC),HL	
	LD	HL,(WORKSP)	HL va indica marcatorul de sfîrsit al zonei de editare
	DEC	HL	si DE locatia dinainte de începutul acestei zone.
	LD	DE,(E-LINE)	
	DEC	DE	
	LD	A,(NSPPC)	Se aduce numărul următoarei instructiuni pentru a se trata înainte de a sări înainte.
	JR	1BD1,NEXT-LINE	

THE 'LINE-NEW' SUBROUTINE (SUBROUTINA 'LINIE NOUA')

În program s-a făcut un salt si trebuie găsită adresa de început a noii linii.

1B9E LINE-NEW	CALL	196E,LINE-ADDR	S-a găsit adresa de început a liniei sau a 'primei linii de după'.
	LD	A,(NSPPC)	Se adună numărul instructiei.
	JR	Z,1BBF,LINE-USE	Salt înainte dacă s-a găsit linia cerută; în caz contrar, se verifică validitatea numărului instructiunii - trebuie să fie zero.
	AND	A	Se mai verifică dacă 'prima linie de după' nu este după actualul 'sfîrsit de program'
	JR	NZ,1BEC,REPORT-N	
	LD	B,A	
	LD	A,(HL)	
	AND	+C0	
	LD	A,B	
	JR	Z,1BBF,LINE-USE	Salt înainte cu adresele valide; altfel se semnalizează eroare 'OK'.

Prezentarea 0 - 'OK'

1BB0 REPORT-0	RST	0008,ERROR-1	Se folosește rutina de
	DEFB	+FF	tratere eroare.

Notă: Evident că nu este o eroare în adevăratul sens - este mai degrabă un salt peste program.

THE 'REM' COMMAND ROUTINE (RUTINA DE COMANDA 'REM')

Adresa de revenire la STNT-RET este coborîta, avînd ca efect fortarea ignorării restului liniei.

1BB2 REM	POP	BC	Coborîre adresă - STNT-RET.
----------	-----	----	-----------------------------

THE 'LINE-END' ROUTINE (RUTINA 'SFIRSIT LINIEI')

Dacă se verifică sintaxa se face o simplă revenire, dar cînd 'se execută', adresa continuată de NXTLIN trebuie verificată înainte de a putea fi utilizată.

1BB3 LINE-END	CALL	2530,SYNTAX-Z	Revenire dacă s-a verificat sintaxa; dacă nu, se aduce adresa în NXTLIN.
	RET	Z	
	LD	HL,(NXTLIN)	revenire si dacă adresa este după sfîrsitul programului - 'executia' este terminată.
	LD	A,+C0	
	AND	(HL)	semnalizare 'instructiune zero' înainte de utilizare.
	RET	NZ	
	XOR	A	

THE 'LINE-USE' ROUTINE (RUTINA 'UTILIZARE LINIEI')

Această rutină are trei functii: i. Schimbă instructia zero cu instructia '1'; ii. Găsește numărul liniei noi si-l introduce în PPC; & iii. Formează adresa de început a liniei următoare.

1BBF LINE-USE	CP	+01	Instructiunea zero devine instructiunea '1'.
	ADC	A,+00	
	LD	D,(HL)	Numărul de linie al liniei ce trebuie folosită este adunat
	INC	HL	

LD	E, (HL)	si trecut în PPC.
LD	(PPC), DE	
INC	HL	Acum se determină 'lungimea' liniei.
LD	E, (HL)	
INC	HL	
LD	D, (HL)	Interschimbarea valorilor.
EX	DE, HL	Se formează adresa de început a liniei următoare în HL si în DE - locatia dinainte de primul caracter al liniei 'următoare'.
ADD	HL, DE	
INC	HL	

THE 'NEXT-LINE' ROUTINE (RUTINA 'URNATOAREA LINIE')

La intrare registrul pereche HL indică locatia după sfîrsitul liniei 'următoare' ce va fi tratată, iar registrul pereche DE indică locatia dinaintea primului caracter al liniei. Aceasta se aplică liniilor din zona programului si de asemenea unei linii din spatiul de editare - unde linia următoare va fi aceeași linie din nou pînă cînd mai sînt instructiuni de interpretat.

1BD1 NEXT-LINE	LD	(NXTLIN), HL	Setare NXTLIN pentru a mai utiliza o dată linia curentă ce a fost completată.
	EX	DE, HL	Ca de obicei CH-ADD indică locatia dinaintea primului caracter considerat.
	LD	(CH-ADD), HL	Se aduce numărul instructiei.
	LD	D, A	Se șterge registrul E dacă se folosește EACH-STMT.
	LD	E, +00	Se șterge registrul E dacă se folosește EACH-STMT.
	LD	(NSPPC), +FF	Se șterge registrul E dacă se folosește EACH-STMT.
	DEC	D	Se șterge registrul E dacă se folosește EACH-STMT.
	LD	(SUBPPC), D	Se șterge registrul E dacă se folosește EACH-STMT.
	JP	Z, 1B28, STMT-LOOP	Se șterge registrul E dacă se folosește EACH-STMT.
	INC	D	Se șterge registrul E dacă se folosește EACH-STMT.
	CALL	198B, EACH-STMT	Se șterge registrul E dacă se folosește EACH-STMT.
	JR	Z, 1BF4, STMT-NEXT	Se șterge registrul E dacă se folosește EACH-STMT.

Prezentarea N - 'Instructiune pierdută'

1BEC REPORT-N	RST	0008, ERROR-1	Se apelează rutina de tratare eroare.
	DEFB	+16	

THE 'CHECK-END' SUBROUTINE (SUBRUTINA 'VERIFICARE SFIRSIT')

Aceasta este o rutină importantă si este apelată din mai multe locuri în programul monitor cînd este verificată sintaxa liniei editate. Scopul rutinei este să dea prezentarea erorii dacă nu s-a completat sfîrsitul instructiunii si pentru a muta în instructiunea următoare dacă sintaxa este corectă.

1BEE CHECK-END	CALL	2530, SYNTAX-Z	Nu se porneste dacă nu este verificată sintaxa.
	RET	NZ	Se șterge registrul E dacă se folosește EACH-STMT.
	POP	BC	Se șterge registrul E dacă se folosește EACH-STMT.
	POP	BC	Se șterge registrul E dacă se folosește EACH-STMT.

THE 'STMT-NEXT' ROUTINE (RUTINA 'STMT-NEXT')

În cazul în care caracterul prezent este un 'carriage return', atunci 'instructiunea următoare' este în 'linia următoare'; dacă '.' este în aceeași linie si totuși se găsește orice alt caracter atunci este eroare de sintaxă.

1BF4 STMT-NEXT	RST	0018, GET-CHAR	Se aduce caracterul prezent.
	CP	+0D	Se consideră 'linia următoare' dacă este un 'carriage return'.
	JR	Z, 1BB3, LINE-END	Se consideră 'instructiunea următoare' dacă este un '.'.
	CP	+3A	Altfel a fost o eroare de sintaxă.
	JP	Z, 1B28, STMT-LOOP	
	JP	1C8A, REPORT-C	

THE 'COMMAND CLASS' TABLE (TABELUL 'CLASA DE COMANDA')

I	II	III	I	II	III
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	84	CLASS-09,1CBE
1C05	67	CLASS-04,1C6C	1C0B	81	CLASS-0A,1C8C
1C06	0B	CLASS-05,1C11	1C0C	CF	CLASS-0B,1CDB

Notă: I - adresa
 II - deplasamentul
 III - numărul clasei

THE 'COMMAND CLASSES - 00, 03 & 05' ('CLASELE DE COMANDA - 00, 03 & 05')
 Comenzile clasei 03 pot fi sau nu urmate de un număr, de exemplu RUN & RUN 200.

1C0D CLASS-03 CALL 1CDE, FETCH-NUM Se aduce un număr, dar dacă lipsește se folosește zero.

Comenzile clasei 00 nu trebuie să aibă nici un operand, de exemplu COPY & CONTINUE.

1C10 CLASS-00 CP A Setare fanion zero pentru mai târziu.

Comenzile din clasa 05 pot fi urmate de un set de numere, de exemplu PRINT & PRINT "222".

1C11 CLASS-05 POP BC In toate cazurile se scoate adresa - SCANN-LOOP.
 CALL Z,1REE,CHECK-END Dacă se tratează comenzi din clasele 00 & 03 și sintaxa a fost verificată se trece la considerarea următoarei instrucțiuni.
 EX DE,HL Se salvează indicatorul de linie în registrul pereche DE.

THE 'JUMP-C-R' ROUTINE (RUTINA 'JUMP-C-R')

După ce s-au luat în considerare intrările claselor de comandă și intrările separatorului în tabelul parametru se face un salt la rutina de comandă corespunzătoare.

1C16 JUMP-C-R LD HL,(T-ADDR) Se aduce indicatorul la intrările în tabelul parametru și se aduce adresa rutinei de comandă cerută.
 LD C,(HL)
 INC HL
 LD B,(HL)
 EX DE,HL Se schimbă indicatorii la loc și se face un salt indirect la rutina de comandă.
 PUSH BC
 RET

THE 'COMMAND CLASSES - 01, 02 & 04' (CLASELE DE COMANDA 01, 02 & 04)

Aceste trei clase de comandă sînt folosite de comenzile care tratează variabilă - LET, FOR & NEXT și indirect de READ & INPUT.

Clasa de comandă 01 se preocupă de indentificarea variabilei într-o instrucțiune LET, READ sau INPUT.

1C22 CLASS-01 CALL 2832,LOOK-VARS Se caută în zona variabilelor pentru a determina dacă cumva s-a folosit deja variabila.

THE 'VARIABLE IN ASSIGNMENT' SUBROUTINE (SUBRUTINA 'ATRIBUIRE VARIABILA')

Această subrutină dezvoltă valorile corespunzătoare pentru variabilele sistem DEST & STRLEN.

1C22 VAR-A-1 LD (FLAGX),+00 Se initializează FLAGX cu +00
 JR NC,1C30,VAR-A-2 Salt înainte dacă variabila a mai fost folosită înainte.
 SET 1,(FLAGX) Semnalare 'variabilă nouă'.
 JR NZ1C46,VAR-A-3 Va apare eroare dacă se încearcă utilizarea unei 'multimi nedimensionate'.

Prezentarea 2- Nu se găsește variabila

1C2E REPORT-2 RST 0008,ERROR-1 Se apelează rutina de tratare eroare.
 DEFB +01

Se continuă cu tratarea variabilelor existente.

1C30 VAR-A-2 CALL Z,2996,STK-VARS Parametrii unui sir simplu de

BIT	6, (FLAGS)	variabile si toată multimea
JR	NZ, 1C46, VAR-A-3	variabilelor sînt trecute în
XOR	A	stiva calculatorului.
CALL	2530, SYNTAX-2	(STK-VARS va 'taia' un sir
CALL	NZ, 2BF1, STK-FETCH	dacă se cere.)
LD	HL, +5C71	Salt înainte dacă se trateaza
OR	(HL)	o variabilă numerică.
LD	(HL), A	Se sterge registrul A.
		Parametrii sirului sau sirul
		multimii variabile se aduc
		numai dacă s-a verificat
		sintaxa.
		Adresa FLAGX.
		Bitul 0 este setat numai cînd
		se tratează un 'sir simplu'
		astfel semnalizînd 'stergerea
		copiei vechi'.
EX	DE, HL	Acum HL indică sirul sau
		elementul multimii.

Trajectoriile se întîlnesc pentru setarea STRLEN & DEST după cum se cere. Pentru toate variabilele numerice si sirul 'nou' & multimea variabilelor sir STRLEN-lo contine 'litera' numelui variabilei. Dar pentru sirul 'vechi' & multimea variabilelor sir dacă sînt bucăți sau complete ea contine 'lungimea' în 'atribuire'.

1C46 VAR-A-3 LD (STRLEN), BC Setare STRLEN cum se cere.

DEST contine adresa de 'destinatie a unei variabile 'vechi' dar de fapt 'sursa' pentru o variabilă 'nouă'.

LD (DEST), HL Setare DEST cum se cere si

RET revenire.

Clasa de comandă 02 se preocupă cu calculul actual al valorii ce trebuie atribuite instructiunii LET.

1C4E CLASS-02	POP	BC	Adresa SCAN-LOOP este trimisă
	CALL	1C56, VAL-FET-1	Se face atribuirea.
	CALL	1BEE, CHECK-END	Se trece la următoarea
			instrucțiune fie prin
			CHECK-END dacă s-a verificat
			sintaxa, fie STMT-RET, dacă
			este 'perioada de executie'.
	RET		

THE 'FETCH A VALUE' SUBROUTINE (SUBROUTINA 'ADUCEREA UNEI VALORI')

Această subrutină este folosită de instrucțiunile LET, READ & INPUT mai întîi pentru a evalua iar apoi pentru a atribui valoarea variabilei desemnate anterior.

Punctul de intrare VAL-FET-1 este folosit de LET & READ si ia în considerare FLAGS în timp ce punctul de intrare VAL-FET-2 este folosit de INPUT si ia în considerare FLAGX.

1C56 VAL-FET-1	LD	A, (FLAGS)	Se foloseste FLAGS.
1C59 VAL-FET-2	PUSH	AF	Se salvează FLAGS sau FLAGX.
	CALL	24FB, SCANNING	Se evaluează următoarea
			expresie.
	POP	AF	Se aduce vechea valoare a lui
			FLAGX sau FLAGX.
	LD	D, (FLAGS)	Se aduce noul FLAGS.
	XOR	D	Natura - numeric sau sir, a
	AND	+40	variabilei si a expresiei
			trebuie să coincidă.
	JR	NZ, 1C8A, REPORT-C	Se dă prezentarea C dacă ele
			nu coincid.
	BIT	7, D	Salt înainte pentru a realiza
	JP	NZ, 2AFF, LET	actuala atribuirea numai dacă
	RET		s-a verificat sintaxa, cînd
			se face o simplă revenire.

THE 'COMMAND CLASS 04' ROUTINE (RUTINA 'CLASA DE COMANDA 04')

Punctul de intrare al clasei de comandă 04 este folosit de instrucțiunile FOR & NEXT.

1C6C CLASS-04	CALL	28B2, LOOK-VARS	Se caută în zona variabilelor
			dacă variabila a mai fost
			utilizată.
	PUSH	AF	Se salvează registrul pereche
	LD	A, C	AF pînă cînd octetul

OR	+9F	discriminator este testat
INC	A	pentru a se asigura că
JR	NZ,1C8A,REPORT-C	variabila este o variabilă de
		control FOR-NEXT.
POP	AF	Se reface registrul
JR	1C22,VAR-A-1	fanioanelor și salt înapoi
		pentru a face variabila care
		a fost găsită 'variabilă în
		atribuire'.

THE 'EXPECT NUMERIC/STRING EXPRESSIONS' SUBROUTINE (SUBROUTINA 'ASTEPTARE EXPRESII NUMERICE/SIR')

Este o serie de subrutine scurte care sînt utilizate pentru a aduce rezultatul evaluării următoarei expresii. Rezultatul dintr-o singură expresie este returnat ca o 'ultimă valoare' în stiva calculatorului.

Punctul de intrare NEXT-2NUM este folosit cînd CH-ADD trebuie mărit pentru a indica începutul primei expresii.

1C79 NEXT-2NUM RST 0020,NEXT-CHAR Avans CH-ADD.

Punctul de intrare EXPT-2NUM (EQU. CLASS-08) permite evaluarea a două expresii numerice, separate prin virgulă.

1C7A EXPT-2NUM (CLASS-08)	CALL	1C82,EXPT-1NUM	Se evaluează fiecare expresie
			pe rînd - așa că se evaluează
			prima.
	CP	+2C	Se dă prezentarea de eroare
	JR	NZ,1C8A,REPORT-C	dacă separatorul nu este o
			virgulă.
	RST	0020,NEXT-CHAR	Avans CH-ADD.

Punctul de intrare EXPT-1NUM (EQU. CLASS-06) permite evaluarea unei singure expresii numerice.

1C82 EXPT-1NUM (CLASS-06)	CALL	24FB,SCANNING	Evaluarea următoarei expresii
	BIT	6,(FLAGS)	Se revine atîta timp cît
	RET	NZ	rezultatul este numeric;
			altfel este o eroare.

Prezentarea C - Nonsens în BASIC

1C8A REPORT-C	RST	0008,ERRDR-1	Se apelează rutina de tratare
	DEFB	+0B	eroare.

Punctul de intrare EXPT-EXP (EQU. CLASS-0A) permite evaluarea unei singure expresii sir.

1C8C EXPT-EXP (CLASS-0A)	CALL	24FB,SCANNING	Se evaluează următoarea
	BIT	6,(FLAGS)	expresie.
	RET	Z	De această dată se revine
	JR	1C8A,REPORT-C	dacă rezultatul indică un
			sir; altfel se prezintă
			eroare.

THE 'SET PERMANENT COLOURS' SUBROUTINE (EQU CLASS -07) (SUBROUTINA 'SETUL CULORILOR PERMANENTE' (EQU CLASS -07))

Această subrutină permite culorilor permanente temporare să devină permanente. Ca și clasa de comandă 07, este de fapt o rutină de comandă pentru comanda a șase culori.

1C96 PERMS (CLASS-07)	BIT	7,(FLAGS)	Fanionul sintaxă/execuție
	RES	0,(TV-FLAG)	este gata.
	CALL	NZ,0B4D,TEMPS	Semnălizare 'ecran principal'
			Doar pe durata 'execuției' se
			apelează TEMPS pentru a
			asigura că culoarea temporară
			este culoarea ecranului
			principal.
	POP	AF	Se trimite adresa de revenire
			SCAN-LOOP.
	LD	A,(T-ADDR)	Se aduce octetul inferior al
			lui T-ADDR și se scade +13
	SUB	+13	pentru a da ordinul +D9 la
			+DE, care sînt codurile
			simbolului INK la OVER.
	CALL	21FC,CO-TEMP-4	Salt înainte pentru a schimba
			culorile temporare așa cum
			este indicat de instrucțiunea

	CALL	1BEE,CHECK-END	BASIC. Se trece la următoarea instrucțiune și se verifică sintaxa.
	LD	HL,(ATTR-T)	Acum valorile culorilor temporare se fac permanente (ATTR-P & MASK-P).
	LD	(ATTR-P),HL	Aceasta este P-FLAG; și aceasta trebuie considerată.
	LD	HL,+5C91	
	LD	A,(HL)	

Următoarele instrucțiuni copiază bitii pari ai octetului înlocuit cu bitii impari. De fapt realizarea bitilor permanenți este la fel ca și realizarea celor temporari.

RLCA		Mutare mască la stînga.
XOR	9HL)	Se imprimă pe mască numai bitii pari ai celui alt octet
AND	+AA	
XOR	(HL)	
LD	(HL),A	Se reface rezultatul.
RET		

THE 'COMMAND CLASS 09' ROUTINE (RUTINA 'CLASA DE COMANDA 09')

Această rutină este folosită de instrucțiunile PLOT, DRAW & CIRCLE în ordine pentru a specifica absența condițiilor 'FLASH 8; BRIGHT 8; PAPER 8'; acestea sînt setate înaintea considerării oricărui număr de culoare introdus.

1CBE CLASS-09	CALL	2530,SYNTAX-Z	Salt înainte dacă sintaxa s-a verificat.
	JR	Z,1CD6,CL-09-1	Semnalizare 'ecran principal'
	RES	0,(TV-FLAG)	Setare culori temporare pentru ecran principal.
	CALL	0D4D,TEMPS	Aceasta este MASK-T.
	LD	HL,+5C90	Se aduce valoarea ei curentă dar se retine numai partea 'nemascată' a lui INK.
	LD	A,(HL)	Se reface valoarea, care acum indică 'FLASH 8; BRIGHT 8; PAPER 8'.
	OR	+F8	De asemenea se asigură că nu este 'PAPER 9'.
	LD	(HL),A	Se aduce caracterul prezent înainte de a se confirma lucrul cu numărul culorii introduse.
	RES	6,(P-FLAG)	Se lucrează cu numărul culorii locale dominante.
	RST	0018,GET-CHAR	Acum se aduc primii doi operanzi pentru PLOT, DRAW sau CIRCLE.
1CD6 CL-09-1	CALL	21E2,CO-TEMP	
	JR	1C7A,EXPT-2NUM	

THE 'COMMAND CLASS 0B' ROUTINE (RUTINA 'CLASA DE COMANDA 0B')

Această rutină este folosită de instrucțiunile SAVE, LOAD, VERIFY & MERGE.

1CDB CLASS-0B	JP	0605,SAVE-ETC	Salt la rutina de tratare a casetei.
---------------	----	---------------	---

THE 'FETCH A NUMBER' SUBROUTINE (SUBRUTINA 'ADUCEREA UNUI NUMAR')

Această subrutină conduce la următoarea expresie numerică ce trebuie evaluată, dar în cazul în care nu este nici o expresie se folosește zero.

1CDE FETCH-NUM	CP	+0D	Salt înainte dacă s-a ajuns la sfîrșitul liniei.
	JR	Z,1CE6,USE-ZERO	Dar salt la EXPT-1NUM dacă nu este sfîrșitul instrucției.
	CP	+3A	
	JR	NZ,1C82,EXPT-1NUM	

Acum calculatorul este folosit pentru aducerea valorii zero în stiva calculatorului.

1CE6 USE-ZERO	CALL	2530,SYNTAX-Z	Nu se prelucrează operația dacă se verificat sintaxa.
	RET	Z	Utilizare calculator.
	RST	0028,FP-CALC	'Ultima valoare' este acum zero.
	DEFB	+A0,stk-zero	Revenire cu zero adăugat la stivă.
	DEFB	+38,end-calc	
	RET		

THE COMMAND ROUTINES (RUTINELE DE COMANDA)

Secțiunea celor 16K de program monitor de la 1CEE la 23FA conține cele mai multe dintre rutinele de comandă ale interpretorului BASIC.

THE 'STOP' COMMAND ROUTINE (RUTINA COMANDA 'STOP')

Rutina comandă pentru STOP conține doar un apel al rutinei de tratare a erorii.

1CEE STOP (REPORT-9)	RST 0008,ERROR-1 DEFB +08	Se apelează rutina de tratare eroare.
-------------------------	------------------------------	---------------------------------------

THE 'IF' COMMAND ROUTINE (RUTINA COMANDA 'IF')

La intrare, valoarea expresiei între IF și THEN este 'ultima valoare' din stiva calculatorului. Dacă este adevărat logic, atunci se ia în considerare următoarea instrucție; în caz contrar se consideră că linia s-a terminat.

1CF0 IF	PDP BC	Se trimite adresa de revenire STMT-RET.
	CALL 2530,SYNTAX-Z	Salt înainte dacă se verifică sintaxa.
	JR Z,1D00,IF-1	

Acum se folosește calculatorul pentru a 'sterge' ultima valoare din stiva calculatorului dar se lasă registrul pereche DE să adreseze primul octet al valorii.

	RST 0028,FP-CALC	Utilizare calculator.
	DEFB +02,delete	Prezenta 'ultima valoare' se șterge.
	DEFB +38,end-calc	
	EX DE,HL	HL indică primul octet și apelează TEST-ZERO.
	CALL 34E9,TEST-ZERO	Dacă valoarea a fost 'FALSE' (fals) salt la linia următoare.
	JP C,1B23,LINE-END	
1D00 IF-1	JP 1B29,STMT-L-1	Dacă a fost 'TRUE' (adevărată) se sare la următoarea instrucție (după THEN).

THE 'FOR' COMMAND ROUTINE (RUTINA DE COMANDA 'FOR')

Această rutină de comandă este introdusă cu VALUE (valoarea) și LIMIT (limita) pentru instrucțiunea FOR aflate deja în vârful stivei calculatorului.

1D03 FOR	CP +CD	Salt înainte dacă nu s-a dat un 'STEP' (pas).
	JR NZ,1D10,F-USE-1	
	RST 0020,NEXT-CHAR	Avans CH-ADD și se aduce valoarea pasului.
	CALL 1C82,EXPT-INUM	Se trece la următoarea instrucție dacă se verifică sintaxa; altfel, salt înainte.
	CALL 1BEE,CHECK-END	
	JR 1D16,F-REORDER	

Nu s-a dat valoarea pentru STEP (pas), așa că se va folosi valoarea '1'.

1D10 F-USE-1	CALL 1BEE,CHECK-END	Se trece la următoarea instrucție dacă se verifică sintaxa; altfel se folosește calculatorul pentru a pune '1' în stiva calculatorului.
	RST 0028,FP-CALC	
	DEFB +A1,stk-one	
	DEFB +38,end-calc	

Cele trei valori din stiva calculatorului sînt VALUE (v) (valoare), LIMIT (l) (limita) și STEP (s) (pas). Acum aceste valori trebuie modificate.

1D16 F-REORDER	RST 0028,FP-CALC	v,l,s	
	DEFB +C0,stk-mem-0	v,l,s	(mem-0 = s)
	DEFB +02,delete	v,l	
	DEFB +01,exchange	l,v	
	DEFB +E0,get-mem-0	l,v,s	
	DEFB +01,exchange	l,v,s	
	DEFB +38,end-calc		

Acum se stabilește o variabilă de control FOR și se tratează ca o zonă de memorie temporară a calculatorului.

CALL 2AFF,LET	Variabila este găsită sau creată dacă este necesar (este folosit v).
LD (MEN),HL	Se face 'zonă de memorie'.

Variabila care s-a găsit poate fi o variabilă numerică simplă care folosește doar șase locații, caz în care este necesară extensia.

DEC	HL	Se aduce singurul caracter al numelui variabilei.
LD	A, (HL)	Se asigură că bitul 7 al numelui este setat.
SET	7, (HL)	Vor fi șase locații la sfârșit.
LD	BC, +0006	HL indică după ele.
ADD	HL, BC	Se rotește numele și salt dacă a fost deja o variabilă FOR.
RLCA		
JR	C, 1D34, F-L&S	Altfel, se mai creează 13 locații.
LD	C, +0D	Din nou HL va indica poziția LIMIT.
CALL	1655, MAKE-ROOM	
INC	HL	

Acum se adaugă valorile inițiale ale lui LIMIT și STEP.

1D34 F-L&S	PUSH	HL	Este salvat indicatorul.
	RST	0028, FP-CLAC	1, s
	DEFB	+02, delete	1
	DEFB	+02, delete	-
	DEFB	+38, end-calc	DE indică încă pe '1'.
	POP	HL	Indicatorul este refăcut și cei doi indicatori sînt interschimbați.
	EX	DE, HL	Se mută cei zece octeți ai lui LIMIT și STEP.
	LD	C, +0A	
	LDIR		

Se introduc acum ciclări pentru numărul liniei și numărul instrucțiunii.

LD	HL, (PPC)	Numărul liniei curente.
EX	DE, HL	Se interschimbă registrii înainte de a adăuga numărul liniei la variabila de control FOR.
LD	(HL), E	Instrucțiunea de ciclare este întotdeauna următoarea instrucțiune - dacă există sau nu.
INC	HL	
LD	(HL), D	
LD	D, (SUBPPC)	
INC	D	
INC	HL	
LD	(HL), D	

Subrutina NEXT-LOOP este apelată pentru a testa posibilitatea unei 'tregeri' și se revine dacă există una posibilă; altfel instrucțiunea după bucla FOR - NEXT trebuie identificată.

CALL	1DDA, NEXT-LOOP	Este 'o trecere' posibilă?
RET	NC	Revenire dacă nu este.
LD	B, (STRLEN-10)	Se aduce numele variabilei.
LD	HL, (PPC)	Se copiază numele liniei prezente în PPC.
LD	(NEMPPC), HL	Se aduce numărul instrucției curente și se completează față de 2.
LD	A, (SUBPPC)	Se transferă rezultatul în registrul D.
NEG		Se aduce valoarea curentă a lui CH-ADD.
LD	D, A	Se caută 'NEXT'.
LD	HL, (CH-ADD)	
LD	E, +F3	

Acum se caută în zona de program, de la indicatorul prezent în continuare, prima intervenție a lui NEXT urmată de o variabilă corectă.

1D64 F-LOOP	PUSH	BC	Se salvează numele variabilei
	LD	BC, (NXTLIN)	Se aduce valoarea curentă a lui NXTLIN.
	CALL	1D86, LOOK-PROG	Acum este cercetat spațiul programului și BC va fi schimbat cu fiecare linie cercetată.
	LD	(NXTLIN), BC	Se salvează indicatorul.
	POP	BC	Se reface numele variabilei.
	JR	C, 1D34, REPORT-1	Dacă mai departe nu există NEXT, atunci se trimite eroare.
	RST	0020, NEXT-CHAR	Se avansează peste NEXT care a fost găsit.
	OR	+20	Se permite despărțirea literelor mici de literele
	CP	B	

```
JR    Z,1D7C,F-FOUND
RST   0020,NEXT-CHAR
JR    1D64,F-LOOP
```

mari înaintea testării
numelui noii variabile.
Salt înainte dacă coincid.
Din nou avans CH-ADD și salt
înapoi dacă nu este corectă
variabila.

NEWPPC conține numărul de linie al liniei în care s-a găsit NEXT corect. Acum
trebuie găsit numărul instrucțiunii și stocat în NSPPC.

```
1D7C F-FOUND      RST   0020,NEXT-CHAR
                  LD     A,+01
                  SUB   D
                  LD     (NSPPC),A
                  RET
```

Adresa CH-ADD.
Contorul de instrucțiuni din
registru D numără
instrucțiunile înapoi de la
zero, așa că trebuie scăzut
cu 1.
Rezultatul este stocat.
Acum revenire - la STMT-RET.

Prezentarea I - FOR fără NEXT

```
1D84 REPORT-1    RST   0008,ERROR-1
                  DEF   +11
```

Se apelează rutina de tratare
eroare.

THE 'LOOK-PROG' SUBROUTINE (SUBROUTINA 'CAUTA PROGRAM')

Această subrutină este folosită pentru a găsi intervențiile pentru DATA, DEF
FN sau NEXT. La intrare codul simbolului corespunzător este în registrul E iar
registru pereche HL indică începutul zonei în care se caută.

```
1D86 LOOK-PROG   LD     A,(HL)
                  CP     +3A
                  JR     Z,1DA3,LOOK-P-2
```

Se aduce prezentul caracter.
Salt înainte dacă este ':'
care va indica faptul că mai
sunt instrucțiuni în linia
prezentă.

Acum se introduce o buclă care va examina mai departe fiecare linie din
program.

```
1D88 LOOK-P-1    INC   HL
                  LD     A,(HL)
                  AND   +C0
                  SCF
                  RET   NZ
                  LD     B,(HL)
                  INC   HL
                  LD     C,(HL)
                  LD     (NEWPPC),BC
                  INC   HL
                  LD     C,(HL)
                  INC   HL
                  LD     B,(HL)
                  PUSH  HL
                  ADD   HL,BC
                  LD     B,B
                  LD     C,L
                  POP   HL
                  LD     D,+00
```

Se aduce octetul superior al
numărului liniei și se revine
cu transportul setat dacă nu
mai sunt linii în program.

Se aduce numărul liniei și se
trece în NEWPPC.

Apoi este colectată lungimea.

```
1DA3 LOOK-P-2    PUSH  BC
                  CALL  1988,EACH-STMT
                  POP   BC
                  RET   NC
                  JR    1D88,LOOK-P-1
```

Indicatorul este salvat până
când adresa sfârșitului de
linie este formată în
registru pereche BC.
Se reface indicatorul.
Se setează contorul
instrucțiunilor pe zero.
Indicatorul sfârșit-de-linie
este salvat până când sunt
examineate instrucțiile de pe
linie.
Se revine dacă a fost o
'intervenție'; altfel se
consideră următoarea linie.

THE 'NEXT' COMMAND ROUTINE (RUTINA COMANDA 'URMATOAREA')

'Atribuirea valorii' a fost deja determinată (vezi CLASS -04, 1C6C); a rămas
să se schimbe VALUE (valoarea) așa cum se cere.

```
1DA8 NEXT        BIT   1,(FLAGX)
                  JP    NZ,1C2E,REPORT-2
                  LD   HL,(DEST)
                  BIT   7,(HL)
                  JR    Z,1DD8,REPORT-1
```

Salt pentru a prezenta eroare
dacă nu s-a găsit variabila.
Se aduce adresa variabilei și
apoi se testează numele.

În continuare calculatorul prelucrează valoarea (VALUE) și pasul (STEP)
variabilei.

INC	HL	Se trece peste nume.
LD	(MEM),HL	Face variabila o 'zona de memorie' temporară.
RST	0028,FP-CALC	-
DEFB	+E0,get-mem-0	v
DEFB	+E2,get-mem-2	v,s
DEFB	+0F,addition	v+s
DEFB	+C0,st-mem-0	v+s
DEFB	+02,delete	-
DEFB	+38,end-calc	-

Rezultatul obtinut prin adunarea lui VALUE si STEP este acum comparat cu LIMIT apelând NEXT-LOOP.

CALL	1DDA,NEXT-LOOP	Se compară noua VALUE față de LIMIT.
RET	C	Acum se revine dacă ciclul FOR-NEXT a fost complet.

Altfel se adună 'ciclarea' numărului de linie si instructiei.

LD	HL,(MEM)	Se caută adresa octetului inferior a numărului liniei ciclare.
LD	DE,+000F	
ADD	HL,DE	Acum se aduce numărul acestei linii.
LD	E,(HL)	
INC	HL	
LD	D,(HL)	
INC	HL	
LD	H,(HL)	Urmează numărul instructiei.
EX	DE,HL	Se schimbă numerele înainte de a sări în față pentru a fi tratate ca linia de destinație a comenzii 60 TD.
JP	1E73,60-TD-2	

Prezentarea 1 - NEXT fără FOR

1DD8 REPORT-1	RST 0008,ERROR-1	Se apelează rutina de tratare eroare.
	DEFB +00	

THE 'NEXT-LOOP' SUBROUTINE (SUBRUTINA 'URMATOAREA BUCLA')

Această buclă este folosită pentru a determina dacă LIMIT a fost depășită de prezenta VALUE. Nota trebuie luată din semnul lui STEP.

Subrutina revine cu fanionul de transport setat dacă LIMIT este depășită.

1DDA NEXT-LOOP	RST 0028,FP-CALC	-
	DEFB +E1,get-mem-1	1
	DEFB +E0,get-mem-0	1,v
	DEFB +E2,get-mem-2	1,v,s
	DEFB +36,less-0	1,v,(1/0)
	DEFB +00,jump-true	1,v,(1/0)
	DEFB +02,to NEXT-1	1,v,(1/0)
	DEFB +01,exchange	v,1
1DE2 NEXT-1	DEFB +03,subtract	v-1 sau 1-v
	DEFB +37,greater-0	(1/0)
	DEFB +00,jump-true	(1/0)
	DEFB +04,to NEXT-2	-
	DEFB +38,end-calc	-
	AND A	
	RET	Se sterge fanionul de transport si revenire - ciclare posibilă.

Dacă ciclarea este imposibilă fanionul de transport este setat.

1DE9 NEXT-2	DEFB +38,end-calc	-
	SCF	Setare fanion de transport si revenire.
	RET	

THE 'READ' COMMAND ROUTINE (RUTINA DE COMANDA 'READ')

Comanda READ (citire) permite citirea unei liste DATA si are un efect similar cu o serie de instructiuni LET.

Fiecare atribuire în cadrul unei singure instructiuni READ este tratată una după alta. Variabila sistem X-PTR este utilizată ca o locație de stocare a indicatorului instructiunii READ pînă cînd CH-ADD este utilizat pentru a parcurge lista DATA.

1DEC READ-3	RST 0020,NEXT-CHAR	Aici se vine la fiecare trecere, după prima, pentru mutare de-a lungul
-------------	--------------------	--

1DED READ	CALL 1C1F, CLASS-01	instrucțiunii READ. Se ia în considerare dacă variabilele au mai fost folosite înainte; se caută intrarea existentă, dacă a fost. Salt înainte dacă se verifică sintaxa. Se salvează indicatorul curent CH-ADD în X-PTR. Se aduce indicatorul curent al listei DATA și salt înainte dacă nu se mai află o nouă instrucțiune DATA. Căutarea este pentru 'DATA'. În cazul în care căutarea a reușit, salt înainte.
	CALL 2530, SYNTAX-2	
	JR Z, 1E1E, READ-2	
	RST 0018, SET-CHAR	
	LD (X-PTR), HL	
	LD HL, (DATADD)	
	LD A, (HL)	
	CP +2C	
	JR Z, 1E0A, READ-1	
	LD E, +E4	
	CALL 1D86, LOOK-PROG	
	JR NC, 1E0A, READ-1	

Prezentarea E - Afară din DATA

1E08 REPORT-E	RST 0008, ERROR-1	Se apelează rutina de tratare eroare.
	DEFB +00	

Se continuă - se scoate o valoare din lista DATA.

1E0A READ-1	CALL 0077, TEMP-PTR1	Indicatorul avansează de-a lungul listei DATA și setare CH-ADD.
	CALL 1C56, VAL-FET-1	Se aduce valoarea și se i se atribuie variabilei.
	RST 0018, SET-CHAR	Se aduce valoarea curentă a lui CH-ADD și se stochează în DATADD.
	LD (DATADD), HL	
	LD HL, (X-PTR)	Se aduce indicatorul la instrucțiunea READ și se șterge X-PTR.
	LD (X-PTR-HI), +00	CH-ADD va indica încă o dată instrucțiunea READ.
	CALL 0078, TEMP-PTR2	Se aduce prezentul caracter și se vede dacă este ' '.
1E1E READ-2	RST 0018, SET-CHAR	Dacă este, atunci salt înapoi ca și cum ar urma numere;
	CP +2C	altfel, se revine fie prin CHECK-END (dacă se verifică sintaxa), fie cu instrucția RET (la STMT-RET).
	JR Z, 1DEC, READ-3	
	CALL 1BEE, CHECK-END	
	RET	

THE 'DATA' COMMAND ROUTINE (RUTINA DE COMANDA 'DATA')

În timpul verificării sintaxei, se verifică o instrucțiune DATA pentru a se asigura că ea conține o serie de expresii valide, separate prin virgulă. Dar în timpul execuției această instrucțiune este omisă.

1E27 DATA	CALL 2530, SYNTAX-2	Salt înainte dacă se verifică sintaxa.
	JR NZ, 1E37, DATA-2	

Acum se introduce o buclă care lucrează cu fiecare expresie din instrucțiunea DATA.

1E2C DATA-1	CALL 24FB, SCANNING	Defilarea expresiei următoare
	CP +2C	Se verifică dacă separatorul este corect - o ' , ' ;
	CALL NZ, 1BEE, CHECK-END	Însă se trece la următoarea instrucțiune dacă nu coincid.
	RST 0020, NEXT-CHAR	Bucla se ciclează pînă cînd mai sînt expresii ce trebuie verificate.
	JR 1E2C, DATA-1	

Instrucțiunea DATA trebuie omisă în 'timpul execuției'.

1E37 DATA-2	LD A, +E4	Este o instrucție DATA care trebuie omisă.
-------------	-----------	--

THE 'PASS-BY' SUBROUTINE (SUBRUTINA 'TRECERE' 'OMITERE')

La intrare registrul A va conține fie simbolul 'DATA' fie simbolul 'DEF FN', în funcție de instrucțiunea care a fost 'sărită'.

1E39 PASS-BY	LD B, A	Registrul pereche BC va conține un număr foarte mare.
	CPDR	Se caută înapoi de-a lungul

LD DE,+0200
JP 198B,EACH-STNT

instrucțiunii simbolul.
Acum caută de-a lungul liniei
de după instrucțiune. (A 'D-1'
instrucție de la poziția
curentă.)

THE 'RESTORE' COMMAND ROUTINE (RUTINA DE COMANDA 'RESTORE')

Operandul unei instrucțiuni RESTORE este luat ca un număr de linie, iar în cazul în care nu este dat nici un operand se consideră zero.

Punctul de intrare REST-RUN este folosit de rutina de comandă RUN.

1E42 RESTORE CALL 1E99,FIND-INT2

Se comprimă operandul în
registru pereche BC.

1E45 REST-RUN LD H,B
LD L,C
CALL 196E,LINE-ADDR

Rezultatul este transferat în
registru pereche HL.

DEC HL
LD (DATADD),HL
RET

Acum se determină adresa
acelei linii sau 'a primei
linii de după'.
DATADD va indica locația de
dinainte.
Revenire odată ce s-a făcut.

THE 'RANDOMIZE' COMMAND ROUTINE (RUTINA DE COMANDA 'RANDOMIZE')

Din nou operandul este comprimat în registrul pereche BC și transferat în variabila sistem cerută. Dacă operandul este zero, imediat se utilizează valoarea din FRAMES1 și FRAMES2.

1E4F RANDOMIZE CALL 1E99,FIND-INT2
LD A,B
OR C
JR NZ,1E5A,RAND-1
LD BC,(FRAMES1)

Se aduce operandul.
Salt înainte dacă valoarea
operandului este zero.

1E5A RAND-1 LD (SEED),BC
RET

Se aduc imediat cei doi
octeți de ordin inferior din
FRAMES.

Înainte de revenire se
introduce rezultatul în
variabila sistem SEED.

THE 'CONTINUE' COMMAND ROUTINE (RUTINA DE COMANDA 'CONTINUARE')

Numărul de linie cerut și numărul instrucțiunii din acea linie care fac obiectul unui salt.

1E5F CONTINUE LD HL,(OLDPPC)
LD D,(OSPCC)
JR 1E73,GO-TO-2

Numărul liniei.
Numărul instrucțiunii.
Salt înainte.

THE 'GO TO' COMMAND ROUTINE (RUTINA DE COMANDA 'GO TO')

Operandul unui GO TO trebuie să fie un număr de linie în limitele '1' la '9999', dat actualul test se face pentru o valoare mai mare de '61439'.

1E67 GO-TO CALL 1E99,FIND-INT2
LD H,B
LD L,C
LD D,+00
LD A,H
CP +F0
JR NC,1E9F,REPORT-B

Se aduce operandul și se
transferă în registrul
pereche HL.

Setare număr instrucțiune cu
zero.

Se dă mesaj de eroare -
partea întreagă iese din
limite - cu linii peste
'61439'.

Punctul de intrare GO-TO-2 este folosit la determinarea numărului liniei a liniei următoare de tratat în câteva situații.

1E73 GO-TO-2 LD (NEWPPC),HL
LD (NSPPC),D
RET

Se introduce numărul liniei
și apoi numărul instrucției.
Revenire; - la STMT-RET.

THE 'OUT' COMMAND ROUTINE (RUTINA DE COMANDA 'IESIRE')

Cei doi parametri și instrucțiunii OUT sînt aduși din stiva calculatorului și folosiți după indicații.

1E7A OUT CALL 1E85,TWO-PARAM
OUT (C),A
RET

Se aduc parametrii.
Instrucția OUT actuală.
Revenire; - la STMT-RET.

THE 'POKE' COMMAND ROUTINE (RUTINA DE COMANDA 'INTRODUCERE')

Intr-o manieră similară se execută și operațiunea POKE.

1E80 POKE CALL 1E85,TWO-PARAM Se aduc operanzii.
 LD (BC),A Operatia POKE actuală.
 RET Revenire; - la STMT-RET.

THE 'TWO-PARAM' SUBROUTINE (SUBRUTINA 'DOI-PARAM')
 Paramterul din vârful stivei calculatorului trebuie comprimat ca să intre într-un singur registru. Dacă este negativ, se completează față de doi. Al doilea paramteru trebuie să fie comprimabil într-un registru pereche.

1E85 TWO-PARAM CALL 2DD5,FP-TO-A Se aduce paramterul.
 JR C,1E9F,REPORT-B Se dă eroare dacă numărul
 este prea mare.
 JR Z,1E8E,TWO-P-1 Salt înainte în cazul
 NEG numerele pozitive, dar dacă
 numărul este negativ se face
 complementul față de doi.
 1E8E TWO-P-1 PUSH AF Se salvează primul paramteru
 CALL 1E99,FIND-INT2 pînă cînd se aduce cel de al
 doilea paramteru.
 POP AF Se restituie primul paramteru
 RET înainte de revenire.

THE 'FIND INTEGERS' SUBROUTINES (SUBRUTINA 'GASIRE PARTE INTREAGA')
 'Ultima valoare' din stiva calculatorului este adusă și comprimată într-un singur registru sau într-un registru pereche cu intrarea pe la FIND-INT1 și, respectiv, FIND-INT2.

1E94 FIND-INT1 CALL 2DD5,FP-TO-A Se aduce 'ultima valoare'.
 JR 1E9C,FIND-I-1 Salt înainte.
 1E99 FIND-INT2 CALL 2DA2,FP-TO-BC Se aduce 'ultima valoare'.
 1E9C FIND-I-1 JR C,1E9F,REPORT-B In ambele cazuri depășirea
 RET Z este indicată prin setarea
 fanionului de transport.

Prezentarea B - Parte întreagă afară din interval.

1E9F REPORT-B RST 0008,ERROR-1 Se apelează rutina de tratare
 DEFB +0A eroare.

THE 'RUN' COMMAND ROUTINE (RUTINA DE COMANDA 'EXECUTIE')
 Paramterul comenzii RUN este trecut în NEWPPC prin apelarea rutinei de comandă GO TO. Se execută apoi operațiile 'RESTORE 0' și 'CLEAR 0' înainte de a se reveni.

1EA1 RUN CALL 1E67,GO-TO Setare NEWPPC după cum este
 LD BC,+0000 cerut.
 CALL 1E4S,REST-RUN Executie 'RESTORE 0'.
 JR 1EAF,CLEAR-1 Iesire prin rutina de comandă
 CLEAR.

THE 'CLEAR' COMMAND ROUTINE (RUTINA DE COMANDA 'CURATIRE')
 Această rutină permite curățirea zonei variabilelor, ecranul este curățat și RANTOP transferat. Ca o consecință a ultimei operații stiva calculatorului este refăcută, aceasta avînd ca efect și curățirea stivei GO SUB.

1EAC CLEAR CALL 1E99,FIND-INT2 Se aduce operandul - în caz
 că lipsește se folosește
 zero.
 1EAF CLEAR-RUN LD A,B Salt înainte dacă operandul
 OR C nu este zero. Cînd se
 JR NZ,1EB7,CLEAR-1 apelează din RUN nu se face
 salt.
 LD BC,(RANTOP) Dacă este zero se folosește
 valoarea existentă în RANTOP.
 1EB7 CLEAR-1 PUSH BC Salvare valoare.
 LD DE,(VARS) Urmează refacerea tuturor
 LD HL,(E-LINE) octetilor din prezenta zonă
 DEC HL a variabilelor.
 CALL 19E5,RECLAIM-1 Se curată spatiul ecranului.
 CALL 0D6B,CLS

Se testează valoarea din registrul pereche BC ce va fi folosită ca RANTOP pentru a se asigura că nu este nici prea mare, nici prea mică.

LD HL,(STKEND) Se incrementează valoarea
 LD DE,+0032 curentă a lui STKEND cu '50'
 ADD HL,DE înainte de a fi testată.
 POP DE Aceasta va fi limita

SBC	HL,DE	inferioară.
JR	NC,1EDA,REPORT-M	RANTOP va fi prea jos.
LD	HL,(P-RANT)	Pentru testul superior
AND	A	valoarea lui RANTOP este
SBC	HL,DE	comparată cu P-RANT.
JR	NC,1EDC,CLEAR-2	Salt înainte dacă este
		acceptată.

Prezentarea M - RANTOP nu este bun

1EDA REPORT-M	RST 0008,ERROR-1	Se apelază rutina de tratare
	DEFB +15	eroare.

Se continuă cu operatia CLEAR (stergere)

1EDC CLEAR-3	EX DE,HL	Acum valoarea poate fi
	LD (RANTOP),HL	trecută în RANTOP.
	POP DE	Se aduce adresa - STMT-RET.
	POP BC	Se aduce 'adresa erorii'.
	LD (HL),+3E	Se introduce marcatorul de
		sfârșit stivă GO SUB.
	DEC HL	Se lasă o locație.
	LD SP,HL	Indicatorul stiva va indica
		stiva GO SUB goală.
	PUSH BC	Apoi se trece 'adresa erorii'
	LD (ERR-SP),SP	în stivă și se salvează
		adresa la ERR-SP.
	EX DE,HL	Se face acum o revenire
	JP (HL)	indirectă la STMT-RET.

Notă: Când rutina este apelată de RUN valorile lui NEWPPC & NSPPC vor fi afectate și nici o instrucțiune care apare după RUN nu va putea fi găsită înainte de a se efectua saltul.

THE 'GO SUB' COMMAND ROUTINE (RUTINA DE COMANDA 'GO SUB')

Prezenta valoare a lui PPC și valoarea incrementată a lui SUBPPC sînt stocate în stiva GO SUB.

1EED GO-SUB	POP DE	Se salvează adresa - STMT-RET
	LD H,(SUBPPC)	Se aduce numărul instrucției
	INC H	și se incrementează.
	EX (SP),HL	Se schimbă 'adresă eroare' cu
		numărul instrucției.
	INC SP	Se reface utilizarea unei
		locații.
	LD BC,(PPC)	Apoi se salvează numărul
	PUSH BC	liniei prezente.
	PUSH HL	Restituire 'adresă eroare' în
	LD (ERR-SP),SP	stiva calculatorului și se
		resetează ERR-SP pentru a o
		indica.
	PUSH DE	Restituire adresă - STMT-RET.
	CALL 1E67,GO-TO-1	Acum se setează NEWPPC &
		NSPPC la valorile cerute.
	LD BC,+0014	Dar înainte de a face saltul
		se face un test pentru
		locație.

THE 'TEST-ROOM' SUBROUTINE (SUBRUTINA 'TEST CAMERA')

Se execută o serie de teste pentru a se asigura că există suficientă memorie liberă accesibilă pentru taskul care a fost preluat.

1F05 TEST-ROOM	LD HL,(STKEND)	Se incrementează valoarea
	ADD HL,BC	luată din STKEND cu valoarea
		transportată în rutină prin
		registru pereche BC.
	JR C,1F15,REPORT-4	Salt înainte dacă rezultatul
		este peste +FFFF.
	EX DE,HL	Se încearcă din nou,
	LD HL,+0050	acceptînd următorii opt
	ADD HL,DE	octeți.
	JR C,1F15,REPORT-4	
	SBC HL,SP	In final valoarea se compară
		cu adresa din stiva
		calculatorului.
	RET C	Dacă rezultatul comparației
		este satisfăcător, se revine.

Prezentarea 4 - Afară din memorie

1F15 REPORT-4 LD L,+03 Este o eroare 'timp de
JP 0055,ERROR-3 executie' si marcatorul de
eroare nu este folosit.

THE 'FREE MEMORY' SUBROUTINE (SUBRUTINA 'MEMORIE LIBERA')
Nu există nici o comandă BASIC 'FREE' în SPECTRUM dar este o subrutină care
realizează acest lucru.
O estimare a cantității de spațiu liber se poate găsi oricând utilizând:
'PRINT 65536-USR 7962'

1F1A FREE-MEM LD BC,+0000 Nu se admite orice deasupra.
CALL 1F05,TEST-ROOM Se face testul si se trece
LD B,H rezultatul în registrul BC
LD C,L înainte de a se reveni.
RET

THE 'RETURN' COMMAND ROUTINE (RUTINA DE COMANDA 'RETURN')
Numărul liniei și numărul instrucțiunii care fac obiectul unui 'RETURN' sînt
aduse din stiva GO SUB.

1F23 RETURN POP BC Se aduce adresa - STMT-RET.
POP HL Se aduce 'adresa eroare'.
POP DE Se aduce ultima intrare în
stiva GO SUB.
LD A,D Intrarea se testează pentru a
CP +3E vedea dacă este marcatorul de
R Z,1F36,REPORT-7 sfîrsit al stivei GO SUB;
salt dacă este.
DEC SP Intreaga intrare foloseste
numai trei locatii.
EX (SP),HL Se schimbă numărul
instrucției cu 'adresa
eroare'.
EX DE,HL Se mută numărul instrucției.
LD (ERR-SP),SP Se resetează indicatorul de
eroare.
PUSH BC Se înlocuiește adresa -
STMT-RET.
JP 1E73,GO-TO-2 Salt înapoi pentru a schimba
NEWPPC & NSPPC.

Prezentarea 7 - RETURN fără GO SUB

1F36 REPORT-7 PUSH DE Se înlocuiește marcatorul de
PUSH HL sfîrsit și 'adresa eroare'.
RST 0008,ERROR-1 Se apelează rutina de tratare
DEFB +06 eroare.

THE 'PAUSE' COMMAND ROUTINE (RUTINA DE COMANDA 'PAUSE')
Perioada unei PAUSE este determinată prin contorizarea numărului de
întreruperi mascabile așa cum intervin la fiecare 1/50 dintr-o secundă.
O PAUSE se termină fie după numărul corespunzător de întreruperi fie prin
variabila sistem FLAGS care indică faptul că s-a apăsător o tastă.

1F3A PAUSE CALL 1E99,FIND-INT2 Se aduce operandul.
1F3D PAUSE-1 HALT Se așteaptă o întrerupere
mascabilă.
DEC BC Se decrementează contorul.
LD A,B Dacă contorul este astfel
OR C redus la zero PAUSE a ajuns
JR Z,1F4F,PAUSE-END la sfîrsit.
LD A,B Dacă operandul a fost zero,
AND C BC va conține acum +FFFF și
INC A această valoare va fi readusă
JR NZ,1F49,PAUSE-2 la zero. Salt pentru toate
INC BC celelalte valori.
1F49 PAUSE-2 BIT 5,(FLAGS) Salt înapoi numai dacă a fost
JR Z,1F3D,PAUSE-1 apăsător o tastă.

Perioada pentru PAUSE s-a terminat acum.

1F4F PAUSE-END RES 5,(FLAGS) Semnalizare 'nici o tastă
RET apăsător'.
Acum se revine - la STMT-RET.

THE 'BREAK-KEY' SUBROUTINE (SUBRUTINA 'TASTA BREAK')
Această subrutină este apelată în mai multe situații pentru a citi tasta
BREAK. Fanionul de transport este returnat resetat numai dacă tastele SHIFT și

BREAK au fost apăstate amândouă deodată.

1F54 BREAK-KEY	LD	A,+7F	Se formează adresa portului +7FFE și se citește într-un octet.
	IN	A,(+FE)	
	RRA		Se examinează bitul 0 prin deplasare în poziția transportului.
	RET	C	Revenire dacă tasta BREAK nu a fost apăsată.
	LD	A,+FE	Se formează adresa portului +FEFE și se citește într-un octet.
	IN	A,(+FE)	
	RRA		Se examinează din nou bitul 0.
	RET		Revenire cu transportul resetat dacă ambele taste au fost apăstate.

THE 'DEF IN' COMMAND ROUTINE (RUTINA DE COMANDA 'DEF IN')

În timpul cât se verifică sintaxa, se verifică instrucția DEF FN pentru a fi sigur că are forma corectă. Spațiul este de asemenea făcut accesibil pentru rezultatul evaluării funcției.

În timpul execuției instrucțiunea DEF FN este omisă.

1F60 DEF-FN	CALL	2530,SYNTAX-7	Salt înainte dacă se verifică sintaxa.
	JR	Z,1F6A,DEF-FN-1	Altfel se omite specificatia 'DEF FN'.
	LD	A,+CE	
	JP	1E39,PASS-BY	

Intîi se consideră variabila funcției.

1F6A DEF-FN-1	SET	6,(FLAGS)	Semnalizare o 'variabilă numerică'.
	CALL	2C8D,ALPHA	Se verifică dacă prezentul cod este o literă.
	JR	NC,1F89,DEF-FN-4	Salt înainte dacă nu este.
	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	CP	+24	Salt înainte dacă nu este un '\$'.
	JR	NZ,1FD7,DEF-FN-2	Se schimbă bitul 6 ca și cum ar fi o variabilă sir.
	RES	6,(FLAGS)	Se aduce următorul caracter.
1F7D DEF-FN-2	RST	0020,NEXT-CHAR	'(' trebuie să urmeze numele variabilei.
	CP	+28	Se aduce următorul caracter.
	JR	NZ,1FB9,DEF-FN-7	Salt înainte dacă este ')' ca și cum nu există parametrii ai funcției.
	RST	0020,NEXT-CHAR	
	CP	+29	
	JR	Z,1FA6,DEF-FN-6	

Acum se introduce o buclă pentru lucrul cu fiecare parametru pe rînd.

1F86 DEF-FN-3	CALL	2C8D,ALPHA	Prezentul cod trebuie să fie o literă.
1F89 DEF-FN-4	JP	NC,1C8A,REPORT-C	Se salvează indicatorul în DE.
	EX	DE,HL	Se aduce următorul caracter.
	RST	0020,NEXT-CHAR	Salt înainte dacă nu este un '\$'.
	CP	+24	Altfel se salvează imediat în DE noul indicator.
	JR	NZ,1F94,DEF-FN-5	Se aduce următorul caracter.
	EX	DE,HL	Se mută indicatorul ultimului caracter al numelui în registrul pereche HL.
1F94 DEF-FN-5	RST	0020,NEXT-CHAR	Acum se fac șase locații după ultimul caracter și se introduce un 'marcător de număr' în prima dintre noile locații.
	EX	DE,HL	Dacă prezentul caracter este '.', atunci salt înapoi ca și cum ar trebui să mai fie un parametru în continuare; altfel se sare afară din buclă.
	LD	BC,+0006	
	CALL	1655,MAKE-ROOM	
	INC	HL	
	INC	HL	
	LD	(HL),+0E	
	CP	+2C	
	JR	NZ,1FA6,DEF-FN-6	
	RST	0020,NEXT-CHAR	
	JR	1F86,DEF-FN-3	

În continuare se consideră definiția funcției.

1FA6 DEF-FN-6	CP +29	Se verifică faptul că ')' există.
	JR NZ,1FB0,DEF-FN-7	Este adus noul caracter.
	RST 0020,NEXT-CHAR	El trebuie să fie '='.
	CP +30	
	JR NZ,1FB0,DEF-FN-7	Se aduce următorul caracter.
	RST 0020,NEXT-CHAR	Se salvează natura variabilei - numeric sau sir.
	LD A,(FLAGS)	Acum se consideră definirea ca o expresie.
	PUSH AF	Se aduce natura variabilei și se verifică dacă este de același tip cum s-a găsit pentru definiție.
	CALL 24FB,SCANNING	Se dă prezentarea erorii, dacă se cere.
	POP AF	Ieșire prin subrutina CHECK-END. (Astfel se face transferul pentru a lua în considerare următoarea instrucție de pe linie.)
	XOR (FLAGS)	
	AND +40	
1FB0 DEF-FN-7	JP NZ,1C8A,REPORT-C	
	CALL 1BEE,CHECK-END	

THE 'UNSTACK-Z' SUBROUTINE (SUBROUTINA 'UNSTACK-Z')

Această subrutină este apelată în câteva cazuri pentru 'a se reveni mai devreme' dintr-o subrutină pentru verificarea sintaxei. Justificarea acestui fapt este ca să permită tipărirea caracterelor actuale sau trecerea valorii în/din stiva calculatorului.

1FC3 UNSTACK-Z	CALL 2530,SYNTAX-Z	A fost verificată sintaxa?
	POP HL	Se aduce adresa de revenire dar se ignoră 'timpul sintaxă'.
	RET Z	In 'timpul execuției' se face o revenire simplă la rutina apelantă.
	JP (HL)	

THE 'LPRINT & PRINT' COMMAND ROUTINES (RUTINELE DE COMANDA 'LPRINT & PRINT')

Canalul corespunzător este deschis după cum este necesar și informația ce trebuie tipărită este considerată pe rând.

1FC9 LPRINT	LD A,+03	Pregătire pentru deschiderea canalului 'P'.
	JR 1FCF,PRINT-1	Salt înainte.
1FC0 PRINT	LD A,+02	Pregătire pentru deschiderea canalului 'S'.
	CALL 2530,SYNTAX-Z	Un canal este deschis numai dacă s-a verificat sintaxa.
	CALL NZ,1601,CHAN-OPEN	Setare temporară a variabilei sistem culoare.
	CALL 0D40,TEMPS	Se apelează subrutina pentru controlul tipării.
	CALL 1FDF,PRINT-2	Se uită pentru a considera următoarea instrucție; se face prin CHECK-END dacă s-a verificat sintaxa.
	CALL 1BEE,CHECK-END	
	RET	

Subrutina pentru controlul tipării este apelată de rutinele de comandă PRINT, LPRINT și INPUT.

1FDF PRINT-2	RST 0018,BET-CHAR	Se aduce primul caracter.
	CALL 2045,PR-END-Z	Salt în față dacă s-a ajuns deja la sfârșitul listei de informații.
	JR Z,1FF2,PRINT-4	

Acum se introduce o buclă pentru lucrul cu 'controlorii de poziți' și se tipărește informația.

1FE5 PRINT-3	CALL 204E,PR-POSN-1	Se lucrează cu orice controlori de poziție consecutivi.
	JR Z,1FE5,PRINT-3	Se lucrează cu un singur număr de tipărire.
	CALL 1FFC,PR-ITEN-1	Se verifică următorii controlori de poziție și informația de tipărit până când nu mai este nici unul.
	CALL 204E,PR-POSN-1	Revenire dacă prezentul caracter este ')'; altfel se consideră execuția unui
	JR Z,1FE5,PRINT-3	
1FF2 PRINT-4	CP +29	
	RET Z	

'carriage return'.

THE 'PRINT A CARRIAGE RETURN' SUBROUTINE (SUBROUTINA 'TIPARIREA UII CARRIAGE RETURN')

1FF5 PRINT-CR	CALL	1FC3,UNSTACK-Z	Revenire dacă se verifică sintaxa.
	LD	A,+0D	Se tipăreste un caracter carriage return si se revine.
	RST	0010,PRINT-A-1	
	RET		

THE 'PRINT ITEMS' SUBROUTINE (SUBROUTINA 'TIPARIRE INFORMATIE')

Această subrutină este apelată de rutinele de comandă PRINT, LPRINT si INPUT. Tipurile diferite de tipărire a informației sînt identificate si tipărite.

1FFC PR-ITEM-1	RST	0018,GET-CHAR	Este adus primul caracter.
	CP	+AC	Salt mai departe numai dacă este un 'AT'.
	JR	NZ,200E,PR-ITEM-2	

Acum se lucrează cu 'AT'.

	CALL	1C79,NEXT-2NUM	Cei doi parametri sînt transferati în stiva calculatorului.
	CALL	1FC3,UNSTACK-Z	Revenire dacă se verifică sintaxa.
	CALL	2307,STK-TO-BC	Parametrii sînt comprimate în registrul pereche BC.
	LD	A,+16	Registrul A este încărcat cu caracterul de control AT înainte de a se executa saltul.
	JR	201E,PR-AT-TAB	

Urmează căutarea unui 'TAB'.

200E PR-ITEM-2	CP	+AD	Salt înainte numai dacă este un 'TAB'.
	JR	NZ,2024,PR-ITEM-3	

Acum se lucrează cu 'TAB'.

	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	CALL	1C82,EXPT-INUM	Se transferă un parametru în stiva calculatorului.
	CALL	1FC3,UNSTACK-Z	Revenire dacă se verifică sintaxa.
	CALL	1E99,FIND-INT2	Valoarea este comprimată în registrul pereche BC.
	LD	A,+17	Registrul A este încărcat cu caracterul de control TAB.

Informațiile de tipărit 'AT' si 'TAB' sînt tipărite prin trei apelări PRINT-OUT.

201E PR-AT-TAB	RST	0010,PRINT-A-1	Se tipăreste caracterul de control.
	LD	A,C	Este urmat de prima valoare.
	RST	0010,PRINT-A-1	
	LD	A,B	In final se tipăreste a doua valoare; apoi se revine.
	RST	0010,PRINT-A-1	
	RET		

Se consideră apoi introducerea informației de culoare.

2024 PR-ITEM-3	CALL	21F2,CO-TEMP-3	Revenire cu transportul resetat dacă s-a găsit o informație de culoare.
	RET	NC	Dacă nu s-a găsit nici una, se continuă.
	CALL	2070,STR-ALTER	Apoi se consideră dacă trebuie schimbat sirul.
	RET	NC	Se continuă numai dacă nu a fost alterat.

Acum informația de tipărire trebuie să fie o expresie, fie numerică fie sir.

	CALL	24FB,SCANNING	Se evaluează expresia, dar se revine dacă se verifică sintaxa.
	CALL	1FC3,UNSTACK-Z	Se testează natura expresiei.
	BIT	6,(FLAGS)	

CALL	Z,2BF1,STK-FETCH	Dacă este sir, se aduc parametrii necesari; dar dacă este numerică, atunci se iese prin PRINT-FP.
JP	NZ,2DE3,PRINT-FP	

Acum se instalează o buclă care să lucreze cu fiecare caracter al sirului pe rând.

203C PR-STRING	LD	A,B	Se revine dacă nu mai este nici un caracter în sir; altfel contorul este decrementat.
	OR	C	
	DEC	BC	
	RET	Z	
	LD	A,(DE)	Se aduce codul și se incrementează indicatorul.
	INC	DE	
	RST	0010,PRINT-A-1	Codul este tipărit și se face un salt pentru a considera caracterele în continuare.
	JR	203C,PR-STRING	

THE 'END OF PRINTING' SUBROUTINE (SUBROUTINA 'SFIRSITUL TIPARIRII')

Dacă nu mai este nimic de tipărit, fanionul zero va fi setat.

2045 PR-END-Z	CP	+29	Revenire dacă este un caracter ')
	RET	Z	
2048 PR-ST-END	CP	+0D	Acum se revine dacă este un 'carriage return'.
	RET	Z	
	CP	+3A	Înainte de revenire se compară cu ')
	RET		

THE 'PRINT POSITION' SUBROUTINE (SUBROUTINA 'POZITIE TIPARIRE')

Prin această subrutină se iau în considerare variatele poziții ale caracterelor de control.

204E PR-POSN-1	RST	0018,GET-CHAR	Se aduce prezentul caracter.
	CP	+3B	Salt înainte dacă este ';
	JR	Z,2067,PR-POSN-3	
	CP	+2C	De asemenea se sare înainte dacă este orice caracter diferit de ';
	JR	NZ,2061,PR-POSN-2	dar nu se tipărește caracterul dacă se verifică sintaxa.
	CALL	2530,SYNTAX-Z	Se încarcă registrul A cu codul de control 'virgulă' și se tipărește; apoi salt înainte;
	JR	Z,2067,PR-POSN-3	Este un '...?'
	LD	A,+06	Revenire dacă nu este nici unul din controlorii de poziție.
	RST	0010,PRINT-A-1	Se tipărește un 'carriage return' numai dacă se verifică sintaxa.
	JR	2067,PR-POSN-3	Se aduce următorul caracter.
2061 PR-POSN-2	CP	+27	Dacă nu s-a ajuns la finalul instrucțiunii de tipărire,
	RET	NZ	atunci se sare înainte; altfel se revine la rutina apelantă.
	CALL	1FF5,PR-CR	Fanionul zero va fi resetat dacă nu s-a atins sfîrșitul instrucțiunii de tipărire.
2067 PR-POSN-3	RST	0020,NEXT-CHAR	
	CALL	2045,PR-END-Z	
	JR	NZ,206E,PR-POSN-4	
	POP	BC	
206E PR-POSN-4	CP	A	
	RET		

THE 'ALTER STREAM' SUBROUTINE (SUBROUTINA 'SIR ALTERAT')

Această subrutină este apelată întotdeauna când este necesar să se considere dacă utilizatorul dorește să folosească un sir diferit.

2070 STR-ALTER	CP	+23	Se revine cu fanionul de transport setat numai dacă prezentul caracter este '#'. Avans CH-ADD.
	SCF		
	RET	NZ	Se trece parametrul în stiva calculatorului.
	RST	0020,NEXT-CHAR	Se șterge fanionul de transport.
	CALL	1CB2,EXPT-1NUM	Revenire dacă se verifică sintaxa.
	AND	A	Valoarea este trecută în registrul A.
	CALL	1FC3,UNSTACK-Z	Se dă prezentarea D dacă valoarea este peste +FF.
	CALL	1E94,FIND-INT1	Se utilizează canalul pentru
	CP	+10	
	JP	NC,160E,REPORT-0	
	CALL	1601,CHAN-OPEN	

AND A
RET

sirul în discutie.
Se sterge fanionul de
transport si se revine.

THE 'INPUT' COMMAND ROUTINE (RUTINA DE COMANDA 'INPUT')

Această rutină permite valorilor introduse de la tastatură să fie atribuite variabilelor. De asemenea este posibil să fie informatii de tipărire introduse în instrucțiunea INPUT si aceste informatii sînt tipărite în partea inferioară a ecranului.

2089 INPUT	CALL 2530,SYNTAX-Z	Salt înainte dacă s-a verificat sintaxa.
	JR Z,2096,INPUT-1	Se deschide canalul 'K'.
	LD A,+01	
	CALL 1601,CHAN-OPEN	
	CALL 0D6E,CLS-LOWER	Se sterge partea inferioară a ecranului.
2096 INPUT-1	LD (DF-SZ),+01	Se setează partea inferioară astfel încît să aibe mărimea de o linie.
	CALL 20C1,IN-ITEM-1	Se apelează rutina pentru lucrul cu informatii INPUT.
	CALL 1BEE,CHECK-END	Se trece la următoarea instrucțiunea dacă s-a verificat sintaxa.
	LD BC,(S-POSN)	Se aduce pozitia curentă pentru tipărit.
	LD A,(DF-SZ)	Salt înainte dacă pozitia curentă este deasupra părții de jos a ecranului.
	CP X	Altfel se setează pozitia de tipărit în vîrfurile părții de jos a ecranului.
	JR C,20AD,INPUT-2	Resetare S-POSN.
	LD C,+21	Acum se setează contorul de defilare.
	LD B,A	
20AD INPUT-2	LD (S-POSN),BC	
	LD A,+19	
	SUB B	
	LD (SCR-RT),A	
	RES 0,(TV-FLAG)	Semnalizare 'ecran principal'
	CALL 00D9,CL-SET	Se setează variabilele sistem si se iese prin CLS-LOWER.
	JP 0D6E,CLS-LOWER	

Informatiile INPUT si informatiile PRINT introduse sînt prelucrate pe rînd de bucla următoare.

20C1 IN-ITEM-1	CALL 204E,PR-POSN-1	Se consideră mai întîi orice caractere de control al pozitiei.
	JR Z,20C1,IN-ITEM-1	Salt înainte dacă prezentul caracter nu este '('.
	CP +28	Se aduce următorul caracter.
	JR NZ,20D8,IN-ITEM-2	Acum se apelează rutina de comandă PRINT pentru tratarea informatiilor dintre paranteze.
	RST 0020,NEXT-CHAR	Se aduce prezentul caracter.
	CALL 1FDF,PRINT-2	Se dă prezentarea C numai dacă este un caracter ')'. Se aduce următorul caracter si salt înainte pentru a verifica dacă mai sînt în continuare informatii INPUT.
	RST 0018,SET-CHAR	
	CP +29	
	JP NZ,1C8A,REPORT-C	
	RST 0020,NEXT-CHAR	
	JP 21B2,IN-NEXT-2	

Acum se ia în considerare dacă s-a folosit INPUT LINE.

20D8 IN-ITEM-2	CP +CA	Salt înainte dacă nu este 'LINE'.
	JR NZ,20ED,IN-ITEM-3	Avans CH-ADD.
	RST 0020,NEXT-CHAR	Se determină adresa de destinație a variabilei.
	CALL 1C1F,CLASS-01	Semnalizare 'utilizare INPUT LINE'.
	SET 7,(FLAGX)	Se dă prezentarea C numai dacă se folosește o variabilă sir.
	BIT 6,(FLAGS)	Salt înainte pentru iesirea mesajului imediat.
	JP NZ,1C8A,REPORT-C	
	JR 20FA,IN-PROMPT	

Se trece la tratarea variabilelor simple INPUT.

20ED IN-IETM-3	CALL 2C8D,ALPHA	Salt înainte pentru ciclarea
	JP NC,21AF,IN-NEXT-1	buclei dacă prezentul
	CALL 1C1F,CLASS-01	caracter nu este o literă.
	RES 7,(FLAGX)	Se determină adresa de
		destinație pentru variabilă.
		Semnalare nu este INPUT
		LINE'.

Mesajul imediat este acum construit în spațiul de lucru.

20FA IN-PROMPT	CALL 2530,SYNTAX-2	Salt înainte numai dacă este
	JP Z,21B2,IN-NEXT-2	verificată sintaxa.
	CALL 16BF,SET-WORK	Spațiul de lucru este setat
		pe zero.
	LD HL,+5C71	Adresa FLAGX.
	RES 6,(HL)	Semnalizare 'sir rezultat'.
	SET 5,(HL)	Semnalizare 'mod INPUT'.
	LD BC,+0001	Se permite doar o locație
		pentru mesajul imediat.
	BIT 7,(HL)	Salt înainte dacă este
		folosit 'LINE'.
	JR NZ,211C,IN-PR-2	Salt înainte dacă se așteaptă
	LD A,(FLAGX)	o intrare numerică.
	AND +40	
	JR NZ,211A,IN-PR-1	O intrare sir va necesita
	LD C,+03	trei locații.
211A IN-PR-1	OR (HL)	Pentru o intrare numerică
	LD (HL),A	bitul 6 al FLAGX va fi setat.
211C IN-PR-2	RST 0030,BC-SPACES	Este făcut accesibil numărul
		de locații cerut.
	LD (HL),+0D	În ultima locație se trece un
		'carriage return'.
	LD A,C	Se testează bitul 6 din
	RRCA	registru C și salt înainte
	RRCA	dacă este cerută doar o
	JR NC,2129,IN-PR-3	locație.
	LD A,+22	Caracterul 'limite duble'
	LD (DE),A	este încărcat în prima și în
	DEC HL	cea de a doua locație.
	LD (HL),A	
2129 IN-PR-3	LD (K-CUR),HL	Acum se poate salva poziția
		cursorului.

În cazul INPUT LINE editorul poate fi apelat fără pregătiri, dar pentru alte tipuri de INPUT stiva de eroare trebuie schimbată astfel încât să prindă erorile.

	BIT 7,(FLAGX)	Salt înainte cu 'INPUT LINE'.
	JR NZ,215E,IN-VAR-3	
	LD HL,(CH-ADD)	Se salvează valoarea curentă
	PUSH HL	a lui CH-ADD & ERR-SP în
	LD HL,(ERR-SP)	stiva calculatorului.
	PUSH HL	
213A IN-VAR-1	LD HL,+213A	Acesta va fi 'punctul de
	PUSH HL	revenire' în caz de eroare.
	BIT 4,(FLAGX2)	Se schimbă doar indicatorul
	JR Z,2148,IN-VAR-2	stivei de eroare dacă se
	LD (ERR-SP),SP	folosește canalul 'K'.
2148 IN-VAR-2	LD HL,(WORKSP)	Setare HL la începutul liniei
	CALL 11A7,REMOVE-FP	INPUT și se elimină orice
		forme de punct flotant. (Nu
		se face nici o excepție,
		poate doar după o eroare.)
	LD (ERR-NR),+FF	Semnalizare 'încă nu este
		eroare'.
	CALL 0F2C,EDITOR	Acum se ia INPUT și cu
	RES 7,(FLAGX)	fanionul sintaxă/execuție
	CALL 21B9,IN-ASSIGN	indicând sintaxă, se verifică
	JR 2161,IN-VAR-4	INPUT pentru erori; salt dacă
		este în ordine; dacă nu, se
		revine în IN-VAR-1.
		Se aduce 'LINE'.
215E IN-VAR-3	CALL 0F2CC,EDITOR	

Toate variabilele sistem trebuie resetate înainte de a se putea face actuala atribuire.

2161 IN-VAR-4	LD (K-CUR-hi),+00	Adresa cursorului este
		resetată.

	CALL 21D6, IN-CHAN-K	Se execută un salt dacă se folosește alt canal decât canalul 'K'.
	JR NZ, 2174, IN-VAR-5	
	CALL 111D, ED-COPY	Linia intrare este copiată pe ecran și poziția din ECHO-E dă poziția curentă din partea de jos a ecranului.
	LD BC, (ECHO-E)	Adresa FLAGX.
	CALL 0DD9, CL-SET	Semnalizare 'mod editare'.
2174 IN-VARS-5	LD HL, +5C71	Salt înainte dacă se tratează INPUT LINE.
	RES 5, (HL)	
	RIT 7, (HL)	
	RES 7, (HL)	
	JR NZ, 219B, IN-VAR-6	Se coboară adresa IN-VAR-1.
	POP HL	Se resttează ERR-SP la adresa ei originară.
	POP HL	Se salvează adresa originară a lui CH-ADD în X-PTR.
	LD (ERR-SP), HL	Acum, cu fanionul 'sintaxă/ execuție' indicînd 'execuție' se face atribuirea.
	POP HL	Se reface adresa originară a lui CH-ADD și se șterge X-PTR
	LD (X-PTR), HL	
	SET 7, (FLAGX)	Salt înainte pentru a vedea dacă mai sînt în continuare informații INPUT.
	CALL 21B9, IN-ASSIGN	S-a găsit lungimea lui 'LINE' în spațiul de lucru.
	LD HL, (X-PTR)	
	LD (X-PTR-HI), +00	
	LD (CH-ADD), HL	
	JR 21B2, IN-NEXT-2	
219B IN-VARS-6	LD HL, (STKBOT)	
	LD DE, (WORKSP)	
	BCF BC	
	SEC HL, DE	DE indică începutul iar BC conține lungimea.
	LD R, H	Acești parametrii sînt stocați și se face actuala atribuire.
	LD C, L	Salt înainte pentru a lua în considerare următoarele informații.
	CALL 2AB2, STK-ST-#	
	CALL 2AFF, LET	
	JR 21B2, IN-NEXT-2	

Se consideră următoarele informații din instrucțiunea INPUT.

21AF IN-NEXT-1	CALL 1FFC, PR-ITEM-1	Se tratează orice informație de tipărit.
21B2 IN-NEXT-2	CALL 204E, PR-POSM-1	Se tratează orice controlori de poziție.
	JP Z, 20C1, IN-IETM-1	Se ciclează din nou bucla dacă mai sînt informații în continuare; altfel revenire.
	RET	

THE 'IN-ASSIGN' SUBROUTINE (SUBRUTINA 'IN-ASSIGN')

Această subrutină este apelată de două ori pentru fiecare valoare INPUT. O dată cu fanionul sintaxă/ execuție resetat (sintaxă) și o dată setat (execuție).

21B9 IN-ASSIGN	LD HL, (WORKSP)	Se setează CH-ADD pentru a indica prima locație a spațiului de lucru și se aduce caracterul.
	LD (CH-ADD), HL	Este un caracter 'STOP'?
	RST 0018, GET-CHAR	Salt dacă este.
	CP +E2	Altfel se face atribuirea 'valorii' la variabilă.
	JR Z, 21D0, IN-STOP	Se aduce caracterul prezent și se verifică dacă este un 'carriage return'. Revenire dacă este.
	LD A, (FLAGX)	
	CALL 1C59, VAL-FET-2	
	RST 0018, GET-CHAR	
	CP +0D	
	RET Z	

Prezentarea C - Nonsense în BASIC

21CE REPORT-C	RST 0008, ERROR-1	Se apelează rutina de tratare eroare.
	DEFB +0B	

Aici se vine dacă linia INPUT începe cu 'STOP'.

21D0 IN-STOP	CALL 2530, SYNTAX-Z	Nu se dă prezentarea eroare trecerea sintaxei.
	RET Z	

Prezentarea H - STOP în INPUT

21D4 REPORT-H RST 0008,ERROR-1 Se apelează rutina de tratare
 DEFB +10 eroare.

THE 'IN-CHAN-K' SUBROUTINE (SUBROUTINA 'IN-CHAN-K')
 Această subrutină revine cu fanionul zero resetat numai dacă s-a folosit
 canalul 'K'.

21D6 IN-CHAN-K LD HL,(CURCHL) Adresa de bază a canalului de
 INC HL informație este adusă și
 INC HL codul canalului este comparat
 INC HL cu caracterul 'K'.
 LD A,(HL)
 CP +4B
 RET Apoi se revine.

THE 'COLOUR ITEM' ROUTINES (RUTINELE 'INFORMATII CULOARE')

Acest set de rutine poate fi împărțit în două părți:

- i. Introducerea tratării 'informației culoare'.
- ii. Tratarea 'variabilei sistem culoare'.

i. Introducerea informației culoare este tratată prin apelarea subrutinei PRINT-OUT, după cum se cere.

Se introduce o buclă pentru tratarea fiecărei informații pe rând. Punctul de intrare este CO-TEMP-2.

21E1 CO-TEMP-1	RST	0020, NEXT-CHAR	Se consideră următorul caracter în instrucția BASIC.
21E2 CO-TEMP-2	CALL	21F2, CO-TEMP-3	Salt înainte pentru a vedea dacă prezentul cod reprezintă o introducere temporară a informației culoare.
	RET	C	Se revine cu transportul setat dacă nu este o informație culoare.
	RST	0018, GET-CHAR	Se aduce prezentul caracter.
	CP	+2C	Salt înapoi dacă este fie o
	JR	Z, 21E1, CO-TEMP-1	';' fie un ' '; altfel a fost
	CP	+3B	o eroare.
	JR	Z, 21E1, CO-TEMP-1	
21F2 CO-TEMP-3	JP	1C8A, REPORT-C	Iesire prin 'prezentarea C'.
	CP	+D9	Revenire cu fanionul de transport.
	RET	C	Se setează dacă codul nu este între limitele +D9 la +DE (INK la OVER).
	CP	+DF	
	CCF		
	RET	C	
	PUSH	AF	Codul informației de culoare este păstrat pînă cînd CH-ADD a avansat la adresa parametrului care îi urmează.
	RST	0020, NEXT-CHAR	
	POP	AF	

Codul informației culoare și parametrul sînt acum 'tipăriti' apelînd PRINT-OUT în două ocazii.

21FC CO-TEMP-4	SUB	+C9	Intervalul dat (+D9 la +DE) este redus la intervalul caracterului de control 8+10 la +15).
	PUSH	AF	Codul caracterului de control este păstrat pînă cînd parametrul este mutat în stiva calculatorului.
	CALL	1C82, EXPT-INUM	In acest punct se face o revenire dacă se verifică sintaxa.
	POP	AF	Codul caracterului de control este păstrat pînă cînd parametrul este mutat în registrul D.
	AND	A	Caracterul de control este trimis afară.
	CALL	1FC3, UNSTACK-Z	Apoi este adus parametrul și trimis afară înainte de a se reveni.
	PUSH	AF	
	CALL	1E94, FIND-INT1	
	LD	D, A	
	POP	AF	
	RST	0010, PRINT-A-1	
	LD	A, D	
	RST	0010, PRINT-A-1	
	RET		

ii. Variabilele sistem culoare - ATTR-T, MASK-T & P-FLAG - sînt modificate după cum se cere. Această subrutină este apelată de PRINT-OUT. La intrare codul caracterului de control este în registrul A iar parametrul este în registrul D.

De notat că toate schimbările sînt pentru variabilele sistem 'temporare'.

2211 CO-TEMP-5	SUB	+11	Se reduce ordinul și salt înainte cu INK & PAPER.
	ADC	A, +00	
	JR	Z, 2234, CO-TEMP-7	
	SUB	+02	Se reduce încă o dată ordinul și salt înainte cu FLASH & BRIGHT.
	ADC	A, +00	
	JR	Z, 2273, CO-TEMP-C	

Codul de control al culorii va di acum +01 pentru INVERSE și +02 pentru OVER, iar variabila sistem P-FLAG este modificată corespunzător.

CP	+01	Pregătire salt cu OVER.
----	-----	-------------------------

	LD	A, D	Se aduce parametrul.
	LD	B, +01	Pregătire mască pentru OVER.
	JR	NZ, 2228, CO-TEMP-6	Acum salt.
	RLCA		Bitul 2 din registrul A
	RLCA		trebuie resetat pentru
	LD	B, +04	INVERSE 0 si setat pentru
			INVERSE 1; masca va avea
			bitul 2 setat.
2228 CO-TEMP-6	LD	C, A	Registrul A este salvat cît
			timp este testat ordinul.
	LD	A, D	Ordinul corect pentru INVERSE
	CP	+02	si OVER este numai '0 - 1'.
	JR	NC, 2244, REPORT-K	
	LD	A, C	Se aduce registrul A.
	LD	HL, +5C91	P-FLAG trebuie schimbat.
	JR	226C, CO-CHANGE	Iesire prin CO-CHANGE si se
			modifică P-FLAG folosind 'B'
			ca mască. De fapt bit 0
			pentru OVER & bit 2 pentru
			INVERSE.

PAPER & INK sînt prelucrate de rutina următoare. La intrare fanionul de transport este setat pentru INK.

2234 CO-TEMP-7	LD	A, D	Se aduce parametrul.
	LD	B, +07	Pregătire mască pentru INK.
	JR	C, 223E, CO-TEMP-8	Salt înainte cu INK.
	RLCA		Se multiplică de opt ori
	RLCA		parametrul pentru PAPER.
	RLCA		
	LD	B, +38	Pregătire mască pentru PAPER.
223E CO-TEMP-8	LD	C, A	Se salvează parametrul în
			registrul C pînă cînd se
			testează rangul parametrului.
	LD	A, D	Se aduce valoarea inițială.
	CP	+0A	Pentru PAPER/INK se permite
	JR	C, 2246, CO-TEMP-9	un ordin doar între '0 la 9'.

Prezentarea K - Culoare invalidă

2244 REPORT	RST	0008, ERROR-1	Se apelează rutina de tratare
	DEFB	+13	eroare.

Se continuă tratarea PAPER & INK.

2246 CO-TEMP-9	LD	HL, +5C8F	Pregătire modificare ATTR-T.
	CP	+08	Salt înainte cu PAPER/INK
	JR	C, 2258, CO-TEMP-B	'0 la 5'7'.
	LD	A, (HL)	Se aduce valoarea curentă a
	JR	Z, 2257, CO-TEMP-A	lui ATTR-T si se foloseste
			neschimbată, sîrind înainte,
			cu PAPER/INK '8'.
	OR	B	Dar pentru PAPER/INK '9'
	CPL		culorile pentru PAPER si INK
	AND	+24	trebuie să fie negru si alb.
	JR	Z, 2257, CO-TEMP-A	Salt pentru INK/PAPER negru;
	LD	A, B	se continuă pentru INK/PAPER
			alb.
2257 CO-TEMP-A	LD	C, A	Se mută valoarea în registrul
			C.

Mască (B) si valoarea (C) sînt acum folosite pentru a schimba ATTR-T.

2258 CO-TEMP-B	LD	A, C	Se mută valoarea.
	CALL	226C, CO-CHANGE	Acum se schimbă ATTR-T, după
			cum este necesar.

In continuare se consideră MASK-T.

	LD	A, +07	Bitii lui MASK-T sînt setati
	CP	D	numai cînd se foloseste
	SBC	A, A	PAPER/INK '8' sau '9'.
	CALL	226C, CO-CHANGE	Acum se schimbă MASK-T, după
			cum este necesar.

Acum se consideră P-FLAG.

	RLCA		Mască corespunzătoare este
	RLCA		construită în registrul B si

AND	+50	trebuie să schimbe bitii 4 &
LD	B,A	6, după cum este necesar.
LD	A,+08	Bitii lui P-FLAG sînt setati
CP	D	doar cînd se foloseste PAPER/
SBC	A,A	INK '9'.
		Se continuă în CO-CHANGE
		prelucrarea lui P-FLAG.

THE 'CO-CHANGE' SUBROUTINE (SUBRUTINA 'CO-CHANGE')

Această subrutină este folosită pentru 'a imprima' unei variabile sistem 'natura' bitilor registrului A. Registrul B contine o mască ce arată care biti trebuie 'copiați peste' din A în (HL).

226C CO-CHANGE	XOR	(HL)	Bitii, specificati de masca
	AND	B	din registrul B, au valoarea
	XOR	(HL)	schimbata si rezultatul trece
	LD	(HL),A	pentru a forma variabila
			sistem.
	INC	HL	Mutare la adresa urmatoarei
			variabile sistem.
	LD	A,B	Revenire cu masca în
	RET		registrul A.

Urmatoarea rutina trateaza FLASH & BRIGHT.

2273 CO-TEMP-C	SBC	A,A	Fanionul zero este setat
			pentru BRIGHT.
	LD	A,D	Parametrul este încarcat si
	RRCA		rotit.
	LD	B,+80	Pregătire mască pentru FLASH.
	JR	NZ,227D,CO-TEMP-D	Salt înainte cu FLASH.
	RRCA		Rotire timp aditional si
	LD	B,+40	pregătire mască pentru BRIGHT
227D CO-TEMP-D	LD	C,A	Se salveaza valoarea în
			registrul C.
	LD	A,D	Se aduce parametrul si i se
	CP	+08	testeaza ordinul; se permite
	JR	Z,2287,CO-TEMP-E	doar '0', '1' si '8'.
	CP	+02	
	JR	NC,2244,REPORT-K	

Acum se poate modifica variabila sistem ATTR-T.

2287 CO-TEMP-E	LD	A,C	Se aduce valoarea.
	LD	HL,+5C8F	Adresa ATTR-T.
	CALL	226C,CO-CHANGE	Acum se schimba variabila
			sistem.

Acum se considera valoarea din MASK-T.

	LD	A,C	Se aduce din nou valoarea.
	RRCA		Bitul setat din FLASH/BRIGHT
	RRCA		'8' (bitul 3) este mutat în
	RRCA		bitul 7 (pentru FLASH) sau
			în bitul 6 (pentru BRIGHT).
	JR	226C,CO-CHANGE	Iesire prin CO-CHANGE.

THE 'BORDER' COMMAND ROUTINE (RUTINA DE COMANDA 'BORDER')

Parametrul comenzii BORDER este folosit cu o comandă OUT pentru actuala schimbare a culorii marginii. Apoi parametrul este salvat în variabila sistem BORDER.

2294 BORDER	CALL	1E94,FIND-INT1	Parametrul este adus si i se
	CP	+08	testeaza rangul.
	JR	NC,2244,REPORT-K	
	OUT	(+FE),A	Instructiunea OUT este apoi
			folosita pentru setarea
			culorii marginii.
	RLCA		Apoi parametrul este
	RLCA		multiplicat de opt ori.
	RLCA		
	BIT	5,A	Dacă culoarea marginii este o
	JR	NZ,22A6,BORDER-1	culoare 'luminoasă' atunci
			culoarea INK (cernelii) din
			spatiul de editare trebuie
			să fie neagră - se execută
			un salt.
	XOR	+07	Se schimba culoarea lui INK.
22A6 BORDER-1	LD	(BORDER),A	Se seteaza variabila sistem,

RET

după cum se cere și se revine.

THE 'PIXEL ADDRESS' SUBROUTINE (SUBRUTINA 'ADRESA PIXEL')

Această subrutină este apelată de subrutina POINT și de rutina de comandă PLOT. Ea este introdusă cu coordonatele unui pixel în registrul pereche BC și revine cu registrul pereche HL conținând adresa octetului fisierului de afisaj care conține acest pixel și cu A indicând poziția pixelului în cadrul octetului.

22AA PIXEL-ADD	LD	A,+AF	Se testează dacă coordonata y
	SUB	B	(în B) nu este mai mare decât
	JP	C,24F9,REPORT-B	175.
	LD	B,A	Acum B conține 175 minus y.
	AND	A	A conține b7b6b5b4b3b2b1b0,
	RRA		bit din B. Si acum
			0b7b6b5b4b3b2b1.
	SCF		
	RRA		Acum 10b7b6b5b4b3b2.
	AND	A	
	RRA		Acum 010b7b6b5b4b3.
	XOR	B	
	AND	+F8	In final 010b7b6b2b1b0, asa
	XOR	B	că H devine $64 + 8 * INT(B/64) +$
	LD	H,A	$B(mod 8)$, octetul superior al
	LD	A,C	adresei pixelului. C conține
			X.
			A începe cu c7c6c5c4c3c2c1c0.
	RLCA		
	RLCA		
	RLCA		Acum este c2c1c0c7c6c5c4c3.
	XOR	B	
	AND	+C7	
	XOR	B	Acum c2c1b5b4b3c5c4c3.
	AND	+C7	
	XOR	B	
	RLCA		
	RLCA		
	LD	L,A	In final b5b4b3c7c6c5c4c3,
	LD	A,C	asa că L devine $32 * INT(B(mod$
	AND	+07	$64)/8) + INT(x/8)$, bit inferior
	RET		A conține $x(mod 8)$; asa că
			pixelul este bitul (A-7) în
			cadrul octetului.

THE 'POINT' SUBROUTINE (SUBRUTINA 'PUNCT')

Această subrutină este apelată de funcția POINT în SCANNING. Este introdusă cu coordonatele pixelului în stiva calculatorului și readuce o ultimă valoare de 1 dacă pixelul este culoarea cernelii, și 0 dacă este culoarea hîrtiei.

22CB POINT-SUB	CALL	2307,STK-TO-BC	Coordonata Y în B, x în C.
	CALL	22AA,PIXEL-ADD	Adresa pixelului în HL.
	LD	B,A	B va număra A+1 bucle pentru
	INC	B	a primi bitul dorit din (HL)
	LD	A,(HL)	în locația 0.
22D4 POINT-LP	RLCA		Deplasare.
	DJNZ	22D4,POINT-LP	
	AND	+01	Bitul este 1 pentru cerneală,
			0 pentru hîrtie.
	JP	2D28,STACK-A	Este pus în stiva
			calculatorului.

THE 'PLOT' COMMAND ROUTINE (RUTINA DE COMANDA 'PLOT')

Această rutină constă dintr-o subrutină principală plus o linie pentru apelarea ei și o linie pentru ieșirea din ea. Rutina principală este utilizată de două ori de CIRCLE iar subrutina este apelată de DRAW. Rutina este introdusă avînd coordonatele unui pixel în stiva calculatorului. Aceasta găsește adresa aceluși pixel și o marchează, ținînd cont de stările lui INVERSE și OVER continute în P-FLAG.

22DC PLOT	CALL	2307,STK-TO-BC	Coordonata Y în B, x în C.
	CALL	22E5,PLOT-SUB	Se apelează subrutina.
	JP	0D4D,TEMPS	Ieșire, se setează temporar
			culoarea.
22E5 PLOT-SUB	LD	(COORDS),BC	Este setată variabila sistem.
	CALL	22AA,PIXEL-ADD	Adresa pixelului în HL.
	LD	B,A	B va număra A+1 bucle pentru
	INC	B	a pune un zero în locul
			potrivit din A.
	LD	A,+FE	Se introduc zerouri.
22F0 PLOT-LOOP	RRCA		Apoi se alipește bitul de

	DJNZ	22F0,PLOT-LOOP	pozitie al pixelului în octet
	LD	B,A	Apoi se copiază în B.
	LD	A,(HL)	Octetul pixelului este
			obținut în A.
	LD	C,(P-FLAG)	S-a obținut P-FLAG și mai
	BIT	0,C	întși se testează față de
			OVER.
	JR	NZ,22FD,PL-TST-IN	Salt dacă OVER este 1.
22FD PL-TST-IN	AND	B	OVER 0 face întși pixel zero.
	BIT	2,C	Se testează INVERSE.
	JR	NZ,2303,PLOT-END	INVERSE 1 lasă pixelul cum a
			fost (OVER 1) sau zero (OVER
			0).
	XOR	B	INVERSE 0 lasă pixelul
	CPL		complementat (OVER 1) sau 1
			(OVER 0).
2303 PLOT-END	LD	(HL),A	Este introdus octetul.
			Ceilalti biti ai săi rămân
			neschimbati în orice caz.
	JP	0BDB,PO-ATTR	Iesire, setare octet atribut.

THE 'STK-TO-BC' SUBROUTINE (SUBRUTINA 'STK-TO-BC')

Această subrutină încarcă două numere în punct flotant în registrul pereche BC. Aceasta va alege parametrul din intervalul +00 - +FF. Ea obține de asemenea în DE 'mutarea diagonală' a valorii (i1,i1) care este utilizată în subrutina de desenare linie DRAW.

2307 STK-TO-BC	CALL	2314,STK-TO-A	Primul număr în A.
	LD	B,A	Deci în B.
	PUSH	BC	Se salvează pentru scurt timp
	CALL	2314,STK-TO-A	Al doilea număr în A.
	LD	E,C	Semnul lui indicator în E.
	POP	BC	Se reface primul număr.
	LD	D,C	Semnul lui indicator în D.
	LD	C,A	Al doilea număr în C.
	RET		Acum BC, DE sînt așa cum s-a
			cerut.

THE 'STK-TO-A' SUBROUTINE (SUBRUTINA 'STK-TO-A')

Această subrutină încarcă registrul A cu numărul în virgulă flotantă conținut în vârful stivei calculatorului. Numărul trebuie să fie în limitele 00-FF.

2314 TSK-TO-A	CALL	2DD5,FP-TO-A	Modulul ultimei valori
	JP	C,24F9,REPORT-B	rotite în A, dacă este
			posibil; altfel se dă eroare.
	LD	C,+01	Se pune unu în C pentru
			ultima valoare pozitivă.
	RET	Z	Revenire dacă valoarea a fost
			pozitivă.
	LD	C,+FF	Altfel se schimbă în C +FF
	RET		(de fapt minus 1). Revenire.

THE CIRCLE COMMAND ROUTINE (RUTINA DE COMANDA CERC)

Această rutină reprezintă o aproximație a cercului cu centrul de coordonate X și Y și de rază Z. Aceste numere se rotunjesc la cel mai apropiat întreg înainte de a se utiliza. Acest Z trebuie să fie mai mic decît 87,5, chiar dacă (X,Y) sînt în centrul ecranului. Metoda utilizată este de a reprezenta o serie de arcuri approximate prin linii drepte. Ea este ilustrată într-un program BASIC dat în anexă. Notările din program sînt prezentate în continuare.

CIRCLE are patru părți:

- i. Se testează raza. Dacă ea este în modul mai mic decît 1, se marchează doar X,Y;
- ii. Se apelează CD-PRMS1 la 2470-24B6, care se utilizează pentru setarea parametrilor initiali atât pentru CIRCLE cît și pentru DRAW.;
- iii. Se setează parametrul rămasi ai lui CIRCLE, incluzînd și înlocuirea inițială pentru primul 'arc' (o linie dreaptă, de fapt);
- iv. Se sare în DRAW pentru utilizarea buclei reprezentare-arc la 2420-24FA.

Acum se explică pe rînd părțile i. la iii.

i. 2320-23AA. Raza, numită Z', este obținută din stiva calculatorului. Modulul ei Z este format și folosit de acum încolo. Dacă Z este mai mic decît 1, este sters din stivă și punctul X,Y este marcat printr-un salt în PLOT.

2320 CIRCLE	RST	0018,GET-CHAR	Se aduce prezentul caracter.
-------------	-----	---------------	------------------------------

CP	+2C	Test față de virgulă.
JP	NZ,1C8A,REPORT-C	Dacă nu este virgulă, se dă eroare.
RST	0020,NEXT-CHAR	Se aduce următorul caracter (raza).
CALL	1C82,EXPT-1NUM	Raza în stiva calculatorului.
CALL	1BEE,CHECK-END	Se trece la următoarea instrucțiune dacă se verifică sintaxa.
RST	0028,FP-CALC	Se folosește calculatorul:
DEFB	+2A,abs	stiva conține: X,Y,Z.
DEFB	+3B,rc-stack	Z este restocat; exponentul
DEFB	+38,end-calc	lui este accesibil.
LD	A,(HL)	se aduce exponentul razei.
CP	+81	Se testează dacă raza este
		mai mică decât 1.
JR	NC,233B,C-R-BRE-1	Dacă nu este, salt.
RST	0028,FP-CALC	Dacă este mai mică, se șterge
		din stivă.
DEFB	+02,delete	Stiva conține X,Y.
DEFB	+38,end-calc	
JR	22DC,PLOT	Se reprezintă punctul doar
		prin X,Y.

ii. 233B-2346 și apel CD-PRMS1. $2*PI$ este stocat în mem-5 și CD-PRMS1 este apelat. Această subrutină stochează în registrul B numărul de arce cerut pentru cer, și anume $A=4*INT(PI*SOR Z/4)+4$, deci 4,8,12..., pînă la maxim 32. De asemenea ea stochează în mem-0 la mem-4 cantitățile $2*PI/A$, $SIN(PI/A)$, 0, $COS(2*PI/A)$ și $SIN(2*PI/A)$.

233B C-R-BRE-1	RST	0028,FP-CLAC	
	DEFB	+A3,stk-pi/2	X,Y,Z,PI/2
	DEFB	+38,end-calc	Acum se crește exponentul la
	LD	(HL),+83	83 hex, schimbînd PI/2 în PI.
	RST	0028,FP-CALC	X,Y,Z,2*PI
	DEFB	+C5,st-mem-5	(2*PI este copiat în mem-5)
	DEFB	+02,delete	X,Y,Z
	DEFB	+38,end-calc	
	CALL	247D,CD-PRMS1	Se setează parametrii inițiali

iii. 2347-2381: parametrii rămași și saltul la DRAW. Se face un test pentru a vedea dacă lungimea 'arcului' inițial este mai mică decât 1. Dacă este, se face un salt simplu pentru reprezentarea X,Y. Altfel, parametrii sînt setați: X+Z și $Y-Z*SIN(PI/A)$ sînt stocați de două ori ca punct de început și de sfîrșit, și copiați în COORDS; în mem-1 și mem-2 se stochează zero și $2*Z*SIN(PI/A)$ ca și creșteri inițiale, dînd ca prim 'arc' linia verticală dreaptă împreună cu X+Z, $Y-Z*SIN(PI/A)$ și X+Z, $Y+Z*SIN(PI/A)$. Bucla trasare-arc din DRAW va asigura că toate punctele subsecvenței rămîn în același cerc ca aceste două puncte, cu unghiul incremental $2*PI/A$. Dar este clar că aceste două puncte de fapt subîntind acest unghi în punctul $X+Z*(1-COS(PI/A))$, Y și nu în X,Y. Deci punctul de sfîrșit al fiecărui arc al cercului este înlocuit la dreapta de $2*(1-COS(PI/A))$, care este mai mic decât o jumătate de pixel, și ajunge la cel mult un pixel.

2347	PUSH	BC	Salvare contor-arc în B.
	RST	0028,FP-CALC	X,Y,Z
	DEFB	+31,duplicate	X,Y,Z,Z
	DEFB	+E1,get-mem-1	X,Y,Z,Z,SIN(PI/A)
	DEFB	+04,multiply	X,Y,Z,Z*SIN(PI/A)
	DEFB	+38,end-calc	SIN(PI/A) este jumătate din
	LD	A,(HL)	lungimea 'arcului' inițial;
	CP	+80	este testat pentru a vedea
			dacă este mai mic de 0,5.
	JR	NC,235A,C-ARC-GE1	Dacă nu, se execută saltul.
	RST	0028,FP-CALC	Altfel, Z este sters din
	DEFB	+02,delete	stivă, de asemenea cu
	DEFB	+02,delete	jumătate de arc; stiva
	DEFB	+38,end-calc	calculatorului este stearsă;
			și salt pentru reprezentarea
			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) trece de acum
			în mem-2)
	DFEB	+01,exchange	X,Y,Z*SIN(PI/A),Z
	DEFB	+C0,st-mem-0	X,Y,Z*SIN(PI/A),Z
	DEFB	+02,delete	X,Y,Z*SIN(PI/A)
	DEFB	+03,subtract	X,Y-Z*SIN(PI/A)

DEFB	+01,exchange	Y-Z*SIN (PI/A),X
DEFB	+E0,get-mem-0	Y-Z*SIN (PI/A),X,Z
DEFB	+0F,addition	Y-Z*SIN (PI/A),X+Z
DEFB	+C0,st-mem-0	(X+Z este copiat în mem-0)
DEFB	+01,exchange	X+Z,Y-Z*SIN (PI/A)
DEFB	+31,duplicate	X+Z,Y-Z*SIN (PI/A),Y-Z*SIN (PI/A)
DEFB	+E0,get-mem-0	sa,sb,sb,sa
DEFB	+01,exchange	sa,sb,sa,sb
DEFB	+31,duplicate	sa,sb,sa,sb,sb
DEFB	+E0,get-mem-0	sa,sb,sa,sb,sb,sa
DEFB	+A0,stk-zero	sa,sb,sa,sb,sa,sb,0
DEFB	+C1,st-mem-1	(mem-1 este setat pe zero)
DEFB	+02,delete	sa,sb,sa,sb,sb,sa
DEFB	+38,end-calc	

(Aici cu sa s-a notat X+Z si cu sb s-a notat Y-Z*SIN (PI/A)).

INC	(mem-2-1st)	Incrementând octetul exponent al mem-2 se setează mem-2 cu $2*Z*SIN (PI/A)$.
CALL	1E94,FIND-INT1	Ultima valoare X+Z este mutată din stivă în A si se copiază în L.
LD	L,A	Salvare în HL.
PUSH	HL	Y-Z*SIN (PI/A) trece din stivă în A si este copiată în H.
CALL	1E94,FIND-INT1	
POP	HL	Acum HL contine punctul initial.
LD	H,A	EI este copiat în COORDS.
LD	(COORDS),HL	Se reface contro-arc.
POP	BC	Se face un salt la DRAW.
JP	2420,DRW-STEPS	

(stiva contine acum X+Z,Y-Z*SIN (PI/A),Y-Z*SIN (PI/A),X+Z)

THE DRAW COMMAND ROUTINE (RUTINA DE COMANDA DESENARE)

Această rutină este introdusă cu coordonatele punctului X_0, Y_0 în COORDS. Dacă se dau numai doi parametrii X, Y cu comanda DRAW, se va desena o aproximare la o linie dreaptă din punctul X_0, Y_0 la X_0+X, Y_0+Y . Dacă se dă un al treilea parametru G, se va desena o aproximare a unui arc de cerc de la X_0, Y_0 la X_0+X, Y_0+Y rotirea făcându-se în sens antiorar cu un unghi de G radiani.

Rutina are patru părți:

- i. Se desenează doar o linie dacă se dau numai 2 parametrii sau dacă diametrul cercului implicat este mai mic decât 1;
- ii. Se apelează CD-PRMS1 la 247D-24B6 pentru setarea primilor parametrii;
- iii. Se aranjează parametrii rămași, incluzând înlocuirea inițială a primului arc.
- iv. Se introduce bucla desenare-arc si se desenează arcul ca o serie de arce mici approximate prin linii drepte, apelând subrutina de desenare-linie la 24B7-24FA, cum este necesar.

Două subrutine, CD-PRMS1 si DRAW-LINE, urmează rutina principală. Cele patru părți de deasupra rutinei principale vor fi tratate pe rând.

i. Dacă sînt numai doi parametrii, se execută un salt la LINE-DRAW la 2477. O linie este de asemenea desenată dacă valoarea $Z=(ABS X+ABS Y)/ABS SIN (G/2)$ este mai mică decât 1. Z face legătura între 1 si 1,5 ori diametrul cercului implicat. În această secțiune mem-0 este setat cu $SIN (G/2)$, mem-1 cu Y, si mem-5 cu G.

2382 DRAW	RST	0018,GET-CHAR	Se aduce caracterul curent.
	CP	+2C	Dacă este virgulă,
	JR	Z,238D,DR-3-PRMS	atunci salt.
	CALL	1BEE,CHECK-END	Se trece la următoarea instrucție dacă se verifică sintaxa.
238D DR-3-PRMS	JP	2477,LINE-DRAW	Salt tocmai la desenare linie
	RST	0020,NEXT-CHAR	Se aduce următorul caracter (unghiul).
	CALL	1C82,EXPT-INUN	Unghiul se trece în stiva calculatorului.
	CALL	1BEE,CHECK-END	Se trece la următoarea instrucție dacă se verifică sintaxa.

	RST	0028,FP-CALC	X,Y,G sînt în stivă.
	DEFB	+C5,st-mem-5	(G este copiat în mem-5.)
	DEFB	+A2,stk-half	X,Y,G,0.5
	DEFB	+04,multiply	X,Y,G/2
	DEFB	+1F,sin	X,Y,SIN(G/2)
	DEFB	+31,duplicate	X,Y,SIN(G/2),SIN(G/2)
	DEFB	+30,not	X,Y,SIN(G/2),(0/1)
	DEFB	+30,not	X,Y,SIN(G/2),(1/0)
	DEFB	+00,jump-true	X,Y,SIN(G/2)
	DEFB	+06,to DR-SIN-NZ	(Dacă SIN(G/2)=0, atunci
	DEFB	+02,delete	G=2*N*PI desenează o linie
			dreaptă.)
	DEFB	+38,end-calc	X,Y
23A3 DR-SIN-NZ	JP	2477,LINE-DRAW	Linia X0,Y0 la X0+X,Y0+Y
	DEFB	+C0,st-mem-0	(SIN G/2 este copiat în
			mem-0)
	DEFB	+02,delete	X,Y sînt acum în stivă.
	DEFB	+C1,st-mem-1	(Y este copiat în mem-1)
	DEFB	+02,delete	X
	DEFB	+31,duplicate	X,X
	DEFB	+2A,abs	X,X' (X' = ABS X)
	DEFB	+E1,get-mem-1	X,X',Y
	DEFB	+01,exchange	X,Y,X'
	DEFB	+E1,get-mem-1	X,Y,X',Y
	DEFB	+2A,abs	X,Y,X',Y' (Y' = ABS Y)
	DEFB	+0F,addition	X,Y,X'+Y'
	DEFB	+E0,get-mem-0	X,Y,X'+Y',SIN(G/2)
	DEFB	+05,division	X,Y,(X'+Y')/SIN(G/2) = Z'
	DEFB	+2A,abs	X,Y,Z (Z = ABS Z')
	DEFB	+e0,get-mem-0	X,Y,Z,SIN(G/2)
	DEFB	+01,exchange	X,Y,SIN(G/2),Z
	DEFB	+3D,re-stack	(Z este restocat pentru a
	DEFB	+38,end-calc	asigura că exponentul lui
			este accesibil)
	LD	A,(HL)	Se aduce exponentul lui Z.
	CP	+81	Dacă Z este mai mare sau egal
	JR	NC,23C1,DR-PRMS	cu 1, atunci salt.
	RST	0028,FP-CALC	X,Y,SIN(G/2),Z
	DEFB	+02,delete	X,Y,SIN(G/2)
	DEFB	+02,delete	X,Y
	DEFB	+38,end-calc	se desenează linia de la
	JP	2477,LINE-DRAW	X0,Y0 la X0+X,Y0+Y.

ii. Doar se apelează CD-PRMS1. Această subrutină salvează în registrul B numărul celui mai mic arc cerut pentru arcul complet, și anume $A=4*INT(G'*SOR Z/8)+4$, unde $G' = \text{mod } G$, sau 252 dacă expresia depășește 252 (cum se poate întâmpla cu o coardă mare și un unghi mic). Așa că A este 4, 8, 12 ..., pînă la 252. De asemenea subrutina stochează în mem-0 la mem-4 cantitățile G/A , $SIN(G/2*A)$, $SIN(G/A)$.

23C1 DR-PRMS CALL 247D,CD-PRMS1 Subrutina este apelată.

iii. Aranjarea restului parametrilor, după cum urmează. Stiva va conține aceste patru informații, preluate din vîrf: $X0+X$ și $Y0+Y$ pentru sfîrșitul ultimului arc; apoi $X0$ și $Y0$ pentru începutul primului arc. Mem-0 va conține $X0$ și mem-5 va conține $Y0$. Mem-1 și mem-2 vor conține înlocuirile initiale pentru primul arc, U și V; iar mem-3 și mem-4 vor conține $COS(G/A)$ și $SIN(G/A)$ pentru utilizarea în bucla de desenare arc.

Formulele pentru U și V pot fi explicate în modul următor. În locul deplasării de-a lungul coardei finale, de lungime L, denumită, cu înlocuirile X și Y, se dorește deplasarea de-a lungul unei coarde initiale (care poate fi mai lungă) de lungime L*M, unde $M=SIN(G/2*A)/SIN(G/2)$, cu deplasările $X*M$ și $Y*M$, dar trecînd prin unghiul $-(G/2 - G/2*A)$, deci cu adevăratele înlocuiris:

$$U = Y*M*SIN(G/2 - G/2*A) + X*M*COS(G/2 - G/2*A)$$

$$V = Y*M*COS(G/2 - G/2*A) - X*M*SIN(G/2 - G/2*A)$$

Aceste formule pot fi verificate printr-o diagramă, folosind întiderile normale ale lui $COS(P-Q)$ și $SIN(P-Q)$, unde $Q=G/2 - G/2*A$.

23C4	PUSH	BC	Salvare contor-arc în B.
	RST	0028,FP-CALC	X,Y,SIN(G/2),Z
	DEFB	+02,delete	X,Y,SIN(G/2)
	DEFB	+E1,get-mem-1	X,Y,SIN(G/2)SIN(G/2*A)
	DEFB	+01,exchange	X,Y,SIN(G/2*A),SIN(G/2)
	DEFB	+05,division	X,Y,SIN(G/2*A)/SIN(G/2)=W
	DEFB	+C1,st-mem-1	(W este copiat în mem-1)
	DEFB	+02,delete	X,Y
	DEFB	+01,exchange	Y,X
	DEFB	+31,duplicate	Y,X,X

DEFB	+E1, get-mem-1	Y, X, X, W
DEFB	+04, multiply	Y, X, X*W
DEFB	+C2, st-mem-2	(X*W este copiat în mem-2)
DEFB	+02, delete	Y, X
DEFR	+01, exchange	X, Y
DEFB	+31, duplicate	X, Y, Y
DEFB	+E1, get-mem-1	X, Y, Y, W
DEFB	+04, multiply	X, Y, Y*W
DEFB	+E2, get-mem-2	X, Y, Y*W, X*W
DEFB	+E5, get-mem-5	X, Y, Y*W, X*W, G
DEFR	+E0, get-mem-0	X, Y, Y*W, X*W, B, B/A
DEFB	+03, subtract	X, Y, Y*W, X*W, B-G/A
DEFB	+A2, stk-half	X, Y, Y*W, X*W, B, B/A, 1/2
DEFB	+04, multiply	X, Y, Y*W, X*W, B/2, G/2*A=F
DEFR	+31, duplicate	X, Y, Y*W, X*W, F, F
DEFB	1F, sin	X, Y, Y*W, X*W, F, SIN F
DEFR	+C5, st-mem-5	(SIN F este copiat în mem-5)
DEFB	+02, delete	X, Y, Y*W, X*W, F
DEFR	+20, cos	X, Y, Y*W, X*W, COS F
DEFB	+C0, st-mem-0	(COS F este copiat în mem-0)
DEFB	+02, delete	X, Y, Y*W, X*W
DEFB	+C2, st-mem-2	(X*W este copiat în mem-2)
DEFB	+02, delete	X, Y, Y*W
DEFB	+C1, st-mem-1	(Y*W este copiat în mem-1)
DEFR	+E5, get-mem-5	X, Y, Y*W, SIN F
DEFB	+04, multiply	X, Y, Y*W*SIN F
DEFR	+E0, get-mem-0	X, Y, Y*W*SIN F, X*W
DEFB	+E2, get-mem-2	X, Y, Y*W, SIN F, X*W, COS F
DEFB	+04, multiply	X, Y, Y*W*SIN F, X*W*COS F
DEFB	+0F, addition	X, Y, Y*W*SIN F+X*W*COS F = U
DEFR	+E1, get-mem-1	X, Y, U, Y*W
DEFB	+01, exchange	X, Y, Y*W, U
DEFR	+C1, st-mem-1	(U este copiat în mem-1)
DEFB	+02, delete	X, Y, Y*W
DEFR	+E0, get-mem-0	X, Y, Y*W, COS F
DEFB	+04, multiply	X, Y, Y*W*COS F
DEFR	+E2, get-mem-2	X, Y, Y*W*COS F, X*W
DEFB	+E5, get-mem-5	X, Y, Y*W*COS F, X*W, SIN F
DEFB	+04, multiply	X, Y, Y*W*COS F, X*W*SIN F
DEFB	+03, subtract	X, Y, Y*W*COS F-X*W*SIN F = V
DEFB	+C2, st-mem-2	(V este copiat în mem-2)
DEFB	+2A, abs	X, Y, V' (V' = ABS V)
DEFB	+E1, get-mem-1	X, Y, V', U
DEFB	+2A, abs	X, Y, V', U' (U' = ABS U)
DEFR	+0F, addition	X, Y, U'+V'
DEFB	+02, delete	X, Y
DEFR	+38, end-calc	(DE indică acum pe U'+V')
LD	A, (DE)	Se încarcă exponențial pentru U'+V'
CP	+81	Dacă U'+V' este mai mic decât
POP	BC	1, doar se curăță stiva și se
JP	C, 2477, LINE-DRAW	desenează linia de la X0, Y0
		la X0+X, Y0+Y.
PUSH	BC	Altfel se continuă cu
RST	0028, FP-CALC	parametrii: X, Y în stivă.
DEFB	+01, exchange	Y, X
DEFR	+38, end-calc	
LD	A, (COORDS-lo)	Se aduce X0 în A și astfel în
CALL	2028, STACK-A	stivă.
RST	0028, FP-CALC	Y, X, X0
DEFB	+C0, st-mem-0	(X0 este copiat în mem-0)
DEFB	+0F, addition	Y, X0+X
DEFB	+01, exchange	X0+X, Y
DEFB	+38, end-calc	
LD	A, (COORDS-hi)	Se aduce Y0 în A și astfel în
CALL	2028, STACK-A	stivă.
RST	0028, FP-CALC	X0+X, Y, Y0
DEFB	+C5, st-mem-5	(Y0 este copiat în mem-5)
DEFB	+0F, addition	X0+X, Y0+Y
DEFB	+E0, get-mem-0	X0+X, Y0+Y, X0
DEFB	+E5, get-mem-5	X0+X, Y0+Y, X0, Y0
DEFB	+38, end-calc	
POP	BC	Se reface contorul-arc în B.

iv. Bucla desenare arc. Este introdusă la 2439 cu coordonatele punctului de început în vârful stivei, și cu înlocuirile inițiale pentru primul arc în mem-1 și mem-2. Se folosește trigonometria simplă pentru asigurarea faptului că toate arcele subsecvenței vor fi desenate în punctele care le leagă în același cerc ca primele două, subînținzând același unghi la centru. Se poate arăta că

dacă 2 puncte $X1, Y1$ și $X2, Y2$ sînt legate pe un cerc și subîntînd un unghi la centru N , care este și originea coordonatelor, atunci $X2 = X1 \cdot \cos N - Y1 \cdot \sin N$, și $Y2 = X1 \cdot \sin N + Y1 \cdot \cos N$. Dar din cauza faptului că originea este aici în colțul de jos din partea dreaptă a ecranului, bucla de desenare-arc aplică aici relații de incrementare, denumind $Un = Xn + 1, Yn$ și $Vn = Yn + 1, Xn$, care realizează rezultatul dorit. Stiva arată după $(n+1)$ pași prin buclă, că Xn și Yn sînt incrementați cu Un și Vn , după acestea fiind obținuți din $Un-1$ și $Vn-1$. Cele 4 valori din vârful stivei la 2425 sînt, în DRAW, citind în sus, $X0+X, Y0+Y, Xn$ și Yn , dar pentru salvarea spațiului acestea nu sînt arătate pînă la 2439. Pentru valorile inițiale în CIRCLE, vezi sfîrșitul lui CIRCLE, mai sus. De asemenea în CIRCLE unghiul θ trebuie luat $2 \cdot \pi$.

2420	DRW-STEPS	DEC	B	B numără trecerile prin buclă
		JR	2,245F,ARC-END	Salt cînd B ajunge la zero.
		JR	2439,ARC-START	Salt la începutul buclei.
2425	ARC-LOOP	RST	0028,FP-CALC	(Vezi textul mai sus de stivă)
		DEFB	+E1,get-mem-1	Un-1
		DEFB	+31,duplicate	Un-1,Un-1
		DEFB	+E3,get-mem-3	Un-1,Un-1,COS(G/A)
		DEFB	+04,multiply	Un-1,Un-1 * COS(G/A)
		DEFB	+E2,get-mem-2	Un-1,Un-1 * COS(G/A),Vn-1
		DEFB	+E4,get-mem-4	Un-1,Un-1 * COS(G/A),Vn-1, SIN(G/A)
		DEFB	+04,multiply	Un-1,Un-1 * COS(G/A),Vn-1 * SIN(G/A)
		DEFB	+03,subtract	Un-1,Un-1 * COS(G/A) - Vn-1 * SIN(G/A) = Un
		DEFB	+C1,st-mem-1	(Un este copiat în mem-1)
		DEFB	+02,delete	Un-1
		DEFB	+E4,get-mem-4	Un-1,SIN(G/A)
		DEFB	+04,multiply	Un-1 * SIN(G/A)
		DEFB	+E2,get-mem-2	Un-1 * COS(G/A),Vn-1
		DEFB	+E3,get-mem-3	Un-1 * SIN(G/A),Vn-1,COS(G/A)
		DEFB	+04,multiply	Un-1 * SIN(G/A),Vn-1 * COS(G/A)
		DEFB	+0F,addition	Un-1 * SIN(G/A) + Vn-1 * COS(G/A) = Vn
		DEFB	+C2,st-mem-2	(Vn este copiat în mem-2)
		DEFB	+02,delete	(Cum s-a notat și în text, stiva conține de fapt $X0+X, Y0+Y, Xn$ și Yn)
		DEFB	+38,end-calc	Salvare contor-arc.
2439	ARC-START	PUSH	BC	$X0+X, Y0+Y$
		RST	0028,FP-CALC	(Yn este copiat în mem-0)
		DEFB	+C0,st-mem-0	$X0+X, Y0+Y, Xn$
		DEFB	+02,delete	$X0+X, Y0+Y, Xn, Un$
		DEFB	+E1,get-mem-1	$X0+X, Y0+Y, Xn+1, Xn+1$
		DEFB	+0F,addition	$X0+X, Y0+Y, Xn+1, Xn+1$
		DEFB	+31,duplicate	Urmează Xn' , valoarea aproximată a lui Xn obținută prin subrutina de desenare-linie, care este copiată în A
		DEFB	+38,end-calc	A
		LD	A,(COORDS-lo)	și de aici în stivă.
		CALL	2028,STACK-A	$X0+X, Y0+Y, Xn+1, Xn'$
		RST	0028,FP-CALC	$X0+X, Y0+Y, Xn+1, Xn+1, Xn' - Xn = Un'$
		DEFB	+03,subtract	Un'
		DEFB	+E0,get-mem-0	$X0+X, Y0+Y, X+1, Un', Yn$
		DEFB	+E2,get-mem-2	$X0+X, Y0+Y, Xn+1, Un', Yn, Vn$
		DEFB	+0F,addition	$X0+X, Y0+Y, Xn+1, Un', Yn+Vn = Yn+1$
		DEFB	+C0,st-mem-0	($Yn+1$ este copiat în mem-0)
		DEFB	+01,exchange	$X0+X, Y0+Y, Xn+1, Yn+1, Un'$
		DEFB	+E0,get-mem-0	$X0+X, Y0+Y, Xn+1, Yn+1, Un', Yn+1$
		DEFB	+38,end-calc	
		LD	A,(COORDS-hi)	Yn' , aproximată ca și Xn' , este copiat în A și de aici în stivă.
		CALL	2028,STACK-A	
		RST	0028,FP-CALC	$X0+X, Y0+Y, Xn+1, Yn+1, Un', Yn+1, Yn'$
		DEFB	+03,subtract	$X0+X, Y0+Y, Xn+1, Yn+1, Un', Vn'$
		DEFB	+38,end-calc	
		CALL	2437,DRAW-LINE	Este desenat următorul 'arc'.
		POP	BC	Restituire contor-arc.
		DJNZ	2425,ARC-LOOP	Salt dacă trebuie desenate mai multe arcuri.
245F	ARC-END	RST	0028,FP-CALC	Coordonatele sfîrșitului ultimului arc care a fost desenat sînt acum sterse din stivă.
		DEFB	+02,delete	
		DEFB	+02,delete	

	DEFB +01,exchange	Y0+Y,X0+X
	DEFB +38,end-calc	
	LD A,(COORDS-lo)	Coordonata X a sfirsitului
	CALL 2028,STACK-A	ultimului arc care a fost
	RST 0028,FP-CALC	desenat, numit Xz', se
		copiază în stivă.
	DEFB +03,subtract	Y0+Y,X0+X-Xz'
	DEFB +01,exchange	X0+X-Xz',Y0+Y
	DEFB +38,end-calc	
	LD A,(COORDS-hi)	S-a obtinut coordonata Y.
	CALL 2028,STACK-A	
	RST 0028,FP-CALC	X0+X-Xz',Y0+Y,Yz'
	DEFB +03,subtract	X0+X-Xz',Y0+Y-Yz'
	DEFB +38,end-calc	
2477 LINE-DRAW	CALL 2487, DRAW-LINE	Arcul final este desenat pînă
		ajunge la X0+X,Y0+Y (sau să
		închidă cercul).
	JP 004D,TEMPS	Iesire, setînd culorile
		temporare.

THE 'INITIAL PARAMETERS' SUBROUTINE (SUBROUTINA 'PARAMETRII INITIALI')

Această subrutină este apelată și de CIRCLE și de DRAW pentru a seta parametrii lor initiali. Ea este apelată de CIRCLE cu X, Y și raza Z în vârful stivei, citind partea de sus. Ea este apelată de DRAW cu ai săi X, Y, SIN (G/2) și Z, cum au fost definiți în DRAW i, deasupra, în vârful stivei. În cele ce urmează stiva este arătată doar de la Z în sus.

Subrutina readuce în B contorul arc, A cum a fost explicat și la începutul lui DRAW și la începutul lui CIRCLE și în mem-0 la mem-5 cantitățile G/A, SIN (G/2*A), 0, COS (G/A), SIN (G/A) și G. Pentru un cerc, G trebuie luat egal cu 2*PI.

247D CD-PRMS1	RST 0028,FP-CALC	Z
	DEFB +31,duplicate	Z,Z
	DEFB +28,sqr	Z,SQR Z
	DEFB +34,stk-data	Z,SQR Z,2
	DEFB +32,exponent+82	
	DEFB +00,(+00,+00,+00)	
	DEFB +01,exchange	Z,2,SQR Z
	DEFB +05,division	Z,2/SQR Z
	DEFB +E5,get-mem-5	Z,2/SQR Z,6
	DEFB +01,exchange	Z,G,2/SQR Z
	DEFB +05,division	Z,G*SQR Z/2
	DEFB +2A,abs	Z,G*SQR Z/2 (G' = MOD G)
	DEFB +38,end-calc	Z,G'*SQR Z/2 = A1
	CALL 2DD5,FP-TO-A	A1 la A din stivă, dacă este
		posibil.
	JR C,2495,USE-252	Dacă A1 ciclează de 256 ori
		sau mai mult, se utilizează
		256.
	AND +FC	4*INT (A1/4) la A.
	ADD A,+04	Se adună 4, dînd contorul-arc
		A.
2495 USE-252	JR NC,2497, DRAW-SAVE	Salt dacă încă este sub 256.
	LD A,+FC	Aici, doar utilizarea 256
		zecimal.
	PUSH AF	Acum salvare contor-arc.
	CALL 2028,STACK-A	Copierea lui în stiva
		calculatorului.
	RST 0028,FP-CALC	Z,A
	DEFB +E5,get-mem-5	Z,A,G
	DEFB +01,exchange	Z,G,A
	DEFB +05,division	Z,G/A
	DEFB +31,duplicate	Z,G/A,G/A
	DEFB +1F,sin	Z,G/A,SIN (G/A)
	DEFB +C4,st-mem-4	(SIN (G/A) se copiază în
		mem-4)
	DEFB +02,delete	Z,G/A
	DEFB +31,duplicate	Z,G/A,G/A
	DEFB +A2,stk-half	Z,G/A,G/A,0.5
	DEFB +04,multiply	Z,G/A,G/2*A
	DEFB +1F,sin	Z,G/A,SIN (G/2*A)
	DEFB +C1,st-mem-1	(SIN (G/2*A) se copiază în
		mem-1)
	DEFB +01,exchange	Z,SIN (G/2*A),G/A
	DEFB +C0,st-mem-0	(G/A se copiază în mem-0)
	DEFB +02,delete	Z,SIN (G/2*A) = S
	DEFB +31,duplicate	Z,S,S
	DEFB +04,multiply	Z,S*S
	DEFB +31,duplicate	Z,S*S,S*S

DEFB	+0F, addition	Z, 2*S*S
DEFB	+A1, stk-one	Z, 2*S*S, 1
DEFB	+03, subtract	Z, 2*S*S-1
DEFB	+1B, negate	Z, 1-2*S*S = COS (G/A)
DEFB	+C3, st-mem-3	(COS (G/A) se copiază în mem-4)
DEFB	+02, delete	Z
DEFB	+3B, end-calc	
POP	BC	Readucerea contorului-arc în B.
RET		Terminare.

THE LINE-DRAWING SUBROUTINE (SUBRUTINA TRASARE LINIE)

Această subrutină este apelată de DRAW pentru a desena o aproximare la o linie dreaptă din punctul X0, Y0 conținut în COORDS la punctul X0+X, Y0+Y, unde incrementele X și Y sînt în vârful stivei calculatorului. Subrutina a fost proiectată inițial pentru ZX80 și ZX81 8K ROM, și este descrisă de un program BASIC la pagina 121 din manualul ZX81. De asemenea este ilustrată și aici în programul Circle din anexă.

Metoda este de a interptrunde atîti pasi orizontal sau vertical cîți sînt necesari de-a lungul setului de bază al pasilor diagonali, folosind un algoritm care asează pasii orizontal sau vertical, după cum este posibil.

24B7 DRAW-LINE	CALL	2307, STK-TO-BC	ABS Y în B; ABS X în C; SGN Y în D; SGN X în C.
	LD	A, C	salt dacă ABS X este mai mare
	CP	B	sau egal cu ABS Y, astfel
	JR	NC, 24C4, DL-X-GE-Y	încît cel mai mic merge în L,
	LD	L, C	și cel mai mare merge în H.
	PUSH	DE	Salvare pas diagonal (±1, ±1)
			în DE.
	XOR	A	Inserare pas vertical (±1, 0)
	LD	E, A	în DE (D conține SGN Y).
24C4 DL-X-GE-Y	JR	24CB, DL-LARGER	Acum salt pentru a seta H.
	OR	C	Revenire dacă și ABS X și
	RET	Z	ABS Y sînt ambii zero.
	LD	L, B	Cel mai mic (aici ABS Y)
			trece în L.
	LD	B, C	În acest caz, ABS X trece în
			B din H.
	PUSH	DE	Salvare pas diagonal.
24CB DL-LARGER	LD	D, +00	Pas orizontal (0, ±1) în DE.
	LD	H, B	Cel mai mare dintre ABS X și
			ABS Y trece acum în H.

Algoritmul începe aici. Cel mai mare dintre ABS X și ABS Y, numit H, este pus în A și redus la INT (H/2). Pasii H-L orizontali sau verticali și pasii diagonali L sînt făcuți (unde L este mai mic decît ABS X și ABS Y) în modul următor: L este adunat la A; Dacă acum A este mai mare sau egal cu H, el este redus cu H și se face un pas diagonal; altfel se face un pas vertical sau orizontal. Aceasta se repetă de H ori (B conține pe H). De notat faptul că în același timp schimbarea registrilor H' și L' este utilizată pentru a păstra COORDS.

	LD	A, B	B în A la fel de bine ca și în H.
24CE D-L-LOOP	RRA		A începe la INT (H/2).
	ADD	A, L	L este adunat la A.
	JR	C, 24D4, D-L-DIAG	Dacă este 256 sau mai mult,
			salt la pas diagonal.
	CP	H	Dacă A este mai mic decît H,
	JR	C, 24DB, D-L-HR-VT	salt pentru pas orizontal sau
24D4 D-L-DIAG			vertical.
	SUB	H	Se reduce A cu H.
	LD	C, A	Se reface în C.
	EXX		Acum se utilizează schimbarea
			registrilor.
	POP	BC	Pas diagonal la B'C'.
	PUSH	BC	Și acesta este salvat.
24DB D-L-HR-VT	JR	24DF, D-L-STEP	Salt pentru a executa pasul.
	LD	C, A	Se salvează A (neredus) în C.
	PUSH	DE	Se trece în stivă pentru
			scurt timp.
	EXX		Se interschimbă registrii.
24DF D-L-STEEP	POP	BC	Se trece în B'C' acum.
	LD	HL, (COORDS)	Acum se face pasul: mai întîi
început.			COORDS în H'L' ca pun
	LD	A, B	Pas Y de la B' la A.

	ADD	A,H	Se adună în H'.
	LD	B,A	Rezultatul în B.
	LD	A,C	Acum pas X; îi va fi testat
	INC	A	rangul (Y trebuie testat în
			PLOT).
	ADD	A,L	Se adună L' la C' în A. Salt
	JR	C,24F7,D-L-RANGE	la transport pentru a testa
			mai departe.
	JR	Z,24F9,REPORT-B	Zero după 'nu este transport'
			denotă poziția X-1, afară
			din rang.
24EC D-L-PLOT	DEC	A	Se reface valoarea adevărată
			a lui A.
	LD	C,A	Valoarea pentru reprezentarea
			grafică în C'.
	CALL	22E5,PLOT-SUB	Reprezentare pas.
	EXX		Se refac principalii registri
	LD	A,C	C înapoi în A pentru a
			continua algoritmul.
	DJNZ	24CE,D-L-LOOP	Salt înapoi pentru B pasi (H
			pasi).
	POP	DE	Se sterge stiva
			calculatorului.
	RET		Terminare.
24F7 D-L-RANGE	JR	24EC,D-L-PLOT	Zero după transport denotă
			poziția X 255, în rang.
Prezentarea B - Intreg afară din domeniu.			
24F9 REPORT-B	RST	0008,ERROR-1	Se apelează rutina de tratare
	DEFB	+0A	eroare.

EXPRESSION EVALUTION

THE 'SCANNING' SUBROUTINE

Această subrutină este utilizată pentru a realiza evaluarea rezultatului 'următoarei expresii'.

Rezultatul este returnat ca 'ultima valoare' în stiva calculatorului. Pentru un rezultat numeric, ultima valoare va fi numărul actual în virgulă mobilă. După cum, pentru un rezultat sir ultima valoare va consta dintr-un set de parametrii. Primul din cinci octeti este nespecificat, al doilea și al treilea octet contin adresa de început a sirului și al patrulea și al cincilea octet contin lungimea sirului.

Bitul 6 al FLASS este setat în cazul unui rezultat numeric și resetat pentru un rezultat sir.

Când o următoare expresie constă dintr-un singur operand, de exemplu simpla valoare obținută prin evaluarea operandului.

După cum, când următoarea expresie conține o funcție și un operand, de exemplu ...CHR\$ A...,...NOT A...,SIN I..., codul operației funcției este stocat în stiva mașinii pînă cînd se calculează ultima valoare a operandului. Această ultimă valoare este apoi supusă unei operații corespunzătoare pentru a da o nouă ultimă valoare.

În cazul în care trebuie efectuată o operație logică sau aritmetică, de exemplu ...A+B...,A*B...,...A=B..., atunci ambele valori ultime ale primului argument și ale codului operației trebuie păstrate pînă se găsește ultima valoare a celui de al doilea argument. Într-adevăr, calculul ultimei valori a celui de al doilea argument poate de asemenea să includă stocarea ultimelor valori și a codurilor operației pînă cînd s-a terminat calculul.

De aceea se poate arăta că atunci cînd se evaluează o expresie complexă, ce de exemplu ...CHR\$ (+A-26*INT ((T+A)/26)+65)..., trebuie realizată o ierarhie a operațiilor care mai sînt de prelucrat pînă cînd se ajunge în punctul din care se poate renunța la ierarhie pentru a găsi în final ultima valoare.

Fiecare cod de operație are asociat un cod corespunzător al priorității și operațiile de prioritate superioară se execută întotdeauna înaintea celor de prioritate inferioară.

Subrutina începe cu registrul A setat astfel încît să contină primul caracter al expresiei și marcatorul priorității de început - zero - încărcat în stiva mașinii.

24FB SCANNING	RST	0018,GET-CHAR	Se aduce primul caracter.
	LD	B,+00	Marcatorul priorității de început este stocat.
	PUSH	BC	
24FF S-LOOP-1	LD	C,A	Punctul principal de intrare.
	LD	HL,+2596	Index în tabelul de baleiere
	CALL	160C,INDEXER	funcției cu codul în C.
	LD	A,C	Se reface codul în A.
	JP	NC,2684,S-ALPHNUM	Salt dacă codul nu s-a găsit în tabel.
	LD	B,+00	Se folosește intrarea găsită în tabel pentru a construi adresa cerută în HL și salt la aceasta.
	LD	C,(HL)	
	ADD	HL,BC	
	JP	(HL)	

Urmază patru subrutine; ele sînt apelate de rutine din tabelul de baleiere funcție. Prima, 'subrutina de baleiere a limitelor', este folosită de S-QUOTE pentru a verifica dacă limita fiecărui sir coincide cu cealaltă.

250F S-QUOTE-S	CALL	0074,CH-ADD+1	Se indică următorul caracter.
	INC	BC	Se incrementează cu 1 contorul lungimii.
	CP	+00	Este un 'carriage return'?
	JP	Z,1C8A,REPORT-C	Prezentare eroare dacă este așa.
	CP	+22	Este un alt ' ' '?
	JR	NZ,250F,S-QUOTE-S	Salt înapoi dacă nu este.
	CALL	0074,CH-ADD+1	Se indică următorul caracter;
	CP	+22	se setează fanionul zero dacă este un alt ' ' '.
	RET		Sfîrșit.

Următoarea subrutină, subrutina 'baleiere: două coordonate', este apelată de S-SCREEN\$, S-ATTR și S-POINT pentru a se asigura că cele două coordonate cerute sînt date în forma lor specifică.

2522 S-2-COORD	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	CP	+28	Este ' ' '?
	JR	NZ,252D,S-RPORT-C	Prezentare eroare dacă nu este.
	CALL	1C79,NEXT-2NUM	Coordonatele în stiva calculatorului.
	RST	0018,GET-CHAR	Se aduce caracterul curent.

252D S-RPORT-C	CF JP	+29 NZ,1C8A,REPORT-C	Este ') ' ? Prezentare eroare dacă nu este.
----------------	----------	-------------------------	---

THE 'SYNTAX-Z' SUBROUTINE (SUBROUTINA 'SYNTAX-Z')
In acest punct se interpolează subrutina 'SYNTAX-Z'. Este apelată de 32 de ori, cu salvarea doar a unui octet la fiecare apelare. Un simplu test al bitului 7 al FLAGS va da resetarea fanionul zero de-a lungul executiei si setarea în timpul verificării sintaxei.

2530 SYNTAX-Z	BIT RET	7, (FLAGS)	Testare bit 7 al FLAGS. Sfîrsit.
---------------	------------	------------	-------------------------------------

Următoarea subrutină este 'subrutina baleiere SCREEN\$', care este folosită de S-SCREEN\$ pentru a găsi caracterul ce apare pe ecran la linia x, coloana y. Aceasta doar caută caracterul setat 'indicat' în CHARS.

Notă: Acesta este în mod normal unul din caracterele +20 (spatiu) la +7F (@), altfel utilizatorul poate modifica CHARS egalînd cu alte caractere, incluzînd caracterele grafice definite de utilizator.

2535 S-SCRN%-S	CALL LD LD ADD	2307,STK-TD-BC HL,(CHARS) DE,+0100 HL,DE	x în C; y în B; 0 ≤ x ≤ 23 zecimal; 0 ≤ y ≤ 31 zecimal CHARS plus 256 zecimal face ca HL să indice caracterul setat.
	LD RRCA RRCA	A,C	x se copiază în A. Numărul 32 (zecimal) * (x mod 8) + y este format în A si copiat în E.
	RRCA AND XOR LD LD AND XOR	+E0 B E,A A,C +18 +40	Acesta este octetul inferior al adresei ecran cerute.
	LD	D,A	Se copiază din nou x în A. Acum numărul 64 (zecimal) + 8*INT(x/8) este introdus în D.
25AF S-SCRN-LP	LD PUSH PUSH PUSH	B,+60 BC DE HL	DE contine acum adresa ecranului. B numără cele 96 de caractere Salvare contor. Salvare indicator ecran. Salvare indicator caracter setat.
	LD	A,(DE)	Se aduce primul rînd al caracterului ecran.
	XOR	(HL)	Se compară cu rîndul caracterului setat.
	JR	Z,255A,S-SC-MATCH	Salt dacă s-a găsit egalitate directă.
	INC	A	Acum se face test egalitate cu caracterul invers (se pune +00 în A în loc de +FF).
	JR	NZ,2573,S-SCR-NXT	Salt dacă nu s-a găsit nici o egalitate.
255A S-SC-MATCH	DEC LD	A C,A	Se reface +FF în A. Inversare stare (+00 sau +FF) în C.
255D S-SC-ROWS	LD INC	B,+07 D	B numără celelalte 7 rînduri. Se mută DE în celălalt rînd (se adună 256 zecimal).
	INC	HL	Se mută HL în rîndul următor (aceasta înseamnă următorul octet).
	LD XOR XOR	A,(DE) (HL) C	Se aduce rîndul de ecran. Se compară cu rîndul din ROM. Se include starea inversă.
	JR	NZ,2573,S-SCR-NXT	Salt dacă rîndul reuseste egalarea.
	DJNZ	255D,S-SC-ROWS	Salt pînă s-au dat toate rîndurile.
	POP	BC	Se înlătură indicatorul caracterului setat.
	POP	BC	Si indicatorul ecran.
	POP	BC	Contor final în BC.
	LD	A,+80	Ultimul cod caracter este setat plus unu.
	SUB	B	Acum A contine codul cerut.

	LD	BC,+0001	Acum este nevoie de un spatiu în spatiul de lucru.
	RST	0030,BC-SPACES	Se face spatiul.
	LD	(DE),A	Se pune caracterul în el.
	JR	257D,S-SCR-STO	Salt pentru stocarea caracterului.
2573 S-SCR-NXT	POP	HL	Restituire indicator caracter setat.
	LD	DE,+0008	Se trece cu 8 octeti, la următorul caracter din set.
	ADD	HL,DE	Restituire indicator ecran.
	POP	DE	Si contor.
	POP	BC	Ciclare înapoi pentru cele 96 de caractere.
	DJNZ	254F,S-SCRN-LP	Se încarcă sir gol (lungime zero).
	LD	C,B	Salt pentru stocarea caracterului egal, sau a sirului nul dacă nu s-a găsit nici o egalitate.
257D S-SCR-STO	JP	2A32,STK-STO-\$	

Notă: Această ieșire, prin STK-STO-\$, este o gresală și conduce la 'stocarea dublă' a sirului rezultat (vezi S-STRING, 25D3). Linia de instrucție trebuie să fie 'RET'.

Ultima din aceste patru subrutine este 'subrutina baleiere atribuite'. Ea este apelată de S-ATTR pentru a returna valoarea lui ATTR (x,y) care codifică atributele liniei x, coloanei y pe ecranul televizorului.

2580 S-ATTR-S	CALL	2307,STK-TO-BC	x în C, y în B. Iarăși, $0 \leq x \leq 23$ zecimal, $0 \leq y \leq 31$ zecimal.
	LD	A,C	
	RRCA		Se copiază x în A și numărul $32 \text{ (zecimal)} * x \pmod{8} + y$ este format în A și copiat în E.
	RRCA		$32 * x \pmod{8} + \text{INT}(x/8)$ este de asemenea copiat în C.
	LD	C,A	
	AND	+E0	
	XOR	B	
	LD	L,A	L conține octetul inferior al adresei atributului.
	LD	A,C	Se copiază în A $32 * x \pmod{8} + \text{INT}(x/8)$.
	AND	+03	$88 \text{ (zecimal)} + \text{INT}(x/8)$ este format în A și copiat în H.
	XOR	+58	H conține octetul superior al adresei atributului.
	LD	H,A	Octetul atribut este copiat în A.
	LD	A,(HL)	Ieșire, stocând octetul cerut
	JP	2D28,STACK-A	

THE SCANNING FUNCTION TABLE (TABELUL FUNCTIEI DE BALEIERE)

Acest tabel conține 8 funcții și 4 operatori. Acesta încorporează 5 noi funcții Spectrum și prevede o cale curată de accesare a unor funcții și operatori care deja există în ZX81.

locatia	cod	depla- sament	nume	adresa rutinei tratate
2596	22	1C	S-QUOTE	25B3
2598	28	4F	S-BRACKET	25E8
259A	2E	F2	S-DECIMAL	268D
259C	2B	12	S-U-PLUS	25AF
259E	A8	56	S-FN	25F5
25A0	A5	57	S-RND	25F8
25A2	A7	84	S-PI	2627
25A4	A6	8F	S-INKEY\$	2634
25A6	C4	E6	S-BIN (EQU,S-DECIMAL)	2680
25A9	AA	3F	S-SCREEN\$	2668
25AA	AB	C7	S-ATTR	2672
25AC	A9	CE	S-POINT	2678
25AE	00			indicator de sfârșit

THE SCANNING FUNCTION ROUTINES (RUTINELE FUNCTIEI BALEIERE)

25AF S-U-PLUS	RST	0020,NEXT-CHAR	Pentru un singur plus, pur și simplu se trece la următorul caracter și salt înapoi la
	JP	24FF,S-LOOP-1	

punctul principal de intrare al lui SCANNING.

'Rutina baleiere LIMITE': Această rutină lucrează cu limite de sir, sau simple ca "name" sau mai complexe ca "a" "white" "lie" sau aparent redundante VAL\$ "a" "a".

25B3 S-QUOTE	RST 0018,GET-CHAR	Se aduce caracterul curent.
	INC HL	Se indică începutul sirului.
	PUSH HL	Se salvează adresa de început
	LD BC,+0000	Se setează lungimea cu zero.
	CALL 250F,S-QUOTE-S	Se apelează rutina de 'coincidentă'.
	JR NZ,25D9,S-Q-PRMS	Salt dacă zero este resetat - nu mai sînt limite.
25BE S-Q-AGAIN	CALL 250F,S-QUOTE-S	Se apelează din nou pentru a treia limită.
	JR Z,25BE,S-Q-AGAIN	Si din nou pentru a cincea, a saptea, etc.
	CALL 2530,SYNTAX-Z	Dacă se testează sintaxa, se execută un salt pentru a
	JR Z,25D9,S-Q-PRMS	reseta bitul 6 al lui FLAGS si a continua baleierea.
	RST 0030,BC-SPACES	Se creează spațiu în spațiul de lucru pentru sir si pentru limita terminală.
	POP HL	Se aduce indicatorul la început.
	PUSH DE	Se salvează indicatorul în primul spațiu.
25CB S-Q-COPY	LD A,(HL)	Se aduce un caracter din sir.
	INC HL	Se indică următorul.
	LD (DE),A	Se copiază ultimul în spațiul de lucru.
	INC DE	Se indică următorul spațiu.
	CP +22	Este ultimul caracter un ""?
	JR NZ,25CB,S-Q-COPY	Dacă nu este, se execută salt pentru a copia următorul caracter.
	LD A,(HL)	Dar dacă a fost, nu se copiază următorul; dacă următorul este un "", se sare pentru a-l copia pe următorul de după el; altfel se termină cu copierea.
	INC HL	Se aduce adevărata lungime în BC.
	CP +22	
	JR Z,25CB,S-Q-COPY	
25D9,S-Q-PRMS	DEC BC	

De notat că prima limită nu a fost contorizată în lungime; limita finală a fost, si este acum înlăturată. În interiorul sirului, prima, a treia, a cincea, etc., limită au fost contorizate dar a doua, a patra, etc., limită nu au fost contorizate.

	POP DE	Se readuce începutul sirului copiat.
25D8 S-STRING	LD HL,+5C3B	FLAGS; acest punct de intrare este folosit întotdeauna cînd
	RES 6,(HL)	bitul 6 trebuie resetat si
	BIT 7,(HL)	un sir stocat dacă se execută
	CALL NZ,2AB2,STK-STD-\$	o linie. Aceasta se realizează acum.
	JP 2712,S-CONT-2	Salt pentru a continua baleierea liniei.

De notat că la copierea sirului în spațiul de lucru, fiecare două perechi de limite ale sirului din interiorul sirului (" ") vor fi reduse la o pereche de limite ale sirului ("").

25E8 S-BRACKET	RST 0020,NEXT-CHAR	'Rutina de baleiere BRACKET'
	CALL 24FB,SCANNING	aduce doar următorul caracter si apelează recursiv SCANNING.
	CP +29	
	JP NZ,1C8A,REPORT-C	Se prezintă eroare dacă nu este paranteză; apoi se continuă baleierea.
	RST 0020,NEXT-CHAR	
	JP 2712,S-CONT-2	
25F5 S-FN	JP 27BD,S-FN-SBRN	'Rutina de baleiere FN'.

Această rutină, pentru funcțiile definite de utilizator, execută doar salt la

	JR	NC,2660,S-1K\$-STK	se stochează sir gol dacă nu este satisfăcătoare.
	DEC	D	Se pune +FF în D pentru mod L (bitul 3 setat).
	LD	E,A	Valoarea tastei se pune în E pentru a fi decodificată.
	CALL	0030,K-DECODE	Se decodifică valoarea tastei
	PUSH	AF	Se salvează pentru scurt timp valoarea ASCII.
	LD	BC,+0001	Este necesar un spatiu în spatiul de lucru.
	RST	0030,BC-SPACES	Acesta se face acum.
	POP	AF	Se readuce valoarea ASCII.
	LD	(DE),A	Pregătire pentru stocare dacă este un sir.
	LD	C,+01	Lungimea sa este unu.
2660 S-1K\$-STK	LD	B,+00	Se completează parametrul lungime.
	CALL	2AB2,STK-STO-\$	Se stochează sirul cerut.
2665 S-1NK\$-EN	JP	2712,S-CONT-2	Salt înainte.
2668 S-SCREEN\$	CALL	2522,S-2-COORD	Se verifică dacă s-au dat cele 2 coordonate.
	CALL	NZ,2535,S-SCRN\$-B	Se apelează subrutina numai dacă s-a verificat sintaxa;
	RST	0020,NEXT-CHAR	apoi se aduce următorul caracter si salt înapoi.
	JP	25DB,S-STRING	Se verifică dacă s-au dat cele 2 coordonate.
2672 S-ATTR	CALL	2522,S-2-COORD	Se apelează subrutina numai dacă s-a verificat sintaxa;
	CALL	NZ,2580,S-ATTR-S	apoi se aduce parametrul următor si salt înapoi.
	RST	0020,NEXT-CHAR	Se verifică dacă s-au dat cele 2 coordonate.
	JR	26C3,S-NUMERIC	Se apelează subrutina numai dacă s-a verificat sintaxa;
2678 S-POINT	CALL	2522,S-2-COORD	apoi se aduce parametrul următor si salt înapoi.
	CALL	NZ,22CB,POINT-SUB	Se verifică dacă s-au dat cele 2 coordonate.
	RST	0020,NEXT-CHAR	Se apelează subrutina numai dacă s-a verificat sintaxa;
	JR	26C3,S-NUMERIC	apoi se aduce caracterul următor si salt înainte.
2684 S-ALPHA NUM	CALL	2C88,ALPHANUM	Este caracter alfanumeric?
	JR	NC,26DF,S-NEGATE	Salt dacă nu este literă sau cifră.
	CP	+41	Acum salt dacă este literă;
	JR	NC,26C9,S-LETTER	altfel se continuă în S-DECIMAL.

'Rutina baleiere DECIMAL' care urmează lucrează cu punct zecimal sau cu un număr care începe cu o cifră. De asemenea are grijă de expresia 'BIN', cu care lucrează în subrutina zecimal cu virgulă mobilă.

268D S-DECIMAL	CALL	2530,SYNTAX-Z	Salt înainte dacă s-a
(EQU,S-BIN)	JR	NZ,26B5,S-STK-DEC	executat o linie.

Actiunea este de acum foarte diferită de verificarea sintaxei si executia liniei. Dacă sintaxa este verificată atunci forma cu virgulă mobilă trebuie calculată si copiată în linia BASIC actuală. După cum, când se execută o linie forma în punct flotant va fi întotdeauna accesibilă așa că este copiată în stiva calculatorului pentru a forma 'ultima valoare'.

CALL	2C9B,DEC-TO-FP	S-a găsit forma în virgulă mobilă.
RST	0018,GET-CHAR	Se setează HL pentru a indica una peste ultima cifră.
LD	BC,+0006	Sînt cerute șase locatii.
CALL	1655,MAKE-ROOM	Se face o cameră în linia BASIC.
INC	HL	Se indică primul spatiu liber
LD	(HL),+0E	Se introduce codul marcatorului numărului.
INC	HL	Se indică a doua locatie.
EX	DE,HL	Acest indicator este dorit în DE.
LD	HL,(STKEND)	Se aduce STKEND 'vechi'.
LD	C,+05	Trebuie mutați 5 octeți.
AND	A	Se șterge fanionul de transport.
SXC	HL,BC	STKEND 'nou' = STKEND 'vechi'
LD	(STKEND),HL	-5
LDIR		Se mută numărul în virgulă mobilă din stiva

EX	DE,HL	calculatorului în linie. Se pune indicatorul liniei în HL.
DEC	HL	Se indică ultimul octet adăugat.
CALL	0077,TEMP-PTR1	Acesta setează CH-ADD.
JR	26C3,S-NUMERIC	Salt înainte.

In timpul executiei liniei:

26B5 S-STK-DEC	RST	0018,GET-CHAR	Se aduce caracterul curent.
26B6 S-SD-SKIP	INC	HL	Acum se trece la caracterul următor unul după altul pînă s-a găsit codul marcatorului numărului.
	LD	A,(HL)	Se indică primul octet al numărului.
	CP	+0E	Se mută numărul în virgulă mobilă.
	JR	NZ,26B6,S-SD-SKIP	Se setează CH-ADD.
	INC	HL	
	CALL	33B4,STACK-NUM	
	LD	(CH-ADD),HL	

Acum s-a identificat un rezultat numeric, provenit din RND, PI, ATTR, POINT sau un număr zecimal, de aceea bitul 6 al FLAGS trebuie setat.

26C3 S-NUMERIC	SET	6,(FLAGS)	Se setează fanionul
	JR	26DD,S-CONT-1	marcatorului numeric.

THE SCANNING VARIABLE ROUTINE (RUTINA DE BALEIERE VARIABILA)

Cînd s-a identificat numele unei variabile se face un apel la LOOK-VARS care privește acele variabile care există deja în spațiul variabilelor (sau în spațiul programului, la instrucțiunile DEF FN pentru o funcție definită de utilizator DEF FN). Dacă s-a găsit o valoare numerică corespunzătoare atunci ea este copiată în stiva calculatorului folosind STACK-NUM. Oricum, un sir sau intrarea unei multimi sir trebuie să aibă parametrii corespunzători trecuți în stiva calculatorului de subrutina STK-VAR (sau în cazul unei funcții definite de utilizator, de subrutina STK-F-ARG ca și apelată din LOOK-VARS).

26C9 S-LETTER	CALL	28B2,LOOK-VARS	Se caută printre variabilele existente intrarea potrivită.
	JP	C,1C2E,REPORT-2	Se prezintă eroare dacă nu este nici o intrare.
	CALL	Z,9996,STK-VARS	Se stochează parametrii intrării sirului/returnare adresă de bază a elementului numeric.
	LD	A,(FLAGS)	Se aduce FLAGS.
	CP	+C0	Se testează împreună bitii 6 și 7.
	JR	C,26DD,S-CONT-1	Unul sau ambii bitii sînt resetati.
	INC	HL	Se stochează o valoare numerică.
	CALL	33B4,STACK-NUM	Se mută numărul.
26DD S-CONT-1	JR	2712,S-CONT-2	Salt înainte.

Caracterul este comparat cu codul pentru '-', care identifică operația 'minus unar'.

Înainte de testul actual registrul B se setează astfel încît să continue prioritatea +09 iar registrul C să continue codul +D8 al operației care sînt cerute pentru această operație.

26DF S-NEGATE	LD	BC,+09DB	Prioritatea +09, codul operației +D8.
	CP	+2D	Este un '-'?
	JR	Z,270D,S-PUSH-PO	Salt înainte dacă este un 'minus unar'.

In continuare caracterul este comparat cu codul lui 'VAL\$', cu prioritatea 16 zecimal și codul operației 18 hex.

	LD	BC,+1018	Prioritatea 16 zecimal, codul operației +18 hex.
	CP	+AE	Este 'VAL\$'?
	JR	Z,270D,S-PUSH-PO	Salt înainte dacă este 'VAL\$'

Prezentul caracter trebuie să reprezinte acum una dintre funcțiile CODE la NOT, cu codurile între +AF la +C3.

	SUB	+AF	Intervalul funcției este
--	-----	-----	--------------------------

schimbat de la +AF la +C3 la intervalul +00 la +14 hex. Se prezintă o eroare dacă se iese din interval.

JP C,1C8A,REPORT-C

Functia 'NOT' este identificată si este tratată separat de celelalte.

LD BC,+04F0 Prioritatea +04, codul operatiei +F0.
 CP +14 Este functia 'NOT'?
 JR Z,270D,S-PUSH-PO Salt dacă este.
 JP NC,1C8A,REPORT-C Se verifică din nou intervalul.

Functiile rămase au prioritatea 16 zecimal. Acum se calculează codurile operatiilor pentru aceste functii. Functiile care operează cu siruri necesită ca bitul 6 să fie resetat si functiile care dau rezultate sir necesită bitul 7 resetat în codurile lor de operatii.

LD B,+10 Prioritatea 16 zecimal.
 ADD A,+DC Intervalul functiei este acum +DC +EF.
 LD C,A Se transferă codul operatiei.
 CP +DF Se separă CODE, VAL si LEN care operează cu siruri pentru a da rezultate numerice.
 JR NC,2707,S-NO-TO-\$ Se separă STR\$ si CHR\$ care operează cu numere si dau rezultate sir.
 RES 6,C Se marchează codul operatiei. Celelalte coduri de operare au bitii 6 si 7 setati amândoi.

2707 S-NO-TO-\$ CP +EE
 JR C,270D,S-PUSH-PO
 RES 7,C

Codul priorității si codul operatiei pentru functia care a fost considerată este pus acum în stiva masinii. S-a construit astfel o ierarhie a operatiilor.

270D S-PUSH-PO PUSH BC Se stochează codurile de prioritate si al operatiei înainte de a trece la considerarea următoarei părți a expresiei.
 RST 0020,NEXT-CHAR
 JP 24FF,S-LOOP-1

Continuă baleierea liniei. Argumentul prezent poate fi urmat de o '(', un operator binar sau, dacă s-a ajuns la capătul expresiei, atunci un caracter 'carriage return' sau două puncte, un separator sau un 'THEN'.

2712 S-CONT-2 RST 0018,GET-CHAR Se aduce caracterul prezent.
 2713 S-CONT-3 CP +28 Salt înainte dacă nu este o '(' care indică o expresie în paranteze.
 JR NZ,2723,S-OPERTR

Dacă 'ultima valoare' este numerică atunci expresia în paranteze este într-adevăr o subexpresie si trebuie evaluată prin însăși. După cum, dacă 'ultima valoare' este un sir atunci expresia în paranteză reprezintă un element dintr-o multime sau o bucată dintr-un sir. Un apel al SLICING modifică parametrii sirului după cum se cere.

BIT 6,(FLAGS) Salt înainte dacă se lucrează cu o expresie numerică în paranteză.
 JR NZ,2734,S-LOOP Se modifică parametrii 'ultimei valori'.
 CALL 2A52,SLICING Se trece la considerarea următorului caracter.
 RST 0020,NEXT-CHAR
 JR 2713,S-CONT-3

Dacă prezentul caracter este într-adevăr un operator binar va fi dat un cod al operatiei în intervalul +C3 - +CF hex, si un cod de prioritate corespunzător.

2723 S-OPERTR LD B,+00 Codul initial în BC pentru indexare în tabelul operatorilor.
 LD C,A Indicatorul în tabel.
 LD HL,+2795 Indexul în tabel.
 CALL 16DC,INDEXER Salt înainte dacă nu s-a găsit nici o operatie.
 JR NC,2734,S-LOOP Se aduce codul cerut din tabel.
 LD C,(HL)

LD	HL,+26ED	Indicatorul în tabelul de prioritate înseamnă că 26ED +C3 dă 2780 ca prima adresă.
ADD	HL,BC	Index în tabel.
LD	B,(HL)	Se aduce prioritatea corespunzătoare.

Acum se introduce bucla principală a acestei subrutine. În această fază există:

- i. O 'ultimă valoare' în stiva calculatorului.
- ii. Inceputul magaziei de priorități în stiva mașinii sub o ierarhie, de mărime necunoscută, a funcțiilor și codurile de operație binare. Această ierarhie trebuie invalidată.
- iii. Registrul pereche BC conține operația 'prezentă' și prioritatea, care dacă s-a ajuns la sfârșitul expresiei va fi prioritate zero.

Inițial, 'ultima' operație și prioritate sunt scoase din stiva mașinii și comparate cu 'prezentă' operație și prioritate.

Dacă prioritatea 'prezentă' este mai mare decât 'ultima' prioritate atunci se iese din buclă, deoarece 'actuala' prioritate este considerată mai prioritară decât 'ultima' prioritate.

Oricum, dacă actuala prioritate este mai mică atunci se efectuează operația specificată ca 'ultima' operație. 'Actuala' operație și prioritate se reîntorc în stiva mașinii pentru a reîntra în ciclu. În acest fel ierarhia funcțiilor și operațiile binare care au fost puse la coadă sunt prelucrate în ordine corectă.

2734	B-LOOP	POP	DE	Se aduce 'ultima' operație și prioritate.
		LD	A,D	Prioritatea trece în registrul A.
		CP	B	Se compară 'actuala' cu 'ultima'.
		JR	C,2773,S-TIGHTER	Iesire pentru a aștepta argumentul.
		AND	A	Sunt ambele priorități zero?
		JP	Z,0018,GET-CHAR	Iesire prin GET-CHAR făcând astfel 'ultima valoare' rezultatul cerut.

Înainte de a se executa 'ultima' operație, se separă funcția 'USR' în 'numărul USR' și 'sirul USR' în concordanță cu bitul 6 din FLAGS care a fost setat sau resetat când argumentul funcției a fost stocat ca 'ultima valoare'.

		PUSH	BC	Se stochează valoarea 'actuală'.
		LD	HL,+5C2B	Adresa FLAGS.
		LD	A,E	'Ultima' operație este comparată cu codul lui USR, care va da 'numărul USR' dacă este modificat; salt dacă nu este 'USR'.
		CP	+ED	Se testează bitul 6 al FLAGS.
		JR	NZ,274C,S-STKLST	Salt dacă el este setat ('număr USR').
		BIT	6,(HL)	Se modifică codul 'ultimei' operații: 'deplasament' 17, +80 pentru intrare sir și rezultat numeric ('sir USR').
		JR	NZ,274C,S-STK-LST	Se stochează 'ultima' valoare pentru scurt timp.
		LD	E,+99	Nu se efectuează actuala operație până când nu s-a verificat sintaxa.
274C	S-STK-LST	PUSH	DE	Codul 'ultimei' operații.
		CALL	2530,SYNTAX-Z	Se scot bitii 6 și 7 pentru a converti codul operației într-un deplasament al calculatorului.
		JR	Z,275B,Y-SYNTEST	Acum se folosește calculatorul.
		LD	A,E	Se efectuează operația actuală.
		AND	+3F	A fost executată.
		LD	B,A	Salt înainte.
		RST	002B,FP-CALC	
		DEFB	+3B,fp-calc-2	
		DEFB	+3B,end-calc	
		JR	2764,S-RUNTEST	

O pare importantă a verificării sintaxei implică testarea operației pentru a se asigura că natura 'ultimei valori' este de tip corect pentru operație care s-a luat în considerare.

275B S-SYNTEST	LD	A,E	Se aduce codul 'ultimei' operații.
	XDR	(FLAGS)	Se compară natura 'ultimei valori' cu cea cerută de operație. Trebuie să fie la fel pentru a fi o sintaxă corectă.
	AND	+40	Salt dacă sintaxa este greșită.
2761 S-RPORT-C	JP	NZ,1C8A,REPORT-C	

Înainte de a sări înapoi pentru a cicla iar bucla natura 'ultimei valori' trebuie înregistrată în FLAGS.

2764 S-RUNTEST	POP	DE	Se aduce codul 'ultimei' operații.
	LD	HL,+5C3B	Adresa FLAGS.
	SET	6,(HL)	Se presupune că rezultatul este numeric.
	BIT	7,E	Salt înainte dacă natura 'ultimei valori' este numerică.
	JR	NZ,2770,S-LOOPEND	Este sir.
2770 S-LOOPEND	RES	6,(HL)	Se aduce valoarea 'prezentă' în BC.
	POP	BC	Salt înapoi.
	JR	2734,S-LOOP	

De câte ori operația 'actuală' este mai prioritară, 'ultima' și 'actuala' valoare se întorc în stiva mașinii. Oricum, dacă operația 'actuală' cere un sir ca operand atunci codul operației este modificat pentru a indica această cerință.

2773 S-TIGHTER	PUSH	DE	'Ultima' valoare trece în stivă.
	LD	A,C	Se aduce codul operației 'actuale'.
	BIT	6,(FLAGS)	Nu se modifică codul operației dacă se lucrează cu un operand numeric.
	JR	NZ,2790,S-NEXT	Se șterg bitii 6 și 7.
	AND	+3F	Se incrementează codul cu +08 hex.
	ADD	A,+08	Se returnează codul în registrul C.
	LD	C,A	Este operația 'AND'?
	CP	+10	Salt dacă nu este așa.
	JR	NZ,2788,S-NOT-AND	'AND' cere un operand numeric
	SET	6,C	Salt înainte.
2788 S-NOT-AND	JR	2790,S-NEXT	Operațiile -,*,/, și OR nu sînt posibile între siruri.
	JR	C,2761,S-RPORT-C	Este operația '+'?
	CP	+17	Salt dacă este așa.
	JR	Z,2790,S-NEXT	Celelalte operații produc un rezultat numeric.
	SET	7,C	Valoarea 'actuală' trece în stiva mașinii.
2790 S-NEXT	PUSH	BC	Se consideră următorul caracter.
	RST	0020,NEXT-CHAR	Se cicleză din nou bucla.
	JP	24FF,S-LOOP-1	

THE TABLE OF OPERATORS (TABELUL OPERATORILOR)

locatie	cod	cod operator	operator	locatie	cod	cod operator	operator
2795	2B	CF	+	27A3	3C	CB	<
2797	2D	C3	-	27A5	C7	C9	<=
2799	2A	C4	.	27A7	C8	CA	>=
289B	2F	C5	/	27A9	C9	CB	<>
279D	5E	C6		27AB	C5	C7	OR
279F	3D	CE	=	27AD	C6	C8	AND
27A1	3E	CC	>	27AF	00		End marker

THE TABLE OF PRIORITIES (TABELUL PRIORITATILOR)

locatie	prioritate	operator	locatie	prioritate	operator
---------	------------	----------	---------	------------	----------

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	<=			

THE 'SCANNING FUNCTION' SUBROUTINE (SUBROUTINA 'BALEIERE FUNCTIE')

Această subrutină este apelată de 'rutina de baleiere FN' pentru a evalua o funcție definită de utilizator care intervine într-o linie BASIC. Subrutina poate fi considerată în patru faze:

- i. Sintaxa instrucției FN este verificată în timpul verificării sintaxei.
- ii. În timpul execuției liniei, se efectuează o căutare în spațiul programului pentru o instrucțiune DEF FN, și numele funcțiilor sînt comparate pînă se găsește o egalitate - sau se raportează o eroare.
- iii. Argumentele lui FN sînt evaluate prin apelarea lui SCANNING.
- iv. Funcția este evaluată ea însăși prin apelarea lui SCANNING, care apelează pe rînd LOOK-VARS și astfel subrutina 'STACK FUNCTION ARGUMENT'.

27BD S-FN-SBRN	CALL	2530,SYNTAX-Z	Se face un salt la SF-RUN numai dacă s-a verificat sintaxa.
	JR	NZ,27F7,SF-RUN	
	RST	0020,NEXT-CHAR	Se aduce primul caracter al numelui.
	CALL	2C8D,ALPHA	Dacă nu este alfabetic, se prezintă eroare.
	JP	NC,1C8A,REPORT-C	
	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	CP	+24	Este '\$'?
	PUSH	AF	Se salvează fanionul zero în stivă.
	JR	NZ,27D0,SF-BRKT-1	Salt dacă nu a fost '\$'.
27D0 SF-BRKT-1	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	CP	+28	Dacă nu este un caracter '(', se prezintă eroare.
	JR	NZ,27E6,SF-RPRT-C	
	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	CP	+29	Este ')'?
	JR	Z,27E9,SF-FLAG-6	Salt dacă este; nu mai există argumente.
27D9 SF-ARGMTS	CALL	24FB,SCANNING	În buclă se apelează SCANNING pentru a verifica sintaxa fiecărui argument și pentru a introduce numere în virgulă mobilă.
	RST	0018,GET-CHAR	Se aduce caracterul care urmează după argument; salt dacă nu este ',' - nu mai sînt argumente.
	CP	+2C	
	JR	NZ,27E4,SF-BRKT-2	
	RST	0020,NEXT-CHAR	Se aduce primul caracter din următorul argument.
	JR	27D9,SF-ARGMTS	Salt înapoi pentru a lua în considerare acest argument.
27E4 SF-BRKT-2	CP	+29	Este caracterul curent ')'?
27E6 SF-RPRT-C	JP	NZ,1C8A,REPORT-C	Se prezintă eroare dacă nu este.
27E9 SF-FLAG-6	RST	0020,NEXT-CHAR	Se indică următorul caracter din linia BASIC.
	LD	HL,+5C3B	FLAGS; se presupune o funcție valoare sir și se resetează bitul 6 al lui FLAGS.
	RES	6,(HL)	
	POP	AF	Se refacă fanionul zero; salt dacă FN este într-adevăr o valoare sir.
	JR	Z,27F4,SF-SYN-EN	Altfel, se setează bitul 6 al lui FLAGS.
	SET	6,(HL)	Salt înapoi pentru a continua baleierea liniei.
27F4 SF-SYN-EN	JP	2712,S-CONT-2	

ii. În timpul execuției liniei, mai întîi trebuie executată o căutare pentru o instrucție DEF FN.

27F7 SF-RUN	RST	0020,NEXT-CHAR	Se aduce primul caracter al numelui.
	AND	+DF	Resetare bit 5.

	LD	B,A	Se copiază numele în B.
	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	SUB	+24	Se scade 24 hex, codul pentru '\$'.
	LD	C,A	Se copiază rezultatul în C (zero pentru un sir, non-zero pentru o functie numerică).
	JR	NZ,2802,SF-ARGMT1	Salt dacă non-zero: functie numerică.
	RST	0020,NEXT-CHAR	Se aduce următorul caracter, '('.
2802 SF-ARGMT1	RST	0020,NEXT-CHAR	Se aduce primul caracter al primului argument.
	PUSH	HL	Se salvează indicatorul lui în stivă.
	LD	HL,(PROG)	Se indică începutul programului.
	DEC	HL	Întoarcere cu o locație.
2808 SF-FND-DF	LD	DE,+00CE	Se caută 'DEF FN'.
	PUSH	BC	Salvare nume și 'stare sir'.
	CALL	1D86,LOOK-PROG	Se caută acum programul.
	POP	BC	Se readuce numele și starea.
	JR	NC,2814,SF-CP-DEF	Salt dacă s-a găsit o instrucție DEF FN.

Prezentarea P - FN fără DEF

2812 REPORT-P	RST	0008,ERROR-1	Se apelează rutina de tratare eroare.
	DEFB	+18	

Cînd s-a găsit o instrucție DEF FN se compară numele și starea celor două funcții; dacă ele nu coincid, se reia căutarea.

2814 SF-CP-DEF	PUSH	HL	Se salvează indicatorul caracterului DEF FN în cazul în care căutarea trebuie reluată.
	CALL	28AB, FN-SKPOVR	Se aduce numele funcției DEF FN.
	AND	+DF	Resetare bit 5.
	CP	B	Coincide cu numele FN?
	JR	NZ,2825,SF-NOT-FD	Salt dacă nu a coincis.
	CALL	28AB, FN-SKPOVR	Se aduce următorul caracter în DEF FN.
	SUB	+24	Se scade 24 hex, codul pentru '\$'.
	CP	C	Se compară starea cu cea din FN.
	JR	Z,2831,SF-VALUES	Salt dacă s-a găsit coincidentă.
2825 SF-NOT-FD	POP	HL	Se repune indicatorul pe 'DEF FN'.
	DEC	HL	Reîntoarcere o locație.
	LD	DE,+0200	Se folosește rutina de căutare pentru a găsi sfîrșitul instrucției DEF FN, pregătind următoarea căutare; în același timp se salvează numele și starea.
	PUSH	BC	
	CALL	198B,EACH-STMT	
	POP	BC	
	JR	2808,SF-FND-DF	Salt înapoi pentru o căutare ulterioară.

iii. Acum s-a găsit instrucțiunea DEF FN corectă. Argumentele instrucțiunii FN vor fi evaluate prin apelarea repetată a lui SCANNING, și cei 5 octeți valoare (sau parametrii, pentru sir) vor fi introdusi în instrucțiunea DEF FN în spațiile rezervate la verificarea sintaxei. HL va fi folosit pentru a indica de-a lungul instrucțiunii DEF FN (apelînd FN-SKPOVR cînd este necesar) în timp ce CH-ADD indică de-a lungul instrucțiunii FN (apelînd RST 0020, NEXT-CHAR, cînd este necesar).

2831 SF-VALUES	AND	A	Dacă acum HL indică un '\$', se trece la '('.
	CALL	Z,28AB, FN-SKPOVR	
	POP	DE	Se înlătură indicatorul lui 'DEF FN'.
	POP	DE	Se trece indicatorul la primul argument al lui FN, și se copiază în CH-ADD.
	LD	(CH-ADD),DE	Acum se trece peste '('.
	CALL	28AB, FN-SKPOVR	Se salvează acest indicator
	PUSH	HL	

	CP	+29	În stivă.
	JR	Z,2885,SF-R-BR-2	Indică acesta ')'?
2843 SF-ARG-LP	INC	HL	Dacă da, salt: FN nu are argumente.
	LD	A,(HL)	Se indică următorul cod.
	CP	+0E	Se pune codul în A.
	LD	D,+40	Este codul 'marcător număr' OE hex?
	JR	Z,2852,SF-ARG-VL	Pentru un argument numeric se setează bitul 6 al lui D.
	DEC	HL	Salt la zero: argument numeric.
	CALL	28AB, FN-SKPOVR	Acum se asigură că HL indică caracterul '\$' (nu un cod de control).
	INC	HL	HL indică acum 'marcătorul numărului'.
	LD	D,+00	Bitul 6 a lui D este resetat: argument sir.
2852 SF-ARG-VL	INC	HL	Se indică primul din cei cinci octeți din DEF FN.
	PUSH	HL	Se salvează acest indicator în stivă.
	PUSH	DE	Se salvează 'starea sir' a argumentului.
	CALL	24FB, SCANNING	Acum se evaluează argumentul.
	POP	AF	Se pune fanionul nu/sir în A.
	XOR	(FLAGS)	Se compară bitul 6 cu rezultatul din SCANNING.
	AND	+40	Se dă prezentarea Q dacă nu coincid.
	JR	NZ,288B,REPORT-Q	Se aduce indicatorul primului din cele 5 spații din DEF FN în DE.
	POP	HL	Indică HL la STKEND.
	EX	DE,HL	BC va număra 5 octeți care trebuie mutați.
	LD	HL,(STKEND)	Mai întâi se decrementează cu 5, așa că se șterge 'ultima valoare' din stivă.
	LD	BC,+0005	Se copiază octeții în spațiul din DEF FN.
	SBC	HL,BC	HL indică următorul cod.
	LD	(STKEND),HL	Se asigură că HL indică caracterul de după cei 5 octeți.
	LDIR		Este ')'?
	EX	DE,HL	Salt dacă este: nu mai sînt argumente în instrucțiunea DEF FN.
	DEC	HL	Este ','; se salvează indicatorul în el.
	CALL	28AB, FN-SKPOVR	Se aduce caracterul de după ultimul argument din FN care a fost evaluat.
	CP	+29	Salt dacă nu este ','; nu este coincidentă între argumentele din FN și DEF FN.
	JR	Z,2885,SF-RBR-2	Se pune CH-ADD pe următorul argument al lui FN.
	PUSH	HL	HL indică din nou ',' din DEF FN.
	RST	0018,GET-CHAR	Se mută HL la următorul argument din DEF FN.
	CP	+2C	Salt înapoi pentru a lua în considerare acest argument.
	JR	NZ,288B,REPORT-Q	Se salvează indicatorul lui ')' în DEF FN.
	RST	0020,NEXT-CHAR	Se aduce caracterul de după ultimul argument din FN.
	POP	HL	Este ')'?
	CALL	28AB, FN-SKPOVR	Dacă da, salt pentru a evalua funcția; dacă nu, se dă prezentarea Q.
	JR	2843,SF-ARG-LP	
2885 SF-R-BR-2	PUSH	HL	
	RST	0018,GET-CHAR	
	CP	+29	
	JR	Z,288D,SF-VALUE	

REPORT Q - Eroare parametru

288B REPORT-Q	RST	0008,ERROR-1	Se apelează rutina de tratare eroare.
	DEFB	+19	

iv. In final, functia este ea însăși evaluată prin apelarea lui SCANNING, după ce mai întâi s-a setat DEFADD pentru a contine adresa argumentelor așa cum intervin în instrucțiunea DEF FN. Aceasta asigură că LOOK-VARS, când apelează SCANNING, va căuta mai întâi aceste argumente pentru valorile cerute, înainte de o căutare în spațiul variabilelor.

288D SF-VALUE	POP	DE	Reducere indicator pe ')' în DEF FN.
	EX	DE,HL	Se aduce acest indicator în HL.
	LD	(CH-ADD),HL	Se introduce în CH-ADD.
	LD	HL,(DEFADD)	Se aduce vechea valoare a lui DEFADD.
	EX	(SP),HL	Ea se stochează, și se aduce adresa de început a spațiului argumentelor din DEF FN în DEFADD.
	LD	(DEFADD),HL	Se salvează adresa lui ')' în FN.
	PUSH	DE	Se mută CH-ADD trecând peste ')' și '=' la începutul expresiei pentru funcția din DEF FN.
	RST	0020,NEXT-CHAR	Acum se evaluează funcția.
	RST	0020,NEXT-CHAR	Se reface adresa lui ')' în FN.
	CALL	24FJ,SCANNING	Ea se salvează în CH-ADD.
	POP	HL	Se readuce valoarea inițială a lui DEFADD.
	LD	(CH-ADD),HL	Se pune înapoi în DEFADD.
	POP	HL	Se aduce următorul caracter din linia BASIC.
	LD	(DEFADD),HL	Salt înapoi pentru a continua baleierea.
	RST	0020,NEXT-CHAR	
	JP	2712,S-CONT-2	

THE 'FUNCTION SKIPOVER' SUBROUTINE

Această subrutină este folosită de FN și de STK-F-ARG pentru a-l muta pe HL de-a lungul instrucțiunii DEF FN lăsând CH-ADD neschimbat, când el indică de-a lungul instrucțiunii FN.

28AB FN-SKPOVR	INC	HL	Se indică următorul cod din instrucție.
	LD	A,(HL)	Se copiază codul în A.
	CP	+21	Salt înapoi pentru a sări peste el dacă este un cod control sau un spațiu.
	JR	C,28AB, FN-SKPOVR	Sfîrsit.
	RET		

THE 'LOOK-VARS' SUBROUTINE

Această subrutină este apelată întotdeauna când se cere o căutare în spațiul variabilelor sau al argumentelor unei instrucțiuni DEF FN. Subrutina este introdusă cu variabila sistem CH-ADD indicând prima literă a numelui variabilei a cărei locație a fost căutată. Numele va fi în spațiul programului sau în zona de lucru. Inițial subrutina construiește un octet discriminator, în registrul C, care este bazat pe prima literă a numelui variabilei. Bitii 5 & 6 indică tipul variabilei care a fost tratată.

Registrul B este folosit ca un registru bit care contine fanioane.

28B2 LOOK-VARS	SET	6,(FLAG)	Se presupune o variabilă numerică.
	RST	0018,GET-CHAR	Se aduce primul caracter în A
	CALL	2C9D,ALPHA	Este alfabetic?
	JP	NC,108A,REPORT-C	Se prezintă eroare dacă nu este așa.
	PUSH	HL	Se salvează indicatorul pentru prima literă.
	AND	+1F	Se transferă bitii 0 la 4 ai literei în registrul C; bitii 5 & 7 sînt întotdeauna resetati.
	LD	C,A	Se aduce al 2-lea caracter în A.
	RST	0020,NEXT-CHAR	Si acest indicator este salvat.
	PUSH	HL	Este al 2-lea caracter ')'?
	CP	+28	Se separă multimea numerelor.
	JR	Z,28EF,V-RUN/SYN	Acum se setează bitul 6.
	SET	6,C	Este al 2-lea caracter '\$'?
	CP	+24	

JR	Z, 28DE, V-STR-VAR	Se separă toate sirurile.
SET	5, C	Acum se setează bitul 5.
CALL	2C88, ALPHANUM	Salt în față dacă numele variabilei are doar un caracter.
JR	NC, 28E3, V-TEST-FN	

Acum se găsește caracterul final al unui nume care are mai mult de un caracter.

28D4 V-CHAR	CALL 2C88, ALPHANUM	Este caracter alfanumeric?
	JR NC, 28EF, V-RUN/SYN	Salt afară din buclă când s-a găsit sfîrsitul numelui.
	RES 6, C	Se marchează octetul discriminator.
	RST 0020, NEXT-CHAR	Salt înapoi pentru a-l testa.

Sirurile simple și multimile de siruri cer ca bitul 6 al lui FLAGS să fie resetat.

28DE V-STR-VAR	RST 0020, NEXT-CHAR	CH-ADD trece peste '*'. Se resetează bitul 6 pentru a indica un sir.
	RES 6, (FLAGS)	

Dacă DEFADD este non-zero, indicînd că o 'funcție' (o 'FN') a fost evaluată, și dacă este întimpul executiei, se execută o căutare a argumentelor din instrucția DEF FN.

28E3 V-TEST-FN	LD A, (DEFADD-hi)	DEFADD-hi este zero?
	AND A	
	JR Z, 28EF, V-RUN/SYN	Dacă da, salt înainte.
	CALL 2530, SYNTAX-Z	Este în 'timpul executiei'?
	JP NZ, 2951, STK-F-ARG	Dacă da, salt înainte pentru a găsi instrucțiunea DEF FN.

În caz contrar (sau dacă variabila nu a fost găsită în instrucțiunea DEF FN) se execută căutarea în spațiul variabilelor numai dacă sintaxa a fost verificată.

28EF V-RUN/SYN	LD B, C	Se copiază octetul discriminator în registrul B.
	CALL 2530, SYNTAX-Z	Salt înainte dacă este în 'timpul executiei'.
	JR NZ, 28FD, V-RUN	Se mută discriminatorul în A.
	LD A, C	Se coboară partea de cod caracter.
	AND +E0	Se indică sintaxa setînd bitul 7.
	SET 7, A	Se readuce discriminatorul.
	LD C, A	Salt înainte pentru a continua.
	JR 2934, V-SYNTAX	

S-a executat o linie BASIC așa că se execută o căutare în spațiul variabilelor.

28FD V-RUN	LD HL, (VARS)	Se scoate indicatorul VARS.
------------	---------------	-----------------------------

Acum se introduce o buclă pentru a lua în considerare numele variabilelor existente.

2900 V-EACH	LD A, (HL)	Prima literă a fiecărei variabile existente.
	AND +7F	Egalare biti 0 la 6.
	JR Z, 2932, V-80-BYTE	Salt cînd se ajunge la 'octetul 80'.
	CP C	Comparare actuală.
	JR NZ, 292A, V-NEXT	Salt înainte dacă primele caractere nu coincid.
	RLA	Se rotește A la stînga și apoi se dublează pentru testarea bitilor 5 & 6.
	ADD A, A	Siruri și multe de variabile.
	JP P, 293F, V-FOUND-2	Numeric simplu și variabile FOR-NEXT.
	JR C, 293F, V-FOUND-2	

Se cere ca numele întregi să coincidă în întregime.

POP DE	Se face o copie a indicatorului pentru cel de
PUSH DE	

	PUSH	HL	al doilea caracter. Se salvează indicatorul primei litere.
2912 V-MATCHES	INC	HL	Se consideră următorul caracter.
2913 V-SPACES	LD	A, (DE)	Se aduce pe rînd fiecare caracter.
	INC	DE	Se indică următorul caracter.
	CP	+20	Este caracterul 'spatiu'?
	JR	Z, 2913, V-SPACES	Se ignoră spațiile.
	OR	+20	Se setează bitul 5 astfel încît să se egaleze porțiunea literelor mici și mari.
	CP	(HL)	Se face compararea.
	JR	Z, 2912, V-MATCHES	Întoarcere pentru un alt caracter în caz de egalitate.
	OR	+80	Va coincide cu bitul 7 setat?
	CP	(HL)	Se încearcă.
	JR	NZ, 2929, V-GET-PTR	Salt înainte dacă 'ultimele caractere' nu coincid.
	LD	A, (DE)	Se verifică dacă s-a ajuns la sfîrșitul numelui înainte de a executa salt înainte.
	CALL	2C88, ALPHANUM	
	JR	NC, 293E, V-FOUND-1	

În toate cazurile în care numele nu coincid registrul pereche HL trebuie făcut să indice următoarea variabilă din spațiul variabilelor.

2929 V-GET-PTR	POP	HL	Se aduce indicatorul.
292A V-NEXT	PUSH	BC	Se salvează B și C pentru scurt timp.
	CALL	19B8, NEXT-ONE	DE este făcut să indice următoarea variabilă.
	EX	DE, HL	Se schimbă cei doi indicatori
	POP	BC	Se aduc B & C înapoi.
	JR	2900, V-EACH	Se intră din nou în buclă.

Se revine în acest punct dacă nu s-a găsit nici o intrare cu numele corect.

2932 V-80-BYTE	SET	7, B	Semnalizare 'nu s-a găsit variabila'.
----------------	-----	------	---------------------------------------

Se vine în acest punct dacă se verifică sintaxa.

2934 V-SYNTAX	POP	DE	Se coboară indicatorul pe al doilea caracter.
	RST	0018, GET-CHAR	Se aduce caracterul prezent.
	CP	+28	Este ''?'
	JR	Z, 2943, V-PASS	Salt înainte.
	SET	5, B	Se indică faptul că nu se lucrează cu o multime și salt înainte.
	JR	294B, V-END	

Se vine în acest punct dacă s-a găsit o intrare cu numele corect.

293E V-FOUND-1	POP	DE	Se coboară indicatorul variabilei salvate.
293F V-FOUND-2	POP	DE	Se coboară indicatorul celui de al doilea caracter.
	POP	DE	Se coboară indicatorul primei litere.
	PUSH	HL	Se salvează indicatorul 'ultimei' litere.
	RST	0018, GET-CHAR	Se aduce caracterul curent.

Dacă numele variabilei egalate are mai mult de o literă atunci se trece peste celelalte caractere.

Notă: Aceasta pare că s-ar fi făcut deja la V-CHAR.

2943 V-PASS	CALL	2C, 88, ALPHANUM	Este alfanumeric?
	JR	NC, 294B, V-END	Salt cînd s-a găsit sfîrșitul numelui.
	RST	0020, NEXT-CHAR	Se aduce următorul caracter.
	JR	2943, V-PASS	Întoarcere pentru a-l testa.

Acum se setează parametrii de ieșire.

294B V-END	POP	HL	HL conține indicatorul literei unui nume scurt sau
------------	-----	----	--

RL B
BIT 6,B
RET

'ultimul' caracter al unui
nume lung.
Se rotește întregul registru.
Se specifică starea bitului 6
sfîrșit.

Parametrii de ieșire ai subrutinei pot fi prezentați după cum urmează:
Variabila sistem CH-ADD indică prima locație de după numele variabilei așa cum
intervine în linia BASIC.

Dacă 'nu se găsește variabila':

- i. Fanionul de transport este setat.
- ii. Fanionul zero este setat numai cînd căutarea s-a făcut pentru o multime
variabilă.
- iii. Registrul pereche HL indică prima literă a numelui variabilei așa cum
intervine într-o linie BASIC.

Dacă 's-a găsit variabila':

- i. Fanionul de transport este resetat.
- ii. Fanionul zero este setat pentru ambele siruri simple de variabile și
toate variabilele multimei.
- iii. Registrul pereche HL indică litera unui nume 'scurt', sau 'ultimul'
caracter al unui nume lung, sau intrarea existentă care s-a găsit în spațiul
variabilelor.

În toate cazurile bitii 5 & 6 ai registrului C indică tipul variabilei care a
fost tratată. Bitul 7 este complementul fanionului SYNTAX/RUN. Numai cînd
subrutina este folosită în 'timpul execuției' bitii 0 la 4 vor conține codul
literei variabilei.

În timpul sintaxei revenirea se face întotdeauna cu fanionul de transport
resetat. Fanionul zero este setat pentru multimi și resetat pentru toate
celelalte variabile, exceptînd un nume sir simplu incorect urmat de o '(' care
setează fanionul zero și, în cazul SAVE "nume" DATA un '\$()', de asemenea trece
sintaxa ???

THE 'STACK FUNCTION ARGUMENT' SUBROUTINE (SUBROUTINA 'STOCARE ARGUMENT
FUNCTIE')

Această subrutină este apelată de LOOK-VARS cînd DEFADD=hi nu este zero,
pentru a efectua o căutare în spațiul argumentelor instrucțiunii DEF FN,
înainte de a căuta în spațiul variabilelor. Dacă variabila s-a găsit în
instrucțiunea DEF FN, atunci parametrii unei variabile sir sînt stocați și se
dă un semnal că nu este necesar să se apeleze STK/VAR. Dar se permite
baleierea în stivă a valorii unei variabile numerice la 26DA în mod uzual.

2951 STK-ARG	LD HL,(DEFADD)	Se indică primul caracter din spațiul argumentelor și se pune în A.
	LD A,(HL)	Este ')'?
	CP +29	
	JP Z,28EF,V-RUN/SYN	Salt pentru a cerceta spațiul variabilelor.
295A SFA-LOOP	LD A,(HL)	Se aduce următorul argument în buclă.
	OR +60	Se setează bitii 5 & 6, initializînd o variabilă numerică simplă; aceasta se copiază în B.
	LD B,A	
	INC HL	Se indică următorul cod.
	LD A,(HL)	Acesta se pune în registrul A
	CP +0E	Este codul 'marcatorului numărului' 0E hex.
	JR Z,296B,SFA-CP-VR	Salt dacă este așa: variabilă numerică.
	DEC HL	Asigurare că HL indică un caracter '\$', și nu un cod control sau saptiu.
	CALL 28AB, FN-SKPOVR	HL indică acum 'marcatorului numărului'.
	INC HL	
	RES 5,B	Se resetează bitul 5 al lui B: variabilă sir.
296B SFA-CP-VR	LD A,B	Se aduce numele variabilei în A.
	CP C	Este una din cele căutate?

JR	Z, 2981, SFA-MATCH	Salt dacă este.
INC	HL	Acum se trece peste cei cinci octeti ai numărului în virgulă flotantă sau parametrul sir pentru a ajunge la următorul argument.
INC	HL	Se trece la următorul caracter.
INC	HL	Este ')?
INC	HL	Dacă este, salt pentru a cerceta spațiul variabilelor.
INC	HL	Se indică următorul argument.
CALL	28A7, FN-SKPOVR	Salt înapoi pentru a-l lua în considerare.
CP	+29	
JP	Z, 28EF, V-RUN/SYN	
CALL	28A7, FN-SKPOVR	
JR	Z95A, SFA-LOOP	

S-a găsit o egalitate. Parametrii unei variabile sir sînt stocați, evitînd necesitatea apelării subrutinei STK-VAR.

2981 SFA-MATCH	BIT	5, C	Se testează dacă este o variabilă numerică.
	JR	NZ, 2991, SFA-END	Salt dacă variabila este numerică; va fi stocată de SCANNING.
	INC	HL	Se indică primul din cei 5 octeti ce trebuie stocați.
	LD	DE, (STKEND)	In DE se indică STKEND.
	CALL	33C0, MOVE-FP	Se stochează cei 5 octeti.
	EX	DE, HL	Se indică în HL noua poziție a lui STKEND, și se resetează variabila sistem.
	LD	(STKEND), HL	Se scoț indicatorii LOOK-VARS (al doilea și primul indicator caracter).
2991 SFA-END	POP	DE	Se revine din căutare cu fanioanele de transport și zero resetate - semnalizînd că nu se cere o apelare a lui STK-VAR.
	POP	DE	Sfîrșit.
	XOR	A	
	INC	A	
	RET		

THE 'STK-VAR' SUBROUTINE

Această subrutină se folosește de obicei fie pentru a găsi parametrii care definesc intrarea unui sir existent în spațiul variabilelor, fie pentru a returna registrului pereche HL adresa de bază a unui element particular sau a unei mulțimi de numere. Cînd apelul se face din DIM subrutina doar verifică sintaxa instrucției BASIC.

De notat că parametrii care definesc un sir pot fi modificați apelînd SLICING dacă acesta poate fi specificată.

Inițial registrul A și B sînt șterși și bitul 7 al registrului C este testat pentru a determina dacă s-a verificat sintaxa.

2996 STK-VAR	XOR	A	Se șterge dispunerea fanioanelor.
	LD	B, A	Se șterge registrul B pentru mai tîrziu.
	BIT	7, C	Salt înainte dacă s-a verificat sintaxa.
	JR	NZ, 29E7, SV-COUNT	

In continuare sirurile simple sînt separate de mulțimea variabilelor.

BIT	7, (HL)	Salt înainte dacă se lucrează cu o mulțime de variabile.
JR	NZ, 29AE, SV-ARRAYS	

Parametrii unui sir simplu s-au găsit ușor.

29A1 SV-SIMPLE\$	INC	A	Semnalizare 'un sir simplu'.
	INC	HL	Trece de-a lungul intrării.
	LD	C, (HL)	Se scoate partea inferioară a contorului.
	INC	HL	Avans indicator.
	LD	B, (HL)	Se scoate partea superioară a contorului.
	INC	HL	Avans indicator.
	EX	DE, HL	Transfer indicator în sirul actual.
	CALL	2AB2, STK-STORE	Se trec acești parametrii în stiv calculatorului.
	RST	0018, GET-CHAR	Se aduce caracterul prezent și se execută salt înainte

JP 2A49,SV-SLICE pentru a vedea dacă se cere o 'tăiere'.

Acum se găsește adresa de bază a unui element dintr-o multime. Initial se colectează 'numărul dimensiunilor'.

29AE SV-ARRAYS INC HL Se trece peste octetii lungime.
 INC HL
 INC HL
 LD B,(HL) Se colectează 'numărul dimensiunilor'.
 BIT 6,C Salt înainte dacă se tratează o multime de numere.
 JR Z,29C0,SV-PTR

Dacă o multime de siruri are 'numărul său de dimensiuni' egal cu 'i' atunci o astfel de multime poate fi tratată ca un sir simplu.

DEC B Se decrementează 'numărul dimensiunilor' și se execută salt dacă acum acest număr este zero.
 JR Z,29A1,SV-SIMPLE#

In continuare se face o verificare pentru a se asigura că în linia BASIC variabila este urmată de o semnătură.

EX DE,HL Salvare indicator în DE.
 RST 0018,GET-CHAR Se aduce caracterul prezent.
 CP +28 Este ' '?
 R NZ,2A20,REPORT-3 Se prezintă eroare dacă nu este așa.
 EX DE,HL Se reface indicatorul.

Si pentru multimile numerice si pentru multimile de siruri indicatorul variabilei este transferat în registrul pereche DE înainte de evaluarea semnăturii.

29C0 SV-PTR EX DE,HL Se trece indicatorul în DE.
 JR 29E7,SV-COUNT Salt înainte.

Următoarea buclă este folosită pentru a găsi parametrii unui element specificat în cadrul unei multimi. Buclă este introdusă prin punctul de mijloc - SV-COUNT - unde contorul elementelor este setat pe zero.

Buclă este accesată de 'B' ori, aceasta fiind, pentru o multime numerică, egal cu numărul dimensiunilor care au fost folosite, dar pentru o multime de siruri 'B' este cu unu mai mic decât numărul dimensiunilor în folosință cum ultima semnătură este folosită pentru a specifica 'segment' al sirului.

29C3 SV-COMMA PUSH HL Salvare contor.
 RST 0018,GET-CHAR Se aduce caracterul prezent.
 POP HL Se reface contorul.
 CP +2C Este caracterul prezent o ' '?
 JR Z,29EA,SV-LOOP Salt înainte pentru a considera o altă semnătură.
 BIT 7,C Dacă o linie a fost executată atunci acolo este o eroare.
 JR Z,2A20,REPORT-3 Salt înainte dacă se lucrează cu o multime de siruri.
 BIT 6,C Este caracterul prezent ' '?
 JR NZ,29D8,SV-CLOSE Se dă eroare dacă nu este așa
 CP +29 Avans CH-ADD.
 JR NZ,2A12,SV-RPT-C Revenire dacă sintaxa este corectă.
 RST 0020,NEXT-CHAR
 RET

Pentru o multime de siruri semnătura prezentă poate reprezenta un 'segment', sau semnătura pentru un 'segment' poate fi încă prezentă în linia BASIC.

29D8 SV-CLOSE CP +29 Este caracterul prezent ' '?
 JR Z,2A48,SV-DIM Salt înainte și verificare dacă acolo mai este o altă semnătură.
 CP +CC Este caracterul prezent 'TO' ?
 JR NZ,2A12,SV-RPT-C Nu trebuie să fie altfel.
 29E0 SV-CH-ADD RST 0018,GET-CHAR Se aduce caracterul prezent.
 DEC HL Se indică precedentul caracter și se setează CH-ADD
 LD (CH-ADD),HL Evaluarea 'segment'.
 JR 2A45,SV-SLICE

Introducere buclă aici.

29E7 SV-COUNT	LD	HL,+0000	Setare contor pe zero.
29EA SV-LOOP	PUSH	HL	Se salvează contorul pentru scurt timp.
	RST	0020,NEXT-CHAR	Avans CH-ADD.
	POP	HL	Se reface contorul.
	LD	A,C	Se aduce octetul discriminator.
	CP	+C0	Salt numai dacă s-a verificat sintaxa pentru o multime de siruri.
	JR	NZ,29FB,SV-MULT	Se aduce caracterul prezent.
	RST	0018,GET-CHAR	Este o ')?
	CP	+29	Salt înainte ca la terminarea contorizării elementelor.
	R	Z,2A48,SV-DIM	Este 'TO'?
	CP	+CC	Salt înapoi dacă se lucrează cu un 'segment'.
	JR	Z,29E0,SV-CH-ADD	Se salvează contorul număr dimensiune și octetul discriminator.
29FB SV-MULT	PUSH	BC	Salvare contor-element.
	PUSH	HL	Se aduce o dimensiune mărime în DE.
	CALL	2AEE,DE,(DE+1)	Se trece contorul în HL și se stochează indicatorul variabilei.
	EX	(SP),HL	Contorul trece în DE și mărimea dimensiune în HL.
	EX	DE,HL	Se evaluează următoarea semnătură.
	CALL	2ACC,INT-EXPI	Se dă eroare dacă iasă din interval.
	JR	C,2A20,REPORT-3	Rezultatul evaluării este decrementat la fel ca și contorul ce contorizează elementele care intervin înaintea elementului specificat.
	DEC	BC	Se multiplică contorul prin mărimea dimensiune.
	CALL	2AF4,GET-HL*DE	Se adună rezultatul lui 'INT-EXPI' la prezentul contor.
	ADD	HL,BC	Se aduce indicatorul variabil
	POP	DE	Se aduce numărul dimensiune și octetul discriminator.
	POP	BC	Se cicleză bucla pînă cînd 'B' devine egal cu zero.
	DJNZ	29C3,SV-COMMA	

Fanionul SYNTAX/RUN este verificat înainte de a se separa multimile de siruri de multimile de numere.

2A12 SV-RPT-C	BIT	7,C	Se dă eroare dacă se verifică sintaxa în acest punct.
	JR	NZ,2A7A,SL-RPT-C	Salare contor.
	PUSH	HL	Salt înainte dacă se tratează o multime de siruri.
	BIT	6,C	
	JR	NZ,2A2C,SV-ELEM#	

Cînd se lucrează cu o multime de numere caracterul prezent trebuie să fie o ')?

	LD	B,D	Se transferă indicatorul variabil în registrul pereche BC.
	LD	C,E	
	RST	0018,GET-CHAR	Se aduce caracterul prezent.
	CP	+29	Este o ')?
	JR	Z,2A22,SV-NUMBER	Salt peste prezentare eroare numai dacă este necesar.

Prezentarea 3- Semnătură afară din interval

2A20 REPORT-3	RST	0008,ERROR-1	Se apelează rutina de tratare eroare.
	DEFB	+02	

Adresa locatiei dinainte de actuala formă în punct flotant poate fi acum calculată.

2A22 SV-NUMBER	RST	0020,NEXT-CHAR	Avans CH-ADD.
	POP	HL	Se aduce contorul.

LD	DE,+0005	Există cinci octeti pentru fiecare element dintr-o multime de numere.
CALL	2AF4,GET-HL*DE	Se calculează numărul total al octetilor dinainte de elementul cerut.
ADD	HL,BC	HL va indica locatia dinainte de elementul cerut.
RET		Revenire cu această adresă.

Cînd se lucrează cu o multime de siruri lungimea unui element este dată de ultima 'mărime-dimensiune'. Se calculează parametrii caracteristici și apoi se trec în stiva calculatorului.

2A2C SV-ELEM\$	CALL	2AEE,DE,(DE+1)	Se aduce ultima mărime - dimensiune.
	EX	(SP),HL	Indicatorul variabil trece în stivă și contorul trece în HL.
	CALL	2AF4,GET-HL*DE	Se multiplică 'contorul' cu 'mărimea dimensiune'.
	POP	BC	Se aduce indicatorul variabil
	ADD	HL,BC	Aceasta face ca HL să indice locatia de dinainte de sir.
	INC	HL	Asa că se indică actualul 'start'.
	LD	B,D	se transferă ultima mărime - dimensiune în BC pentru a forma 'lungimea'.
	LD	C,E	Se mută 'startul' în DE.
	EX	DE,HL	Se trec acești parametrii în stiva calculatorului. Notă: Primul parametru este zero cînd indică un sir dintr-o 'multime de siruri' și deci intrarea existentă nu va trebui refăcută.
	CALL	2AB1,STK-ST-0	

Există trei forme posibile pentru ultima semnătură. Prima este ilustrată de - A\$(2,4 TO 8) -, a doua de - A\$(2)(4 TO 8) - și a treia de - A\$(2) - care este o formă absentă și indică faptul că întregul sir este cerut.

	RST	0018,GET-CHAR	Se aduce prezentul caracter.
	CP	+29	Este o ')?'
	JR	Z,2A48,SV-DIN	Salt dacă este.
	CP	+2C	Este o ',)?'
	JR	NZ,2A20,REPORT-3	Prezentare eroare dacă nu este.
2A45 SV-SLICE	CALL	2A52,SLICING	Se folosește SLICING pentru a modifica setul de parametrii.
2A48,SV-DIN	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
2A49 SV-SLICE?	CP	+28	Este o '()?'
	JR	Z,2A45,SV-SLICE	Salt înapoi dacă trebuie considerat un 'segment'.

Se poate face o revenire cînd s-a terminat considerarea ultimei semnături.

RES	6,(FLAGS)	Semnalizare - rezultat sir.
RET		Revenire cu parametrii sirului formînd o 'ultimă valoare' în stiva calculatorului.

THE 'SLICING' SUBROUTINE (SUBROUTINA 'SEGMENTARE')

Prezentul sir poate fi segmentat folosind această subrutină. Subrutina este introdusă cu parametrii sirului prezenti în vîrfurile stivei calculatorului și în registrii A, B, C, D & E. Initial se testează fanionul SYNTAX/RUN și parametrii sirului sînt adusi numai pentru o linie care a fost executată.

2A52 SLICING	CALL	2530,SYNTAX-Z	Se verifică fanionul.
	CALL	NZ,2BF1,STK-FETCH	Se iau parametrii din stivă în 'timpul executiei'.

Se consideră posibilitatea 'segmentului' ce a fost '()'

RST	0020,NEXT-CHAR	Se aduce următorul caracter.
CP	+29	Este o ')?'
JR	Z,2AAD,SL-STORE	Salt înainte dacă este așa.

Înainte de executia registrilor sînt prelucrați după cum urmează:

PUSH	DE	'Startul' trece în stiva masinii.
XOR	A	Registrul A este sters și salvat.
PUSH	AF	Se salvează 'lungimea' pentru scurt timp.
PUSH	BC	Se presupune că 'segmentul' urmează să înceapă cu primul caracter.
LD	DE,+0001	Se aduce primul caracter, se trece 'lungimea' în HL.
RST	0018,GET-CHAR	
POP	HL	

Acum se evaluează primul parametru al 'segmentului'.

CP	+CC	Este caracterul prezent 'TD'?
JR	Z,2A81,SL-SECOND	Primul parametru, prin absență, va fi '1' dacă se execută saltul.
POP	AF	In această fază A este zero.
CALL	2ACD,INT-EXP2	BC va conține primul parametru. A va conține +FF dacă a fost o eroare 'afară din interval'.
PUSH	AF	Oricum se salvează valoarea.
LD	D,B	Se transferă primul parametru în DE.
LD	E,C	
PUSH	HL	Se salvează 'lungimea' pentru scurt timp.
RST	0018,GET-CHAR	Se aduce caracterul prezent.
POP	HL	Se reface 'lungimea'.
CP	+CC	Este caracterul prezent 'TD'?
JR	Z,2A81,SL-SECOND	Dacă este așa, salt înainte pentru a lua în considerare a doilea parametru.
CP	+Z9	altfel se arată că acolo este o paranteză închisă.
2A7A SL-RPT-C	JP	NZ,1C8A,REPORT-C

În acest punct a fost identificat un 'segment' de un singur caracter, cum ar
fi - A5(4).

LD	H,D	Ultimul caracter al segmentului este de asemenea primul caracter.
LD	L,E	
JR	2A94,SL-DEFINE	Salt înainte.

Acum se evaluează al doilea parametru al 'segmentului'.

2A81 SL-SECOND	PUSH	HL	Se salvează 'lungimea' pentru scurt timp.
	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	POP	HL	Se reface 'lungimea'.
	CP	+Z9	Este caracterul prezent ')'?
	JR	Z,2A94,SL-DEFINE	Salt dacă nu există un al doilea parametru.
	POP	AF	Dacă primul parametru a fost în interval A va conține zero
	CALL	2ACD,INT-EXP2	BC va conține al doilea parametru.
	PUSH	AF	Salvare 'registru eroare'.
	RST	0018,GET-CHAR	Se aduce caracterul prezent.
	LD	H,B	Se trece rezultatul obținut din INT-EXP2 în registrul pereche HL.
	LD	L,C	
	CP	+Z9	Se verifică dacă este o paranteză închisă.
	JR	NZ,2A7A,SL-RPT-C	

Acum se definesc 'noi' parametrii.

2A94 SL-DEFINE	POP	AF	Se aduce 'registru eroare'.
	EX	(SP),HL	Al doilea parametru trece în stivă iar 'startul' trece în HL.
	ADD	HL,DE	Primul parametru este adunat cu 'startul'.
	DEC	HL	Întoarcere cu o locație pentru corectare.
	EX	(SP),HL	'Noul start' trece în stivă

	AND	A	iar al doilea parametru trece în HL.
	SRC	HL, DE	Se scade primul parametru din al doilea parametru pentru a afla 'lungimea' segmentului.
	LD	BC, +0000	Se initializează 'lungimea nouă'.
	JR	C, 2AA8, SL-OVER	Un 'segment' negativ este un 'sir nul' mai degrabă decât o condiție de eroare (vezi manual).
	INC	HL	Incuviintarea octetului inclusiv.
	AND	A	Doar acum se testează 'registru eroare'.
	JP	M, 2A20, REPORT-3	Salt dacă fiecare parametru a format în afara intervalului sirului.
	LD	B, H	Se transferă 'noua lungime' în BC.
2AA8 SL-OVER	LD	C, L	Se aduce 'noul start'.
	POP	DE	Asigurare că încă se mai indică un sir.
	RES	6, (FLAGS)	Revenire în acest punct dacă se verifică sintaxa; altfel se continuă în subrutina STK-STORE.
2AAD SL-STORE	CALL	2530, SYNTAX-Z	
	RET	Z	

THE 'STK-STORE' SUBROUTINE

Această subrutină trece valorile continute în registrii A, B, C, D & E în stiva calculatorului. Astfel stiva crește ca mărime cu 5 octeti cu fiecare apelare a acestei subrutine.

În mod normal subrutina se folosește pentru a transfera parametrul sirului dar ea se mai folosește și se STACK-BC și LOG (Z/A) pentru a transfera 'numere întregi mici' în stivă.

De notat că atunci când se stochează parametrul unui sir, prima valoare stocată (venită din registrul A) va fi zero dacă sirul vine dintr-o multime de siruri sau este un 'segment' al unui sir. Valoarea pentru un sir simplu complet va fi '1'. Acest 'fanion' este folosit în rutina de comandă 'LET' când '1' semnalizează că vechea copie a sirului trebuie 'refăcută'.

2AB1 STK-ST-0	XOR	A	Semnalizare - un sir dintr-o multime de siruri sau un 'segment' dintr-un sir.
2AB2 STK-STO-#	RES	6, (FLAGS)	Asigurare că fanionul indică un rezultat sir.
2AB6 STK-STORE	PUSH	BC	Se salvează pentru scurt timp B & C.
	CALL	33A9, TEST-5-SP	Este loc pentru 5 octeti?
	POP	BC	Se refacă B & C.
	LD	HL, (STKEND)	Se aduce adresa primei locații de deasupra stivei prezente.
	LD	(HL), A	Se transferă primul octet.
	INC	HL	Se trece mai departe.
	LD	(HL), E	Se transferă al doilea și al treilea octet; pentru un sir acestia vor reprezenta 'startul'.
	INC	HL	Se trece mai departe.
	LD	(HL), C	Se transferă al patrulea și al cincilea octet; pentru un sir acestia vor reprezenta 'lungimea'.
	INC	HL	Se trece mai departe astfel încât să se indice locația de deasupra stivei.
	LD	(HL), B	Se salvează această adresă în STKEND și se revine.
	INC	HL	
	LD	(STKEND), HL	
	RET		

THE 'INT-EXP' SUBROUTINE

Această subrutină revine cu rezultatul evaluării 'expresiei următoare' ca o valoare întreagă continuată în registrul pereche HL. De asemenea subrutina compară acest rezultat cu o valoare limită înlocuită în registrul pereche HL. Fanionul de transport va fi resetat dacă este o eroare în afara intervalului.

Registrul A este folosit ca un 'registru de eroare' și conține +00 dacă nu este eroare anterioară și +FF dacă a fost una.

2ACC INT-EXP1	XOR	A	Se șterge 'registru	de eroare'.
2ACD INT-EXP2	PUSH	DE	Se salvează pe tot parcursul	registrii pereche DE & HL.
	PUSH	HL	Se salvează temporar	'registru de eroare'.
	PUSH	AF	Se evaluează 'expresia	următoare pentru a obține în
	CALL	1C82,EXPT-1NUM	stiva calculatoului 'ultima	valoare'.
	POP	AF	Se reface 'registru de	eroare'.
	CALL	2530,SYNTAX-2	Salt înainte dacă se verifică	sintaxa.
	JR	Z,2AEB,I-RESTORE	Se salvează din nou registru	de eroare.
	PUSH	AF	'Ultima valoare' este	comprimată în BC.
	CALL	1E99,FIND-INT2	Registru de eroare în DE.	O 'expresie următoare' care
	POP	DE	dă zero este întotdeauna o	eroare, așa că salt înainte
	LD	A,B	dacă este așa.	Se face o copie a valorii
	OR	C	limită. Aceasta va fi o	'mărime dimensiune', o 'DIM
	SCF		limit' sau o 'lungime sir'.	Acum se compară rezultatul
	JR	Z,2AEB,I-CARRY	evaluării expresiei cu limita	
	POP	HL		
	PUSH	HL		
	AND	A		
	SBC	HL,BC		

Acum se modifică starea fanionului de transport și valoarea conținută în registru D astfel încât să se obțină valoarea corespunzătoare pentru 'registru de eroare'.

2AEB I-CARRY	LD	A,D	Se aduce 'vechea valoare a	erorii'.
	SBC	A,+00	Se formează 'noua valoare a	erorii': +00 dacă nu este
			niciodată eroare / +FF sau	mai mică decât s-a găsit o
			eroare 'în afara	intervalului' în această
			trecere sau într-una	anterioară.

Înainte de revenire se refac registrii.

2AEB I-RESTORE	POP	HL	Se refac HL & DE.
	POP	DE	
	RET		Revenire; 'registru de
			eroare' este registru A.

THE 'DE,(DE+1)' SUBROUTINE

Această subrutină execută structura - LD DE,(DE+1) - și revine cu HL indicând 'DE+2'.

2AEE DE,(DE+1)	EX	DE,HL	Pentru construire se	folosește HL.
	INC	HL	Se indică 'DE+1'.	
	LD	E,(HL)	De fapt - LD E,(DE+1).	
	INC	HL	Se indică 'DE+2'.	
	LD	D,(HL)	De fapt - LD D,(DE+2).	
	RET		Sfârșit.	

THE 'GET-HL*DE' SUBROUTINE

Numai dacă sintaxa a fost verificată această subrutină apelează 'HL=HL*DE' care execută structura implicit.

Excesul celor 16 bit accesibili în registru pereche HL dă prezentarea 'afară din memorie'. Aceasta nu este adevărata situație dar implică faptul că memoria nu este suficient de mare pentru taskul examinat de programator.

2AF4 GET-HL*DE	CALL	2530,SYNTAX-2	Revenire directă dacă s-a	verificat sintaxa.
	RET	Z	Se execută înmulțirea.	
	CALL	30A9,HL=HL*DE	Prezentare 'Afară din	memorie'.
	JP	C,IF15,REPORT-4	Sfârșit.	
	RET			

THE 'LET' COMMAND ROUTINE (RUTINA DE COMANDA 'LET')

Aceasta este rutina actuală de desemnare a comenzilor LET, READ și INPUT.

Dacă variabila destinatie este o 'variabilă nou declarată', atunci DEST va indica prima literă a numelui variabilei așa cum intervine în linia BASIC. Bitul 1 al lui FLAGX va fi setat.

Oricum, dacă variabila destinatie 'deja există', atunci bitul 1 al lui FLAGX va fi resetat și DEST va indica o variabilă numerică în locația dinaintea celor cinci octeți ai 'vechiului număr'; și pentru o variabilă sir va indica prima locație a 'sirului vechi'. Utilizarea lui DEST în acest fel se aplică la variabile simple și la elementele unei multimi.

Bitul 0 al lui FLAGX este setat dacă variabila destinatie este o variabilă sir simplu 'complet'. (Semnalizare - se șterge copia veche.)

Initial, valoarea curentă a lui DEST este colectată și se testează bitul 1 al lui FLAGX.

2AFF LET	LD	HL,(DEST)	Se aduce adresa prezentă în DEST.
	BIT	1,(FLAGX)	Salt dacă se tratează o variabilă care 'există deja'.
	JR	Z,2B66,L-EXISTS	

Se folosește o 'variabilă nou declarată'. Așa că mai întâi se găsește lungimea numelui său.

	LD	BC,+0005	Se presupune că se lucrează cu o variabilă numerică - 5 octeți.
--	----	----------	---

Se introduce o buclă care va lucra cu caracterele unui nume lung. Se ignoră orice cod de culoare sau spațiu din nume.

2B0B L-EACH-CH	INC	BC	Se adună '1' la contor pentru fiecare caracter al numelui.
2B0C L-NO-SP	INC	HL	Se trece de-a lungul numelui variabilei.
	LD	A,(HL)	Se aduce 'codul prezent'.
	CP	+20	Salt înapoi dacă este un 'spațiu'; astfel se ignoră spațiile.
	JR	Z,2B0C,L-NO-SP	Salt înainte dacă codul este +21 la +FF.
	JR	NC,2B1F,L-TEST-CH	Se acceptă, ca și cod final, cel din intervalul +00 la +FF.
	CP	+10	De asemenea se acceptă și intervalul +16 la +1F.
	JR	C,2B29,L-SPACES	Se trece peste codul de control după fiecare INK la OVER.
	CP	+16	
	JR	NC,2B29,L-SPACES	Salt înapoi ca și cum aceste coduri de control sînt tratate ca spații.
	INC	HL	
	JR	2B0C,L-NO-SP	

Se separă numele 'numeric' și 'sir'.

2B1F L-TEST-CH	CALL	2C88,ALPHANUM	Este cod alfanumeric?
	JR	C,2B0B,L-EACH-CH	Dacă da, atunci se acceptă ca și cum ar fi un caracter al unui nume 'lung'.
	CP	+24	Este codul prezent '\$'?
	JP	Z,2B0C,L-NEW*	Salt înainte ca și cum s-ar trata un sir simplu 'nou declarat'.

'Variabila numerică nou declarată' tratată în prezent va cere 'BC' spații în spațiul variabilelor pentru numele și valoarea sa. Spațiul este accesat și numele variabilei este copiat peste cu caracterele 'marcate' cum s-a cerut.

2B29 L-SPACES	LD	A,C	Se copiază 'lungimea' în A.
	LD	HL,(E-LINE)	HL va indica al 80-lea octet la sfîrșitul zonei variabilelor.
	DEC	HL	Acum se deschide zona variabilelor.
	CALL	1655,MAKE-ROOM	Notă: De fapt spațiile 'BC' se fac înainte de a înlocui 'octetul 80'.
	INC	HL	Se indică primul octet 'nou'.
	INC	HL	DE va indica al doilea octet 'nou'.
	EX	DE,HL	Se salvează acest indicator.
	PUSH	DE	Se aduce indicatorul la începutul numelui.
	LD	HL,(DEST)	

DEC	DE	DE	va indica primul octet 'nou'.
SUB	+06	B	va contine 'numarul de litere aditionale' care sint gasite intr-un 'nume lung'.
LD	B,A		Salt inainte daca se lucreaza cu o variabila cu 'nume scurt'.
JR	Z,2B4F,L-SINGLE		

Codurile 'aditionale' ale unui nume lung sint trecute in spatiul variabilelor.

2B3E L-CHAR	INC	HL	Se indica fiecare cod 'aditional'.
	LD	A,(HL)	Se aduce codul.
	CP	+21	Se accepta coduri de la +21 la +FF; se ignora codurile de la +00 la +20.
	JR	C,2B3E,L-CHAR	Se seteaza bitul 5 pentru portiunea cu litere mici.
	OR	+20	Se transfera pe rind codurile in al 2-lea octet 'nou' in continuare.
	INC	DE	Se cicleaza bucla pentru toate codurile 'aditionale'.
	LD	(DE),A	
	DJNZ	2B3E,L-CHAR	

Ultimul cod al unui nume 'lung' trebuie sa fie facut SAU (OR) cu +80.

OR	+80	Se marcheaza codul cum se cere si se suprainprima ultimul cod.
LD	(DE),A	

Se considera acum prima litera a numelui variabilei ce a fost tratata.

	LD	A,+C0	Pregatire pentru marcarea literei unui 'nume' lung.
2B4F L-SINGLE	LD	HL,(DEST)	Se aduce indicatorul la litera.
	XOR	(HL)	A contine +00 pentru un nume 'scurt' si +C0 pentru un nume 'lung'.
	OR	+20	Setare bit 5 pentru portiunea cu litere mici.
	POP	HL	Acum se coboara indicatorul.

Acum se apeleaza subrutina L-FIRST pentru a introduce 'litera' in locatia ei corespunzatoare.

CALL	2BEA,L-FIRST	Se introduce litera si se revine cu HL indicind 'noul octet 80'.
------	--------------	--

Acum se poate transfera 'ultima valoare' in spatiul variabilelor. De notat ca in acest punct HL indica intotdeauna locatia de dupa cele cinci locatii alocate numarului.

Se foloseste o instructiune 'RST 0028' pentru a apela CALCULATOR si 'ultima valoare' este stearsa. Oricum aceasta valoare nu este suprainprimata.

2B59 L-NUMERIC	PUSH	HL	Se salveaza indicatorul 'destinatie'.
	RST	0028,FP-CALC	Utilizare calculator.
	DEFB	+02,delete	Acesta muta STKEND inapoi cu cinci octeti.
	DEFB	+38,end-calc	Se reface indicatorul.
	POP	HL	Se da numarului o 'lungime' de cinci octeti.
	LD	BC,+0005	HL va indica prima din cele cinci locatii si se executa salt pentru a realiza actualul transfer.
	AND	A	
	SBC	HL,BC	
	JR	2BA6,L-ENTER	

Se vine aici daca se considera o variabila care 'exista deja'. Mai intii se testeaza bitul 6 al lui FLAGS astfel incit sa se separe variabilele numerice de variabile sir sau multiimi de siruri.

2B66 L-EXISTS	BIT	6,(FLAGS)	Salt mai departe daca se trateaza orice fel de variabila sir.
	JR	Z,2B72,L-DELETE-#	

Pentru variabile numerice numarul 'nou' se suprainprima peste numarul

'vechi'. Aşa că mai întâi HL va trebuie să indice locația de după cei cinci octeți ai intrării existente. În prezent HL indică locația de dinaintea de cei cinci biți.

LD	DE,+0006	Cei cinci octeți ai unui număr + '1'.
ADD	HL,DE	HL indică acum 'după'.
JR	ZB59,L-NUMERIC	Salt înapoi pentru a efectua transferul actual.

Se aduc parametrii variabilei sir și sirurile simple complete se separă de sirurile 'segmentate' și mulțimi de siruri.

2B72 L-DELETE-#	LD	HL,(DEST)	Se aduce 'începutul'. Notă: Această linie este de prisos. Se aduce 'lungimea'.
	LD	BC,(STRLEN)	Salt dacă se lucrează cu un sir simplu complet; doar în acest caz este necesar să se ștergă sirul 'vechi'.
	BIT	0,(FLAGX)	
	JR	NZ,ZBAF,L-ADD#	

Cînd se lucrează cu un segment al unui sir simplu existent, un 'segment' al sirului dintr-o mulțime de siruri sau un sir complet dintr-o mulțime de siruri există două faze distincte. În prima fază se construiește 'noul' sir în spațiul de lucru, lungindu-l sau scurtîndu-l, după cum se cere. În a doua fază se copiază 'noul' sir în spațiul alocat lui în spațiul variabilelor.

LD	A,D	Revenire dacă sirul este invalidat.
OR	C	
RET	Z	

Apoi se realizează numărul cerut de spații accesibile în spațiul de lucru.

PUSH	HL	Salvare 'început' (DEST).
RST	0030,BC-SPACES	Se realizează spațiul total necesar în spațiul de lucru.
PUSH	DE	Se salvează indicatorul primei locații.
PUSH	BC	Se salvează 'lungimea' pentru a fi folosită mai târziu.
LD	D,H	DE va indica ultima locație.
LD	E,L	
INC	HL	HL va indica 'una peste' locația nouă.
LD	(HL),+20	Se introduce un caracter 'spatiu'.
LDDR		Se copiază acest caracter în toate locațiile noi. Se termină cu HL indicînd prima locație nouă.

Acum se aduc din stiva calculatorului parametrii sirului care a fost tratat.

PUSH	HL	Se salvează indicatorul pentru scurt timp.
CALL	ZBF1,STK-FETCH	Se aduc parametrii 'noi'.
POP	HL	Se reface indicatorul

Notă: În acest punct necesarul total de spații a fost făcut accesibil în spațiul de lucru pentru 'variabilele în atribuire'. De exemplu - pentru instrucțiunea - LET A\$(4 TO 8)="abcdefg" - au fost făcute cinci locații.

Parametrii încărcati deasupra ca o 'ultimă valoare' reprezintă sirul care trebuie copiat în noile locații cu fortarea lungirii sau scurtării, după cum se cere.

Lungimea sirului 'nou' se compară cu lungimea spațiului făcut accesibil pentru el.

EX	(SP),HL	'Lungimea' noii zone în HL.
		'Indicatorul' noii zone în stivă.
AND	A	Se compară cele două 'lungimi' și se execută salt înainte dacă 'noul' sir va corespunde spațiului.
SBC	HL,BC	
ADD	HL,BC	Aceasta înseamnă că nu se cere scurtarea.
JR	NC,ZB9B,L-LENGTH	Oricum se modifică 'noua' lungime dacă este prea mare.
LD	B,H	'Lungimea' noii zone în
LD	C,L	
2B9B L-LENGTH	EX	(SP),HL

stivă.

Indicatorul' noii zone în HL.

Atîta timp cît noul sir nu este un 'sir invalidat' el este copiat în spațiul de lucru. Lungirea forțată se realizează automat dacă 'noul' sir este scurt decît spațiul accesat pentru acesta.

EX	DE,HL	'Inceputul' sirului nou în HL
LD	A,B	'Indicatorul' noii zone în DE
OR	C	Salt înainte dacă sirul 'nou'
JR	Z,2BA3,L-IN-W/S	este un sir invalidat.
LDIR		Altfel se mută sirul 'nou' în
		spațiul de lucru.

Se refac valorile care au fost salvate în stiva mașinii.

2BA3 L-IN-W/S	POP BC	'Lungimea' zonei noi.
	POP DE	'Indicatorul' zonei noi.
	POP HL	'Inceputul' - indicatorul
		'variabilei în atribuire'
		care a fost inițial în DEST.
		Acum se folosește L-ENTER
		pentru a trece sirul 'nou' în
		spațiul variabilelor.

THE 'L-ENTER' SUBROUTINE

Această scurtă subrutină este folosită pentru a trece fie o valoare numerică, din stiva calculatorului, sau un sir, din spațiul de lucru, în poziția sa corespunzătoare în spațiul variabilelor.

De aceea această subrutină este folosită pentru toate valorile numerice și sirurile, exceptînd sirurile simple 'nou declarate' și sirurile simple 'completă & existente'.

2RA6 L-ENTER	EX DE,HL	Schimbare între indicatori.
	LD A,D	Se verifică încă o dată că
	OR C	lungimea este diferită de
	RET Z	zero.
	PUSH DE	Salvare indicator destinație.
	LDIR	Se transferă valoarea
		numerică sau sirul.
	POP HL	Revenire cu registrul pereche
	RET	HL indicînd primul octet al
		valorii numerice sau al
		sirului.

THE LET SUBROUTINE CONTIUNUES HERE (SUBROUTINA LET CONTINUA AICI)

Cînd se tratătează un sir simplu 'complet & existent' noul sir este introdus ca și cum ar fi fost un sir simplu 'nou declarat' înainte ca versiunea existentă să fie cerută.

2RAF L-ADD\$	DEC HL	HL va indica litera numelui
	DEC HL	variabilei.
	DEC HL	Aceasta înseamnă DEST - 3.
	LD A,(HL)	Se extrage litera.
	PUSH HL	Se salvează indicatorul
		'versiunii existente'.
	PUSH BC	Se salvează 'lungimea'
		'sirului existent'.
	CALL 2BC6,L-STRING	Se folosește L-STRING pentru
		a adăuga noul sir spațiului
		variabilelor.
	POP BC	Se readuce 'lungimea'.
	POP HL	Se readuce indicatorul.
	INC BC	Este permis un octet pentru
	INC BC	literă și doi octeti pentru
	INC BC	lungime.
	JP 19E8,RECLAIM-2	Iesire prin salt la RECLAIM-2
		care va cere întreaga
		versiune existentă.

Sirurile simple 'nou declarate' sînt tratate după cum urmează:

2BC0 L-NEW\$	LD A,+DF	Pregătire marcarea literă
	LD HL,(DEST)	variabilă.
	AND (HL)	Se aduce indicatorul pe
		literă.
		Se marchează litera așa cum
		se cere. L-STRING se

foloseste acum pentru a adauga noul sir spatiului variabilelor.

THE 'L-STRING' SUBROUTINE

Se aduc parametrii 'noului' sir, se face accesibil suficient spatiu pentru el si apoi se transfera sirul.

2BC6 L-STRING	PUSH AF	Se salveaza litera variabilei
	CALL 2BF1,STK-FETCH	Se aduce 'inceputul' si 'lungimea' sirului 'nou'.
	EX DE,HL	Se muta 'inceputul' in HL.
	ADD HL,BC	HL va indica 'unu peste' sir.
	PUSH BC	Salvare 'lungime'.
	DEC HL	HL va indica sfirsitul sirului.
	LD (DEST),HL	Se salveaza indicatorul pentru scurt timp.
	INC BC	Este permis un octet pentru litera si doi octeti pentru lungime.
	INC BC	
	LD HL,(E-LINE)	HL va indica al '80-lea octet' la sfirsitul zonei variabilelor.
	DEC HL	
	CALL 1655,MAKE-ROOM	Acum se deschide zona variabilelor.
		Nota: De fapt 'BC' spatii sint facute inaintea inlocuirii 'octetului 80'.
	LD HL,(DEST)	Se readuce indicatorul la sfirsitul sirului 'nou'.
	POP BC	Se face o copie a lungimii sirului 'nou'.
	PUSH BC	Se aduna unu la lungime daca sirul 'nou' este un sir 'nul'.
	INC BC	Acum se copiaza 'noul' sir + un octet.
	LDDR	HL va indica octetul care contine partea cea mai semnificativa a lungimii.
	EX DE,HL	Se aduce 'lungimea'.
	INC HL	Se introduce partea cea mai semnificativa a lungimii.
	POP BC	Inapoi cu unu.
	LD (HL),B	Se introduce partea cea mai putin semnificativa a lungimii.
	DEC HL	Inapoi cu unu.
	LD (HL),C	Se introduce partea cea mai semnificativa a lungimii.
		Se aduce litera variabilei.
	POP AF	

THE 'L-FIRST' SUBROUTINE

Aceasta subrutina este introdusa cu litera variabilei, convenabil marcată, in registrul A. Litera este suprapusa peste al '80-lea octet' in spatiul variabilelor. Subrutina revine cu registrul pereche HL indicind 'noul octet 80'.

2BEA L-FIRST	DEC HL	HL va indica 'vechiul octet 80'.
	LD (HL),A	Acesta este suprapus cu litera variabilei.
	LD HL,(E-LINE)	HL va indica 'noul octet 80'.
	DEC HL	
	RET	Se termina cu toate 'variabilele nou declarate'.

THE 'STK-FETCH' SUBROUTINE

Acesta subrutina importanta colecteaza 'ultima valoare' din stiva calculatorului. Cei cinci octeti pot fi sau un numar in virgula mobilă, in forma 'lunga' sau 'scurta', sau un set de parametrii care definesc un sir.

2BF1 STK-FETCH	LD HL,(STKEND)	Se aduce STKEND.
	DEC HL	Inapoi cu unu.
	LD B,(HL)	A cincea valoare.
	DEC HL	Inapoi cu unu.
	LD C,(HL)	A patra valoare.
	DEC HL	Inapoi cu unu.
	LD D,(HL)	A treia valoare.
	DEC HL	Inapoi cu unu.
	LD E,(HL)	A doua valoare.

DEC	HL	Inapoi cu unu.
LD	A, (HL)	Prima valoare.
LD	(STKEND), HL	Se resetează STKEND pe noua sa pozitie.
RET		Sfîrsit.

THE 'DIM' COMMAND ROUTINE (RUTINA DE COMANDA 'DIM')

Această rutină stabilește noi mulțimi în spațiul variabilelor. Rutina porneste căutînd spațiul variabilelor existent pentru a determina dacă este vreo mulțime existentă cu același nume. Dacă s-a găsit o asemenea mulțime aceasta este refăcută înainte de stabilirea noii dispunerii.

O mulțime nouă va avea toate elementele setate pe zero, dacă este o mulțime numerică, sau pe 'spatii', dacă este o mulțime de siruri.

2C02 DIM	CALL	28B2, LOOK-VARS	Se caută în spațiul variabilelor.
2C,5 D-RPORT-C	JP	NZ, 1C8A, REPORT-C	Se dă prezentarea C ca și cum ar fi fost o eroare.
	CALL	2530, SYNTAX-Z	Salt înainte dacă este în 'timpul executiei'.
	JR	NZ, 2C15, D-RUN	Se verifică sintaxa pentru mulțimile sir ca și pentru cele numerice.
	RES	6, C	Se verifică sintaxa expresiilor între paranteze.
	CALL	2996, STK-VAR	Se trece la considerarea următoarei instrucțiuni dacă sintaxa a fost satisfăcătoare.
	CALL	1BEE, CHECK-END	

Se cere o 'mulțime existentă'.

2C15 D-RUN	JR	C, 2C1F, D-LETTER	Salt înainte dacă nu există nici o 'mulțime existentă'.
	PUSH	BC	Se salvează octetul discriminator.
	CALL	19B8, NEXT-ONE	Se determină începutul următoarei variabile.
	CALL	19E8, RECLAIM-2	Se reface 'dispunerea existentă'.
	POP	BC	Se reface octetul discriminator.

S-au găsit paramterii initiali ai noii mulțimi.

2C1F D-LETTER	SET	7, C	Se setează bitul 7 al octetului discriminator.
	LD	B, +00	Se pune contorul dimensiunii pe zero.
	PUSH	BC	Se salvează contorul și octetul discriminator.
	LD	HL, +0001	Registrul pereche HL va conține mărimea elementelor din mulțime, '1' pentru o mulțime de siruri/ '5' pentru o mulțime numerică.
	BIT	6, C	
	JR	NZ, 2C2D, D-SIZE	
	LD	L, +05	Mărime element în DE.
2C2D D-SIZE	EX	DE, HL	

Bucla următoare este accesată pentru fiecare dimensiune care este specificată în expresia dintre paranteze a instrucțiunii DIM. Numărul total de octeți ceruți pentru elementele mulțimii este construit în registrul pereche DE.

2C2E D-NO-LOOP	RST	0020, NEXT-CHAR	Avans CH-ADD la fiecare trecere.
	LD	H, +FF	Se setează o 'valoare-limită'.
	CALL	2ACC, INT-EXP1	Se evaluează un parametru.
	JP	C, 2A20, REPORT-3	Se dă eroare dacă este în afara intervalului.
	POP	HL	Se aduce contorul-dimensiune și octetul discriminator.
	PUSH	BC	Se salvează parametrul la fiecare trecere prin buclă.
	INC	H	Se incrementează contorul - dimensiune la fiecare trecere prin buclă.
	PUSH	HL	Se stochează din nou contorul dimensiune și octetul discriminator.
	LD	H, B	Parametrul este mutat în

LD	L,C	registru pereche HL.
CALL	2AF4,GET-HL*DE	Octetul total este construit
EX	DE,HL	în HL și apoi este transferat
		în DE.
RST	0018,GET-CHAR	Se aduce prezentul caracter
CP	+2C	și se cicleează din nou bucla
JR	7,2C2E,D-ND-LOOP	dacă mai există o altă
		dimensiune.

Notă: În acest punct registrul pereche DE indică numărul de octeți cerut pentru elementele noii mulțimi și mărimea fiecărei dimensiuni este stocată în stiva mașinii.

Acum se verifică dacă este într-adevăr o virgulă care închide la expresia dintre paranteze.

CP	+29	Este o ')?
JR	NZ,2C05,D-RPORT-C	Salt înapoi dacă nu este.
RST	0020,NEXT-CHAR	Avans CH-ADD peste acesta.

Acum este permisă aducerea mărimilor dimensiune.

POP	BC	Se aduce contorul dimensiune
		și octetul discriminator.
LD	A,C	Se trece octetul în registrul
		A octetul discriminator
		pentru mai târziu.
LD	L,B	Se mută contorul în L.
LD	H,+00	Se șterge registrul H.
INC	HL	Se incrementează contorul -
INC	HL	dimensiune cu doi și se
ADD	HL,HL	dublează rezultatul și se
ADD	HL,DE	formează lungimea totală
		corectă prin adăugarea
		elementului octet total.
JR	C,IF15,REPORT-4	Se dă prezentarea 'Afară din
		memorie' dacă se cere.
PUSH	DE	Salvare element octet total.
PUSH	BC	Se salvează contorul -
		dimensiune și octetul
		discriminator.
PUSH	HL	Se salvează lungimea totală.
LD	B,H	Se mută lungimea totală în BC
LD	C,L	

Spatiul total cerut este făcut accesibil pentru noua mulțime la sfârșitul spațiului variabilelor.

LD	HL,(E-LINE)	Registrul pereche HL va
DEC	HL	indica 'bitul 80'.
CALL	1655,MAKE-ROOM	Spatiul este făcut accesibil.
INC	HL	HL va indica prima locație
		nouă.

Acum se introduc parametrii.

LD	(HL),A	Litera, convenabil marcată,
		se introduce prima.
POP	BC	Se aduce lungimea totală și
DEC	BC	se micșorează cu '3'.
DEC	BC	
DEC	BC	
INC	HL	Avans HL.
LD	(HL),C	Se introduce partea cea mai
		putin semnificativă a
		lungimii.
INC	HL	Avans HL.
LD	(HL),B	Se introduce partea cea mai
		semnificativă a lungimii.
POP	BC	Se aduce contorul-dimensiune.
LD	A,B	Acesta se pune în registrul A
INC	HL	Avans HL.
LD	(HL),A	Se introduce contorul -
		dimensiune.

Acum se 'șterg' elementele noii mulțimi.

LD	H,D	HL va indica ultima locație a
LD	L,E	mulțimii iar DE locația de
DEC	DE	dinainte de aceasta.

	LD	(HL),+00	Se introduce un zero în ultima locație dar se suprapune un 'spatiu' dacă se lucrează cu o multime de siruri.
	BIT	6,C	
	JR	Z,2C7C,DIM-CLEAR	
	LD	(HL),+20	
2C7C DIM-CLEAR	POP	BC	Se aduce întregul octet al elementului.
	LDDR		Se șterge multimea + o locație aditională.

Acum se introduc 'mărimile dimensiune'.

2C7F DIM-SIZES	POP	BC	Se aduce o mărime dimensiune.
	LD	(HL),B	Se introduce octetul cel mai semnificativ.
	DEC	HL	Înapoi cu unu.
	LD	(HL),C	Se introduce octetul mai puțin semnificativ.
	DEC	HL	Înapoi cu unu.
	DEC	A	Se decrementează contorul dimensiune.
	JR	NZ,2C7F,DIM-SIZES	Se repetă operația pînă cînd s-au luat în considerare toate dimensiunile; apoi se revine.
	RET		

THE 'ALPHANUM' SUBROUTINE

Această subrutină revine cu fanionul de transport setat dacă valoarea prezentă a registrului A denotă o cifră sau o literă validă.

2C88 ALPHANUM	CALL	2D1B,NUMERIC	Testare cifră; transportul va fi resetat dacă este o cifră.
	CCF		Se completează fanionul de transport.
	RET	C	revenire dacă este cifră; altfel se continuă cu ALPHA.

THE 'ALPHA' SUBROUTINE

Această subrutină revine cu fanionul de transport setat dacă valoarea prezentă în registrul A denotă o literă validă din alfabet.

2C8D ALPHA	CP	+41	Se compară cu 41 hex. codul lui 'A'.
	CCF		Se completează fanionul de transport.
	RET	NC	Revenire dacă nu este un cod caracter valid.
	CP	+5B	Se compară cu 5B hex., 1 mai mult decît codul pentru 'Z'.
	RET	C	Revenire dacă este majusculă.
	CP	+61	Se compară cu 61 hex., codul pentru 'a'.
	CCF		Se completează fanionul de transport.
	RET	NC	Revenire dacă nu este codul unui caracter valid.
	CP	+7B	Se compară cu 7B hex., cu 1 mai mult decît codul pentru 'z'.
	RET		Sfîrșit.

THE 'DECIMAL TO FLOATING POINT' SUBROUTINE (SUBROUTINA 'ZECIMAL IN VIRGULA MOBILA')

Ca parte a verificării sintaxei, numerele zecimale care intervin într-o linie BASIC sînt convertite în forma lor în virgulă mobilă. Această subrutină citește numărul zecimal cifră cu cifră și dă rezultatul ca o 'ultimă valoare' în stiva calculatorului. Dar mai întîi ea lucrează cu notația alternativă BIN, care introduce o secevntă de 0 și 1 dînd reprezentarea binară a numărului cerut.

2C9B DEC-TO-FP	CP	+C4	Este caracterul un 'BIN'?
	JR	NZ,2C88,NOT-BIN	Salt dacă nu este 'BIN'.
	LD	DE,+0000	Initializare rezultat pe zero în DE.
2CA2 BIN-DIGIT	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	SUB	+31	Se scade '1' din codul caracterului.
	ADC	A,+00	0 dă acum 0 cu transportul setat; 1 dă 0 cu transportul

	JR	NZ,2CB3,BIN-END	resetat. Orice alt caracter cauzează salt la BIN-END și se va verifica sintaxa în timpul sau după baleiere. Deocamdată rezultatul este în HL.
	EX	DE,HL	Se completează fanionul de transport.
	CCF		Se deplasează rezultatul la stînga, cu transportul trecînd în bitul 0.
	ADC	HL,HL	Se prezintă depășire dacă este mai mare decît 65535.
	JP	C,3IAD,REPORT-6	Se readuce deocamdată rezultatul în DE.
	EX	DE,HL	Salt înapoi pentru următorul 0 sau 1.
	JR	2CA2,BIN-DIGIT	Se copiază rezultatul în BC pentru a fi stocat.
2CB3 BIN-END	LD	B,D	Salt înainte pentru a stoca rezultatul.
	LD	C,E	
	JP	2D2B,STACK-BC	
Pentru celelalte numere, mai întîi se converteste partea întregă; dacă următorul caracter este unul zecimal, atunci se consideră fracție zecimală.			
2CB8 NOT-BIN	CP	+2E	Este primul caracter un '.'?
	JR	Z,2CCB,DECIMAL	Salt mai departe dacă este.
	CALL	2D3B,INT-TO-FP	Altfel, se formează o 'ultima valoare' a întregului.
	CP	+2E	Este următorul caracter un '.'?
	JR	NZ,2CEB,E-FORMAT	Salt înainte pentru a vedea dacă este un 'E'.
	RST	0020,NEXT-CHAR	Se aduce următorul caracter.
	CALL	2D1B,NUMERIC	Este o cifră?
	JR	C,2CEB,E-FORMAT	Salt dacă nu (De exemplu este permis 1.E4).
	JR	2CD5,DEC-STO-1	Salt înainte pentru a prelucra cifrele de după virgulă.
2CCB DECIMAL	RST	0020,NEXT-CHAR	Dacă numărul începe cu un zecimal, se verifică dacă următorul caracter este o cifră.
	CALL	2D1B,NUMERIC	Dacă nu este, se dă eroare.
2CCF DEC-RPT-C	JP	C,1C8A,REPORT-C	Se utilizează calculatorul pentru a stoca zero ca parte întregă a unui astfel de număr.
	RST	0028,FP-CLAC	Utilizare calculator.
	DEFB	+A0,stk-zero	Se caută forma în virgulă mobilă a numărului zecimal '1' și se salvează în spațiul memoriei.
	DEFB	+38,end-calc	Se aduce caracterul prezent.
2CD5 DEC-STO-1	RST	0028,FP-CALC	Acesta se stochează dacă este cifră.
	DEFB	+A1,stk-one	Dacă nu este, salt înainte.
	DEFB	+C0,stk-mem-0	Utilizare calculator.
	DEFB	+02,delete	Pentru fiecare trecere prin buclă, numărul salvat în spațiul memoriei este adus, împărțit cu 10 și restocat; aceasta înseamnă plecînd de la .1 la .01 la .001 etc.
2CDA NXT-DGT-1	DEFB	+38,end-calc	Cifra prezentă este înmulțită cu 'numărul' salvat și adunată cu 'ultima valoare'.
	RST	0018,BET-CHAR	Se aduce următorul caracter.
	CALL	2D22,STK-DIGIT	Salt înapoi pentru a-l lua în considerare (cu un octet mai mult decît este necesar).
	JR	C,2CEB,E-FORMAT	
	RST	0028,FP-CALC	
	DEFB	+E0,get-mem-0	
	DEFB	+A4,stk-ten	
	DEFB	+05,division	
	DEFB	+C0,stk-mem-0	
	DEFB	+04,multiply	
	DEFB	+0F,addition	
	DEFB	+38,end-calc	
	RST	0020,NEXT-CHAR	
	JR	2CDA,NXT-DGT-1	

În continuare se consideră orice 'notatie E', de exemplu forma xEm sau xem, unde m un întreg pozitiv sau negativ.

2CEB E-FORMAT CP +45 Este caracterul prezent un 'E'?

	JR	Z, 2CF2, SIGN-FLAG	Salt înainte dacă este.
	CP	+65	Este un 'e'?
	RET	NZ	Da, terminare.
2CF2 SIGN-FLAG	LD	R, +FF	Se folosește B ca un fanion de semn, FF pentru +.
	RST	0020, NEXT-CHAR	Se aduce următorul caracter.
	CP	+2B	Este un '+'?
	JR	Z, 2CFE, SIGN-DONE	Salt înainte.
	CP	+2D	Este '-'?
	JR	NZ, 2CFF, ST-E-PAT	Salt dacă nu este nici '+' nici '-'.
	INC	B	Se schimbă semnul fanionului.
2CFE SIGN-DONE	RST	0020, NEXT-CHAR	Se indică prima cifră.
2CFF ST-E-PART	CALL	2D1B, NUMERIC	Este într-adevăr o cifră?
	JR	C, 2CCF, DEC-RPT-C	Se prezintă eroare dacă nu este.
	PUSH	BC	Se salvează fanionul în B pentru scurt timp.
	CALL	2D3B, INT-TO-FP	Se stochează ABS m, unde m este exponentul.
	CALL	2DD5, FP-TO-A	Se transferă ABS m în A.
	POP	BC	Se reface fanionul de semn în B.
	JP	C, 31AD, REPORT-6	Acum se prezintă depășire dacă ABS m este mai mare decât 255 sau într-adevăr mai mare decât 127 (celelalte valori mai mari decât 39 vor fi detectate ai târziu).
	AND	A	Se testează fanionul de semn în B: '+' (de fapt +FF) va seta acum fanionul de zero.
	JP	M, 31AD, REPORT-6	Salt dacă semnul lui m este '+'.
	INC	B	Se face negarea lui m dacă semnul lui este '-'.
	JR	Z, 2D1B, E-FP-JUMP	Salt pentru a atribui 'ultimei valori' rezultatul lui $x \times 10^m$.
2D1B E-FP-JUMP	JP	2D4F, E-TO-FP	

THE 'NUMERIC' SUBROUTINE (SUBROUTINA 'NUMERIC')

Această subrutină revine cu fanionul de transport resetat dacă valoarea prezentă în registrul A denotă o cifră validă.

2D1B NUMERIC	CP	+30	Se compară cu 30 hex, codul pentru '0'.
	RET	C	Revenire dacă nu este codul unui caracter valid.
	CP	+3A	Se compară cu limita superioară.
	CCF		Se complementează fanionul de transport.
	RET		Sfîrsit.

THE 'STK-DIGIT' SUBROUTINE (SUBROUTINA 'STK-DIGIT')

Această subrutină revine imediat dacă valoarea curentă continuată în registrul A nu reprezintă o cifră, dar dacă reprezintă, atunci forma în virgulă mobilă a cifrei devine 'ultima valoare' din stiva calculatorului.

2D22 STK-DIGIT	CALL	2D1B, NUMERIC	Este caracterul o cifră?
	RET	C	Revenire dacă nu este în interval.
	SUB	+30	Se înlocuiește codul cu cifra actuală.

THE 'STACK-A' SUBROUTINE (SUBROUTINA 'STACK-A')

Această subrutină dă forma în virgulă mobilă pentru valoarea binară absolută curentă continuată în registrul A.

2D28 STACK-A	LD	C, A	Se transferă valoarea în registrul C.
	LD	B, +00	Se șterge registrul B.

THE 'STACK-BC' SUBROUTINE (SUBROUTINA 'STACK-BC')

Această subrutină dă forma în virgulă mobilă pentru valoarea binară absolută curentă continuată în registrul BC.

Forma utilizată în aceasta și deci la fel de bine în două subrutine anterioare este aceea utilizată în Spectrum pentru numere întregi mici n, unde $-65535 \leq n \leq 65535$. Octetii unu și cinci sînt zero; al treilea și patrulea

octet sînt octetii cei mai puțin și cei mai semnificativi octeti ai celor 16 biți de întreg n în complement față de doi (dacă n este negativ, acești doi octeti vor conține 65536+n); și al doilea octet este un octet de semn, 00 pentru + și FF pentru '-'.

2D2B STACK-BC	LD	IY,+5C3A	Reinitializare IY pe ERR-NR.
	XOR	A	Se șterge registrul A.
	LD	E,A	Și registrul E, pentru a indica '+'. Se copiază octetul cel mai puțin semnificativ în D.
	LD	D,C	Și octetul cel mai semnificativ în C.
	LD	B,A	Se șterge registrul B.
	CALL	2AB6,STK-STORE	Acum se stochează numărul.
	RST	0028,FP-CALC	HL va indica STKEND - 5.
	DEFB	+38,end-calc	
	AND	A	Se șterge fanionul de transport.
	RET		Sfîrșit.

THE 'INTEGER TO FLOATING-POINT' SUBROUTINE (SUBROUTINA 'INTREG IN VIRGULA MOBILA')

Această subrutină revine cu 'ultima valoare' din stiva calculatorului ca rezultat al conversiei unui întreg dintr-o linie BASIC, adică a părții întregi a unui număr zecimal sau a unui număr de linie în forma sa în virgulă mobilă.

Prin apelarea repetată a lui CH-ADD+1 se aduce pe rînd fiecare cifră a întregului. Se iese atunci cînd s-a adus un cod care nu reprezintă o cifră.

2D3R INT-TO-FP	PUSH	AF	Se salvează prima cifră în A.
	RST	0028,FP-CALC	Utilizare calculator.
	DEFB	+A0,stk-zero	Acum 'ultima valoare' este zero.
	DEFB	+38,end-calc	
	POP	AF	Se readuce prima cifră.

Acum se introduce o buclă. Atîta timp cît codul reprezintă o cifră se găsește și se stochează sub 'ultima valoare'. Apoi 'ultima valoare' este multiplicată cu 10 zecimal și adunată cu 'cifra' pentru a forma noua 'ultima valoare' care este transportată înapoi la începutul buclei.

2D40 NXT-DGT-2	CALL	2D22,STK-DIGIT	Dacă codul reprezintă o cifră, atunci se stochează forma în virgulă mobilă.
	RET	C	Se utilizează calculatorul.
	RST	0028,FP-CALC	'Cifra' trece sub 'ultima valoare'.
	DEFB	+01,exchange	Se definește zecimalul 10.
	DEFB	+A4,stk-ten	'Ultima valoare' = 'ultima valoare' * 10
	DEFB	+04,multiply	'Ultima valoare' = 'ultima valoare' + 'cifra'
	DEFB	+0F,addition	
	DEFB	+38,end-calc	
	CALL	0074,CH-ADD+1	Următorul cod trece în registrul A.
	JR	2D40,NXT-DGT-2	Șalt înapoi cu acest cod.

THE ARITHMETIC ROUTINES (RUTINELE ARITMETICE)

THE 'E-FORMAT TO FLOATING-POINT' SUBROUTINE (SUBROUTINA 'E - FORMAT IN VIRGULA MOBILA')

(Deplasament 3C - vezi CALCULATE după: 'E-to-fp')

Această subrutină dă 'ultima valoare' din vârful stivei calculatorului care este rezultatul conversiei numărului în forma xEm , unde m este un întreg pozitiv sau negativ. Subrutina este introdusă cu x în vârful stivei calculatorului și cu m în registrul A.

Metoda care se folosește este de a afla valoarea absolută a lui m , numită p , și de a înmulți sau a împărți x cu 10^p în concordanță cu faptul că m este pozitiv sau negativ.

Pentru a realiza acest lucru, p este deplasat la dreapta pînă cînd devine zero, și x este înmulțit sau împărțit cu $10^{(2^n)}$ pentru fiecare bit setat $b(n)$ al p . Pînă cînd p nu este niciodată mai mare decît 39 zecimal, bitii 6 și 7 ai lui p nu vor fi în mod normal setați.

2D4F E-TO-FP	RLCA RLCA		Se testează semnul lui m rotind bitul 7 din A în transport fără a-l schimba pe A.
	JR CPL INC	NC, 2D55, E-SAVE A	Salt dacă m este pozitiv. Se face negarea lui m din A fără a distruge fanionul de transport.
2D55 E-SAVE	PUSH LD CALL	AF HL, +5C92 350R, FP-0/1	Se salvează m în A pentru scurt timp. Este MEMBOT: un fanion de semn este acum stocat în primul octet al lui mem-0, adică 0 pentru '+' și '1' pentru '-'.
	RST DEFB DEFB POP SRL	0028, FP-CALC +A4, stk-ten +38, end-calc AF A	Stiva conține x . $x, 10$ (zecimal). $x, 10$ Se readuce m în A.
2D60 E-LOOP	JR PUSH RST	NC, 2D71, E-TST-END AF 0028, FP-CALC	În buclă, se deplasează afară următorul bit al lui m , modificînd corespunzător fanionul zero și de transport; salt dacă transportul este resetat.
	DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB	+C1, st-mem-1 +E0, get-mem-0 +00, jump-true +04, to E-DIVSN +04, multiply +33, jump +02, to E-FETCH +05, division	Se salvează restul din m și fanioanele. Stiva conține x' și $10^{(2^n)}$, unde x' este o etapă intermediară în multiplicarea lui x cu 10^m , și $n=0, 1, 2, 3, 4$ sau 5. $(10^{(2^n)})$ este copiat în mem-1.) $x', 10^{(2^n)}, (1/0)$ $x', 10^{(2^n)}$ $x', 10^{(2^n)}$ $x' * 10^{(2^n)} = x''$ x'' x'' $x'' / 10^{(2^n)} = x'''$ (x''' este $N * 10^{(2^n)}$ sau $x'' / 10^{(2^n)}$ în concordanță cu faptul că m este '+' sau '-') $x'', 10^{(2^n)}$ $x'', 10^{(2^n)}$
2D6D E-DIVSN	JR PUSH RST DEFB DEFB DEFB DEFB DEFB DEFB	Z, 2D7B, E-END AF 0028FP-CALC +31, duplicate +04, multiply +38, end-calc POP AF 2D60, E-LOOP	Se reface restul din m în A, și fanioanele. Salt dacă m a fost redus la zero. Se salvează restul lui m în A $x'', 10^{(2^n)}$ $x'', 10^{(2^n)}, 10^{(2^n)}$ $x'', 10^{(2^{(n+1)})}$ $x'', 10^{(2^{(n+1)})}$ Se readuce restul lui m în A. Salt înapoi pentru toți bitii lui m .
2D6E E-FETCH	DEFB DEFB POP	+E1, get-mem-1 +38, end-calc AF	
2D71 E-TST-END	JR PUSH RST DEFB DEFB DEFB POP JR	Z, 2D7B, E-END AF 0028FP-CALC +31, duplicate +04, multiply +38, end-calc POP AF 2D60, E-LOOP	
2D7B E-END	RST DEFB	0028, FP-CALC +02, delete	Se folosește calculatorul pentru a șterge puterea

DEFB +38,end-calc
RET

finală a lui 10 la care s-a
ajuns, lăsând în stivă
'ultima valoare' $x \cdot 10^m$.

THE 'INT-FETCH' SUBROUTINE (SUBROUTINA 'INT-FETCH')

Această subrutină colectează în DE un număr întreg n ($-65535 \leq n \leq 65535$) din locația adresată de HL; aceasta înseamnă că n este în mod normal primul (sau al doilea) număr din vârful stivei calculatorului; dar HL poate de asemenea accesa (prin schimbare cu DE) un număr care a fost sters din stivă.

Subrutina nu șterge ea însăși numărul din stivă sau din memorie; ea revine cu HL indicând al patrulea octet al numărului în poziția sa inițială.

2D7F INT-FETCH	INC	HL	Se indică octetul de semn al numărului.
	LD	C, (HL)	Se copiază octetul de semn în C.

Mecanismul următor va complementa față de 2 numărul dacă este negativ (C este FF) dar îl va lăsa neschimbat dacă este pozitiv (C este 00).

INC	HL	Se indică octetul cel mai puțin semnificativ.
LD	A, (HL)	Se colectează octetul în A.
XOR	C	Dacă este negativ se face complementul său față de 1.
SUB	C	Aceasta adună 1 pentru numere negative; se setează transportul numai dacă octetul a fost 0.
LD	E, A	Acum se pune octetul cel mai puțin semnificativ în E.
INC	HL	Se indică octetul cel mai semnificativ.
LD	A, (HL)	Acesta se colectează în A.
ADC	A, C	Se termină complementarea față de doi în cazul unui număr negativ; de notat că transportul este întodeauna lăsat resetat.
XOR	C	Acum se trece octetul cel mai semnificativ în D.
LD	D, A	Sfârșit.
RET		

THE 'INT-STORE' SUBROUTINE (SUBROUTINA 'INT-STORE')

Această subrutină stochează un număr întreg mic n ($-65535 \leq n \leq 65535$) în locația adresată de HL și următoarele patru locații; aceasta înseamnă că se înlocuiește primul (sau al doilea) număr din vârful stivei calculatorului.

Subrutina revine cu HL indicând primul octet al lui n din stivă.

2D8C P-INT-STO	LD	C, +00	Acest punct de intrare poate stoca un număr știut ca fiind pozitiv.
2D8E INT-STORE	PUSH	HL	Este salvat indicatorul primei locații.
	LD	(HL), +00	Primul octet este setat pe zero.
	INC	HL	Se indică a doua locație.
	LD	(HL), C	Se introduce octetul de semn.

Acelasi mecanism este folosit acum ca în 'INT-FETCH' pentru a complementa față de doi numere negative. Aceasta este necesar, de exemplu, înainte și după înmulțirea numerelor întregi mici. Adunarea este oricum efectuată fără vreo complementare față de doi înainte sau după aceea.

INC	HL	Se indică a treia locație.
LD	A, E	Se colectează cel mai puțin semnificativ octet.
XOR	C	Dacă numărul este negativ se face complementul față de doi.
SUB	C	Se stochează octetul.
LD	(HL), A	Se indică a patra locație.
INC	HL	Se colectează octetul cel mai semnificativ.
LD	A, D	Dacă numărul este negativ se face complementul față de doi.
ADC	A, C	Se stochează octetul.
XOR	C	Se indică a cincea locație.
LD	(HL), A	
INC	HL	

LD (HL),+00
POP HL
RET

Al cincilea octet este setat pe zero.
Se revine cu HL indicînd primul octet al lui n din stivă.

THE 'FLOATING-POINT TO BC' SUBROUTINE (SUBROUTINA 'VIRGULA MOBILA IN BC')

Această subrutină este apelată din patru locuri diferite pentru diferite situații și este folosită pentru a comprima 'ultima valoare' în virgulă mobilă în registrul pereche BC.

Dacă rezultatul este prea mare, adică mai mare decît 65535 zecimal, atunci subrutina revine cu fanionul de transport setat. Dacă 'ultima valoare' este negativă atunci fanionul zero este resetat.

Octetul cel mai puțin semnificativ al rezultatului este copiat în registrul A.

2DA2 FP-TO-BC

RST 0028,FP-CALC
DEFB +38,end-calc

LD A,(HL)
AND A
JR Z,2DAD,FP-DELETE

Se utilizează calculatorul pentru a face ca HL să indice STKEND - 5.

Se colectează octetul exponent al 'ultimei valori'; salt dacă este zero, el indicînd un 'număr întreg mic'.

RST 0028,FP-CALC
DEFB +A2,stk-half
DEFB +0F,addition
DEFB +27,int
DEFB +38,end-calc

Acum se utilizează calculatorul pentru a rotunji 'ultima valoare' la cel mai apropiat întreg, care de asemenea se schimbă în forma 'întregă cea mai mică' din stiva calculatorului dacă acest lucru este posibil, adică $-65535.5 < x < 65535.5$.

2DAD FP-DELETE

RST 0028,FP-CALC
DEFB +02,delete
DEFB +38,end-calc

PUSH HL
PUSH DE
EX DE,HL
LD B,(HL)
CALL 2D7F,INT-FETCH

Se folosește calculatorul pentru a șterge întregul din stivă; DE încă îl indică în memorie (la STKEND).

Se salvează ambii indicatori.

XOR A
SUB B

BIT 7,C

Acum HL indică numărul.

Se copiază primul octet în B. Se copiază octetii 2, 3 și 4 în C, E și D.

Se șterge registrul A.

Aceasta setează transportul numai dacă B este zero.

Aceasta setează fanionul zero numai dacă numărul este pozitiv (NZ denotă negativ).

Se copiază octetul cel mai semnificativ în B.

Si octetul cel mai puțin semnificativ în C.

Octetul cel mai puțin semnificativ este copiat și în A.

Se readuc indicatorii stivă.

POP DE
POP HL
RET

Sfîrsit.

THE 'LOG (2^A)' SUBROUTINE (SUBROUTINA 'LOG (2 ^ A)

Această subrutină este apelată de subrutina 'PRINT-FP' pentru a calcula numărul aproximativ de cifre dinaintea punctului zecimal în x, numărul de tipărit, sau, dacă nu sînt cifre înaintea punctului zecimal, atunci numărul aproximativ de zerouri de fond după punctul zecimal. Ea este introdusă cu registrul A conținînd e', exponentul real al lui x, sau e⁻², și calculează $z = \log$ în baza 10 din (2^A). Apoi îl setează pe A egal cu ABS INT (z+0.5), cum se cere, folosind în acest scop FP-TO-A.

2DC1 LOG (2 A)

LD D,A
RLA
SBC A,A

LD E,A
LD C,A
XOR A
LD B,A

CALL 2AB6,STK-STORE

Întregul A este stocat, fie ca 00 00 A 00 00 (pentru A pozitiv) sau ca 00 FF A FF 00 (pentru A negativ).

Acești octeți sînt mai întîi încărcati în A, E, D, C, B și apoi STKEND este apelat pentru a pune numărul în stiva calculatorului.

RST	0028,FP-CALC	Utilizare calculator.
DEFB	+34,stk-data	Acum este stocat log 2 în baza 10.
DEFB	+EF,exponent+7F	Acum stiva contine A, log 2.
DEFB	+1A,+20,+9A,+85	
DEFB	+04,multiply	A*log 2, adică log (2^A).
DEFB	+27,int	INT log (2^A)
DEFB	+38,end-calc	

Subrutina continuă în FP-TO-A pentru a completa calculul.

THE 'FLOATING-POINT TO A' SUBROUTINE (SUBROUTINA 'A IN VIRGULA MOBILA')

Această subrutină scurtă dar vitală este apelată de cel puțin 8 ori pentru diferite scopuri. Ea folosește penultima subrutină, FP-TO-BC, pentru a aduce 'ultima valoare' în registrul A unde este posibil. De aceea ea testează dacă modulul numărului rotunjit este mai mare decât 255 și dacă a realizat revenirea subrutinei cu fanionul de transport setat. Altfel ea revine cu modulul numărului, rotunjit la cel mai apropiat întreg, în registrul A, și fanionul zero setat implică faptul că numărul a fost pozitiv, sau resetat, ceea ce implică faptul că a fost negativ.

2DD5 FP-TO-A	CALL	2DA2,FP-TO-BC	Se comprimă 'ultima valoare' în BC.
	RET	C	Se revine dacă s-a ieșit deja din interval.
	PUSH	AF	Se salvează rezultatul și fanioanele.
	DEC	B	Din nou se va ieși din interval dacă registrul B nu conține zero.
	INC	B	
	JR	Z,2DE1,FP-A-END	Salt dacă este în interval.
	POP	AF	Se aduce rezultatul și fanioanele.
	SCF		Se semnalează că rezultatul este afară din interval.
	RET		Sfîrsit - nereușită.
2DE1 FP-A-END	POP	AF	Se aduce rezultatul și fanioanele.
	RET		Sfîrsit - reușită.

THE 'PRINT A FLOATING-POINT NUMBER' SUBROUTINE (SUBROUTINA 'TIPĂRIRE NUMAR IN VIRGULA MOBILA')

Această subrutină este apelată de rutina de comandă PRINT la 2039 și de STR\$ la 3630, ea convertind într-un șir numărul așa cum va fi tipărit. Subrutina tipărește x, 'ultima valoare' din stiva calculatorului. Formatul pentru tipărire nu ocupă niciodată mai mult de 14 spații.

Cele mai semnificative 8 cifre ale lui x, rotunjit corect, sînt stocate într-un buffer de tipărire ad-hoc în mem-3 și mem-4. Numerele mici, numeric mai mici decât 1, și numerele mari, numeric mai mari decât 2^{27} , sînt prelucrate separat. Numărul anterior este înmulțit cu 10^n , unde n este numărul aproximativ de zerouri de fond după punctul zecimal, pe cînd cel recent se împarte prin $10^{(n-7)}$, unde n este numărul aproximativ de cifre înainte de punctul zecimal. Aceasta aduce toate numerele într-un interval de mijloc, și numărul de cifre cerute înainte punctului zecimal este construit în al doilea octet al lui mem-5. În final se efectuează tipărirea, folosind formatul E dacă sînt mai mult de 8 cifre înainte de punctul zecimal, sau, pentru numere mici, mai mult de 4 zerouri de fond după punctul zecimal.

Următorul program arată ordinea de tipărire a formelor:

```
10 FOR a = -11 TO 12: PRINT $6N a*9^a.: NEXT a
```

i. În primul rînd semnul lui x tine cont de:

Dacă x este negativ, subrutina sare la PF-NEGATIVE, ia ABS x și tipărește semnul minus.

Dacă x este zero, x este sters din stiva calculatorului, se tipărește un zero și se revine din subrutină.

Dacă x este pozitiv, subrutina continuă.

2DE3 PRINT-FP	RST	0028,FP-CALC	Utilizare calculator.
	DEFB	+31,duplicate	x,x
	DEFB	+36,less-0	x,(1/0) Valoarea logică a lui x.
	DEFB	+00,jump-true	x
	DEFB	+0B,to PF-NEGATIVE	x
	DEFB	+31,duplicate	x,x
	DEFB	+37,greater-0	x,(1/0) Valoarea logică a lui x.
	DEFB	+00,jump-true	x
	DEFB	+0B,to PF-POSTIVE	x De aici încolo x'=ABS x
	DEFB	+02,delete	-

	DEFB +38,end-calc	-
	LD A,+30	Se introduce codul caracter pentru '0'.
	RST 0010,PRINT-A-1	Se tipărește '0'.
	RET	Se termină când 'ultima valoare' este zero.
2DF2 PF-NEGTV	DEFB +2A,abs	x' x'=ABS x
	DEFB +38,end-calc	x'
	LD A,+2D	Se introduce codul caracter pentru '-'.
	RST 0010,PRINT-A-1	se tipărește '-'.
	RST 0028,FP-CALC	Se folosește din nou calculatorul.
2DF3 PF-POSTVE	DEFB +A0,stk-zero	Cei 15 octeți ai lui mem-3, mem-4 și mem-5 sînt acum inițializați pe zero pentru a fi folosiți ca buffer pentru tipărire și două contoare.
	DEFB +C3,stk-mem-3	Stiva este stearsă, cu excepția lui x'.
	DEFB +C4,stk-mem-4	x'
	DEFB +C5,stk-mem-5	H'L', care este folosit pentru a ține deplasamentul calculatorului (de exemplu 'STR\$') este salvat în stiva mașinii.
	DEFB +02,delete	
	DEFB +38,end-calc	
	EXX	
	PUSH HL	
	EXX	

ii. Acesta este începutul unei bucle care prelucrează numere mari. Oricum fiecare număr x este mai întâi despărțit în partea sa întreagă i și în partea sa fracționară f. Dacă i este un întreg mic, adică dacă $-65535 \leq i \leq 65535$, el este stocat în DE pentru inserarea în bufferul de tipărire.

2E01 PF-LOOP	RST 0028,FP-CALC	Utilizare calculator.
	DEFB +31,duplicate	x',x'
	DEFB +27,int	x', INT (x')=i
	DEFB +C2,stk-mem-2	(i este stocat în mem-2)
	DEFB +03,subtract	x'-i=f
	DEFB +E2,get-mem-2	f,i
	DEFB +01,exchange	i,f
	DEFB +C2,stk-mem-2	(f este stocat în mem-2)
	DEFB +02,delete	i
	DEFB +38,end-calc	i
	LD A,(HL)	Este i un întreg mic (primul octet zero), adică ABS i \leq 65535?
	AND A	Salt dacă nu este.
	JR NZ,2E56,PF-LARGE	i este copiat în DE (i, ca și x', ≥ 0)
	CALL 2D7F,INT-FETCH	B este setat pentru a număra 16 biti.
	LD B,+10	D este copiat în A pentru a fi testat:
	LD A,D	este zero?
	AND A	Salt dacă nu este zero.
	JR NZ,2E1E,PF-SAVE	Acum se testează E.
	OR E	Salt dacă E este zero: x este o fracție curată.
	JR Z,2E24,PF-SMALL	Se mută E în D și se setează B pentru 8 biti: D a fost zero iar E nu a fost.
	LD D,E	Se transferă DE în D'E', prin stiva mașinii, pentru a fi mutați în bufferul de tipărire la PF-BITS.
	LD B,+08	Salt înainte.
2E1E PF-SAVE	PUSH DE	
	EXX	
	POP DE	
	EXX	
	JR 2E7B,PF-BITS	

iii. Frațiile curate sînt înmultite cu 10^n , unde n este numărul aproximat de zerouri de fond după punctul zecimal; și -n este adunat la al doilea octet al lui mem-5, care conține numărul de cifre necesare înainte de punctul zecimal; un număr negativ aici este indicat prin zerouri de fond după punctul zecimal.

2E24 PF-SMALL	RST 0028,FP-CALC	i (i= zero aici)
	DEFB +E2,get-mem-2	i,f
	DEFB +38,end-calc	i,f

De notat că stiva acum este nesimetrică. Este necesar un octet additional 'DEFB +02,delete' la 2E25, imediat după RST 0028. Acum, o expresie ca "2" +STR\$ 0.5, este evaluată incorect ca 0.5; zeroul din stînga din stivă înlocuiește "2" și este tratată ca un sir invalid. Analog toate comparațiile sirului pot

produce valori incorecte dacă al doilea sir ia forma STR# x, unde x este numeric mai mic decât 1, de exemplu expresia "50"<STR# 0.1 produce valoarea logică "adevărat"; din nou "" este folosit în loc de "50".

LD	A, (HL)	Octetul exponent al lui f este copiat în A.
SUB	+7E	A devine e-126 zecimal, adică e'+2, unde e' este exponentul real al lui f.
CALL	2DC1, LOG(2^A)	Este realizată construcția A = ABS INT (LOG (2^A)) (LOG este în baza 10); asta înseamnă A=n, unde n este copiat din A în D.
LD	D, A	Contorul curent este colectat din al doilea octet al lui mem-5 și n este scăzut din el.
LD	A, (mem-5-2nd)	n este copiat din D în A.
SUB	D	y=f*10^n este format și stocat.
LD	(mem-5-2nd), A	
LD	A, D	
CALL	2D4F, E-TO-FP	
RST	0028, FP-CALC	i, y
DEFB	+31, duplicate	i, y, y
DEFB	+27, int	i, y, INT (y) = i2
DEFB	+C1, st-mem-1	(i2 este copiat în mem-1)
DEFB	+03, subtract	i, y-i2
DEFB	+E1, get-mem-1	i, y-i2, i2
DEFB	+38, end-calc	i, i2 (f2 = y - i2)
CALL	2DD5, FP-TO-A	i2 este transferat din stivă în A.
PUSH	HL	Indicatorul lui f2 este salvat.
LD	(mem-3-1st), A	i2 este stocat în primul octet al lui mem-3; o cifră pentru tipărire.
DEC	A	i2 nu va număra cifra pentru tipărire dacă ea este zero; A este modificat astfel încât zero va produce zero dar cifră diferită de zero y produce 1.
RLA		Zero sau unu este inserat în primul octet al lui mem- (numărul de cifre pentru tipărire) și adunat cu a doilea octet al lui mem-5. (numărul de cifre dinainte de punctul zecimal).
SBC	A, A	Se readuce indicatorul lui f2.
INC	A	
LD	HL, +5CAB	
LD	(HL), A	
INC	HL	
ADD	A, (HL)	
POP	HL	
JP	2ECF, PF-FRACTN	Salt pentru a stoca f2 în buffer (acum HL indică f2, iar DE indică i2).

iv. Numerele mai mari decât 2^{27} sînt multiplicat similar de $2^{-(n+7)}$, reducînd numărul cifrelor dinainte de punctul zecimal la 8, și bucla reintră la PF-LOOP.

2E56 PF-LARGE	SUB	+80	e-80 hex=e', exponentul real al lui i.
	CP	+1C	Este e' mai mic decât 28 zecimal?
	JR	C, 2E6F, PF-MEDIUM	Salt dacă este mai mic.
	CALL	2DC1, LOG(2^A)	n este format în A.
	SUB	+07	Si redus la n-7.
	LD	B, A	Apoi este copiat în B.
	LD	HL, +5CAC	n-7 este adăugat la al doilea octet al lui mem-5, numărul de cifre cerut înainte de punctul zecimal al lui x.
	ADD	A, (HL)	Apoi i este înmulțit cu $10^{-(n+7)}$.
	LD	(HL), A	Aceasta îl va da într-un interval de mijloc pentru tipărire.
	LD	A, B	Se ciclează din nou bucla pentru a prelucra acum numărul mărime-medie.
	NEG		
	CALL	2D4F, E-TO-FP	
	JR	2E01, PF-LOOP	

v. Partea întreagă a lui x este acum stocată în bufferul de tipărire în mem-3

si mem-4.

2E6F PF-MEDIUM	EX	DE,HL	Acum DE îl indică pe i, HL pe f.
	CALL	2FBA, FETCH-TWO	Mantisa lui i este acum în D', E', D, E.
	EXX		Se schimbă registrii.
	SET	7, D	Bitul 7 numeric adevărat în D'.
	LD	A, L	octetul exponent e al lui i în A.
	EXX		Se aduc înapoi registrii principali.
	SUB	+80	Exponentul real e' = e-80 hex în A.
	LD	B, A	Acesta dă contorul de bit care s-a cerut.

De notat că în cazul în care i este un număr întreg mic (mai mic decât 65536) se reintră aici.

2E7B PF-BITS	SLA	E	Acum mantisa lui i se rotește la stînga și toți bitii lui i sînt aceia deplasati în mem-4 și fiecare octet al lui mem-4 este ajustat zecimal la fiecare deplasare.
	RL	D	Toti cei patru octeti ai lui i.
	EXX		Întoarcere la registrii principali.
	LD	HL, +5CAA	Se pune în HL adresa celui de al cincilea octet al lui mem-4; contor pentru 5 octeti în C.
	LD	C, +05	Se aduce octetul din mem-4. Se deplasează la stînga, luînd noul bit.
2E8A PF-BYTES	LD	A, (HL)	Ajustarea zecimală a octetului.
	ADC	A, A	Este readus în mem-4.
	DAA		Se indică următorul octet al lui mem-4.
	LD	(HL), A	Se decrementează contorul de biti cu unu.
	DEC	HL	Salt pentru fiecare octet al lui mem-4.
	DEC	C	Salt pentru fiecare bit al lui INT (x).
	JR	NZ, 2E8A, PF-BYTES	
	DJNZ	, PF-BITES	

Ajustînd zecimal fiecare octet al lui mem-4 se obțin două cifre zecimale per octet, fiind cel mult nouă cifre. Cifrele vor fi acum reîmpachetate, una într-un octet, în mem-3 și mem-4, folosind instrucția RLD.

	XOR	A	A este sters pentru primirea cifrelor.
	LD	HL, +5CA6	Adresa sursă: primul octet al lui mem-4.
	LD	DE, +5CA1	Destinatia: primul octet al lui mem-3.
	LD	B, +09	Sînt cel mult 9 cifre.
	RLD		Bucata din stînga al lui mem-4 este înlăturată.
	LD	C, +FF	FF în C va semnala un zero de fond, iar 00 va semnala un nonzero de fond.
2EA1 PF-DIGITS	RLD		Partea stîngă a lui (HL) în A iar partea dreaptă a lui (HL) la stînga.
	JR	NZ, 2EA9, PF-INSERT	Salt dacă cifra din A nu este zero.
	DEC	C	Testare zero de fond;
	INC	C	aceasta va da acum resetarea lui zero.
	JR	NZ, 2E83, PF-TEST-2	Salt dacă a fost un zero de de fond.
2EA9 PF-INSERT	LD	(DE), A	Acum se introduce cifra.
	INC	DE	Se indică următoarea indicație.

	INC	(mem-5-1st)	Încă o cifră de tipărit, și
	INC	(mem-5-2nd)	încă una după punctul zecimal
	LD	C,+00	Se schimbă fanioul din zero de fond în alt zero.
2EB3 PF-TEST-2	BIT	0,B	Indicatorul sursă trebuie incrementat la fiecare a doua trecere prin buclă, când B este impar.
	JR	Z,2EB8,PF-ALL-9	Salt înapoi pentru toate cele 9 cifre.
	INC	HL	Se aduce contorul: există acolo cele 9 cifre care exclud zerourile de fond?
2EB8 PF-ALL-9	DJNZ	2EA1,PF-DIGITS	Pregătire ciclare: se reduce contorul la 8.
	LD	A,(mem-5-1st)	Se compară a 9-a cifră din octetul 4 al lui mem-4, cu -4 pentru a seta transportul pentru rotunjire.
	SUB	+09	Salt mai departe pentru rotunjire.
	JR	C,2EC8,PF-MORE	Se folosește din nou calculatorul.
	DEC	(mem-5-1st)	(și este acum sters.)
	LD	A,+04	f
	CP	(mem-4-4th)	f
	JR	2FOC,PF-ROUND	
2EC8 PF-MORE	RST	0028,FP-CALC	
	DEFB	+02,delete	
	DEFB	+E2,get-mem-2	
	DEFB	+38,end-calc	
vi. Partea fracționară a lui x este acum stocată în bufferul de tipărire.			
2ECF PF-FRACTN	EX	DE,HL	Acum DE îl indică pe f.
	CALL	2FDA,FETCH-TWO	Mantisa lui f este acum în D',E',D,E.
	EXX		Schimbare registrii.
	LD	A,+80	Exponentul lui f este redus la zero, rotind bitii lui f cu 80 hex. - e locuri la dreapta, unde L' conține e.
	SUB	L	Bitul numeric real în bitul 7 din D'.
	LD	L,+00	Se readuc registrii principali.
	SET	7,D	Acum se face deplasarea.
	EXX		Se aduce contorul cifrelor.
2EDF PF-FRN-LP	CALL	2FDD,SHIFT-FP	Sînt deja 8 cifre?
	LD	A,(mem-5-1st)	Dacă nu, salt înainte.
	CP	+08	Dacă sînt 8 cifre, se folosește f pentru a-l rotunji pe i, rotind D' la stînga pentru a seta transportul.
	JR	C,2EEC,PR-FR-DGT	Se readuc registrii principali și salt în față pentru rotunjire.
	EXX		Initial zero în C, contorul de 2 în B.
	RL	D	D'E'DE este înmulțit cu 10 în 2 etape, mai întîi DE apoi D'E', fiecare octet cu octet în 2 etape, și partea întregă a rezultatului este obținută în C pentru a fi trecută în bufferul de tipărire.
2EEC PF-FR-DGT	JR	2FOC,PF-ROUND	Contorul și rezultatul alternează între BC și B'C'.
	LD	BC,+0200	
2EEF PF-FR-EXX	LD	A,E	Privire înapoi de-a lungul schimbării registrilor, începutul - primul octet al lui mem-3.
	CALL	2F8B,CA=10*A+C	Rezultatul este pus în A pentru stocare.
	LD	E,A	Contorul cifrelor de pînă acum din număr în C.
	LD	A,D	Este adresat primul octet gol
	CALL	2F8B,CA=10*A+C	Se stochează următoarea cifră.
	LD	D,A	Se mărește contorul cifrelor.
	PUSH	BC	Se cicleză pînă sînt opt
	EXX		
	POP	BC	
	DJNZ	2EEF,PF-FR-EXX	
	LD	HL,+CA1	
	LD	A,C	
	LD	C,(mem-5-1st)	
	ADD	HL,BC	
	LD	(HL),A	
	INC	(mem-5-1st)	
	JR	2EDF,PF-FRN-LP	

cifre.

vii. Cifrele stocate în bufferul de tipărire sînt rotunjite la maxim 8 cifre pentru tipărire.

2F0C PF-ROUND	PUSH AF	Se salvează fanionul de transport pentru rotunjire.
	LD HL,+5CA1	Adresa de bază a numărului: mem-3, octetul 1.
	LD C,(mem-5-1st)	Deplasamentul (numărul de cifre din număr) în BC.
	LD B,+00	Se adresează ultimul octet al numărului.
	ADD HL,BC	Se copiază C în B ca si contor.
	LD B,C	Se reduce fanionul de transport.
	POP AF	Acesta este ultimul octet al numărului.
2F18 PF-RND-LP	DEC HL	Se aduce octetul în A.
	LD A,(HL)	Adunare în transport, adică rotunjire.
	ADC A,+00	Se stochează octetul rotunjit în buffer.
	LD (HL),A	Dacă octetul este 0 sau 10, B va fi decrementat si zero din fina (sau 10) nu vor fi numărați pentru tipărire.
	AND A	Se resetează transportul pentru o cifră validă.
	JR Z,2F25,PF-R-BACK	Salt dacă transportul este resetat.
	CP +0A	Salt înapoi pentru mai multe rotunjiri sau mai multe zerouri în final.
	CCF	Este o depășire la sfîngaj este necesar aici un 1 additional.
	JR NC,2F2D,PF-COUNT	Este de asemenea o cifră additională înainte de punctul zecimal.
2F25 PF-R-BACK	DJNZ 2F18,PF-RND-LP	Acum B setează contorul cifrelor ce vor fi tipărite (zerourile finale nu vor fi tipărite).
	LD (HL),+01	f trebuie sters.
	INC B	-
	INC (mem-5-2nd)	-
2F2D PF-COUNT	LD (mem-5-1st),B	Deplasamentul calculatorului care a fost salvat în stivă este readus în H'L.
	RST 0028,FP-CALC	
	DEFB +02,delete	
	DEFB +38,end-calc	
	EXX	
	POP HL	
	EXX	

viii. Acum se poate tipări numărul. Mai întîi C va fi setat astfel încît să contină numărul cifrelor ce trebuie tipărite, necontorizînd zerourile finale, în timp ce B va contine numărul cifrelor cerute înainte de punctul zecimal.

LD BC,(mem-5-1st)	Contoarele sînt setate.
LD HL,+5CA1	Inceputul cifrelor.
LD A,B	Dacă cifrele cerute înainte virgulă sînt mai mult de 9,
CP +09	sau mai puțin de 4, atunci este necesar formatul E.
JR C,2F46,PF-NOT-E	Mai puțin de 4 înseamnă mai mult de 4 zerouri de fond după virgulă.
CP +FC	Nu există cifre înainte de virgulă? Dacă da, se tipăreste un zero initial.
JR C,2F6C,PF-E-FRMT	
AND A	
CALL Z,15EF,OUT-CODE	

Următorul punct de intrare este de asemenea folosit pentru a tipări cifrele necesare tipărite în formatul E.

2F4A PF-E-SBRN	XOR A	Se începe prin setarea lui A pe zero.
	SUB B	Se scade B: minus va însemna că există cifre înainte de virgulă; salt înainte pentru a le tipări.
	JR M,2F52,PF-OUT-LP	

	LD	B,A	Acum se cere ca A să fie contor.
	JR	2F5E,PF-DC-OUT	Salt înainte pentru a tipări partea zecimală.
2F52 PF-OUT-LP	LD	A,C	Se copiază în A numărul cifrelor ce trebuie tipărite.
	AND	A	Dacă A este zero, adică mai sînt zerouri finale de tipărit (B este diferit de zero), atunci se execută salt.
	JR	Z,2F59,PF-OUT-DT	Se aduce o cifră din bufferul de tipărire.
	LD	A,(HL)	Se indică următoarea cifră.
	INC	HL	Se decrementează contorul cu unu.
	DEC	C	Se tipărește cifra corespunzătoare.
2F59 PF-OUT-DT	CALL	15EF,OUT-CODE	Se ciclează înapoi pînă cînd B este zero.
	DJNZ	2F52,PF-OUT-LP	Este momentul să se tipărească virgula, numai dacă C este acum zero; în acest caz, se revine - sfîrsit.
2F5E PF-DC-OUT	LD	A,C	Se adună 1 la B - se include punctul zecimal.
	AND	A	Se pune codul pentru '.' în A.
	RET	Z	Se tipărește '.'.
	INC	B	Se introduce codul caracter pentru '0'.
	LD	A,+2E	Se ciclează înapoi pentru a tipări toate zerourile necesare.
2F64 PF-DEC-OS	RST	0010,PRINT-A-1	Se setează contorul pentru toate cifrele rămase.
	LD	A,+30	Salt înapoi pentru a le tipări.
	DJNZ	2F64,PF-DEC-OS	Contorul cifrelor este copiat în D.
	LD	B,C	EI este decrementat pentru a da exponentul.
	JR	2F52,PF-OUT-LP	In format E se cere o cifră înainte de punctul zecimal.
2F6C PF-E-FRMT	LD	D,B	Acum se tipărește toată partea numărului dinainte de E.
	DEC	D	Se introduce codul caracter pentru 'E'.
	LD	B,+01	Se tipărește 'E'.
	CALL	2F4A,PF-E-SBRN	Acum exponentul este în C pentru a fi tipărit.
	LD	A,+45	Si în A pentru a fi testat.
	RST	0010,PRINT-A-1	I se testează semnul.
	LD	C,D	Salt dacă este pozitiv.
	LD	A,C	Altfel, este negat în A.
	AND	A	Apoi este copiat înapoi în C pentru a fi tipărit.
	JP	P,2F83,FP-E-POS	Se introduce codul caracter pentru '-'.
	NEG	C,A	Salt pentru a tipări semnul.
	LD	C,A	Se introduce codul caracter pentru '+'.
2F83 PF-E-POS	JR	2F85,PF-E-SIGN	Acum se tipărește semnul '+' sau '-'.
	LD	A,+2E	BC conține exponentul pentru a fi tipărit.
2F85 PF-E-SIGN	RST	0010,PRINT-A-1	Salt înapoi pentru a- tipări si sfîsit.
	LD	B,+00	
	JP	1A1B,OUT-NUM	

THE 'CA=10*A+C' SUBROUTINE (SUBRUTINA CA=10*A+C')

Această subrutină este apelată de subrutina PRINT-FP pentru a multiplica fiecare octet din D'E'DE cu 10 și returnează partea întregă a rezultatului în registrul C. La intrare, registrul A conține octetul ce trebuie multiplicat cu 10 iar registrul C conține încă transportul octetului anterior. La revenire, registrul A conține octetul rezultat iar registrul C conține transportul mai departe al octetului următor.

2F8B CA=10*A+C	PUSH DE	Salvarea oricărei perechi DE în utilizare.
	LD L,A	Se copiază înmultitorul din A în HL.
	LD H,+00	Se copiază și în DE.
	LD E,L	
	LD D,H	
	ADD HL,HL	Se dublează HL.
	ADD HL,HL	Se dublează încă o dată.
	ADD HL,DE	Adunare DE pentru a obține HL=5*A.
	ADD HL,HL	Se dublează din nou: acum HL=10*A.
	LD E,C	Se copiază C în DE (D este zero) pentru adunare.
	ADD HL,DE	Acum HL=10*A+C.
	LD C,H	H este copiat în C.
	LD A,L	L este copiat în A, completând taskul.
	POP DE	Se readuce registrul pereche DE.
	RET	Sfîrsit.

THE 'PREPARE TO ADD' SUBROUTINE (SUBROUTINA 'PREGATIRE PENTRU ADUNARE')

Această subrutină este prima din cele patru subrutine care sînt folosite de principalele rutine de operații aritmetice - SUBTRACTION (scădere), ADDITION (adunare), MULTIPLICATION (înmulțire) și DIVISION (împărțire).

Această subrutină particulară pregătește un număr în virgulă mobilă pentru adunare, mai ales prin înlocuirea bitului de semn cu un bit numeric adevărat 1, și negarea numărului (complementul față de doi) dacă este negativ. Exponentul este returnat în registrul A și primul octet este setat pe 00 hex. pentru un număr pozitiv și pe FF hex. pentru un număr negativ.

2F9B PREP-ADD	LD A,(HL)	Se transferă exponentul în A.
	LD (HL),+00	Se consideră un număr pozitiv
	AND A	Dacă numărul este zero,
	RET Z	atunci pregătirea s-a și terminat.
	INC HL	Acum se indică octetul de semn.
	BIT 7,(HL)	Se setează fanionul zero dacă numărul este pozitiv.
	SET 7,(HL)	Se reface bitul numeric adevărat.
	DEC HL	Se indică din nou primul octet.
	RET Z	Numerele pozitive au fost pregătite, dar numerele negative trebuie complementate față de doi.
	PUSH BC	Salvarea oricărui exponent anterior.
	LD BC,+0005	Sînt 5 octeți care trebuie tratați.
	ADD HL,BC	Se indică unul după ultimul octet.
	LD B,C	Se transferă '5' în B.
	LD C,A	Se salvează exponentul în C.
	SCF	Se setează fanionul zero pentru negare.
2FAF NEG-BYTE	DEC HL	Se indică rînd pe rînd fiecare octet.
	LD A,(HL)	Se aduce fiecare octet.
	CPL	Se face complementul față de unu al octetului.
	ADC A,+00	Se adună transportul pentru negare.
	LD (HL),A	Se reface octetul.
	DJNZ 2FAF,NEG-BYTE	Se ciclează de '5' ori.
	LD A,C	Se reface exponentul în A.
	POP BC	Se reface orice exponent anterior.
	RET	Sfîrsit.

THE 'FETCH TWO NUMBERS' SUBROUTINE (SUBROUTINA 'ADUCERE DOUA NUMERE')

Această subrutină este apelată de ADDITION, MULTIPLICATION și DIVISION pentru a aduce două numere din stiva calculatorului și pentru a le pune în registrul inclusiv în registrul de schimb.

La intrarea în subrutină registrul pereche HL indică primul octet al primului număr iar registrul pereche DE indică primul octet al celui de al

doilea număr.

Cînd subrutina este apelată din MULTIPLICATION sau DIVISION semnul rezultatului este salvat în al doilea octet al primului număr.

2FBA FETCH-TWO PUSH HL Salvare HL.
 PUSH HL Salvare AF.

Se apelează cei cinci octeti ai primului număr - M1, M2, M3, M4 & M5
 și ai primului număr - N1, N2, N3, N4 & N5

LD	C, (HL)	M1 în C.
INC	HL	Următorul
LD	B, (HL)	M2 în B.
LD	(HL), A	Se copiază semnul rezultatului în (HL).
INC	HL	Următorul.
LD	A, C	M1 în A.
LD	C, (HL)	M3 în C.
PUSH	BC	Se salvează M2 & M3 în stiva mașinii.
INC	HL	Următorul.
LD	C, (HL)	M4 în C.
INC	HL	Următorul.
LD	B, (HL)	M5 în B.
EX	DE, HL	Acum HL indică M1.
LD	D, A	M1 în D.
LD	E, (HL)	M1 în E.
PUSH	DE	Se salvează M1 & M1 în stiva mașinii.
INC	HL	Următorul.
LD	D, (HL)	N2 în D.
INC	HL	Următorul.
LD	E, (HL)	N3 în E.
PUSH	DE	Se salvează N2 & N3 în stiva mașinii.
EXX		Se aduc registrii de schimb.
POP	DE	N2 în D' & N3 în E'.
POP	HL	M1 în H' & M1 în L'.
POP	BC	M2 în B' & M3 în C'.
EXX		Se aduce setul original de registrii.
INC	HL	Următorul.
LD	D, (HL)	N4 în D.
INC	HL	Următorul.
LD	E, (HL)	N5 în E.
POP	AF	Se restaurează AF original.
POP	HL	Se restaurează HL original.
RET		Sfîrsit.

Sumar: M1 - M5 sînt în H', B', C', C, B.
 N1 - N5 sînt în L', D', E', D, E.
 HL indică primul octet al primului număr.

THE 'SHIFT ADDEND' SUBROUTINE (SUBROUTINA 'DEPLASARE TERMENI ADUNARE')

Această subrutină deplasează un număr în virgulă mobilă pînă la 32 zecimale, 20 hex, plasat la dreapta liniei așa cum trebuie pentru adunare. Numărul cu exponentul mai mic a fost pus în poziția termenului adunării înaintea apelării subrutinei. Fiecare depășire la dreapta, în transport, este adunată înapoi la număr. Dacă diferența exponentilor este mai mare de 32 zecimal sau transportul revine la începutul numărului atunci numărul este setat pe zero astfel încît adunarea nu va modifica alte numere (termeni ai adunării).

2FDD SHIFT-FP	AND	A	Dacă diferența exponentilor este zero, subrutina revine imediat. Dacă diferența este mai mare decît 20 hex, salt înainte.
	RET	Z	
	CP	+21	
	JR	NC, 2FF9, ADDEND-0	
	PUSH	BC	Se salvează BC pentru scurt timp.
	LD	B, A	Se transferă diferența exponent în B pentru a număra deplasările la dreapta.
2FEE ONE-SHIFT	EXX		Deplasare aritmetică la dreapta pentru L', păstrînd bitul marcător de semn.
	SRA	L	
	RR	D	Rotire la dreapta cu transport D', E', D&E.
	RR	E	
	EXX		Astfel se deplasează totii cei cinci octeti ai numărului la
	RR	D	

	RR	D	dreapta de atîtea ori de cîte ori numără B.	
	DJNZ	2FE5, ONE-SHIFT	Se ciclează pînă cînd B ajunge la zero.	
	POP	BC	Se reface BC original.	
	RET	NC	Revenire dacă nu se mai găsește nici un transport.	
	CALL	3004, ADD-BACK	Regăsire transport.	
	RET	NZ	Se revine numai dacă transportul revine înapoi (în acest caz nu este nimic de adunat).	
2FF9	ADDEND-0	EXX	Se aduce L', D' & E'.	
		XOR	A	Se șterge registrul A.
2FFB	ZEROS-4/5	LD	L, +00	Se setează termenul adunării la zero în D', E', D&E, împreună cu octetul marcator (indicator semn) L', care a fost 00 hex pentru un număr pozitiv și FF hex pentru un număr negativ.
		LD	D, A	
		LD	E, L	
		EXX		
		LD	DE, +0000	ZEROS-4/5 produce numai 4 octeți zero cînd apelează în apropiere la 3160.
	RET			Sfîrșit.

THE 'ADD-BACK' SUBROUTINE (SUBRUTINA 'ADD-BACK')

Această subrutină adună înapoi în număr orice transport care a depășit la dreapta. În caz extrem, transportul revine înapoi la stînga numărului.

Cînd această subrutină este apelată în timpul adunării, această fluctuație înseamnă că o mantisă a lui 0,5 a fost deplasată cu totul 32 locuri la dreapta, și termenul adunării va fi acum setat pe zero; cînd este apelată din MULTIPLICATION, înseamnă că exponentul trebuie incrementat, iar aceasta poate rezulta din depășire.

3004	ADD-BACK	INC	E	Se adună transportul la octetul cel mai din dreapta.
		RET	NZ	Se revine dacă nu este depășire la stînga.
		INC	D	Se continuă cu următorul octet.
		RET	NZ	Revenire dacă nu este depășire la stînga.
		EXX		Se aduce următorul octet.
		INC	E	Și acesta se incrementează.
		JR	NZ, 300D, ALL-ADDED	Salt dacă nu este depășire.
		INC	D	Se incrementează ultimul octet.
300D	ALL-ADDED	EXX		Se readuc registrii originari
		RET		Sfîrșit.

THE 'SUBTRACTION' OPERATION (OPERATIA 'SUBTRACTION' (scădere)) (Deplasament 03 - vezi CALCULATE după: 'subtract')

Această subrutină doar schimbă semnul scăzătorului și îl transportă în ADDITION.

De notat că HL indică descăzutul și DE indică scăzătorul. (Vezi ADDITION pentru mai multe detalii).

300F	SUBTRACT	EX	DE, HL	Se interschimbă indicatorii.
		CALL	346E, NEGATE	Se schimbă semnul scăzătorului.
		EX	DE, HL	Se interschimbă indicatorii înapoi și se continuă în ADDITION.

THE 'ADDITION' OPERATION (OPERATIA 'ADDITION' (adunare)) (Deplasament 0F - vezi CALCULATE după: 'addition')

Prima din cele mai importante trei subrutine aritmetice, această subrutină execută adunarea în virgulă mobilă a două numere, fiecare cu 4 octeți pentru mantisă și 1 octet pentru exponent. În aceste trei subrutine, cele două numere din vârful stivei calculatorului sînt adunate/înmultite/împartite pentru a obține un număr în vârful stivei calculatorului, o 'ultima valoare'. HL indică al doilea număr din vîrf, primul termen al adunării/deînmultitului/deîmpartitului. DE indică numărul din vârful stivei calculatorului, al doilea termen al adunării/înmultitorului/împartitorului. Apoi HL indică rezultatul 'ultima valoare' a cărei adresa se poate considera ca fiind în STKEND-5.

Dar subrutina de adunare testează mai întîi dacă cele două numere ce

trebuie adunate sînt 'numere întregi mici'. Dacă sînt, le adună simplu în HL și BC, și pune rezultatul direct în stivă. Nu este necesară complementarea față de doi înainte sau după adunare, deoarece asemenea numere sînt conținute în stivă în forma complementată față de doi, gata pentru adunare.

3014 addition	LD	A, (DE)	Se testează dacă primii
	OR	(HL)	octeți ai ambelor numere
	JR	NZ, 303E, FULL-ADDN	sînt zero.
	PUSH	DE	Dacă nu, salt pentru adunare
	INC	HL	deplină.
	PUSH	HL	Se adună indicatorul la al
	INC	HL	doilea număr.
	LD	E, (HL)	Se indică al doilea octet al
	INC	HL	primului număr și se salvează
	LD	D, (HL)	indicatorul.
	INC	HL	Se indică cel mai puțin
	INC	HL	semnificativ octet.
	LD	A, (HL)	acesta este adus în E.
	INC	HL	Se indică octetul cel mai
	LD	C, (HL)	semnificativ.
	INC	HL	Acesta este adus în D.
	LD	B, (HL)	Se trece la al doilea octet
	POP	HL	al celui de al doilea număr.
	EX	DE, HL	Acesta este adus în A (acesta
	ADD	HL, BC	este octetul semn).
	EX	DE, HL	Se indică octetul cel mai
	ADD	A, (HL)	puțin semnificativ.
	RRCA		Acesta este adus în C.
	ADC	A, +00	Se indică octetul cel mai
	JR	NZ, 303C, ADDN-OFLW	semnificativ.
	SBC	A, A	Aceasta este adus în B.
3032	LD	(HL), A	Se aduce indicatorul
	INC	HL	octetului de semn al primului
	LD	(HL), E	număr; este pus în DE, iar
	INC	HL	numărul în HL.
	LD	(HL), D	Se execută adunarea:
	DEC	HL	rezultatul în HL.
	DEC	HL	Rezultatul în DE, octetul de
	DEC	HL	semn în HL.
	POP	DE	Se adună octetul de semn și
	RET		transportul în A; aceasta va
			detecta dacă este depășire.
			Un A diferit de zero indică
			depășire.
			Salt pentru a reseta
			indicatorii și pentru adunare
			deplină.
			Definire octet de semn corect
			pentru rezultat.
			Se stochează în stivă.
			se indică următoarea locație.
			Se stochează octetul cel mai
			puțin semnificativ al
			rezultatului.
			Se indică următoarea locație.
			Se stochează octetul cel mai
			semnificativ al rezultatului.
			Se mută indicatorul înapoi la
			adresa primului octet al
			rezultatului.
			Se readuce STKEND în DE.
			Sfîrșit.

De notat că numărul -65536 zecimal poate apărea aici în forma 00 FF 00 00 00 ca rezultatul adunării a două numere întregi mici negative, de exemplu -65000 și -536. Este mai simplu de stocat în această formă. Aceasta este o greșală. Sistemul Spectrum nu poate trata acest număr. Mai multe funcții îl tratează ca un zero, și este tipărit ca -1E38 obținut prin tratarea ca 'minus zero' într-un format ilegal.

Un remediu posibil ar fi testarea acestui număr în jurul octetului 3032 și, dacă este prezent, să se facă al doilea octet 80 hex și primul octet 91 hex, astfel umplînd cei cinci octeți ai numărului în virgulă mobilă, adică 91 80 00 00 00, care ridică nici o problemă. Vezi și precizările din 'truncate' de după, înaintea octetului 3225, și Anexa.

303C ADDN-OFLW	DEC	HL	Se readuce indicatorul la
	POP	DE	primul număr.
			Se readuce indicatorul al al

303E FULL-ADDN CALL 3293,RE-ST-TWO

doilea număr.
Se restocchează ambele numere
umplînd cei cinci octeți ai
formeî în virgulă mobilă.

Subrutina ADDITION în totalitatea ei apelează mai întîi PREP-ADD pentru fiecare număr, apoi aduce cele două numere din stiva calculatorului și-l pune pe cel cu exponentul mai mic în poziția de prim termen al adunării. Apoi apelează SHIFT-FP pentru a deplasa acest termen al adunării cu 32 zecimal locuri la dreapta pentru a-l alinia pentru adunare. Adunarea actuală se face în cîțiva octeți, se face o singură deplasare pentru transport (depășire la stînga) dacă este necesar, rezultatul este complementat față de doi dacă este negativ, și orice depășire aritmetică este raportată; altfel subrutina sare la TEST-NORM pentru a normaliza rezultatul și îl pune în stivă cu bitul de semn corect introdus în al doilea octet.

EXX		Se interschimbă registrii.
PUSH	HL	Salvarea următoarei adrese corecte.
EXX		Se interschimbă registrii.
PUSH	DE	Se salvează indicatorul primului termen al adunării.
PUSH	HL	Se salvează indicatorul celui de al doilea termen al adunării.
CALL	2F9B,PREP-ADD	Se pregătește al doilea termen al adunării.
LD	B,A	Se salvează exponentul lui în B.
EX	DE,HL	Se interschimbă indicatorii.
CALL	2F9X,PREP-ADD	Se pregătește primul termen al adunării.
LD	C,A	Se salvează exponentul lui.
CP	B	Dacă primul exponent este mai mic, se păstrează primul număr în poziția primului termen al adunării; altfel se schimbă din nou exponentii și indicatorii.
JR	NC,3055,SHIFT-LEN	Se salvează exponentul mai mare în A.
LD	A,B	Diferența dintre exponenți este lungimea deplasării la dreapta.
LD	B,C	Se aduc cele două numere din stivă.
EX	DE,HL	Se deplasează la dreapta primul termen al adunării.
3055 SHIFT-LEN	PUSH AF	Se readuce exponentul mai mare.
	SUB B	HL va indica rezultatul.
	CALL 2FBA,FETCH-TWO	Se stocchează exponentul rezultatului.
	CALL 2FDD,SHIFT-FP	Se salvează indicatorul din nou.
	POP AF	M4 în H & M5 în L, (vezi FETCH-TWO).
	POP HL	Se adună cei doi octeți din dreapta.
	LD (HL),A	M2 în H' & N3 în L', (vezi FETCH-TWO).
	PUSH HL	Se adună octeții din stînga cu transportul.
	LD L,B	Rezultatul înapoi în D'E'.
	LD H,C	Se adună H',L' și transportul mecanismul rezultat asigurînd că se va apela o singură deplasare la dreapta dacă suma a 2 numere pozitive a dat depășire la stînga, sau suma a două numere negative nu a dat depășire la stînga.
	ADD HL,DE	Acum rezultatul este în DED'E'.
	EXX	Se aduce indicatorul la exponent.
	EX DE,HL	Test pentru deplasare (H',L' au fost 00 hex pentru numere pozitive și FF hex pentru
	ADC HL,BC	
	EX DE,HL	
	LD A,H	
	ADC A,L	
	LD L,A	
	RRA	
	XOR L	
	EXX	
	EX DE,HL	
	POP HL	
	RRA	
	JR NC,307C,TEST-NEG	

	LD	A, +01	numere negative). A numără o singură deplasare la dreapta.
	CALL	2FDD, SH1DT-FP	Apelare deplasare.
	INC	(HL)	Se adună 1 la exponent;
	JR	Z, 309F, AD-REP-6	aceasta poate conduce la o depășire aritmetică.
307C TEST-NEG	EXX		Test pentru rezultat negativ;
	LD	A, L	se aduce bitul semn al lui L
	AND	+80	în A (acesta indică acum corect semnul rezultatului).
	EXX		Aceasta se va stoca în al doilea octet pozitiv a rezultatului în stiva calculatorului.
	INC	HL	
	LD	(HL), A	
	DEC	HL	
	JR	Z, 3DA5, GO-NC-MLT	Dacă este zero, atunci nu se complementează față de doi rezultatul.
	LD	A, E	Se aduce primul octet.
	NEG		Se face negarea lui.
	CCF		Se complementează transportul pentru a continua negarea și se stochează octetul.
	LD	E, A	
	LD	A, D	Se aduce următorul octet.
	CPL		Se complementează față de unu
	ADC	A, +00	Adunare în transport pentru negare.
	LD	D, A	Se stochează octetul.
	EXX		Se trece la aducerea următorului octet în registrul A.
	LD	A, E	
	CPL		Se complementează față de unu
	ADC	A, +00	Adunare în transport pentru negare.
	LD	E, A	Se stochează octetul.
	LD	A, D	Se aduce ultimul octet.
	CPL		Se complementează față de unu
	ADC	A, +00	Se adună în transport pentru negare.
	JR	NC, 3DA3, END-COMPL	Se execută dacă nu este transport.
	RRA		Altfel, se duce .5 în mantisă și se adună 1 la exponent; acesta este necesar când se adună două numere negative pentru a da exact puterea lui 2, și poate conduce la o depășire aritmetică.
	EXX		Se prezintă eroare dacă se cere.
	INC	(HL)	
309F ADD-REP-6	JP	Z, 31AD, REPORT-6	
	EXX		
	LD	D, A	Se stochează ultimul octet.
30A3 END-COMPL	EXX		
30A5 GO-NC-MLT	XOR	A	Se șterge fanionul de transport.
	JP	3155, TEST-NORM	Iesire prin TEST-NORM.

THE 'HL=HL*DE' SUBROUTINE (SUBRUTINA 'HL=HL*DE')

Această subrutină este apelată de 'SET-HL*DE' și de 'MULTIPLICATION' pentru a efectua înmulțirea pe 16 biti ca atare.

Orice depășire a celor 16 biti admisi este tratată cu o revenire din subrutină.

30A9 HL=HL*DE	PUSH	BC	Este salvat BC.
	LD	B, +10	Este o înmulțire pe 16 biti.
	LD	A, H	A conține octetul cel mai semnificativ.
	LD	C, L	C conține octetul cel mai puțin semnificativ.
	LD	HL, +0000	Se initializează rezultatul pe zero.
30B1 HL-LOOP	ADD	HL, HL	Se dublează rezultatul.
	JR	C, 30BE, HL-END	Salt dacă este depășire.
	RL	C	Se rotește bitul 7 din C în transport.
	RLA		Se rotește bitul de transport în bitul 0 și bitul 7 în fanionul de transport.
	JR	NC, 30BC, HL-AGAIN	Salt dacă fanionul de

	ADD	HL, DE	transport este resetat.
	JR	C, 30BE, HL-END	Altfel se adună DE imediat.
30BC HL-AGAIN	DJNZ	30B1, HL-LOOP	Salt dacă este depășire.
30BE HL-END	POP	BC	Se ciclează de 16 ori.
	RET		Se readuce BC.
			Sfârșit.

THE 'PREPARE TO MULTIPLY OR DIVIDE' SUBROUTINE (SUBROUTINA 'PREGATIRE PENTRU INMULTIRE SAU IMPARTIRE')

Această subrutină pregătește un număr în virgulă mobilă pentru înmulțire sau împărțire, revenind cu transportul setat dacă numărul este zero, aducând semnul rezultatului în registrul A, și înlocuind bitul semn în număr prin bitul numeric adevărat, 1.

30C0 PREP-M/D	CALL	34E9, TEST-ZERO	Dacă numărul este zero, se revine cu fanionul de transport resetat.
	RET	C	Se indică octetul de semn.
	INC	HL	Se aduce semnul rezultatului în A (semn dorit de plus, semn nedorit de minus); de asemenea se resetează fanionul de transport.
	XOR	(HL)	Se setează bitul numeric adevărat.
	SET	7, (HL)	Se indică din nou următorul exponent.
	DEC	HL	Se revine cu fanionul de transport resetat.
	RET		

THE 'MULTIPLICATION' OPERATION (OPERATIA 'MULTIPLICATION' (înmulțire))
(Deplasament 04 - vezi CALCULATE după: 'multiply')

Această subrutină testează mai întâi dacă cele două numere care trebuie înmulțite sînt 'întregi mici'. Dacă sînt, subrutina folosește INT-FETCH pentru a aduce numerele din stivă, apoi HL=HL*DE pentru a le înmulți și INT-STORE pentru a returna rezultatul în stivă, orice depășire a acestei 'înmulțiri scurte' (adică dacă rezultatul nu este el însuși un 'întreg mic') cauzează un salt la înmulțirea celor cinci octeți ai formei în virgulă mobilă (vezi după).

30CA multiply	LD	A, (DE)	Se testează care din primii octeți ai ambelor numere este zero.
	OR	(HL)	Dacă nu este zero, salt pentru înmulțire 'lungă'.
	JR	NZ, 30F0, MULT-LONG	Salvare indicatori: pentru al doilea număr.
	PUSH	DE	Și pentru primul număr.
	PUSH	HL	Și din nou pentru al doilea număr.
	PUSH	DE	Se aduce semnul în C, numărul în DE.
	CALL	2D7F, INT-FETCH	Acum numărul în HL.
	EX	DE, HL	Numărul în stivă, al doilea indicator în HL.
	EX	(SP), HL	Se salvează primul semn în B.
	LD	B, C	Se aduce al doilea semn în C, numărul în DE.
	CALL	2D7F, INT-FETCH	Se formează semnul rezultatului în A: semn dorit de plus (00, semn nedorit de minus (FF)).
	LD	A, B	Se stochează semnul rezultatului în C.
	XOR	C	Se readuce primul număr în HL.
	LD	C, A	Se efectuează înmulțirea actuală.
	POP	HL	Se stochează rezultatul în DE.
	CALL	30A9, HL=HL*DE	Se readuce indicatorul primului număr.
	EX	DE, HL	Salt în depășirea înmulțirii 'depline'.
	POP	HL	Acești 5 octeți asigură că 00 FF 00 00 00 este înlocuit cu zero; ei nu ar fi necesari dacă acest număr ar fi fost exclus din sistem (vezi după 303E deasupra).
	JR	C, 30EF, MULT-OFLW	Acum se stochează rezultatul
30E5	LD	A, D	
	OR	E	
	JR	NZ, 30EA, MULT-RSLT	
	LD	C, A	
30EA MULT-RSLT	CALL	2D8E, INT-STORE	

	POP	DE	În stivă.
	RET		Se readuce STKEND în DE.
30EF MULT-OFLM	POP	DE	Sfîrsit.
			Se readuce indicatorul celui de al doilea număr.
30F0 MULT-LONG	CALL	3293,RE-ST-TWO	Se restochează ambele numere cu toți cei cinci octeți ai formei în virgulă mobilă.
	XDR	A	A este setat cu 00 hex, astfel încît semnul primului număr trece în A.
	CALL	30C0,PREP-M/D	Se pregătește primul număr, se revine dacă este zero.
	RET	C	(Rezultatul este deja zero.)
	EXX		Se inter schimbă registrii.
	PUSH	HL	Salvarea următoarei adrese corecte.
	EXX		Se interschimbă registrii.
	PUSH	DE	Se salvează indicatorul înmulatorului.
	EX	DE,HL	Se interschimbă indicatorii.
	CALL	30C0,PREP-M/D	Se pregătește al doilea număr.
	EX	DE,HL	Se interschimbă indicatorii din nou.
	JR	C,315D,ZERO-RSLT	Salt înainte dacă al doilea număr este zero.
	PUSH	HL	Se salvează indicatorul rezultatului.
	CALL	2F8A,FETCH-TWO	Se aduc cele două numere din stivă.
	LD	A,B	M5 în A (vezi (FETCH-TWO).
	AND	A	Pregătire pentru scădere.
	SBC	HL,HL	Se inițializează HL cu zero pentru rezultat.
	EXX		Se interschimbă registrii.
	PUSH	HL	Se salvează M1 & N1 (vezi FETCH-TWO).
	SBC	HL,HL	De asemenea se inițializează H'L' pentru rezultat.
	EXX		Se interschimbă registrii.
	LD	B,+21	B numără 33 zecimal, 21 hex, rotiri.
	JR	3125,STRT-MLT	Salt înainte în buclă.
- Acum se introduce bucla de înmulțire.			
3114 MLT-LOOP	JR	NC,311B,NO-ADD	Salt înainte la NO-ADD dacă nu este transport (adică dacă bitul de înmulțitului a fost resetat).
	ADD	HL,DE	Altfel, se adună înmulatorul în D'E'DE (vezi FETCH-ADD) în rezultatul care a fost construit în H'L'HL.
	EXX		
	ADC	HL,DE	
	EXX		
311B NO-ADD	EXX		Dacă înmulatorul a fost sau nu adunat, se deplasează rezultatul la dreapta în H'L'HL, adică deplasarea se face rotind fiecare octet cu transportul, astfel încît orice bit care trece în transport este scos de următorul octet, și deplasarea continuă în B'C'BA.
	RR	H	
	RR	L	
	EXX		
	RR	H	
	RR	L	
3125 STRT-MLT	EXX		Se deplasează la dreapta înmulatorul în B'C'CA (vezi FETCH-TWO & deasupra).
	RR	B	
	RR	C	
	EXX		
	RR	C	
	RRA		Un bit final coborît în transport va atrage o altă adunare a înmulatorului la rezultat.
	DJNZ	3114,MLT-LOOP	Se ciclează de 33 de ori pentru a aduce toți bitii.
	EX	DE,HL	Se trece rezultatul din:
	EXX		

	EXX	DE,HL	H'L'HL în D'E'DE.
Acum se adună exponentii împreună.			
	POP	BC	Se readuc exponentii - M1&N1.
	POP	HL	Se readuce indicatorul octetului exponent.
	LD	A,B	Se aduce suma celor doi octeti exponenti în A, și transportul corect.
	ADD	A,C	
	JR	NZ,313B,MAKE-EXPT	Dacă suma este egală cu zero, atunci se șterge transportul; altfel se lasă neschimbat.
	AND	A	Pregătire pentru incrementare a exponentului cu 80 hex.
313B MAKE-EXPT	DEC	A	
	CCF		
Restul subrutinei este comun și pentru MULTIPLICATION și pentru DIVISION.			
313D DIVN-EXPT	RLA		Acești câțiva octeți foarte iscusiti corectează octetul exponent.
	CCF		
	RRA		Rotind la stînga apoi la dreapta se aduce octetul exponent (adevăratul exponent plus 80 hex) în A.
	JP	P,3146,OFLW1-CLR	Dacă fanionul de semn este resetat, nu este necesară prezentarea unei depășiri aritmetice.
	JR	NC,31AD,REPORT-6	Se raportează depășire dacă transportul este resetat.
3146 OFLW1-CLR	AND	A	Acum se șterge transportul.
	INC	A	Octetul exponent este acum complet; Dar dacă A este zero, atunci este necesară o verificare a depășirii mai departe.
	JR	NZ,3151,OFLW2-CLR	Dacă transportul nu este setat și rezultatul este deja în forma normală (bitul 7 al lui D' setat) atunci trebuie raportată depășire; dar dacă bitul 7 al lui D' este resetat, rezultatul este chiar în interval, adică chiar sub 2**127.
	JR	C,3151,OFLW2-CLR	Se stochează octetul exponent
	EXX		Se trece al cincilea octet rezultat în A pentru secvența de normalizare, adică depășirea din L B'.
	BIT	7,D	
	EXX		
	JR	NZ,31AD,REPORT-6	
3151 OFLW2-CLR	LD	(HL),A	
	EXX		
	LD	A,B	
	EXX		
Restul subrutinei lucrează cu normalizarea și este comună pentru toate rutinele aritmetice.			
3155 TEST-NORM	JR	NC,316C,NORMALISE	Dacă nu este transport atunci se normalizează acum.
	LD	A,(HL)	Altfel, se lucrează cu sub nivel (rezultat zero) sau în jur de sub nivel (rezultat 2**(-128));
3159 NEAR-ZERO	AND	A	exponentul este returnat în A, se testează dacă A este zero (cazul 2**(-128) și dacă este așa se execută 2**(-128) dacă numărul este normal, altfel este produs zero.
	LD	A,+80	Apoi exponentul trebuie setat pe zero (pentru zero) sau 1 (pentru 2**(-128)).
315D ZERO-RBLT	JR	Z,315E,SKIP-ZERO	Se reface octetul exponent.
315E SKIP-ZERO	XOR	A	Salt dacă este cazul 2**(-128).
	EXX		Altfel, se pune zero în al doilea octet al rezultatului în stiva calculatorului.
	AND	D	Salt înainte pentru a transfera rezultatul.
	CALL	2FFB,ZEROS-4/5	
	RLCA		
	LD	(HL),A	
	JR	C,3195,OFLOW-CLR	
	INC	HL	
	LD	(HL),A	
	DEC	HL	
	JR	3195,OFLOW-CLR	

Normalizarea operatiei actuale.

316C NORMALISE	LD	R, +20	Se normalizează rezultatul făcînd 32 zecimal, 20 hex, deplasări la stînga în D'E'DE (cu A alăturat) pînă cînd bitul 7 din D' este setat. A conține zero după adunare așa că nu se cîștigă și nu se pierde nici o precizie; A conține al cincilea octet din B' după înmulțire sau împărțire, dar cum numai în jur de 32 de biti pot fi corecțati, nu se pierde din precizie; De notat că A este rotit circular, cu ramură în transport ... eventual un proces întâmplător.
316E SHIFT-ONE	EXX		Exponentul este decrementat la fiecare deplasare.
	RIT	7, D	Dacă exponentul devine zero, atunci numărul din 2** - 129 este rotunjit la 2** - 128.
	EXX		Se cicleză înapoi de 32 de ori.
	JR	NZ, 3186, NORML-NOW	Dacă bitul 7 nu devine 1 niciodată, atunci întregul rezultat trebuie să fie zero.
	RLCA		
	RL	E	
	RL	D	
	EXX		
	RL	E	
	RL	D	
	EXX		
	DEC	(HL)	
	JR	Z, 3159, NEAR-ZERO	
	DJNZ	316E, SHIFT-ONE	
	JR	315D, ZERO-RSLT	

Se termină normalizarea luînd în considerare 'transportul'.

3186 NORML-NOW	RLA		După normalizare se adună înapoi orice transport final care a trecut în A.
	JR	NC, 3195, OFLOW-CLR	Salt înainte dacă transportul nu oscilează direct înapoi.
	CALL	3004, ADD-BACK	Dacă el poate oscila direct înapoi atunci mantisa este setată pe 0.5 și exponentul este incrementat.
	JR	NZ, 3195, OFLOW-CLR	Această operație poate conduce la o depășire aritmetică (caz final).
	EXX		
	LD	D, +80	
	EXX		
	INC	(HL)	
	JR	Z, 31AD, REPORT-6	

Partea finală a subrutinei implică trecerea rezultatului în octetii rezervati pentru el în stiva calculatorului și resetarea indicatorilor.

3195 OFLOW-CLR	PUSH	HL	Se salvează indicatorul rezultatului.
	INC	HL	Se indică octetul semn al rezultatului.
	EXX		Rezultatul este mutat din registrii actuali, D'E'DE, în BCDE; și apoi în ACDE.
	PUSH	DE	
	EXX		
	POP	BC	
	LD	A, B	Bitul de semn este readus din stiva sa temporară și transferat în poziția lui corectă a bitului 7 al primului octet al mantisei.
	RLA		Este stocat primul octet.
	RL	(HL)	Următorul.
	RRA		Este stocat al doilea octet.
	LD	(HL), A	Următorul.
	INC	HL	Este stocat al treilea octet.
	LD	(HL), C	Următorul.
	INC	HL	Este stocat al patrulea octet.
	LD	(HL), D	Se readuce indicatorul rezultatului.
	INC	HL	Se readuce indicatorul celui de al doilea număr.
	LD	(HL), E	Se interschimbă registrii.
	POP	HL	Readucerea următoarei adrese corecte.
	EXX		Se interschimbă registrii.
	RET		Sfîrsit.

Prezentarea 6 - Depășire aritmetică

31AD REPORT-6 RST 0008,ERROR-1 Se apelează rutina de tratare
 DEFB +05 eroare.

THE 'DIVISION' OPERATION (OPERATIA 'DIVISION' (împărțire))
 (Deplasament 05 - vezi CALCULATE după: 'division')

Această subrutină pregătește mai întâi împărțitorul apelînd PREP-M/D, prezentînd depășire aritmetică dacă este zero; apoi ea pregătește deîmpărțitul apelînd din nou PREP-M/D, revenind dacă este zero. Apoi se aduc cele două numere din stiva calculatorului și se împart mantisele lor în înțelesul unei uzuale redări a împărțirii, încercînd scăderea împărțitorului din deîmpărțit și redarea dacă este transport, altfel se adaugă 1 la cîț. Precizia maximă se obține pentru împărțirea a 4 octeți, și după scăderea exponentilor subrutina iese alăturînd ultima parte de la MULTIPLICATON.

31AF division CALL 3293,RE-ST-TWO Se utilizează forma lungă în virgulă mobilă.
 EX DE,HL Se schimbă indicatorii.
 XOR A A este setat pe 00 hex, așa că semnul primului număr va merge în A.
 CALL 30C0,PREP-M/D Se pregătește împărțitorul și dacă este zero se raportează depășire aritmetică.
 JR C,31AD,REPORT-6 Se schimbă indicatorii.
 EX DE,HL Se pregătește deîmpărțitul și se revine dacă este zero (rezultatul este deja zero).
 CALL 30C0,PREP-M/D Se schimbă indicatorii.
 RET C Se salvează următoarea adresă corectă.
 EXX PUSH HL Se schimbă indicatorii.
 EXX PUSH DE Salvare indicator împărțitor.
 PUSH HL Salvare indicator deîmpărțit.
 CALL 2FBA,FETCH-TWO Se adu cele două numere din stivă.
 EXX PUSH HL Schimbare indicatori.
 LD H,B Se salvează M1 & M1 în stiva mașinii.
 LD L,C Se copiază cei patru octeți ai deîmpărțitului din registrii B'C'CB (adică M2, M3, M4 & M5 - vezi FETCH-TWO) în registrii H'L'HL.
 LD L,B
 XOR A Se șterge A și se resetează fanionul de transport.
 LD B,+DF B va număra în sus de la -33 la -1, în complement față de doi. DF la FF hex, ciclînd în minus și va sări din nou la zero pentru precizie mărită.
 JR 31E2,DIV-START Salt înainte în bucla de împărțire pentru prima încercare de scădere.

Acum se intră în bucla de împărțire.

31D2 DIV-LOOP RLA Se deplasează rezultatul la stînga în B'C'CA, deplasînd afară bitii ce sînt deja acolo, scotînd 1 din transport de cîte ori este setat, și rotînd la stînga fiecare octet cu transportul pentru a ajunge la deplasarea bitului 32.
 RL C
 EXX
 RL C
 RL B
 EXX
 31DB DIV-34TH ADD HL,HL Se transferă ce a mai rămas din deîmpărțit la stînga în H'L'HL înainte de următoarea încercare de scădere; dacă un bit coboară în transport, se forțază neînapoierea și un bit pentru cîț, care regăsește bitul pierdut și permite un împărțitor plin de 32 biti.
 EXX
 ADC HL,HL
 EXX
 JR C,31F2,SUBN-ONLY

31E2 DIV-START	SBC	HL, DE	Se încearcă scăderea împărțitorului din D'E'DE din restul de împărțitului din M'L'HL; nu este nici un transport initial (vezi pasul anterior).
	EXX		
	SBC	HL, DE	
	EXX		
	JR	NC, 31F9, NO-RSTORE	Salt înainte dacă nu este transport.
	ADD	HL, DE	Altfel se readuce, adică se adună înapoi împărțitorul.
	EXX		
	ADC	HL, DE	Apoi se șterge transportul astfel încât acolo nu va fi nici un bit pentru cât (divizorul 'nu a mers').
	EXX		
	AND	A	Salt înainte la numărator.
31F2 SUBN-ONE	JR	31FA, COUNT-ONE	Se scade doar, fără nici o înapoiere și se trece la setarea fanionului de transport deoarece bitul pierdut al de împărțitului trebuie regăsit și folosit pentru cât.
	AND	A	Unu pentru cât în B'C'CA.
	SBC	HL, DE	Se trece prin două contorizând unu.
	EXX		
	SBC	HL, DE	Se cicleză de 32 de ori pentru toți bitii.
	EXX		
31F9 NO-STORE	SCF		Se salvează fiecare bit 33 pentru o precizie suplimentară (transportul prezent).
31FA COUNT-ONE	INC	B	Se încearcă încă o dată scăderea pentru fiecare al 34-lea bit; PUSH AF de deasupra salvează și acest bit.
	JP	M, 31D2, DIV-LOOP	
	PUSH	AF	
	JR	Z, 31E2, DIV-START	

Notă: Acest salt se face într-un loc greșit. Nici un al 34-lea bit nu va fi niciodată obținut (fără a fi mai întâi deplasat la stînga. Apoi rezultatele importante ca 1/10 și 1/1000 nu sînt rotunjite așa cum ar trebui. Rotunjirea nu intervine niciodată cînd depinde de al 34-lea bit. Saltul trebuia să fi fost la 31DB DIV-34TH după; adică octetul 3200 hex în ROM trebuie să citească DA hex (218 zecimal) în locul lui E1 hex (225 zecimal).

LD	E, A	Acum se mută cei patru octeți care formează octeții mantisei rezultatului din B'C'CA în D'E'DE.
LD	D, C	
EXX		
LD	E, C	
LD	D, B	
POP	AF	Apoi se pun bitii 34 și 33 în B' pentru a fi scoși dacă nu este normalizare.
RR	B	
POP	AF	
RR	B	
EXX		
POP	BC	Se readuc octeții exponent MI & MI.
POP	HL	Readucere indicator rezultat.
LD	A, B	Se aduce diferența dintre cei doi octeți exponent în A și se setează fanionul de transport dacă se cere.
SUB	C	Iesire prin DIVN-EXPT.
JP	31D0, DIVN-EXPT	

THE 'INTEGER TRUNCATION TOWARDS ZERO' SUBROUTINE (SUBROUTINA 'TRUNCHIERE INTREG FATA DE ZERO')
(Deplasament 3A - vezi CALCULATE după: 'truncate')

Această subrutină (numită I(x)) redă rezultatul trunchierii numărului întreg x, 'ultima valoare', față de zero. Astfel, I(2,4) este 2 și I(-2,4) este -2. Subrutina revine imediat dacă x este în forma de 'întreg scurt' ('mic'). Ea redă zero dacă octetul exponent a lui x este mai mic decît 81 hex (ABS x este mai mic decît 1). Dacă I(x) este 'un întreg scurt' subrutina îl readuce în acea formă. x este redat dacă octetul exponent este A0 hex sau mai mare (x are partea neîntreagă nesemnificativă). Altfel numărul corect de octeți ai lui x sînt setați pe zero și, dacă este necesar, încă un octet este despărțit cu o mască.

3214 truncate LD A, (HL) Se aduce octetul exponent al

	AND	A	lui x în A.	
	RET	Z	Dacă A este zero, se revine pînă cînd x este deja un întreg mic.	
	CP	+81	Se compară e, exponentul, cu 81 hex.	
	JR	NC,3221,T-GR-ZERO	Salt dacă e este mai mare decît 80 hex.	
	LD	(HL),+00	Altfel, setare exponent pe zero; se introduce 32 zecimal	
	LD	A,+20	20 hex, în A și salt mai departe la NIL-BYTES pentru a face totii bitii lui x zero.	
	JR	3272,NIL-BYTES	Se compară e cu 91 hex, 145 zecimal.	
3221	T-GR-ZERO	CP	+91	Se compară e cu 91 hex, 145 zecimal.
3223	JR	NZ,323F,T-SMALL	Salt dacă e nu este 91 hex.	

Următorii 26 octeti par a descrie testul dacă x este de fapt -65536 zecimal, adică 91 80 00 00 00, și dacă este, se setează pe 00 FF 00 00 00. Aceasta este o greșeală. Cum deja s-a explicat la octetul 303B de deasupra, sistemul Spectrum nu poate trata acest număr. Aici rezultatul este simplu de făcut INT (-65536) să revină la valoarea -1. Este păcat, pînă cînd numărul nu va putea fi perfect în regulă dacă este lăsat singur. Remediul pare a fi simplu, de a omite 28 octeti din 3223 deasupra lui 323E inclusiv din program.

3225	INC	HL	HL indică cei patru octeti ai lui x, unde cei 17 bitii ai părții întregi a lui x se termină după primul bit.	
	INC	HL	Primul bit este obținut în A, folosind 80 hex ca mască.	
	LD	A,+80	Acest bit și cei 8 bitii anteriori sînt testați împreună față de zero.	
	AND	(HL)	HL indică al doilea octet din HL.	
	DEC	HL	Dacă încă este diferit de zero, testul se poate termina.	
	OR	(HL)	Altfel, testul pentru -65536 este acum completat: 91 80 00 00 00 va lăsa acum fanionul de zero setat.	
	DEC	HL	HL indică primul octet a lui x.	
	JR	NZ,3223,T-FIRST	Dacă zero este resetat, se face saltul.	
	LD	A,+80	Primul octet este setat pe zero.	
	XOR	(HL)	HL indică al doilea octet. Al doilea octet este setat pe FF.	
3233	T-FIRST	DEC	HL	HL indică din nou primul octet.
	JR	NZ,326C,T-EXPONENT	Ultimii 24 de bitii trebuie să fie zero.	
	LD	(HL),A	Saltul la NIL-BYTES completează numărul 00 FF 00 00 00.	
	INC	HL		
	LD	(HL),+FF		
	DEC	HL		
	LD	A,+18		
	JR	3272,NIL-BYTES		

Dacă octetul exponent al lui x este între 81 și 90 hex (129 și 144 zecimal) inclusiv, I(x) este un 'întreg mic', și va fi comprimat în unul sau doi octeti. Dar mai întîi se face un test pentru a vedea dacă x este, după toate, un număr mare.

323F	T-SMALL	JR	NC,326D,X-LARGE	Salt cu octetul exponent 92 sau mai mare (este mai bine de sărit cu 91).
		PUSH	DE	Se salvează STKEND în DE.
		CPL		Intervalul 129<=A<=144 devine 126<=A<=111.
		ADD	A,+91	Intervalul este acum 15 dec=> A>=0.
		INC	HL	HL indică al doilea octet.
		LD	D,(HL)	Al doilea octet în D.
		INC	HL	HL indică al treilea octet.
		LD	E,(HL)	Al treilea octet în E.
		DEC	HL	HL indică din nou primul octet.
		DEC	HL	

	LD	C,+00	Se presupune un număr pozitiv
	BIT	7,D	Acum se testează dacă este negativ (bitul 7 setat).
	JR	Z,3252,T-NUMERIC	Salt dacă este pozitiv.
3252 T-NUMERIC	DEC	C	Se schimbă semnul.
	SET	7,D	Se inserează bitul adevărat numeric 1 în D.
	LD	B,+08	Acum se testează dacă A>=8 (numai un octet) sau doi octeti sînt necesari.
	SUB	B	Se lasă A neschimbat.
	ADD	A,B	Salt dacă sînt necesari doi octeti.
	JR	C,325E,T-TEST	Se pune un octet în E.
	LD	E,D	Si se setează D pe zero.
	LD	D,+00	Acum $1 \leq A \leq 7$ pentru a număra deplasările necesare.
325E T-TEST	SUB	B	Salt dacă nu este necesară nici o deplasare.
	JR	Z,3267,T-STORE	B va contoriza deplasările.
	LD	B,A	Se deplasează D și E la dreapta de B ori pentru a da numărul corect.
3261 T-SHIFT	SRL	D	Se cicleză pînă B este zero.
	RR	E	Se stochează rezultatul în stivă.
	DJNZ	3261,T-SHIFT	Se refacă STKEND în DE.
3267 T-STORE	CALL	208E,INT-STORE	Sfîrșit.
	POP	DE	
	RET		

A rămas să se ia în considerare valorile mari pentru x.

326C T-EXPONENT	LD	A,(HL)	Se aduce octetul exponent al lui x în A.
326D X-LARGE	SUB	+A0	Se scade din e 160 zecimal, A0 hex.
	RET	P	Se revine pentru plus - partea întregă a lui x este nesemnificativă. (Dacă adevăratul exponent a fost redus la zero, 'punctul binar' va veni la sau după sfîrșitul celor patru octeti ai mantisei).
	NEG		Altfel, este negat restul; aceasta dă numărul de biti ce vor deveni zero (numărul de biti de după 'punctul binar').

Acum pot fi stersi bitii mantisei.

3272 NIL-BYTES	PUSH	DE	Se salvează valoarea curentă a lui DE (STKEND).
	EX	DE,HL	HL va indica cu unu după al cincilea octet.
	DEC	HL	Acum HL indică al cincilea octet al lui x.
	LD	B,A	Se aduce numărul bitilor ce vor fi setati pe zero în B și se împarte cu 8 pentru a da numărul tuturor bitilor implicați.
	SRL	B	Salt dacă rezultatul este zero.
	SRL	B	Altfel, se setează octetii pe zero;
	SRL	B	B îi numără.
327E BYTE-ZERO	JR	Z,3283,BITS-ZERO	
	LD	(HL),+00	Se aduce A (mod 8); acesta este numărul de biti care mai trebuie setati pe zero.
	DEC	HL	Salt la sfîrșit dacă nu mai este nimic de făcut.
3283 BITS-ZERO	DJNZ	327E,BYTE-ZERO	B va număra bitii acum.
	AND	+07	Pregătire mască.
	JR	Z,3290,IX-END	Cu fiecare buclă un zero intră în mască și astfel se realizează o mască de lungime corectă.
	LD	B,A	
	LD	A,+FF	
328A LESS-MASK	SLA	A	
	DJNZ	328A,LESS-MASK	

	AND	(HL)	Bitii nedoriti din (HL) sînt pierduti cum s-a făcut mascarea.
	LD	(HL),A	Revenire indicator în HL.
3290 IX-END	EX	DE,HL	Se readuce STKEND în DE.
	POP	DE	Sfîrsit.
	RET		

THE 'RE-STACK TWO' SUBROUTINE (SUBROUTINA 'RE-STACK TWO')
 Această subrutină este apelată pentru a restoca doi 'întregi mici' în totalitatea celor cinci octeti ai formei în virgulă mobilă pentru operațiile binare de adunare, înmulțire și împărțire. Acestea sînt realizate apelînd subrutina următoare de două ori.

3293 RE-ST-TWO	CALL	3296,RESTK-SUB	Apelare subrutină, și apoi se continuă în ea pentru a doua apelare.
3296 RESTK-SUB	EX	DE,HL	Se interschimbă indicatorii la fiecare apelare.

THE 'RE-STACK' SUBROUTINE (SUBROUTINA 'RE-STACK')
 (Deplasament 3D - vezi CALCULATE după: 're-stack')

Această subrutină este apelată pentru a restoca un număr (care poate fi un 'întreg mic') în totalitatea celor cinci octeti ai formei în virgulă mobilă. Este folosită pentru un singur număr de ARCTAN și de asemenea, de-a lungul deplasamentului calculatorului, de EXP, LN și 'get-argt'.

3297 RE-STACK	LD	A,(HL)	Dacă primul octet nu este zero, se revine - numărul nu poate fi un 'întreg mic'.
	AND	A	Se salvează 'celălalt' indicator în DE.
	RET	NZ	Se aduce semnul în C și numărul în DE.
	PUSH	DE	Se șterge registrul A.
	CALL	2D7F,INT-FETCH	Se indică a cincea locație.
	XOR	A	Se setează al cincilea octet pe zero.
	INC	HL	Se indică a patra locație.
	LD	(HL),A	Se setează a patra locație pe zero; octetii 2 și 3 vor conține mantisa.
	DEC	HL	Se setează B pe 145 zecimal pentru exponent, adică pentru mai mult de 16 biti în întreg.
	LD	B,+91	Se testează dacă D este zero, astfel încît să fie necesari cel mult 8 biti.
	LD	A,D	Salt dacă este nevoie de mai mult de 8 biti.
	AND	A	Acum se testează si E.
	JR	NZ,32B1,RS-NRMLSE	Se salvează zero în B (va da exponent zero dacă si E este tot zero).
	OR	E	Salt dacă E este într-adevăr zero.
	LD	B,D	Se mută E în D (D a fost zero E nu).
	JR	Z,32BD,RS-STORE	Se setează E pe zero acum.
	LD	D,E	Se setează B pe 137 zecimal pentru exponent - nu mai mult de 8 biti acum.
	LD	E,B	Indicatorul în DE, numărul în HL.
	LD	B,+89	Se decrementează exponntul la fiecare deplasare.
32B1 RS-NRMLSE	EX	DE,HL	Se deplasează numărul la dreapta cu o poziție.
32B2 RSTK-LOOP	DEC	B	Pînă cînd transportul este setat.
	ADD	HL,HL	Bitul de semn în fanionul de transport acum.
	JR	NC,32B2,RSTK-LOOP	Este inserat în locul unde numărul s-a deplasat înapoi cu un spațiu - acum normal.
	RRC	C	Indicatorul octetului 4 înapoi în HL.
	RR	H	Se indică a treia locație.
	RR	L	
	EX	DE,HL	
32BD RS-STORE	DEC	HL	

LD	(HL),E	Se stochează al treilea octet
DEC	HL	Se indică a doua locație.
LD	(HL),D	Se stochează al doilea octet.
DEC	HL	Se indică prima locație.
LD	(HL),B	Se stochează octetul exponent
POP	DE	Se readuce 'celălalt'
		indicator în DE.
RET		Sfârșit.

THE FLOATING-POINT CALCULATOR (DISPOZITIVUL DE CALCUL IN VIRGULA MOBILA)

THE TABLE OF CONSTANTS (TABELUL DE CONSTATE)

Acest prim tabel contine cele mai folosite si frecvent necesare cinci numere zero, unu, jumătate, jumătatea lui pi si zece. Numerele sînt continute într-o formă condensată care este extinsă cu subrutina STACK LITERALS, vezi mai jos, pentru a da forma în virgulă mobilă cerută.

	data	constanta	prin extensie se obtine: exp.mantixa: ((Hex.)
32C5 stk-zero	DEFB +00 DEFB +80 DEFB +00	zero	00 00 00 00 00
32C8 stk-one	DEFB +40 DEFB +80 DEFB +00 DEFB +01	unu	00 00 01 00 00
32CC stk-half	DEFB +30 DEFB +00	jumătate	80 00 00 00 00
32CE stk-pi/2	DEFB +F1 DEFB +49 DEFB +0F DEFB +DA DEFB +A2	jumătatea lui pi	81 49 0F DA A2
32D3 stk-ten	DEFB +40 DEFB +80 DEFB +00 DEFB +0A	zece	00 00 0A 00 00

THE TABLE OF ADDRESSES (TABELUL DE ADRESE)

Acest al doilea tabel este un tabel de căutare a adreselor celor 66 de subrutine operationale ale calculatorului. Deplasamentele folosite la indexarea din tabel provin fie din codurile operatiilor folosite în SCANNING, vezi 2734, etc., fie din literalurile care urmează instructia RST 0028.

deplasa- sament	etichetă	adresă	deplasa- sament	etichetă	adresă
32D7 00	jump-true	8F 36	3319 21	tan	DA 37
32D9 01	exchange	3C 34	331B 22	asn	33 38
32DB 02	delete	A1 33	331D 23	acs	43 38
32DD 03	subtract	0F 30	331F 24	atn	E2 37
21DF 04	multiply	CA 30	3321 25	ln	13 37
32E1 05	division	AF 31	3323 26	exp	CA 36
32E3 06	to-power	51 38	3325 27	int	AF 36
32E5 07	or	1B 35	3327 28	sqr	4A 38
32E7 08	no-&-no	24 35	3329 29	sgn	92 34
32E9 09	no-l-eql	3B 35	332B 2A	abs	4A 34
32EB 0A	no-gr-eq	3B 35	332D 2B	peek	4C 34
32ED 0B	nos-neql	3B 35	332F 2C	in	45 34
32EF 0C	no-grtr	3B 35	3331 2D	usr-no	83 34
32F1 0D	no-less	3B 35	3333 2E	strf	1F 36
32F3 0E	nos-eql	3B 35	3335 2F	chr#	69 35
32F5 0F	addition	14 30	3337 30	not	01 35
32F7 10	str-&-no	2D 35	3339 31	duplicate	C0 33

32F9	11	str-l-eq1	38	333B	32	n-and-a	40
			38				36
32FB	12	str-gr-eq	38	333D	33	jump	86
			38				36
32FD	13	strs-meq1	38	333F	34	stk-data	84
			38				33
32FF	14	str-grtr	38	3341	35	dec-ir-nz	78
			38				36
3301	15	str-less	38	3343	36	less-0	08
			38				35
3303	16	strs-eq1	38	3345	37	greater-0	F9
			35				34
3305	17	strs-add	9C	3347	38	end-calc	98
			35				36
3307	18	valf	DE	3349	39	get-argt	83
			35				37
3309	19	usr-f	8C	334B	3A	truncate	14
			34				32
330B	1A	read-in	45	334D	3B	fp-calc-2	A2
			36				33
330D	1B	negate	4E	334F	3C	e-to-fp	4F
			34				2D
330F	1C	code	49	3351	3D	re-stack	97
			36				32
3311	1D	val	DE	3353	3E	series-06	49
			35			etc.	34
3313	1E	len	74	3355	3F	stk-zero	1B
			36			etc.	34
3315	1F	sin	85	3357	40	st-mem-0	2D
			37			etc.	34
3317	20	cos	AA	3359	41	get-mem-0	0F
			37			etc.	34

Notă: Ultimele patru subrutine sînt subrutine multifuncționale și sînt introduse cu un parametru care este o copie a celor cinci biți din partea dreaptă a literalului original. Urmează prezentarea întregului set:

Deplasament 3E: series-06, series-08, & series-0C; literaluri 86, 88 & 8C.

Deplasament 36: stk-zero, stk-one, stk-half, stk-pi/2 & stk-ten; literaluri A0 la A4.

Deplasament 40: st-mem-0, st-mem-1, st-mem-2, st-mem-3, st-mem-4 & st-mem-5; literaluri C0 la C5.

Deplasament 41: get-mem-0, get-mem-1, get-mem-2, get-mem-3, get-mem-4 & get-mem-5; literaluri E0 la E5.

THE 'CALCULATE' SUBROUTINE (SUBROUTINA 'CALCULATE')

Această subrutină este folosită pentru a realiza calculele în virgulă mobilă. Acestea pot fi considerate ca fiind de trei tipuri:

i. Operații binare, cum ar fi adunarea, unde două numere în forma cu virgulă mobilă sînt adunate împreună pentru a da o 'valoare ultimă'.

ii. Operații unare, cum ar fi sin, unde 'ultima valoare' este schimbată pentru a da rezultatul corespunzător funcției ca o nouă 'ultima valoare'.

iii. Operații de manevră, cum ar fi st-mem-0, unde 'ultima valoare' este copiată în primii cinci octeți din zona de memorie a calculatorului.

Operațiile ce trebuie efectuate sînt specificate ca o serie de octeți de informații, literalurile, care urmează o instrucție RST 0028 care apelează această subrutină. Ultimul literal din listă este întotdeauna '3B', care conduce la terminarea întregii operații.

În cazul în care trebuie efectuată o singură operație, deplasamentul operației se poate trece în CALCULATOR în registrul B, și se efectuează operația '3B', operația SINGLE CALCULATION.

De asemenea, mai este posibil să se apeleze această subrutină recursiv, adică se cicleză pe însăși, și într-un asemenea caz este posibilă utilizarea variabilei sistem BREG ca un numărător care controlează cîte operații sînt efectuate înainte de revenire.

Prima parte a acestei subrutine este complicată, dar în esență ea realizează cele două taskuri pentru setarea registrilor astfel încît ei să conțină valorile cerute, și pentru a produce un deplasament și probabil un parametru, dintr-un literal, care a fost considerat în mod curent.

Deplasamentul este folosit pentru indexarea tabelului de adrese al calculatorului, vezi deasupra, pentru a găsi subrutina de adresă cerută.

Parametrul se folosește cînd sînt apelate subrutinele multifuncționale.

Notă: Un număr în virgulă mobilă poate fi în realitate un set de parametrii

sir.					
335B	CALCULATE	CALL	35RF,STK-PNTRS		
335E	GEN-ENT-1	LD LD	A,B (BREG),A		
3362	GEN-ENT-2	EXX EX EXX	(SP),HL		
3365	RE-ENTRY	LD	(STKEND),DE		
336C	SCAN-ENT	EXX LD	A,(HL)		
		INC	HL		
		PUSH	HL		
		AND JP	A P,3380,FIRST-3D		
		LD AND	D,A +60		
		RRCA RRCA RRCA RRCA ADD LD	A,+7C L,A		
		LD AND	A,D +1F		
		JR	338E,ENT-TABLE		
		3380	FIRST-3D	CP JR EXX LD LD LD ADD EXX	+18 NC,338C,DOUBLE-A BC,+FFFB D,A E,L HL,BC
		338C	DOUBLE-A	RLCA LD	L,A

Se presupune o operatie unara si de aceea se seteaza HL pentru a indica inceputul 'ultimei valori' din stiva calculatorului si DE unul dupa acest numar in virgula mobila (STKEND).

Sau se transfera temporar deplasamentul unei operatii singulare in BREG sau, cind se foloseste subrutina recursiv, se trece in BREG parametrul ce va fi folosit ca si contor.

Adresa de revenire a subrutinei se stocheaza in H'L'. Acesta salveaza indicatorul primului literal. Intrarea in CALCULATOR prin GEN-ENT-2 este folosita cind BREG este folosit ca si contor si nu trebuie conturbat.

Acum se introduce o bucla pentru a trata fiecare literal din lista care urmeaza instructiunea apelanta; asa ca mai intii, intotdeauna, se seteaza STKEND.

Se trece la setul de registrii auxiliari si se aduce literalul pentru aceasta bucla. H'L' va indica urmatorul literal.

Acest indicator este salvat pentru scurt timp in stiva masinii. SCAN-ENT este folosit de subrutina SINGLE CALCULATION pentru a afla subrutina care este ceruta.

Se testeaza registrul A. Se separa literalul simplu de literalurile multifunctionale Salt pentru literalurile 00 - 3D.

Salvare literal in D. Se continua doar cu bitii 5 & 4.

Patru deplasari la dreapta ii fac acum sa fie bitii 1 & 2.

Deplasamentele cerute sint 3E - 41 si L va contine acum copia deplasamentului cerut.

Acum se realizeaza parametrul luind bitii 1, 2, 3 & 4 ai literalului; se pastreaza parametrul in A.

Salt inainte pentru a determina adresa subrutinei cerute.

Salt inainte daca se efectueaza o operatie unara.

Toate subrutinele care efectueaza operatii binare solicita ca HL sa indice primul operand si DE sa indice al doilea operand ('ultima valoare'), asa cum apar in stiva calculatorului. Asa cum fiecare intrare in tabelul de adrese necesita doi octeti, deplasamentul realizat este dublat.

338E ENT-TABLE	LD	DE,+32D7	Adresa de bază a tabelului.
	LD	H,400	Adresa intrării tabelului
	ADD	HL,DE	cerut este formată în HL; iar
	LD	E,(HL)	adresa subrutinei cerute este
	INC	HL	încărcată în registrul
	LD	D,(HL)	pereche DE.
	LD	HL,+3365	Adresa RE-ENTRY la 3365 este
	EX	(SP),HL	pusă în stiva masinii sub
	PUSH	DE	adresa subrutinei.
	EXX		Revenire la setul de
	LD	BC,(STKEND-hi)	registrii principali.
			Valoarea curentă a lui BREG
			este transferată în registrul
			B astfel returnând
			deplasamentul operației
			singulare. (Vezi COMPARISON
			la 353B.)
33A1 delete	RET		Salt indirect la subrutina
			cerută.

THE 'DELETE' SUBROUTINE (SUBROUTINA 'DELETE' (stergere))
(Deplasament 02: 'delete')

Această subrutină conține doar instrucția singulară RET la 33A1, deasupra. Literalul '02' rezultat în această subrutină a fost considerat ca o operație binară care trebuie introdusă cu un prim număr adresat de registrul pereche HL și un al doilea număr adresat de registrul pereche DE și rezultatul obținut este din nou adresat de registrul pereche HL.

Instrucția singulară RET astfel conduce la primul număr ce a fost considerat ca 'ultima valoare' rezultantă și al doilea număr se consideră ca fiind sters. Bineînțeles că numărul nu a fost sters din memorie, dar rămâne inactiv și probabil că în curând va fi suprăimprimat.

THE 'SINGLE OPERATION' SUBROUTINE (SUBROUTINA 'OPERATIE SINGULARA')
(Deplasament 3B: 'fp-calc-2')

Această subrutină este apelată numai din SCANNING la 2757 hex și este folosită pentru a efectua o operație aritmetică singulară. Deplasamentul care specifică ce operație trebuie efectuată este înlocuit în calculator în registrul B și transferat ulterior în variabila sistem BREG.

Efectul apelării acestei subrutine este în esență de a executa un salt la subrutina corespunzătoare pentru operația singulară.

33A2 fp-calc-2	POP	AF	Se înlătură adresa RE-ENTRY.
	LD	A,(BREG)	Se transferă deplasamentul în
			A.
	EXX		Se introduce setul de
			registrii auxiliari.
	JR	336C,SCAN-ENT	Salt înapoi pentru a găsi
			adresa cerută; se stochează
			adresa RE-ENTRY și salt la
			subrutina pentru operație.

THE 'TEST 5-SPACES' SUBROUTINE (SUBROUTINA 'TESTARE 5 SPATII')

Această subrutină testează dacă în memorie există suficient spațiu pentru alți cinci octeți de număr în virgulă mobilă care să fie adăugați în stiva calculatorului.

33A9 TEST-5-SP	PUSH	DE	Se adresează DE pentru scurt
			timp.
	PUSH	HL	Se salvează HL pentru scurt
			timp.
	LD	BC,+0005	Se specifică faptul că testul
			se face pentru 5 octeți.
	CALL	1F05,TEST-ROOM	Se execută testul.
	POP	HL	Se readuce HL.
	POP	DE	Se readuce DE.
	RET		Sfârșit.

THE 'STACK NUMBER' SUBROUTINE (SUBROUTINA 'STOCARE NUMAR')

Această subrutină este apelată de DEEP și de SCANNING de două ori pentru a copia STKEND în DE, pentru a muta numărul în virgulă mobilă în stiva calculatorului și pentru a reseta STKEND din DE. Pentru a face mutarea actuală se apelează 'MOVE-FP'.

3384 STACK-NUM	LD	DE,(STKEND)	Se copiază STKEND în DE ca
			adresa destinației.
	CALL	33C0,MOVE-FP	Se mută numărul.
	LD	(STKEND),DE	Se resetează STKEND din DE.

THE 'MOVE A FLOATING-POINT NUMBER' SUBROUTINE (SUBROUTINA 'MUTARE NUMAR IN VIRGULA MOBILA')
(Deplasament 31: 'duplicate')

Această subrutină mută un număr în virgulă mobilă în vârful stivei calculatorului (3 cazuri) sau din vârful stivei în spațiul memoriei calculatorului (1 caz). Este de asemenea apelată în calculator când pur și simplu se dublează numărul din vârful stivei calculatorului, 'ultima valoare' astfel extinzând stiva cu cinci octeți.

33C0 MOVE-FP	CALL 33A9,TEST-5-SP	Se face un test pentru spațiu
	LDIR	Se mută cei cinci octeți implicați.
	RET	Sfîrșit.

THE 'STACK LITERALS' SUBROUTINE (SUBROUTINA 'STOCARE LITERALURI')
(Deplasament 34: 'stk-data')

Această subrutină plasează în stiva calculatorului ca o 'ultimă valoare' numărul în virgulă mobilă ce o înlocuiește ca 2, 3, 4 sau 5 literaluri. Când este apelată folosind deplasamentul '34' literalul urmează '34' în lista de literaluri; când este apelată de SERIES GENERATOR, vezi în continuare, literalurile sînt înlocuite de subrutina care apelează pentru o serie care trebuie generată; iar când este apelată de SKIP CONSTANTS & STACK A CONSTANT literalurile se obțin din tabelul de constante al calculatorului (32C5 - 32D4).

În fiecare caz, primul literal înlocuit este divizat de 40 hex și răstul întreg plus 1 determină dacă 1, 2, 3 sau 4 literaluri în continuare vor fi luate din sursă pentru a forma mantisa numărului. Oricare octeți incompleți din cei cinci octeți care vor forma un număr în virgulă mobilă pe cinci octeți sînt setați pe zero. Primul literal este de asemenea folosit pentru a se determina exponentul, după reducerea mod. 40 hex, în afară de cazul când restul este zero, caz în care se folosește al doilea literal, așa cum se găsește el, fără reducere în mod 40 hex. În fiecare caz, se adaugă 50 hex la literal, rezultînd octetul exponent mărît, e (adevăratul exponent e plus 80 hex.). Restul din cei cinci octeți sînt stocați, incluzînd orice zerouri necesare, și subrutina revine.

33C6 STK-DATA	LD H,D	Această subrutină realizează operația de manevră a adăugării unei ultime valori în stiva calculatorului; deci HL este setat pentru a indica cu un număr după ultima valoare prezentă și deci indică rezultatul.
	LD L,E	

33C8 STK-CONST	CALL 33A9,TEST-5-SP	Se testează dacă există într-adevăr loc.
	EXX	Se trece la setul regisrilor auxiliari și se stochează indicatorul pentru următorul literal.
	PUSH HL	
	EXX	
	EX (SP),HL	Se schimbă indicatorul rezultatului și indicatorul următorului literal.
	PUSH BC	Se salvează BC pentru scurt timp.
	LD A,(HL)	Primul literal este pus în A și împărțit cu 40 hex pentru a da valorile întregi 0, 1, 2 sau 3.
	AND +C0	Valoarea întregă este transferată în C și incrementată, dînd astfel ordinul 1, 2, 3 sau 4 numărului de literaluri care va fi necesar.
	RLCA	Literalul este adus din nou; reducerea mod 40 hex. și înălțurare ca necorespunzător dacă restul este zero; caz în care următorul literal este adus și utilizat neredus.
	RLCA	
	LD C,A	Exponentul, e, este format prin adunarea cu 50 hex. și este trecut în stiva calculatorului ca fiind
	INC C	
	LD A,(HL)	
	AND +3F	
	JR NZ,33DE,FORM-EXP	
	INC HL	
	LD A,(HL)	

33DE FORM-EXP	ADD A,+50	
	LD (DE),A	

	LD	A,+05	primul din cei cinci octeti si rezultatului.
	SUB	C	Numarul de literaluri specificat in C este luat din sursa si introdus in octetii rezultatului.
	INC	HL	
	INC	DE	
	LD	B,+00	
	LDIR		
	POP	BC	Se restituie BC.
	EX	(SP),HL	Se readuce indicatorul rezultatului in HL si indicatorul urmatorului literal in pozitia sa obisnuita in H' & L'.
	EXX		
	POP	HL	
	EXX		
	LD	B,A	Numarul cerut de octeti zero in aceasta faza este dat de 5-C-1; si acest numar de zerouri este adaugat rezultatului pentru a forma cei cinci octeti ceruti.
33F1	XOR	A	
	DEC	B	
	RET	Z	
	LD	(DE),A	
	INC	DE	
	JR	33F1,STK-ZEROS	

THE 'SKIP CONSTANTS' SUBROUTINE (SUBROUTINA 'OMITERE CONSTANTE')

Aceasta subrutina este introdusa cu registrul pereche HL continand adresa de baza a tabelului de constante a calculatorului si cu registrul A continand un parametru care arata care dintre cele cinci constante a fost ceruta.

Subrutina efectueaza operatia invalida de incarcare a celor cinci octeti pentru fiecare constanta nedorita in locatiile 0000, 0001, 0002, 0003 si 0004 la inceputul lui ROM ina cind constanta ceruta este gasita.

Subrutina revine cu registrul pereche HL continand adresa de baza a constantei cerute in cadrul tabelului de constante.

33F7	AND	A	Revenire subrutina daca
33F8	RET	Z	parametrul a fost zero sau
			cind constanta ceruta a fost
			gasita.
	PUSH	AF	Salvare parametru.
	PUSH	DE	Salvare indicator rezultat.
	LD	DE,+0000	Inlocuire adresa.
	CALL	33C8,STK-CONST	Se efectueaza o stocare
			imaginara a unei constante
			dezvoltate.
	POP	DE	Se readuce indicatorul
			rezultatului.
	POP	AF	Se readuce parametru.
	DEC	A	Contorizare bucle.
	JR	33F8,SKIP-NEXT	Salt inapoi pentru a
			considera valoarea
			contorului.

THE 'MEMORY LOCATION' SUBROUTINE (SUBROUTINA 'LOCATIE DE MEMORIE')

Aceasta subrutina gaseste adresa de baza pentru fiecare portiune de cinci octeti din spatiul de memorie al calculatorului in care sau din care un numar in virgula mobila trebuie mutat din sau in stiva calculatorului. Ea realizeaza aceasta operatie adaugand de cinci ori parametrul inlocuit la adresa de baza a zonei care este continuata in registrul pereche HL.

De notat ca atunci cind o variabila FOR-NEXT a fost tratata, atunci indicatorii sint schimbati astfel incit variabila este tratata ca si cind ea ar fi fost in zona de memorie a calculatorului (vezi adresa 1D20).

3406	LD	C,A	Se copiaza parametrul in C.
	RLCA		Se dubleaza parametrul.
	RLCA		Se dubleaza acest rezultat.
	ADD	A,C	S aduna valoarea parametrului
			pentru a obtine de cinci ori
			valoarea initiala.
	LD	C,A	Acest rezultat este dorit in
	LD	B,+00	registrul pereche BC.
	ADD	HL,BC	Se produce noua adresa de
			baza.
	RET		Sfirsit.

THE 'GET FROM MEMORY AREA' SUBROUTINE (SUBROUTINA 'ADUCERE DIN ZONA DE MEMORIE')

(Deplasamente C0 la C5: 'st-mem-0' la 'st-mem-5')

Această subrutină este apelată folosind literalurile E0 la E5, iar parametrul provenit din aceste literaluri este conținut în registrul A. Subrutina apelează MEMORY LOCATION pentru a pune adresa sursă cerută în registrul pereche HL și MOVE A FLOATING-POINT NUMBER pentru a copia cei cinci octeți implicați din spațiul de memorie al calculatorului în vârful stivei calculatorului pentru a forma o nouă 'valoare ultimă'.

340F get-mem-0 etc.	PUSH DE LD HL, (MEM)	Salvare indicator rezultat. Se aduc indicatorul în zona curentă de memorie (vezi deasupra).
	CALL 3406, LOC-MEM	S-a găsit adresa de bază.
	CALL 33C0, MOVE-FP	Cei cinci octeți sînt mutați.
	POP HL	Setare indicator rezultat.
	RET	Sfîrșit.

THE 'STACK A CONSTANT' SUBROUTINE (SUBROUTINA 'STOCARE CONSTANTA')
(Deplasamente A0 la A4: 'stk-zero', 'stk-one', 'stk-half', 'stk-pi/2' & 'stk-ten')

Această subrutină folosește SKIP CONSTANTS pentru a găsi adresa de bază a constantelor cerute din tabelul de constante a calculatorului și apoi apelează STACK LITERALS, introdus pe la STK-CONST, pentru a realiza forma extinsă a constantei 'ultima valoare' din stiva calculatorului.

341B stk-zero etc.	LD H, D	Se setează HL pentru a conține indicatorul rezultat.
	LD L, E	
	EXX	Se trece la setul de registrii auxiliari și se salvează următorul indicator al literalului.
	PUSH HL	
	LD HL, +3205	Adresa de bază a tabelului de constante a calculatorului.
	EXX	Înapoi la setul principal de registrii.
	CALL 33F7, SKIP-CONS	Se găsește adresa de bază cerută.
	CALL 33C8, STK-CONST	Extindere constantă.
	EXX	
	POP HL	Se readuce următorul indicator literal.
	EXX	
	RET	Sfîrșit.

THE 'STORE IN MEMORY AREA' SUBROUTINE (SUBROUTINA 'STOCARE ÎN ZONA DE MEMORIE')
(Deplasamente de la C0 la C5: 'st-mem-0' la 'st-mem-5')

Această subrutină este apelată folosind literalurile de la C0 la C5 iar parametrul provenit din aceste literaluri este conținut în registrul A. Această subrutină este foarte asemănătoare cu subrutina GET FROM MEMORY, dar indicatorii sursă și destinație sînt schimbați.

342D st-mem-0 etc.	PUSH HL	Salvare indicator rezultat.
	EX DE, HL	Sursa în DE pentru scurt timp
	LD HL, (MEM)	Se aduce indicatorul în zona curentă de memorie.
	CALL 3406, LOC-MEM	S-a găsit adresa de bază.
	EX DE, HL	Se interschimbă indicatorii sursă și destinație.
	CALL 33C0, MOVE-FP	Cei cinci octeți sînt transferați.
	EX DE, HL	'Ultima valoare' +5, adică STKEND, în DE.
	POP HL	Indicatorul rezultat în HL.
	RET	Sfîrșit.

De notat că indicatorii HL și DE rămîn așa cum au fost, indicînd STKEND-5 și respectiv STKEND, astfel încît 'ultima valoare' rămîne în stiva calculatorului. Dacă se cere, aceasta poate fi îndepărtată prin utilizarea lui 'delete'.

THE 'EXCHANGE' SUBROUTINE (SUBROUTINA 'INTERSCIMBARE')
(Deplasament 01: 'exchange')

Această operație binară 'interschimbă' primul număr cu al doilea număr, aceasta însemnînd că cele două numere din vârful stivei calculatorului sînt interschimbate.

343E EXCHANGE	LD	A;+05)	Sînt implicați cinci octeți.
343E SWAP-BYTE	LD	C,(HL)	Fiecare octet al celui de al
	EX	DE,HL	doilea număr.
	LD	(DE),A	Fiecare octet al primului
	LD	(HL),C	număr.
	INC	HL	Se schimbă sursa și
	INC	DE	destinatia.
	DJNZ	343E,SWAP-BYTE	Acum la primul număr.
	EX	DE,HL	Acum la al doilea număr.
	RET		Se trece la considerarea
			următoarei perechi de octeți.
			Se interschimbă cei cinci
			octeți.
			Se corectează indicatorii ca
			și cum numărul 5 este un
			număr impar.
			Sfîrșit.

THE 'SERIES GENERATOR' SUBROUTINE (SUBROUTINA 'GENERATOR SERII')
(Deplasamentele 86, 88 & 8C: 'series-06', 'series-08' & 'series-0C')

Această subrutină importantă generează seriile polinoamelor lui Chebisev, care se folosesc la aproximarea lui SIN, ATN, LN și EXP și de aici la derivarea altor funcții aritmetice care depind de acestea (COS, TAN, ASN, ACS, ** și SQR).

Polinoamele sînt generate, de la $n = 1, 2, \dots$, prin relația de recurență:

$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z)$ unde $T_n(z)$ este al n-lea polinom Chebisev în z .

Seriile de fapt generează:

$T_0, 2T_1, 2T_2, \dots, 2T_{n-1}$, unde n este 6 pentru SIN, 8 pentru EXP și 12 zecimală pentru LN și ATN.

Coefficientii puterilor lui z din aceste polinoame pot fi găsiți în "Handbook of Mathematical Functions" de M. Abramowitz și I.A. Stegun (Dover 1965), pag. 795.

Programele BASIC care arată generarea fiecăreia din cele patru funcții este dată în Anexă.

În termeni simplii, această subrutină este apelată cu 'ultima valoare' în stiva calculatorului, numită Z, fiind un număr care duce la o legătură simplă cu argumentul, numit X, cînd taskul trebuie să evalueze, de exemplu, SIN X. Subrutina apelantă de asemenea furnizează lista constantelor care vor fi cerute (șase constante pentru SIN). SERIES GENERATOR manevrează apoi informațiile sale și revine la rutina apelantă ca o 'ultimă valoare' care duce la o simplă legătură cu funcția cerută, de exemplu, SIN X.

Se poate considera că această subrutină are patru părți mari:

i. Setarea contorului buclei:

Subrutina apelantă trece parametrii săi în registrul A pentru utilizarea lui ca și contor. Intrarea se face pe la GEN-ENT-1, așa că se poate seta contorul.

3449 series-06	LD	B,A	Se mută parametrul în B.
etc.	CALL	335E,GEN-ENT-1	De fapt o instrucțiune RST
			0028 setează contorul.

ii. Tratarea 'ultimei valori', Z:

Bucșa generatorului cere plasarea lui $2*Z$ în mem-0, zero să fie plasat în mem-2 și 'ultima valoare' să fie zero.

		stiva calculatorului
DEFB	+31,duplicate	Z,Z
DEFB	+0F,addition	2*Z
DEFB	+C0,st-mem-0	2*Z mem-0 contine 2*Z
DEFB	+02,delete	-
DEFB	+A0,stk-zero	0
DEFB	+C2,st-mem-2	0 mem-2 contine 0

iii. Bucșa principală:

Seria este generată prin ciclare, folosind BREG ca și contor; constantele din subrutina apelantă se stochează pe rînd apelînd STK-DATA; calculatorul intră din nou prin GEN-ENT-2, așa că nu trebuie conturbată valoarea lui BREG; și seria este construită în forma:

$$B(R) = 2*Z*B(R-1) - B(R-2) + A(R), \text{ pentru } R = 1, 2, \dots, N,$$

unde $A(1), A(2), \dots, A(N)$ sînt constantele înlocuite de rutina apelantă (SIN, ATN, LN și EXP) și $B(0) = 0 = B(-1)$.

A (R+1)-a buclă începe cu $B(R)$ în stivă și cu $2*Z, B(R-2)$ și $B(R-1)$ în mem-0, mem-1 și mem-2 respectiv.

3453 G-LOOP	DEFB	+31,duplicate	B(R),B(R)
-------------	------	---------------	-----------

DEFB	+E0, get-mem-0	B(R), B(R), 2*Z
DEFB	+E2, get-mem-2	B(R), 2*B(R)*Z, B(R-1)
DEFB	+C1, st-mem-1	mem-1 contine B(R-1)
DEFB	+03, subtract	B(R), 2*B(R)*Z-B(R-1)
DEFB	+38, end-calc	

Se pune în stiva calculatorului următoarea constantă.

CALL	33C6, STK-DATA	B(R), 2*B(R)*Z-B(R-1), A(R+1)
------	----------------	-------------------------------

Calculatorul reintră fără a-1 conturba pe BREG.

CALL	3362, GEN-ENT-2	
DEFB	+0F, addition	B(R), 2*B(R)*Z-B(R-1)+A(R+1)
DEFB	+01, exchange	2*B(R)*Z-B(R-1)+A(R+1), B(R)
DEFB	+C2, st-mem-2	mem-2 contine B(R)
DEFB	+02, delete	2*B(R)*Z-B(R-1)+A(R+1)=B(R+1)
DEFB	+35, dec-jr-nz	B(R+1)
DEFB	+EE, to3453, G-LOOP	

iv. Scăderea lui B(N-2):

Bucula de deasupra lasă B(N) în stivă și rezultatul cerut este dat de B(N)-B(N-2).

DEFB	+E1, get-mem-1	B(N), B(N-2)
DEFB	+03, subtract	B(N)-B(N-2)
DEFB	+38, end-calc	
RET		Sfîrsit.

THE 'ABSOLUTE MAGNITUDE' FUNCTION (FUNCTIA 'MARIME ABSOLUTA')
(Deplasament 2A: 'abs')

Această subrutină efectuează operația unară prin asigurarea că bitul de semn al numărului în virgulă mobilă este resetat.

'Intregii mici' trebuie tratați separat. Cea mai mare parte a muncii se referă la operația 'minus unar'.

346A abs	LD	B, +FF	B este setat pe FF.
	JR	3474, NEG-TEST	Saltul se face la 'minus unar'.

THE 'UNARY MINUS' OPERATION (OPERATIA 'MINUS UNAR')
(Deplasament 1B: 'negate')

Această subrutină efectuează operația sa unară prin schimbarea semnului 'ultimei valori' din stiva calculatorului.

Zero este returnat pur și simplu neschimbat. Toți cei cinci octeți ai numărului în virgulă mobilă au bitul de semn modificat astfel încât în final va fi resetat (pentru 'abs') sau schimbat (pentru 'negate'). 'Intregii mici' au octetul lor de semn setat pe zero (pentru 'abs') sau schimbat (pentru 'negate').

346E NEGATE	CALL	34E9, TEST-ZERO	Dacă numărul este zero,
	RET	C	subrutina revine lăsînd
			00 00 00 00 00 neschimbat.
	LD	B, +00	B este setat pe +00 hex
			pentru 'negate'.

Aici se introduce 'ABS'.

3474 NEG-TEST	LD	A, (HL)	Dacă primul octet este zero,
	AND	A	se face saltul pentru a lucra
	JR	Z, 3483, INT-CASE	cu 'intregi mici'.
	INC	HL	Se indică al doilea octet.
	LD	A, B	Se aduce +FF pentru 'abs',
			+00 pentru 'negate'.
	AND	+80	Acum +80 pentru 'abs', +00
			pentru 'negate'.
	OR	(HL)	Aceasta setează bitul 7
			pentru 'abs', dar nu modifică
			nimic pentru 'negate'.
	RLA		Acum bitul 7 este schimbat,
	CCF		ceea ce conduce la resetarea
	RRA		bitului 7 din octetul 2
			pentru 'negate'.
	LD	(HL), A	Noul octet secund este
			stocat.
	DEC	HL	HL indică din nou primul

	RET	octet. Sfirsit.
'Cazul întregilor' dă o operație similară cu octetul semn.		
3483 INT-CASE	PUSH DE	Se salvează STKEND în DE.
	PUSH HL	Se salvează indicatorul numărului în HL.
	CALL 2D7F,INT-FETCH	Se aduce semnul în C, numărul în DE.
	POP HL	Se readuce indicatorul numărului în HL.
	LD A,B	Se aduce +FF pentru 'abs', +00 pentru 'negate'.
	OR C	Acum +FF pentru 'abs', nici o schimbare pentru 'negate'.
	CPL	Acum +00 pentru 'abs' și o schimbare octet pentru 'negate': stocare în C.
	LD C,A	Se stochează rezultatul în stivă.
	CALL 2D8E,INT-STORE	Se readuce STKEND în DE.
	POP DE	Sfirsit.
	RET	

THE 'SIGNUM' FUNCTION (FUNCTIA 'SEMN')
(Deplasament 29: 'sgn')

Această subrutină tratează funcția sgn X și de aceea redă o 'ultimă valoare' 1 dacă X este pozitiv, zero dacă X este zero și -1 dacă X este negativ.

3492 sgn	CALL 34E9,TEST-ZERO	Dacă X este zero, doar se revine cu zero ca 'ultima valoare'.
	RET C	
	PUSH DE	Se salvează indicatorul pentru STKEND.
	LD DE,+0001	Se stochează 1 în DE.
	INC HL	Se indică al doilea octet al lui X.
	RL (HL)	Se rotește bitul 7 în fanionul de transport.
	DEC HL	Se indică din nou destinația.
	SBC A,A	Se setează C pe zero dacă X este pozitiv și pe FF dacă X este negativ.
	LD C,A	Se stochează 1 sau -1, după cum este cazul.
	CALL 2D8E,INT-STORE	Se readuce indicatorul pentru STKEND.
	POP DE	Sfirsit.
	RET	

THE 'IN' FUNCTION (FUNCTIA 'IN')
(Deplasament 2C: 'in')

Această subrutină tratează funcția IN X. Aceasta intră la nivelul procesorului prin portul X încărcând BC cu X și executând instrucțiunea IN A,(C).

34A5 in	CALL 1E99,FIND-INT2	'Ultima valoare', X, este comprimată în BC.
	IN A,(C)	Semnalul este receptionat.
	JR 34B0,IN-PK-STK	Salt pentru stocarea rezultatului.

THE 'PEEK' FUNCTION (FUNCTIA 'PEEK')
(Deplasament 2B: 'peek')

Această subrutină tratează funcția PEEK X. 'Ultimă valoare' este scoasă din stivă apelând FIND-INT2 și este înlocuită cu valoarea conținutului locației cerute.

34AC peek	CALL 1E99,FIND-INT2	Se evaluează 'ultima valoare' rotunjită la cel mai apropiat întreg; se testează dacă se află în interval și se returnează în BC.
	LD A,(BC)	Se aduce octetul cerut.
34B0 IN-PK-STK	JP 2D28,STACK-A	Iesire prin salt la STACK-A.

THE 'USR' FUNCTION (FUNCTIA 'USR')
(Deplasament 2D: 'usr-no')

Această subrutină ('USR număr' ca formă distinctă de 'USR sir') tratează funcția USR X, unde X este un număr. Valoarea lui X este obținută în C, o adresă de revenire este stocată și codul mașină este executat din locația X.

3483	usr-no	CALL	1E99, FIND-INT2	Se evaluează 'ultima valoare' rotunjită la cel mai apropiat întreg; se testează dacă se află în interval și se returnează în BC.
		LD	HL, +2D2B	Se face ca adresa de revenire să fie cea a subrutinei STACK-BC.
		PUSH	HL	Se face un salt indirect la locația cerută.
		PUSH	BC	
		RET		

Notă: Este interesant faptul că registrul pereche IV este reinitializat când s-a făcut revenirea din STACK-BC, dar 'H'L', care conține indicatorul următorului literal, nu este restaurat și ar putea fi conturbat. Pentru o revenire corectă în BASIC, 'H'L' trebuie să iasă din codul mașină conținând adresa SCANNING din instrucțiunea 'end-calc', 2758 hex (10072 zecimal).

THE 'USR STRING' FUNCTION (FUNCTIA 'USR SIR')
(Deplasament 19: 'usr-\$')

Această subrutină tratează funcția USR X\$, unde X\$ este un sir. Subrutina revine cu adresa bitului șablon în BC pentru corespondența grafică definită de utilizator pentru X\$. Ea dă prezentarea A dacă X\$ nu este o singură literă între a și u sau un singur semn grafic definit de utilizator.

348C	usr-\$	CALL	2RF1, STK-FETCH	Se aduc parametrii sirului X\$.
		DEC	BC	Se decrementează lungimea cu 1 pentru a o testa.
		LD	A, B	Dacă lungimea a fost diferită de 1, atunci se sare pentru a da prezentarea A.
		OR	C	
		JR	NZ, 34E7, REPORT-A	Se aduce codul singular al sirului.
		LD	A, (DE)	Denotă acesta o literă?
		CALL	2C8D, ALPHA	Dacă da, salt pentru aducerea adresei sale.
		JR	C, 34D3, USR-RANGE	Se reduce intervalul pentru graficele actuale definite de utilizator la 0 - 20 zecimal.
		SUB	+90	Se dă prezentarea A dacă se iasă din interval.
		JR	C, 34E7, REPORT-A	Se testează din nou intervalul.
		CP	+15	Se dă prezentarea A dacă este afară din interval.
		JR	NC, 34E7, REPORT-A	Se face intervalul graficelor definite de utilizator 1 la 21 zecimal, ca pentru a la u.
		INC	A	Acum se face intervalul 0 la 20 zecimal în fiecare caz.
34D3	USR-RANGE	DEC	A	Se înmulțește cu 8 pentru a da un deplasament pentru adresă.
		ADD	A, A	Se testează intervalul pentru deplasament.
		ADD	A, A	Se dă prezentarea A dacă este în afara intervalului.
		ADD	A, A	Se aduce adresa primului primul grafic definit de utilizator în BC.
		CP	+A8	Se adună C la deplasament.
		JR	NC, 34E7, REPORT-A	Se stochează din nou rezultatul în C.
		LD	BC, (UD8)	Salt dacă nu este nici un transport.
		ADD	A, C	Se incrementează B pentru a completa adresa.
		LD	C, A	Salt pentru a stoca adresa.
		JR	NC, 34E4, USR-STACK	
		INC	B	
		JP	2D2B, STACK-BC	

Prezentarea A - Argument invalid

34E7	REPORT-A	RST	0008, ERROR-1	Se apelează rutina de tratare a erorii.
		DEFB	+09	

THE 'TEST-ZERO' SUBROUTINE (SUBRUTINA 'TEST ZERO')

Această subrutină este apelată de cel puțin nouă ori pentru a testa dacă un număr în virgulă mobilă este zero. Acest test necesită ca primii patru octeți ai numărului să fie fiecare zero. Subrutina revine cu fanionul de transport setat dacă numărul a fost de fapt zero.

34E9 TEST-ZERO	PUSH HL	Se salvează HL în stivă.
	PUSH BC	Se salvează BC în stivă.
	LD B,A	Se salvează valoarea din A în B.
	LD A,(HL)	Se aduce primul octet.
	INC HL	Se indică al doilea octet.
	OR (HL)	Se face SAU primul octet cu al doilea.
	INC HL	Se indică al treilea octet.
	OR (HL)	Se face SAU rezultatul cu al treilea octet.
	INC HL	Se indică al patrulea octet.
	OR (HL)	Se face SAU rezultatul cu al patrulea octet.
	LD A,B	Se readuce valoarea inițială a lui A.
	POP BC	Si a lui BC.
	POP HL	Se readuce indicatorul numărului în HL.
	RET NZ	Se revine cu transportul resetat dacă oricare din cei patru octeți a fost diferit de zero.
	SCF	Se setează fanionul de transport pentru a indica faptul că acel număr a fost zero, și se revine.
	RET	

THE 'GREATER THAN ZERO' OPERATION (OPERATI 'MAI MARE CA ZERO')
(Deplasament 37: 'greater-0')

Această subrutină redă o 'ultimă valoare' unu dacă prezenta 'ultimă valoare' este mai mare decât zero și da zero în caz contrar. Ea este de asemenea folosită de către alte subrutine pentru a executa 'salt la plus'.

34F9 GREATER-0	CALL 34E9,TEST-ZERO	Este 'ultima valoare' zero?
	RET C	Dacă da, revenire.
	LD A,+FF	Salt mai departe la LESS THAN ZERO ???
	JR 3507,16N-TO-C	

THE 'NOT' FUNCTION (FUNCTIA 'NOT')
(Deplasament 30: 'not')

Această subrutină redă o 'ultimă valoare' unu dacă prezenta 'ultimă valoare' este zero și zero în caz contrar. Ea este de asemenea folosită de alte subrutine pentru a executa 'salt la zero'.

3501 NOT	CALL 34E9,TEST-ZERO	Fanionul de transport va fi setat numai dacă 'ultima valoare' este zero; aceasta da rezultatul corect.
	JR 350B,FP-0/1	Salt înainte.

THE 'LESS THAN ZERO' OPERATION (OPERATIA 'MAI MIC CA ZERO')
(Deplasament 36: 'less-0')

Această subrutină redă o 'ultimă valoare' unu dacă prezenta 'ultimă valoare' este mai mică decât zero și zero în caz contrar. Ea este de asemenea folosită și de alte subrutine pentru a executa 'salt la minus'.

3506 less-0	XOR A	Se șterge registrul A.
3507 SIGN-TO-C	INC HL	Se indică octetul de semn.
	XOR (HL)	Transportul este resetat pentru un număr pozitiv și este setat pentru un număr negativ; când se intră din GREATER-0 semnul opus trece în transport.
	DEC HL	
	RLCA	

THE 'ZERO OR ONE' SUBROUTINE (SUBRUTINA 'ZERO SAU UNU')

Această subrutină setează 'ultima valoare' pe zero dacă fanionul de transport este resetat și o setează pe unu dacă fanionul de transport este setat. Când

este apelată din 'E-TO-FP' oricum creează zero sau unu, nu în stivă ci în mem-
o.

350B FP-0/1	PUSH	HL	Se salvează indicatorul rezultatului.
	LD	A,+00	Se șterge A fără a afecta transportul.
	LD	(HL),A	Se setează primul octet pe zero.
	INC	HL	Se indică al doilea octet.
	LD	(HL),A	Se setează al doilea octet pe zero.
	INC	HL	Se indică al treilea octet.
	RLA		Se rotește transportul în A, se face A unu dacă transportul a fost setat și zero dacă transportul a fost resetat.
	LD	(HL),A	Se setează al treilea octet pe unu sau pe zero.
	RRA		Asigurare că A este zero din nou.
	INC	HL	Se indică al patrulea octet.
	LD	(HL),A	Se setează al patrulea octet pe zero.
	INC	HL	Se indică al cincilea octet.
	LD	(HL),A	Se setează al cincilea octet pe zero.
	POP	HL	Se readuce indicatorul rezultat.
	RET		Sfârșit.

THE 'OR' OPERATION (OPERATIA 'OR')

(Deplasament 07: 'or')

Această subrutină execută operația binară 'X OR Y' ('X SAU Y') și redă X dacă Y este zero și valoarea zero în caz contrar.

351B or	EX	DE,HL	HL indică Y, cel de al doilea număr.
	CALL	34E9,TEST-ZERO	Se testează dacă Y este zero.
	EX	DE,HL	Se refac indicatorii.
	RET	C	Se revine dacă Y a fost zero; acum X este 'ultima valoare'.
	SCF		Se setează fanionul de transport și salt înapoi pentru a seta 'ultima valoare' pe 1.
	JR	350B,FP-0/1	

THE 'NUMBER AND NUMBER' OPERATION (OPERATIA 'NUMAR SI NUMAR')

(Deplasament 08: 'no-&-no')

Această subrutină execută operația binară 'X AND Y' ('X SI Y') și redă X dacă Y este diferit de zero și valoarea zero în caz contrar.

3524 no-&-no	EX	DE,HL	HL îl indică pe Y, DE pe X.
	CALL	34E9,TEST-ZERO	Se testează dacă Y este zero.
	EX	DE,HL	Se reschimbă indicatorii.
	RET	NC	Se revine cu X ca 'ultima valoare' dacă Y a fost diferit de zero.
	AND	A	Se resetează fanionul de transport și salt înapoi pentru a seta 'ultima valoare' pe zero.
	JR	350B,FP-0/1	

THE 'STRING AND NUMBER' OPERATION (OPERATIA 'SIR SI NUMAR')

(Deplasament 10: 'str-&-no')

Această subrutină execută operația binară 'X\$ AND Y' ('X\$ SI Y') și redă X\$ dacă Y este diferit de zero și un sir nul în caz contrar.

352B str-&-no	EX	DE,HL	HL îl indică pe Y, DE pe X\$.
	CALL	34E9,TEST-ZERO	Se testează dacă Y este zero.
	EX	DE,HL	Se schimbă indicatorii la loc.
	RET	NC	Se revine cu X\$ ca 'ultima valoare' dacă a fost diferit de zero.
	PUSH	DE	Se salvează indicatorul

DEC	DE	numărului. Se indică al cincilea octet al parametrului sir, adică octetul cel mai semnificativ al lungimii.
XOR	A	Se șterge registrul A.
LD	(DE),A	Octetul cel mai semnificativ al lungimii este acum setat pe zero.
DEC	DE	Se indică octetul cel mai puțin semnificativ al lungimii.
LD	(DE),A	Acum octetul cel mai puțin semnificativ al lungimii este setat pe zero.
POP	DE	Se readuce indicatorul.
RET		Revenire cu parametrul sir ca fiind 'ultima valoare'.

THE 'COMPARISON' OPERATIONS (OPERATIA 'COMPARARE')

(Deplasamente 09 la 0E & 11 la 16: 'no-l-eql', 'no-gr-eq', 'nos-neql', 'no-grtr', 'no-less', 'nos-eql', 'str-l-eql', 'str-gr-eq', 'strs-neql', 'str-grtr', 'str-less' & 'strs-eql')

Această subrutină este folosită pentru a executa 12 operații de comparare posibile. Deplasamentul operației singulare este prezent în registrul B la începutul acestei subrutine.

353B no-l-eql etc.	LD A,B	Deplasamentul singular trece în registrul A.
	SUB +08	Intervalul este acum 01-06 & 09-0E.
	BIT 2,A	Acest interval este schimbat
	JR NZ,3543,EX-OR-NOT	cu: 00-02, 04-06, 08-0A & 0C-0E.
	DEC A	
3543 EX-OR-NOT	RRCA	Apoi este redus la 00-07 cu transportul setat pentru 'mai mare sau egal' & 'mai mic'; apoi operațiile cu transportul setat sînt tratate ca și operațiile de complementare față de unu ce schimbă valoarea.
	JR NC,354E,NU-OR-STR	
	PUSH AF	
	PUSH HL	
	CALL 343C,EXCHANGE	
	POP DE	
	EX DE,HL	
	POP AF	
354E,NU-OR-STR	BIT 2,A	Comparatiile numerice sînt acum separate de partiile de siruri, testînd bitul 2.
	JR NZ,3559,STRINGS	Operațiile numerice au acum intervalul 00-01 cu transportul setat pentru 'egal' și 'diferit'.
	RRCA	Se salvează deplasamentul. Numerele sînt scăzute pentru testul final.
	PUSH AF	Comparatiile de siruri au acum intervalul 02-03 cu transportul setat pentru 'egal' și 'diferit'.
	CALL 300F,SUBTRACT	Salvare deplasament.
	JR 358C,END-TESTS	Se aduc acum din stiva calculatorului lungimile și adresele de bază ale sirurilor.
3559 STRINGS	RRCA	Lungimea celui de al doilea sir.
	PUSH AF	
	CALL 2BF1,STK-FETCH	
	PUSH DE	
	PUSH BC	
	CALL 2BF1,STK-FETCH	
	POP HL	
3564 BYTE-COMP	LD A,H	
	OR L	
	EX (SP),HL	
	LD A,B	
	JR NZ,3575,SEC-PLUS	Salt numai dacă al doilea sir este nul.
	OR C	Aici al doilea sir este fie nul fie mai mic decît primul.
356B SECND-LOW	POP BC	
	JR Z,3572,BOTH-NULL	
	POP AF	
	CCF	
	JR 3588,STR-TEST	Transportul este complementat pentru a da corect testul

3572 BOTH-NULL	POP AF	rezultatului.
	JR 3588,STR-TEST	Aici transportul este
3575 SEC-PLUS	OR C	utilizat cum s-a mentinut.
	JR Z,3585,FRST-LESS	Acum primul sir este nul, al
	LD A,(DE)	doilea sir nu este nul.
	SUB (HL)	Nici un sir nu este nul, asa
		ca se compară următorii lor
	JR C,3585,FRST-LESS	octeti.
	JR NZ356B,SECND-LOW	Primul octet este mai mic.
	DEC BC	Al doilea octet este mai mic.
	INC DE	Octetii sînt egali; asa ca
	INC HL	lungimile sînt decrementate
	EX (SP),HL	si se face un salt la BYTE-
	DEC HL	COMP pentru a compara
		următorii octeti ai
	JR 3564,BYTE-COMP	sirurilor reduse.
3585 FRST-LESS	POP BC	
	POP AF	Aici transportul este sters
	AND A	pentru un test corect
		rezultat.
3588 STR-TEST	PUSH AF	Pentru testele sirului, se
	RST 0028,FP-CALC	pune un zero în stiva
		calculatorului.
	DEFB +A0,stk-zero	
	DEFB +38,end-calc	
358C END-TESTS	POP AF	Aceste trei teste, apelate
	PUSH AF	după cum este necesar, dau
	CALL C,3501,NOT	rezultatele corecte pentru
	POP AF	toate cele 12 comparatii.
	PUSH AF	Transportul initial este
	CALL NC,34F9,GREATER-0	setat pentru 'diferit' si
	POP AF	'egal', iar transportul final
		este pentru 'mai mare', 'mai
		mic' si 'egal'.
	RRCA	
	CALL NC,3501,NOT	
	RET	Sfîrsit.

THE 'STRING CONCATENATION' OPERATION (OPERATIA 'CONCATENARE SIR')
(Deplasament 17: 'strs-add')

Această subrutină execută operația binară 'A\$+B\$'. Parametrii acestor siruri sînt adusi si este găsită lungimea totală. Se face accesibil suficient spațiu în spațiul de lucru pentru a contine ambele siruri si se copiază sirurile. Rezultatul acestei subrutine este de aceea să producă o variabilă temporară A\$+B\$ care este rezidentă în spațiul de lucru.

359C strs-add	CALL 2BF1,STK-FETCH	Se aduc si se salvează
	PUSH DE	parametrii celui de al doilea
		sir.
	PUSH BC	
	CALL 2BF1,STK-FETCH	Se aduc parametrii primului
		sir.
	POP HL	
	PUSH HL	Acum lungimile sînt în H1 si
		în DE.
	PUSH DE	Se salvează parametrii
	PUSH BC	primului sir.
	ADD HL,BC	Se calculează lungimea totală
	LD B,H	a celor două siruri si se
	LD C,L	trece în BC.
	RST 0030,BC-SPACES	Se face accesibil suficient
		spațiu.
	CALL 2AB2,STK-STORE	Parametrii noului sir sînt
		trecuti în stiva
		calculatorului.
	POP BC	Parametrii primului sir sînt
	POP HL	regăsiți si sirul este copiat
	LD A,B	în spațiu de lucru atîta timp
	OR C	cît nu este un sir nul.
	JR Z,35B7,OTHER-STR	
	LDIR	
35B7 OTHER-STR	POP BC	Exact aceeasi procedură este
	POP HL	urmată de al doilea sir
	LD A,B	rezultînd astfel 'A\$+B\$'.
	OR C	

JR Z,35BF,STK-PNTRS
LDIR

THE 'STK-PNTRS' SUBROUTINE (SUBRUTINA 'STK-PNTRS')

Această subrutină resetează registrul pereche HL astfel încît să indice primul octet al 'ultimei valori', adică STKEND-5, iar registrul pereche DE să indice cu unu după 'ultima valoare', adică STKEND.

35BF STK-PNTRS	LD	HL,(STKEND)	Se aduce valoarea curentă a lui STKEND.
	LD	DE,+FFFB	Se setează DE pe -5, completînd fată de doi.
	PUSH	HL	Se stochează valoarea lui STKEND.
	ADD	HL,DE	Se calculează STKEND-5.
	POP	DE	Acum DE conține STKEND și HL conține STKEND-5.
	RET		

THE 'CHR\$' FUNCTION (FUNCTIA 'CHR\$')

(Deplasament 2F: 'chr\$')

Această subrutină tratează funcția CHR\$ X și creează un sir caracter singular în spațiul de lucru.

35C9 chr\$	CALL	2DD5,FP-TO-A	'Ultima valoare' este comprimată în registrul A.
	JR	C,35DC,REPORT-B	Se prezintă eroare dacă X a fost mai mare decît 255 zecimal, sau X a fost un număr negativ.
	JR	NZ,35DC,REPORT-B	Se salvează valoarea comprimată a lui X.
	PUSH	AF	Se face accesibil un spațiu în spațiul de lucru.
	LD	BC,+0001	Se aduce valoarea.
	RST	0030,BC-SPACES	Se copiază valoarea în spațiul de lucru.
	POP	AF	Se trec parametrii noului sir în stiva calculatorului.
	LD	(DE),A	Se resetează indicatorii.
	CALL	2AB2,STK-STORE	Sfîrsit.
	EX	DE,HL	
	RET		

Prezentarea B - Intreg afară din interval.

35DC REPORT-B	RST	0008,ERROR-1	Se apelează rutina de tratare eroare.
	DEFB	+0A	

THE 'VAL' AND 'VAL\$' FUNCTION (FUNCTIA 'VAL' SI 'VAL\$')

(Deplasamente 1D: 'val' și 1B: 'val\$')

Această subrutină tratează funcțiile VAL X\$ și VAL\$ X\$. Cînd tratează VAL X\$, ea revine cu o 'ultimă valoare' care este rezultatul evaluării sirului (fără cotele limită) ca o expresie numerică. Cînd tratează VAL\$ X\$, ea evaluează X\$ (fără cotele limită) ca o expresie sir, și returnează parametrii acestei expresii sir ca o 'ultimă valoare' în stiva calculatorului.

35DE val (also val\$)	LD	HL,(CH-ADD)	Valoarea curentă a lui CH-ADD este păstrată în stiva masinii.
	PUSH	HL	'Deplasamentul' lui 'val' sau a lui 'val\$' trebuie să fie în registrul B; acum el este copiat în A.
	LD	A,B	Se produce +00 și transport setat pentru 'val', +FB și transport resetat pentru 'val\$'.
	ADD	A,+E3	Este produs +FF (bitul 6 este astfel setat) pentru 'val', și +00 (bitul 6 este resetat) pentru 'val\$'.
	SBC	A,A	Se salvează 'fanionul' în stiva masinii.
	PUSH	AF	Se aduc parametrii sirului; salvare adresa de început; se adună un octet la lungime și se face accesibil spațiu pentru sir (+1) în spațiul de lucru.
	CALL	2RF1,STK-FETCH	
	PUSH	DE	
	INC	BC	
	RST	0030,BC-SPACES	

	POP	HL	Adresa de început a sirului trece în HL ca adresă sursă.
	LD	(CH-ADD),DE	Indicatorul primului spațiu nou trece în CH-ADD și în stiva masinii.
	PUSH	DE	
	LDIR		Sirul este copiat în spațiul de lucru, împreună cu un octet adițional.
	EX	DE,HL	Se schimbă indicatorii.
	DEC	HL	Octetul adițional este înlocuit cu un caracter 'carriage return'.
	LD	(HL),+0D	Fanionul de sintaxă este resetat și sirul este baleiat pentru sintaxă corectă.
	RES	7,(FLAGS)	Se aduce caracterul de după sir.
	CALL	24FB,SCANNING	Se testează dacă s-a ajuns la sfârșitul expresiei.
	RST	0018,GET-CHAR	Dacă nu, se prezintă eroare.
	CP	+0D	Se aduce adresa de început a sirului.
	JR	NZ,360C,V-RPORT-C	
	POP	HL	Este adus 'fanionul' pentru 'val/val#' și bitul 6 este comparat cu bitul 6 al rezultatului baleierii sintaxei.
	POP	AF	Se prezintă eroare dacă cei doi biti nu coincid.
	XOR	(FLAGS)	Din nou adresa de început în CH-ADD.
	AND	+40	Fanionul este setat pentru execuție linie.
360C V-RPORT-C	JP	NZ,1C8A,REPORT-C	Sirul este tratat ca o 'expresie următoare' și este produsă o 'ultimă valoare'.
	LD	(CH-ADD),HL	Se reduce valoarea inițială a lui CH-ADD.
	SET	7,(FLAGS)	Subrutina iese prin STK-PNTRS care resetează indicatorii.
	CALL	24FB,SCANNING	
	POP	HL	
	LD	(CH-ADD),HL	
	JR	358F,STK-PNTRS	

THE 'STR#' FUNCTION (FUNCTIA 'STR#')
(Deplasament 2E: 'str#')

Această subrutină tratează funcția STR# X și returnează o 'ultimă valoare' care este un set de parametrii care definesc un sir continut care ar apărea pe ecran dacă X ar fi afișat de o comandă PRINT.

361F str#	LD	BC,+0001	Se face un spațiu în spațiul de lucru și adresa lui este copiată în K-CUR, adresa cursorului.
	RST	0030,BC-SPACES	
	LD	(K-CUR),HL	Această adresă este salvată de asemenea în stivă.
	PUSH	HL	Adresa canalului curent este salvată în stiva masinii.
	LD	HL,(CURCHL)	Este deschis canalul 'R',
	PUSH	HL	permițând 'tipărirea' sirului
	LD	A,+FF	afară din spațiul de lucru.
	CALL	1601,CHAN-OPEN	'Ultima valoare', X, este acum tipărită în afara
	CALL	2DE3,PRINT-PP	spațiului de lucru și spațiul de lucru este extins cu
			fiecare caracter.
	POP	HL	Se reduce CURCHL în HL și se
	CALL	1615,CHAN-FLAG	readuc fanioanele corespunzătoare.
	POP	DE	Se reduce adresa de început a sirului.
	LD	HL,(K-CUR)	Acum adresa cursorului este
	AND	A	una după sfârșitul sirului și
	ABC	HL,DE	deci diferența este lungimea.
	LD	B,H	Se transferă lungimea în BC.
	LD	C,L	
	CALL	2A82,STK-STO-#	Se trec în stiva calculatorului
			parametrii noului sir.
	EX	DE,HL	Se resetează indicatorii.
	RET		Sfârșit.

Notă: Vezi PRINT-FP pentru lămurirea erorii 'PRINT "A"+STR\$ 0.1'.

THE 'READ-IN' SUBROUTINE (SUBRUTINA 'READ-IN')
(Deplasament 1A: 'read-in')

Subrutina este apelată prin intermediul deplasamentului calculatorului în cadrul primei linii a rutinei S-INKEY\$ în SCANNING. Aceasta vine să furnizeze informațiile citite de-a lungul diferitelor siruri accesibile în standardul Spectrum. Ca și INKEY\$, subrutina revine cu un sir.

3645 read-in	CALL 1E94	Parametrul numeric este comprimat în registrul A.
	CP +10	Este mai mic decât 16 zecimal?
	JP NC,1E9F,REPORT-B	Dacă nu, se prezintă eroare.
	LD HL,(CURCHL)	Adresa canalului curent este salvată în stiva masinii.
	PUSH HL	Se deschide canalul care este specificat de către parametru.
	CALL 1601,CHAN-OPEN	Acum semnalul este acceptat, ca o 'valoare-tastă'.
	CALL 15E6,INPUT-AD	Lungimea absentă a sirului rezultat este zero.
	LD BC,+0000	Salt dacă nu este nici un semnal.
	JR NC,365F,R-I-STORE	Acum se setează lungimea pe 1.
	INC C	Se face un spațiu în spațiul de lucru.
	RST 0030,BC-SPACES	Se pune sirul în acest spațiu
365F R-I-STORE	LD (DE),A	Se trec parametrii sirului în stiva calculatorului.
	CALL 2AB2,STK-STD-\$	Se readuce CURCHL și fanioanele corespunzătoare.
	POP HL	Iesire, setînd indicatorii.
	CALL 1615,CHAN-FLAG	
	JP 35BF,STK-PNTRS	

THE 'CODE' FUNCTION (FUNCTIA 'CODE')
(Deplasament 1C: 'code')

Această subrutină tratează funcția CODE A\$ și revine codul Spectrum al primului caracter în A\$, sau zero dacă A\$ trebuie să fie nul.

3669 code	CALL 2BF1,STK-FETCH	Se aduc parametrii sirului.
	LD A,B	Este testată lungimea și registrul A conținînd zero
	OR C	este transportat mai departe ca și cînd A\$ este un sir nul.
	JR Z,3671,STK-CODE	Altfel codul primului caracter este pus în A.
	LD A,(DE)	Această subrutină iese prin STACK-A care dă 'ultima valoare' corectă.
3671 STK-CODE	JP 2D28,STACK-A	

THE 'LEN' FUNCTION (FUNCTIA 'LEN')
(Deplasament 1E: 'len')

Această subrutină tratează funcția LEN A\$ și returnează o 'ultimă valoare' care este egală cu lungimea sirului.

3674 len	CALL 2BF1,STK-FETCH	Sînt aduși parametrii sirului
	JP 2D28,STACK-B	Subrutina iese prin STACK-BC, care dă 'ultima valoare' corectă.

THE 'DECREASE THE COUNTER' SUBROUTINE (SUBRUTINA 'DECREMENTARE CONTOR')
(Deplasament 35: 'dec-jr-nz')

Această subrutină este apelată doar de subrutina SERIES GENERATOR și de fapt este o operație 'DJNZ' dar contorul este variabila sistem BREG, mai degrabă decât registrul B.

367A dec-jr-nz	EXX	Se trece la setul registrilor auxiliari și se salvează indicatorul următorului literal în stiva masinii.
	PUSH HL	HL va indica BREG.
	LD HL,+5C67	

DEC	(HL)	Se decrementează BREG.
POP	HL	Se readuce indicatorul următorului literal.
JR	NZ,3687,JUMP-2	Se execută salt dacă este diferit de zero.
INC	HL	Se trece peste următorul literal.
EXX		Se revine la setul principal de registrii.
RET		Sfîrsit.

THE 'JUMP' SUBROUTINE (SUBROUTINA 'JUMP' ('SALT'))
(Deplasament 33: 'jump')

Subrutina execută un salt necondiționat cînd este apelată prin literalul '33'. De asemenea ea mai este folosită de către subrutinele DECREASE THE COUNTER și JUMP ON TRUE.

3686 JUMP	EXX		Se trece în setul registrilor auxiliari.
3687 JUMP-2	LD	E,(HL)	Următorul literal (salt lungime) este pus în registreul E.
	LD	A,E	Numărul 00 hex sau FF hex este format în A în concordanță cu faptul dacă E este pozitiv sau negativ, și apoi este copiat în D.
	RLA		
	SBC	A,A	
	LD	D,A	Acum registrii H' & L' contin indicatorul următorului literal.
	ADD	HL,DE	
	EXX		Sfîrsit.
	RET		

THE 'JUMP ON TRUE' SUBROUTINE (SUBROUTINA 'JUMP ON TRUE' ('SALT CONDITIONAT'))
(Deplasament 00: 'jump-true')

Această subrutină execută un salt condiționat dacă 'ultima valoare' din stiva calculatorului, sau mai exact numărul curent adresat de registrul pereche DE, este adevărat.

368F jump-true	INC	DE	Se indică al treilea octet, care este zero sau unu.
	INC	DE	
	LD	A,(DE)	Se colectează acest octet în registrul A.
	DEC	DE	Se indică încă o dată primul octet.
	DEC	DE	
	AND	A	Se testează al treilea octet: este zero?
	JR	NZ,3686,JUMP	Se execută saltul dacă octetul este diferit de zero, adică numărul nu este fals.
	EXX		Se trece la setul registrilor auxiliari.
	INC	HL	Se trece peste lungime salt.
	EXX		Se revine la setul registrilor principali.
	RET		Sfîrsit.

THE 'END-CALC' SUBROUTINE (SUBROUTINA 'END-CALC')
(Deplasament 38: 'end-calc')

Această subrutină termină o operație RST 0028.

369B end-calc	POP	AF	Este înlăturată adresa de revenire în calculator ('RE-ENTRY').
	EXX		In schimb, adresa din H'L' este pusă în stiva mașinii și se face un salt indirect la ea. H'L' va conține acum orice adresă anterioară în lanțul de adrese al calculatorului.
	EX	(SP),HL	
	EXX		
	RET		Sfîrsit.

THE 'MODULUS' SUBROUTINE (SUBROUTINA 'MODUL')
(Deplasament 32: 'n-mod-m')

Această subrutină calculează $N \pmod{M}$, unde M este un întreg pozitiv continuu în vârful stivei calculatorului, 'ultima valoare', iar N este un întreg continuu în stivă dedesubt de M .

Subrutina revine cu partea întreagă a cîtului $\text{INT}(N/M)$ în vârful stivei calculatorului, 'ultima valoare', iar restul $N - \text{INT}(N/M)$ în locul al doilea din stivă.

Această subrutină este apelată de-a lungul calculului unui număr oarecare, pentru a reduce $N \pmod{65537}$ zecimal.

36A0 n-mod-m	RST	0028,FP-CALC	N,M
	DEFB	+C0,st-mem-0	N,M mem-0 contine M
	DEFB	+02,delete	N
	DEFB	+31,duplicate	N,N
	DEFB	+E0,get-mem-0	N,N,M
	DEFB	+05,division	N,N/M
	DEFB	+27,int	N,INT(N/M)
	DEFB	+E0,get-mem-0	N,INT(N/M),M
	DEFB	+01,exchange	N,M,INT(N/M)
	DEFB	+C0,st-mem-0	N,M,INT(N/M) mem-0 contine
			INT(N/M)
	DEFB	+04,multiply	N,M*INT(N/M)
	DEFB	+03,subtract	n-M*INT(N/M)
	DEFB	+E0,get-mem-0	n-M*INT(N/M),INT(N/M)
	DEFB	+38,end-calc	
	RET		Sfîrsit.

THE 'INT' FUNCTION (FUNCTIA 'INT')
(Deplasament 27: 'int')

Această subrutină tratează funcția $\text{INT } X$ și redă o 'ultimă valoare' care este 'partea întreagă' a valorii înlocuite. Astfel $\text{INT } 2.4$ dă 2 dar cum subrutina întotdeauna rotunjește rezultatul în jos, $\text{INT } -2.4$ va da -3.

Subrutina folosește subrutina INTEGER TRUNCATION TOWARDS ZERO de la 3214 pentru a produce $I(X)$ astfel încît $I(2.4)$ dă 2 și $I(-2.4)$ dă -3. Astfel, $\text{INT } X$ este dat de $I(X)$ pentru valorile lui X care sînt mai mari sau egale cu zero, și $I(X)-1$ pentru valorile neegative ale lui X care încă nu sînt întregi, cînd rezultatul este, bineînțeles, $I(X)$.

36AF int	RST	0028,FP-CALC	X
	DEFB	+31,duplicate	X,X
	DEFB	+36,less-0	X,(1/0)
	DEFB	+00,jump-true	X
	DEFB	+04,to 36B7,X-NEG	X

Pentru valori ale lui X care s-au arătat a fi mai mari sau egale cu zero nu se face nici un salt și $I(X)$ este gata găsit.

DEFB	+3A,truncate	I(X)
DEFB	+38,end-calc	
RET		Sfîrsit.

Cînd X este un întreg negativ $I(X)$ este returnat, altfel se returnează $I(X)-1$.

36B7 X-NEG	DEFB	+31,duplicate	X,X
	DEFB	+3A,truncate	X,I(X)
	DEFB	+C0,st-mem-0	X,I(X) mem-0 contine I(X)
	DEFB	+03,subtract	X-I(X)
	DEFB	+E0,get-mem-0	X-I(X),I(X)
	DEFB	+01,exchange	I(X),X-I(X)
	DEFB	+30,not	I(X),(1/0)
	DEFB	+00,jump-true	I(X)
	DEFB	+03,to 36C2,EXIT	I(X)

Saltul este executat pentru valori ale lui X care sînt întregi negativi, altfel nu se face nici un salt și se calculează $I(X)-1$.

DEFB	+A1,stk-one	I(X),1
DEFB	+03,subtract	I(X)-1

În fiecare caz subrutina se termină cu:

36C2 EXIT	DEFB	+38,end-calc	I(X) sau I(X)-1
	RET		

THE 'EXPONENTIAL' FUNCTION (FUNCTIA 'EXPONENTIAL')
(Deplasament 26: 'exp')

Această subrutină tratează funcția $\text{EXP } X$ și este prima dintre cele patru

rutine care foloseste SERIES GENERATOR pentru a produce polinoame Cebîsev.
Aproximarea pentru EXP X se găseste în felul următor:

i. X este împărțit cu LN 2 pentru a da Y, așa că 2 la puterea Y este acum rezultatul cerut.

ii. Se găseste valoarea N, astfel încît $N = \text{INT } Y$.

iii. Se găseste valoarea lui W, astfel încît $W = Y - N$, unde $0 \leq W < 1$, așa cum este necesar pentru ca seriile să fie convergente.

iv. Este format argumentul Z, astfel încît $Z = 2 * W - 1$.

v. SERIES GENERATOR este folosit pentru a reda $2 ** W$.

vi. În final N este adunat la exponent, rezultînd $2 ** (N + W)$, care este $2 ** Y$ și de aceea răspunsul cerut pentru EXP X.

Metoda folosită este ilustrată folosind un program BASIC în Anexă.

36C4 EXP RST 0028,FP-CALC X

Se execută pasul i.

DEFB	+3D, re-stack	X (în forma în virgulă mobilă desfășurată)
DEFB	+34, stk-data	X, 1/LN 2
DEFB	+F1, exponent+81	
DEFB	+39, +AA, +3B, +29	
DEFB	+04, multiply	X/LN 2 = Y

Se execută pasul ii.

DEFB	+31, duplicate	Y, Y
DEFB	+27, int, 1C46	Y, INT Y = N
DEFB	+C3, st-mem-3	Y, N mem-3 contine N.

Se execută pasul iii.

DEFB	+03, subtract	Y - N = W
------	---------------	-----------

Se execută pasul iv.

DEFB	+31, duplicate	W, W
DEFB	+0F, addition	2 * W
DEFB	+A1, stk-one	2 * W, 1
DEFB	+03, subtract	2 * W - 1 = Z

Se execută pasul v, trecînd la SERIES GENERATOR parametrul '8' și cele opt constante cerute.

	DEFB	+88, series-08	Z
1.	DEFB	+13, exponent+63	
	DEFB	+36, (+00, 00, +00)	
2.	DEFB	+58, exponent+68	
	DEFB	+65, +66, (+00, +00)	
3.	DEFB	+9D, exponent+6D	
	DEFB	+78, +65, +40, (+00)	
4.	DEFB	+A2, exponent+72	
	DEFB	+60, +32, +C9, (+00)	
5.	DEFB	+E7, exponent+77	
	DEFB	+21, +F7, +AF, +24	
6.	DEFB	+EB, exponent+7B	
	DEFB	+2F, +30, +30, +14	
7.	DEFB	+EE, exponent+7E	
	DEFB	+7E, +BB, +94, +58	
8.	DEFB	+F1, exponent+81	
	DEFB	+3A, +7E, +F8, +CF	

La sfîrșitul ultimei bucle 'ultima valoare' este $2 ** W$.

Se execută pasul vi.

DEFB	+E3, get-mem-3	$2 ** W, N$
DEFB	+38, end-calc	
CALL	2DD5, FP-TO-A	Valoarea absolută a lui N mod 256 zecimal este pusă în registru A.
JR	NZ, 3705, N-NEGTV	Sălt înainte dacă N a fost

JR	C,3703,REPORT-6	negativ.
ADD	A,(HL)	Este eroare dacă ABS N este mai mare decât 255 zecimal.
JR	NC,370C,RESULT-OK	Acum se adună ABS N la exponent.
		Salt numai dacă e este mai mare decât 255 zecimal.

Prezentarea 6 - Număr prea mare

3703 REPORT-6	RST 0008,ERROR-1 DEFB +05	Se apelează rutina de tratare eroare.
3705 N-NEGTV	JR C,370E,RSLT-ZERO SUB (HL)	Rezultatul trebuie să fie zero dacă N este mai mic decât -255 zecimal.
	JR NC,370E,RSLT-ZERO	Se scade ABS N din exponent ca și cum N ar fi fost negativ.
370C RESULT-OK	NEG LD (HL),A RET	Rezultat zero dacă e este mai mic decât zero.
370E RSLT-ZERO	RST 0028,FP-CALC DEFB +02,delete DEFB +A0,stk-zero DEFB +38,end-calc RET	Minus e este schimbat în e. Se introduce exponentul, e. Sfîrsit: 'ultima valoare' este EXP X. Se folosește calculatorul pentru a face 'ultima valoare' zero.
		Sfîrsit, cu EXP X = 0.

THE 'NATURAL LOGARITHM' FUNCTION (FUNCTIA 'LOGARITM NATURAL')
(Deplasament 25: 'ln')

Această subrutină tratează funcția LN X și este a doua din cele patru rutine care utilizează SERIES GENERATOR pentru a produce polinoamele Cebisev. Aproximarea lui LN X se găsește în modul următor:

- i. Se testează X și se dă prezentarea A dacă X nu este pozitiv.
- ii. Apoi X este despărțit în exponentul său adevărat, e', și în mantisa sa X' = X/(2**e'), unde X' este mai mare sau egal cu 1.
- iii. Se formează valoarea cerută Y1 sau Y2. Dacă X' este mai mare decât 0.8 atunci Y1=e'*LN 2 și în caz contrar Y2=(e'-1)*LN 2.
- iv. Dacă X' este mai mare decât 0.8 atunci cantitatea X'-1 este stocată; altfel se stochează 2*X'-1.
- v. Acum este format argumentul Z, care, dacă X' este mai mare decât 0.8, este Z=2.5*X'-3; altfel Z=5*X'-3. În fiecare caz, -1<=Z<=1, așa cum este necesar pentru ca seriile să fie convergente.
- vi. SERIES GENERATOR este folosit pentru a produce funcția cerută.
- vii. În final o simplă înmulțire și o adunare conduc la returnarea lui LN X ca 'ultima valoare'.

3713 ln	RST 0028,FP-CALC	X
---------	------------------	---

Se execută pasul i.

DEFB +3D,re-stack	X (în forma în virgulă mobilă desfășurată)
DEFB +31,duplicate	X,X
DEFB +37,greater-0	X,(1/0)
DEFB +00,jump-true	X
DEFB +04,to 371C,VALID	X
DEFB ++38,end-calc	X

Prezentarea A - Argument invalid

371A REPORT-A	RST 0008,ERROR-1 DEFB +09	Se apelează rutina de tratare eroare.
---------------	------------------------------	---------------------------------------

Se execută pasul ii.

371C VALID	DEFB +A0,stk-zero	X,0	1 sters este
	DEFB +02,delete	X	suprainprimat cu zero.
	DEFB +38,end-calc	X	
	LD A,(HL)	Exponentul, e, trece în A.	
	LD (HL,+80)	X este redus la X'.	
	CALL 2D28,STACK-A	Stiva contine: X',e.	
	RST 0028,FP-CALC	X',e	
	DEFB +34,stk-data	X',e,128 (zecimal)	
	DEFB +38,exponent+88		
	DEFB +00,(+00,+00,+00)		
	DEFB +03,subtract	X',e'	

Se execută pasul iii.

	DEFB ++01,exchange	e',X'	
	DEFB +31,duplicate	e',X',X'	
	DEFB +34,stk-data	e',X',X',0.8 (zecimal)	
	DEFB +F0,exponent+80		
	DEFB +4C,+CC,+CC,+CD		
	DEFB +03,subtract	e',X',X'-0.8	
	DEFB +37,greater-0	e',X',(1/0)	
	DEFB +00,jump-true	e',X'	
	DEFB +08,to 373D,GRE.8	e',X'	
	DEFB +01,exchange	X',e'	
	DEFB +A1,stk-one	X',e',1	
	DEFB +03,subtract	X',e',-1	
	DEFB +01,exchange	e'-1,X'	
	DEFB +38,end-calc	e'-1,X'	
	INC (HL)	Se dublează X' pentru a da	
		2*X'	
	RST 0028,FP-CALC	e'-1,2*X'	
373D GRE.8	DEFB +01,exchange	X',e	- X' mare
		2*X',e'-1	- X' mic
	DEFB +34,stk-data	X',e',LN 2	
	DEFB +F0,exponent+80	2*X',e'-2,LN 2	
	DEFB +31,+72,+17,+F8		
	DEFB +04,multiply	X',e'*LN 2 = Y1	
		2*X',(e'-1)*LN 2 = Y2	

Se execută pasul iv.

DEFB +01,exchange	Y1,X'	- X' mare
	Y2,2*X'	- X' mic
DEFB +A2,stk-half	Y1,X',.5 (zecimal)	
	Y2,2*X',.5	
DEFB +03,subtract	Y1,X',-.5	
	Y2,2*X',-.5	
DEFB +A2,stk-half	Y1,X',-.5,.5	
	Y2,2*X',-.5,.5	
DEFB +03,subtract	Y1,X',-1	
	Y2,2*X',-1	

Se execută pasul v.

DEFB +31,duplicate	Y,X'-1,X'-1	
	Y2,2*X',-1,2*X',-1	
DEFB +34,stk-data	Y1,X'-1,X',-1,2.5 (zecimal)	
	Y2,2*X',-1,2*X',-1,2.5	
DEFB +32,exponent+82		
DEFB +20,(+00,+00,+00)		
DEFB +04,multiply	Y1,X'-1,2.5*X',-2.5	
	Y2,2*X',-1,5*X',-2.5	
DEFB +A2,stk-half	Y1,X'-1,2.5*X',-2.5,.5	
	Y2,2*X',-1,5*X',-2.5,.5	
DEFB +03,subtract	Y1,X'-1,2.5*X',-3=7	
	Y2,2*X',-1,5*X',-3=7	

Se execută pasul vi, trecînd în SERIES GENERATOR parametrul '12' zecimal, si a 12 constantă cerută.

	DEFB +8C,series-0C	Y1,X'-1,7 sau Y2,2*X',-1,7
1.	DEFB +11,exponent+61	
	DEFB +AC,(+00,+00,+00)	
2.	DEFB +14,exponent+64	
	DEFB +09,(+00,+00,+00)	
3.	DEFB +56,exponent+66	
	DEFB +DA,+A5,(+00,+00)	
4.	DEFB +59,exponent+69	
	DEFB +30,+C5,(+00,+00)	
5.	DEFB +5C,exponent+6C	

```

6.  DEFB +90,+AA,(+00,+00)
    DEFB +9E,exponent+6E
    DEFB +70,+6F,+61,(+00)
7.  DEFB +A1,exponent+71
    DEFB +CB,+DA,+96,(+00)
8.  DEFB +A4,exponent+74
    DEFB +31,+9F,+B4,(+00)
9.  DEFB +E7,exponent+77
    DEFB +A0,+FE,+5C,+FC
10. DEFB +EA,exponent+7A
    DEFB +1B,+43,+CA,+36
11. DEFB +ED,exponent+7D
    DEFB +A7,+9C,+7E,+5E
12. DEFB +F0,exponent+80
    DEFB +6E,+23,+80,+93

```

La sfîrsitul ultimei bucle 'ultima valoare' este:

```

sau LD X'/(X'-1)      pentru valorile mai mari
                        ale lui X'
sau LD (2*X')/(2*X'-1) pentru valorile mai mici
                        ale lui X'

```

Se execută pasul vii.

```

DEFB +04,multiply      Y1=LN (2**e'),LN X'
DEFB +0F,addition      Y2=LN (2**(e'-1)),LN (2*X')
DEFB +38,end-calc      LD (2**e')*X'      =LN X
RET                     LN (2**(e'-1))*2*X'    =LN X
                        LN X
                        Sfîrsit.

```

THE 'REDUCE ARGUMENT' SUBROUTINE (SUBROUTINA 'REDUCERE ARGUMENT')
(Deplasament 39: 'get-argt')

Această subrutină transformă argumentul X al lui SIN X sau COS X într-o valoare V.

Subrutina găsește mai întâi o valoare Y în felul următor:

$Y = X/(2*PI) - INT(X/2*PI) + 0,5$, unde Y este mai mare sau egal cu -5 dar mai mic decât +5.

Subrutina revine cu:

```

V = 4*Y      dacă -1<=4*Y<=-1      - cazul i.
sau V = 2 - 4*Y  dacă 1< 4*Y < 2      - cazul ii.
sau V = -1*Y - 2  dacă -2<=4*Y<=-1    - cazul iii.

```

La fiecare caz, $-1<=v<=1$ și $SIN(PI*V/2) = SIN X$.

```

3783 get-argt      RST 0028,FP-CALC      X
DEFB +3D,re-stack  X (în forma în virgulă mobilă
DEFB +34,stk-data   desfășurată)
DEFB +EE,exponent+7E X,1/(2*PI)
DEFB +22,+F9,+83,+6E
DEFB +04,multiply   X/(2*PI)
DEFB +31,duplicate  X/(2*PI),X/(2*PI)
DEFB +A2,stk-half   X/(2*PI),X/(2*PI),0,5
DEFB +0F,addition   X/(2*PI),X/(2*PI)+0,5
DEFB +27,int.1C46   X/(2*PI),INT (X/(2*PI)+0,5)
DEFB +03,subtract,174C X/(2*PI)-INT (X/(2*PI)+0,5)=Y

```

Notă: Adunînd 0.5 și luînd INT se rotunjește rezultatul la cel mai apropiat întreg.

```

DEFB +31,duplicate  Y,Y
DEFB +0F,addition   2*Y
DEFB +31,duplicate  2*Y,2*Y
DEFB +0F,addition   4*Y
DEFB +31,duplicate  4*Y,4*Y
DEFB +2A,abs        4*Y,ABS (4*Y)
DEFB +A1,stk-one    4*Y,ABS (4*Y),1
DEFB +03,subtract   4*Y,ABS (4*Y)-1-Z
DEFB +31,duplicate  4*Y,Z,Z
DEFB +37,greater-0  4*Y,Z,(1/0)
DEFB +C0,st-meme-0 Mem-0 contine rezultatul
                        testului.
DEFB +00,jump-true  4*Y,Z
DEFB +04,to 37A1,ZPLUS 4*Y,Z
DEFB +02,delete     4*Y

```

DEFB +38,end-calc	4*Y = V - cazul i.
RET	Terminat.

Dacă s-a făcut saltul se continuă.

37A1 ZPLUS	DEFB +A1,stk-one	4*Y,Z,1
	DEFB +03,subtract	4*Y,Z-1
	DEFB +01,exchange	Z-1,4*Y
	DEFB +36,less-0	Z-1,(1/0)
	DEFB +00,jump-true	Z-1
	DEFB +02,to37A8,YNEG	Z-1
	DEFB +1B,negate	1-Z
37A8 YNEG	DEFB +38,end-calc	1-Z = V - cazul ii.
	RET	Z-1 = V - cazul iii.
		Terminat.

THE 'COSINE' FUNCTION (FUNCTIA 'COS')
(Deplasament 20: 'cos')

Această subrutină tratează funcția COS X și redă o 'ultimă valoare' care este o aproximare a lui COS X.

Subrutina folosește expresia:

$$\text{COS } X = \text{SIN } (\text{PI} * W / 2), \text{ unde } -1 \leq W \leq 1$$

Derivind W din X subrutina folosește rezultatul testului obținut în subrutina anterioară și în acest scop îl stochează în mem-0. Apoi se execută un salt la SINE, subrutină, intrând la C-ENT, pentru a produce o 'ultimă valoare' a lui COS X.

37AA cos	RST 0028,FP-CALC	X
	DEFB +39,get-argt	V
	DEFB +2A,abs	ABS V
	DEFB +A1,stk-one	ABS V,1
	DEFB +03,subtract	ABS V-1
	DEFB +E0,get-mem-0	ABS V-1,(1/0)
	DEFB +00,jump-true	ABS V-1
	DEFB +06,to 37B7,C-ENT	ABS V-1 = W

Dacă nu s-a executat saltul atunci se continuă.

DEFB +1B,negate	1-ABS V
DEFB +33,jump	
DEFB +03,to 37B7,C-ENT	1-ABS V = W

THE 'SINE' FUNCTION (FUNCTIA 'SIN')
(Deplasament 1F: 'sin')

Această subrutină tratează funcția SIN X și este a treia din cele patru subrutine care folosesc SERIES GENERATOR pentru a produce polinoamele Cebîsev.

Aproximarea pentru SIN X se găsește în modul următor:

- i. Argumentul X este redus și în acest caz $W=V$ direct.
De notat că $-1 \leq W \leq 1$, așa cum este necesar pentru ca seriile să fie convergente.
- ii. Este format argumentul Z, astfel încît $Z=2*W*W-1$.
- iii. Se folosește SERIES GENERATOR pentru a reda $(\text{SIN } (\text{PI} * W / 2)) / W$.
- iv. În final o simplă înmulțire dă SIN X.

37B5 sin	RST 0028,FP-CALC	X
----------	------------------	---

Se execută pasul i.

DEFB +39,get-argt	W
-------------------	---

Se execută pasul ii. De acum înainte subrutina este comună și pentru funcția SINE (SINUS) și pentru funcția COSINE (COSINUS).

37B7 C-ENT	DEFB +31,duplicate	W,W
	DEFB +31,duplicate	W,W,W
	DEFB +04,multiply	W,W*W
	DEFB +31,duplicate	W,W*W,W*W
	DEFB +0F,addition	W,2*W*W
	DEFB +A1,stk-one	W,2*W*W,1
	DEFB +03,subtract	W,2*W*W-1 = Z

Se execută pasul iii, trecînd în SERIES GENERATOR parametrul '6' și cele șase constante cerute.

1.	DEFB +86,series-06	W,Z
	DEFB +14,exponent+64	

```

      DEFB +E6,(+00,+00,+00)
2.    DEFB +5C,exponent+6C
      DEFB ++1F,+0B,(+00,+00)
3.    DEFB +A3,exponent+73
      DEFB +8F,+3B,+EE,(+00)
4.    DEFB +E9,exponent+79
      DEFB +15,+63,+AB,+23
5.    DEFB +EE,exponent+7E
      DEFB +92,+0D,+CD,+ED
6.    DEFB +F1,exponent+81
      DEFB +23,+5B,+1B,+EA

```

La sfîrsitul ultimei bucle 'ultima valoare' este (SIN (PI*W/2))/W.

Se execută pasul v.

```

      DEFB +04,multiply      SIN (PI*W/2) = SIN X (sau =
      DEFB +38,end-calc     COS X)
      RET                   Sfîrsit: 'ultima valoare' =
                           SIN X sau ('ultima valoare' =
                           COS X)

```

THE 'TAN' FUNCTION (FUNCTIA 'TANGENTA')
(Deplasament 21: 'tan')

Această subrutină tratează funcția TAN X. Această subrutină pur și simplu redă SIN X/COS X, cu depășire aritmetică dacă COS X = 0.

```

37DA tan      RST    0028,FP-CALC      X
              DEFB  +31,duplicate    X,X
              DEFB  +1F,sin          X,SIN X
              DEFB  +01,exchange     SI X,X
              DEFB  +20,cos          SIN X,COS X
              DEFB  +05,division     SIN X/COS X = TAN X
              DEFB  +38,end-calc     Se raportează depășire
              RET                   aritmetică dacă este necesar.
                                   TAN X
                                   Sfîrsit: 'ultima valoare' =
                                   TAN X.

```

THE 'ARCTAN' FUNCTION (FUNCTIA 'ARCTAN')
(Deplasament 24: 'atn')

Această subrutină tratează funcția ATN X și este ultima din cele patru subrutine care folosesc SERIES GENERATOR pentru a produce polinoamele Cebîșev. Ea redă un număr real cuprins între $-\pi/2$ și $\pi/2$, care este egal cu valoarea în radiani a unghiului a cărui tangentă este X.

Aproximarea pentru ATN X este găsită în felul următor:

- i. Valorile lui W și Y sînt găsite pentru trei cazuri ale lui X, care sînt:
- | | | | |
|-------------------|-------------------|------------|-------------|
| dacă $-1 < X < 1$ | atunci $W=0$ | & $Y=X$ | - cazul i |
| dacă $1 < X$ | atunci $W=\pi/2$ | & $Y=-1/X$ | - cazul ii |
| dacă $X < -1$ | atunci $W=-\pi/2$ | & $Y=-1/X$ | - cazul iii |

În fiecare caz, $-1 < Y < 1$, așa cum este necesar pentru ca seriile să fie convergente.

- ii. Argumentul Z este format după cum urmează:
- | | | |
|-------------------|------------------------------|-------------|
| dacă $-1 < X < 1$ | atunci $Z=2*Y*Y-1=2*X*X-1$ | - cazul i |
| dacă $1 < X$ | atunci $Z=2*Y*Y-1=2/(X*X)-1$ | - cazul ii |
| dacă $X < -1$ | atunci $Z=2*Y*Y-1=2/(X*X)-1$ | - cazul iii |

iii. SERIES GENERATOR este folosit pentru producerea funcției cerute.

iv. În final o simplă înmulțire și o adunare va da ATN X.

Se execută faza i.

```

37E2 atn      CALL   3297,RE-STACK    Se folosește forma în virgulă
              LD     A,(HL)          mobilă desfășurată a lui X.
              CP     +81             Se aduce exponentul lui X.
              JR     C,37F8,SMALL    Salt înainte în cazul i: Y=X.
              RST   0028,FP-CALC     X
              DEFB  +A1,stk-one      X,1
              DEFB  +1B,negate       X,-1
              DEFB  +01,exchange     -1,X
              DEFB  +05,division     -1/X
              DEFB  +31,duplicate     -1/X,-1/X

```

	DEFB +36,less-0	-1/X,(1/0)
	DEFB +A3,stk-pi/2	-1/X,(1/0),PI/2
	DEFB +01,exchange	-1/X,PI/2,(1/0)
	DEFB +00,jump-true	-1/X,PI/2
	DEFB +06,to 37FA,CASES	Salt înainte pentru cazul ii: Y = -1/x W = PI/2
	DEFB +1B,negate	-1/X,PI/2
	DEFB +33,jump	- - - - -
	DEFB +03,to 37FA,CASES	Salt înainte pentru cazul iii: Y = -1/X W = -PI/2
37F8 SMALL	RST 0028,FP-CALC	Y
	DEFB +A0,stk-zero	Y,0
		Se continuă pentru cazul i: W = 0

Se execută pasul ii.

37FA CASES	DEFB +01,exchange	W,Y
	DEFB +31,duplicate	W,Y,Y
	DEFB +31,duplicate	W,Y,Y,Y
	DEFB +04,multiply	W,Y,Y*Y
	DEFB +31,duplicate	W,Y,Y*Y,Y*Y
	DEFB +0F,addition	W,Y,2*Y*Y
	DEFB +A1,stk-one	W,Y,2*Y*Y,1
	DEFB +03,subtract	W,Y,2*Y*Y-1 = Z

Se execută pasul iii, trecînd în SERIES GENERATOR parametrul '12' zecimal, si cele 12 constante cerute.

	DEFB +8C,series-0	W,Y,Z
1.	DEFB +10,exponent+60	
	DEFB +B2,(+00,+00,+00)	
2.	DEFB +13,exponent+63	
	DEFB +0E,(+00,+00,+00)	
3.	DEFB +55,exponent+65	
	DEFB +E4,+8D,(+00,+00)	
4.	DEFB +58,exponent+68	
	DEFB +39,+BC,(+00,+00)	
5.	DEFB +5B,exponent+6B	
	DEFB +98,+FD,(+00,+00)	
6.	DEFB +9E,exponent+6E	
	DEFB +00,+36,+75,(+00)	
7.	DEFB +A0,exponent+70	
	DEFB +DB,+E8,+B4,(+00)	
8.	DEFB 63,exponent+73	
	DEFB +42,+C4,(+00,+00)	
9.	DEFB +E6,exponent+76	
	DEFB +B5,+09,+36,+BE	
10.	DEFB +E9,exponent+79	
	DEFB +36,+73,1B,+5D	
11.	DEFB +EC,exponent+7C	
	DEFB +D8,+DE,+63,+BE	
12.	DEFB +F0,exponent+80	
	DEFB +61,+A1,+B3,+0C	

La sfîrsitul ultimei bucle 'ultima valoare' este:

ATN X/X	- cazul i.
ATN (-1/X)/(-1/X)	- cazul ii.
ATN (-1/X)/(-1/X)	- cazul iii.

Se execută pasul iv.

DEFB +04,multiply	W,ATN X	- cazul i.
	W,ATN (-1/X)	- cazul ii.
	W,ATN (-1/X)	- cazul iii.
DEFB +0F,addition	ATN X	- toate cazurile
DEFB +38,end-calc		
RET	Sfîrsit: 'ultima valoare' =	ATN X.

THE 'ARCSIN' FUNCTION (FUNCTIA 'ARCSIN')
(Deplasament 22: 'asn')

Această subrutină tratează funcția ASN X și redă un număr real cuprins între -PI/2 și PI/2 inclusiv, care este egal cu valoarea în radiani a unghiului al cărui sinus este X.

Astfel dacă Y = ASN X atunci X = SIN Y

Această subrutină folosește identitatea trigonometrică:

$TAN (Y/2) = SIN Y / (1 + COS Y)$
 pentru a obtine $TAN (Y/2)$ si de aici (utilizând ATN) $Y/2$ si în final Y .

3833 asn	RST	0028,FP-CALC	X
	DEFB	+31,duplicate	X,X
	DEFB	+31,duplicate	X,X,X
	DEFB	+04,multiply	X,X*X
	DEFB	+A1,stk-one	X,X*X,1
	DEFB	+03,subtract	X,X*X-1
	DEFB	+1B,negate	X,1-X*X
	DEFB	+28,sqr	X,SQR (1-X*X)
	DEFB	+A1,stk-one	X,SQR (1-X*X),1
	DEFB	+0F,addition	X,1*SQR (1-X*X)
	DEFB	+05,division	X/(1+SQR (1-X*X)) = TAN (Y/2)
	DEFB	+24,atn	Y/2
	DEFB	+31,duplicate	Y/2,Y/2
	DEFB	+0F,addition	Y = ASN X
	DEFB	+38,end-calc	
	RET		Sfîrsit: 'ultima valoare' = ASN X

THE 'ARCCOS' FUNCTION (FUNCTIA 'ARCCOS')
 (Deplasament 23: 'acs')

Această subrutină tratează funcția ACS X și redă un număr real cuprins între zero și PI inclusiv care este egal cu valoarea în radiani a unghiului al cărui cosinus este X.

Subrutina folosește relația:
 $ACS X = PI/2 - ASN X$

3843 acs	RST	0028,FP-CALC	X
	DEFB	+22,asn	ASN X
	DEFB	+A3,stk-pi/2	ASN X,PI/2
	DEFB	+03,subtract	ASN X-PI/2
	DEFB	+1B,negate	PI/2-ASN X = ACS X
	DEFB	+38,end-calc	
	RET		Sfîrsit: 'ultima valoare' = ACS X

THE 'SQUARE ROOT' FUNCTION (FUNCTIA 'RADACINA PATRATA')
 (Deplasament 28: 'sqr')

Această subrutină tratează funcția SQR X și redă rădăcina pătrată pozitivă a unui număr real X dacă X este pozitiv, și zero dacă X este negativ. O valoare negativă a lui X dă dreptul la prezentarea A - argument invalid (prin ln în subrutina EXPONENTIATION).

Această subrutină tratează operația rădăcina pătrată ca fiind $X**5$ și astfel stochează valoarea .5 și trece direct la subrutina EXPONENTIATION.

384A sqr	RST	0028,FP-CALC	X
	DEFB	+31,duplicate	X,X
	DEFB	+30,not	X,(1/0)
	DEFB	+00,jump-true	X
	DEFB	+1E,to 384C,LAST	X

Se execută saltul dacă $X = 0$, în caz contrar se continuă cu:

DEFB	+A2,stk-half	X,.5
DEFB	+38,end-calc	

și apoi se găsește rezultatul lui $X**5$.

THE 'EXPONENTIATION' OPERATION (FUNCTIA 'EXPONENTIALIZARE')
 (Deplasament 06: 'to-power')

Această subrutină execută operația binară de ridicare a primului număr, X, la puterea celui de al doilea număr, Y.

Subrutina tratează rezultatul $X**Y$ ca fiind echivalent cu $EXP (Y*LN X)$. Ea redă această valoare numi dacă X este pozitiv și raportează depășire aritmetică dacă Y este negativ.

3851 to-power	RST	0028,FP-CALC	X,Y
	DEFB	+01,exchange	Y,X
	DEFB	+31,duplicate	Y,X,X
	DEFB	+30,not	Y,X,(1/0)
	DEFB	+00,jump-true	Y,X
	DEFB	+07,to 385D,XISO	Y,X

Se execută saltul dacă $X = 0$, în caz contrar se formează $EXP (Y*LN X)$

DEFB	+25,ln	Y, LN X
		Se dă prezentarea A dacă X este negativ.
DEFB	+04,multiply	Y*LN X
DEFB	+38,end-calc	
JP	36C4,EXP	Iesire prin EXP pentru a forma EXP (Y*LN X).

Valoarea lui X este zero așa că se iau în considerare cele 3 cazuri implicate.

385D XISD	DEFB	+02,delete	Y
	DEFB	+31,duplicate	Y,Y
	DEFB	+30,not	Y,(1/0)
	DEFB	+00,jump-true	Y
	DEFB	+09,to 386A,ONE	Y

Se execută saltul dacă X = 0 și Y = 0, altfel se continuă.

DEFB	+A0,stk-zero	Y,0
DEFB	+01,exchange	0,Y
DEFB	+37,greater-0	0,(1/0)
DEFB	+00,jump-true	0
DEFB	+06,to 386C,LAST	0

Se execută saltul dacă X = 0 și Y este pozitiv, altfel se continuă.

DEFB	+A1,stk-one	0,1
DEFB	+01,exchange	1,0
DEFB	++05,division	Se iese prin 'division' ca si cum s-ar împarti la zero, obținând 'depășire aritmetică'.

Rezultatul operației trebuie să fie 1.

386A ONE	DEFB	+02,delete	-
	DEFB	+A1,stk-one	1

Acum se revine cu 'ultima valoare' din stivă ca fiind 0**Y.

386C LAST	DEFB	38,end-calc	(1/0)
	RET		Sfârșit: 'ultima valoare' este 0 sau 1.

386E - 3CFF Aceste locații sînt de 'rezervă'. Ele contin +FF. (sint 1169)

3D00 - 3FFF Aceste locații contin 'setul caracterelor'. Sînt 8 octeți reprezentati pentru toate caracterele cu codurile cuprinse între +20 (spatiu) și +7F (@).

De exemplu litera 'A' are reprezentarea 00 3C 42 42 7E 42 42 00 și deci are forma:

```

00000000
00111100
01000010
01000010
01111110
01000010
01000010
01000010
00000000

```

APENDIX (ANEXA)

BASIC PROGRAMS FOR THE MAIN SERIES (PROGRAMELE BASIC PENTRU SERIILE PRINCIPALE)

Următoarele programe BASIC au fost incluse deoarece ele oferă o bună imagine a modului în care sînt folosite polinoamele Cebîsev pentru a realiza aproximările funcțiilor SIN, EXP, LN și ATN.

Generatorul seriilor:

Această subrutină este apelată de toate programele 'funcție':

```

500 REM SERIES GENERATOR, ENTER
510 REM USING THE COUNTER BREG
520 REM AND ARRAY-A HOLDING THE
530 REM CONSTANTS
540 REM FIRST VALUE IN 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 LAST VALUE IN T

```

Z - valoarea de intrare
T - valoarea de iesire
M0 - mem-0
M1 - mem-1
M2 - mem-2
I - contorul pentru BREG
U - o variabilă temporară pentru T

A(1) la
A(BREG) - constante
BREG - numărul de constante ce vor fi folosite

Pentru a vedea cum sînt generate polinoamele Cebîsev, se înregistrează pe hîrtie valorile lui U, M1, M2 și T din liniile 550 la 630, trecînd, să zicem, de 6 ori prin buclă, și păstrînd expresiile algebrice de la A(1) la A(6) fără a se substitui valorile numerice. Apoi se înregistrează T-M1. Inmutitorii constantelor A(1) la A(6) vor fi deci polinoamele Cebîsev. Mai exact, înmulțitorul lui A(1) va fi $2*T_5(Z)$, al lui A(2) va fi $2*T_4(Z)$ și așa mai departe, pînă la $2*T_1(Z)$ pentru A(5) și în final $T_0(Z)$ pentru A(6).

De notat că $T_0(Z)=1$, $T_1(Z)=Z$ și, pentru $n \geq 2$, $T_n(Z)=2*Z*T_{n-1}(Z) - T_{n-2}(Z)$.

SIN X

```

10 REM DEMONSTRATION FOR SIN X
20 REM USING THE 'SERIES GENERATOR'
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)=-.142630785
90 LET A(6)=1.276278962
100 PRINT
110 PRINT "ENTER START VALUE IN DEGREES"
120 INPUT C
130 CLS
140 LET C=C-10
150 PRINT "BASIC PROGRAM", "ROM PROGRAM"
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 USE 'SERIES GENERATOR'
270 GO SUB 550

```

```

280 PRINT TAB 6;"SIN";C;"DEGREES"
290 PRINT
300 PRINT T,W,SIN(PI*C/180)
310 PRINT
320 NEXT J
330 GO TO 100

```

NOTE:

i. Când se introduce C, acest program calculează și tipărește SIN C grade, SIN (C+10) grade, SIN (C+20) grade și SIN (C+30) grade. De asemenea tipărește valorile obținute folosind programul ROM. Pentru un exemplu de rezultate, încercați să introduceți aceste valori în grade: 0,5; 100; -80; -260; 3600; -7200.

ii. Constantele A(1) la A(6) din liniile 40 la 90 sînt date (diferit de un factor 1/2) în Abramowitz și Stegun - Handbook of Mathematical Functions (Dover 1965) pag.76. Ele pot fi verificate integrînd $(\text{SIN } (\text{PI} \cdot \text{X}/2))/\text{X}$ în intervalul $U=0$ la PI , după ce mai întîi s-a multiplicat cu $\text{COS } (N \cdot U)$ pentru fiecare constantă (aducă $N=1,2,\dots,6$) și înlocuind $\text{COS } U = 2 \cdot \text{X} \cdot \text{X} - 1$. Fiecare rezultat trebuie apoi împărțit cu PI . (Această integrare poate fi realizată prin metode aproximative, ca de exemplu Regula lui Simpson, dacă se lucrează cu un calculator adecvat sau cu un calculator programabil.

EXP X

```

10 REM DEMONSTRATION FOR EXP X
20 REM USING THE 'SERIES GENERATOR'
30 LET T=0 (Aceasta face T prima variabilă.)
40 DIMA(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 "ENTER START VALUE"
150 INPUT C
160 CLS
170 LET C=C-10
180 PRINT "BASIC PROGRAM","ROM PROGRAM"
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 "));EXP C=2**D).
240 LET N=INT D
250 LET Z=D-N (2**(N+Z) este acum cerut.)
260 LET Z=2-Z-1
270 LET BREG=8
280 REM USE "SERIES GENERATOR"
290 GO SUB 550
300 LET V=PEEK 32627+256*PEEK 23628+1 (V=(VARS)+1)
310 LET N=N+PEEK V
320 IF N>255 THEN STOP (STOP cu depășire aritmetică.)
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

```

NOTE:

i. Când se introduce C acest program calculează și tipărește EXP C, EXP (C+10), EXP (C+20) și EXP (C+30). De asemenea tipărește valorile obținute folosind programul ROM. Pentru un exemplu de rezultate, încercați să introduceți următoarele valori: 0; 15; 65 (cu depășire la sfîrșit); -100; -40.

ii. Exponentul este testat pentru depășire și pentru un rezultat zero în liniile 320 și 330. Aceste teste sînt mai simple în BASIC decît în cod masină,

cînd variabila N, spre deosebire de registrul A, nu este limitată la un octet.

iii. Constantele A(1) la A(8) din liniile 50 la 120 se pot obtine integrînd 2^*X în intervalul U=0 la PI, după ce mai întîi s-a multiplicat cu COS (N*U) pentru fiecare constantă (adică pentru N=1, 2, ..., 8) și s-a înlocuit COS U=2*X-1. Fiecare rezultat trebuie apoi împărțit la PI.

LN X

```

10 REM DEMONSTRATION FOR LN X
20 REM USING THE 'SERIES GENERATOR'
30 LET D=0 (Aceasta face D prima variabilă.)
40 DIM A(12)
50 LET A(1)=-.0000000003
60 LET A(2)=0.0000000020
70 LET A(3)=-.0000000127
80 LET A(4)=0.0000000023
90 LET A(5)=-.00000005389
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 "ENTER START VALUE"
190 INPUT C
200 CLS
210 PRINT "BASIC PROGRAM", "ROM PROGRAM"
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 cu 'argument invalid')
280 LET D=C
290 LET V=PEEK 23627+256*PEEK 23628+1
300 LET N=PEEK V-128 (N contine e')
310 POKE V,128
320 IF D<=0.8 THEN GO TO 360 (D contine e')
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 contine N*LN 2)
400 LET BREG=12
410 REM USE 'SERIES GENERATOR'
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

```

NOTE:

i. Cînd se introduce C, acest program calculează și tipărește LN C, LN (C**2), LN (C**4) și LN (C**8). De asemenea tipărește valorile obținute folosind programul ROM.

Pentru un exemplu de rezultate, încercați să introduceți următoarele valori: 1.1; 0.9; 300; 0.004; 1E5 (pentru depășire) și 1E-5 (STOP ca fiind un 'argument invalid').

ii. Constantele A(1) la A(12) din liniile 50 la 100 pot fi obținute integrînd $S*LN (4*(X+1)/5)/(4*X-1)$ în intervalul U=0 la PI, după ce mai întîi s-a înmulțit cu COS (N*U) pentru fiecare constantă (adică N=1, 2, ..., 12) și s-a substituit COS U=2*X-1. Apoi fiecare rezultat trebuie împărțit cu PI.

ATN X

```

10 REM DEMONSTRATION FOR ATN X
20 REM USING THE 'SERIES GENERATOR'
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)=-.0006905975
130 LET A(10)=0.0055679210
140 LET A(11)=-.0529464623
150 LET A(12)=0.8813735870
160 PRINT
170 PRINT "ENTER START VALUE"
180 INPUT C
190 CLS
200 PRINT "BASIC PROGRAM";"ROM PROGRAM"
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 DE=-1/B
270 LET Z=2.D.D-1
280 LET BREG=12
290 REM USE "SERIES GENERATOR"
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           (sau PRINT T*180/PI,ATN B*180/PI
370 PRINT                  pentru a obtine rezultatul în grade)
380 NEXT J
390 GO TO 160

```

NOTE:

i. Când se introduce C, acest program calculează și tipărește ATN C, ATN (C*2), ATN (C*3) și ATN (C*4).

Pentru un exemplu de rezultate, încercați să introduceți valorile: 0.2; -1; 10 și -100. Rezultatele pot fi găsite mai interesant dacă se converteste pentru a produce gradele, multiplicând răspunsul din linia 350 cu 180/PI.

ii. Constantele A(1) la A(12) din liniile 40 la 150 se dau (spre deosebire de un factor de 1/2) în Abramowitz și Stegun, Handbook of Mathematical Functions (Dover 1965) pagina 82. Ele pot fi verificate intergrând ATN X/X în intervalul U = 0 la PI, după ce mai întâi s-a multiplicat cu COS (N*U) pentru fiecare parametru (adică pentru N = 1, 2, ..., 12) și înlocuind COS U = 2*X*X-1. Fiecare rezultat trebuie apoi împărțit cu PI.

0 subrutină alternativă pentru SIN X:

Este cinstit să se realizeze întreaga dezvoltare pentru polinoamele Cebîșev și aceasta se poate scrie în BASIC după cum urmează:

```

550 LET T=+(32.Z.Z.Z.Z.Z-40.Z.Z.Z.Z+10.Z).A(1)
      +(16.Z.Z.Z.Z-16.Z.Z.Z+2).A(2)
      +(8.Z.Z.Z-6.Z).A(3)
      +(4.Z.Z-2).A(4)
      +2.Z.A(5)
      +A(6)
560 RETURN

```

Această subrutină este apelată în locul SERIES GENERATOR și se observă că au aceeași acurătate.

0 subrutină alternativă pentru EXP X:

Întreaga dezvoltare pentru EXP X este:

```

550 LET T=(128.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-96.Z.Z.Z.Z+36.Z.Z-2).A(2)
      +(32.Z.Z.Z.Z.Z-40.Z.Z.Z+10.Z).A(3)
      +(16.Z.Z.Z.Z-16.Z.Z.Z+2).A(4)
      +(8.Z.Z.Z-6.Z).A(5)
      +(4.Z.Z-2).A(6)
      +2.Z.A(7)
      +A(8)

```

560 RETURN

Dezvoltarea pentru LN X si ATN X, dată în formă algebrică, va fi:

$$\begin{aligned} & (2048z^{11} - 5632z^9 + 5632z^7 - 2464z^5 + 440z^3 - 22z) * A(1) \\ + & (1024z^{10} - 2560z^8 + 2240z^6 - 800z^4 + 100z^2 - 2) * A(2) \\ + & (512z^8 - 1152z^6 + 864z^4 - 210z^2 + 18z) * A(3) \\ + & (256z^7 - 512z^5 + 320z^3 - 64z + 2) * A(4) \\ + & (128z^6 - 224z^4 + 12z^2 - 14z) * A(5) \\ + & (64z^5 - 96z^3 + 36z - 2) * A(6) \\ + & (32z^4 - 40z^2 + 10z) * A(7) \\ + & (16z^3 - 16z + 2) * A(8) \\ + & (18z^2 - 6z) * A(9) \\ + & (4z - 2) * A(10) \\ + & (2z) * A(11) \\ + & A(12) \end{aligned}$$

THE 'DRAW' ALGORITHM (ALGORITHM 'DRAW' ('DESENARE'))

Următorul program BASIC ilustrează partea principală a operației DRAW (desenare) care a fost folosită la desenarea unei linii drepte. Programul în forma sa prezentă este permis doar pentru linii în care $X > Y$.

```

10 REM DRAW 255,175 PROGRAM
20 REM SET ORIGIN
30 LET PLOTx=0; LET PLOTy=0
40 REM SET LIMITS
50 LET X=255; LET Y=175
60 REM SET INCREMENT,i
70 LET i=X/2
80 REM ENTER LOOP
90 FOR B=X TO 1 STEP -1
100 LET A=Y+i
110 IF X>A THEN GO TO 160
120 REM UP A PIXEL ON THIS PASS
130 LET A=A-X
140 LET PLOTy+1
150 REM RESET INCREMENT,i
160 LET i=A
170 REM ALWAYS ALONG ONE PIXEL
180 LET PLOTx=PLOTx+1
190 REM NOW MAKE A PLOT
200 PLOT PLOTx,PLOTy
210 NEXT B

```

Un algoritm complet va fi găsit în programul următor, ca o subrutină care va desena o linie ('DRAW A LINE') din ultima poziție până la X,Y.

THE 'CIRCLE' ALGORITHM (ALGORITHM 'CIRCLE' ('CERC'))

Următorul program BASIC ilustrează cum o comandă CIRCLE realizează cercurile sale.

Mai întâi se calculează numărul cerut de arcuri. Apoi se pregătește un set de parametri în 'spațiul de memorie' și în 'stiva calculatorului'.

Apoi se desenează arcurile prin apelări repetate ale subrutinei de desenare linie, care la fiecare apelare desenează o singură linie din 'ultima poziție' până în poziția X,Y.

Notă: În programul ROM există o linie finală 'de închidere' dar această caracteristică nu a fost inclusă aici.

```

10 REM A CIRCLE PROGRAM
20 LET X=127; LET Y=87; LET Z=87
30 REM How many arcs?
40 LET Arcs=4*INT(INT(ABS(PI-SQR Z)+0.5)/4+4)
50 REM Set up memory area;M0-M5
60 LET M0=X+Z
70 LET M1=0
80 LET M2=2*Z*SIN(PI/Arcs)
90 LET M3=1-2*(SIN(PI/Arcs)) 2
100 LET M4=SIN(2*PI/Arcs)
110 LET M5=2*PI
120 REM Set up stack; Sa-Sd
130 LET Sa=X+Z
140 LET Sb=Y-Z*SIN(PI/Arcs)
150 LET Sc=Sa
160 LET Sd=Sb
170 REM Initialise COORDS

```

```

180 POKE 23677,Sa; POKE 23678,Sb
190 LET M0=Sd
200 REM 'DRAW THE ARCS'
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 Arcs=Arcs-1; IF Arcs=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 'DRAW LINE' from last position to 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=Y; LET B=Y
560 LET ddx=0; LET ddy=0
570 GO TO 610
580 IF X+Y=0 THEN STOP
590 LET L=Y; LET B=X
600 LET ddx=dx; LET ddy=dy
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 PLOTx<0 OR PLOTx>225 THEN STOP
740 PLOT PLOTx,PLOTy
750 NEXT N
760 RETURN

```

NOTE ON SMALL INTEGERS AND -65536 (NOTA ASUPRA INTREGILOR MICI SI ASUPRA LUI -65536)

1. Intregii mici n sînt aceia pentru care -65535 este mai mic sau egal cu n care este mai mic sau egal cu 65535. Forma în care sînt tratate este descrisă în 'STACK-BC'. De notat că manualul este inexact cînd spune că al treilea și al patrulea octet contin n plus 131072 dacă n este negativ. Intrucît intervalul lui n este atunci -1 la -65535, cei doi octeti pot să contină doar n plus 131072 dacă este luat mod 65536; aceasta înseamnă că ei contin 65536. Manualul modifică ieșirea. Adevărul este că aceasta nu este o adevărată formă în complement față de doi (asa cum este forma n plus 131072, în anumite circumstanțe, ar putea fi). Aici, aceiași număr poate fi pus pentru două numere diferite, în concordanță cu octetul semn; de exemplu 00 01 se pune pentru 1 dacă octetul semn este FF; în mod similar FF FF se pune pentru 65535 dacă octetul semn este 00 și pentru -1 octetul semn este FF.

2. Acceptînd că numerele negative sînt date într-o formă specială de 'complement față de doi', trăsătura caracteristică a acestei metode de tratare a numerelor este aceea că ele sînt gata pentru o 'scurtă adunare' fără vreo complementare față de doi mai departe. Ele sînt aduse simplu și stocate direct prin subrutina de adunare. Dar pentru înmulțire ele trebuie aduse prin INT-FETCH și stocate după aceea prin INT-STORE. Aceste subrutine completează numărul în raport cu doi la aducerea sau stocarea lui. Apelul subrutinei INT-STORE este din 'multiply' (după 'short multiplication'), din 'truncate' (după formarea unui 'întreg mic' cuprins între -65535 și 65535 inclusiv), din 'negate'/'abs' pentru 'integer case' și din 'sgn' pentru a stoca 1 sau -1. Apelul subrutinei INT-FETCH se face din PRINT-FP pentru aducerea părții întregi a numărului cînd este 'small', din 'multiply' de două ori pentru a aduce doi 'small integers', din 'RE-STACK' pentru aducerea unui 'small integer' pentru restocare, din 'negate'/'abs' pentru a aduce un 'small integer' pentru modificare și din FP-TO-BC pentru a aducerea întregului pentru a-l transfera în BC.

Numărul -65536

3. numărul -65536 poate fi reprezentat printr-un 'întreg mic' format ca 00 FF 00 00 00. El este apoi 'număr limitare', unul care cînd este complementat față

de doi depăsește (se compară 80 hex într-un simplu octet sau 7 biti sistem, adică -128 zecimal, care când se completează față de doi mai dă 80 hex, adică -128 zecimal, cât timp numărul pozitiv 128 zecimal nu poate fi reprezentat în sistem).

4. Câteva asemănări dintre acestea pot inspira abandonarea în timpul creerii lui 00 FF 00 00 00 în 'truncate'. Este abandonată întrucât aceasta nu ar mai supraviețui rutinei INT din care s-a 'trunchiat' o parte. Aceasta doar conduce la greșeala INT (-65536) egal cu -1.

5. Dar principala eroare este aceea că acest număr a fost încuviințat să apară din 'adunarea scurtă' a doi întregi mici negativi și apoi a fost pus simplu în stivă ca 00 FF 00 00 00. Sistemul nu poate face față la acest număr. Soluția propusă în 'addition' ('adunare') este aceea de a forma imediat toți cei cinci octeți ai formei în virgulă mobilă; această înseamnă prima dată testarea numărului, în jurul octetului 3032, după cum urmează:

3032	PUSH	AF	Se salvează octetul semn în A
3033	INC	A	Se transformă orice FF din A în 00.
3034	OR	E	Se testează toți cei trei octeți pentru zero.
3035	OR	D	
3036	JR	NZ,3040,ADD-STORE	Salt dacă nu este -65536.
3038	POP	AF	Se șterge stivă.
3039	LD	(HL),+80	Se introduce 80 hex în al doilea octet.
303B	DEC	HL	Se indică primul octet.
303C	LD	(HL),+91	Se introduce 91 hex în primul octet.
303E	JR	3049,ADD-RSTOR	Salt pentru a seta indicatorul și ieșire.
3040 ADD-STORE	POP	AF	Se readuce octetul semn în A.
3041	LD	(HL),A	Acesta este stocat în stivă.
3042	INC	HL	Este indicată următoarea locație.
3043	LD	(HL),E	Se stochează octetul cel mai puțin semnificativ al rezultatului.
3044	INC	HL	Este indicată următoarea locație.
3045	LD	(HL),D	Se stochează octetul cel mai semnificativ al rezultatului.
3046	DEC	HL	Se mută indicatorul înapoi
3047	DEC	HL	pentru a adresa primul octet
3048	DEC	HL	al rezultatului.
3049 ADD-RSTOR	POP	DE	Se readuce STKEND în DE.
304A	RET		Sfîrșit.

6. Îmbunătățirea de deasupra (adică 15 octeți aditionali) cu omiterea octetilor 3223 la 323E inclusiv din 'truncate' trebuie să rezolve problema. Ar fi bine să se poată testa aceasta. Apelul subrutinei INT-STORE nu ar trebui să ducă la stocarea lui 00 FF 00 00 00. În 'multiply' (înmulțire) numărul va conduce la depășire dacă se cere, cât timp 65536 va seta fanionul de transport; așa că se va folosi înmulțirea 'lungă'. Cum s-a notat la 30E5, cei 5 octeți începînd de aici pot fi probabil omisi dacă s-a făcut îmbunătățirea de deasupra. 'Negarea' evită stocarea lui 00 FF 00 00 00 tratînd zero separat și returnîndu-l nemodificat. Trunchierea lucrează separat cu -65535, cum s-a notat deasupra. SGN stochează numai 1 și -1.

INDEXAREA RUTINELOR

adresa	rutina	Pag.
RUTINELE DE REPORNIRE SI TABELE		
0000	START	6
0008	Eroare	6
0010	Tipărirea unui caracter	6
0018	Colectare caracter	6
0020	Colectarea caracterului următor	6
0028	Calcul	7
0030	Executarea a 80 spații	7
0038	Intrerupere mascabilă	7
0053	ERROR-2	7
0066	Intrerupere nemascabilă	8
0074	CH-ADD+1	8
007D	SKIP-OVER	8
0095	Tabele de simboluri	9
0205	Tabele de taste	10
RUTINELE TASTATURII		
028E	Baleierea tastaturii	11
028F	Tastatura	12
031C	Repetare tastă	14
031F	K-TEST	15
0333	Decodificarea tastaturii	15
RUTINELE DE DIFUZOR		
0385	BEEPER	19
03F8	BEEP	21
046E	Tabela de semitonuri	23
RUTINELE DE TRATARE A CASETEI		
04C2	SA-BYTES	24
053F	SA/LD-RET	27
0556	LD-BYTES	28
05E3	LD-EDGE-2	32
0605	SAVE-ETC	33
07CB	Control VERIFY	40
0802	Încărcarea unui bloc de informații	41
0808	Control LOAD	41
0886	Control MERGE	44
092C	MERGE Line/Var	47
0970	Control SAVE	48
09A1	Mesajele casetei	49
RUTINELE DE TRATARE A ECRANULUI SI IMPRINANTEI		
09F4	PRINT-OUT	49
0A11	Tabelul caracterelor de control	50
0A23	Cursor la stînga	50
0A3D	Cursor la dreapta	51
0A4F	Carriage return	51
0A5F	Tipărire virgulă	51
0A69	Tipărire semn de întrebare	51
0AED	Caractere de control cu operanți	52
0AD9	PO-ABLE	53
0ADC	Rezervare poziție	54
0B03	Aducere poziție	54
0B24	Tipărirea oricărui caracter	54
0B7F	Tipărirea tuturor caracterelor	56
0BDB	Setarea octetului atribut	58
0CCA	Tipărire mesaj	59
0C3B	PQ-SAVE	60
0C41	Căutare în tabel	60
0C55	Testare scroll (defilare)	60
0CF8	Mesajul 'scroll?'	63
0D4D	Numerale culorii temporare	64
0D6B	Comanda CLS	64
0DAF	Stergerea întregului spațiu al ecranului	65
0DD9	CL-SET	66
0DFE	Defilare	67
0E44	Stergere linie	68
0E8B	CL-ATTR	70
0E9B	CL-ADDR	70
0EAC	Comanda COPY	71
0ECD	COPY-BUF	72
0EDF	CLEAR PRINTER BUFFER	72
0EF4	COPY-LINE	72

0F2C	EDITOR	74
0F81	ADD-CHAR	75
0FA0	Tabelul tastelor de editare	75
0FA9	Tasta EDIT	76
0FF3	Editare cursor jos	77
1007	Editare cursor stînga	77
100C	Editare cursor dreapta	77
1015	Editare DELETE (stergere)	77
101E	ED-IGNORE	77
1024	Editare ENTER	77
1031	ED-EDGE	78
1059	Editare cursor sus	79
1076	ED-SYMBOL	79
107F	ED-ERROR	79
1097	CLEAR-SP	79
10AB	Introducere tastatură	80
111D	Copierea părții inferioare a ecranului	82
1190	SET-HL	83
11A7	REMOVE-FP	83

RUTINELE EXECUTIVULUI

11B7	Comanda NEW	84
11CB	Intrarea principală (Initializare)	84
11DA	RAM-CHECK	84
12A2	Bucla de executie principală	87
1391	Mesaje de prezentare	90
155D	MAIN-ADD	90
15AF	Canalul initial de informatie	92
15C6	Sirul initial de informatie	92
15D4	WAIT-KEY	92
15E6	INPUT-AD	93
15EF	Tipărirea principală	93
1601	CHAN-OPEN	93
1615	CHAN-FLAG	94
162D	Tabelul imagine al codului canalului	94
1634	Fanionul canalului K	95
1642	Fanionul canalului S	95
164D	Fanionul canalului P	95
1652	ONE-SPACE	95
1655	MAKE-ROOM	95
1664	POINTERS	96
168F	Colectarea unui număr de linie	97
169E	RESERVE	97
16D0	SET-MIN	98
16D4	Readucerea liniei editare	98
16DB	INDEXER	98
16E5	CLOSE (inchiderea) unei comenzi	99
1716	Tabelul 'Trecere prin sir închis'	100
171E	Sir de informatii	100
1736	Comanda OPEN#	101
177A	Tabelul 'Trecere prin sir deschis'	102
1793	Comenzile CAT, ERASE, FORMAT & MOVE	103
1795	Comenzile LIST & LLIST	103
1795	AUTO-LIST	103
17F5	LLIST	104
17F9	LIST	104
1855	Tipărirea unei întregi linii BASIC	106
18B6	NUMBER	107
18C1	Tipărirea unui caracter pîlpîitor	107
18E1	Tipărire cursor	108
190F	LN-FETCH	109
1925	Tipărirea de caractere într-o linie BASIC	109
196E	LINE-ADDR	111
1980	Compararea numerelor de linie	111
1988	Găsirea fiecărei instrucțiuni	111
19B8	NEXT-ONE	112
19DD	Diferențiere	113
19E5	Refacere	114
19FB	E-LINE-NO	114
1A1B	Tipărire prezentare si număr linie	115

INTERPRETAREA LINIEI BASIC SI A COMENZILOR

1A48	Tabelele de sintaxă	116
1B17	Analiza principală (interpretorul BASIC)	119
1B28	Instrucțiunea de ciclare	119
1B52	SCAN-LOOP	120
1B6F	SEPARATOR	121
1B76	STMT-RET	121
1B8A	LINE-RUN	121

1B9E	LINE-NEW	122
1BB2	Comanda REM	122
1BB3	LINE-END	122
1BBF	LINE-USE	123
1BD1	NEXT-LINE	123
1BEF	CHECK-END	124
1BF4	STMT-NEXT	124
1C01	Tabelul 'Clasa de comandă'	124
1C0D	Clasele de comandă - 00, 03 & 05	124
1C1E	JUMP-C-R	125
1C1F	Clasele de comandă - 01, 02 & 04	125
1C22	Variabilă în atribuire	125
1C5E	Aducerea unei valori	126
1C79	Asteptare expresie numerică/sir	127
1C96	Setarea culorii permanente (clasa 07)	128
1CBE	Clasa de comandă 0	129
1CDB	Clasa de comandă 0B	129
1CDE	Aducerea unui număr	129
RUTINELE DE COMANDA		130
1CEE	Comanda STOP	130
1CF0	Comanda IF	130
1D03	Comanda FOR	130
1D86	LOOK-PROG	133
1DAB	Comanda NEXT	134
1DDA	NEXT-LOOP	135
1DEC	Comanda READ	135
1E27	Comanda DATA	136
1E39	PASS-BY	137
1E42	Comanda RESTORE	137
1E4F	Comanda RANDOMIZE	137
1E5F	Comanda CONTINUE	138
1E67	Comanda GO TO	138
1E7A	Comanda OUT	138
1E80	Comanda POKE	138
1E85	TWO-PARAM	139
1E94	Băsirea părții întregi	139
1EA1	Comanda RUN	139
1EAC	Comanda CLEAR	139
1EED	Comanda GO SUB	140
1F05	TEST-ROOM	141
1F1A	Memorie liberă	141
1F23	Comanda RETURN	142
1F3A	Comanda PAUSE	142
1F54	BREAK-KEY	143
1F60	Comanda DEF FN	143
1FC3	UNSTACK-Z	145
1FC9	Comanda LPRINT	145
1FCF	Comanda PRINT	146
1FF3	Tipărire 'carriage return'	146
1FFC	Trimitere informație	146
2043	Sfîrsitul tipăririi	147
204E	Pozitie tipărire	148
2070	Alterare sir	148
2089	Comanda INPUT	149
21B9	IN-ASSIGN	152
21D6	IN-CHAN-K	153
RUTINE INFORMATII CULOARE		153
21E1		153
226C	Co-CHANGE	156
2294	Comanda BORDER	157
22AA	Adresa pixelului	157
22C3	Punct	158
22DC	Comanda PLOT	158
2307	STK-TO-BC	159
2314	STK-TQ-A	159
2320	Comanda CIRCLE	160
2382	Comanda DRAW	162
247D	Parametrii initiali	168
24B7	Trasare linie	169
EVALUAREA EXPRESIEI		
24FB	SCANNING	172
2530	SYNTAX-Z	173
2535	Baleiere SCREEN#	173
2580	Baleiere ATTR	175
2596	Baleierea tabelului functie	175
26AF	Baleierea rutinelor functie	180
26C9	Baleierea rutinei variabile	181

2734	Baleierea buclei principale	184
2795	Tabelul operatorilor	186
2780	Tabelul de priorități	186
278D	Funcția de baleiere (FN)	186
28AB	FN-SKPOVR	191
28B2	LOOK-VARS	192
2951	Argumentul funcției stocate	196
2996	STK-VAR	197
2A52	SLICING	202
2AB1	STK-STORE	204
2ACC	INT-EXP	205
2AEE	DE, (DE+1)	206
2AF4	GET-HL*DE	206
2AFF	Comanda LET	207
2BF1	STK-FETCH	214
2C02	Comanda DIM	214
2C88	ALPHANUM	217
2C8D	ALPHA	218
2C96	Zecimal în virgulă mobilă	218
2D1B	NUMERIC	221
2D22	STK-DIGIT	221
2D28	STACK-A	221
2D2B	STACK-BC	221
2D3B	Întreg în virgulă mobilă	222

RUTINELE ARITMETICE

2D4F	E- format în virgulă mobilă	222
2D7F	INT-FETCH	224
2D8E	INT-STORE	225
2DA2	Virgulă mobilă în BC	226
2DC1	LOG (2:A)	226
2DDS	Virgulă mobilă în A	227
2DE3	Tipărirea unui număr în virgulă mobilă	228
2F8B	CA=10*A+C	237
2F9B	Pregătire pentru adunare	238
2FBA	Aducerea a două numere	238
2FDD	Deplasare termen adunare	240
3004	ADD-BACK	241
300F	Scădere (03)	241
3014	Adunare (0F)	241
30A9	HL=HL*DE	245
30C0	Pregătire pentru înmulțire sau împărțire	246
30CA	Înmulțire (04)	246
31AF	Împărțire (05)	252
3214	Trunchiere întreg față de zero	254
3293	Restocare doi	258
3297	RE-STACK (3D)	258

CALCULUL ÎN VIRGULĂ MOBILĂ

32C5	Tabela de constante	259
32D7	Tabela de adrese	260
335B	CALCULATE	262
33A1	Stergere (02)	264
33A2	Operație	264
33A9	Testare 5-spacii	265
33B4	Stocare număr	265
33C0	Mutarea unui număr în virgulă mobilă (31)	265
33C6	Stocare literaluri (34)	265
33F7	Omitere constante	267
3406	Locatarea memoriei	268
340F	Aducere din spațiul memoriei (E0 etc.)	268
341B	Stocare constantă (A0, etc.)	268
342D	Stocare în spațiul memoriei	269
343C	EXCHANGE (01)	269
3449	Generator serii (86, etc.)	270
346A	Mărime absolută (2A)	272
346E	Minus unar (1B)	272
3492	Signum (29)	273
34A5	IN (2C)	273
34AC	PEEK (2B)	274
34B3	Număr USR (2D)	274
34DC	Sir USR (19)	274
34E9	TEST-ZERO	276
34F9	Mai mare ca zero (37)	276
3501	NOT (30)	276
3506	Mai mic ca zero (36)	277
350B	Zero sau unu	277
351B	OR (07)	278
3524	Număr AND număr (08)	278

352D	Sir AND număr (10)	278
353D	Comparare (09-0E, 11-16)	279
359C	Concatenarea sirului (17)	281
358F	STK-PNTRS	281
35C9	CHR\$ (2F)	282
35DE	VAL si VAL\$ (1D, 18)	282
361F	STR\$ (2E)	284
3645	READ-IN (1A)	284
3669	CODE (1C)	285
3674	LEN (1E)	285
367A	Decrementare contor (35)	285
3686	Salt (33)	286
368F	Salt conditionat (00)	286
369B	END-CALC (38)	287
36A0	Modul (32)	287
36AF	INT (27)	288
36C4	Exponential (26)	289
3713	Logaritm natural (25)	291
3783	Reducerea argumentului (39)	294
37AA	Cosinus (20)	295
37B5	SINE (1F)	295
37DA	Tangenta (21)	296
37E2	ARCTAN (24)	296
3833	Arccosinus (22)	299
3843	Arccosinus (23)	299
384A	Radical (extragerea rădăcinii pătrate) (28)	300
3851	Exponentializare (06)	300
ANEXA		
Programe BASIC pentru cele mai importante serii:		302
- generator serii		302
- SIN X		302
- EXP X		303
- LN X		305
- ATN X		306
Algoritmul 'DRAW'		308
Algoritmul 'CIRCLE'		308
Notă asupra întregilor mici si -65536		310

Va multumim că ati cumpărat manualul firmei noastre. Acest manual a fost editat si corectat cu toată atenta si presupunem că este corect (dar desigur perfectibil).

ALPHA Ltd. va îmbunătățește permanent manualele editate si de aceea vă sintem recunoscători pentru orice sesizare. Vă așteptăm cu orice problemă la sediul firmei si la tel. 961/12936

AJ