

ING. DANIEL JENNE A KOLEKTIV ZX ROMI VÝPIS

ZENITCENTRUM 08

ZX



VÝPIS

ROMI

ZX  VÝPIS

ROM

Vydavatelství Naše vojsko, n. p., Praha - 1989

Vytiskly JČT, n. p., České Budějovice, provoz 6 Jindřichův Hradec



## Úvod

16K monitorový program Spectra je složitý program ve strojovém kódu procesoru Z80. Jeho struktura je velmi zřetelná a to proto, že je rozdělena do tří hlavních částí:

- a) Vstupní a výstupní podprogramy
- b) Interpret jazyka BASIC
- c) Manipulace s výrazy

Tyto bloky jsou ale ještě příliš velké na to, aby je bylo možno snadno pochopit vcelku, a proto je monitorový program v této knize rozdělen do deseti částí.

Nyní nastíníme každou z těchto částí:

### Restartovací podprogramy a tabulky

Na začátku monitorového programu jsou různé restartovací podprogramy, které jsou volány jednobajtovými instrukcemi RST. Tyto restarty jsou využity všechny. Například restart #08 je využíván pro hlášení různých sdělení nebo chyb. Tabulky v této části monitorového programu obsahují nezkrácené tvary klíčových slov a kódy tlačítek.

### Podprogram pro obsluhu klávesnice

Klávesnice je testována každou 1/50 sec, a po testu je na příslušnou systémovou proměnnou uložen kód příslušné klávesy. Všechna tlačítka na klávesnici "opakují" jestliže jsou stlačena delší dobu. O to se stará také tento podprogram.

### Podprogram pro obsluhu reproduktoru

Spectrum má vestavěný reproduktor a tóny vznikají opakovaným užitím instrukce OUT. V řídícím podprogramu byla věnována velká pozornost tomu, aby bylo zajištěno, že žádaný tón bude mít příslušnou výšku a délku trvání.

### Podprogramy obsluhující kazetový magnetofon

Jednou ze slabých stránek počítače ZX-81 bylo to, že jen velmi malá část jeho monitorového programu byla určena k obsluze kazetového magnetofonu. Ale ve Spectru je už rozsáhlý blok strojového kódu a dá se říci, že jedním z nejuspěšnějších rysů Spectra je právě vysoká kvalita ovládacích podprogramů. Basicové programy nebo bloky dat jsou ukládány pomocí hlavičkového bloku, který má 17 bajtů a je ukládán první. Hlavička popisuje charakter dat ukládaných v následujícím bloku. Jedinou "nevýhodou" tohoto systému je to, že není možno zabezpečit utajení bloků dat.

### Podprogramy obsluhující obrazovku a tiskárnu

Všechny ostatní vstupní a výstupní podprogramy jsou "vektorovány" pomocí kanálových a "streamových" (proudových) adres. Ve standardním Spectru je vstup možný pouze z klávesnice, ale výstup může být směřován na tiskárnu a horní nebo dolní část televizního displeje. Hlavní "vstupní" podprogram v této části monitorového programu je EDITOR, který umožňuje uživateli vkládat znaky do dolní části televizního displeje. Podprogram PRINT OUT je dosti pomalý, protože je společný pro všechny druhy tisku. Například přidání jednoho bajtu na displej v sobě zahrnuje též současné posouzení stavu funkcí OVER a INVERSE při každém použití tohoto podprogramu.

## Prováděcí podprogramy

V této části monitorového programu můžeme najít iniciační podprogram a hlavní prováděcí smyčku interpretu jazyka Basic. Ve Spectru je kontrolována syntaxe basicového řádku, který je posléze uložen do programové oblasti, jestliže měl číslo, jinak je okamžitě vykonán, což může ve svém důsledku vést k různým situacím. (Zřetelně je to vidět v případě operace RUN).

## Interpretace příkazů jazyka Basic

Tato část monitorového programu považuje basicový řádek za soubor příkazů. Pro každý příkaz je zde "příkazový" podprogram, který provede (interpretuje) příslušný příkaz jako sled instrukcí ve strojovém kódu.

## Vyhodnocování výrazů

Spectrum má velmi obsažný program pro vyhodnocování výrazů, který umožňuje využít širokou škálu typů proměnných, funkcí a operací. Tato část monitoru je opět dosti pomalá, a to právě proto, že se zabývá všemi možnými alternativami. Zvláště zpracování řetězců je velmi dobře zvládnuto. Všechny jednoduché řetězce jsou uloženy dynamicky a jejich staré kopie jsou okamžitě, jakmile se stanou nadbytečnými, zničeny. To znamená, že není potřeba provádět vytřídování "smetí" tak jako u jiných systémů.

## Aritmetické podprogramy

Spectrum používá dva typy čísel. Celočíselné hodnoty v rozsahu -65535 až 65535 jsou takzvané krátké (integer) formy, zatímco všechna ostatní čísla jsou ukládána v pětibajtové formě zvané pohyblivá řádová čárka (floating point). Současná verze monitoru má však bohužel také dvě chyby.

- 1) Chyba v dělení (34. bit je při dělení ztracen).
- 2) Hodnota -65535 je někdy uložena ve formě integer a jindy ve formě floating point, což někdy působí problémy.

## Kalkulátor pro reálná čísla

Kalkulátor Spectra zpracovává čísla a řetězce. Jeho operace jsou specifikovány tzv. "literály". Tato část monitorového programu obsahuje podprogramy pro všechny matematické funkce. Funkce jako sin, exp, ln atd. jsou získávány aproximačním způsobem. Aproximace se provádí rozvojem Čebyševových polynomů.

Celkově se dá říci, že 16K monitor nabízí extrémně široký rozsah různých basicových příkazů a funkcí, ale protože jeho tvůrce měli k dispozici jen omezený prostor v paměti, je tento program spíše kompaktnější než rychlejší.

## PODPROGRAMY RESTARTŮ A TABULKY

### RST #00 START

Maskované přerušení je zakázáno a registrový pár DE nastaven tak, aby obsahoval nejvyšší možnou adresu paměti RAM.

0000	START	DI		Zákaz přerušení i testu klávesnice
		XOR	A	#00 pro start, ale #FF pro NEW
		LD	DE,#FFFF	Nejvyšší možná adresa RAM
		JP	#11CB,START/NEW	Skok dopředu

### RST #08 ERROR RESTART

Chybový ukazatel je nastaven tak, aby ukazoval na pozici chyby.

0008	ERROR-1	LD	HL,(#5C5D) {CH-ADD}	Adresa dosažená interpretem
		LD	(#5C5F),HL {X-PTR}	je okopírována do chybového ukazatele,
		JP	#0053,ERROR-2	dříve než se bude pokračovat.

### RST #10 RESTART TISK ZNAKU

Registr A obsahuje kód znaku, který má být vytištěn.

0010	PRINT-A-1	JP	#15F2,PRINT-A-2	Okamžitý skok dopředu.
		DEFB	#FF,#FF,#FF,#FF,#FF	Volné místo.

### RST #18 RESTART NAČTENÍ ZNAKU

Obsah místa adresovaného systémovou proměnnou CH-ADD je přečten. Návrat, jestliže se jedná o znak použitelný k tisku, jinak je CH-ADD zvětšen a testy se opakují.

0018	GET-CHAR	LD	HL,(#5C5D) {CH-ADD}	Vyzvedni hodnotu adresovanou
		LD	A,(HL)	systémovou proměnnou CH-ADD.
001C	TEST-CHAR	CALL	#007D,SKIP-OVER	Zjistí, zda se jedná o znak použitelný k tisku a
		RET	NC	vrat se, je-li to pravda.

### RST #20 RESTART NAČTENÍ DALŠÍHO ZNAKU

Při interpretaci basicového řádku je tento podprogram opakovaně vyvoláván, což způsobuje postupování po řádku.

0020	NEXT-CHAR	CALL	#0074,CH-ADD+1	Je nutno zvětšit CH-ADD.
		JR	#001C,TEST-CHAR	Skok zpět a test nové hodnoty.
		DEFB	#FF,#FF,#FF	Volná místa.

### RST #28 RESTART KALKULÁTORU

Kalkulátor pracující s pohyblivou řádovou čárkou začíná na adrese #335B.

0028	FP-CALC	JP	#335B,CALCULATE	Okamžitě skoč dopředu.
		DEFB	#FF,#FF,#FF,#FF,#FF	Volná místa.

### RST #30 RESTART VYTVOŘENÍ BC PROSTORU

Tento podprogram vytváří volná místa v pracovním prostoru. Počet vytvářených míst je dán hodnotou uloženou v reg. BC.

0030	BC-SPACES	PUSH BC	Uchovej čítač.
		LD HL,(#5C61) (WORKSP)	Vyzvedni současnou adresu pracovního prostoru a
		PUSH HL	uchovej ji před
		JP #169E,RESERVE	vstupem do samotného podprogramu.

### RST #38 PODPROGRAM MASKOVATELNÉHO PŘERUŠENÍ

Hodiny reálného času ve SPECTRU jsou zvětšeny a současně je testována klávesnice kdykoliv dojde k maskovatelnému přerušení.

0038	MASK-INT	PUSH AF	Uchovej hodnoty uložené
		PUSH HL	v těchto registrech.
		LD HL,(#5C78) (FRAMES)	Dolní dva bajty systémové proměnné FRAMES
		INC HL	jsou zvětšeny
		LD (#5C78),HL (FRAMES)	každých 20 ms. (Evropská norma)
		LD A,H	Nejvyšší bajt této proměnné je
		OR L	zvětšen pouze tehdy,
		JR NZ,#0048,KEY-INT	je-li hodnota nižších dvou
		INC (IY+64) (FRAMES+2)	bajtů nulová.
0048	KEY-INT	PUSH BC	Uchovej hodnoty obsažené
		PUSH DE	v těchto registrech.
		CALL #02BF,KEYBOARD	Nyní testuj klávesnici.
		POP DE	Obnov hodnoty
		POP BC	v registrech.
		POP HL	
		POP AF	
		EI	Maskovatelné přerušení je před
		RET	návratem znovu povoleno.

### PODPROGRAM ERROR-2

Návratové adresy do interpretu ukazují na data, která určují, ke které chybě došlo. Tato data jsou vyzvednuta a přenesena do systémové proměnné ERR-NR. Než se provede odskok dopředu a vyčištěn zásobníku kalkulátoru, je také vyčištěn zásobník.

0053	ERROR-2	POP HL	Adresa na zásobníku
		LD L,(HL)	ukazuje na kód chyby.
0055	ERROR-3	LD (IY+0),L	(ERR-NR) Je přenesena do ERR-NR a provede
		LD SP,(#5C3D) {ERR-SP}	se vyčištění zásobníku před
		JP #16C5,SET-STK	odskokem do SET-STK.
		DEFB #FF,#FF,#FF,#FF	Volná místa.
		DEFB #FF,#FF,#FF	

### PODPROGRAM NEMASKOVATELNÉHO PŘERUŠENÍ

Tento podprogram není v normálním SPECTRU využit, ale měl by umožňovat NMI (nemaskovatelné přerušení). Systémová proměnná na adr. #5CB0, která se jmenuje NMIADD, musí mít nulovou hodnotu aby došlo k resetu.

0066	RESET	PUSH AF	Uchovej hodnoty
		PUSH HL	obsažené v registrech.
		LD HL,(#5CB0) {NMIADD}	Dva bajty proměnné NMIADD
		LD A,H	musí být oba
		OR L	nulové.



	JR	NZ,#0070,NO-RESET	Pozn.: zde mělo správně být JR z,#0070.
	JP	(HL)	Skok na START.
0070	NO-RESET	POP HL	Obnov hodnoty v
	POP	AF	registrech a
	RET		vrat se.

#### PODPROGRAM CH-ADD+1

Adresa, která se nachází v CH-ADD je zvětšena a opět uložena. Nyní se vezme obsah místa adresovaného CH-ADD. Vstupní body podprogramu TEMP-PTR1 a TEMP-PTR2 jsou použity, aby nastavily CH-ADD na přechodnou periodu.

0074	CH-ADD+1	LD	HL,(#5C5D)	(CH-ADD)	Vyzvedni adresu.
0077	TEMP-PTR1	INC	HL		Zvětši ukazatel.
0078	TEMP-PTR2	LD	(#5C5D),HL	(CH-ADD)	Nastav adresu.
		LD	A,(HL)		Vyzvedni adresovanou hodnotu
		RET			a vrat se.

#### PODPROGRAM SKIP-OVER

Hodnota, která do tohoto podprogramu vstupuje v registru A, je testována zda se jedná o znak, který je možno tisknout. Různé speciální kódy způsobují, že reg. HL je jednou nebo dvakrát zvětšen a podle toho se současně upravuje hodnota CH-ADD.

007D	SKIP-OVER	CP	#21		Vrat se s vynulovaným CY,
		RET	NC		jestli se jedná o obyčejný znak.
		CP	#0D		Vrat se, jestliže
		RET	Z		bylo dosaženo konce řádku.
		CP	#10		Jestliže se jedná o hodnotu v rozsahu #00 až #0F,
		RET	C		vrat se s nastaveným CY.
		CP	#18		Vrat se s kódy #18-#20 rovněž
		CCF			s nastaveným CY.
		RET	C		
		INC	HL		Jednou přeskoč.
		CP	#16		Skoč dopředu při
		JR	C,#0090,SKIPS		kódu #10 až #15 (INK až OVER)).
		INC	HL		Ještě jednou skoč (AT & TAB).
0090	SKIPS	SCF			Vrat se s nastaveným CY
		LD	(#5C5D),HL	(CH-ADD)	a s CH-ADD nastavenou na
		RET			příslušnou adresu.

#### TABULKA KÓDOVANÝCH KLÍČOVÝCH SLOV - "TOKENS"

Všechny TOKENS, které SPECTRUM používá jsou expandovány odkazy na tuto tabulku. Poslední bajt každého TOKEN je označen nastavením bitu 7.

0095	BF 52 4E C4 49 4E 4B 45	?	R	N	D	I	N	K	E
009D	59 A4 50 C9 46 CE 50 4F	Y	\$	P	I	F	N	P	O
00A5	49 4E D4 53 43 52 45 45	I	N	T	S	C	R	E	E
00AD	4E A4 41 54 54 D2 41 D4	N	\$	A	T	T	R	A	T
00B5	54 41 C2 56 41 4C A4 43	T	A	B	V	A	L	\$	C
00BD	4F 44 C5 56 41 CC 4C 45	O	D	E	V	A	L	L	E
00C5	CE 53 49 CE 43 4F D3 54	N	S	I	N	C	O	S	T
00CD	41 CE 41 53 CE 41 43 D3	A	N	A	S	N	A	C	S
00D5	41 54 CE 4C CE 45 58 D0	A	T	N	L	N	E	X	P
00DD	49 4E D4 53 51 D2 53 47	I	N	T	S	Q	R	S	G
00E5	CE 41 42 D3 50 45 45 CB	N	A	B	S	P	E	E	K
00ED	49 CE 55 53 D2 53 54 52	I	N	U	S	R	S	T	R

00F5	A4 43 48 52 A4 4E 4F D4	\$ C H R \$ N O T
00FD	42 49 CE 4F D2 41 4E C4	B I N O R A N D
0105	3C BD 3E BD 3C BE 4C 49	< = > = < > L I
010D	4E C5 54 48 45 CE 54 CF	N E T H E N T O
0115	53 54 45 D0 44 45 46 20	S T E P D E F
011D	46 CE 43 41 D4 46 4F 52	F N C A T F O R
0125	4D 41 D4 4D 4F 56 C5 45	M A T M O V E E
012D	52 41 53 C5 4F 50 45 4E	R A S E O P E N
0135	20 A3 43 4C 4F 53 45 20	# C L O S E
013D	A3 4D 45 52 47 C5 56 45	# M E R G E V E
0145	52 49 46 D9 42 45 45 D0	R I F Y B E E P
014D	43 49 52 43 4C C5 49 4E	C I R C L E I N
0155	CB 50 41 50 45 D2 46 4C	K P A P E R F L
015D	41 53 C8 42 52 49 47 48	A S H B R I G H
0165	D4 49 4E 56 45 52 53 C5	T I N V E R S E
016D	4F 56 45 D2 4F 55 D4 4C	O V E R O U T L
0175	50 52 49 4E D4 4C 4C 49	P R I N T L L I
017D	53 D4 53 54 4F D0 52 45	S T S T O P R E
0185	41 C4 44 41 54 C1 52 45	A D D A T A R E
018D	53 54 4F 52 C5 4E 45 D7	S T O R E N E W
0195	42 4F 52 44 45 D2 43 4F	B O R D E R C O
019D	4E 54 49 4E 55 C5 44 49	N T I N U E D I
01A5	CD 52 45 CD 46 4F D2 47	M R E M F O R G
01AD	4F 20 54 CF 47 4F 20 53	O T O G O S
01B5	55 C2 49 4E 50 55 D4 4C	U B I N P U T L
01BD	4F 41 C4 4C 49 53 D4 4C	O A D L I S T L
01C5	45 D4 50 41 55 53 C5 4E	E T P A U S E N
01CD	45 58 D4 50 4F 4B C5 50	E X T P O K E P
01D5	52 49 4E D4 50 4C 4F D4	R I N T P L O T
01DD	52 55 CE 53 41 56 C5 52	R U N S A V E R
01E5	41 4E 44 4F 4D 49 5A C5	A N D O M I Z E
01ED	49 C6 43 4C D3 44 52 41	I F C L S D R A
01F5	D7 43 4C 45 41 D2 52 45	W C L E A R R E
01FD	54 55 52 CE 43 4F 50 D9	T U R N C O P Y

### TABULKY KLÁVES

Ve SPECTRU je 6 oddělených tabulek kláves. Výsledný kód znaku závisí na stlačení určité klávesy a na momentálně použitém režimu.

#### (a) Tabulka hlavních kláves - L režim a CAPS SHIFT.

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

**(b) Režim EXTEND. Klávesy písmen.**

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

**(c) Režim EXTEND. Klávesy písmen a SYMBOL SHIFT.**

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

**(d) Řídicí kódy. Klávesy číslí a CAPS SHIFT.**

0260	0C 07 06 04	DELETE	EDIT	CAPS LOCK	TRUE VIDEO
0264	05 08 0A 0B	INV.VIDEO	kurzor vlevo	dolů	nahoru
0268	09 0F	vpravo	GRAPHICS		

**(e) Kódy symbolů. Klávesy písmen a SYMBOL SHIFT.**

026A	E2 2A 3F CD	STOP	*	?	STEP
026E	C8 CC CB 5E	>=	TO	THEN	^
0272	AC 2D 2B 3D	AT	-	+	=
0276	2E 2C 3B 22	.	,	;	"
027A	C7 3C C3 3E	<=	<	NOT	>
027E	C5 2F C9 60	OR	/	<>	£
02B2	C6 3A	AND	:		

**(f) Režim EXTEND. Klávesy číslí a SYMBOL SHIFT.**

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

## KLÁVESNICOVÉ PODPROGRAMY

### PODPROGRAM VYHODNOCENÍ KLÁVESNICE

Tento velmi důležitý podprogram je vyvoláván z hlavního klávesnicového podprogramu a z podprogramu INKEY\$ (při SCANNINGU). Ve všech případech obsahuje registr E hodnotu v rozsahu #00-#27 byla-li stlačena některá z kláves, nebo hodnotu #FF nebyla-li stlačena žádná klávesa. Registr D obsahuje hodnotu, která indikuje stišťení SHIFTovacích kláves. Jestliže byly stlačeny obě SHIFTovací klávesy, obsahují registry D a E hodnoty pro CAPS SHIFT a SYMBOL SHIFT. Jestliže nebyla stišťena žádná klávesa obsahuje registrový pár DE hodnotu #FFFF. Nulový indikátor se vrací vynulován, když došlo ke stišťení dvou kláves a ani jedna z nich nebyla klávesa SHIFT.

```
028E KEY-SCAN LD L,#2F Počáteční hodnota pro každý řádek bude: #2F, #2E, ..., #28
              LD DE,#FFFF (8 řádků).
              LD BC,#FEFE Nastav registrový pár DE na signál "žádná klávesa".
                          Registr C obsahuje adresu portu, registr B je čítač.
```

Nyní se vstupuje do smyčky, která bude vykonána osmkrát a při každém průchodu se bude začínat s novou počáteční hodnotou v registru L jak bylo uvedeno výše.

```
0296 KEY-LINE IN A,(C) Je čten specifikovaný port.
              CPL Stišťená klávesa v této řádce nastaví příslušný bit
              AND #1F (jedná se o bity 0-4).
              JR Z,#02AB,KEY-DONE Nebyla-li stišťena žádná klávesa provede se odskok,
              LD H,A jinak jsou klávesové bity uloženy do registru H a
              LD A,L kód "první" klávesy v tomto řádku je uložen do registru A.
029F KEY-3KEYS INC D Jestliže byly stišťeny 3 klávesy, nemůže registr D
              RET NZ obsahovat hodnotu #FF a
              SUB #08 provede se návrat.
02A1 KEY-BITS SRL H V této smyčce je
              JR NC,#02A1,KEY-BITS opakovaně odečítána hodnota #08, dokud není
              LD D,E nalezen klávesový bit.
              LD E,A Okopíruj předchozí klávesovou hodnotu do registru D a
              JR NZ,#029F,KEY-3KEYS ulož novou klávesovou hodnotu do registru E.
              Scoč zpět, jsou-li ještě další stisknuté klávesy
              v tomto řádku.
02AB KEY-DONE DEC L Řádka byla prozkoumána počáteční hodnota může být snížena.
              RLC B Čítač je posunut doleva a
              JR C,#0296,KEY-LINE provede se odskok, jestliže se nejednalo o poslední řádek.
```

Nyní se provedou čtyři testy:

```
LD A,D Pokud registr D obsahuje hodnotu #FF,
INC A je zároveň s touto hodnotou přijata i
RET Z jakákoliv hodnota v registru E.
CP #28 Je-li v registru D hodnota #28 (CAPS SHIFT), je opět
RET Z přijata i jakákoliv hodnota v registru E.
CP #19 Je-li v registru D hodnota #19 (SYMBOL SHIFT), je také
RET Z přijata i jakákoliv hodnota v registru E.
LD A,E Je ovšem možné,
LD E,D že v registru E je
LD D,A hodnota pro SYMBOL SHIFT a
CP #18 toto musí být také zvaženo.
RET Z=0, jestliže se nejednalo o SYMBOL SHIFT a nějakou klávesu
```

## PODPROGRAM KLÁVESNICE

Tento podprogram je volán při každém maskovatelném přerušení, což se děje 50krát za sekundu, je-li procesor v modu IM1. Úkolem tohoto podprogramu je otestovat klávesnici a dekodovat hodnoty stišťených kláves. Výsledný kód stišťené klávesy je předán do systémové proměnné LAST-K, pokud to umožní "opakovací statut". Po uložení získaného kódu do syst. proměnné je nastaven bit 5 v systémové proměnné FLAGS jako signál, že byla stlačena klávesa.

02BF KEYBOARD CALL #028E,KEY-SCAN číslo určující typ stlačené klávesy je uloženo do reg. DE  
RET NZ Bylo-li stlačeno více kláves (ne SHIFT) provede se návrat.

Je využíván zdvojený systém systémových proměnných KSTATE (KSTATE0-KSTATE3 (1.sada) a KSTATE4-KSTATE7 (2.sada)), který umožňuje detekci stišťení další klávesy i když program ještě pokračuje v opakovací periodě předchozí klávesy. Systém přijme novou klávesu, byla-li stišťena alespoň 1/10 sec., což odpovídá pěti voláním podprogramu klávesnice.

LD HL,#5C00 (KSTATE0) Začne se s KSTATE0.  
02C6 K-ST-LOOP BIT 7,(HL) Je-li systém volný (tj. KSTATE0/4 obsahují hodnotu #FF),  
JR NZ,#02D1,K-CH-SET provede se odskok dopředu.  
INC HL HL ukazuje na čítač pěti volání  
DEC (HL) který je následovně  
DEC HL zmenšen a  
JR NZ,#02D1,K-CH-SET provede se odskok, nebyla-li 1.sada volná.  
LD (HL),#FF Signál: 1.sada volná.

Po otestování 1. sady systému bude změněn ukazatel a testována 2. sada.

02D1 K-CH-SET LD A,L Nižší bajt adresy 1.sady je porovnán  
LD HL,#5C04 (KSTATE4) s nižším bajtem  
CP L adresy 2.sady a  
JR NZ,#02C6,K-ST-LOOP provede se odskok při kontrole 2. sady.

Provede se návrat, nebyla-li stišťena žádná klávesa nebo pouze jedna z kláves SHIFT.

CALL #031E,K-TEST Proveď nezbytné testy  
RET NC a vrať se, je-li to třeba.

Nyní bude rozlišeno zda se jedná o nový stisk, nebo byla-li stišťena nějaká klávesa již delší dobu.

LD HL,#5C00 (KSTATE0) Nejdříve zkontroluj 1.sadu  
CP (HL) a jestliže se hodnoty shodují,  
JR Z,#0310,K-REPEAT skoč dopředu.  
EX DE,HL Je uschována adresa KSTATE0.  
LD HL,#5C04 (KSTATE4) Podívej se do 2.sady  
CP (HL) a jsou-li kódy shodné,  
JR Z,#0310,K-REPEAT skoč dopředu (opakuji).

Nová klávesa bude akceptována, je-li jedna ze sad systému KSTATE volná.

BIT 7,(HL) Posuď 2.sadu a skoč  
JR NZ,#02F1,K-NEW dopředu, je-li volná.  
EX DE,HL Nyní testuj  
BIT 7,(HL) 1.sadu a  
RET Z pokračuj, je-li volná.

Nové tlačítko musí být nyní akceptováno. Ale před tím než bude do LAST-K uložena nová hodnota, musí být bajty systému KSTATE nastaveny na správné hodnoty tak, aby neobsahovaly žádné opakovací hodnoty. Nyní může být zjištěn kód tlačítka.

02F1	K-NEW	LD	E,A	Kód je okopírován do registru E a
		LD	(HL),A	do KSTATE0/4.
		INC	HL	
		LD	(HL),#05	Počítadlo 5-ti průchodů
		INC	HL	je nastaveno na hodnotu 5.
		LD	A,(#5C09) (REPDEL)	Třetí proměnná tohoto systému je nastavena
		LD	(HL),A	na hodnotu systémové proměnné REPDEL (normálně 0.7 sec).
		INC	HL	HL ukazuje nyní na KSTATE3/7.

Dekódování hlavního kódu závisí na stavu MODE, bitu 3 systémové proměnné FLAGS a SHIFTového bajtu.

LD	C,(IY+7) (MODE)	Vyzvedni hodnotu MODE.
LD	D,(IY+1) (FLAGS)	Vyzvedni hodnotu FLAGS.
PUSH	HL	Ušchovej ukazatel
CALL	#0333,K-DECODE	po dobu dekodování hlavního kódu.
POP	HL	Konečná hodnota je uložena v proměnné
LD	(HL),A	KSTATE3/7 odkud je vybírána v případě opakování.

Další tři instrukční řádky jsou společné pro vyhodnocení jak nově vložených kláves tak opakujících kláves.

0308	K-END	LD	(#5C08),A (LAST-K)	Vlož novou hodnotu do LAST-K a
		SET	5,(IY+1) (FLAGS)	signalizuj, že byla stlačena nová klávesa
		RET		a konečně se vrať.

#### PODPROGRAM OPAKOVÁNÍ KLÁVESY

Klávesa bude opakována při prvním výskytu po zpoždovací periodě REPDEL (normálně 0.7 sec.) a při dalších výskytích (opakování) po zpoždovací periodě REPPER (normálně 0.1 sec).

0310	<b>K-REPEAT</b>	INC	HL	Ukazuj na čítač 5ti volání v sadě, která je právě používána,
		LD	(HL),#05	a nastav jí na hodnotu 5.
		INC	HL	HL nyní ukazuje na 3. systémovou proměnnou
				(tj. REPPER/REPDEL),
		DEC	(HL)	která je právě zmenšena a
		RET	NZ	je proveden návrat, jestliže zpoždovací perioda neproběhla.
		LD	A,(#5C0A) (REPPER)	V opačném případě bude hodnotu zpoždění
		LD	(HL),A	udávat hodnota na sys. proměnné REPPER.
		INC	HL	Opakování bylo přijato,
		LD	A,(HL)	takže bude nakonec do registru A uložena hodnota KSTATE3/7
		JR	#0308,K-END	a předána podprogramu K-END.

#### PODPROGRAM K-TEST

Hodnota klávesy je testována a provede se návrat, jestliže nebyla stišťena žádná klávesa nebo pouze některý SHIFT. Jinak bude nalezen kód pro příslušnou klávesu.

031E	K-TEST	LD	B,D	Je okopírován SHIFTovací bajt.
		LD	D,#00	Vynuluj registr D pro pozdější užití.
		LD	A,E	Přenes číslo klávesy do registru A a
		CP	#27	byl-li to CAPS SHIFT nebo žádná klávesa,
		RET	NC	proved návrat.
		CP	#18	Pokud se nejedná o SYMBOL SHIFT, provede se
		JR	NZ,#032C,K-MAIN	odsok dopředu.
		BIT	7,B	SYMBOL SHIFT a jiné klávesy jsou akceptovány a
		RET	NZ	je proveden návrat, jednalo-li se pouze o SYMBOL SHIFT.

Hlavní kód je nalezen v tabulce hlavních kláves.

032C	K-MAIN	LD	HL,#0205	Bázová adresa tabulky.
		ADD	HL,DE	Najdi v tabulce
		LD	A,(HL)	kód klávesy.
		SCF		Signál: platné stišťení.
		RET		Vrať se.

#### PODPROGRAM DEKÓDOVÁNÍ KLÁVESNICE

Do tohoto programu se vstupuje s "hlavním kódem" v registru E, s hodnotou systémové proměnné FLAGS v registru D, hodnotou MODE v registru C a se SHIFT bajtem v registru B. Posouzením těchto čtyř hodnot s pomocí šesti klávesových tabulek je získán "finální kód", který je na výstupu uložen v registru A.

0333	K-DECODE	LD	A,E	Okopíruj "hlavní kód".
		CP	#3A	Pokud se jedná o posouzení číslicových kláves, ENTER, SPACE a obou SHIFTů,
		JR	C,#0367,K-DIGIT	provede se skok.
		DEC	C	Zmenší hodnotu MODE
		JP	M,#034F,K-KLC-LET	a skoč dopředu, vyžadují-li to
		JR	Z,#0341,K-E-LET	mody K,L,C nebo mod E.

Zůstává pouze grafický mod a "finální kód" je vypočítán přímo z "hlavního kódu".

		ADD	A,#4F	Přičti doplněk a
		RET		vrať se s "finálním kódem".

Jsou posouzeny znaky v EXTENDED MODE.

0341	K-E-LET	LD	HL,#01EB	Bázová adresa tabulky "b".
		INC	B	Nebyla-li stisknuta žádná ze SHIFTových kláves,
		JR	Z,#034A,K-LOOK-UP	užij tabulku "b".
		LD	HL,#0205	Jinak užij bázovou tabulku "c".

Tabulky "b-f" jsou obslouženy následujícím podprogramem. Ve všech případech je nalezen a předán "finální kód".

034A	K-LOOK-UP	LD	D,#00	Vynuluj registr D.
		ADD	HL,DE	Najdi v tabulce
		LD	A,(HL)	"finální kód".
		RET		Pak se hezky vrať.

Nyní jsou posuzovány znaky v modech K, L a C. Ale nejprve se musí pracovat s kódy pro SYMBOL SHIFT.

034F	K-KLC-LET	LD	HL,#0229	Bázová adresa tabulky "e".
		BIT	0,B	Jedná-li se o SYMBOL SHIFT + písmeno
		JR	Z,#034A,K-LOOK-UP	skoč zpět.
		BIT	3,D	Jseš-li se v modu K,
		JR	Z,#0364,K-TOKENS	skoč dopředu.
		BIT	3,(IY+4B) (FLAGS2)	Byl-li nastaven
		RET	NZ	CAPS LOCK, vrať se s "finálním kódem".
		INC	B	Byl-li stišťen CAPS SHIFT,
		RET	NZ	vrať se s tou samou hodnotou.
		ADD	A,#20	Jedná-li se o malé písmeno, je třeba přičíst #20 k hlavnímu kódu

RET a provést návrat.

Číselné hodnoty pro "tokens" jsou získány přičtením hodnoty #A5 k "hlavnímu kódu".

0364 K-TOKENS ADD A,#A5 Přečti požadovaný doplněk  
RET a vrať se.

Nyní jsou posuzovány číselné klávesy, SPACE, ENTER a oba SHIFTY.

0367 K-DIGIT CP #30 Pokračuj jedině v tom případě, jedná-li se o číslo, tzn.  
RET C vrať se při SPACE (#20), ENTER (#0D) nebo oba SHIFTY (#0E).  
DEC C Nyní rozděl číselné klávesy do tří skupin  
JP M,#039D,K-KLC-DGT a to podle modů K,L,C  
JR NZ,#0389,K-GRA-DGT a modu G.  
LD HL,#0254 Pokračuj v modu E. Bázová adresa tabulky "f".  
BIT 5,B Použij tuto tabulku pro SYMBOL SHIFT a  
JR Z,#034A,K-L00K-UP číselné klávesy v EXTENDED MODU.  
CP #38 Jedná-li se o klávesy  
JR NC,#0382,K-8-&-9 8 a 9, skoč dopředu.

Číselné klávesy 0 až 7 mají dát buď kód barvy papíru nebo inkoustu - když je stlačen CAPS SHIFT.

SUB #20 Rozsah #30 až #37 je snížen na #10 až #17 a  
INC B jestliže nebyl užit CAPS LOCK,  
RET Z vrať se s hodnotou tohoto papíru.  
ADD A,#08 Ale kdyby byl rozsah #18 až #1E, znamená to  
RET že se jedná o barvu inkoustu.

Klávesy 8 a 9 musí dávat BRIGHT a FLASH.

0382 K-8-&-9 SUB #36 Hodnoty #38 a #39 dávají #02 a #03.  
INC B Nebyl-li použit CAPS SHIFT,  
RET Z provede se návrat (jedná se o kód pro BRIGHT).  
ADD A,#FE Je odečtena dvojka a vzniknou tak hodnoty #00 a #01 (FLASH).  
RET Návrat s těmito kódy.

Číselné klávesy v grafickém modu mají dát skupinu grafických znaků (#80 až #8f), grafický kód (#0F) a kód pro DELETE (#0C).

0389 K-GRA-DGT LD HL,#0230 Bázová adresa tabulky "d".  
CP #39 Použij tuto tabulku přímo jak pro klávesu 9, která  
JR Z,#034A,K-L00K-UP má provést GRAPHICS,  
CP #30 tak pro klávesu 0,  
JR Z,#034A,K-L00K-UP která provede DELETE.  
AND #07 Pro klávesy 1 až 8  
ADD A,#80 uprav rozsah na hodnoty #80 až #87 a  
INC B pokud nebyla stlačena ani jedna ze SHIFTovacích kláves,  
RET Z proved návrat.  
XOR #0F Byl-li stlačen některý SHIFT, uprav rozsah na  
hodnoty #88 až #8F.  
RET Návrat.

Nakonec posuzuj číselnicové klávesy v modech K, L, & C.

039D K-KLC-DGT INC B Pokud nebyl stlačen žádný SHIFT,  
RET Z vrať se přímo s "finálními kódy" #30 až #39.  
BIT 5,B Je-li stisknuta



LD HL,#0230 klávesa CAPS SHIFT,  
JR NZ,#034A,K-LOOK-UP použij tabulku "d".

Nyní mohou být nalezeny kódy pro různé číslicové klávesy při použití SYMBOL SHIFTu.

	SUB	#10	Sniž rozsah
	CP	#22	na #20 až #29 a odděl
	JR	Z,#03B2,K-&-CHAR	zavináč ( @ ) od ostatních.
	CP	#20	Znak "_" musí být také oddělen a proto se
	RET	NZ	vrať s "finálními kódy" #21, #23 až #29.
	LD	A,#5F	Ulož do registru A kód znaku "_" a
	RET		vrať se.
03B2	K-&-CHAR	LD	A,#40 Ulož do registru A kód znaku "@ a
		RET	vrať se.

## PODPROGRAMY PRO OVLÁDÁNÍ REPRODUKTORU

Dva podprogramy v této části jsou:

- a) podprogram BEEPER obsluhující reproduktor;
- b) podprogram pro příkaz BEEP.

Reproduktor je aktivován logickou nulou na D4 během instrukce OUT #FE, která používá port 254. Podobně při logické 1 je reproduktor deaktivován. Střídáním logických úrovní na D4 vzniká tón. (pozn.překl.: D4 označuje bit 4 na datové sběrnici) Uvažujeme tón střední C jehož frekvence je 261.63Hz. K vytvoření tohoto tónu musí být reproduktor střídavě zapínán a vypínán každou 1/523.26 sekundy. Systémové hodiny ve Spectru jsou nastaveny na kmitočet 3.5 MHz a tón střední C bude tedy vyžadovat vystřídání logických úrovní každých 6689 T stavů. Tato poslední hodnota je nepatrně snížena aby se předešlo nevyhnutelným časovým ztrátám. Představuje délku časové smyčky pro podprogram BEEPER.

### **PODPROGRAM BEEPER**

Do tohoto podprogramu se vstupuje s hodnotou f\*t v registrovém páru DE, kde f je daná frekvence po dobu t sekund. Registrový pár HL obsahuje počet T stavů děleno 4. Tedy pro tón střední C v trvání jedné sekundy DE obsahuje hodnotu #0105 což je INT (261.63\*1) a HL obsahuje +066A což je odvozeno od 6689/4-30,125.

```
03B5 BEEPER      DI                Zákaz přerušení během trvání BEEPu.
                 LD      A,L          Přechodná úschova L.
                 SRL     L             Každá "1" v registru L má počítat 4 T stavy, ale provedením
                 SRL     L             INT(L/4) dochází k čítání 16 T stavu.
                 CPL                Vrací původní hodnotu do L a zjišťuje zbytek po dělení
                 AND     #03
                 LD      C,A
                 LD      B, #00
                 LD      IX,#03D1     Bázová adresa časové smyčky
                 ADD     IX,BC        Oprava délky časové smyčky za každou "1" zlomkové části.
                 LD      A,(#5C48) (BORDER) Vyzvedni současnou barvu pro BORDER a
                 AND     #38          ulož ji do bitů 0,1 a 2 registru A.
                 RRCA
                 RRCA
                 RRCA
                 OR      #08          Výstup MIC je vypnut.
```

Zde se vstupuje do smyčky, která produkuje vlastní zvuk. V DE je počet kompletních průchodů. HL obsahuje "délku časové smyčky".

```
03D1 BE-IX+3     NOP                Přičíte 4 T stavy za
03D2 BE-IX+2     NOP                každý dřívější vstupní
03D3 BE-IX+1     NOP                bod, který je použit.
03D4 BE-IX+0     INC     B           Hodnoty v BC přijdou
                                     z HL viz. dále.
03D6 BE-H&L-LP  DEC     C           časová smyčka;
                                     JR      NZ,#03D6,BE-H&L-LP tedy BC*4 T stavy.
                                     LD      C,#3F (Při polovičním cyklu
                                     DEC     B se bude C rovnat L+1)
                                     JP      NZ,#03D6,BE-H&L-LP
```

Reproduktor je nyní střídavě zapínán a vypínán.

```
XOR     #10      Otoč bit 4.
OUT     (#FE),A  Vykonej instrukci OUT při zachování barvy BORDERu.
LD      B,H      Resetuj registr B.
LD      C,A      Uschovej registr A.
```

```

BIT 4,A
JR NZ,#03F2,BE-AGAIN Skok, jestliže jsi v bodě polovičního cyklu.

```

Po úplném cyklu se testuje registrový pár DE.

```

LD A,D
OR E
JR Z,#03D6,BE-END Skok zpět, jestliže již byl vykonán poslední průchod
LD A,C Obnovení uschované hodnoty.
LD C,L Resetuj registr C.
DEC DE Zmenšení čítače průchodů smyčky.
JP (IX) Skok zpět na počátek smyčky.

```

Parametry pro druhý poloviční cyklus jsou nastaveny.

```

03F2 BE-AGAIN LD C,L Resetuj registr C.
INC C Přičti 16 T stavů, protože tato cesta je kratší.
JP (IX) Skok zpět na počátek smyčky.

```

Maskovatelné přerušení je obnoveno po ukončení BEEPu.

```

03F6 BE-END EI Obnovení přerušení.
RET Závěrečný návrat.

```

#### PODPROGRAM PŘÍKAZU BEEP

Před vstupem do tohoto podprogramu musí být na vrcholu zásobníku kalkulátoru uložena výška tónu P a pod ní délka trvání tónu t.

```

03F8 BEEP RST #28,FP-CALC Volání FP kalkulátoru.
DEFB #31,zdvojení t,P,P
DEFB #27,int t,P,i ( i = INT P )
DEFB #C0,st-mem-0 t,P,i ( i uloženo do mem-0 )
DEFB #03,rozdíl t,P ( p je zlomková část P )
DEFB #34,stk-data konstanta K na zásobník (K=0.057762606)
DEFB #EC,exponent+7C
DEFB #6C,#98,#1F,#F5
DEFB #04,násobení t,pK
DEFB #A1,stk-one t,pK,1
DEFB #0F,součet t,pK+1
DEFB #38,konec výpočtu

```

Nyní se provede řada testů i (celočíselná hodnota výšky tónu u).

```

LD HL,#5C92 Toto je mem-0-1 (MEMBOT).
LD A,(HL) Vyzvedni exponent i
AND A
JR NZ,#046C,REPORT-B a vyvolej chybu, pokud se nejedná o "krátkou formu".
INC HL Kopíruj znaménkový
LD C,(HL) bajt do registru C.
INC HL Kopíruj nižší bajt
LD B,(HL) do registru B a
LD A,B do registru A.
RLA
SBC A,A
CP C
JR NZ,#046C,REPORT-B Vyvolej chybu pokud neplatí, že -128 <= i <= +127.
INC HL

```

CP	(HL)	
JR	NZ,#046C,REPORT-B	
LD	A,B	Dále testuj nižší bajt.
ADD	A,#3C	
JP	P,#0425,BE-i-OK	Akceptuj pouze -60 <= i <= +67.
JP	PO,#046C,REPORT-B	Odmítní hodnoty od -128 do -61.

Poznámka: Rozsah +70 až +127 bude zamítnut později. Nyní může být nalezena správná frekvence pro výšku i.

0425	BE-i-OK	LD	B,#FA	Začni "6" oktáv pod středním C.
0427	BE-OCTAVE	INC	B	Opakovaně snižuj i,
		SUB	#0C	aby byla nalezena
		JR	NC,#0427,BE-OCTAVE	správná oktáva.
		ADD	A,#0C	Přičti zpět poslední odečet.
		PUSH	BC	Uchovej číslo oktávy.
		LD	HL,#046E	Bázová adresa tabulky půl tónů.
		CALL	#3406,LOC-MEM	Prohledej tabulku a nalezenou FP hodnotu
		CALL	#33B4,STACK-NUM	přidej na zásobník kalkulátoru.

Nyní se vezme v úvahu zlomková část výšky.

RST	#28,FP-CALC	t,pK+1,C
DEFB	#04,násobení	t,C(pK+1)
DEFB	#38,konec výpočtu	

Konečná frekvence f je nalezena modifikací poslední hodnoty podle čísla oktávy.

POP	AF	Vyzvedni číslo oktávy.
ADD	A,(HL)	a vynásob "poslední hodnotu"*2*x.
LD	(HL),A	(x=číslo oktávy)
RST	#28,FP-CALC	t,f
DEFB	#C0,st-mem-0	frekvence provizorně
DEFB	#02,výmaz	uložena v mem-0.

Výpočet trvání tónu.

DEFB	#31,zdvojení	t,t
DEFB	#38,konec výpočtu	
CALL	#1E94,FIND-INT1	Hodnota INT t musí být v rozsahu #00 až #0A.
CP	#0B	
JR	NC,#046C,REPORT-B	

Počet kompletních cyklů pro BEEP je dán součinem f\*t.

RST	#28,FP-CALC	t
DEFB	#E0,get-mem-0	t,f
DEFB	#04,násobení	f*t
DEFB	#E0,get-mem-0	f*t,f
DEFB	#34,stk-data	Na vrcholu zásobníku se vytvoří kons. KK=3.5*10 <sup>6</sup> /8.
DEFB	#80,4 bajty	(KK=437500)
DEFB	#43,exponent #93	
DEFB	#55,#9F,#80,#00	f*t,f,KK
DEFB	#01,záměna	f*t,KK,f
DEFB	#05,dělení	f*t,KK/f
DEFB	#34,stk-data	
DEFB	#35,exponent #85	
DEFB	#71,#00,#00,#00	f*t,KK/f,30.125

DEFB #03,odčítání f\*t,KK/f,-30.125  
 DEFB #38,konec výpočtu

Poznámka: Hodnota 437500/f dává délku polovičního cyklu. Její snížení o 30.125 poskytne 120.5T stavů, během kterých dojde k produkci noty, nastavení čítačů a podobně.

Nyní se přenesou hodnoty do patřičných registrů.

CALL #1E99,FIND-INT2 Hodnota časové smyčky je převedena do BC  
 PUSH BC a je uschována.

Poznámka: Jestliže je tato hodnota příliš vysoká dojde k vyvolání chybového podprogramu což vyloučí hodnoty od +70 do +127.

CALL #1E99,FIND-INT2 Hodnota f\*t je přenesena do BC,  
 POP HL časová smyčka do HL.  
 LD D,B  
 LD E,C f\*t do DE

Nyní poslední test hodnoty f\*t.

LD A,D  
 OR E  
 RET Z Návrát, pokud nejsou požadovány žádné cykly.  
 DEC DE Zmenšení čítače průchodů smyčkou a  
 JP #03B5,BEEPER skok do podprogramu BEEPER.

046C REPORT-B RST #08,ERROR-1 Ohlaš:  
 DEFB #0A B-Integer out of range

#### TABULKA PŮLTÓNŮ

Tato tabulka obsahuje frekvence dvanácti půltónů v oktávě.

		frekvence	tón
046E	DEFB #89,#02,#D0,#12,#86	261.63	C
	DEFB #89,#0A,#97,#60,#75	277.18	C#
	DEFB #89,#12,#D5,#17,#1F	293.66	D
	DEFB #89,#1B,#90,#41,#02	311.12	D#
	DEFB #89,#24,#D0,#53,#CA	329.63	E
	DEFB #89,#2E,#9D,#36,#B1	349.23	F
	DEFB #89,#38,#FF,#49,#3E	369.99	F#
	DEFB #89,#43,#FF,#6A,#73	392	G
	DEFB #89,#4F,#A7,#00,#54	415.30	G#
	DEFB #89,#5C,#00,#00,#00	440	A
	DEFB #89,#69,#14,#F6,#24	466.16	A#
	DEFB #89,#76,#F1,#10,#05	493.88	H

#### "NÁZEV PROGRAMU" (ZX81)

Následující podprogram je pro ZX81 a nebyl odstraněn když se program přepisoval pro SPECTRUM.

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

## PODPROGRAMY SAVE - LOAD - VERIFY

Vstupní bod pro tyto podprogramy je na adrese #0605 SAVE-ETC. (Niméně skutečné podprogramy pro SAVE, LOAD nebo VERIFY bloků bajtů začínají od #04C2). Ve všech případech obsahuje DE počet bajtů, v IX je adresa umístění začátku bloku dat v paměti. Registr A obsahuje #00 pro hlavičku a #FF pro vlastní blok dat. Rychlost přenosu je 1200 bit/sec.

### PODPROGRAM SA-BYTES

Je volán pro SAVE hlavičky (z #09BA) a později pro vlastní blok dat (z #099E).

04C2	SA-BYTES	LD HL,#053F	Uložení adresy SA/LD-RET
		PUSH HL	do zásobníku.
		LD HL,#1F80	Konstanta 5 sec. pro zaváděcí signál hlavičky.
		BIT 7,A	
04D0	SA-FLAG	JR Z,#04D0,SA-FLAG	Skok, když se zaznamenává hlavička.
		LD HL,#0C98	Pouze 2 sec. pro datový blok.
		EX AF,AF'	Uložení podmínkového registru.
		INC DE	Délka o jedničku zvětšena.
		DEC IX	Bázová adresa snížena.
		DI	Přerušení zakázáno po dobu SAVE.
		LD A,#02	Maska pro MIC,BORDER červený.
		LD B,A	Přenos hodnoty 2 do B.

Nyní následuje podprogram pro tvorbu pulsů zaváděcího signálu. Pulsy MIC - on a MIC - off jsou dlouhé 2.168 T (taktů hodin). Barva BORDERu se mění v témže rytmu z červené do cyanu (při nástupné a sestupné hraně signálu).

04D8	SA-LEADER	DJNZ #04D8,SA-LEADER	Hlavní časová perioda.
		OUT (#FE),A	MIC on-off, BORDER červený - cyan
		XOR #0F	v každém průchodu.
		LD B,#A4	Hlavní časová konstanta.
		DEC L	Snížení nižšího bajtu čítače.
		JR NZ,#04D8,SA-LEADER	Skok pro další puls.
		DEC B	Další průchod, redukce o 13 T stavů.
		DEC H	Snížení vyššího bajtu čítače.
		JP P,#04D8,SA-LEADER	Zpět pro další puls dokud není zaváděcí část kompletní.

Dále je vyslán synchronizační puls (sync):

04EA	SA-SYNC-1	LD B,#2F	
		DJNZ #04EA,SA-SYNC-1	MIC off: 667 T od OUT k OUT.
		OUT (#FE),A	MIC on a červená.
		LD A,#0D	Signál MIC off a cyan.
04F2	SA-SYNC-2	LD B,#37	MIC on: 735 T od OUT k OUT
		DJNZ #04F2,SA-SYNC-2	
		OUT (#FE),A	MIC off a BORDER cyan.

Bajt indikace bude zaznamenán jako první.

LD BC,#3B0E	3B je časovací konstanta, #0E = MIC off a žlutou.
EX AF,AF'	Indikační bajt do
LD L,A	reg. A a jeho převod do L s odesláním.
JP #0507,SA-START	Skok do záznamové smyčky.

Je zaznamenán indikační bajt, dál už následují bajty dat. Poslední je paritní bajt, který je konstruován průběžně postupným porovnáváním hodnot všech zaznamenávaných bajtů (pro kontrolu správnosti záznamu při verifikaci nebo načítání).

04FE	SA-LOOP	LD A,D OR E JR Z,#050E,SA-PARITY LD L,(IX+00)	Čítač délky dat je testován a když dosáhne 0, provede se skok. Do L další bajt pro záznam.
0505	SA-LOOP-P	LD A,H XOR L	H je momentální paritní bajt. Přidej nový bajt záznamu.
0507	SA-START	LD H,A LD A,#01 SCF JP #0525,SA-8-BITS	Nový paritní bajt do H. Signál MIC on a modrá. CY jako návěští pro 8 bitů zaznamenávaného bajtu. Skok do tvorby bajtu a jeho zaznamenání na pásek.

Před posláním paritního bajtu k nahrání je bajt převeden do L:

050E	SA-PARITY	LD L,H JR #0505,SA-LOOP-P	Konečná podoba paritního bajtu. Skok zpět.
------	-----------	------------------------------	---

Následující podprogram tvaruje jednotlivé bity nahrávaného bajtu tak, že každý obsahuje on i off puls, přičemž pulsy pro bity log.1 jsou přesně dvakrát delší než pulsy bitu log.0 (o 855 T).

0511	SA-BIT-2	LD A,C BIT 7,B	Druhý průchod; MIC off a žlutá. Indikátor nuly je 1 (při 2. průchodu).
0514	SA-BIT-1	DJNZ #0514,SA-BIT-1 JR NC,#051C,SA-OUT LD B,#42	Hlavní časovací smyčka, 801 T při 2. průchodu. Skok na kratší průchod při tvorbě log.0. Při tvorbě log.1 o 855 T víc.
051A	SA-SET	DJNZ #051A,SA-SET	
051C	SA-OUT	OUT (#FE),A LD B,#3E JR NZ,#0511,SA-BIT-2 DEC B XOR A INC A	Při 1. průchodu MIC on. Při 2. průchodu MIC off a žlutá. Čas. konstanta pro 2. průchod. Zpět na konec 1. průchodu, jinak puls 13 T. CY nastaven na 0. Registr A je 1; MIC on a modrá.

Smyčka 8 bitů: Reg. L obsahuje nahrávaný bajt a bit přenosu, CY je nastaven na jedna (jako návěští) z podprogramu SA-START. Každý bit je odeslán na výstup a CY nulován. Po osmi průchodech podprogramem bude registr L vynulován a program pokračuje dál.

0525	SA-8-BITS	RL L JP NZ,#0514,SA-BIT-1 DEC DE INC IX LD B,#31 LD A,#7F IN A,(#FE) RRA RET NC LD A,D INC A JP NZ,#04FE,SA-LOOP LD B,#3B	Bit 7 do CY a návěští do nulového bitu. Opakování do vynulování registru L. Sniž čítač bajtů. Posuň básovou adresu o 1 dopředu. Nastav časovou konstantu pro první bit dalšího bajtu. Adresa brány pro klávesu BREAK a její otestování. Návrat (do SA/LD RET) v případě, že je BREAK. Jinak test čítače bajtů a skok zpět i když je čítač nulový.
053C	SA-DELAY	DJNZ #053C,SA-DELAY RET	Návrat jestliže je čítač bajtů nastaven na #FFFF.

Poznámka: Vynulovaný bit způsobí puls MIC off následovaný pulsem MIC on, oba v délce 855T stavů. Jedničkový bit dává pulsy 2 krát delší. Pověšměte si, že mezi synchronizačním pulsem, indikačním bajtem a dalšími bajty nejsou žádné mezery.

### PODPROGRAM SA/LD-ROUTINE

Je společný pro SAVE i LOAD. BORDER je nastaven na původní barvu a naposledy je testováno tlačítko BREAK.

053F	SA/LD-RET	PUSH AF	Uložení CY (CY je resetováno při chybném LOADu).
		LD A,(#5C4B) (BORDCR)	Vyzvednutí barvy ze systémové proměnné BORDCR.
		AND #38	
		RRCA	
		RRCA	
		RRCA	Přenos barvy BORDER do bitů 2,1,0.
		OUT (#FE),A	Nastavení BORDERu na původní barvu.
		LD A,#7F	Test tlačítka BREAK
		IN A,(#FE)	
		RRA	
		EI	Povolení maskovaného přerušení.
		JR C,#0554,SA/LD-END	Není-li BREAK, skok.
0552	REPORT-D	RST #08,ERROR-1	Ohlaš:
		DEFB #0C	D-BREAK-CONT repeats
0554	SA/LD-END	POP AF	Obnovení původního stavu CY.
		RET	Návrat.

### PODPROGRAM LD-BYTES

Tento podprogram je volán jako funkce LOAD nebo VERIFY hlavičky (z #076E) nebo dat (z #0802).

0556	LD-BYTES	INC D	Z flag je vynulován (neboť D nemůže obsahovat #FF).
		EX AF,AF'	A=#00 - hlavička, A=#FF pro blok. CY=0 - VERIFY.
		DEC D	CY=1 - LOAD. Registr D zpět na původní hodnotu.
		DI	Zákaz přerušení.
		LD A,#0F	Bílý BORDER.
		OUT (#FE),A	
		LD HL,#053F	Adresa SA/LD-RET
		PUSH HL	na zásobník.
		IN A,(#FE)	Test brány #FE.
		RRA	Rotace načteného
		AND #20	bajtu, ale zvážení jen bitu EAR.
		OR #02	Signál BORDER červený je uložen i do
		LD C,A	registru C. (#22 pro OFF a #02 pro ON stav vstupu EAR)
		CP A	Nulový indikátor je nastaven na 1.

Prvním úkolem při čtení dat z pásku je zjistit, zda existuje nějaký pulsní signál (tedy hrany on-off a off-on).

056B	LD-BREAK	RET NZ	Návrat při BREAKu.
056C	LD-START	CALL #05E7,LD-EDGE-1	Není-li přítomen signál během 1400 T
		JR NC,#056B,LD-BREAK	stavů, návrat s CY=1. Jinak je BORDER nastaven na cyan.

Dále se čeká a zjišťuje, zda je signál stále přítomen.

0574	LD-WAIT	LD HL,#0415	Délka čekání je téměř 1 sekunda.
		DJNZ #0574,LD-WAIT	
		DEC HL	
		LD A,H	
		OR L	
		JR NZ,#0574,LD-WAIT	Čekací smyčka.
		CALL #05E3,LD-EDGE-2	Pokračuj při zachycení dvou po sobě
		JR NC,#056B,LD-BREAK	jdoucích hran v dané periodě.



Nyní bude přijat jen zaváděcí signál.

0580	LD-LEADER	LD	B,#9C	Časovací konstanta,
		CALL	#05E3,LD-EDGE-2	Pokračuj při zachycení dvou po sobě
		JR	NC,#056B,LD-BREAK	jdoucích hran v dané periodě.
		LD	A,#C6	Tyto hrany musí být zachyceny během
		CP	B	3000 T.
		JR	NC,#056C,LD-START	
		INC	H	Počet párů hran je ukládán do reg. H
		JR	NZ,#0580,LD-LEADER	dokud jich není 256.

Po zaváděcím signálu přicházejí části off a on pulsu sync.

058F	LD-SYNC	LD	B,#C9	Časovací konstanta.
		CALL	#05E7,LD-EDGE-1	Každá hrana je testována,
		JR	NC,#056B,LD-BREAK	dokud nejsou nalezeny dvě hrany blízko sebe.
		LD	A,B	(startovací puls sync).
		CP	#D4	
		JR	NC,#058F,LD-SYNC	
		CALL	#05E7,LD-EDGE-1	Na konci musí být ještě konečná hrana části on
		RET	NC	synchronizačního pulsu sync.

Teď už se mohou načítat bajty hlavičky nebo programu (bloku dat) v operacích LOAD, VERIFY.  
První bajt určuje typ.

		LD	A,C	BORDER na modrou a žlutou.
		XOR	#03	
		LD	C,A	
		LD	H,#00	Inicializace bajtu parity na 0.
		LD	B,#B0	Časovací konstanta pro bajt určující typ.
		JR	#05C8,LD-MARKER	Skok do smyčky čtení bajtu.

Smyčka čtení bajtu se používá k načtení vždy jednoho bajtu. První je typový, následován datovými a na závěr bajt paritní.

05A9	LD-LOOP	EX	AF,AF'	Vyzvednutí indikátorů.
		JR	NZ,#05B3,LD-FLAG	Skok pro první (typový) bajt.
		JR	NC,#05BD,LD-VERIFY	Skok při VERIFY nahrávky.
		LD	(IX+00),L	Uložení načteného bajtu na správnou adresu.
		JR	#05C2,LD-NEXT	Skok pro načtení dalšího bajtu.
05B3	LD-FLAG	RL	C	Dočasné uložení CY.
		XOR	L	Návrat když se typový bajt liší od typového bajtu
		RET	NZ	z pásky; CY=0.
		LD	A,C	CY je obnoveno na původní hodnotu.
		RRA		
		LD	C,A	
		INC	DE	Čítač je zvětšen, aby bylo kompenzováno jeho
		JR	#05C4,LD-DEC	zmenšení po odskoku.

Při VERIFYkaci je nově načtený bajt porovnán s původním:

05BD	LD-VERIFY	LD	A,(IX+00)	Zjištění hodnoty původního bajtu a
		XOR	L	porovnání s právě načteným bajtem.
		RET	NZ	Návrat, jestliže se oba bajty nerovnejí.

Nový bajt bude načten po jednotlivých bitech.

05C2	LD-NEXT	INC	IX	Zvýšení adresy pro uložení dalšího bajtu.
05C4	LD-DEC	DEC	DE	Snížení čítače délky bloku.
		EX	AF,AF'	Uložení indikátorů.
		LD	B,#B2	Časovací konstanta.
05C8	LD-MARKER	LD	L,#01	Uložení značkového bitu.

Tato smyčka sestavuje načítaný bajt do registru L.

05CA	LD-8-BITS	CALL	#05E3,LD-EDGE-2	Nalezení délky pulsů jednotlivých bitů.
		RET	NC	Návrat při nesprávné (větší) délce pulsu, (pak CY=0)
		LD	A,#CB	Porovnání délky oproti asi 2400 T,
		CP	B	kdy pro nulový bit je CY=0 a pro jedničkový bit je CY=1.
		RL	L	Uložení nového bitu do registru L.
		LD	B,#B0	Časová konstanta pro další bit.
		JP	NC,#05CA,LD-8-BITS	Nejednalo-li se o poslední (osmý)bit skok zpět do smyčky.
		LD	A,H	Výzvední paritní bajt z reg.H.
		XOR	L	a přičte j nový bajt.
		LD	H,A	Výsledek zpět do H.

Průchody se opakují do vynulování čítače DE, pak musí být i paritní bajt nulový.

LD	A,D	
OR	E	
JR	NZ,#05A9,LD-L00P	Je-li DE nenulový, skok zpět pro další bajt.
LD	A,H	
CP	#01	Test paritního bajtu.
RET		Je-li paritní bajt nulový - návrat s CY=1, jinak CY=0.

#### PODPROGRAMY LD-EDGE-2 A LD-EDGE-1

Tyto dva podprogramy jsou nejdůležitější částí operací LOAD a VERIFY. Vstupuje se do nich s časovou konstantou v registru B a barvou BORDERu i "typem hrany" v registru C.

Návrat z těchto podprogramů je s CY=1, když požadovaný počet hran byl nalezen v povolené periodě - pak změní v registru B u CY=0 při chybě. Z flag=0 pokud byla stisknuta klávesa BREAK. Zflag=1 znamená "bez nálezů" a provede se návrat. LD-EDGE-2 slouží pro nalezení délky kompletního pulsu. LD-EDGE-1 slouží k určení času, který uplyne do nalezení hrany.

05E3	LD-EDGE-2	CALL	#05E7,LD-EDGE-1	Zde se v podstatě volá ještě jednou LD-EDGE-1 a návrat pokud došlo k chybě.
		RET	NC	
05E7	LD-EDGE-1	LD	A,#16	Čekání 358 T před vstupem do
05E9	LD-DELAY	DEC	A	vzorkovací smyčky.
		JR	NZ,#05E9,LD-DELAY	
		AND	A	

Následuje vzorkovací smyčka. Obsah registru B je zvýšen při každém průchodu. Návrat "bez nálezů" při dosažení 0 v reg. B.

05ED	LD-SAMPLE	INC	B	Čítání průchodů.
		RET	Z	CY=0 & Z=1 "bez nálezů".
		LD	A,#7F	Čtení z brány #7FFE (BREAK a EAR).
		IN	A,(#FE)	
		RRA		
		RET	NC	CY=0 & Z=0, když byla stisknuta klávesa BREAK.
		XOR	C	Test bajtu s posledním typem hrany.
		AND	#20	
		JR	Z,#05ED,LD-SAMPLE	Skok zpět, když je beze změny.

Byla nalezena nová 'hrana' ve stanovené době k hledání. Takže následuje změna barvy a nastavení

CY flagu.

LD	A,C	Změna posledního "typu hrany"
CPL		a barvy BORDERu.
LD	C,A	
AND	#07	Bere se jen barva BORDERu.
OR	#08	Signál MIC-off.
OUT	(#FE),A	Změna barvy BORDERu červená/fialová nebo modrá/žlutá.
SCF		Signalizace úspěšného nalezení
RET		před návratem.

Podprogram LD-EDGE-1 trvá 465T, plus 58T z každého průchodu vzorkovací smyčkou při neúspěšném hledání. Například při očekávání sync pulsu (viz. LD-SYNC na #058F) je povoleno 10 průchodů vzorkovací smyčkou. Hledání hrany probíhá během cca 1100T (465+10+58+průchody). Tak je zajištěno zachycení části off pulsu sync, který přichází po dlouhých pulsech zaváděcího signálu.

#### PODPROGRAMY PŘÍKAZU SAVE, LOAD, VERIFY, MERGE

Užívá se jako vstupní bod pro všechny čtyři povely. Hodnota v T-ADDR se u jednotlivých funkcí liší. V první části podprogramu probíhá konstrukce hlavičky v pracovní oblasti.

0605	SAVE-ETC	POP AF	Zničení návratové adresy SCAN-LOOP.
		LD A,(#5C74) (T-ADDR-lo)	Redukce T-ADDR-lo
		SUB #E0	o #E0; pak je 00 pro SAVE, 01 pro
		LD (#5C74),A (T-ADDR-lo)	LOAD, 02 pro VERIFY 03 pro MERGE.
		CALL #1C8C,EXPT-EXP	Uložení parametrů na zásobník kalkulátoru.
		CALL #2530,SYNTAX-Z	Skok při testování syntaxe.
		JR Z,#0652,SA-DATA	
		LD BC,#0011	17 adres pro hlavičku SAVE,
		LD A,(#5C74) (T-ADDR-lo)	ale 34 pro ostatní povely.
		AND A	
		JR Z,#0621,SA-SPACE	
		LD C,#22	
0621	SA-SPACE	RST #30,BC-SPACES	Vytvoří se požadovaný prostor v pracovní oblasti.
		PUSH DE	
		POP IX	Předání parametrů z DE do IX.
		LD B,#0B	Název programu může mít až 10 znaků, ale
		LD A,#20	nejdříve se uloží
0629	SA-BLANK	LD (DE),A	11 mezer na připravené místo.
		INC DE	
		DJNZ #0629,SA-BLANK	
		LD (IX+1),#FF	Nulové jméno jen při #FF.
		CALL #2BF1,STK-FETCH	Parametry jména jsou vyvolány a je testována jeho délka.
		LD HL,#FFF6	Tato hodnota je přímo -10 (doplněk).
		DEC BC	Není-li jméno delší než 10
		ADD HL,BC	pokračuje se dál.
		INC BC	
		JR NC,#064B,SA-NAME	
		LD A,(#5C74) (T-ADDR-lo)	Ale LOAD, VERIFY, MERGE jsou umožněny
		AND A	i při nulových nebo velmi dlouhých názvech.
		JR NZ,#0644,SA-NUL	
0642	REPORT-F	RST #08,ERROR-1	Ohlaš:
		DEFB #0E	F-Invalid file name

Pokračování práce se jménem programu:

0644	SA-NULL	LD A,B	Skok dál, když je
		OR C	délka názvu nulová.
		JR Z,#0652,SA-DATA	
		LD BC,#000A	Useknutí přebytečných znaků.

Nyní je název přenesen do WORKSPACE (od jeho druhé adresy):

064B	SA-NAME	PUSH IX	Přesun počáteční
		POP HL	adresy do HL.
		INC HL	Posun na druhou adresu.
		EX DE,HL	Výměna ukazatelů a
		LDIR	přenos jména.

Dále jsou zvažovány některé parametry připojené za povel (pokud existují).

0652	SA-DATA	RST #18,GET-CHAR	Načtení znaku.
		CP #E4	Jedná se o znak "DATA" ?
		JR NZ,#06A0,SA-SCR\$	Skoč, když ne.
		LD A,(#5C74) (T-ADDR-lo)	
		CP #03	
		JP Z,#1C8A,REPORT-C	Není možný MERGE "název" DATA
		RST #20,NEXT-CHAR	Načtení dalšího znaku a zvýšení syst. proměnné CH-ADD.
		CALL #28B2,LOOK-VARS	Vyhledání řetězce v oblasti proměnných.
		SET 7,C	Nastavení bitu 7 v názvu pole.
		JR NC,#0672,SA-V-OLD	Skok dopředu, jedná-li se o existující pole.
		LD HL,#0000	Signál: užití nového pole.
		LD A,(#5C74) (T-ADDR-lo)	Porovnání obsahu T-ADDR a vyvolání
		DEC A	chybového hlášení,
		JR Z,#0685,SA-V-NEW	když bylo požadováno SAVE nebo VERIFY nového pole.

0670	REPORT-2	RST #08,ERROR-1	Ohlaš:
		DEFB #01	2-Variable not found

Pokračování v práci s existujícími poli:

0672	SA-V-OLD	JP NZ,#1C8A,REPORT-C	
		CALL #2530,SYNTAX-Z	Skok dopředu při
		JR Z,#0692,SA-DATA-1	kontrole syntaxe.
		INC HL	Nižší bajt délky
		LD A,(HL)	pole je přesouván
		LD (IX+11),A	do pracovní oblasti
		INC HL	a je následován
		LD A,(HL)	příslušným vyšším bajtem délky pole.
		LD (IX+12),A	
		INC HL	

Další část je společná pro "nová" i "stará" pole.

0685	SA-V-NEW	LD (IX+14),C	Kopie jména pole.
		LD A,#01	Je pole číselné?
		BIT 6,C	
		JR Z,#068F,SA-V-TYPE	Skok, když je číselné.
		INC A	Je znakové.
068F	SA-V-TYPE	LD (IX+0),A	Uložení typu na první adresu hlavičkové oblasti.

Test poslední části příkazu.

0692	SA-DATA-1	EX DE,HL	Uložení ukazatele do DE.
------	-----------	----------	--------------------------

RST	#20,NEXT-CHAR	Načtení dalšího znaku a zvýšení syst. proměnné CH-ADD.
CP	#29	Je konec závorky ")" ?
JR	NZ,#0672,SA-V-OLD	Hlášení "REPORT-C" když ne.
RST	#20,NEXT-CHAR	Načtení dalšího znaku a zvýšení syst. proměnné CH-ADD.
CALL	#1BEE,CHECK-END	Pokračuj na další část příkazu při zjišťování syntaxe.
EX	DE,HL	Ukazatel zpět do HL (indikuje začátek obsahu existujícího pole).
JP	#075A,SA-ALL	

Test na token SCREEN\$.

06A0	SA-SCR\$	CP	#AA	Není-li token
		JR	NZ,#06C3,SA-CODE	SCREEN\$, skok dopředu.
		LD	A,(#5C74) (T-ADDR-lo)	
		CP	#03	
		JP	Z,#1C8A,REPORT-C	Je zakázáno MERGE "název" SCREEN\$
		RST	#20,NEXT-CHAR	Vyzvednutí následujícího znaku a zvýšení CH-ADD.
		CALL	#1BEE,CHECK-END	Posun na další část příkazu při zjišťování syntaxe.
		LD	(IX+11),#00	Do hlavičky je ukládán počet nahrazených bajtů, což
		LD	(IX+12),#1B	je pro obrazovou paměť právě #1B00.
		LD	HL,#4000	Adresa 1. bajtu obrazové paměti je právě #4000.
		LD	(IX+13),L	A toto číslo je také uloženo mezi
		LD	(IX+14),H	hlavičkové údaje.
		JR	#0710,SA-TYPE-3	

Test na token CODE.

06C3	SA-CODE	CP	#AF	Není-li token CODE,
		JR	NZ,#0716,SA-LINE	skok dopředu.
		LD	A,(#5C74) (T-ADDR-lo)	
		CP	#03	
		JP	Z,#1C8A,REPORT-C	Není dovoleno MERGE "název" CODE
		RST	#20,NEXT-CHAR	Vyzvednutí následujícího znaku a zvýšení CH-ADD.
		CALL	#2048,PR-ST-END	Skok, když příkaz pokračuje.
		JR	NZ,#06E1,SA-CODE-1	
		LD	A,(#5C74) (T-ADDR-lo)	
		AND	A	
		JP	Z,#1C8A,REPORT-C	Není povoleno SAVE "název" CODE bez uvedení parametrů.
		CALL	#1CE6,USE-ZERO	Uložení 0 na zásobník kalkulátoru pro startovací adresu.
		JR	#06F0,SA-CODE-2	

Hledání startovací adresy.

06E1	SA-CODE-1	CALL	#1C82,EXPT-1NUM	Vyzvedni 1. číslo za CODE.
		RST	#18,GET-CHAR	Následuje-li
		CP	#2C	další číslo
		JR	Z,#06F5,SA-CODE-3	skok dopředu, první číslo bylo startovací adresou.
		LD	A,(#5C74) (T-ADDR-lo)	
		AND	A	
		JP	Z,#1C8A,REPORT-C	Není povoleno SAVE CODE bez parametrů, startovací adresy a délky.
06F0	SA-CODE-2	CALL	#1CE6,USE-ZERO	Uložení nuly do kalkulátorového zásobníku pro počet bajtů.
		JR	#06F9,SA-CODE-4	

Zjištění počtu bajtů.

06F5	SA-CODE-3	RST	#20,NEXT-CHAR	Vyzvednutí následujícího znaku a zvýšení CH-ADD.
		CALL	#1C82,EXPT-1NUM	Zjištění počtu bajtů.

**Parametry jsou uloženy do hlavičkové části pracovní oblasti.**

06F9	SA-CODE-4	CALL #1BEE,CHECK-END	Posun na další část příkazu při zjišťování syntaxe.
		CALL #1E99,FIND-INT2	Přenos délky do BC.
		LD (IX+11),C	
		LD (IX+12),B	a její uložení do hlavičky.
		CALL #1E99,FIND-INT2	Přenos startovací adresy do BC
		LD (IX+13),C	
		LD (IX+14),B	a její uložení do hlavičky.
		LD H,B	Přenos ukazatele do
		LD L,C	HL jako obvykle.

SCREEN\$ a CODE jsou oba typu 3.

0710	SA-TYPE-3	LD (IX+0),#03	Nyní vlož číslo pro typ bloku dat.
		JR #075A,SA-ALL	Vrať se na dráhu.

Zjištění přítomnosti LINE a dalších možných parametrů.

0716	SA-LINE	CP #CA	Je-li token LINE,
		JR Z,#0723,SA-LINE-1	skok dopředu.
		CALL #1BEE,CHECK-END	Posun na další část povelu při zjišťování syntaxe.
		LD (IX+14),#80	Už nejsou další parametry.
		JR #073A,SA-TYPE-0	

Zjištění čísla, které musí následovat po výrazu LINE.

0723	SA-LINE-1	LD A,(#5C74) {T-ADDR-lo}	
		AND A	
		JP NZ,#1C8A,REPORT-C	Nelze provádět SAVE "název" LINE bez udání čísla.
		RST #20,NEXT-CHAR	Vyzvednutí následujícího znaku a zvýšení CH-ADD.
		CALL #1C82,EXPT-1NUM	Uložení čísla do kalkulátorového zásobníku.
		CALL #1BEE,CHECK-END	Posun na další část povelu při zjišťování syntaxe.
		CALL #1E99,FIND-INT2	Přenos čísla řádku autostartu do BC
		LD (IX+13),C	
		LD (IX+14),B	a jeho uložení.

LINE a žádný parametr jsou oba typu 0.

073A	SA-TYPE-0	LD (IX+00),#00	Uložení čísla typu.
------	-----------	----------------	---------------------

Parametry určující program a jeho proměnné jsou nalezeny a uloženy do hlavičkové části pracovní oblasti.

LD HL,(#5C59) {E-LINE}	Ukazatel konce oblasti proměnných.
LD DE,(#5C53) {PROG}	Ukazatel počátku basicového programu.
SCF	Odečet pro určení
SBC HL,DE	délky programu +
LD (IX+11),L	proměnných a její uložení.
LD (IX+12),H	
LD HL,(#5C4B) {VARS}	Opakování odečtu,
SBC HL,DE	ale jen pro zjištění délky samotného programu.
LD (IX+15),L	
LD (IX+16),H	
EX DE,HL	Přesun ukazatele do HL.

Ve všech případech byly připraveny hlavičkové informace.

IX+0           Obsahuje číslo typu dat.  
 IX+1 až IX+10 Obsahuje jméno (#FF v IX+1 znamená data beze jména).  
 IX+11 a IX+12 Obsahuje počet bajtů bloku dat.  
 IX+13 až IX+16 Obsahuje různé parametry, jejichž obsah záleží na typovém čísle.

Nyní se separuje jedna z operací SAVE, LOAD, VERIFY nebo MERGE.

```
075A SA-ALL       LD   A,(#5C74) (T-ADDR-lo)
                  AND  A                    Je-li povel SAVE
                  JP   Z,#0970,SA-CONTRL   skoč dopředu
```

V případě zbylých tří operací je prvních 17 bajtů hlavičkové části pracovní oblastí vybaveno informacemi, jak uvedeno výše. Teď je čas pro odebrání hlavičky z pásky.

```
                  PUSH HL                  Uložení ukazatele adresy určení.
                  LD   BC,#0011           Formování 1. adresy
                  ADD  IX,BC              2.hlavičkové oblasti do IX.
```

Smyčka pro načtení hlavičky.

```
0767 LD-L00K-H   PUSH IX                  Uložení 1. adresy 2. hlavičkové oblasti.
                  LD   DE,#0011          LOAD 17ti bajtů.
                  XOR  A                  Signál: hlavička.
                  SCF                    Signál: LOAD.
                  CALL #0556,LD-BYTES   Hledání hlavičky.
                  POP  IX                Obnovení 1. adresy 2.hlavičkové oblasti.
                  JR   NC,#0767,LD-L00K-H Skoky dokud neskončeno.
```

Nová hlavička se vypíše na obrazovce, ale podprogram pokračuje jen tehdy, když se "nový" typ neliší od "starého":

```
                  LD   A,#FE             Ujištění, že kanál
                  CALL #1601,CHAN-OPEN   S je otevřen.
                  LD   (IY+82),#03 {SCR-CT} Nastavení čítače skrolování.
                  LD   C,#80             Signál: typy se neliší.
                  LD   A,(IX+0)          Porovnání "nového"
                  CP   (IX-17)          typu se "starým".
                  JR   NZ,#078A LD-TYPE   Liší-li se, skoč.
                  LD   C,#F6             Když OK., tak dej signál: 10 znaků a porovnej je.
078A LD-TYPE     CP   #04                Je-li číslo "typu 4" a více, je
                  JR   NC,#0767 LD-L00K-H hlavička nesmyslná a odkakuje se.
```

Objeví se jedno z hlášení: Program:, Number array:, Character array:, Bytes:

```
                  LD   DE,#09C0          První adresa bloku uvedených hlášení.
                  PUSH BC                Uložení registru C, dokud není zobrazeno.
                  CALL #0COA,PO-MSG      Patříčné hlášení
                  POP  BC                se zobrazí.
```

Zobrazí se nové jméno a potom se toto porovná se starým.

```
                  PUSH IX                Registrový pár DE bude
                  POP  DE                ukazovat na nový typ a
                  LD   HL,#FFFF0        registrový pár HL na
                  ADD  HL,DE            staré jméno.
                  LD   B,#0A            10 znaků jména.
                  LD   A,(HL)          Znak z názvu
```

INC	A	je to #FF ?
JR	NZ,#07A6,LD-NAME	ne - porovnávat dál
LD	A,C	Pokud je staré jméno prázdné
ADD	A,B	indikovat signál
LD	C,A	všechny znaky souhlasí.

Následující smyčka tiskne znaky nového jména. Pokud toto jméno souhlasí, je při výstupu ze smyčky čítač nulový.

07A6	LD-NAME	INC DE	Znak nového
		LD A,(DE)	jména
		CP (HL)	porovnat se starým
		INC HL	
		JR NZ,#07AD,LD-CH-PR	nečítat pokud písmeno nesouhlasí.
		INC C	
07AD	LD-CH-PR	RST #10,PRINT-A-1	Tisknout znak jména.
		DJNZ #07A6,LD-NAME	Opakovat pro všech deset znaků.
		BIT 7,C	Jméno souhlasilo pokud je
		JR NZ,#0767,LD-LOOK-H	čítač nulový.
		LD A,#OD	Nové jméno ukončit
		RST #10,PRINT-A-1	znakem CR.

Hlavička souboru odpovídá požadované hlavičce. Dále se rozlišuje, zda se bude provádět LOAD, VERIFY nebo MERGE.

POP	HL	Obnovit ukazatel.
LD	A,(IX+0)	SCREEN\$ a CODE jsou
CP	#03	ošetřeny podprogramem pro VERIFY.
JR	Z,#07CB,VR-CONTRL	
LD	A,(#5C74) (T-ADDR-lo)	Je požadována funkce
DEC	A	LOAD ?
JP	Z,#0808,LD-CONTRL	
CP	#02	Je požadována funkce MERGE ?
JP	Z,#08B6,ME-CONTRL	Pokud ne zbývá pouze VERIFY.

#### PODPROGRAM ŘÍZENÍ \*VERIFY\*

Proces ověřování probíhá podobně jako čtení, s tím rozdílem, že data se neukládají, ale pouze ověřují. Tento podprogram je také používán při čtení bloku dat.

07CB	VR-CONTRL	PUSH HL	Ulož ukazatel.
		LD L,(IX-06)	Vyzvedni počet bajtů ze staré hlavičky.
		LD H,(IX-05)	
		LD E,(IX+11)	Počet bajtů z nové hlavičky.
		LD D,(IX+12)	
		LD A,H	Skoč dopředu v případě, že délka není určena.
		OR L	
		JR Z,#07E9,VR-CONT-1	T.j. příkaz 'LOAD jméno CODE'.
		SBC HL,DE	Vypiš chybu R v případě, že blok dat
		JR C,#0806,REPORT-R	je větší než je požadováno.
		JR Z,#07E9,VR-CONT-1	Délky jsou shodné - v pořádku.
		LD A,(IX+00)	Chybové hlášení R se také vypíše, pokud se pokusíme
		CP #03	ověřovat bloky s nestejnou délkou.
		JR NZ,#0806,REPORT-R	
07E9	VR-CONT-1	POP HL	Obnovení ukazatele (kam se bude ukládat).
		LD A,H	Tento ukazatel se použije, pokud
		OR L	není roven nule.
		JR NZ,#07F4,VR-CONT-2	V tom případě se totiž místo něho použije



LD	L,(IX+13)	hodnota z nové hlavičky.
LD	H,(IX+14)	

Nyní se podle příznaku VERIFY/LOAD provede buď načtení nebo ověření.

07F4	VR-CONT-2	PUSH HL	Přesuň pointer do registrového páru IX.
		POP IX	
		LD A,(#5C74) (T-ADDR-lo)	Nastavit příznak pro zpracování bloku dat.
		CP #02	A to buď příznak
		SCF	LOAD, nebo příznak
		JR NZ,#0800,VR-CONT-3	VERIFY.
		AND A	
0800	VR-CONT-3	LD A,#FF	Příznak pro zpracování pouze datového bloku.

#### PODPROGRAM PRO PŘEČTENÍ BLOKU DAT

Tento podprogram je společný pro všechny čtecí podprogramy.

0802	LD-BLOCK	CALL #0556,LD-BYTES	LOAD/VERIFY bloku dat.
		RET C	Pokud není chyba - návrat.

Chybové hlášení R - Tape loading error.

0806	REPORT-R	RST #08,ERROR-1	Zavolej hlášení chyby.
		DEFB #1A	

#### PODPROGRAM PRO ŘÍZENÍ PŘÍKAZU LOAD

Tento podprogram řídí čtení basicovského programu včetně proměnných.

0808	LD-CONTRL	LD E,(IX+11)	Vyzvedni délku bloku
		LD D,(IX+12)	z nové hlavičky.
		PUSH HL	Uložit cílový ukazatel.
		LD A,H	Skoč dopředu v případě pokusu o načtení
		OR L	dosud nedeklarovaného pole.
		JR NZ,#0819,LD-CONT-1	
		INC DE	Přidej tři bajty pro novou proměnnou.
		INC DE	Místo pro jméno
		INC DE	a délku nové proměnné.
		EX DE,HL	
		JR #0825,LD-CONT-2	<b>Skoč dopředu.</b>

Nyní je tedy v paměti dost místa pro blok dat.

0819	LD-CONT-1	LD L,(IX-06)	Vezmi délku existujícího programu
		LD H,(IX-05)	včetně proměnných.
		EX DE,HL	
		SCF	Skoč dopředu pokud není vyžadována
		SBC HL,DE	žádná paměť navíc.
		JR C,#082E,LD-DATA	

Otestuj volné místo.

0825	LD-CONT-2	LD DE,#0005	5 bajtů navíc.
		ADD HL,DE	
		LD B,H	Výsledek do BC.
		LD C,L	
		CALL #1F05,TEST-ROOM	Otestuj.

**082E LD-DATA**

POP HL		Vyzvedni ukazatel.
LD A,(IX+0)		Skoč dopředu pokud se čte
AND A		basicovský program.
JR Z,#0873,LD-PROG		
LD A,H		Skoč dopředu pokud se čte
OR L		nové pole.
JR Z,#084C,LD-DATA-1		
DEC HL		Vezmi délku existujícího pole
LD B,(HL)		
DEC HL		
LD C,(HL)		
DEC HL		Ukazatel posuň na jméno.
INC BC		Přičti tři bajty pro délku a název.
INC BC		
INC BC		
LD (#5C5F),IX {X-PTR}		Ulož registr IX po dobu
CALL #19E8,RECLAIM-2		rušení staré proměnné.
LD IX,(#5C5F) {X-PTR}		

Nyní je vyhrazeno místo pro novou proměnnou - na konci platné oblasti pro data.

084C LD-DATA-1	LD HL,(#5C59) {E-LINE}	Nalezení konce oblasti proměnných,
	DEC HL	který je označen bajtem #80.
	LD C,(IX+11)	Do BC je uložena
	LD B,(IX+12)	délka nového pole,
	PUSH BC	která je okamžitě uschována.
	INC BC	Nyní se přičtou tři
	INC BC	bajty pro jméno a
	INC BC	délku pole.
	LD A,(IX-3)	Je zjištěn název pole a okamžitě
	PUSH AF	uschován.
	CALL #1655,MAKE-ROOM	Je vytvořen prostor v počtu bajtů udaných registry BC.
	INC HL	HL nyní ukazuje na 1. bajt pole, kam
	POP AF	bude po zjištění
	LD (HL),A	uloženo jeho jméno.
	POP DE	Opět je vyzvednuta délka pole
	INC HL	
	LD (HL),E	
	INC HL	
	LD (HL),D	a po bajtech uložena.
	INC HL	HL nyní ukazuje na adresu, kam bude ukládán již samotný
	PUSH HL	obsah pole. Tato adresa je zároveň okamžitě přenesena
	POP IX	do IX.
	SCF	Signál: LOAD.
	LD A,#FF	Signál: blok dat.
	JP #0802,LD-BLOCK	Je proveden odskok na LOADovací podprogram.

LOAD BASICového programu a jeho proměnných:

0873 LD-PROG	EX DE,HL	Uschování 1. adresy programu do DE.
	LD HL,(#5C59) {E-LINE}	Nalezení konce oblasti proměnných,
	DEC HL	který je označen bajtem #80.
	LD (#5C5F),IX {X-PTR}	Uschování IX.
	LD C,(IX+11)	Do BC je uložena
	LD B,(IX+12)	délka programu a
	PUSH BC	okamžitě uschována.
	CALL #19E5,RECLAIM-1	Podprogram zničí starý existující BASICový program.
	POP BC	Délka nového programu je opět uložena do BC.

PUSH HL		Jsou uschovány ukazatele programové
PUSH BC		oblasti a délky programu.
CALL #1655,MAKE-ROOM		Vytvoření nového místa pro program a jeho proměnné.
LD IX,(#5C5F) {X-PTR}		V IX je obnovena jeho původní hodnota.
INC HL		
LD C,(IX+15)		
LD B,(IX+16)		
ADD HL,BC		Systémová proměnná VARS je nastavena na novou hodnotu,
LD (#5C4B),HL {VARS}		kteřá již odpovídá parametrům nového programu.
LD H,(IX+14)		Je testováno,
LD A,H		zda byl přítomen
AND #C0		příkaz LINE.
JR NZ,#08AD,LD-PROG-1		Odskok jestliže se nejedná o autostart.
LD L,(IX+13)		Jinak jsou nastaveny systémové
LD (#5C42),HL {NEWPPC}		proměnné NEWPPC a
LD (IY+10),#00 {NSPPC}		NSPPC.

Nyní může být načten blok dat:

08AD	LD-PROG-1	POP DE	Je vyzvednuta délka
		POP IX	a adresa 1.bajtu programu.
		SCF	Signál: LOAD.
		LD A,#FF	Signál: blok dat.
		JP #0802,LD-BLOCK	<b>A nahra j ho.</b>

#### PODPROGRAM **RÍZENÍ MERGE**

Tento podprogram se skládá ze tří částí:

- LOAD bloku dat do oblasti WORKSPACE.
- MERGE řádků nového programu mezi řádky starého programu.
- MERGE nových proměnných do starých proměnných.

Začíná se s načtením bloku dat.

08B6	ME-CONTRL	LD C,(IX+11)	Načti délku bloku
		LD B,(IX+12)	do BC
		PUSH BC	a uschovej ji.
		INC BC	Vytvoř prostor o délce LEN+1
		RST #30,BC-SPACES	v pracovním prostoru
		LD (HL),#80	a do bajtu "navíc" vlož koncový znak.
		EX DE,HL	Přesun ukazatele startu do HL.
		POP DE	Vyzvedni původní délku.
		PUSH HL	Uschovej kopii startu.
		PUSH HL	A převeď start do IX.
		POP IX	
		SCF	Signál: LOAD.
		LD A,#FF	Signál: blok dat.
		CALL #0802,LD-BLOCK	Převeď LOAD bloku dat.

Nyní dojde k splnutí řádků nového a starého programu.

POP HL		Vyzvedni start nového programu.
LD DE,(#5C53) {PROG}		Vyzvedni start starého programu.

Vstup do smyčky, která zpracovává řádky nového programu.

08D2	ME-NEW-LP	LD A,(HL)	Vyzvedni číslo řádku
------	-----------	-----------	----------------------

AND #CO a testuj je.  
 JR NZ,#08FO,ME-VAR-LP Skok, když jsou řádky skončeny.

Vstup do **vnitřní** smyčky, která zpracovává řádky starého programu.

08D7	ME-OLD-LP	LD A,(DE)	Vyzvedni vyšší bajt čísla řádku
		INC DE	a porovnej jej
		CP (HL)	s novým
		INC HL	inkrementuj ukazatele.
		JR NZ,#08DF,ME-OLD-L1	Skok dopředu, není-li shoda.
		LD A,(DE)	Opakuj porovnáni
		CP (HL)	také pro nižší bajty čísla řádku.
08DF	ME-OLD-L1	DEC DE	Obnov oba ukazatele.
		DEC HL	
		JR NC,#08EB,ME-NEW-L2	Skok při nalezení správného místa pro řádek nového programu.
		PUSH HL	Jinak najdi adresu začátku dalšího
		EX DE,HL	starého
		CALL #19B8,NEXT-ONE	řádku.
		POP HL	
		JR #08D7,ME-OLD-LP	Procházej smyčkou pro každý starý řádek.
08EB	ME-NEW-L2	CALL #092C,ME-ENTER	Vlož nový řádek a
		JR #08D2,ME-NEW-LP	skoč zpět do hlavní smyčky.

Obdobným způsobem budou nyní splývat nové a staré proměnné.

<b>08FO</b>	<b>ME-VAR-LP</b>	LD A,(HL)	Vyzvedni postupně
		LD C,A	jména všech proměnných a testuj je.
		CP #80	Test na konec a
		RET Z	návrat, když byly všechny proměnné posouzeny.
		PUSH HL	Uschovej ukazatel.
		LD HL,(#5C4B) {VARS}	Vyzvedni VARS (pro starý program).

Nyní vstup do smyčky, kde se naleznou všechny existující proměnné.

08F9	ME-OLD-VP	LD A,(HL)	Vyzvedni jméno proměnné.
		CP #80	Jakmile je nalezen koncový bajt,
		JR Z,#0923,ME-VAR-L2	proved skok dopředu (bude se přidávat).
		CP C	Porovnej první bajt
		JR Z,#0909,ME-OLD-V2	názvu a skoč dopředu pro další posouzení.
0901	ME-OLD-V1	PUSH BC	Uschovej název nové proměnné a hledej
		CALL #19B8,NEXT-ONE	další starou proměnnou.
		POP BC	Obnov ukazatele.
		EX DE,HL	
		JR #08F9,ME-OLD-VP	a pokračuj ve smyčce.

Stará a nová proměnná se shodují podle prvních bajtů, ale proměnné s dlouhými názvy je třeba porovnat celé.

0909	ME-OLD-V2	AND #E0	V úvahu se vezmou pouze bity 7,6 a 5.
		CP #A0	Akceptuj všechny typy proměnných,
		JR NZ,#0921,ME-VAR-L1	kromě proměnných s dlouhým názvem.
		POP DE	DE bude ukazovat na první znak názvu nové proměnné
		PUSH DE	a bude uschován.
		PUSH HL	Úschova ukazatele na první znak názvu staré proměnné.

Vstup do smyčky k porovnání všech znaků názvu **dlouhé proměnné**.

0912	ME-OLD-V3	INC HL	Posuň
		INC DE	oba ukazatele.
		LD A,(DE)	
		CP (HL)	Porovnej oba znaky
		JR NZ,#091E,ME-OLD-V4	a skoč, jestliže se neshodují.
		RLA	Dokud není nalezen poslední znak,
		JR NC,#0912,ME-OLD-V3	pokračuj ve smyčce.
		POP HL	Vyzvedni ukazatel na první znak názvu staré proměnné
		JR #0921,ME-VAR-L1	a skoč s úspěšným nálezem dopředu.
091E	ME-OLD-V4	POP HL	Vyzvedni ukazatel na první znak názvu staré proměnné
		JR #0901,ME-OLD-V1	a skoč s neúspěšným nálezem zpět.

Zde je vstup pro shodu.

0921	ME-VAR-L1	LD A,#FF	Signál: náhrada proměnné.
------	-----------	----------	---------------------------

Zde je vstup pro neshodu. A obsahuje #80 což je signál, že proměnná má být přidána.

0923	ME-VAR-L2	POP DE	Vyzvedni ukazatel na první znak názvu nové proměnné.
		EX DE,HL	Zaměň registry.
		INC A	Z flag=1 jde-li o náhradu a Z flag=0 jde-li o přidání.
		SCF	Signál: proměnná.
		CALL #092C,ME-ENTER	Nyní proved vložení.
		JR #08F0,ME-VAR-LP	Pokračuj ve smyčce a posuzuj další proměnnou.

#### MERGE řádky nebo proměnné

Do tohoto podprogramu se vstupuje s následujícími parametry:

CY=0	- MERGE basicového řádku.	CY=1	- MERGE proměnné.
Z flag=0	- Bude se přidávat.	Z flag=1	- Bude se nahrazovat.
HL	ukazuje na start nového vstupu.	DE	ukazuje kam se provede MERGE.

092C	ME-ENTER	JR NZ,#093E,ME-ENT-1	Skoč, jde-li o "přídavek".
		EX AF,AF'	Uchověj příznaky.
		LD (#5C5F),HL {X-PTR}	Uchověj "nový"
		EX DE,HL	ukazatel,
		CALL #19B8,NEXT-ONE	zatímco bude provedena
		CALL #19EB,RECLAIM-2	likvidace řádku nebo proměnné.
		EX DE,HL	<b>Obnov</b>
		LD HL,(#5C5F) {X-PTR}	všechny ukazatele
		EX AF,AF'	a příznaky.

Nyní může dojít k vložení "nového".

093E	ME-ENT-1	EX AF,AF'	Uchověj příznaky.
		PUSH DE	Uchověj cílový ukazatel.
		CALL #19B8,NEXT-ONE	Nalezni délku "nového".
		LD (#5C5F),HL {X-PTR}	Uchověj ukazatel na "nové".
		LD HL,(#5C53) {PROG}	Vyzvedni PROG aby se předešlo poškození.
		EX (SP),HL	Uchověj PROG na zásob. a současně vyzvedni ukaz. nov.
		PUSH BC	Uchověj délku.
		EX AF,AF'	Obnov příznaky.
		JR C,#0955,ME-ENT-2	Skoč, jestliže přidáváš novou proměnnou.
		DEC HL	Nový řádek je vložen před cílovou lokací.
		CALL #1655,MAKE-ROOM	Vytvoř prostor pro nový řádek.
		INC HL	
		JR #0958,ME-ENT-3	Skoč dopředu.

0955	ME-ENT-2	CALL #1655,MAKE-ROOM	Vytvoř prostor pro novou proměnnou.
0958	ME-ENT-3	INC HL	Ukazuj na první novou lokaci.
		POP BC	Obnov délku.
		POP DE	Vyzvedni PROG
		LD (#5C53),DE {PROG}	a ulož na patřičné místo.
		LD DE,(#5C5F) {X-PTR}	Také vyzvedni "nový" ukazatel
		PUSH BC	a opět ulož délku
		PUSH DE	a "nový" ukazatel.
		EX DE,HL	Zaměň ukazatele
		LDIR	a kopíruj "nové" do prostoru, který byl vytvořen.

Nová proměnná nebo řádek musí být nyní odstraněny z pracovního prostoru.

POP HL	Vyzvedni ukazatel na "nové".
POP BC	Vyzvedni délku.
PUSH DE	Ušchvej ukazatel na "staré".
CALL #19EB,RECLAIM-2	Odstraň řádek nebo proměnnou z pracovního prostoru.
POP DE	Vyzvedni ukazatel na "staré"
RET	a vrať se.

#### PODPROGRAM ŘÍZENÍ SAVE

Podprogram SAVE je velmi **pochopitelný**.

0970	SA-CONTRL	PUSH HL	Ušchvej si ukazatel.
		LD A,#FD	Signál: kanál K.
		CALL #1601,CHAN-OPEN	Otevři kanál K.
		XOR A	Signál 1. hlášení.
		LD DE,#09A1	Báze tabulky kazetových hlášení.
		CALL #0C0A,PO-MSG	Tiskni hlášení: Start tape, then press any key.
		SET 5,(IY+2) (TV-FLAG)	Signál: obrazovka bude vyžadovat smazání.
		CALL #15D4,WAIT-KEY	Čekej na stlačení klávesy.

Po stisknutí tlačítka se začne nahrávat hlavička.

PUSH IX	Ušchvej si adresu hlavičky.
LD DE,#0011	Počítadlo 17ti bajtů, které budou nahrány.
XOR A	Signál: hlavička.
CALL #04C2,SA-BYTES	Nahrání hlavičky.
POP IX	Obnovení adresy hlavičky

Následuje pauza asi jednu sec. před nahráním dat.

		LD B,#32	
0991	SA-1-SEC	HALT	
		DJNZ #0991,SA-1-SEC	Pauza 50ti přerušení.
		LD E,(IX+11)	Do DE je vložena
		LD D,(IX+12)	délka zaznamenaného bloku dat.
		LD A,#FF	Signál: blok dat.
		POP IX	Obnovení ukazatele 1. adresy bloku
		JP #04C2,SA-BYTES	a konečně se všechno nahrává!

## TEXTY POUŽÍVANÉ PŘI PRÁCI S KAZETOU

Každé hlášení je zakončeno invertovaným znakem, tzn.: bit 7 tohoto znaku je logická 1. Bajt #0D (carriage return) způsobí tisk od levého okraje na dalším řádku.

```
09A1      DEFB #80
09A2      DEFM 'Start tape, then press any key.'
09C1      DEFM #0D,'Program:'
09CB      DEFM #0D,'Number array:'
09DA      DEFM #0D,'Character array:'
09EC      DEFM #0D,'Bytes:'
```

## PODPROGRAMY OBSLUHUJÍCÍ TISK NA OBRAZOVKU A TISKÁRNU

### PODPROGRAM PRINT-OUT

Veškerý tisk na hlavní část obrazovky, dolní část obrazovky a na tiskárnu obstarávají následující podprogramy. Do podprogramu PRINT-OUT vstupuje kód znaku, kontrolního znaku, nebo tokenu v registru A.

09F4	PRINT-OUT	CALL #0B03,PO-FETCH	Aktuální tisková pozice.
		CP #20	Jestliže se jedná o znak který lze tisknout,
		JR NC,#0AD9,PO-ABLE	provede se skok.
		CP #06	Tiskni otazník pro
		JR C,#0A69,PO-QUEST	hodnoty #00 až #05 a pro
		CP #18	hodnoty #18 až #1F.
		JR NC,#0A69,PO-QUEST	
		LD HL,#0A0B	Báze "řídící" tabulky.
		LD E,A	Převod kód
		LD D,#00	do registrového páru DE.
		ADD HL,DE	Indexuj v tabulce a
		LD E,(HL)	vyzvedni doplněk.
		ADD HL,DE	Přičti doplněk
		PUSH HL	a proved nepřímý skok
		JP #0B03,PO-FETCH	do příslušného podprogramu.

### TABULKA ŘÍDÍCÍCH ZNAKŮ

adresa	doplněk	znak	adresa	doplněk	znak
0A11	4E	PRINT	0A1A	4F	nevyužito
0A12	57	EDIT	0A1B	5F	INK
0A13	10	kurzor vlevo	0A1C	5E	PAPER
0A14	29	kurzor vpravo	0A1D	5D	FLASH
0A15	54	kurzor dolů	0A1E	5C	BRIGHT
0A16	53	kurzor nahoru	0A1F	5B	INVERSE
0A17	52	DELETE	0A20	5A	OVER
0A18	37	ENTER	0A21	54	AT
0A19	50	nevyužito	0A22	53	TAB

### PODPROGRAM KURZOR VLEVO

Na vstupu obsahuje registr B číslo aktuálního řádku a registr C číslo aktuálního sloupce.

0A23	PO-BACK-1	INC C	Pohyb doleva o jeden sloupec.
		LD A,#22	Akceptuj tuto změnu
		CP C	pokud nepřevyší max. hodnotu pro levou stranu.
		JR NZ,#0A3A,PO-BACK-3	
		BIT 1,(IY+1) (FLAGS)	Při obsluze tiskárny
		JR NZ,#0A3B,PO-BACK-2	skok dopředu.
		INC B	Jeden řádek nahoru.
		LD C,#02	Nastav sloupcovou hodnotu.
		LD A,#18	Testuj na nejvyšší řádek.
		CP B	Poznámka: správně mělo být #19.
		JR NZ,#0A3A,PO-BACK-3	Akceptuj změnu, pokud není vrchol obrazovky.
		DEC B	Nepřijatelné, takže o řádek dolů.
0A3B	PO-BACK-2	LD C,#21	Nastav na levou krajní pozici.
0A3A	PO-BACK-3	JP #0DD9,CL-SET	Proved nepřímý návrat přes CL-SET & PO-STORE.



#### PODPROGRAM KURZOR VPRAVO

Tento podprogram provede identickou operaci jako basicové PRINT OVER 1; CHR\$ 32;.

0A3D	PO-RIGHT	LD A,(#5C91) (P-FLAG)	Vyzvedni P-FLAG a
		PUSH AF	uschovej na zásobníku.
		LD (IY+87),#01 (P-FLAG)	Nastav P-FLAG na OVER 1.
		LD A,#20	Tiskni
		CALL #0B65,PO-CHAR	mezeru.
		POP AF	Obnov původní hodnotu
		LD (#5C91),A (P-FLAG)	pro P-FLAG.
		RET	Poznámka: Programátor zapomněl na návrat přes PO-STORE.

#### PODPROGRAM "NÁVRATU VOZÍKU"

Jestliže se jedná o výstup na tiskárnu, provede se vyprázdnění bafru. Při výstupu na obrazovku se bude testovat rolování obrazovky před snížením čísla řádku.

0A4F	PO-ENTER	BIT 1,(IY+1) (FLAGS)	Při obsluze tiskárny
		JP NZ,#0ECD,COPY-BUFF	skok dopředu.
		LD C,#21	Nastav levý krajní sloupec.
		CALL #0C55,PO-SCR	Je-li potřeba roluj obrazovku.
		DEC B	Nyní o řádek dolů.
		JP #0DD9,CL-SET	Nepřímý návrat přes CL-SET & PO-STORE.

#### PODPROGRAM TISKNI ČÁRKU

Současná sloupcová hodnota je **manipulována** a registr A je nastaven na #00 pro TAB 0, nebo #10 pro TAB 16.

0A5F	PO-COMMA	CALL #0B03,PO-FETCH	Proč znovu?
		LD A,C	Číslo sloupce.
		DEC A	
		DEC A	Posun doprava o dva sloupce a test.
		AND #10	V registru A bude nyní #00 nebo #10.
		JR #0AC3,PO-FILL	Výstup přes PO-FILL.

#### PODPROGRAM TISK OTAZNÍKU

Kdykoliv dojde k pokusu o tisk znaku, který tisknout nelze, vytiskne se místo něj otazník.

0A69	PO-QUEST	LD A,#3F	Kód pro otazník.
		JR #0AD9,PO-ABLE	Tisk znaku.

#### PODPROGRAM PRO TISK ŘÍDÍCÍCH ZNAKŮ S OPERANDY

Řídící znaky od INK do OVER vyžadují jeden operand, zatímco řídící znaky AT & TAB musí být následovány dvěma operandy. Tento podprogram v případě jednoho operandu uloží kód řídícího znaku do TVDATA-lo a operand je v registru A. V případě dvou operandů je první operand uložen do TVDATA-hi a druhý v registru A.

0A6D	PO-TV-2	LD DE,#0A87	Změna adresy pro "výstupní podprogram" na #0A87,PO-CONT.
		LD (#5C0F),A (TVDATA-hi)	Ulož první operand do TVDATA-hi.
		JR #0A80,PO-CHANGE	

Pro AT a TAB je vstupní bod zde.

0A75	PO-2-OPER	LD DE,#0A6D	Změna adresy pro "výstupní podprogram" na #0A6D,PO-TV-2.
		JR #0A7D,PO-TV-1	a ulož kód znaku do TVDATA-lo.

Vstupní bod pro řídicí znaky barev INK až OVER.

0A7A PO-1-OPER LD DE,#0A87 Změna adresy pro "výstupní podprogram" na #0A87,PO-CONT.  
0A7D PO-TV-1 LD (#5C0E),A (TVDATA-lo) Ulož kód řídicího znaku.

Adresa "výstupního" podprogramu je změněna přechodně.

0A80 PO-CHANGE LD HL,(#5C51) (CURCHL) HL ukazuje na adresu výstupního podprogramu.  
LD (HL),E Vložení této nové adresy způsobí, že další kód bude  
INC HL považován za operand.  
LD (HL),D  
RET

Jakmile jsou operandy uloženy, podprogram pokračuje.

0A87 PO-CONT LD DE,#09F4 Obnovení původní adresy pro PRINT-OUT (#09F4).  
CALL #0A80,PO-CHANGE Vyzvedni řídicí kód  
LD HL,(#5C0E) (TVDATA) a první operand,  
LD D,A (jestli budou dva operandy)  
LD A,L "poslední" operand a řídicí kód jsou přesunuty.  
CP #16 Jedná-li se o INK až OVER,  
JP C,#2211,CO-TEMPS pak skok dopředu.  
JR NZ,#0AC2,PO-TAB Skok dopředu jde-li o TAB.

Nyní zpracuj řídicí charakter AT.

LD B,H Číslo řádku.  
LD C,D Číslo sloupce.  
LD A,#1F Převrácení čísla sloupce t.j. #00 až  
SUB C #1F se změní na #1F až #00.  
JR C,#0AAC,PO-AT-ERR Musí být v rozsahu.  
ADD A,#02 Přičtení doplňku, aby C obsahoval #21 až #22.  
LD C,A  
BIT 1,(IY+1) (FLAGS) Při obsluze tiskárny  
JR NZ,#0ABF,PO-AT-SET skok dopředu.  
LD A,#16 Převrácení čísla řádku.  
SUB B t.j. #00 až #15 se změní na #16 až #01.  
0AAC PO-AT-ERR JP C,#1E9F,REPORT-B Je-li vše v pořádku skok dopředu.  
INC A Rozsah #16 až #01 se změní na #17 až #02  
LD B,A  
INC B A nyní na #18 až #03.  
BIT 0,(IY+2) (TV-FLAG) Jedná-li se o tisk v dolní části obrazovky, proved  
JP NZ,#0C55,PO-SCR případné rolování obrazovky.  
CP (IY+49) (DF-SZ)  
JP C,#0C86,REPORT-5 Vyvolej chybové hlášení při nedostatku místa.  
0ABF PO-AT-SET JP #0DD9,CL-SET Návrat přes CL-SET & PO-STORE.

Zde zpracuj řídicí znak TAB.

0AC2 PO-TAB LD A,H Vyzvedni první operand.  
0AC3 PO-FILL CALL #0B03,PO-FETCH Aktuální tisková pozice.  
ADD A,C Přičtení aktuální sloupcové hodnoty.  
DEC A Zjistí kolik mezer, modulo 32  
AND #1F je zapotřebí a vrať  
RET Z se, je-li výsledek nula.  
LD D,A Použij D jako čítač.  
SET 0,(FLAGS) Potlač "úvodní mezeru".

OADO	PO-SPACE	LD	A,#20	
		CALL	#0C3B,PO-SAVE	
		DEC	D	
		JR	NZ,#OADO,PO-SPACE	Vytiskni D * mezer.
		RET		Konec.

#### ZNAKY KTERÉ LZE TISKNOUT.

Požadovaný znak (nebo znaky) je vytisknut voláním PO-ANY následované podprogramem PO-STORE.

OAD9	PO-ABLE	CALL	#0B24,PO-ANY	Vytiskni znak(y) a pokračuj přes PO-STORE.
------	---------	------	--------------	--

#### PODPROGRAM ÚSCHOVA POZICE

Nové hodnoty pro řádek, sloupec a adresa "bodu" jsou uschovány v příslušných systémových proměnných.

OADC	PO-STORE	BIT	1,(IY+1) (FLAGS)	Skok dopředu při obsluze tiskárny.
		JR	NZ,#0AFC,PO-ST-PR	
		BIT	0,(IY+2) (TV-FLAG)	Skok při obsluze dolní části obrazovky.
		JR	NZ,#0AFO,PO-ST-E	
		LD	(#5C88),BC (S-POSN)	Uschovej hodnoty,
		LD	(#5C84),HL (DF-CC)	které se vztahují k hlavní části obrazovky
		RET		a vrať se.
OAF0	PO-ST-E	LD	(#5C8A),BC (S-POSNL)	
		LD	(#5C82),BC (ECHO-E)	Uschovej hodnoty,
		LD	(#5C86),HL (DF-CCL)	které se vztahují k <b>dolní části obrazovky</b> a
		RET		vrať se.
O AFC	PO-ST-PR	LD	(IY+69),C (P-POSN)	Uschovej hodnoty,
		LD	(#5C80),HL (PR-CC)	které se vztahují k <b>bafru tiskárny</b>
		RET		a vrať se.

#### PODPROGRAM VYZVEDNUTÍ POZICE

Parametry aktuální tiskové pozice jsou vyzvednuty z příslušných systémových proměnných.

OB03	PO-FETCH	BIT	1,(IY+1) (FLAGS)	Skok dopředu při obsluze tiskárny.
		JR	NZ,#0B1D,PO-F-PR	
		LD	BC,(#5C88) (S-POSN)	Vyzvedni hodnoty,
		LD	HL,(#5C84) (DF-CC)	které se vztahují k hlavní části
		BIT	0,(IY+2) (TV-FLAG)	obrazovky a vrať se,
		RET	Z	pokud toto bylo úmyslem.
		LD	BC,(#5C8A) (S-POSNL)	Jinak vyzvedni hodnoty, které se
		LD	HL,(#5C86) (DF-CCL)	vztahují k dolní části obrazovky a
		RET		vrať se.
OB1D	PO-F-PR	LD	C,(IY+69) (P-POSN)	Vyzvedni hodnoty,
		LD	HL,(#5C80) (PR-CC)	které se vztahují k bafru tiskárny a
		RET		vrať se.

#### PODPROGRAM TISKU JAKÉHOKOLIV ZNAKU

Obyčejné znaky, token, grafické znaky a UDG znaky jsou pojednány odděleně.

OB24	PO-ANY	CP	#80	Při obyčejném znaku
		JR	C,#0B65,PO-CHAR	skok dopředu.
		CP	#90	Při UDG znaku
		JR	NC,#0B52,PO-T&UDG	skok dopředu.
		LD	B,A	Přesun grafického znaku.
		CALL	#0B38,PO-GR-1	Vytvoření grafické formy.

CALL #0B03,PO-FETCH	Obnovení HL.
LD DE,#5C92 (MEMBOT)	DE ukazuje na začátek grafické formy.
JR #0B7F,PR-ALL	Skok dopředu a tisk grafického znaku.

Grafické znaky jsou konstruovány pouze pro tento případ v oblasti paměti pro kalkulátor t.j. MEM-0 až MEM-1.

OB38 PO-GR-1	LD HL,# 5C92 (MEMBOT)	
	CALL #0B3E,PO-GR-2	Ve skutečnosti volej následující podprogram dvakrát.
OB3E PO-GR-2	RR B	Určení bitu 0 (a později i bitu 2) grafického kódu.
	SBC A,A	A bude obsahovat #00, nebo
	AND #0F	#0F podle hodnoty bitu v kódu.
	LD C,A	Ušchvej výsledek v C.
	RR B	Určení bitu 1 (a později i bitu 3) grafického kódu.
	SBC A,A	A bude obsahovat #00, nebo
	AND #F0	#0F podle hodnoty bitu v kódu.
	OR C	Oba výsledky jsou kombinovány.
	LD C,#04	A obsahuje polovinu znakové formy
OB4C PO-GR-3	LD (HL),A	
	INC HL	a bude použito čtyřikrát.
	DEC C	
	JR NZ,#0B4C,PO-GR-3	Po horní polovině znakové formy přijde dolní polovina.
	RET	

Nyní se oddělí znaky UDG a tokens.

OB52 PO-T&UDG	SUB #A5	Skok dopředu při
	JR NC,#0B5F,PO-T	tokens.
	ADD A,#15	Znaky UDG jsou nyní od #00 do #0F.
	PUSH BC	Ušchvej hodnoty aktuální tiskové pozice.
	LD BC,(#5C7B) (UDG)	Vyzvedni báзовou adresu oblasti UDG
	JR #0B6A,PO-CHAR-2	a skoč dopředu.
OB5F PO-T	CALL #0C10,PO-TOKENS	Vytiskni token a
	JP #0B03,PO-FETCH	vrať se přes PO-FETCH.
OB65 PO-CHAR	PUSH BC	Ušchvej hodnoty současné pozice.
	LD BC,(#5C36) (CHARS)	Vyzvedni báзовou adresu oblasti CHARS
OB6A PO-CHAR-2	EX DE,HL	a uschvej tiskovou adresu v HL.
	LD HL,#5C3B (FLAGS)	Umožní
	RES 0,(HL)	úvodní mezeru.
	CP #20	Jestliže se nejedná o mezeru, potom
	JR NZ,#0B76,PO-CHAR-3	skoč dopředu.
	SET 0,(HL)	Ale potlač úvodní mezeru, jestliže už existuje.
OB76 PO-CHAR-3	LD H,#00	Převedení
	LD L,A	kódu znaku do HL.
	ADD HL,HL	
	ADD HL,HL	
	ADD HL,HL	HL=HL*8
	ADD HL,BC	Nalezení báзовé adresy znaku.
	POP BC	Je vyzvednuta aktuální tisková pozice
	EX DE,HL	a báзовá adresa převedena do DE.

#### PODPROGRAM TISK VŠECH ZNAKŮ

Tento podprogram tiskne osmibitové matice všech znaků. Na vstupu obsahuje DE báзовou adresu znakové formy, HL cílovou adresu a v BC je aktuální tisková pozice pro řádek a sloupec.

OB7F PR-ALL	LD A,C	Veźmi čísto sloupce
	DEC A	a posuň se o jednu pozici doprava.

	LD A,#21	Nejde-li o nový řádek
	JR NZ,#0893,PR-ALL-1	skoč dopředu.
	DEC B	Posuň o jednu pozici dolů.
	LD C,A	
	BIT 1,(IY+1) (FLAGS)	Při obsluze obrazovky
	JR Z,#0893,PR-ALL-1	skok dopředu.
	PUSH DE	Uchovej bázu adresu během
	CALL #0ECD,COPY-BUFF	vyprazdňování bafru
	POP DE	a obnov ji.
	LD A,C	Vezmi číslo nového sloupce.
OB93 PR-ALL-1	CP C	Testuj zda-li byl použit nový řádek.
	PUSH DE	
	CALL Z,#0C55,PO-SCR	Jestliže ano, podívej se jestli není
	POP DE	potřeba rolovat displej.

Nyní posuď současný stav INVERSE & OVER.

	PUSH BC	Uchovej poziční
	PUSH HL	a cílové hodnoty.
	LD A,(#5C91) (P-FLAG)	Vyzvedni P-FLAG.
	LD B,#FF	Připrav masku #FF pro OVER 1.
	RRA	Testuj bit 0
	JR C,#0BA4,PR-ALL-2	a proved skok při OVER 1.
	INC B	Uprav masku na #00 pro OVER 0.
OBA4 PR-ALL-2	RRA	Rotuj bit 2 <b>P-FLAG</b> do CY
	RRA	a připrav masku #00 pro INVERSE 0
	SBC A,A	a masku #FF pro INVERSE 1.
	LD C,A	Ulož masku do registru C.
	LD A,#08	Nastav registr A jako čítač mikrořádků.
	AND A	Nuluj CY flag.
	BIT 1,(IY+1) (FLAGS)	Při obsluze obrazovky
	JR Z,#0BB6,PR-ALL-3	skoč dopředu.
	SET 1,(IY+4B) (FLAGS2)	Signál: bafr již není prázdný.
	SCF	Nastav CY jako signál: používá se tiskárna.
OB66 PR-ALL-3	EX DE,HL	Před vstupem do smyčky zaměň bázu a cílovou adresu.

Zde bude během osmi průchodů smyčkou proveden tisk znaku.

	EX AF,AF'	Uchovej CY flag v registru F'.
	LD A,(DE)	Vyzvedni stávající mikrořádek.
	AND B	Použij masku pro OVER a výsledek
	XOR (HL)	XORuj s mikrořádkem.
	XOR C	Proveď XOR také s maskou pro INVERSE
	LD (DE),A	a vlož na cílovou adresu.
	EX AF,AF'	Vyzvedni CY flag
	JR C,#0BD3,PR-ALL-6	a skoč, je-li nastaven pro tiskárnu.
	INC D	Uprav cílovou adresu.
OBC1 PR-ALL-5	INC HL	Posuň se na další řádek zdrojové matice.
	DEC A	Zmenš čítač
	JR NZ,#0BB7,PR-ALL-4	a skoč zpět, pokud není nulový.
	EX DE,HL	HL musí obsahovat
	DEC H	správnou vyšší adresu znakové oblasti.
	BIT 1,(IY+1) (FLAGS)	Při obsluze obrazovky
	CALL Z,#0BDB,PO-ATTR	nastav příslušný bajt atributů.
	POP HL	Obnov původní cílovou
	POP BC	a poziční adresu.
	DEC C	Snížení čísla sloupce.
	INC HL	Zvýšení cílové adresy.



RET

Návrat.

Při obsluze tiskárny musí být cílová adresa zvětšována po přírůstcích #20.

OBD3	PR-ALL-6	EX	AF,AF'	Opět uschovej CY flag.
		LD	A,#20	Přírůstek.
		ADD	A,E	Přičtení
		LD	E,A	do registru E.
		EX	AF,AF'	Vyzvednutí CY flagu.
		JR	#0BC1,PR-ALL-5	Skok zpět do smyčky.

#### PODPROGRAM NASTAVENÍ ATRIBUTU

Je nalezen příslušný bajt. Starý obsah je modifikován podle stavu ATTR-T, MASK-T a P-FLAG a tato nová hodnota je vložena zpět.

OBDB	PO-ATTR	LD	A,H	Vyšší bajt
		RRCA		cílové adresy
		RRCA		je dělen osmi.
		RRCA		
		AND	#03	AND #03 určí, která třetina obrazovky je adresována.
		OR	#58	Nyní je <b>formován</b>
		LD	H,A	vyšší bajt adresy atributu.
		LD	(#5C8F) (ATTR-T)	D obsahuje ATTR-T a E obsahuje MASK-T.
		LD	A,(HL)	Hodnota "starého" atributu do A.
		XOR	E	<b>Maskování.</b>
		AND	D	
		XOR	E	
		BIT	6,(IY+87) (P-FLAG)	Pokud není PAPER 9,
		JR	Z,#0BFA,PO-ATTR-1	skoč dopředu.
		AND	#C7	Původní barva PAPER se ignoruje a podle
		BIT	2,A	odstínu INK bude nová barva PAPER
		JR	NZ,#0BFA,PO-ATTR-1	buď černá (000),
		XOR	#38	nebo bílá (111).
OBFA	PO-ATTR-1	BIT	4,(IY+87) (P-FLAG)	Pokud není INK 9,
		JR	Z,#0C08,PO-ATTR-2	skoč dopředu.
		AND	#F8	Původní barva INK se ignoruje a podle
		BIT	5,A	odstínu PAPER bude nová barva INK
		JR	NZ,#0C08,PO-ATTR-2	buď černá (000),
		XOR	#07	nebo bílá (111).
OC08	PO-ATTR-2	LD	(HL),A	Vlož novou hodnotu atributu
		RET		a vrať se.

#### PODPROGRAM TISKU HLÁŠENÍ

Tento podprogram se používá pro tisk hlášení a tokens. Na vstupu obsahuje registr A číslo hlášení nebo token v tabulce a v DE je bazová adresa této tabulky.

OC0A	PO-MSG	PUSH	HL	Vyšší bajt poslední úložky na zásobníku
		LD	H,#00	je vynulován pro potlačení koncových mezer
		EX	(SP),HL	viz.dále v programu.
		JR	#0C14,PO-TABLE	Skok dopředu.

Zde je vstupní bod pro rozvinutí tokens. (#00 až #5A = RND až COPY).

OC10	PO-TOKENS	LD	DE,#0095	Bázová adresa pro tabulku tokens.
		PUSH	AF	Uschovej kód na zásobníku.

Tabulka je prohledána a správný vstup vytisknut.

0C14	PO-TABLE	CALL #0C41,PO-SEARCH	Nalezení požadované položky.
		JR C,#0C22,PO-EACH	Skok na tisk, není-li požadována
		LD A,#20	úvodní mezera.
		BIT 0,(IY+1) (FLAGS)	
		CALL Z,#0C3B,PO-SAVE	

Jednotlivé znaky hlášení nebo tokens jsou postupně vytisknuty.

0C22	PO-EACH	LD A,(DE)	Vyzvedni kód a
		AND #7F	zruš "invertovaný" bit.
		CALL #0C3B,PO-SAVE	Tiskni znak.
		LD A,(DE)	Opět vyzvedni kód
		INC DE	a posuň ukazatel dopředu.
		ADD A,A	Invertovaný bit jde do CY jako signál posledního znaku.
		JR NC,#0C22,PO-EACH	Skok, pokud nejsi u konce hlášení nebo token.

Nyní se posoudí potřeba závěrečné mezery.

		POP DE	Pro hlášení obsahuje D #00 a pro tokens #00 až #5A.
		CP #48	Jestliže poslední byl znak \$,
		JR Z,#0C35,PO-TRSP	pak skok dopředu.
		CP #82	Jestliže poslední byl kterýkoliv znak před invertovaným
		RET C	"A", pak se vrať.
0C35	PO-TRSP	LD A,D	Prozkoumej hodnotu v D a jestliže
		CP #03	indikuje, že bylo tisknuto hlášení,RND,INKEY\$ nebo PI,
		RET C	vrať se.
		LD A,#20	Všechny ostatní případy vyžadují následnou mezera.

#### PODPROGRAM PO-SAVE

Tento podprogram umožňuje tzv. rekurzivní tisk (tím, že volá sám sebe). Obsahy registrů jsou uschovány.

0C3B	PO-SAVE	PUSH DE	Uschovej DE.
		EXX	Uschovej HL a BC.
		RST #10,PRINT-A-1	Vytiskni jeden znak.
		EXX	Obnovení HL a BC.
		POP DE	Obnovení DE.
		RET	Hotovo.

#### PODPROGRAM PROHLEDÁVÁNÍ TABULEK

Tento podprogram vrací DE jako ukazatel na první znak požadované položky a CY=0, má-li být posouzena úvodní mezera.

0C41	PO-SEARCH	PUSH AF	Uschovej číslo položky.
		EX DE,HL	Bázová adresa do HL.
		INC A	Uprav rozsah na #0 až #?
0C44	PO-STEP	BIT 7,(HL)	Čkej na invertovaný znak
		INC HL	a pokud není,
		JR Z,#0C44,PO-STEP	vrať se do smyčky.
		DEC A	Počítej položky,
		JR NZ,#0C44,PO-STEP	dokud nenalezneš tu požadovanou.
		EX DE,HL	DE ukazuje na první znak.
		POP AF	Vyzvedni číslo položky a pro prvních
		CP #20	32 čísel

RET C	se vracěj s CY=1.
LD A,(DE)	Testuj první znak
SUB #41	a je-li to písmeno
RET	bude zapotřebí úvodní mezera.

### PODPROGRAM TESTOVÁNÍ SCROLL

Kdykoliv by mohlo dojít k potřebě rolování displeje, volá se tento podprogram. K tomu dochází ve třech případech:

- Při zpracování řídicího znaku CR (carriage return).
- Je-li použito AT v INPUT řádku.
- Jestliže výpis přesáhne aktuální řádek a bude zapotřebí pokračovat na dalším řádku.

Na vstupu obsahuje B číslo řádku, který má být testován.

0C55 PO-SCR	BIT 1,(IY+1) (FLAGS)	Jestliže obsluhuješ
	RET NZ	tiskárnu, vrať se.
	LD DE,#0DD9	Adresa CL-SET
	PUSH DE	na zásobník.
	LD A,B	Číslo řádku do A.
	BIT 0,(IY+2) (TV-FLAG)	Jedná-li se o INPUT
	JP NZ,#0D02,PO-SCR-4	...AT,skok dopředu.
	CP (IY+49) (DF-SZ)	Je-li číslo řádku nižší než DF-SZ,
	JR C,#0C86,REPORT-5	skoč do podprogramu chybového hlášení a
	RET NZ	je-li vyšší, vrať se přes CL-SET.
	BIT 4,(IY+2) (TV-FLAG)	Pokud neobsahuješ automatický výpis
	JR Z,#0C88,PO-SCR-2	skoč dopředu.
	LD E,(IY+45) (BREG)	Vyzvedni čítač řádků,
	DEC E	dekrementuj ho a
	JR Z,#0CD2,PO-SCR-3	skoč dopředu, je-li potřeba rolovat.
	LD A,#00	Jinak otevři
	CALL #1601,CHAN-OPEN	kanál "K",
	LD SP,(#5C3F) (LIST-SP)	obnov zásobník,
	RES 4,(IY+2) (TV-FLAG)	signalizuj konec automatického výpis
	RET	a vrať se přes CL-SET.

0C86 REPORT-5	RST #08,ERROR-1	Ohlaš:
	DEFB #04	5-Out of screen

Nyní posuď, je-li potřeba vypsát sdělení "scroll?".

0C88 PO-SCR-2	DEC (IY+82) (SCR-CT)	Dekrementuj skrolovací čítač.
	JR NZ,#0CD2,PO-SCR-3	Není-li nulový, ke sdělení nedojde.
	LD A,#18	Čítač je nastaven.
	SUB B	na novou
	LD (#5C8C),A (SCR-CT)	hodnotu.
	LD HL,(#5C8F) (ATTR-T)	Aktuální hodnoty ATTR-T a MASK-T
	PUSH HL	jsou uschovány.
	LD A,(#5C91) (P-FLAG)	Aktuální hodnota P-FLAG
	PUSH AF	je uschována.
	LD A,#FD	Kanál "K"
	CALL #1601,CHAN-OPEN	je otevřen.
	XOR A	Sdělení "scroll?" má číslo 0.
	LD DE,#0CF8	Báze.
	CALL #0C0A,PO-MSG	Vypsání na obrazovku.
	SET 5,(IY+2) (TV-FLAG)	Signál: čistit dolní část obrazovky po stisku klávesy.
	LD HL,#5C3B	Toto je FLAGS.
	SET 3,(HL)	Signál: " L mod ".



RES 5,(HL)	Signál: zatím žádná klávesa.
EXX	Poznámka: Zde měl být také uschován DE.
CALL #15D4,WAIT-KEY	Vyzvední kód stisknuté klávesy.
EXX	Obnov registry.
CP #20	Při stisku BREAK,
JR Z,#0D00,REPORT-D	skoč na chybové hlášení.
CP #E2	Při stisku STOP,
JR Z,#0D00,REPORT-D	skoč na chybové hlášení.
OR #20	Při stisku n
CP #6E	nebo N,
JR Z,#0D00,REPORT-D	skoč na chybové hlášení.
LD A,#FE	Ostatní znaky vyvolávají rolování.
CALL #1601,CHAN-OPEN	Otevření kanálu "S".
POP AF	Vyzvednutí
LD (#5C91),A (P-FLAG)	a obnovení P-FLAG.
POP HL	Vyzvednutí
LD (#5C8F),HL (ATTR-T)	a obnovení ATTR-T a MASK-T.

Nyní je displej odrolován.

OCD2 PO-SCR-3	CALL #0DFE,CL-SC-ALL	Celý displej je skrolován.
	LD B,(IY+49) (DF-SZ)	Jsou nalezena čísla řádku
	INC B	a sloupce
	LD C,#21	nad dolní částí obrazovky
	PUSH BC	a uschována.
	CALL #0E9B,CL-ADDR	Je nalezena
	LD A,H	adresa
	RRCA	odpovídajícího
	RRCA	atributu
	RRCA	a tato
	AND #03	je
	OR #58	uložena
	LD H,A	do HL.

Řádek o který se jedná, obdrží atributy z dolní části obrazovky a nový řádek na nejspodnějším okraji displeje bude mít hodnoty z ATTR-P. Stačí tedy hodnoty atributů zaměnit.

	LD DE,#5AE0	DE ukazuje na první atribut nejspodnějšiho řádku.
	LD A,(DE)	Hodnota bajtu do A.
	LD C,(HL)	Bajt z řádku nad dolní částí obrazovky do C.
	LD B,#20	32 bajtů je ve hře.
	EX DE,HL	Záměna ukazatelů.
OCF0 PO-SCR-3A	LD (DE),A	Proveď první
	LD (HL),C	záměnu
	INC DE	a pokračuj se
	INC HL	stejnými hodnotami
	DJNZ #OCF0,PO-SCR-3A	pro celý řádek.
	POP BC	Vyzvedni číslo nejspodnějšiho řádku a sloupce horní
		části obrazovky a
	RET	vrať se.

Sdělení "scroll?"

OCF8	DEFB #80	Úvodní značka - bude přeskočena.
	DEFB #73,#63,#72,#6F	s c r o
	DEFB #6C,#6C,#BF	( l ? (invertovaný).

OD00 REPORT-D	RST #08,ERROR-1	Ohlaš:
---------------	-----------------	--------

DEFB #0C

D-BREAK-CONT repeats

Dolní část obrazovky je ošetřena následovně.

0D02	PO-SCR-4	CP #02	Je-li příliš velká,
		JR C,#0C86,REPORT-5	skok na chybové hlášení.
		ADD A,(IY+49) (DF-SZ)	Jestliže není
		SUB #19	potřeba rolovat,
		RET NC	provede se návrat.
		NEG	V A je počet skrolů které se mají vykonat.
		PUSH BC	Uschování čísla řádku a sloupce.
		LD B,A	Počet rolování,
		LD HL,(#5C8F) (ATTR-T)	ATTR-T a MASK-T
		PUSH HL	jsou uschovány.
		LD HL,(#5C91) (P-FLAG)	Rovněž P-FLAG
		PUSH HL	jde na zásobník.
		CALL #0D4D,TEMPS	Budou použity "permanentní" barvy.
		LD A,B	Počet rolování do A.

Nyní je dolní část obrazovky skrolována A krát.

0D1C	PO-SCR-4A	PUSH AF	Ulož počet.
		LD HL,#5C6B	Hodnota v DF-SZ
		LD B,(HL)	jde do B
		LD A,B	a do A,
		INC A	je inkrementována
		LD (HL),A	a vrácena do DF-SZ.
		LD HL,#5C89	Toto je S-POSN-hi.
		CP (HL)	Skok se provede jedině tehdy, když
		JR C,#0D2D,PO-SCR-4B	B=původní hodnota DF-SZ, tedy rolování pouze dolní části.
		INC (HL)	Zvětšení S-POSN-hi
		LD B,#18	Hodnota pro rolování celého displeje.
0D2D	PO-SCR-4B	CALL #0E00,CL-SCROLL	Roluj B krát.
		POP AF	Vyzvednutí a zmenšení
		DEC A	počtu rolování.
		JR NZ,#0D1C,PO-SCR-4A	Skok zpět, není-li skončeno.
		POP HL	Vyzvedni
		LD (IY+87),L (P-FLAG)	a obnov P-FLAG.
		POP HL	Vyzvedni
		LD (#5C8F),HL (ATTR-T)	a obnov ATTR-T a MASK-T.
		LD BC,(#5C88) (S-POSN)	V případě změny S-POSN
		RES 0,(IY+2) (TV-FLAG)	
		CALL #0DD9,CL-SET	volej CL-SET, aby se do DF-CC uložila odpovídající hodnota.
		SET 0,(IY+2) (TV-FLAG)	Signál: obsluha dolní částí obrazovky.
		POP BC	Vyzvedni číslo řádku a sloupce
		RET	a vrať se.

**PODPROGRAM BAREVNÝCH POLOŽEK**

Toto je jeden z nejdůležitějších podprogramů. Je používán kdykoliv je zapotřebí, aby permanentní barevné hodnoty byly zkopírovány do přechodných systémových proměnných. Nejprve jsou zpracovávány ATTR-T a MASK-T.

0D4D	TEMPS	XOR A	Registr A je vynulován.
		LD HL,(#5C8D) (ATTR-P)	Jsou vyzvednuty současně hodnoty ATTR-P a MASK-P.
		BIT 0,(IY+2) (TV-FLAG)	Jedná-li se o hlavní část obrazovky,
		JR Z,#0D5B,TEMPS-1	skoč dopředu.

	LD H,A	Jinak použij hodnotu #00 a hodnotu
	LD L,(IY+14) (BORDCR)	systémové proměnné BORDER.
OD5B TEMPS-1	LD (#5C8F),HL (ATTR-T)	Obě pak ulož na ATTR-T a MASK-T.

Další na řadě je P-FLAG.

	LD HL,#5C91	Toto je P-FLAG.
	JR NZ,#0D65,TEMPS-2	Skoč dopředu, jestliže je to dolní část obrazovky A=#00.
	LD A,(HL)	Jinak vyzvedni hodnotu P-FLAG a
	RRCA	přesuň liché bity na místa sudých.
OD65 TEMPS-2	XOR (HL)	Pokračuj v kopírování
	AND #55	sudých bitů
	XOR (HL)	do registru A a do
	LD (HL),A	systémové proměnné P-FLAG.
	RET	

#### PODPROGRAM CLS

Hned na začátku je smazán celý displej - "body" jsou vynulovány a atributové bajty jsou nastaveny na hodnotu, která odpovídá systémové proměnné ATTR-P. Potom bude smazána také nižší část obrazovky.

OD6B CLS	CALL #0DAF,CL-ALL	Hlavní část displeje je smazána.
OD6E CLS-LOWER	LD HL,#5C3C	Toto je TV-FLAG.
	RES 5,(HL)	Signál: nemaž spodní část obrazovky po stisknutí tlačítka
	SET 0,(HL)	Signál: spodní část obrazovky.
	CALL #0D4D,TEMPS	Nastav barevné hodnoty, tzn. nastav ATTR-T podle BORDCR.
	LD B,(IY+49) (DF-SZ)	Dolní část obrazovky je pak vyčištěna
	CALL #0E44,CL-LINE	za pomocí těchto hodnot.

S výjimkou atributových bajtů pro řádky 22 a 23 jsou všechny atributové bajty nastaveny podle hodnot v ATTR-P.

	LD HL,#5AC0	Adresa 1. atributového bajtu pro řádek 22.
	LD A,(#5CBD) (ATTR-P)	Vyzvedni ATTR-P.
	DEC B	Čítač řádků.
	JR #0D8E,CLS-3	Skoč dopředu do smyčky.
OD87 CLS-1	LD C,#20	32 znaků na řádek.
OD89 CLS-2	DEC HL	Vracej se zpět po řádku a nastavuj
	LD (HL),A	atributové bajty.
	DEC C	
	JR NZ,#0D89,CLS-2	
OD8E CLS-3	DJNZ #0D87,CLS-1	Opakuj smyčku.

Velikost spodní části obrazovky bude nastavena.

LD (IY+49),#02 (DF-SZ)	Její velikost jsou dva řádky.
------------------------	-------------------------------

Nyní ještě zbývají další "úklidové práce".

OD94 CL-CHAN	LD A,#FD	Kanáal "K"
	CALL #1601,CHAN-OPEN	bude otevřen.
	LD HL,(#5C51) (CURCHL)	Vyzvedni adresu současného kanálu a
	LD DE,#09F4	jako výstupní adresu použij #09F4.
	AND A	
ODAO CL-CHAN-A	LD (HL),E	(=PRINT-OUT) a vstupní adresu
	INC HL	#10A8 (=KEY-INPUT).
	LD (HL),D	
	INC HL	

LD	DE,#10A8	
CCF		První je uložena výstupní adresa,
JR	C,#0DA0,CL-CHAN-A	pak adresa vstupní.
LD	BC,#1721	Protože se jedná o dolní část displeje bude nejnižší
JR	#0DD9,CL-SET	tiskový řádek číslo 23. Vrať se přes CL-SET.

#### PODPROGRAM MAZÁNÍ CELÉHO DISPLEJE

Tato podprogram je volán:

- a) příkazem CLS
- b) hlavním prováděcím podprogramem
- c) automatickým výpisem.

ODAF	CL-ALL	LD	HL,#0000	Jsou uloženy nuly
		LD	(#5C7D),HL (COORDS)	do systémové proměnné COORDS.
		RES	0,(IY+4B) (FLAGS2)	Signál: obrazovka je čistá.
		CALL	#0D94,CL-CHAN	Proveď "úklidové práce".
		LD	A,#FE	Kaná l "S"
		CALL	#1601,CHAN-OPEN	bude otevřen.
		CALL	#0D4D,TEMPS	Permanentní barvy budou nastaveny.
		LD	B,#18	24 řádků displeje
		CALL	#0E44,CL-LINE	bude smazáno.
		LD	HL,(#5C51) (CURCHL)	Zajistí,
		LD	DE,#09F4	
		LD	(HL),E	
		INC	HL	
		LD	(HL),D	Že současná výstupní adresa je #09F4 (PRINT-OUT).
		LD	(IY+82),#01 (SCR-CT)	Nastav rolovací čítač.
		LD	BC,#1821	Protože se jedná o horní část obrazovky, bude nejvyšší tisknutý řádek "řádek 0". Pokračuj do podprogramu CL-SET.

#### PODPROGRAM CL-SET

Na vstupu do tohoto podprogramu obsahuje registrový pár BC číslo řádku a sloupce na obrazovce, nebo registr C obsahuje číslo sloupce v bafru tiskárny a pak je nalezena příslušná adresa bitu prvního znaku. Podprogram se vrací přes PO-STORE tak, aby byly uloženy všechny potřebné hodnoty.

ODD9	CL-SET	LD	HL,#5B00	Adresa 1.bajtu bafru tiskárny.
		BIT	11,(IY+1) (FLAGS)	Obsluhuješ-li bafr tiskárny,
		JR	NZ,#0DF4,CL-SET-2	skoč dopředu.
		LD	A,B	Okopíruj číslo řádku.
		BIT	0,(IY+2) (TV-FLAG)	Obsluhuješ-li hlavní část obrazovky,
		JR	Z,#0DEE,CL-SET-1	skoč dopředu.
		ADD	A,(IY+49) (DF-SZ)	První řádek dolní části obrazovky se nazývá "řádek #18"
		SUB	#18	a to musí být převedeno.
ODEE	CL-SET-1	PUSH	BC	Číslo řádku a sloupce jsou uschována
		LD	B,A	a číslo řádku je převedeno do reg. B.
		CALL	#0E9B,CL-ADDR	V HL je nyní adresa začátku řádku a
		POP	BC	čísla řádku a sloupce jsou opět uložena do BC.
ODF4	CL-SET-2	LD	A,#21	Číslo sloupce
		SUB	C	je nyní převráceno
		LD	E,A	převedeno do registrového
		LD	D,#00	páru DE a
		ADD	HL,DE	na závěr je v HL vytvořena požadovaná adresa.
		JP	#0ADC,PO-STORE	Tyto hodnoty jsou uschovány podprogramem PO-STORE.

## ROLOVACÍ PODPROGRAM

Počet řádků, který má být rolován, vstupuje do podprogramu v registru B.

ODFE CL-SC-ALL LD B,#17 Vstupní bod po dotazu "scroll?".

Hlavní vstupní bod z předcházejícího a nebo z příkazu INPUT AT.

OE00 CL-SCROLL CALL #0E9B,CL-ADDR Najdi počáteční adresu řádku.  
LD C,#08 8 mikrořádků tvoří kompletní znakový řádek.

Nyní přichází hlavní rolovací smyčka. Registr B obsahuje počet rolovaných řádků, HL ukazuje na adresu 1. rolovaného bajtu a registr C je využit jako počítadlo mikrořádků.

OE05 CL-SCR-1 PUSH BC Uschovej oba čítače.  
PUSH HL Uschovej adresu.  
LD A,B Nejedná-li se o přechod na  
AND #07 další třetinu  
LD A,B displeje,  
JR NZ,#0E19,CL-SCR-3 skoč dopředu.

Mikrořádky prvního znakového řádku každé třetiny musí být přeneseny přes 2K hranici (každá třetina má 2KB).

OE0D CL-SCR-2 EX DE,HL Po skončení těchto  
LD HL,#F8E0 operací zůstává v  
ADD HL,DE HL původní hodnota  
EX DE,HL a DE ukazuje na novou cílovou adresu.  
LD BC,#0020 Jedná se o 32 znaků.  
DEC A Zmenši počítadlo za každý přenesený řádek  
LDIR a přenes požadovaných 32 bajtů.

Pixlové řádky uvnitř třetin mají být nyní rolovány. Registr A obsahuje při prvním průběhu hodnoty #01 až #07, #09 až #0F nebo #11 až #17.

OE19 CL-SCR-3 EX DE,HL Znovu je DE nastaven tak,  
LD HL,#FFEO aby ukazoval na  
ADD HL,DE požadovanou cílovou  
EX DE,HL adresu, ale tentokrát pouze o 32 pozic dále.  
LD B,A Uschovej číslo řádku v registru B.  
AND #07 Nyní zjistí,  
RRCA kolik znaků zůstává  
RRCA v této třetině.  
RRCA  
LD C,A Předej celkový počet znaků do registru C.  
LD A,B Vyzvedni číslo řádku.  
LD B,#00 A nyní již obsahuje celkový počet znaků i BC  
LDIR a mikrořádek každého znaku je rolován.  
LD B,#07 Příprav pro zvětšení adresu, na kterou se bude  
ADD HL,BC ukládat při třetinové hranici a zvětší HL o #0700.  
AND #F8 Nejsou-li zde již žádné další zvažované třetiny,  
JR NZ,#0E0D,CL-SCR-2 skoč zpět.

Nyní zjistí zda byla smyčka použita 8 krát; tj. pro 8 mikrořádků.

POP HL Vyzvedni adresu originálu.  
INC H Adresuj další mikrořádek.  
POP BC Vyzvedni čítače.  
DEC C Zmenši čítač mikrořádků a pokud nebylo

JR NZ,#0E05,CL-SR-1 přeneseno všech 8 mikrořádků skákej zpět.

Dále jsou rolvány atributové bajty. Povšimněte si, že registr B obsahuje stále počet řádků, které mají být rolvány, a registr C obsahuje nulu.

CALL #0E88,CL-ATTR	Je nalezena požadovaná adresa v oblasti proměnných a počet znaků v "B" řádcích.
LD HL,#FFEO	
ADD HL,DE	Přemístovací konstanta pro všechny atributové bajty je 32 míst.
EX DE,HL	
LDIR	Atributové bajty jsou nyní rolvány.

Zbývá ještě vyčistit spodní řádky displeje.

LD B,#01 B obsahuje 1, se kterou vstoupí do dalšího podprogramu.

#### PODPROGRAM SMAZÁNÍ ŘÁDKU

Tento podprogram vyčistí spodní řádky obrazovky, jejichž počet je v registru B.

0E44 CL-LINE	PUSH BC	Číslo řádku je během podprogramu uschováno.
	CALL #0E9B,CL-ADDR	V HL je <b>vytvorena</b> počáteční adresa řádku.
	LD C,#08	Znovu je C nastaveno jako čítač 8 mikrořádků.

Zde přichází smyčka, která vyčistí všechny "pixlové" řádky.

0E4A CL-LINE-1	PUSH BC	Uchověj číslo řádku a čítač mikrořádků.
	PUSH HL	Uchověj adresu.
	LD A,B	Uchověj číslo řádku v registru A.
0E4D CL-LINE-2	AND #07	Zjistí, o kolik se jedná znaků nebo-li "B mod 8" řádků
	RRCA	a tuto hodnotu předej do registru
	RRCA	C (který bude obsahovat #00 pro novou třetinu).
	LD C,A	Vyzvedni číslo řádku a nastav registr BC tak, aby číslo které obsahuje bylo o jednu menší než počet znaků.
	LD A,B	Nastav DE tak, aby ukazoval na první znak, vyčisti bajt prvního znaku,
	LD B,#00	nastav DE na druhý znak a vyčisti celý mikrořádek.
	DEC C	
	LD D,H	Pro každou třetinu displeje bude HL zvýšeno o konstantu #0701.
	LD E,L	Dekrementuj čítač řádků.
	LD (HL),#00	Předej číslo třetiny do registru B.
	INC DE	
	LDIR	Skoč zpět, jsou-li zde ještě nějaké třetiny k ošetření.
	LD DE,#0701	
	ADD HL,DE	
	DEC A	
	AND #FB	
	LD B,A	
	JR NZ,#0E4D,CL-LINE-2	

Zde se zjistí, jestli smyčka proběhla 8 krát.

POP HL	Připrav adresu pro každý mikrořádek.
INC H	
POP BC	Vyzvedni čítače.
DEC C	Zmenši čítač mikrořádků a skoč zpět, pokud není konec.
JR NZ,#0E4A,CL-LINE-1	

Nyní musí být nastaveny atributové bajty. Hodnota ATTR-P bude použita při obsluze hlavní části displeje a hodnota BORDCR při dolní části displeje.

	CALL #0E88,CL-ATTR	Je nalezen počet atributových bajtů a adresa prvního z nich.
	LD H,D	HL tedy ukazuje na první atributový bajt a
	LD L,E	DE na druhý.
	INC DE	Vyzvedni hodnotu ATTR-P.
	LD A,(#5C8D) (ATTR-P)	Jedná-li se o hlavní část obrazovky,
	BIT 0,(IY+2) (TV-FLAG)	skoč dopředu, jinak
	JR Z,#0E80,CL-LINE-3	použij hodnotu BORDCR.
0E80	LD A,(#5C4B) (BORDCR)	Nastav atributový bajt.
	LD (HL),A	Jeden bajt už je hotov.
	DEC BC	A nyní bude jeho hodnota okopírována do všech ostatních bajtů.
	LDIR	Obnov si číslo řádku.
	POP BC	Nastav číslo sloupce na nejkrajnější levou pozici a vrať se.
	LD C,#21	
	RET	

#### PODPROGRAM CL-ATTR

Tento podprogram má dvě rozdílné funkce:

a) Vypočítat adresu atributového bajtu pro danou adresu v paměti a uložit ji do DE. Povšimněte si, že hodnoty na vstupu ukazují na devátý mikrořádek znaku.

b) Pro daný počet řádků v registru B je v registrovém páru BC navrácen počet znaků připadajících na tyto řádky.

0E88	CL-ATTR	LD A,H	Vyzvedni vyšší bajt
		RRCA	adresy a
		RRCA	vynásob toto číslo číslem 32.
		RRCA	
		DEC A	Jdi zpátky na 8. mikrořádek.
		OR #50	Adresuj oblast atributů.
		LD H,A	Vytvoř nový vyšší bajt
		EX DE,HL	a celou adresu převeď do DE.
		LD H,C	Zde je vždy nula a
		LD L,B	toto je číslo řádku.
		ADD HL,HL	
		ADD HL,HL	
		ADD HL,HL	
		ADD HL,HL	
		ADD HL,HL	Hodnotu vynásobenou číslem 32 převeď
		LD B,H	zpět do registrového
		LD C,L	páru BC před
		RET	návratem.

#### PODPROGRAM CL-ADDR

Pro zadané číslo řádku v registru B vytvoří v HL tento podprogram odpovídající adresu obrazové paměti.

0E9B	CL-ADDR	LD A,#18	Číslo řádku
		SUB B	je převráceno a
		LD D,A	výsledek je uložen do registru D.
		RRCA	Ve skutečnosti $(A \bmod 8) * 32$ .
		RRCA	V každé třetině displeje má nižší bajt tyto hodnoty:
		RRCA	

AND #E0	1. řádek = #00, 2. řádek = #20, atd.
LD L,A	Nižší bajt je uložen do L.
LD A,D	Skutečné číslo řádku je vyzvednuto,
AND #18	ve skutečnosti se jedná o $64 + 8 * INT(A/8)$
OR #40	Vyšší bajt horní třetiny displeje má hodnotu #40, bajt
LD H,A	střední třetiny je #48 a nižší třetina je #50. Tyto
RET	hodnoty jsou uloženy do registru H a provede se návrat.

#### PODPROGRAM PŘÍKAZU COPY

Tento podprogram obslouží 176 mikrořádků obrazovky, a to jeden po druhém.

OEAC COPY	DI	Po dobu provádění tohoto podprogramu je zakázáno maskované přerušení.
	LD B,#B0	Konstanta 176 řádků.
	LD HL,#4000	Bázová adresa obrazovky.

Zde se vstupuje do smyčky.

0EB2 COPY-1	PUSH HL	Uchovej bázovou adresu a
	PUSH BC	počet řádků.
	CALL #0EF4,COPY-LINE	Tento podprogram bude zavolán 176 krát.
	POP BC	Vyzvedni číslo řádku
	POP HL	a bázovou adresu.
	INC H	Bázová adresa je posunuta na další mikrořádek.
	LD A,H	Nebylo-li přeneseno ještě všech 8
	AND #07	mikrořádků,
	JR NZ,#0EC9,COPY-2	skoč dopředu.

Pro každý nový znakový řádek musí být bázová adresa upravena.

	LD A,L	Vyzvedni nižší bajt.
	ADD A,#20	Zvyš ho o #20.
	LD L,A	CY flag bude nulový pohybuješ-li se uprostřed třetiny obrazovky.
	CCF	Neguj CY flag.
	SBC A,A	Registr A bude obsahovat #F8,
	AND #F8	pokud jsi uvnitř třetiny, ale #00 pokud jdeš do nové třetiny.
	ADD A,H	Vyšší bajt adresy
	LD H,A	je nyní <b>upraven</b> .
0EC9 COPY-2	DJNZ #0EB2,COPY-1	Skoč zpět,dokud jsi nevytiskl 176 mikrořádků.
	JR #0EDA,COPY-END	Skoč na konec podprogramu COPY.

#### PODPROGRAM COPY-BUFF

Tento podprogram je volán, kdykoliv je zapotřebí vyvolat obsah bafry na tiskárnu.

0ECD COPY-BUFF	DI	Maskované přerušení je zakázáno.
	LD HL,#5B00	Adresa prvního bajtu bafry.
	LD B,#08	Je zde osm mikrořádků.
0ED3 COPY-3	PUSH BC	Uchovej číslo řádku.
	CALL #0EF4,COPY-LINE	Tento podprogram je volán 8-krát.
	POP BC	Vyzvedni číslo řádku.
	DJNZ #0ED3,COPY-3	Opakuj do vytištění osmi mikrořádků.

Pokračuj přes podprogram COPY-END.



OEDA	COPY-END	LD A,#04	Zastav motor tiskárny.
		OUT (#FB),A	
		EI	Povol nemaskovatelné přerušení a pokračuj přes podprogram CLEAR-PRB.

#### PODPROGRAM \*VYČISTI BUFFER TISKÁRNÝ\*

Při zavolání tohoto podprogramu je vyčištěn bafr tiskárny.

OEDF	CLEAR-PRB	LD HL,#5B00	Adresa prvního bajtu bafru tiskárny.
		LD (IY+70),L (PR-CC-lo)	Nastav tiskový sloupec na nulu.
		XOR A	Vynuľuj registr A a
		LD B,A	registr B (ve skutečnosti obsahuje dekadicky 256).
OEE7	PRB-BYTES	LD (HL),A	Nyní je vyčištěno 256 bajtů bafru tiskárny,
		INC HL	jeden po druhém.
		DJNZ #OEE7,PRB-BYTES	
		RES 1,(IY+4B) (FLAGS2)	Signál: bafr je prázdný.
		LD C,#21	Nastav tiskovou pozici
		JP #ODD9,CL-SET	a vrať se přes CL-SET a PO-STORE.

#### PODPROGRAM COPY-LINE

Na vstupu do tohoto podprogramu obsahuje registrový pár HL adresu 32 bajtů, které představují mikrořádek a B obsahuje číslo mikrořádku.

OEF4	COPY-LINE	LD A,B	Okopíruj číslo řádky.
		CP #03	Registr A bude obsahovat #00 dokud se nebudou
		SBC A,A	obsluhovat poslední dva řádky.
		AND #02	Zpomal motor, ale pouze pro poslední
		OUT (#FB),A	dva mikrořádky.
		LD D,A	Registr D bude obsahovat buďto #00 nebo #02.

Před vykonáním jakéhokoliv tisku se musí provést tři testy.

OEFD	COPY-L-1	CALL #1F54,BREAK-KEY	Nebyl-li stlačen BREAK,
		JR C,#0F0C,COPY-L-2	skoč dopředu.
		LD A,#04	Signál: motor stop.
		OUT (#FB),A	
		EI	Povol maskované přerušení.
		CALL #OEDF,CLEAR-PRB	Vyčisti bafr tiskárny a
		RST #0B,ERROR-1	Ohlaš:
		DEFB #0C	D-BREAK-CONT repeats.
0F0C	COPY-L-2	IN A,(#FB)	Zjistí v jakém stavu
		ADD A,A	se nachází tiskárna.
		RET M	Proveď okamžitý návrat, není-li tiskárna připojena.
		JR NC,#OEFD,COPY-L-1	Počkej na jehlu.
		LD C,#20	Konstanta pro 32 bajtů.

Nyní vstup do smyčky, která obsluhuje tyto bajty.

OF14	COPY-L-3	LD E,(HL)	Vyzvedni bajt.
		INC HL	Posuň ukazatel.
		LD B,#08	V jednom bajtu je 8 bitů.
OF18	COPY-L-4	RL D	Posuň D doleva.
		RL E	Každý bit posuň do CY.
		RR D	Posuň D zpět a naber CY z E.
OF1E	COPY-L-5	IN A,(#FB)	Znovu otestuj stav tiskárny
		RRA	

JR	NC,#0F1E,COPY-L-5	a čekej na signál.
LD	A,D	Nyní pokračuj a předej
OUT	(#FB),A	bit tiskárně.

Poznámka: bit 2=0 signál: zapni motor, bit 1=1 signál: zpomal motor, bit 7=1 signál: vlastní tisk.

DJNZ	#0F18,COPY-L-4	Tiskni každý bit.
DEC	C	Sniž čítač bajtů.
JR	NZ,#0F14,COPY-L-3	Skoč zpět, jestli jsou zde ještě nějaké bajty.
RET		Jinak se vrať.

## PODPROGRAMY EDITORU

Editor se volá při dvou příležitostech:

- Z hlavní prováděcí procedury tak, aby mohl uživatel vložit řádek do programu.
- Z podprogramu příkazu INPUT.

Nejprve je uložen chybový zásobník a podvržena alternativní adresa.

0F2C	EDITOR	LD HL,(#5C3D) (ERR-SP)	Na zásobník je
		PUSH HL	uložena aktuální adresa.
0F30	ED-AGAIN	LD HL,#107F	Toto je adresa ED-ERRORu.
		PUSH HL	Jakákoliv událost,
		LD (#5C3D),SP (ERR-SP)	kteřá by vedla ke spuštění podpr. chybových hlášení,
			způsobí návrat do podprogramu ED-ERROR.

Nyní se vstupuje do smyčky, která obsluží **každý úder na klávesu**

0F38	ED-LOOP	CALL #15D4,WAIT-KEY	Z tohoto programu se vrát po stisknutí klávesy.
		PUSH AF	Přechodně si uschovej kód.
		LD D,#00	Vynuluj registr D a
		LD E,(IY-1) (PIP)	vyzvedni si trvání klávesového
		LD HL,#00C8	cvaknutí, jeho výšku
		CALL #03B5,BEEPER	a udělej píp.
		POP AF	Obnov si kód a
		LD HL,#0F38	ulož na zásobník adresu
		PUSH HL	ED-LOOP.

Nyní analyzuj získaný kód.

CP	#18	Akceptuj všechny znakové kódy, grafický kód a
JR	NC,#0FB1,ADD-CHAR	a všechny tokens.
CP	#07	
JR	C,#0FB1,ADD-CHAR	Také akceptuj čárku ",", (tisk v druhé polovině řádku).
CP	#10	
JR	C,#0F92,ED-KEYS	Akceptuj všechny editační klávesy.

Nyní jsou posuzeny klávesy od INK po TAB.

LD	BC,#0002	INK a PAPER vyžadují dvě místa.
LD	D,A	Okopíruj kód do D.
CP	#16	
JR	C,#0F6C,ED-CONTR	Skoč dopředu pro INK a PAPER.

Zde budou posuzovány AT a TAB:

INC	BC	Jsou požadována tři místa.
BIT	7,(IY+55) (FLAGX)	Skoč dopředu, neobsluhuješ-li
JP	Z,#101E,ED-IGNORE	INPUT LINE ...
CALL	#15D4,WAIT-KEY	Vezmi druhý kód
LD	E,A	a dej ho do E.

Jsou vyzvednuty ostatní bajty pro řídící znaky.

0F6C	ED-CONTR	CALL #15D4,WAIT-KEY	Vyzvedni další kód.
		PUSH DE	Uschovej <b>oba</b> kódy.
		LD HL,(#5C5B) (K-CUR)	Vyzvedni K-CUR.
		RES 0,(IY+7) (MODE)	Signál "K" mode.
		CALL #1655,MAKE-ROOM	Vytvoř prostor dvou nebo tří bajtů.

POP	BC	Obnov předchozí kódy.
INC	HL	Ukazuj na první místo a
LD	(HL),B	vlož tam první kód.
INC	HL	
LD	(HL),C	<b>Vlož druhý kód.</b> Druhý kód bude přepsán, jedná-li se o INK a PAPER.
JR	#0F8B,ADD-CH-1	<b>Skoč dopředu.</b>

#### PODPROGRAM ADD-CHAR

Tento podprogram ve skutečnosti přidává kód znaku do současného EDIT nebo INPUT řádku.

0F81	ADD-CHAR	RES 0,(IY+7) (MODE)	Signál "K mode".
		LD HL,(#5C5B) (K-CUR)	Vyzvedni pozici kurzoru.
		CALL #1652,ONE-SPACE	Udělej jednu mezeru.
0F8B	ADD-CH-1	LD (DE),A	Vlož kód do této mezery a signalizuj,
		INC DE	že kurzor se musí
		LD (#5C5B),DE (K-CUR)	postavit na další místo.
		RET	Pak se vrať do ED-LOOP.

Editační tlačítka jsou zpracována takto:

0F92	ED-KEYS	LD E,A	Kód je převeden do registrového páru DE.
		LD D,#00	
		LD HL,#0F99	Bázořová adresa tabulky editačních kláves do HL.
		ADD HL,DE	Vstupní bod je adresován a
		LD E,(HL)	vyzvednut do E.
		ADD HL,DE	Adresa obslužného podprogramu
		PUSH HL	je uschována na zásobníku a
		LD HL,(#5C5B) (K-CUR)	do HL je uložena pozice kurzoru.
		RET	Je proveden nepřímý skok do <b>žádaného</b> podprogramu.

#### TABULKA EDITAČNÍCH KLÁVES

adresa	doplňek	znak
0FA0	09	EDIT
0FA1	66	kurzor vlevo
0FA2	6A	kurzor vpravo
0FA3	50	kurzor dolů
0FA4	B5	kurzor nahoru
0FA5	70	DELETE
0FA6	7E	ENTER
0FA7	CF	SYMBOL SHIFT
0FA8	D4	GRAPHICS

#### PODPROGRAM TLAČÍTKA EDIT

Je-li systém v editačním modu, způsobí stisknutí klávesy EDIT snesení aktuálního basicového řádku (tj. řádku, na který ukazuje programový kurzor). Je-li systém v modu INPUT je po stisknutí klávesy EDIT vymazán text, vložený uživatelem.

0FA9	ED-EDIT	LD HL,(#5C49) (E-PPC)	Vyzvedni číslo současného řádku,
		BIT 5,(IY+55) (FLAGX)	ale skoč dopředu,
		JP NZ,#1097,CLEAR-SP	jestliže jsi v modu INPUT.
		CALL #196E,LINE-ADDR	Nalezni adresu začátku řádku a s ní
		CALL #1695,LINE-NO	i jeho číslo.
		LD A,D	Je-li obdrženo číslo řádku nula,
		OR E	

JP	Z,#1097,CLEAR-SP	vyčisti jen editační oblast.
PUSH	HL	Uschovej adresu tohoto řádku.
INC	HL	Ulož si do BC
LD	C,(HL)	
INC	HL	
LD	B,(HL)	jeho délku.
LD	HL,#000A	Pak k této délce
ADD	HL,BC	přičti 10 a
LD	B,H	výsledek opět
LD	C,L	ulož do BC.
CALL	#1F05,TEST-ROOM	Testuj, je-li zde dostatek místa pro tento řádek.
CALL	#1097,CLEAR-SP	Nyní vyčisti editační oblast.
LD	HL,(#5C51) (CURCHL)	Vyzvedni současnou adresu kanálu a
EX	(SP),HL	zaměň ji s adresou tohoto řádku.
PUSH	HL	Přechodně si ji ulož.
LD	A,#FF	Aby mohl být řádek okopírován do editační oblasti,
CALL	#1601,CHAN-OPEN	otevři kanál "R".
POP	HL	Vyzvedni adresu řádku a postav
DEC	HL	jeden bajt před řádek.
DEC	(IY+15) (E-PPC-lo)	Zmenši číslo současného řádku, aby se zabránilo
		vytisknutí kurzoru
CALL	#1855,OUT-LINE	a vytiskni basicový řádek.
INC	(IY+15) (E-PPC-lo)	Zvětši číslo řádku.

Poznámka: tyto úpravy někdy nesplní svou funkci a kurzor je vytisknut.

LD	HL,(#5C59) (E-LINE)	Vyzvedni adresu řádku v editační oblasti a
INC	HL	přeskoč číslo řádku
INC	HL	
INC	HL	a délku řádku tak,
INC	HL	
LD	(#5C5B),HL (K-CUR)	abys našel pozici kurzoru.
POP	HL	Vyzvedni dřívější kanálovou adresu a
CALL	#1615,CHAN-FLAG	nastav příslušné vlajky.
RET		Výstup do ED-LOOP.

#### EDITAČNÍ PODPROGRAM KURZOR DOLŮ

OFF3	ED-DOWN	BIT	5,(IY+55) (FLAGX)	V INPUT modu
		JR	NZ,#1001,ED-STOP	skoč dopředu.
		LD	HL,#5C49	Toto je E-PPC.
		CALL	#190F,LN-FETCH	Je nalezeno číslo dalšího řádku a
		JR	#106E,ED-LIST	je proveden nový automatický výpis.
1001	ED-STOP	LD	(IY+0),#10 (ERR-NR)	Hlášení "STOP in INPUT".
		JR	#1024,ED-ENTER	Skoč dopředu.

#### EDITAČNÍ PODPROGRAM KURZOR DOLEVA

1007	ED-LEFT	CALL	#1031,ED-EDGE	Kurzor je posunut.
		JR	#1011,ED-CUR	Skok dopředu.

#### EDITAČNÍ PODPROGRAM KURZOR DOPRAVA

100C	ED-RIGHT	LD	A,(HL)	Současný znak je testován a
		CP	#0D	jednalo-li se o ENTER,

	RET Z	je proveden návrat.
	INC HL	Postav kurzor za znak.
1011 ED-CUR	LD (#5C5B),HL (K-CUR)	Nastav systémovou proměnnou K-CUR.
	RET	

#### PODPROGRAM DELETE V EDITU

1015 ED-DELETE	CALL #1031,ED-EDGE	Posuň kurzor doleva.
	LD BC,#0001	zruš současný
	JP #19E8,RECLAIM-2	znak.

#### PODPROGRAM ED-IGNORE

101E ED-IGNORE	CALL #15D4,WAIT-KEY	Další dva znaky z klávesnicového
	CALL #15D4,WAIT-KEY	vstupního podprogramu budou ignorovány.

#### PODPROGRAM ENTER EDIT

1024 ED-ENTER	POP HL	Návratové adresy
	POP HL	ED-LOOP a ED-ERROR jsou odhozeny.
1026 ED-END	POP HL	Původní hodnota
	LD (#5C3D),HL (ERR-SP)	ERR-SP je obnovena.
	BIT 7,(IY+0) (ERR-NR)	Jestliže nebyly
	RET NZ	žádné chyby, vrať se.
	LD SP,HL	Jinak proveď
	RET	nepřímý skok do chybového podprogramu.

#### PODPROGRAM ED-EDGE

Adresa je v registrovém páru HL a bude zmenšena v případě, že kurzor nebude stát mezi řídícími znaky a jejich parametry.

1031 ED-EDGE	SCF	DE obsahuje E-LINE (při editaci), nebo
	CALL #1195,SET-DE	WORKSP (při INPUTu).
	SBC HL,DE	CY flag bude nastaven, má-li být
	ADD HL,DE	kurzor na začátku řádku.
	INC HL	Oprava po odečtení.
	POP BC	Odhoď návratovou adresu
	RET C	a vrať se přes ED-LOOP, bylo-li nastaveno CY.
	PUSH BC	Obnov návratovou adresu.
	LD B,H	Ulož současnou adresu
	LD C,L	kurzoru do registrového páru <b>BC</b> .

Tato smyčka zajistí, že řídící znaky nebudou odděleny od svých parametrů.

103E ED-EDGE-1	LD H,D	HL bude ukazovat na
	LD L,E	znak, který stojí
	INC HL	za znakem, na který ukazuje DE.
	LD A,(DE)	Vyzvedni kód znaku, který je adresován registrovým
		párem DE.
	AND #F0	Nejedná-li se o
	CP #10	znaky od INK až TAB
	JR NZ,#1051,ED-EDGE-2	skoč dopředu.
	INC HL	Překroč jeden znak
	LD A,(DE)	a opět vyzvedni jeho kód.

SUB #17 CY flag je při TAB.  
 ADC A,#00 nulován.

Poznámka: Touto instrukcí jsou odděleny AT a TAB, ale ty stejně nejsou v této formě implementovány, takže to nevedí.

	JR	NZ,#1051,ED-EDGE-2	Skoč dopředu, pokud se nejedná o AT a TAB, které jsou-li použity, mají dva parametry.	
	INC	HL		
1051	ED-EDGE-2	AND	A	Připrav se na
	SBC	HL,BC	pravdivé odečtení.	
	ADD	HL,BC	CY flag bude nastaven na nulu, když posouvaný ukazatel dosáhne K-CUR.	
	EX	DE,HL	Pro další smyčku použij posunutý ukazatel, ale	
	JR	C,#103E,ED-EDGE-1	skoč zpět, jestliže jsi překročil K-CUR.	
	RET			

Poznámka: Bude to řídící znak, který bude vymazán použitím DELETE.

#### PODPROGRAM EDITAČNÍ KURZOR NAHORU

1059	ED-UP	BIT	5,(IY+55) (FLAGX)	
		RET	NZ	Vrať se, když jsi v modu INPUT.
		LD	HL,(#5C49) (E-PPC)	Zjisti číslo současného řádku a jeho počáteční adresu.
		CALL	#196E,LINE-ADDR	
		EX	DE,HL	HL nyní ukazuje na předchozí řádek.
		CALL	#1695,LINE-NO	Číslo řádku je vyzvednuto.
		LD	HL,#5C4A	Toto je systémová proměnná E-PPC.
		CALL	#191C,LN-STORE	Číslo řádku je uloženo.
106E	ED-LIST	CALL	#1795,AUTO-LIST	Automatický listing je znovu zavolán a potom
		LD	A,#00	
		JP	#1601,CHAN-OPEN	otevřen kanál "K" před návratem do ED-LOOP.

#### PODPROGRAM ED-SYMBOL

Jestliže byly použity kódy SYMBOL & GRAPHICS, budou ošetřeny následovně:

1076	ED-SYMBOL	BIT	7,(IY+55) (FLAGX)	Skoč zpět, pokud
		JR	Z,#1024,ED-ENTER	nejsi v INPUT LINE.
107C	ED-GRAPH	JP	#0F81,ADD-CHAR	Skoč zpět.

#### PODPROGRAM ED-ERROR

Toto je vstupní bod, došlo-li k nějaké chybě.

107F	ED-ERROR	BIT	4,(IY+48) (FLAGS2)	Skoč zpět, používáš-li
		JR	Z,#1026,ED-END	jiný kanál než "K".
		LD	(IY+0),#FF (ERR-NR)	Zruš číslo chyby a
		LD	D,#00	proved zavrčení, než
		LD	E,(IY-2) (RASP)	
		LD	HL,#1A90	
		CALL	#03B5,BEEPER	se vrátíš do
		JP	#0F30,ED-AGAIN	editoru.

#### PODPROGRAM CLEAR-SP

Editační oblast nebo WORKSPACE jsou vyčištěny.

1097	CLEAR-SP	PUSH HL	Uschovej ukazatel prostoru.
		CALL #1190,SET-HL	DE bude ukazovat na prvni bajt a HL na posledni.
		DEC HL	Uprav hodnotu.
		CALL #19E5,RECLAIM-1	Nyni se zrusi potrebny pocet bajtu.
		LD (#5C5B),HL (K-CUR)	Systemove promenne K-CUR a
		LD (IY+7),#00 (MODE)	MODE "K mode" jsou nastaveny pred
		POP HL	vyzvednutim ukazatele a provedenim
		RET	návratu.

#### PODPROGRAM PRO VSTUP KLÁVESY

Tento důležitý podprogram vrací kód poslední stisknuté klávesy, ale povšimněte si, že klávesa CAPS LOCK, změny modů a změny barevných parametrů jsou v tomto podprogramu také zpracovány.

10A8	KEY-INPUT	BIT 3,(IY+2) (TV-FLAG)	Okopíruj editační nebo inputový řádek
		CALL NZ,#111D,ED-COPY	na obrazovku, jestliže byl změněn mód.
		AND A	Vrať se s CY flag=0 a Z flag=0,
		BIT 5,(IY+1) (FLAGS)	jestliže nebyla stisknuta žádná nová klávesa
		RET Z	
		LD A,(#5C0B) (LAST-K)	Jestliže byla, vyzvedni kód a poznač vyzvednutí.
		RES 5,(IY+1) (FLAGS)	
		PUSH AF	Přechodně si uschovej kód a smaž dolní část obrazovky
		BIT 5,(IY+2) (TV-FLAG)	a vyčisti
		CALL NZ,#0D6E,CLS-LOWER	dolní část obrazovky, pokud je třeba (např. po "scroll?")
		POP AF	Vyzvedni kód a
		CP #20	
		JR NC,#111B,KEY-DONE	akceptuj všechny znaky a kódy pro tokens.
		CP #10	
		JR NC,#10FA,KEY-CONTR	Skoč dopředu při téměř všech kódech pro řídící znaky.
		CP #06	
		JR NC,#10DB,KEY-M&CL	Skoč dopředu s kódem pro CAPS LOCK nebo pro změnu "mode".

Nyní se vypořádej s FLASH, BRIGHT a INVERSE.

LD B,A	Ulož si kód.
AND #01	Ponech si pouze bit 0.
LD C,A	C obsahuje #00 (=off) nebo #01 (=on).
LD A,B	Vyzvedni kód.
RRA	Jednou ho rotuj, čím přijdeš o bit 0.
ADD A,#12	Přičti #12 kvůli BRIGHT, FLASH a INVERSE
JR #1105,KEY-DATA	á pokračuj dále s parametrem v C.

Kód pro CAPS LOCK a kód modů jsou ošetřeny místně.

10DB	KEY-M&CL	JR NZ,#10E6,KEY-MODE	Skoč s kódy modů.
		LD HL,#5C6A	Toto je FLAGS2.
		LD A,#08	Flípuj
		XOR (HL)	bit 3 FLAGS2.
		LD (HL),A	Jedná se o vložku CAPS LOCK.
		JR #10F4,KEY-FLAG	Skoč dopředu.
10E6	KEY-MODE	CP #0E	
		RET C	Zkontroluj dolní limit.
		SUB #0D	Sniž rozsah.
		LD HL,#5C41	Toto je MODE.
		CP (HL)	Bylo změněno?
		LD (HL),A	Vlož kód nového
		JR NZ,#10F4,KEY-FLAG	mode a skoč, když bylo změněno.
		LD (HL),#00	Jinak nastav "L" mode.



10F4	KEY-FLAG	SET 3,(IY+2) (TV-FLAG)	Signál mode mohlo být změněno.
		CP A	Nuluj CY flag a
		RET	vrať se.

Nyní jsou zpracovány kódy řídících znaků (mimo FLASH, BRIGHT a INVERSE) pro barvy (16-23 INK a 24-31 PAPER).

10FA	KEY-CONTR	LD B,A	Ušchovej kód.
		AND #07	Nastav registr C na
		LD C,A	#00 až #07.
		LD A,#10	Registr A nyní obsahuje kód pro INK.
		BIT 3,B	Ale nebyl-li kód definován pomocí SHIFTu,
		JR NZ,#1105,KEY-DATA	
		INC A	nastav registr A na hodnotu PAPER.

Parametr je nyní uschován v proměnné K-DATA a kanálová adresa je změněna z KEY-INPUT na KEY-NEXT.

1105	KEY-DATA	LD (IY-45),C (K-DATA)	Ulož parametr.
		LD DE,#110D	Toto je KEY-NEXT.
		JR #1113,KEY-CHAN	Skoč dopředu.

Poznámka: Při prvním průchodu přes KEY-INPUT je v registru A vrácena hodnota kontrolního kódu. Při dalším vstupu přes KEY-NEXT se vrací parametr.

110D	KEY-NEXT	LD A,(#5C0D) (K-DATA)	Vyzvedni parametr.
		LD DE,#10A8	Toto je KEY-INPUT.

Nyní nastav vstupní adresu na první kanálovou oblast.

1113	KEY-CHAN	LD HL,(#5C40) (CHANS)	Vyzvedni kanálovou adresu.
		INC HL	
		INC HL	
		LD (HL),E	Nyní nastav vstupní adresu.
		INC HL	
		LD (HL),D	

Konečně výstup s požadovaným kódem v registru A.

111B	KEY-DONE	SCF	Signalizuj, že kód byl nalezen a
		RET	vrať se.

#### PODPROGRAM PRO OKOPÍROVÁNÍ DOLNÍ ČÁSTI OBRAZOVKY

Tento podprogram je volán kdykoli má být řádek z editační oblasti nebo z oblasti INPUTu vytisknut na obrazovku.

111D	ED-COPY	CALL #0D4D,TEMPS	Použij permanentní barvy.
		RES 3,(IY+2) (TV-FLAG)	Signál: mód nezměněn
		RES 5,(IY+2) (TV-FLAG)	Signál: Nečistit dolní část obrazovky.
		LD HL,(#5C8A) (S-POSNL)	
		PUSH HL	Ušchovej současnou hodnotu S-POSNL.
		LD HL,(#5C3D) (ERR-SP)	
		PUSH HL	Ušchovej současnou hodnotu ERR-SP.
		LD HL,#1167	Toto je ED-FULL.
		PUSH HL	Ušchovej tuto hodnotu na zásobník
		LD (#5C3D),SP (ERR-SP)	a tím se tato hodnota stane vstupním bodem při chybě.
		LD HL,(#5C82) (ECHO-E)	
		PUSH HL	Ulož na zásobník hodnotu ECHO-E.

SCF		HL bude ukazovat na začátek prostoru a
CALL #1195,SET-HL		DE na jeho konec.
EX DE,HL		Zaměň ukazatele.
CALL #187D,OUT-LINE2		Vytiskni řádek.
EX DE,HL		Opět zaměň ukazatele a
CALL #18E1,OUT-CURS		vytiskni kurzor.
LD HL,(#5CBA) (S-POSNL)		Potom vyzvedni současnou hodnotu S-POSNL a
EX (SP),HL		zaměň ji s ECHO-E.
EX DE,HL		Předej ECHO-E do DE.
CALL #0D4D,TEMPS		Opět vyzvedni permanentní barvy.

Zbytek řádku, který byl právě obsluhován je nyní vyplněn mezerami, tištěnými v barvách permanentního PAPERu.

1150 ED-BLANK	LD A,(#5C8B)(S-POSNL-hi)	Vyzvedni číslo současného řádku a
	SUB D	odečti číslo starého řádku.
	JR C,#117C,ED-C-DONE	Skoč vpřed, není-li třeba dodat mezery.
	JR NZ,#115E,ED-SPACES	Skoč vpřed, jestliže se nenacházíš na stejném řádku.
	LD A,E	Vyzvedni staré číslo sloupce a
	SUB (IY+80) (S-POSNL-lo)	odečti ho od nového čísla sloupce.
	JR NC,#117C,ED-C-DONE	Skoč, když není zapotřebí mezerovat.
115E ED-SPACES	LD A,#20	Toto je mezera.
	PUSH DE	Ušchvej staré hodnoty.
	CALL #09F4,PRINT-OUT	Vytiskni mezeru.
	POP DE	Vyzvedni staré hodnoty.
	JR #1150,ED-BLANK	A skoč opět zpět.

Nyní ošetři jakoukoli chybu.

1167 ED-FULL	LD D,#00	
	LD E,(IY-2) (RASP)	
	LD HL,#1A90	
	CALL #03B5,BEEPER	Zavrč.
	LD (IY+0),#FF (ERR-NR)	Zruš číslo chyby.
	LD DE,(#5CBA) (S-POSNL)	Vyzvedni současnou hodnotu S-POSNL a
	JR #117E,ED-C-END	skoč dopředu.

Zde je normální výstup po dokončení kopírování editačního řádku.

117C ED-C-DONE	POP DE	Hodnota nové pozice.
	POP HL	a adresa chyby.

Sem ale přijde po chybě.

117E ED-C-END	POP HL	Stará hodnota ERR-SP
	LD (#5C3D),HL (ERR-SP)	je obnovena.
	POP BC	Vyzvedni starou hodnotu S-POSNL a
	PUSH DE	uschvej hodnoty nové pozice.
	CALL #0DD9,CL-SET	Nastav systémové proměnné a starou
	POP HL	hodnotu S-POSNL dej
	LD (#5C82),HL (ECHO-E)	do ECHO-E.
	LD (IY+38),#00(X-PTR-hi)	X-PTR-hi je vynulována a je proveden
	RET	návrat.

#### PODPROGRAM NA NASTAVENÍ "HL" A "DE"

Tyto podprogramy vrací registry HL a DE tak, že HL ukazuje na první a DE na poslední lokaci editačního prostoru nebo WORKSPACE.

1190	SET-HL	LD HL,(#5C61) (WORKSP)	Ukazuj na poslední <b>lokaci</b> editační oblasti.
		DEC HL	
		AND A	Nuluj CY flag.
1195	SET-DE	LD DE,(#5C59) (E-LINE)	Ukazuj na začátek editační oblasti a
		BIT 5,(IY+55) (FLAGX)	jsi-li v editačním
		RET Z	módu, vrať se.
		LD DE,(#5C61) (WORKSP)	Jinak změň DE a
		RET C	vrať se, bylo-li to úmyslem.
		LD HL,(#5C63) (STKBOT)	Vyzvedni STKBOT
		RET	a vrať se.

#### PODPROGRAM ODSTRANĚNÍ FP

Tento podprogram odstraňuje skryté floating point formy z BASICového řádku.

11A7	REMOVE-FP	LD A,(HL)	Všechny znaky jsou postupně ohledány.
		CP #0E	Je to značka čísla?
		LD BC,#0006	Číslo zabírá šest míst.
		CALL Z,#19E8,RECLAIM-2	Zrušení FP čísla <b>pokud je třeba</b> .
		LD A,(HL)	Vyzvedni opět tento kód.
		INC HL	Posuň ukazatel.
		CP #0D	<b>Je to</b> CR? (=ENTER)
		JR NZ,#11A7,REMOVE-FP	Opakuj když ne,
		RET	jinak proved návrat.

## PROVÁDĚCÍ PODPROGRAMY

### INICIAČNÍ PODPROGRAM

Hlavní vstupní bod do tohoto podprogramu je na START/NEW (#11CB). Při vstupu ze startu (#0000), například je-li do systému při zapojení poprvé přivedeno napětí, registr A obsahuje #00 a registr DE obsahuje #FFFF. Nicméně vstupní bod může být též dosažen při vykonání příkazu NEW.

### PODPROGRAM PŘÍKAZU NEW

11B7	NEW	DI	Zakázáno maskované přerušení.
		LD A,#FF	Signál: NEW.
		LD DE,(#5CB2) (RAMTOP)	Existující hodnota systémové proměnné RAMTOP vyzvednuta.
		EXX	Zrcadlové registry
		LD BC,(#5CB4) (P-RAMT)	jsou naplněny
		LD DE,(#5C38) (RASP/PIP)	těmito systémovými
		LD HL,(#5C7B) (UDG)	proměnnými, které mají být rovněž
		EXX	uschovány.

Hlavní vstupní bod.

11CB	START/NEW	LD B,A	Uschovej signál pro
		LD A,#07	pozdější použití.
		OUT (#FE),A	Barva pro BORDER: bílá.
		LD A,#3F	
		LD I,A	Nastav registr I na hodnotu #3F.
		DEFB #00,#00,#00	Čekej 24 T stavů.
		DEFB #00,#00,#00	

Nyní se bude kontrolovat paměť.

11DA	RAM-CHECK	LD H,D	Přenes hodnotu v
		LD L,E	registrovém páru DE do HL (START = #FFFF, NEW = RAMTOP).
11DC	RAM-FILL	LD (HL),#02	Ulož hodnotu #02 do
		DEC HL	všech paměťových
		CP H	míst nad #3FFF.
		JR NZ,#11DC, RAM-FILL	
11E2	RAM-READ	AND A	Připrav se pro
		SBC HL,DE	pravdivý odečet.
		ADD HL,DE	CY flag=0 při dosažení konce,
		INC HL	Uprav ukazatel.
		JR NC,#11EF, RAM-DONE	Skoč, jsi-li na konci.
		DEC (HL)	#02 se mění na #01.
		JR Z,#11EF, RAM-DONE	Ale je-li zde nula, znamená to že paměť je vadná.
			Použij současnou hodnotu HL jako nejvyšší možnou adresu.
		DEC (HL)	#01 se mění na #00.
		JR Z,#11E2, RAM-READ	Průchod dalším testem, dokud nedošlo k selhání.
11EF	RAM-DONE	DEC HL	HL ukazuje na poslední provozuschopné místo.

Nyní budou obnoveny původní uschované systémové proměnné (což nemá žádný smysl jestliže program běží od STARTU).

		EXX	Přepni registry.
		LD (#5CB4),BC (P-RAMT)	Obnov P-RAMT
		LD (#5C38),DE (RASP/PIP)	RASP/PIP a
		LD (#5C7B),HL (UDG)	UDG.
		EXX	Přepni registry.
		INC B	Testuj signál START/NEW.
		JR Z,#1219, RAM-SET	Skoč dopředu, přicházíš-li z podprogramu NEW.

Přepiš systémové proměnné, jestliže přicházíš od STARTU, a inicializuj oblast UDG.

LD	(#5CB4),HL (P-RAMT)	To je maximální fyzická <b>adresa</b> RAM.
LD	DE,#3EAF	Poslední bajt písmene "U" ve znakovém souboru.
LD	BC,#00AB	Toto je počet bajtů pro 21 písmen.
EX	DE,HL	Přepni ukazatele.
LDDR		Nyní okopíruj znakové formy písmen "A" až "U".
EX	DE,HL	<b>Přepni zpět.</b>
INC	HL	Ukazuj na první bajt
LD	(#5C7B),HL (UDG)	a nastav UDG.
DEC	HL	0 jedno místo dolů.
LD	BC,#0040	
LD	(#5C38),BC (RASP/PIP)	Nastav systémové proměnné RASP a PIP.

Zbytek tohoto podprogramu je společný jak pro START, tak pro NEW.

1219	RAM-SET	LD	(#5CB2),HL (RAMTOP)	Nastav RAMTOP a
		LD	HL,#3C00	systémovou
		LD	(#5C36),HL (CHARS)	proměnnou CHARS.

Nyní nastav zásobník.

LD	HL,(#5CB2) (RAMTOP)	Jeho nejvyšší místo bude obsahovat
LD	(HL),#3E	hodnotu #3E.
DEC	HL	Další místo je ponecháno, aby obsahovalo nulu.
LD	SP,HL	Tato dvě místa reprezentující poslední úložku.
DEC	HL	Přeskoč dvě místa,
DEC	HL	
LD	(#5C3D),HL ( <b>ERR-SP</b> )	abys získal správnou hodnotu pro ERR-SP.

Iniciační podprogram pokračuje takto:

IM	1	Přerušovací mód IM 1.
LD	IY,#5C3A	IY obsahuje vždy hodnotu ERR-NR.
EI		<b>Maskované přerušování bude povoleno, hodiny nastaveny a každou 1/50 sec. test klávesnice.</b>
LD	HL,#5CB6	Bázová adresa
LD	(#5C4F),HL (CHANS)	oblasti kanálových informací.
LD	DE,#15AF	Počáteční kanálová data jsou přesunuta z tabulky
LD	BC,#0015	(na adrese #15AF)
EX	DE,HL	
LDIR		do oblasti kanálových informací.
EX	DE,HL	Systémová proměnná DATADD je nastavena tak, aby ukazovala
DEC	HL	na poslední místo
LD	(#5C57),HL ( <b>DATADD</b> )	oblasti kanálových dat.
INC	HL	A proměnné PROG a VARS na místo za ní.
LD	(#5C53),HL (PROG)	
LD	(#5C4B),HL (VARS)	
LD	(HL),#80	Ulož koncový znak <b>této oblasti</b> .
INC	HL	Jedno místo dopředu
LD	(#5C59),HL (E-LINE)	pro správnou hodnotu E-LINE.
LD	(HL),#0D	Zajisti, aby v editační oblasti byl pouze znak CR (=ENTER).
INC	HL	Nyní vlož další
LD	(HL),#80	koncový bajt a
INC	HL	<b>pohni</b> se znovu dopředu, abys získal
LD	(#5C61),HL (WORKSP)	hodnoty pro WORKSP,

LD	(#5C63),HL (STKBOT)	STKBOT a STKEND.
LD	(#5C65),HL (STKEND)	
LD	A,#38	Inicializuj systémové proměnné pro
LD	(#5C8D),A (ATTR-P)	barvy na: BORDER 7,
LD	(#5C8F),A (ATTR-T)	INK 0, PAPER 7,
LD	(#5C48),A (BORDCR)	BRIGHT 0 a FLASH 0.
LD	HL,#0523	Hodnota pro systémové proměnné
LD	(#5C09),HL (REPDEL)	REPDEL a REPPER.
DEC	(IY-58) (KSTATE0)	Na KSTATE0 ulož hodnotu #FF
DEC	(IY-54) (KSTATE4)	Na KSTATE4 #FF
LD	HL,#15C6	Přenes data z tab. proudových dat do
LD	DE,#5C10	informační oblasti
LD	BC,#000E	proudových dat.
LDIR		
SET	1,(IY+1) (FLAGS)	Signál: používá se tiskárna.
CALL	#0EDF,CLEAR-PRB	Nuluj bafr tiskárny.
LD	(IY+49),#02 (DF-SZ)	Nastav velikost dolní části obrazovky a smaž
CALL	#0D6B,CLS	celý displej.
XOR	A	Tiskni sdělení
LD	DE,#1538	© 1982 Sinclair Research Ltd
CALL	#0C0A,PO-MSG	na spodní řádek obrazovky.
SET	5,(IY+2) (TV-FLAG)	Signál: spodní část obrazovky bude třeba smazat.
JR	#12A9,MAIN-1	Skok do hlavní prováděcí smyčky.

#### HLAVNÍ PROVÁDĚCÍ SMYČKA

Tato smyčka se v paměti rozkládá od adresy #12A2 až do adresy #15AE a řídí editační mod, provádění přímých příkazů a výpisy sdělení.

12A2	MAIN-EXEC	LD (IY+49),#02 (DF-SZ)	Dolní část obrazovky bude velká dva řádky.
		CALL #1795,AUTO-LIST	Je proveden automatický výpis.
12A9	MAIN-1	CALL #16B0,SET-MIN	Všechny oblasti od E-LINE dopředu budou mít minimální velikost.
12AC	MAIN-2	LD A,#00	Kanál "K"
		CALL #1601,CHAN-OPEN	je otevřen.
		CALL #0F2C,EDITOR	Je zavolán editor, aby uživatel mohl vytvořit nový řádek,
		CALL #1B17,LINE-SCAN	ve kterém je následovně ošetřena syntaxe.
		BIT 7,(IY+0) (ERR-NR)	Je-li v pořádku,
		JR NZ,#12CF,MAIN-3	skok dopředu.
		BIT 4,(IY+4B) (FLAGS2)	Při použití jiného kanálu než "K" je
		JR Z,#1303,MAIN-4	proveden skok dopředu.
		LD HL,(#5C59) (E-LINE)	Ukazuj na začátek řádku kde je chyba.
		CALL #11A7,REMOVE-FP	Odstraň FP formy z tohoto řádku, nastav
		LD (IY+0),#FF (ERR-NR)	ERR-NR na "OK" a
		JR #12AC,MAIN-2	skoč na MAIN-2, aniž by se ve výpisu cokoliv změnilo.

Editovaný řádek prošel syntaxí a nyní je třeba rozlišit tři různé typy řádků.

12CF	MAIN-3	LD HL,(#5C59) (E-LINE)	Ukazuj na začátek řádku a tam také
		LD (#5C5D),HL (CH-ADD)	nastav CH-ADD.
		CALL #19FB,E-LINE-NO	Vyzvedni jakékoliv číslo řádku do BC a
		LD A,B	
		OR C	je-li to platné číslo řádku,
		JR NZ,#155D,MAIN-ADD	skoč dopředu a zařaď jej do programu.
		RST #18	Vyzvedni první znak tohoto řádku a
		CP #0D	podívej se, jestli jde o řádek nebo o pouhý CR (=ENTER).
		JR Z,#12A2,MAIN-EXEC	Skoč zpět, bylo-li tomu tak.

Editovaný řádek musí začínat přímým basicovým příkazem, takže tento se stává prvním, který bude interpretován.

BIT 0,(IY+4B) (FLAGS2)	Celá obrazovka bude
CALL NZ,#0DAF,CL-ALL	smazána v případě potřeby,
CALL #0D6E,CLS-LOWER	jinak jen spodní část.
LD A,#19	Nastav příslušné
SUB (IY+79) (S-POSN-hi)	hodnoty pro
LD (#5C8C),A (SCR-CT)	rotovací čítač.
SET 7,(IY+1) (FLAGS)	Signál: vykonání řádku.
LD (IY+0),#FF (ERR-NR)	Zajištění hodnoty "OK", na ERR-NR a
LD (IY+10),#01 (NSPPC)	ošetření prvního příkazu v řádku.
CALL #1B8A,LINE-RUN	Nyní proved tento řádek.

Poznámka: na zásobník bude uložena hodnota #1303 a bude adresována systémovou proměnnou ERR-SP.

Po interpretaci řádku a po vykonání všech z toho vyplývajících akcí se program vrací do MAIN-4, aby mohlo být vytisknuto hlášení.

1303	MAIN-4	HALT	Musí být <b>obnoveno</b> maskované přerušení.
		RES 5,(IY+1) (FLAGS)	Signál: možno další klávesu.
		BIT 1,(IY+4B) (FLAGS2)	Byl-li použit bafr tiskárny, bude
		CALL NZ,#0ECD,COPY-BUFF	vyčištěn.
		LD A,(#5C3A) (ERR-NR)	Vyzvedni číslo chyby
		INC A	a zvětš ho.
1313	MAIN-G	PUSH AF	Ušchovej si tuto novou hodnotu.
		LD HL,#0000	Nuluj systémové proměnné FLAGX,
		LD (IY+55),H (FLAGX)	X-PTR-hi a DEFADD.
		LD (IY+3B),H (X-PTR-hi)	
		LD (#5C0B),HL (DEFADD)	
		LD HL,#0001	Zajisti, aby proud #00
		LD (#5C16),HL ( <b>STRMS+6</b> )	ukazoval na kanál "K".
		CALL #16B0,SET-MIN	Vyčisti všechny pracovní oblasti a kalkulátorový
			zásobník.
		RES 5,(IY+55) (FLAGX)	Signál editační mód.
		CALL #0D6E,CLS-LOWER	Vyčisti dolní část obrazovky.
		SET 5,(IY+2) (TV-FLAG)	Dolní část obrazovky bude potřeba vyčistit.
		POP AF	Vyzvedni hodnotu hlášení,
		LD B,A	okopíruj ji do B a
		CP #0A	s čísly hlášení 0-9
		JR C,#133C,MAIN-5	skoč dopředu.
		ADD A,#07	Přičti doplněk ASCII znaku.
133C	MAIN-5	CALL #15EF,OUT-CODE	Vytiskni <b>kód</b> sdělení
		LD A,#20	
		RST #10,PRINT-A-1	a vlož mezeru.
		LD A,B	Vyzvedni znovu hodnotu sdělení
		LD DE,#1391	a použij ji k
		CALL #0C0A,PO-MSG	identifikaci sdělení.
		XOR A	
		LD DE,#1536	
		CALL #0C0A,PO-MSG	Vytiskni sdělení a udělej za ním čárku a mezeru.
		LD BC,(#5C45) (PPC)	Vyzvedni číslo aktuálního řádku a
		CALL #1A1B, <b>OUT-NUM-1</b>	také ho vytiskni.
		LD A,#3A	
		RST #10,PRINT-A-1	Dále vytiskni ":".
		LD C,(IY+13) (SUBPPC)	Vyzvedni číslo aktuálního příkazu
		LD B,#00	do registru BC a
		CALL #1A1B,OUT-NUM1	vytiskni ho.

	CALL #1097,CLEAR-SP	Vyčisti editační oblast.
	LD A,(#5C3A) (ERR-NR)	Vyzvedni opět číslo chyby a zvěšči ho jako obvykle.
	INC A	
	JR Z,#1386,MAIN-9	Skoč dopředu, nedošlo-li k chybě.
	CP #09	Byl-li program zastaven příkazem STOP či BREAK, skoč dopředu.
	JR Z,#1373,MAIN-6	
	CP #15	Jinak bude proměnná SUBPPC nezměněna.
	JR NZ,#1376,MAIN-7	
1373	INC (IY+13) (SUBPPC)	Zvyš SUBPPC na následující příkaz.
1376	LD BC,#0003	Systémové proměnné OLDPPC a OSPPC musí být upraveny tak, aby ukazovaly na číslo příkazu ve správném řádku, kde se bude pokračovat. Hodnoty, které budou použity, budou nalezeny v
	LD DE,#5C70	
	LD HL,#5C44	
	BIT 7,(HL) (NSPPC)	PPC a SUBPPC, pokud ovšem NSPPC neukazuje, že k BREAKu došlo při skoku (např. GOTO).
	JR Z,#1384,MAIN-8	
	ADD HL,BC	
1384	LDDR	
1386	LD (IY+10),#FF (NSPPC)	NSPPC je nastavena tak, aby ukazovala "Žádný skok".
	RES 3,(IY+1) (FLAGS)	Je nastaven "K" mod a je proveden skok,ale tak, aby nebyl proveden programový výpis, pokud to není třeba.
	JP #12AC,MAIN-2	

## CHYBOVÁ HLÁŠENÍ

Poslední znak každého hlášení je "invertován" (+#80).

1391	DEFB #80	- zde se překročí počáteční bajt
1392	Report 0	- "OK"
1394	Report 1	- "NEXT without FOR"
13A4	Report 2	- "Variable not found"
13B6	Report 3	- "Subscript wrong"
13C6	Report 4	- "Out of memory"
13D2	Report 5	- "Out of screen"
13DF	Report 6	- "Number too big"
13ED	Report 7	- "RETURN without GOSUB"
1401	Report 8	- "End of file"
140C	Report 9	- "STOP statement"
141A	Report A	- "Invalid argument"
142A	Report B	- "Integer out of range"
143E	Report C	- "Nonsense in BASIC"
144F	Report D	- "BREAK - CONT repeats"
1463	Report E	- "Out of DATA"
146E	Report F	- "Invalid file name"
147F	Report G	- "No room for line"
148F	Report H	- "STOP in INPUT"
149C	Report I	- "FOR without NEXT"
14AC	Report J	- "Invalid I/O device"
14BE	Report K	- "Invalid colour"
14CC	Report L	- "BREAK into program"
14DE	Report M	- "RAMTOP no good"
14EC	Report N	- "Statement lost"
14FA	Report O	- "Invalid stream"
1508	Report P	- "FN without DEF"
1516	Report Q	- "Parameter error"
1525	Report R	- "Tape loading error"

Jsou zde také ještě následující dvě sdělení:

1537	", "	- "čárka" a "mezera"
------	------	----------------------



1539 "© 1982 Sinclair Research Ltd"

1555	REPORT-G	LD	A,#10	"G" má kód "10+07+30"
		LD	BC,#0000	Nuluj BC.
		JP	#1313,MAIN-G	Skoč zpět na tisk hlášení G.

#### PODPROGRAM MAIN-ADD

Tento podprogram umožňuje basicovému řádku, aby mohl být přidán do existujícího basicového programu v programové oblasti. Existuje-li jak nová, tak i stará verze řádku se stejným číslem, pak bude starý řádek zničen. Pokud nový řádek obsahuje pouze svoje číslo, nebude do programu zařazen.

155D	MAIN-ADD	LD	(#5C49),BC (E-PPC)	Nechť je číslo nového řádku aktuálním číslem.
		LD	HL,(#5C5D) (CH-ADD)	Vyzvedni CH-ADD a
		EX	DE,HL	uschovej její adresu v DE.
		LD	HL,#1555	
		PUSH	HL	Uschovej adresu hlášení "G" na zásobník, kam ukazuje i ERR-SP.
		LD	HL,(#5C61) (WORKSP)	Vyzvedni WORKSP.
		SCF		Najdi délku řádku
		SBC	HL,DE	od místa za číslem řádku až po znak CR (=ENTER) a tuto
		PUSH	HL	délku uschovej.
		LD	H,B	Přenes číslo řádku
		LD	L,C	do HL.
		CALL	#196E,LINE-ADDR	Existuje již číslo takového řádku?
		JR	NZ,#157D,MAIN-ADD1	Skoč, když ne.
		CALL	#19B8,NEXT-ONE	Najdi délku starého
		CALL	#19EB,RECLAIM-2	řádku a znič ho.
157D	MAIN-ADD1	POP	BC	Vyzvedni délku nového řádku, ale
		LD	A,C	jednalo-li se pouze
		DEC	A	o číslo řádku +CR,
		OR	B	
		JR	Z,#15AB,MAIN-ADD2	skoč dopředu.
		PUSH	BC	Uschovej délku.
		INC	BC	Jsou zapotřebí
		INC	BC	čtyři místa navíc.
		INC	BC	Dvě pro číslo a
		INC	BC	dvě pro délku.
		DEC	HL	Nastav HL tak, aby ukazovalo před místo určení.
		LD	DE,(#5C53) (PROG)	Uschovej současnou hodnotu PROG, aby
		PUSH	DE	se zabránilo zhroucení po přidání <b>jednoho</b> řádku.
		CALL	#1655,MAKE-ROOM	Místo pro přidání jednoho řádku je vytvořeno.
		POP	HL	Stará hodnota PROG je vyzvednuta a
		LD	(#5C53),HL (PROG)	obnovena.
		POP	BC	Kopie délky řádku (bez parametrů)
		PUSH	BC	je vyzvednuta.
		INC	DE	DE ukazuje na poslední místo nové oblasti a HL na
		LD	HL,(#5C61) (WORKSP)	CR (=ENTER) v
		DEC	HL	novém řádku, který
		DEC	HL	je v editační oblasti.
		LDDR		Nyní přepokopíruj celý řádek.
		LD	HL,(#5C49) (E-PPC)	Vyzvedni číslo nového řádku.
		EX	DE,HL	Adresa určení jde do HL a číslo do DE.
		POP	BC	Vyzvedni délku nového řádku a
		LD	(HL),B	ulož její vyšší a
		DEC	HL	
		LD	(HL),C	nižší bajt.

Nyní si povšimněte, že dvoubajtová hodnota udávající číslo řádku zde na rozdíl od systému procesoru Z80 ukládána v opačném pořadí.

```

DEC HL
LD (HL),E           Nižší bajt čísla řádku !!
DEC HL
LD (HL),D           Vyšší bajt čísla řádku !!
15AB MAIN-ADD2 POP AF      Odhoď adresu hlášení "G" a po
JP #12A2,MAIN-EXEC   odkokou zpět proved i automatický výpis.

```

### POČÁTEČNÍ KANÁLOVÉ INFORMACE

Na počátku existují čtyři kanály "K", "S", "R" a "P" pro komunikaci s klávesnicí, obrazovkou, pracovním prostorem a tiskárnou. Data jsou uložena v tomto pořadí: adresa vstupního podprogramu, adresa výstupního podprogramu a kód kanálu.

```

15AF DEFW 09F4 PRINT-OUT
      DEFW 10A8 KEY-INPUT
      DEFB 4B 'K'
15B4 DEFW 09F4 PRINT-OUT
      DEFW 15C4 REPORT-J
      DEFB 53 'S'
15B9 DEFW 0F81 ADD-CHAR
      DEFW 15C4 REPORT-J
      DEFB 52 'R'
15BE DEFW 09F4 PRINT-OUT
      DEFW 15C4 REPORT-J
      DEFB 50 'P'
15C3 DEFB 80 Koncový znak.

15C4 REPORT-J RST #08,ERROR-1      Ohlaš:
              DEFB #12             J-Invalid I/O device

```

### POČÁTEČNÍ PROUDOVÁ DATA

Na počátku existuje sedm proudů: od #FD do #03.

```

15C6 DEFB #01, #00 proud #FD směřuje ke kanálu 'K'
15C8 DEFB #06, #00 proud #FE směřuje ke kanálu 'S'
15CA DEFB #0B, #00 proud #FF směřuje ke kanálu 'R'
15CC DEFB #01, #00 proud #00 směřuje ke kanálu 'K'
15CE DEFB #01, #00 proud #01 směřuje ke kanálu 'K'
15D0 DEFB #06, #00 proud #02 směřuje ke kanálu 'S'
15D2 DEFB #10, #00 proud #03 směřuje ke kanálu 'P'

```

### PODPROGRAM ČEKEJ NA KLÁVESU

Tento podprogram je řídící podprogram pro volání aktuálních vstupních podprogramů.

```

15D4 WAIT-KEY BIT 5,(IY+2) (TV-FLAG)   Není-li třeba čistit dolní část obrazovky
              JR NZ,#15DE,WAIT-KEY1   skoč dopředu.
              SET 3,(IY+2) (TV-FLAG)   Jinak signál: "posuzuj mód jako kdyby se změnil".
15DE WAIT-KEY1 CALL #15E6,INPUT-AD     Volej vstupní podprogram nepřímou přes INPUT-AD.
              RET C                    Vrať se s akceptovatelnými kódy.
              JR Z,#15DE,WAIT-KEY1    Skoč, nebyla-li stišťena žádná klávesa.

```

Jsou-li CY i Z nastaveny na nulu, znamená to, že nebylo stišťeno žádné tlačítko a došlo k nějaké chybě.

15E4	REPORT-8	RST #0B,ERROR-1	Ohlaš:
		DEFB #07	8-End of file.

#### PODPROGRAM INPUT-AD

Registry jsou uschovány a HL ukazuje na adresu vstupního podprogramu.

15E6	INPUT-AD	EXX	Uschovej registry.
		PUSH HL	
		LD HL,(#5C51) (CURCHL)	Vyzvedni bázoou adresu aktuální kanálové informace.
		INC HL	Překroč výstupní adresu a
		INC HL	
		JR #15F7,CALL-SUB	skoč dopředu.

#### HLAVNÍ TISKOVÝ PODPROGRAM

Tento podprogram je volán buď s absolutní hodnotou nebo řádným znakovým kódem v registru A.

15EF	OUT-CODE	LD E,#30	
		ADD A,E	Hodnota v registru A je zvýšena o #30.
15F2	PRINT-A-2	EXX	Uschovej registry.
		PUSH HL	
		LD HL,(#5C51) (CURCHL)	Vyzvedni bázoou adresu aktuální kanálové informace.

**Tato ukazuje na výstupní podprogram.**

Nyní volej skutečný podprogram. HL ukazuje na adresu vstupního nebo výstupního podprogramu.

15F7	CALL-SUB	LD E,(HL)	Vyzvedni nižší bajt.
		INC HL	
		LD D,(HL)	Vyzvedni vyšší bajt.
		EX DE,HL	Přenes adresu do HL.
		CALL #162C,CALL-JUMP	Zavolej požadovaný podprogram.
		POP HL	Obnov registry.
		EXX	
		RET	Odtud bude proveden návrat, pokud cestou nedošlo k chybě.

#### PODPROGRAM CHAN-OPEN

Do tohoto podprogramu vstupuje v registru A platné číslo proudu (normálně #FD až #03) a v závislosti na tomto čísle je otevřen příslušný kanál.

1601	CHAN-OPEN	ADD A,A	Hodnota v registru A je zdvojena a
		ADD A,#16	dále zvětšena o #16.
		LD L,A	Výsledek je předán do L.
		LD H,#5C	Adresa #5C16 je bázoou adresou pro proud #00.
		LD E,(HL)	Vyzvedni první a
		INC HL	
		LD D,(HL)	druhý bajt požadovaných kanálových dat.
		LD A,D	
		OR E	
		JR NZ,#1610,CHAN-OP-1	Skoč dopředu, nejsou-li oba bajty nulové, jinak
160E	REPORT-0	RST #0B,ERROR-1	ohlaš:
		DEFB #17	0-Invalid stream

S použitím proudových dat nyní najdi bázoou adresu kanálové informace, která souvisí s daným kanálem.

1610	CHAN-OP-1	DEC DE	Sniž proudová data.
------	-----------	--------	---------------------

LD HL,(#5C4F) (CHANS) Toto je bázová adresa informační oblasti kanálových dat  
 ADD HL,DE v níž bude nyní nalezena požadovaná adresa.

#### PODPROGRAM CHAN-FLAG

Tento podprogram nastavuje příslušné vlajky pro různé kanály.

1615 CHAN-FLAG LD (#5C51),HL (CURCHL) V HL je bázová adresa daného kanálu.  
 RES 4,(IY+4B) (FLAGS2) Signál: používá se jiný kanál než "K".  
 INC HL Překroč výstupní a  
 INC HL vstupní adresy.  
 INC HL  
 INC HL  
 LD C,(HL) Vyzvedni kód kanálu.  
 LD HL,#162D Toto je bázová adresa tabulky kanálových kódů.  
 CALL #16DC,INDEXER Hledej v této tabulce a urči vhodný doplněk, ale  
 RET NC vrať se v případě, že jsi nenalezl vhodný kanálový kód.  
 LD D,#00 Předej doplněk  
 LD E,(HL) do registrového páru DE a  
 ADD HL,DE skoč dopředu na  
 162C CALL-JUMP JP (HL) příslušný podprogram, nastavující flag.

#### TABULKA KANÁLOVÝCH KÓDŮ

162D	DEFB	4B	06	kanál 'K' ofset #06	adresa #1634
162F	DEFB	53	12	kanál 'S' ofset #12	adresa #1642
1631	DEFB	50	1B	kanál 'P' ofset #1B	adresa #164D
1633	DEFB	00		koncová značka.	

#### PODPROGRAM FLAG KANÁLU " K "

1634 CHAN-K SET 0,(IY+2) (TV-FLAG) Signál: užitá dolní část obrazovky.  
 RES 5,(IY+1) (FLAGS) Signál: připraven přijmout další klávesu.  
 SET 4,(IY+4B) (FLAGS2) Signál: použit kanál "S".  
 JR #1646,CHAN-S-1 Skok dopředu.

#### PODPROGRAM FLAG KANÁLU " S "

1642 CHAN-S RES 0,(IY+2) (TV-FLAG) Signál: užitá hlavní část obrazovky.  
 1646 CHAN-S-1 RES 1,(IY+1) (FLAGS) Signál: tiskárna není právě používána.  
 JP #0D4D,TEMPS Vrať se přes TEMPS, aby byly nastaveny správné barvy.

#### PODPROGRAM FLAG KANÁLU " P "

164D CHAN-P SET 1,(IY+1) (FLAGS) Signál: nyní bude použita tiskárna.  
 RET

### PODPROGRAM "VYTVOŘ PROSTOR"

Toto je velmi důležitý podprogram. Je volán při mnoha příležitostech, aby vytvořil nějaký prostor. Proto musí HL ukazovat na adresu za prvním bajtem požadovaného prostoru a v BC musí být délka požadovaného prostoru. Je-li vyžadován prostor o velikosti jediného bajtu, vstupuje se do tohoto podprogramu v bodě ONE-SPACE.

1652	ONE-SPACE	LD BC,#0001	Pouze jeden bajt.
1655	MAKE-ROOM	PUSH HL	Uschovej ukazatel.
		CALL #1F05,TEST-ROOM	Zjistí, zda je zde dost volné paměti pro požadovaný prostor.
		POP HL	Obnov ukazatel.
		CALL #1664,POINTERS	Změň všechny ukazatele, než vytvoříš žádaný prostor.
		LD HL,(#5C65) (STKEND)	Ulož do HL hodnotu STKEND.
		EX DE,HL	Zaměň "staré" a "nové".
		LDDR	Nyní vytvoř prostor
		RET	a vrať se.

Poznámka: Po návratu z tohoto podprogramu ukazuje registr HL na místo před novým prostorem a registr DE ukazuje na poslední bajt nově vytvořeného prostoru. Nový prostor může být popsán takto: od (HL)+1 až do (DE) včetně. Nicméně "nové místa" mají i své "staré adresy" a tak je možno popsat nový prostor také takto: od (HL)+2 až do (DE)+1. Ve skutečnosti se zdá, že programátor dával přednost druhému popisu, a to může být matoucí.

### PODPROGRAM "UKAZATELE"

Kdykoliv je vytvářen nebo rušen nějaký prostor, musí být všechny ukazatele, které nějak souvisí s tímto prostorem, nastaveny. Na vstupu obsahuje registrový pár BC počet bajtů, o které se bude jednat, a registrový pár HL adresu místa před novým prostorem.

1664	POINTERS	PUSH AF	Tyto registry jsou uschovány.
		PUSH HL	Kopíruj adresu "pozice".
		LD HL,#5C4B	Toto je VARS, první
		LD A,#0E	ze čtrnácti systémových ukazatelů (proměnných).

Tato smyčka sice obslouží všechny ukazatele, ale změněny budou jenom ty, které ukazují za danou "pozici".

166B	PTR-NEXT	LD E,(HL)	Vyzvedni dva bajty současného ukazatele.
		INC HL	
		LD D,(HL)	
		EX (SP),HL	Zaměň systémové proměnné s adresou "pozice".
		AND A	
		SBC HL,DE	CY flag bude, nastaven
		ADD HL,DE	jestliže má být hodnota této proměnné pozměněna.
		EX (SP),HL	Obnov "pozici".
		JR NC,#167F,PTR-DONE	Skoč dopředu, jestliže se nemá tento ukazatel měnit.
		PUSH DE	Jinak uschovej starou hodnotu.
		EX DE,HL	
		ADD HL,BC	Nyní k ní přičti hodnotu v BC
		EX DE,HL	a vlož tuto novou hodnotu do systémové proměnné.
		LD (HL),D	Nejprve vyšší bajt,
		DEC HL	
		LD (HL),E	pak nižší.
		INC HL	Ukazuj znovu na první bajt.
		POP DE	Vyzvedni starou hodnotu.
167F	PTR-DONE	INC HL	Ukazuj na další systémovou proměnnou
		DEC A	a jestliže nebylo posouzeno všech 14 syst. proměnných,
		JR NZ,#166B,PTR-NEXT	vrať se zpět do smyčky.

Nalezení velikosti bloku, který se bude přesouvat.

EX DE,HL	Stará hodnota STKEND jde do HL
POP DE	a hodnoty
POP AF	ostatních registrů jsou obnoveny.
AND A	Nyní nalezní rozdíl mezi starou hodnotou STKEND
SBC HL,DE	a "pozicí".
LD B,H	Převedl výsledek
LD C,L	do BC
INC BC	a zvýš o 1 pro bajt navíc.
ADD HL,DE	Uprav zpět na starou hodnotu STKEND
EX DE,HL	a převedl ji do DE
RET	před návratem.

#### PODPROGRAM VYZVEDNUTÍ ČÍSLA ŘÁDKU

Na vstupu ukazuje HL na místo, které má být posouzeno. Jestliže se na tomto místě nachází bajt, jehož hodnota je vyšší bajt čísla řádku, vrací se toto číslo v DE. V opačném případě se stejným způsobem otestuje lokace adresovaná DE a jestliže ani tato nespĺňuje podmínku, vrací se číslo řádku 0.

168F LINE-ZERO	DEFB #00	Číslo řádku 0.
	DEFB #00	
1691 LINE-NO-A	EX DE,HL	Posuzuj druhý ukazatel.
	LD DE,#168F	Použij řádek číslo 0.

Obvyklý vstupní bod je na LINE-NO.

1695 LINE-NO	LD A,(HL)	Vyzvedni vyšší bajt
	AND #CO	a testuj.
	JR NZ,#1691,LINE-NO-A	Skoč, je-li nevhodný.
	LD D,(HL)	Vyzvedni vyšší bajt.
	INC HL	
	LD E,(HL)	Vyzvedni nižší bajt.
	RET	Vrať se.

#### PODPROGRAM "REZERVUJ MÍSTO"

Tento podprogram je většinou volán použitím RST #30,BC-SPACES. V tomto vstupním bodě se však předpokládá, že na zásobníku je uložen WORKSP a pod ním počet míst, která mají být rezervována. Podprogram vytváří prostor mezi pracovním prostorem a zásobníkem kalkulátoru.

169E RESERVE	LD HL,(#5C63) (STKBOT)	Vyzvedni aktuální hodnotu STKBOT
	DEC HL	a zmenš ji k získání poslední lokace pracovního prostoru.
	CALL #1655,MAKE-ROOM	Vytvoř BC míst.
	INC HL	Ukazuj na druhý
	INC HL	bajt nového prostoru.
	POP BC	Vyzvedni starou hodnotu WORKSP
	LD (#5C61),BC (WORKSP)	a obnov ji.
	POP BC	Obnov BC, tedy počet míst.
	EX DE,HL	Zaměň ukazatele.
	INC HL	HL ukazuje na první z přemístěných bajtů.
	RET	Návrat.

Poznámka: Lze také tvrdit, že po návratu z podprogramu ukazuje DE na první bajt "navíc" a HL na poslední bajt "navíc", když tyto bajty navíc byly přidány za původní místo (HL)+1.

### PODPROGRAM SET-MIN

Tento podprogram nastavuje editační oblast a následné oblasti na jejich minimální hodnoty, tedy je vlastně "čistí".

16B0	SET-MIN	LD	HL,(#5C59) (E-LINE)	Vyzvedni E-LINE.
		LD	(HL),#0D	Editační oblast bude obsahovat pouze znak CR
		LD	(#5C5B),HL (K-CUR)	následovaný
		INC	HL	koncovým
		LD	(HL),#80	bajtem.
		INC	HL	<b>Další pozice bude</b>
		LD	(#5C61),HL (WORKSP)	<b>začátek WORKSP a pokračujeme čištěním.</b>

Vstup zde způsobí "vyčištění" pracovního prostoru a zásobníku kalkulátoru.

<b>16BF</b>	<b>SET-WORK</b>	LD	HL,(#5C61) (WORKSP)	Vyzvedni WORKSP.
		LD	(#5C63),HL (STKBOT)	Toto čistí pracovní prostor.

Vstup zde způsobí "vyčištění" pouze zásobníku kalkulátoru.

16C5	SET-STK	LD	HL,(#5C63) (STKBOT)	Vyzvedni STKBOT.
		LD	(#5C65),HL (STKEND)	Toto "čistí" zásobník.

Ve všech případech nastav permanentní hodnotu MEM.

	PUSH	HL		Uschovej STKEND.
	LD	HL,#5C92		Na tuto adresu
	LD	(#5C68),HL (MEM)		nastav MEM.
	POP	HL		Obnov STKEND do HL.
	RET			Vrať se.

### ZNIČENÍ EDITAČNÍHO ŘÁDKU

16D4	REC-EDIT	LD	DE,(#5C59) (E-LINE)	Vyzvedni E-LINE.
		JP	#19E5,RECLAIM-1	Proveď "zničení".

### PODPROGRAM "INDEXER"

Tento podprogram se používá ve více případech a slouží k prohledávání tabulek. Vstupní bod je INDEXER.

16DB	INDEXER-1	INC	HL	Postup na posouzení dalšího páru.
16DC	INDEXER	LD	A,(HL)	Vyzvedni první pár
		AND	A	a je-li to koncový znak #0D,
		RET	Z	vrať se.
		CP	C	Porovnej s dodaným znakem.
		INC	HL	Ukazatel na další položku.
		JR	NZ,#16DB,INDEXER-1	Skákej zpět, když nebyla nalezena správná položka.
		SCF		Při úspěšném nálezu je nastaven CY
		RET		a provede se návrat.

### PODPROGRAM PŘÍKAZU CLOSE#

Tento příkaz dává uživateli možnost zavírat proudy. Pro proudy #00 až #03 jsou jejich data neustále systémem obnovována, a proto je nelze zavřít.

16E5	CLOSE	CALL	#171E,STR-DATA	Jsou vyzvednuta existující data proudy.
------	-------	------	----------------	---

	CALL #1701,CLOSE-2	Ověř kód kanálu tohoto proudu.	
	LD BC,#0000	Připrav si nuly pro vynulování proudu.	
	LD DE,#A3E2	Připrav si hodnoty pro identifikaci	
	EX DE,HL	použití proudů #00 až #03.	
	ADD HL,DE	Testuj.	
	JR C,#16FC,CLOSE-1	Pro proudy #04 až #0F se nastaví CY a provede skok.	
	LD BC,#15D4	Jinak nalezni správné hodnoty	
	ADD HL,BC	v tabulce iniciačních proudových dat.	
	LD C,(HL)	Vyzvedni	
	INC HL	iniciační data	
	LD B,(HL)	pro proudy #00 až #03.	
16FC	CLOSE-1	EX DE,HL	Nyní vlož data, a to buď obě nuly,
	LD (HL),C	nebo	
	INC HL	iniciační	
	LD (HL),B	hodnoty	
	RET	a vrať se.	

#### PODPROGRAM CLOSE-2

Kód kanálu připojeného k zavřanému proudu musí být "K", "S", nebo "P".

1701	CLOSE-2	PUSH HL	Ušchovej adresu proudových dat.
		LD HL,(#5C4F) (CHANS)	Vyzvedni bázeovou adresu oblasti
		ADD HL,BC	kanálových informací
		INC HL	a nalezni kanálová data
		INC HL	pro proud,
		INC HL	který je právě
		LD C,(HL)	zavírán.
		EX DE,HL	Ušchovej ukazatel.
		LD HL,#1716	Báze tabulky uzavírání proudů.
		CALL #16DC,INDEXER	Nalezni správný doplněk.
		LD C,(HL)	Převeď doplněk
		LD B,#00	do BC,
		ADD HL,BC	přičti ho k bázi
		JP (HL)	a skoč do příslušného podprogramu.

#### TABULKA UZAVÍRÁNÍ PROUDŮ

1716	DEFB #4B #05	kanál 'K', doplněk #05, adresa 171C
1718	DEFB #53 #03	kanál 'S', doplněk #03, adresa 171C
171A	DEFB #50 #01	kanál 'P', doplněk #01, adresa 171C

Poznámka: Tato tabulka nemá koncový bajt.

#### PODPROGRAM ZAVŘÍ PROUD

171C	CLOSE-STR	POP HL	Vyzvedni ukazatel oblasti kanálových dat
		RET	a vrať se.

#### PODPROGRAM PROUDOVÁ DATA

Podprogram vrací proudová data pro daný proud v registrovém páru DE.

171E	STR-DATA	CALL #1E94,STK-T0-A	Číslo proudu jde ze zásobníku kalkulátoru do registru A.
		CP #10	Jestliže je číslo proudu větší než #0F,
		JR C,#1727,STR-DATA1	vyvolá se chybové hlášení.



1725	REPORT-0	RST #08,ERROR-1	Ohlaš:
		DEFB #17	0-Invalid stream.

Zde se pokračuje s platnými čísly proudů.

1727	STR-DATA1	ADD A,#03	Úprava na rozsah #03 až #12.
		RLCA	Úprava na rozsah #06 až #24.
		LD HL,#5C10	Bázová adresa oblastí proudových dat.
		LD C,A	Kode proudu
		LD B,#00	jde do BC.
		ADD HL,BC	Výpočet adresy v oblasti proudových dat
		LD C,(HL)	a přenesení
		INC HL	nalezených dat
		LD B,(HL)	do BC.
		DEC HL	Ukazatel zpět na první bajt dat.
		RET	Návrat.

#### PODPROGRAM PŘÍKAZU OPEN#

Tento příkaz poskytuje uživateli možnost otevírat proudy. Musí být udán kód kanálu, a to "K", "k", "S", "s", "P", "p", nebo "p". Povšimněte si, že není snaha přidělit proudům #00 až #03 jejich iniciační hodnoty.

1736	OPEN	RST #28,FP-CALC	Použij kalkulátor.
		DEFB #01,záměna	Záměna čísla proudu
		DEFB #38, konec výpočtu	a kód kanálu.
		CALL #171E,STR-DATA	Vyzvedni data pro proud.
		LD A,B	Jsou-li datové bajty
		OR C	nulové, znamená to že proud byl uzavřen
		JR Z,#1756,OPEN-1	a provede se skok.
		EX DE,HL	Uschovej DE.
		LD HL,(#5C4F) (CHANS)	Vyzvedni básovou adresu
		ADD HL,BC	oblasti kanálových informací
		INC HL	a nalezní
		INC HL	kód kanálu,
		INC HL	který je připojen
		LD A,(HL)	k právě otevíranému proudu.
		EX DE,HL	Obnov DE.
		CP #4B	Získaný kód musí být "K",
		JR Z,#1756,OPEN-1	
		CP #53	nebo "S",
		JR Z,#1756,OPEN-1	
		CP #50	nebo "P",
		JR NZ,#1725,REPORT-0	jinak skok na chybové hlášení.
1756	OPEN-1	CALL #175D,OPEN-2	Vyzvedni příslušná data do DE.
		LD (HL),E	Vlož tyto data
		INC HL	do dvou bajtů
		LD (HL),D	v oblasti proudových dat
		RET	a vrať se.

#### PODPROGRAM OPEN-2

Podprogram nalezne příslušné proudové datové bajty pro kanál, který je připojen k právě otevíranému proudu.

175D	OPEN-2	PUSH HL	Uschovej HL.
		CALL #2BF1,STK-FETCH	Vyzvedni parametry kódu kanálu.

	LD	A,B	Jestliže výraz
	OR	C	má nulovou hodnotu,
	JR	NZ,#1767,OPEN-3	vyvolej chybové hlášení.
1765	REPORT-F	RST #08,ERROR-1	Ohlaš:
	DEFB	#0E	F-Invalid file name.

Pokračuj zde, pokud se nevyskytla žádná chyba.

1767	OPEN-3	PUSH BC	Je uschována délka výrazu.
		LD A,(DE)	Vyzvedni první znak.
		AND #DF	Převed malá písmena na velká
		LD C,A	a přenos kódu do C.
		LD HL,#177A	Bázová adresa tabulky otevírání proudů.
		CALL #16DC,INDEXER	Vyhledej požadovaný doplněk.
		JR NC,#1765,REPORT-F	Skoč zpět, nebyl-li nalezen.
		LD C,(HL)	Převed
		LD B,#00	doplněk do BC.
		ADD HL,BC	HL nyní ukazuje na začátek příslušného podprogramu.
		POP BC	Vyzvedni délku výrazu
		JP (HL)	a skoč do příslušného podprogramu.

#### TABULKA OTVÍRÁNÍ PROUDŮ

177A	DEFB	#4B #06	kanál 'K', doplněk #06, adresa 1781
177C	DEFB	#53 #08	kanál 'S', doplněk #08, adresa 1785
177E	DEFB	#50 #0A	kanál 'P', doplněk #0A, adresa 1789
1780	DEFB	#00	koncový znak.

#### PODPROGRAM OPEN-K

1781	OPEN-K	LD E,#01	Datové bajty budou #01 a #00.
		JR #178B,OPEN-END	

#### PODPROGRAM OPEN-S

1785	OPEN-S	LD E,#06	Datové bajty budou #06 a #00.
		JR #178B,OPEN-END	

#### PODPROGRAM OPEN-P

1789	OPEN-P	LD E,#10	Datové bajty budou #10 a #00.
178B	OPEN-END	DEC BC	Zkrať délku výrazu
		LD A,B	a pokud se nejednálo
		OR C	o jediný znak,
		JR NZ,#1765,REPORT-F	skoč na chybové hlášení.
		LD D,A	Vynuluj D,
		POP HL	vyzvedni HL
		RET	a vrať se.

#### PŘÍKAZY CAT, ERASE, FORMAT & MOVE

V systému standardního SPECTRA vede použití těchto příkazů k vyvolání chybového hlášení. Podrobné informace lze najít v knize "Komentovaný výpis strojového kódu stínové ROM v Interface 1".

1793 CAT-ETC JR #1725,REPORT-0 Skok na chybové hlášení.

## PŘÍKAZY LIST & LLIST

Podprogramy v této části monitoru produkují výpis aktuálního programu. U každého řádku je posouzeno číslo, jsou expandovány tokens a umístěn aktuální programový kurzor. Vstupní bod **AUTO-LIST** je používán jak podprogramem hlavní prováděcí smyčky, tak editorem k vytvoření jedné stránky výpisu.

Ukazatel zásobníku je uschován, což umožňuje opětovné nastavení zásobníkové paměti po vykonání výpisu (viz. PO-SCR,#0C55).

1795	AUTO-LIST	LD (#5C3F),SP (LIST-SP)	Uschovej ukazatel zásobníku.
		LD (IY+2),#10 (TV-FLAG)	Signál: automatický výpis na hlavní obrazovce.
		CALL #0DAF,CL-ALL	Vyčisti tuto část obrazovky.
		SET 0,(IY+2) (TV-FLAG)	Přepni na editační oblast.
		LD B,(IY+49) (DF-SZ)	Vyčisti také
		CALL #0E44,CL-LINE	dolní část obrazovky.
		RES 0,(IY+2) (TV-FLAG)	Přepni zpět z editační oblasti.
		SET 0,(IY+48) (FLAGS2)	Signál: obrazovka je čistá.
		LD HL,(#5C49) (E-PPC)	Vyzvedni číslo aktuálního řádku
		LD DE,(#5C6C) (S-TOP)	a číslo "automatického" řádku.
		AND A	Jestliže číslo
		SBC HL,DE	aktuálního řádku
		ADD HL,DE	je menší než číslo "automatického" řádku,
		JR C,#17E1,AUTO-L-2	skoč dopředu.

Je třeba upravit "automatické" číslo tak, aby se aktuální řádek objevil na spodní části hlavní obrazovky.

PUSH DE	Uschovej "automatické" číslo.
CALL #196E,LINE-ADDR	Najdi adresu začátku aktuálního řádku
LD DE,#02C0	a vytvoř adresu zhruba
EX DE,HL	jednu "obrazovku"
SBC HL,DE	před ním a výsledek
EX (SP),HL	uschovej na zásobníku a vyzvedni číslo "automatického" řádku.
CALL #196E,LINE-ADDR	Adresa "automatického" řádku do HL.
POP BC	"Výsledek" jde do BC.

Vstup do smyčky. Při každém průchodu smyčkou je zvýšeno číslo "automatického" řádku dokud se aktuální řádek neobjeví ve výpisu.

17CE	AUTO-L-1	PUSH BC	Uschovej "výsledek".
		CALL #19B8,NEXT-ONE	Najdi adresu řádku následujícího za "automatickým" řádkem (v DE).
		POP BC	Vyzvedni výsledky,
		ADD HL,BC	proved výpočet
		JR C,#17E4,AUTO-L-3	a skoč dopředu, je-li hotovo.
		EX DE,HL	Převed adresu dalšího řádku do HL
		LD D,(HL)	a vyzvedni
		INC HL	jeho
		LD E,(HL)	číslo do DE.
		DEC HL	
		LD (#5C6C),DE (S-TOP)	Nastav S-TOP
		JR #17CE,AUTO-L-1	a pokračuj s tímto dalším řádkem.

Nyní může dojít k automatickému výpisu.

17E1	AUTO-L-2	LD	(#5C6C),HL (S-TOP)	Pro případ, že E-PPC je menší než S-TOP.
17E4	AUTO-L-3	LD	HL,(#5C6C) (S-TOP)	Vyzvedni číslo nejvyššího řádku
		CALL	#196E,LINE-ADDR	a jeho adresu.
		JR	Z,#17ED,AUTO-L-4	Nebyla-li nalezena,
		EX	DE,HL	použij adresu v DE.
17ED	AUTO-L-4	CALL	#1833,LIST-ALL	Výpis.
		RES	4,(IY+2) (TV-FLAG)	Do tohoto místa se provede návrat, pokud nebylo
		RET		zapotřebí rolování k vypsání aktuálního řádku.

#### VSTUPNÍ BOD LLIST

Je nutné otevření kanálu pro tiskárnu.

17F5	LLIST	LD	A,#03	Proud #03
		JR	#17FB,LIST-1	a skok dopředu.

#### VSTUPNÍ BOD LIST

Je nutné otevření kanálu pro obrazovku.

17F9	LIST	LD	A,#02	Proud #02
17FB	LIST-1	LD	(IY+2),#00 (TV-FLAG)	a signál: Obvyčejný výpis v hlavní části obrazovky.
		CALL	#2530,SYNTAX-Z	Při kontrole syntaxe
		CALL	NZ,#1601,CHAN-OPEN	kanál neotvírej.
		RST	#18,GET-CHAR	Vyzvedni aktuální znak do A
		CALL	#2070,STR-ALTER	a podívej se, je-li nutné měnit proud.
		JR	C,#181F,LIST-4	Skoč, když proud nebyl změněn.
		RST	#18,GET-CHAR	Vyzvedni aktuální znak do A.
		CP	#3B	Je-li to středník,
		JR	Z,#1814,LIST-2	skoč.
		CP	#2C	Je-li to čárka,
		JR	NZ,#181A,LIST-3	neskoč.
1814	LIST-2	RST	#20,NEXT-CHAR	Vyzvedni další znak a zvyš CH-ADD.
		CALL	#1C82,EXPT-1NUM	Musí následovat číselný výraz (například LIST #5,20).
		JR	#1822,LIST-5	Skoč s tímto dopředu.
181A	LIST-3	CALL	#1CE6,USE-ZERO	Použij nulu
		JR	#1822,LIST-5	a skoč dopředu.
181F	LIST-4	CALL	#1CDE,FETCH-NUM	Vyzvedni jakýkoliv řádek pokud nebyl zadán.
1822	LIST-5	CALL	#1BEE,CHECK-END	V případě kontroly syntaxe, přejdi na další příkaz.
		CALL	#1E99,FIND-INT2	Číslo řádku do BC.
		LD	A,B	Vyšší bajt do A.
		AND	#3F	Limitace vyššího bajtu na správnou velikost
		LD	H,A	a převedení celého
		LD	L,C	čísla řádku do HL.
		LD	(#5C49),HL (E-PPC)	Nastav E-PPC.
		CALL	#196E,LINE-ADDR	Najdi adresu tohoto nebo nejbližšího existujícího řádku.
1833	LIST-ALL	LD	E,#01	Signál: "před aktuálním řádkem".

Vstup do řídící smyčky pro výpis série řádků.

1835	LIST-ALL-1	CALL	#1855,OUT-LINE	Vypiš celý basicový řádek.
		RST	#10,PRINT-A-1	"Návrat vozíku".
		BIT	4,(IY+2) (TV-FLAG)	Pokud neprovádíš automatický výpis
		JR	Z,#1835,LIST-ALL-1	skoč zpět.
		LD	A,(#5C6B) (DF-SZ)	Zbývá-li ještě místo
		SUB	(IY+79) (S-POSN-hi)	v hlavní části obrazovky,
		JR	NZ,#1835,LIST-ALL-1	rovněž skoč.
		XOR	E	Jestliže obrazovka je již plná a aktuální řádek

RET Z	byl vytištěn, vrať se.
PUSH HL	Nicméně pokud aktuální řádek vytištěn nebyl,
PUSH DE	je potřeba
LD HL,#5C6C	zvýšit S-TOP
CALL #190F,LN-FETCH	a tisknout
POP DE	další řádek
POP HL	při použití rolování.
JR #1835,LIST-ALL-1	Skok zpět do smyčky.

### VYPÁSÁNÍ CELÉHO BASICOVÉHO ŘÁDKU

Na vstupu ukazuje HL na začátek řádku, tedy na vyšší bajt čísla řádku. Než bude číslo vypásáno, zjistí se, zdali řádek stojí před a nebo za aktuálním řádkem.

1855 OUT-LINE	LD BC,(#5C49) (E-PPC)	Vyzvedni číslo aktuálního řádku
	CALL #1980,CP-LINES	a porovnávej.
	LD D,#3E	Kurzor aktuálního řádku do D.
	JR Z,#1865,OUT-LINE1	Skok dopředu, tiskneš aktuální řádek.
	LD DE,#0000	Nuly do DE a E na #01, jestliže řádek stojí před aktuálním řádkem a
	RL E	E na #00, stojí-li <b>na něm nebo za</b> (CY flag přichází z CP-LINES).
1865 OUT-LINE1	LD (IY+45),E (BREG)	Ušchovej značku.
	LD A,(HL)	Vyzvedni vyšší bajt čísla řádku
	CP #40	a je-li výpis u konce,
	POP BC	zahoď návratovou adresu
	RET NC	a proved' plný návrat.
	PUSH BC	Vrať návratovou adresu
	CALL #1A28,OUT-NUM-2	a tiskni číslo řádku včetně úvodních mezer.
	INC HL	Posuň
	INC HL	ukazatel
	INC HL	na první příkazový kód v řádku.
	RES 0,(IY+1) (FLAGS)	Signál: Úvodní mezera povolena.
	LD A,D	Vyzvedni kód kurzoru
	AND A	a pokud jej není třeba tisknout,
	JR Z,#1881,OUT-LINE3	skoč dopředu.
	RST #10,PRINT-A-1	Tiskni kurzor.
187D OUT-LINE2	SET 0,(IY+1) (FLAGS)	Signál: už žádná mezera.
1881 OUT-LINE3	PUSH DE	Úschova DE.
	EX DE,HL	Ukazatel do DE.
	RES 2,(IY+48) (FLAGS2)	Signál: ne v uvozovkách.
	LD HL,#5C3B	Toto je FLAGS.
	RES 2,(HL)	Signál: tisk v modu K.
	BIT 5,(IY+55) (FLAGX)	Pokud nejsi v inputu,
	JR Z,#1894,OUT-LINE4	skoč dopředu.
	SET 2,(HL)	Signál: tisk v modu L.

Zde je vstup do smyčky, která tiskne všechny znaky až do konce basicového řádku a v případě potřeby přeskočí čísla v FP formě.

1894 OUT-LINE4	LD HL,(#5C5F) (X-PTR)	Vyzvedni ukazatel syntaktické chyby
	AND A	a v případě, že není
	SBC HL,DE	potřeba tisknout chybový znak,
	JR NZ,#18A1,OUT-LINE5	skoč dopředu.
	LD A,#3F	Nyní tiskni chybový znak.
	CALL #18C1,OUT-FLASH	Je to blikající otazník.
18A1 OUT-LINE5	CALL #18E1,OUT-CURS	Případný tisk kurzoru.
	EX DE,HL	Ukazatel do HL.
	LD A,(HL)	Vyzvedni znak.

CALL #18B6,NUMBER	Jedná-li se o FP identifikační bajt,nebude číslo vypsáno.
INC HL	Posuň ukazatel pro další průchod.
CP #0D	Je-li aktuální znak "CR",
JR Z,#18B4,OUT-LINE6	skoč dopředu.
EX DE,HL	Ukazatel do DE.
CALL #1937,OUT-CHAR	Tiskni znak.
JR #1894,OUT-LINE4	Zpět do smyčky.

Řádek byl vytisknut.

18B4	OUT-LINE6	POP DE	Obnovení DE
		RET	a návrat.

#### PODPROGRAM "NUMBER"

Jestliže registr A obsahuje příznak čísla t.j. #0E, potom bude HL ukazovat za vnitřní reprezentaci čísla.

18B6	NUMBER	CP #0E	Je to příznak čísla ?
		RET NZ	Pokud ne vrať se.
		INC HL	Posuň ukazatel o šest bajtů dopředu.
		INC HL	Tím se přeskočí příznak čísla
		INC HL	a pět bajtů vnitřní reprezentace čísla
		INC HL	v pohyblivé řádové čárce.
		INC HL	
		INC HL	
		LD A,(HL)	Před návratem vyzvedni kód za číslem.
		RET	

#### PODPROGRAM PRO TISK BLIKAJÍCÍHO ZNAKU

Tímto podprogramem jsou tisknuty kurzory - normální i chybový.

18C1	OUT-FLASH	EXX	Schovej registry.
		LD HL,(#5C8F) (ATTR-T)	Ulož ATTR-T a MASK-T
		PUSH HL	na zásobník.
		RES 7,H	Poznač blikání jako
		SET 7,L	aktivní.
		LD (#5C8F),HL	Tyto hodnoty dej jako aktivní do ATTR-T a MASK-T.
		LD HL,#5C91	Tahle adresa je P-FLAG.
		LD D,(HL)	Ulož P-FLAG
		PUSH DE	na zásobník.
		LD (HL),#00	Nastav standardní hodnoty
		CALL #09F4,PRINT-OUT	Vytiskni znak.
		POP HL	Obnov hodnotu P-FLAG.
		LD (IY+87),H (P-FLAG)	
		POP HL	Obnov hodnoty ATTR-T a MASK-T.
		LD (#5C8F),HL (ATTR-T)	
		EXX	Obnov registry.
		RET	Návrat.

#### PODPROGRAM PRO TISK KURZORU

Pokud je nastavena správná pozice pro tisk kurzoru, vytiskne se jeden z kurzorů 'C','E','G','K' nebo 'L'. V případě nesprávné pozice se provede návrat.

18E1	OUT-CURS	LD HL,(#5C5B) (K-CUR)	Vyzvedni adresu kurzoru
		AND A	a pokud není na správném

	SBC HL,DE	místě,	
	RET NZ	proved návrat.	
	LD A,(#5C41) (MODE)	Vyzvedni platnou hodnotu režimu	
	RLC A	a vynásob ji 2.	
	JR Z,#18F3,OUT-C-1	Skoč dopředu pokud není režim EXTEND nebo GRAPHICS.	
	ADD A,#43	Přičti offset - získá se znak 'E' nebo 'G'.	
	JR #1909,OUT-C-2	Jdi ho vytisknout.	
18F3	OUT-C-1	LD HL,#5C3B	Tohle je adresa FLAGS.
	RES 3,(HL)	Režim 'K'.	
	LD A,#4B	Vezmi znak "K".	
	BIT 2,(HL)	Skoč vytisknout "K", pokud je ovšem	
	JR Z,#1909,OUT-C-2	nastaven režim 'K'.	
	SET 3,(HL)	Je tedy režim 'L' nebo 'C'.	
	INC A	Uprav znak na "L".	
	BIT 3,(IY+72) (FLAGS)	Pokud je nastaven režim 'C',	
	JR Z,#1909,OUT-C-2	vezmi jako kurzor	
	LD A,#43	znak "C".	
1909	OUT-C-2	PUSH DE	Ulož DE registr.
	CALL #18C1,OUT-FLASH	Tiskni blikající znak kurzoru.	
	POP DE	<b>Obnov DE.</b>	
	RET	<b>Vrať se.</b>	

#### PODPROGRAM VYZVEDNUTÍ ČÍSLA ŘÁDKU

Na vstupu adresuje HL systémovou proměnnou S-TOP nebo E-PPC. Po návratu obsahuje S-TOP nebo E-PPC číslo následujícího řádku.

190F	LN-FETCH	LD E,(HL)	Vyzvedni
		INC HL	číslo řádku
		LD D,(HL)	z dané systémové proměnné
		PUSH HL	a uschovej ukazatel.
		EX DE,HL	Nalezená hodnota jde do HL
		INC HL	a je zvětšena.
		CALL #196E,LINE-ADDR	Je nalezena adresa tohoto, nebo následujícího řádku
		CALL #1695,LINE-NO	a jeho číslo.
		POP HL	Obnovení ukazatele na systémovou proměnnou.

Tento vstupní bod je používán editorem.

191C	LN-STORE	BIT 5,(IY+55) (FLAGX)	Pokud jsi v modu INPUT,
		RET NZ	vrať se.
		LD (HL),D	Ulož
		DEC HL	číslo řádku
		LD (HL),E	na požadovanou systémovou proměnnou
		RET	a vrať se.

#### TISK ZNAKŮ BASICOVÉHO ŘÁDKU

Opakováním voláním tohoto podprogramu je zajištěn tisk všech znaků a tokens v basicovém řádku. Vstupní bod OUT-SP-N0 se používá pro tisk čísel řádků, která vyžadují úvodní mezery.

1925	OUT-SP-2	LD A,E	A obsahuje #20 pro mezeru a #FF pro "mezeru".
		AND A	Testuj tuto hodnotu
		RET M	a vrať se, nemá-li být mezera.
		JR #1937,OUT-CHAR	Skoč na vytisknutí mezery.
192A	OUT-SP-N0	XOR A	Vynuluj A.

HL obsahuje číslo řádku a BC hodnotu pro opakované odčítání. (BC--1000, -100 nebo -10).

192B	OUT-SP-1	ADD HL,BC	Pokusné odečtení.
		INC A	Počítej každý pokus.
		JR C,#192B,OUT-SP-1	Skoky zpět do vyčerpání.
		SBC HL,BC	Vrať poslední odečet
		DEC A	a nepočítej ho.
		JR Z,#1925,OUT-SP-2	Jestliže nebylo možné provést odečítání, skoč zpět
		JP #15EF,OUT-CODE	na případný tisk mezery. Jinak vytiskni číslici.

Vstupní bod OUT-CHAR se používá pro tisk znaků, tokens a řídících znaků.

1937	OUT-CHAR	CALL #2D1B,NUMERIC	CY=0, jednalo-li se o číslici.
		JR NC,#196C,OUT-CH-3	Skoč dopředu a tiskni číslici.
		CP #21	Také všechny řídící znaky
		JR C,#196C,OUT-CH-3	a mezeru.
		RES 2,(IY+1) (FLAGS)	Signál: tisk v modu K.
		CP #CB	Jedná-li se o token THEN,
		JR Z,#196C,OUT-CH-3	skoč dopředu.
		CP #3A	Pokud se nejedná o dvojtečku,
		JR NZ,#195A,OUT-CH-1	skoč dopředu.
		BIT 5,(IY+55) (FLAGX)	Jsi-li v modu INPUT
		JR NZ,#196B,OUT-CH-2	skoč dopředu a tiskni dvojtečku.
		BIT 2,(IY+48) (FLAGS2)	Pokud dvojtečka není v uvozovkách,
		JR Z,#196C,OUT-CH-3	skoč dopředu (jedná se o oddělovač příkazů).
195A	OUT-CH-1	JR #196B,OUT-CH-2	Skoč na tisk dvojtečky, která byla v uvozovkách.
		CP #22	Akceptuj všechny znaky,
		JR NZ,#196B,OUT-CH-2	kromě " (uvozovka).
		PUSH AF	Ušchovej kód znaku.
		LD A,(#5C6A) (FLAGS2)	Otoč bit 2
		XOR #04	systémové proměnné FLAGS2.
		LD (#5C6A),A (FLAGS2)	
		POP AF	Obnov kód znaku.
196B	OUT-CH-2	SET 2,(IY+1) (FLAGS)	Signál: další znak bude vytisknut v modu L.
196C	OUT-CH-3	RST #10,PRINT-A-1	Vlastní tisk znaku.
		RET	Vrať se.

Poznámka: Je to právě toto pořadí testů, které určuje, v jakém modu má být vytištěn následující znak. Také si povšimněte, že program neobsahuje dvojtečky v řídících REM.

## ADRESA ŘÁDKU

Pro dané číslo řádku v HL vrací tento podprogram počáteční adresu tohoto nebo následujícího řádku v HL a v DE adresu předchozího řádku. Pokud hledané číslo řádku existuje bude Z flag=1, avšak při nahrazování číslem následujícího řádku bude Z flag=0.

196E	LINE-ADDR	PUSH HL	Ušchovej číslo řádku.
		LD HL,(#5C53) (PROG)	Vyzvedni hodnotu PROG
		LD D,H	a převeď ji do DE.
		LD E,L	

Nyní vstup do smyčky, která testuje čísla všech programových řádků, dokud nenalezne shodné nebo vyšší číslo.

1974	LINE-AD-1	POP BC	Dané číslo řádku.
		CALL #1980,CP-LINES	Porovnej číslo daného řádku s číslem adresovaného řádku
		RET NC	a vrať se když CY=0.
		PUSH BC	Jinak adresuj číslo
		CALL #19B8,NEXT-ONE	dalšího řádku.



EX	DE,HL	Zaměň ukazatele a skoč zpět
JR	#1974,LINE-AD-1	na posouzení dalšího programového řádku.

### PODPROGRAM POROVNÁNÍ ŘÁDKU

Číslo řádku v BC se porovná s číslem řádku jehož adresa je v HL.

1980	CP-LINES	LD A,(HL)	Vyzvedni vyšší bajt adresovaného
		CP B	čísla řádku a porovnej.
		RET NZ	Vrať se pokud se neshodují.
		INC HL	Dále porovnej
		LD A,(HL)	nižší bajt
		DEC HL	
		CP C	
		RET	a vrať se s CY=1, jestliže nebylo dosaženo hledané číslo.

### NALEZENÍ JEDNOTLIVÝCH PŘÍKAZŮ

Tento podprogram plní dvě rozličné funkce.

a) Může být použit k nalezení "D-tého" příkazu v basicovém řádku, kdy v HL bude adresa před začátkem tohoto příkazu a Z=1.

b) Vyhledá adresu příkazu, jehož token je v registru E. (Pokud takový příkaz existuje).

1988		INC HL	Nevyužito.
		INC HL	
		INC HL	
198B	EACH-STMT	LD (#5C5D),HL (CH-ADD)	Nastav CH-ADD na aktuální bajt.
		LD C,#00	Nastav signál: " uvozovky pryč ".

Vstup do smyčky, která zkoumá každý příkaz v řádku.

1990	EACH-S-1	DEC D	<b>Dekrementuj</b> D
		RET Z	a vrať se, jestliže byl nalezen požadovaný příkaz.
		RST #20,NEXT-CHAR	Vyzvedni kód dalšího znaku
		CP E	a pokud neodpovídá danému token,
		JR NZ,#199A,EACH-S-3	skoč dopředu.
		AND A	V případě shody nuluj CY
		RET	a vrať se též s nulovým Z flag.

Tato smyčka posuzuje jednotlivé znaky v řádku, aby našla kde příkaz končí.

199B	EACH-S-2	INC HL	Posuň ukazatel
		LD A,(HL)	a vyzvedni další kód.
199A	EACH-S-3	CALL #18B6,NUMBER	Překračuj všechna čísla
		LD (#5C5D),HL (CH-ADD)	a obnovuj CH-ADD.
		CP #22	Jestliže znak není uvozovka,
		JR NZ,#19A5,EACH-S-4	skoč dopředu.
		DEC C	Nastav signál: "uvozovky ano".
19A5	EACH-S-4	CP #3A	Jestliže znak je dvojtečka,
		JR Z,#19AD,EACH-S-5	skoč dopředu.
		CP #CB	Jestliže znak není token pro THEN,
		JR NZ,#19B1,EACH-S-6	skoč dopředu.
19AD	EACH-S-5	BIT 0,C	Přečti signál "uvozovky"
		JR Z,#1990,EACH-S-1	a skoč na konci každého příkazu. (A to i po THEN).
19B1	EACH-S-6	CP #0D	Pokud se nenacházíš na konci řádku,
		JR NZ,#199B,EACH-S-2	skoč zpět.

DEC	D	Zmenši čítač príkazů
SCF		a nastav CY=1.
RET		Vrať se.

#### PODPROGRAM NEXT-ONE

Tento podprogram umí nalézt další řádek v programu, nebo další proměnnou v oblasti VARS.

19B8	NEXT-ONE	PUSH HL	Ušchovej adresu aktuálního řádku nebo proměnné.
		LD A,(HL)	Vyzvedni první bajt.
		CP #40	Jestliže hledáš řádek,
		JR C,#19D5,NEXT-0-3	skoč dopředu.
		BIT 5,A	Jedná-li se o řetězec nebo pole,
		JR Z,#19D6,NEXT-0-4	skoč dopředu.
		ADD A,A	Jedná-li se o jednopísmennou číselnou proměnnou
		JP M,#19C7,NEXT-0-1	a proměnnou FOR-NEXT, skoč dopředu.
		CCF	Pouze číselné proměnné s dlouhými názvy.
19C7	NEXT-0-1	LD BC,#0005	Číselná proměnná zabírá pět míst,
		JR NC,#19CE,NEXT-0-2	
		LD C,#12	ale řídící proměnná FOR-NEXT potřebuje 18 míst.
19CE	NEXT-0-2	RLA	CY bude 0 pro proměnné s dlouhými názvy, dokud nebude
		INC HL	nalezen poslední znak. Zvyš ukazatel a
		LD A,(HL)	vyzvedni další znak.
		JR NC,#19CE,NEXT-0-2	Skok zpět, pokud nejsi na konci.
		JR #19DB,NEXT-0-5	Skok dopředu (BC=#0005 nebo #0012).
19D5	NEXT-0-3	INC HL	Krok za nižší bajt čísla řádku.
19D6	NEXT-0-4	INC HL	Nyní ukazuj na nižší bajt délky.
		LD C,(HL)	Vyzvedni
		INC HL	délku řádku
		LD B,(HL)	do BC.
		INC HL	Ukazuj na obsah Basicového řádku/proměnné.

"Další" adresa je nalezena ve všech případech.

19DB	NEXT-0-5	ADD HL,BC	Ukazuj na první bajt dalšího řádku nebo proměnné.
		POP DE	Vyzvedni adresu předchozího a pokračuj.

#### PODPROGRAM DIFFERENCE

V registru BC se vrací délka mezi dvěma ukazateli. Ukazatele se vrací zaměněné.

19DD	DIFFER	AND A	Příprava pravdivého
		SBC HL,DE	odečítání.
		LD B,H	Nalezenou délku
		LD C,L	předej do BC
		ADD HL,DE	a obnov ukazatel.
		EX DE,HL	Zaměň ukazatele
		RET	a vrať se.

#### PODPROGRAM RECLAIMING

Vstupní bod RECLAIM-1 se používá v případě, kdy DE ukazuje na první místo, které má být "zrušeno" a HL ukazuje na první místo, které má být ponecháno. Vstupní bod RECLAIM-2 se používá v případě, kdy HL ukazuje na první místo, které má být "zrušeno" a v BC je počet bajtů, které mají být "zrušeny".

19E5	RECLAIM-1	CALL #19DD,DIFFER	K získání potřebných hodnot použij DIFFERENCE.
19E8	RECLAIM-2	PUSH BC	Ušchovej počet bajtů.
		LD A,B	Všechny systémové ukazatele
		CPL	nad touto oblastí musí být
		LD B,A	sníženy o "BC",

LD A,C	a proto je jeho
CPL	obsah
LD C,A	komplementován.
INC BC	
CALL #1664,POINTERS	Vrať adresu prvního místa
EX DE,HL	do DE
POP HL	a vypočti adresu prvního místa,
ADD HL,DE	které má být ponecháno.
PUSH DE	Ušchovej adresu prvního místa
LDIR	po dobu <b>reklamace</b>
POP HL	vyzvedni ji do HL
RET	a vrať se.

#### PODPROGRAM E-LINE-NO

Podprogram vyzvedne číslo řádku v editační oblasti. Pokud řádek nemá číslo (přímý příkaz), bude se považovat za řádek 0. Ve všech případech se číslo řádku vrací v BC.

19FB E-LINE-NO	LD HL,(#5C59) (E-LINE)	Vyzvedni ukazatel editovaného řádku.
	DEC HL	Nastav ukazatel CH-ADD tak, aby ukazoval
	LD (#5C5D),HL (CH-ADD)	<b>pred</b> číslo.
	RST #20,NEXT-CHAR	Vyzvedni první kód.
	LD HL,#5C92	Předtím ale inicializuj
	LD (#5C65),HL (STKEND)	zásobník kalkulátoru.
	CALL #2D3B,INT-T0-FP	Nyní přečti číslíce čísla řádky. Z=0 pokud číslo není.
	CALL #2DA2,FP-T0-BC	Převed' číslo řádky do registru BC.
	JR C,#1A15,E-L-1	Skoč na chybu pokud je číslo řádky větší než 65535.
	LD HL,#D8F0	Jinak porovnej s číslem 10000 <b>tj.</b>
	ADD HL,BC	přičti -10000.
1A15 E-L-1	JP C,#1C8A,REPORT-C	Chybné číslo řádky ( <b>větší než 9999</b> ).
	JP #16C5,SET-STK	Obnovit kalkulátorový zásobník.

#### PODPROGRAM PRO TISK HLÁŠENÍ A ČÍSLA ŘÁDKU

Vstupní bod OUT-NUM-1 vyžaduje číslo v registru BC. Pokud bude hodnota čísla větší než 9999, nebude správně vytištěno.

Vstupní bod OUT-NUM-2 vyžaduje číslo v paměti, adresované registrem HL. I v tomto případě nelze tisknout číslo větší než 9999.

1A1B OUT-NUM-1	PUSH DE	Ulož registry.
	PUSH HL	
	XOR A	Vynuluj registr A.
	BIT 7,B	Pokud je číslo záporné
	JR NZ,#1A42,OUT-NUM-4	tiskni raději nulu.
	LD H,B	Přesuň číslo do HL.
	LD L,C	
	LD E,#FF	Poznač že zatím nebudou úvodní mezery.
	JR #1A30,OUT-NUM-3	Jdi vytisknout číslo.
1A2B OUT-NUM-2	PUSH DE	Ulož DE.
	LD D,(HL)	Vyzvedni číslo do DE.
	INC HL	
	LD E,(HL)	
	PUSH HL	<b>Ušchovej ukazatel.</b>
	EX DE,HL	Číslo do HL.
	LD E,#20	Poznač že se bude tisknout úvodní mezera.

Nyní se vytiskne celé číslo obsažené v registru HL.

1A30	OUT-NUM-3	LD BC,#FC18	Tohle je -1000.
		CALL #192A,OUT-SP-NO	Tiskni první číslici.
		LD BC,#FF9C	Tohle je -100.
		CALL #192A,OUT-SP-NO	Tiskni druhou číslici.
		LD C,#F6	Tohle je -10.
		CALL #192A,OUT-SP-NO	Tiskni třetí číslici.
		LD A,L	Vezmi zbytek do A.
1A42	OUT-NUM-4	CALL #15EF,OUT-CODE	Tiskni číslici.
		POP HL	<b>Obnov registry HL a DE.</b>
		POP DE	
		RET	<b>Návrat.</b>

## INTERPRETACE PŘÍKAZŮ V BASICOVÉM ŘÁDKU

### SYNTAKTICKÉ TABULKY

#### a) TABULKA DOPLŇKŮ:

V této tab. je doplňková hodnota pro každý z 50 Basic. příkazů.

token	příkaz	adr.	token	příkaz	adr.		
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	1AB6
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	1AF0	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	1AB1
1A5C	DEFB #2E	STOP	1A8A	1A75	DEFB#49	CLS	1ABE
1A5D	DEFB #6C	READ	1AC9	1A76	DEFB#5C	DRAW	1AD2
1A5E	DEFB #6E	DATA	1ACC	1A77	DEFB#44	CLEAR	1ABB
1A5F	DEFB #70	RESTORE	1ACF	1A78	DEFB#15	RETURN	1ABD
1A60	DEFB #48	NEW	1AAB	1A79	DEFB#5D	COPY	1AD6

#### b) TABULKA PARAMETRŮ

V této je až 8 položek pro každý z 50 Basic. příkazů. Tyto položky obsahují údaje o třídě příkazu, o požadovaných separátorech a je-li to potřeba, též adresy příkazových podprogramů.

1A7A	P-LET	DEFB #01	CLASS-01
		DEFB #3D	rovnítka 1=1
		DEFB #02	CLASS-02
1A7D	P-GO-TO	DEFB #06	CLASS-06
		DEFB #00	CLASS-00
		DEFB #67,#1E	GO-TO,1E67
1AB1	P-IF	DEFB #06	CLASS-06
		DEFB #CB	THEN
		DEFB #05	CLASS-05
		DEFB #F0,#1C	IF,1CF0
1AB6	P-GO-SUB	DEFB #06	CLASS-06
		DEFB #00	CLASS-00
		DEFB #ED,#1E	GO-SUB,1EED
1ABA	P-STOP	DEFB #00	CLASS-00
		DEFB #EE,#1C	STOP,1CEE
1ABD	P-RETURN	DEFB #00	CLASS-00
		DEFB #23,#1F	RETURN,1F23

1A90	P-FOR	DEFB #04	CLASS-04
		DEFB #3D	rovnftko 1 1 1
		DEFB #06	CLASS-06
		DEFB #CC	TO
		DEFB #06	CLASS-06
		DEFB #05	CLASS-05
		DEFB #03,#1D	FOR,1D03
1A98	P-NEXT	DEFB #04	CLASS-04
		DEFB #00	CLASS-00
		DEFB #AB,#1D	NEXT,1DAB
1A9C	P-PRINT	DEFB #05	CLASS-05
		DEFB #CD,#1F	PRINT,1FCD
1A9F	P-INPUT	DEFB #05	CLASS-05
		DEFB #89,#20	INPUT,2089
1AA2	P-DIM	DEFB #05	CLASS-05
		DEFB #02,#2C	DIM,2C02
1AA5	P-REM	DEFB #05	CLASS-05
		DEFB #B2,#1B	REM,1BB2
1AA8	P-NEW	DEFB #00	CLASS-00
		DEFB #B7,#11	NEW,11B7
1AAB	P-RUN	DEFB #03	CLASS-03
		DEFB #A1,#1E	RUN,1EA1
1AAE	P-LIST	DEFB #05	CLASS-05
		DEFB #F9,#17	LIST,17F9
1AB1	P-POKE	DEFB #08	CLASS-08
		DEFB #00	CLASS-00
		DEFB #80,#1E	POKE,1E80
1AB5	P-RANDOM	DEFB #03	CLASS-03
		DEFB #4F,#1E	RANDOMIZE,1E4F
1ABB	P-CONT	DEFB #00	CLASS-00
		DEFB #5F,#1E	CONTINUE,1E5F
1ABB	P-CLEAR	DEFB #03	CLASS-03
		DEFB #AC,#1E	CLEAR,1EAC
1ABE	P-CLS	DEFB #00	CLASS-00
		DEFB #6B,#0D	CLS,0D6B
1AC1	P-PLOT	DEFB #09	CLASS-09
		DEFB #00	CLASS-00
		DEFB #DC,#22	PLOT,22DC
1AC5	P-PAUSE	DEFB #06	CLASS-06
		DEFB #00	CLASS-00
		DEFB #3A,#1F	PAUSE,1F3A
1AC9	P-READ	DEFB #05	CLASS-05
		DEFB #ED,#1D	READ,1DED
1ACC	P-DATA	DEFB #05	CLASS-05
		DEFB #27,#1E	DATA,1E27
1ACF	P-RESTORE	DEFB #03	CLASS-03
		DEFB #42,#1E	RESTORE,1E42
1AD2	P-DRAW	DEFB #09	CLASS-09
		DEFB #05	CLASS-05
		DEFB #82,#23	DRAW,2382
1AD6	P-COPY	DEFB #00	CLASS-00
		DEFB #AC,#0E	COPY,0EAC
1AD9	P-LPRINT	DEFB #05	CLASS-05
		DEFB #C9,#1F	LPRINT,1FC9
1ADC	P-LLIST	DEFB #05	CLASS-05
		DEFB #F5,#17	LLIST,17F5
1ADF	P-SAVE	DEFB #0B	CLASS-0B
1AE0	P-LOAD	DEFB #0B	CLASS-0B

1AE1	P-VERIFY	DEFB #0B	CLASS-0B
1AE2	P-MERGE	DEFB #0B	CLASS-0B
1AE3	P-BEEP	DEFB #08	CLASS-08
		DEFB #00	CLASS-00
		DEFB #F8,#03	BEEP,03F8
1AE7	P-CIRCLE	DEFB #09	CLASS-09
		DEFB #05	CLASS-05
		DEFB #20,#23	CIRCLE,2320
1AEB	P-INK	DEFB #07	CLASS-07
1AEC	P-PAPER	DEFB #07	CLASS-07
1AED	P-FLASH	DEFB #07	CLASS-07
1AEE	P-BRIGHT	DEFB #07	CLASS-07
1AEF	P-INVERSE	DEFB #07	CLASS-07
1AF0	P-OVER	DEFB #07	CLASS-07
1AF1	P-OUT	DEFB #08	CLASS-08
		DEFB #00	CLASS-00
		DEFB #7A,#1E	OUT,1E7A
1AF5	P-BORDER	DEFB #06	CLASS-06
		DEFB #00	CLASS-00
		DEFB #94,#22	BORDER,2294
1AF9	P-DEF-FN	DEFB #05	CLASS-05
		DEFB #60,#1F	DEF-FN,1F60
1AFC	P-OPEN	DEFB #06	CLASS-06
		DEFB #2C	čárka ', '
		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,atd,1793
1B0A	P-MOVE	DEFB #0A	CLASS-0A
		DEFB #2C	čárka ', '
		DEFB #0A	CLASS-0A
		DEFB #00	CLASS-00
		DEFB #93,#17	CAT,atd,1793
1B10	P-ERASE	DEFB #0A	CLASS-0A
		DEFB #00	CLASS-00
		DEFB #93,#17	CAT,atd,1793
1B14	P-CAT	DEFB #00	CLASS-00
		DEFB #93,#17	CAT atd.,1793

Poznámka: Požadavky různých tříd jsou tyto:

CLASS-00	bez dalších operandů
01	použito pro LET, vyžaduje se proměnná
02	" , musí následovat výraz numerický či řetězcový
03	může následovat numerický výraz, <b>použije se nula pokud není</b>
04	musí následovat jednoznaková proměnná
05	může následovat soubor položek
06	musí následovat numerický výraz
07	pro obsluhu barevných položek
08	musí následovat 2 numerické výrazy, oddělené čárkou
09	jako 08, ale mohou předcházet barevné položky
0A	musí následovat řetězcový výraz
0B	obsluha kazetového MGF

## HLAVNÍ "VĚTNÝ ROZBOR" V BASICOVÉM INTERPRETU

Do tohoto podprogramu se vstupuje v bodě **LINE-SCAN** při kontrole syntaxe a v bodě **LINE-RUN**, jestliže má být proveden Basicový program, nebo jeden, či více příkazů. Každý příkaz je posuzován postupně a systémová proměnná **CH-ADD** je použita jako ukazatel na jednotlivé kódy příkazů tak, jak se vyskytují v programové, či editační zóně.

1B17	LINE-SCAN	RES 7,(IY+1) (FLAGS)	Signál: kontrola syntaxe.
		CALL #19FB, <b>E-LINE-NO</b>	CH-ADD nastaven na první znak za jakýmkoliv číslem řádku
		XOR A	SUBPPC nastav na
		LD (#5C47),A (SUBPPC)	hodnotu #00
		DEC A	a ERR-NR na #FF.
		LD (#5C3A),A (ERR-NR)	
		JR #1B29,STMT-L-1	Skoč na posouzení prvního příkazu v řádce.

## PŘÍKAZOVÁ SMYČKA

Jsou posouzeny všechny příkazy, dokud není dosaženo konce řádky.

1B28	STMT-LOOP	RST #20,NEXT-CHAR	Posun CH-ADD po řádce.
1B29	STMT-L-1	CALL #16BF,SET-WORK	Pracovní prostor je vyčištěn.
		INC (IY+13) (SUBPPC)	Každým průchodem smyčkou je zvětšena proměnná SUBPPC,
		JP M,#1C8A,REPORT-C	ale v jednom řádku je povoleno pouze 127 příkazů.
		RST #1B,GET-CHAR	Vyzvedni znak
		LD B,#00	a vyčisti registr B.
		CP #0D	Je to CR? (ENTER).
		JR Z,#1BB3,LINE-END	ANO: skoč.
		CP #3A	Je to ":"?
		JR Z,#1B28,STMT-LOOP	ANO: vrať se do smyčky.

Příkaz je identifikován, takže nyní se posoudí jeho 1.část.

LD HL,#1B76	Návratová adresa STMT-RET
PUSH HL	je uložena na <b>TOP</b> .
LD C,A	Ulož si přechodně příkaz do registru C,
RST #20,NEXT-CHAR	zatímco opět posuneš CH-ADD
LD A,C	redukuji kód příkazu.
SUB #CE	Konst. #CE,tím dosáhneš rozsah od 0 do #31 pro 50 příkazů
JP C,#1C8A,REPORT-C	Chyba, není-li kód příkazem.
LD C,A	Kód příkazu do registrového páru <b>BC</b> .
LD HL,#1A48	Bázová adresa tabulky syntaxe.
ADD HL,BC	Požadovaný doplněk
LD C,(HL)	je předán do registru C
ADD HL,BC	a použít k výpočtu báz. adr. pro položky v tab. parametrů
JR #1B55,GET-PARAM	a s touto adresou skoč do prohlížečí smyčky.

Všechny programy pro posouzení třídy daného příkazu jsou postupně vykonány a všechny požadované separátory jsou též posouzeny.

1B52	SCAN-LOOP	LD HL,(#5C74) (T-ADDR)	Přechodný ukazatel na položku v tabulce parametrů.
1B55	GET-PARAM	LD A,(HL)	Vyzvedni postupně všechny položky.
		INC HL	Posuň ukazatel
		LD (#5C74),HL (T-ADDR)	na položku pro další průchod.
		LD BC,#1B52	Ulož na <b>TOP</b> návratovou
		PUSH BC	adresu SCAN-LOOP.
		LD C,A	Okopíruj položku do C.



CP	#20	Je položka separátor?
JR	NC,#1B6F,SEPARATOR	ANO:skoč.
LD	HL,#1C01	Bázová adresa tabulky příkazových tříd.
LD	B,#00	Vynuluj registr B a
ADD	HL,BC	hledej v této tabulce
LD	C,(HL)	a doplněk do registru C.
ADD	HL,BC	Vypočítej startovní adresu požadovaného podprogramu
PUSH	HL	a ulož ji na TOP.
RST	#1B,GET-CHAR	Před nepřímým skok. do podpr. pro zjišťování tříd příkazů
DEC	B	předej příkazový kód do reg.A a nastav reg.B na #FF.
RET		

#### PODPROGRAM SEPARÁTOR

Hlášení "Nonsense in BASIC" je vypsanó, jestliže požadovaný separátor není přítomen. Ale povšimněte si, že při kontrole syntaxe není toto hlášení tištěno, pouze se objeví blikající "?".

1B6F	SEPARATOR	RST #1B,GET-CHAR	Aktuální znak je
		CP C	vyzvednut a porovnán s parametrem v tabulce.
		JP NZ,#1C8A,REPORT-C	Chyba při neshodě.
		RST #20,NEXT-CHAR	Postup na správný znak
		RET	a vrať se.

#### PODPROGRAM STMT-RET

Po správné interpretaci příkazu se program navrácí na toto místo.

1B76	STMT-RET	CALL #1F54,BREAK-KEY	Po každém příkazu je otestována klávesa BREAK.
		JR C,#1B7D,STMT-R-1	NE: skok.
1B7B	REPORT-L	RST #0B,ERROR-1	ANO: ohlaš
		DEFB #14	L-BREAK into program.

Zde se pokračuje, nebyl-li BREAK stisknut.

1B7D	STMT-R-1	BIT 7,(IY+10) (NSPPC)	Není-li potřeba žádný "skok",
		JR NZ,#1BF4,STMT-NEXT	pokračuj na NEXT.
		LD HL,(#5C42) (NEWPPC)	Číslo nového řádku
		BIT 7,H	a pokud neobsluhuješ další příkaz v editační oblasti,
		JR Z,#1B9E,LINE-NEW	skoč dopředu.

#### VSTUPNÍ BOD LINE-RUN

Tento vstupní bod je používán, kdykoliv má být řádek v editační oblasti spuštěn příkazem RUN. V tomto případě bude nastaven bit 7 ve FLAGS jako signál, že se provádí RUN (a ne SYNTAX). Tento vstupní bod je též používán při kontrole řádku v editační oblasti, který obsahuje víc než jeden příkaz (bit 7 FLAGS=0).

1B8A	LINE-RUN	LD HL,#FFFE	Řádek v editační oblasti je posuzován jako
		LD (#5C45),HL (PPC)	řádek "-2".
		LD HL,(#5C61) (WORKSP)	HL=koncový byte v
		DEC HL	editační oblasti
		LD DE,(#5C59) (E-LINE)	a DE na
		DEC DE	byte před editační oblastí.
		LD A,(#5C44) (NSPPC)	Vezmi číslo dalšího příkazu, který má být zpracován,
		JR #1BD1,NEXT-LINE	než skočíš dopředu.

## PODPROGRAM LINE NEW

V programu byl proveden skok a počáteční adresa nového řádku bude nalezena.

1B9E	LINE-NEW	CALL #196E,LINE-ADDR	Je nalezena adresa následujícího řádku.
		LD A,(#5C44) (NSPPC)	Vežmi číslo příkazu.
		JR Z,#1BBF,LINE-USE	Skoč, byl-li požadovaný řádek nalezen, jinak zkontroluj
		AND A	platnost čísla příkazu,
		JR NZ,#1BEC,REPORT-N	musí to být 0.
		LD B,A	Zkontroluj není-li
		LD A,(HL)	"následující" řádek
		AND #CO	za skutečným
		LD A,B	"koncem" programu.
		JR Z,#1BBF,LINE-USE	Skoč s platnými adresami.
1BB0	REPORT-0	RST #0B,ERROR-1	<b>Jinak</b> ohlaš:
		DEFB #FF	0-OK

Poznámka: toto není chyba v normálním smyslu chápání, ale skok **na konec programu**.

## PODPROGRAM PŘÍKAZU REM

Návratová adresa do STMT-RET je odložena, čímž je program donucen ignorovat **konec** řádku.

1BB2	REM	POP BC	Odhoď adresu STMT-RET.
------	-----	--------	------------------------

## PODPROGRAM "KONEC ŘÁDKU"

Při kontrole syntaxe se provede jednoduchý návrat, ale pokud je program v běhu, musí být zkontrolována adresa na NXTLIN než bude použita.

1BB3	LINE-END	CALL #2530,SYNTAX-Z	Při kontrole syntaxe
		RET Z	se vrať,
		LD HL,(#5C55) (NXTLIN)	jinak vyzvedni adresu na NXTLIN a
		LD A,#CO	je-li tato za
		AND (HL)	koncem programu
		RET NZ	vrať se též, čímž je příkaz RUN vykonán.
		XOR A	Signál: příkaz 0, než postoupíš dále.

## PODPROGRAM POUŽITÍ ŘÁDKU

Tato krátký podprogram má 3 funkce:

- změní příkaz "0" na "1"
- najde číslo nového řádku a vloží do systémové proměnné PPC
- vytvoří adresu začátku následujícího řádku

1BBF	LINE-USE	CP #01	Z příkazu "0" se
		ADC A,#00	stává příkaz "1".
		LD D,(HL)	Číslo aktuálního
		INC HL	řádku se vyzvedne
		LD E,(HL)	
		LD (#5C45),DE (PPC)	a předá do PPC.
		INC HL	Nyní je nalezena
		LD E,(HL)	délka řádku.
		INC HL	
		<b>LD D,(HL)</b>	
		EX DE,HL	Záměna hodnot a

ADD HL,DE	vytvoření adresy následujícího řádku v HL <b>příčemž</b>
INC HL	pozice před 1.znakem následujícího řádku je v DE.

### PODPROGRAM DALŠÍ ŘÁDEK

Na vstupu ukazuje HL za poslední místo "dalšího" řádku a DE ukazuje na místo před prvním bajtem tohoto řádku. Týká se řádků, které jsou v programové oblasti a též řádků v editační oblasti, kdy "další řádek" je ten samý řádek a zbyvajjí zde ještě příkazy, které je nutno přeložit.

1BD1	NEXT-LINE	LD (#5C55),HL (NXTLIN)	Nastav NXTLIN pro <b>další</b> použití.
		EX DE,HL	CH-ADD ukazuje jako obvykle na pozici před znakem,
		LD (#5C5D),HL (CH-ADD)	který má být posouzen.
		LD D,A	Číslo příkazu je vyzvednuto.
		LD E,#00	Nulován E při použití EACH-STMT.
		LD (IY+10),#FF (NSPPC)	Signál: Žádný skok.
		DEC D	Číslo příkazu - 1
		LD (IY+13),D (SUBPPC)	jde do SUBPPC.
		JP Z,#1B28,STMT-LOOP	Nyní může být posouzen 1.příkaz.
		INC D	Avšak pro další
		CALL #198B,EACH-STMT	příkaz musí být nalezena počáteční adresa.
		JR Z,#1BF4,STMT-NEXT	Existuje-li - skoč.
1BEC	REPORT-N	RST #0B,ERROR-1	Ohlaš:
		DEFB #16	N-Statement lost

### PODPROGRAM "KONTROLA KONCE"

Toto je velmi důležitý podprogram, který je volán z mnoha míst monitorového programu při kontrole syntaxe, nebo editačního řádku. Úkolem tohoto podprogramu je vypsat chybové hlášení, jestliže nebyl nalezen konec příkazu a přesunout se na další příkaz, byla-li syntaxe O.K.

1BEE	CHECK-END	CALL #2530,SYNTAX-Z	
		RET NZ	Nepokračuj, dokud nekontroluješ syntaxi.
		POP BC	Odhoď adresy SCAN-LOOP
		POP BC	a STMT-RET před pokračováním do STMT-NEXT.

### PODPROGRAM "DALŠÍ PŘÍKAZ"

Jestliže aktuální znak je CR (ENTER), pak "další příkaz" je na "dalším řádku". Je-li to ":", je další příkaz na tom samém, ale je-li zde jakýkoliv jiný znak, jedná se o syntaktickou chybu.

1BF4	STMT-NEXT	RST #1B,GET-CHAR	Vyzvedni aktuální znak a jedná-li se o
		CP #0D	ENTER, jdi
		JR Z,#1BB3,LINE-END	na "další řádek".
		CP #3A	Je-li to ":" jdi
		JP Z,#1B28,STMT-LOOP	na "další příkaz",
		JP #1C8A,REPORT-C	jinak je syntaktická chyba.

### TABULKY TŘÍD PŘÍKAZŮ

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

## PŘÍKAZOVÉ TŘÍDY 00, 03 A 05

Příkazy, které mají třídy 03 mohou, ale nemusí být následovány číslem (např. RUN a RUN 00)

1COD CLASS-03 CALL #1CDE,FETCH-NUM Číslo je zjištěno, ale neexistuje-li použije se 0.

Příkazy třídy 00 nesmí být následovány operandy (COPY ap.)

1C10 CLASS-00 CP A Nastav Z flag pro pozdější využití.

Příkazy třídy 05 mohou být následovány parametry (PRINT ap.)

1C11 **CLASS-05** POP BC Ve všech případech odhod adresu SCAN-LOOP.  
CALL Z,#1BEE,CHECK-END

U přík.tříd 00 a 03 se po kontrole syntaxe pokračuje v posouzení dalšího příkazu.

EX DE,HL Do DE uschovej ukazatel řádku.

## PODPROGRAM \*SKOK C-R\*

Po posouzení všech tříd příkazů a všech oddělovacích znaků se provede skok do příslušného příkazového podprogramu.

1C16 JUMP-C-R LD HL,(#5C74) (T-ADDR) Vyzvedni ukazatel na položku v tabulce parametrů a  
LD C,(HL) vyzvedni adresu  
INC HL požadovaného příkazového  
LD B,(HL) podprogramu.  
EX DE,HL Zaměň ukazatele a  
PUSH BC proveď nepřímý  
RET skok do příkazového podprogramu.

## PŘÍKAZOVÉ TŘÍDY 01,02 A 04

Tyto třídy jsou používány pro příkazy, které pracují s proměnnými (LET, FOR,.. a nepřímo též READ, INPUT). Příkaz třídy 01 je volán při identifikaci proměnné pro přík. LET, READ a INPUT.

1C1F CLASS-01 CALL #28B2,LOOK-VARS Urči, zda proměnná již byla či nebyla definována.

## PROMĚNNÁ V PŘIDĚLENÍ

Tento podprogram vytváří příslušné hodnoty pro systémové proměnné DEST a STRELEN.

1C22 VAR-A-1 LD (IY+55),#00 (FLAGX) Nastav FLAGX.  
JR NC,#1C30,VAR-A-2 Skoč, když proměnná už existuje.  
SET 1,(IY+55) (FLAGX) Signál: nová proměnná.  
JR NZ,#1C46,VAR-A-3 Ohlaš chybu, jestliže dochází k pokusu užít nedim. pole.  
1C2E REPORT-2 RST #0B,ERROR-1 Ohlaš:  
DEFB #01 2-Variable not found

Zde pokračuj, jestliže obsluhuješ existující proměnnou.

1C30 VAR-A-2 CALL Z,#2996,**STK-VAR** Parametry řetězců a všech polí jsou předány na **kalkul.**  
BIT 6,(IY+1) (FLAGS) zásobník (**STK-VAR** odkrojí konec řetězce,je-li to třeba)  
JR NZ,#1C46,VAR-A-3 Skoč dopředu, obsluhuješ-li číselnou proměnnou.  
XOR A Vynuluj registr A.  
CALL #2530,**SYNTAX-Z** Parametry řetězce nebo řetězcového pole jsou vyzvednuty  
CALL NZ,#2BF1,STK-FETCH v případě, že se nekontroluje syntaxe.  
LD HL,#5C71 Toto je FLAGX.

OR	(HL)	Bit 0 je nastaven v tom případě, kdy se jedná o úplný
LD	(HL),A	jednoduchý řetězec, což signalizuje: smazat starou kopii.
EX	DE,HL	HL nyní ukazuje na řetězec nebo část pole.

Obě cesty programu se zde opět stýkají, aby STRLEN a DEST byly nastaveny na požadované hodnoty. Pro všechny numerické proměnné a nové řetězce a řetězcová pole obsahuje STRLEN-lo znak názvu proměnné. Ale pro staré řetězce a řetězcová pole, ať už jsou seřiznuty nebo kompletní, obsahuje tato systémová proměnná délku přidělení.

1C4E VAR-A-3 LD (#5C72),BC (STRLEN) Nastav STRLEN na požadovanou hodnotu.

DEST obsahuje "cílovou" adresu "staré" proměnné, ale zároveň i "zdrojovou" adresu "nové" proměnné.

LD	(#5C4D),HL (DEST)	Nastav DEST na požadovanou hodnotu
RET		a vrať se.

Příkazy třídy 02 jsou doplněny aktuálním výpočtem hodnoty, která je přidělena v příkazu LET.

1C4E	CLASS-02	POP BC	Adresa SCAN-LOOP je odhozena.
		CALL #1C56,VAL-FET-1	Je provedeno přidělení.
		CALL #1BEE,CHECK-END	Přesun na další příkaz, buď přes CHECK-END při kontrole
		RET	syntaxe, nebo přes STMT-RET při programovém běhu.

#### PODPROGRAM "VYZVEDNI HODNOTU"

Tento podprogram se využívá pro LET, READ a INPUT, aby ohodnotil a přidělil hodnoty deklarovaným proměnným. Vstupní bod VAL-FET-1 je používán příkazy LET a READ a posuzuje FLAGS, zatímco vstupní bod VAL-FET-2 je využíván příkazem INPUT a posuzuje FLAGX.

1C56	VAL-FET-1	LD A,(#5C3B) (FLAGS)	Použij FLAGS.
1C59	VAL-FET-2	PUSH AF	Ušchovej hodnocení dalšího příkazu.
		CALL #24FB,SCANNING	Proveď hodnocení dalšího příkazu.
		POP AF	Vyzvedni starou hodnotu FLAGS nebo FLAGX.
		LD D,(IY+1) (FLAGS)	Vyzvedni novou hodnotu FLAGS.
		XOR D	Typ proměnné numerické nebo řetězcové se musí shodovat
		AND #40	s typem výrazu,
		JR NZ,#1C8A,REPORT-C	není-li tomu tak, vypiš hlášení C.
		BIT 7,D	Pokud nekontroluješ syntaxi,
		JP NZ,#2AFF,LET	skoč dopředu k provedení vlastního přidělení,
		RET	jinak se vrať.

#### PŘÍKAZY TŘÍDY 04

Tento vstupní bod se používá pro příkazy FOR a NEXT.

1C6C	CLASS-04	CALL #28B2,LOOK-VARS	Zjistí zda je tato proměnná používána.
		PUSH AF	Ušchovej registrový pár AF
		LD A,C	zatímco budeš testovat
		OR #9F	diskriminační bajt.
		INC A	
		JR NZ,#1C8A,REPORT-C	Ověř si také, že se jedná o smyčku FOR NEXT.
		POP AF	Obnov podmínkový registr a skoč zpět,
		JR #1C22,VAR-A-1	abys zjistil, že nalezená proměnná bude přidělena.

## PODPROGRAM "OČEKÁVEJ ČÍSELNÝ NEBO ŘETĚZCOVÝ VÝRAZ"

Toto jsou sledy krátkých podprogramů, které jsou používány k vyzvednutí výsledků dalšího ohodnoceného výrazu. Výsledek jednoduchého výrazu je vrácen jako poslední hodnota na zásobníku kalkulátoru. Vstupní bod NEXT-2NUM se používá, když je třeba upravit CH-ADD tak, aby ukazovala na začátek prvního výrazu.

1C79 NEXT-2NUM RST #20,NEXT-CHAR Posuň CH-ADD.

Vstupní bod EXPT-2NUM (odpovídá třídě 8) umožňuje dvěma číselným výrazům odděleným čárkou, aby byly ohodnoceny.

1C7A EXPT-2NUM CALL #1C82,EXPT-1NUM Ohodnoť oba výrazy po sobě.  
(CLASS-08)  
CP #2C Není-li separátor čárka,  
JR NZ,#1C8A,REPORT-C ohlaš chybu.  
RST #20,NEXT-CHAR **Posuň CH-ADD**

Vstupní bod **EXPT-1NUM** (odpovídá třídě 6) umožňuje ohodnocení **jednoho** číselného výrazu.

1C82 EXPT-1NUM CALL #24FB,SCANNING Ohodnoť další výraz.  
(CLASS-06)  
BIT 6,(IY+1) (FLAGS) Byl-li výsledek číselný  
RET NZ vrať se, jinak  
1C8A REPORT-C RST #0B,ERROR-1 ohlaš:  
DEFB #0B C-Nonsense in BASIC.

Vstupní bod EXPT-EXP (odpovídá třídě 0A) umožňuje ohodnocení **jednoho** řetězcového výrazu.

1C8C EXPT-EXP CALL #24FB,SCANNING Ohodnoť další výraz.  
(CLASS-0A)  
BIT 6,(IY+1) (FLAGS) Je-li indikován řetězec,  
RET Z vrať se,  
JR #1C8A,REPORT-C jinak ohlaš chybu.

## PODPROGRAM "NASTAVENÍ PERMANENTNÍCH BAREV" (= TŘÍDA 07)

Tento podprogram umožňuje nastavení aktuálních přechodných barev. Jakožto příkaz třídy 07 představuje ve skutečnosti příkazový podprogram pro šest barevných příkazů.

1C96 PERMS BIT 7,(IY+1) (FLAGS) Je testována **vlažka** "syntaxe/run".  
**(CLASS-07)** RES 0,(IY+2) (TV-FLAG) Signál: hlavní obrazovka.  
CALL NZ,#0D4D,TEMPS V případě RUN uprav podle přech. barev barvy obrazovky.  
POP AF Odhoď návratovou adresu SCAN-LOOP.  
LD A,(#5C74) (T-ADDR) Vyzvedni nižší bajt T-ADDR a  
SUB #13 odečti #13 - rozsah #D9 až #DE což jsou INK až OVER.  
CALL #21FC,CO-TEMP-4 Skoč dopředu k výměně barev podle basicového příkazu.  
CALL #1BEE,CHECK-END Přesuň se na další příkaz, jestliže kontroluješ syntaxi.  
LD HL,(#5C8F) (ATTR-T) Nyní se přechodné barvy stanou  
LD (#5C8D),HL (ATTR-P) permanentními. A to jak ATTR-T tak i MASK-P.  
LD HL,#5C91 Toto je P-FLAG a ten  
LD A,(HL) musí být také posouzen.

Následující instrukce velmi rafinovaně okopírují liché bity na místa sudých bajtů, což ve skutečnosti způsobí, že permanentní barvy budou mít hodnoty přechodných barev.

RLCA Posuň masku doleva a  
XOR (HL) výsledek nechť se projeví pouze na  
AND #AA lichých bitech tohoto bajtu.

XOR (HL)	
LD (HL),A	Obnov výsledek
RET	<b>a vrať se.</b>

### PODPROGRAM PŘÍKAZŮ TŘÍDY 09

Tento podprogram se používá pro příkazy PLOT, DRAW a CIRCLE, aby určil konstantní hodnoty pro "FLASH 8, BRIGHT 8 a PAPER 8", které jsou nastaveny ještě předtím, než budou posouzeny případné barevné položky.

1CBE CLASS-09	CALL #2530,SYNTAX-Z	
	JR Z,#1CD6,CL-09-1	Skoč dopředu při kontrole syntaxe.
	RES 0,(IY+2) (TV-FLAG)	Signál: hlavní obrazovka.
	CALL #0D4D,TEMPS	Nastav přechodné barvy pro hlavní obrazovku.
	LD HL,#5C90	Toto je MASK-T.
	LD A,(HL)	Vyzvedni jeho hodnotu a
	OR #F8	vezmi z ní pouze část odpovídající nemaskovanému INK.
	LD (HL),A	Obnov hodnotu, která indikuje FLASH 8,INK 8,PAPER 8.
	RES 6,(IY+8) (P-FLAG)	Také zajisti, že nebude PAPER 9.
	RST #18,GET-CHAR	Vyzvedni znak před obsluhou dalších barevných položek.
1CD6 CL-09-1	CALL #21E2,CO-TEMP	Zpracuj místně dominantní barevnou položku.
	JR #1C7A,EXPT-2NUM	Nyní vyzvedni první dva operandy pro PLOT, DRAW a CIRCLE.

### PŘÍKAZY TŘÍDY 0B

Tento podprogram je používán příkazy SAVE, LOAD, VERIFY a MERGE.

1CDB CLASS-0B	JP #0605, <b>SAVE-ETC</b>	Skoč do kazetových podprogramů.
---------------	---------------------------	---------------------------------

### PODPROGRAM "VYZVEDNUTÍ ČÍSLA"

Tento podprogram způsobí, že následující číselný výraz bude vyhodnocen, ale jestliže neexistuje, bude nahrazen nulou.

1CDE FETCH-NUM	CP #0D	Jedná-li se o konec řádku
	JR Z,#1CE6,USE-ZERO	skoč dopředu.
	CP #3A	Nejde-li o konec příkazu
	JR NZ,#1C82,EXPT-1NUM	skoč na EXPT-1NUM.

Nyní je použit kalkulátor, aby uložil nulu na zásobník.

1CE6 USE-ZERO	CALL #2530,SYNTAX-Z	
	RET Z	Neprováděj tuto operaci při kontrole syntaxe.
	RST #28,FP-CALC	Použij kalkulátor.
	DEFB #A0, <b>stk-nula</b>	"Poslední hodnota" je nyní nula.
	DEFB #38,konec výpočtu	
	RET	<b>Návrat s nulou na zásobníku kalkulátoru.</b>

## PŘÍKAZOVÉ PODPROGRAMY

Tato sekce 16k monitorového programu od adresy #1CEE až do adresy #23FA obsahuje většinu příkazových programů pro basicový interpret.

### **PŘÍKAZ "STOP"**

Podprogram příkazu STOP obsahuje pouze volání chybového hlášení.

1CEE STOP	RST #08,ERROR-1	Ohlaš:
(REPORT-9)	DEFB #08	9-STOP statement

### **PŘÍKAZ "IF"**

Hodnota výrazu mezi IF a THEN se stane "poslední hodnotou" na kalkulátorovém zásobníku. Je-li logicky pravdivá, bude se posuzovat další výraz a v ostatních případech je řádek považován za skončený.

1CF0 IF	POP BC	Odhoď návratovou adresu STMT-RET.
	CALL #2530,SYNTAX-Z	
	JR Z,#1D00,IF-1	Skoč dopředu při kontrole syntaxe.

Nyní použij kalkulátor k vymazání poslední položky na zásobníku kalkulátoru, ale nech DE, aby adresoval první bajt této hodnoty.

	RST #28,FP-CALC	Použij kalkulátor.
	DEFB #02,výmaz	Poslední hodnota je vymazána.
	DEFB #38,konec výpočtu	
	EX DE,HL	HL ukazuje na první bajt a
	CALL #34E9,TEST-ZERO	je zavolán TEST-ZERO.
	JP C,#1BB3,LINE-END	Byla-li hodnota "nepravda", skoč na další řádek.
1D00 IF-1	JP #1B29,STMT-L-1	Ale byla-li "pravda" skoč na další příkaz po THEN.

### **PŘÍKAZ "FOR"**

Do tohoto příkazového podprogramu se vstupuje s hodnotami VALUE a LIMIT již na zásobníku kalkulátoru.

1D03 FOR	CP #CD	Pokud není udán krok
	JR NZ,#1D10,F-USE-1	skoč dopředu.
	RST #20,NEXT-CHAR	Posuň CH-ADD a
	CALL #1C82,EXPT-1NUM	vyzvedni hodnotu pro krok.
	CALL #1BEE,CHECK-END	Při kontrole syntaxe se posuň na další příkaz, jinak
	JR #1D16,F-REORDER	skoč dopředu.

Nebyl-li specifikován krok, použij hodnotu "1".

1D10 F-USE-1	CALL #1BEE,CHECK-END	Další příkaz jestliže kontroluješ syntaxi, jinak
	RST #28,FP-CALC	použij kalkulátor a ulož
	DEFB #A1,stk-jedna	hodnotu "1" na jeho zásobník.
	DEFB #38,konec výpočtu	

Tři hodnoty na zásobníku kalkulátoru jsou nyní VALUE(v), LIMIT(l) a STEP(s). Tyto hodnoty budou nyní posouzeny.

1D16 F-REORDER	RST #28,FP-CALC	v,l,s
	DEFB #C0,st-mem-0	v,l,s (paměť č.0*s)
	DEFB #02,výmaz	v,l



DEFB #01,výměna	l,v
DEFB #E0,get-mem-0	l,v,s
DEFB #01,výměna	l,s,v
DEFB #38,konec výpočtu	

Řídící proměnná FOR je nyní vytvořena a považována za přechodnou oblast kalkulátorové paměti.

CALL #2AFF,LET	Proměnná je nalezena nebo vytvořena (má hodnotu v).
LD (#5C68),HL (MEM)	Udělej z ní paměťovou oblast.

Proměnná, která byla nalezena může být pouze jednoduchá číselná proměnná používající pouze šest míst a nyní pro ni bude vyhrazeno místo.

DEC HL	
LD A,(HL)	Vyzvedni jednopísmenný název proměnné.
SET 7,(HL)	Zajisti, že bit 7 v názvu je nastaven.
LD BC,#0006	Bude třeba šest míst.
ADD HL,BC	HL nechť ukazuje za ně.
RLCA	Rotuj název a
JR C,#1D34,F-L&S	skoč byla-li to již proměnná FOR.
LD C,#0D	V opačném případě
CALL #1655,MAKE-ROOM	vytvoř dalších 13 míst.
INC HL	HL ukazuje opět za pozici LIMIT.

Počáteční hodnoty pro LIMIT a STEP jsou nyní sečteny.

1D34 F-L&S	PUSH HL	Je uschován ukazatel.
	RST #28,FP-CALC	l,s
	DEFB #02,výmaz	l
	DEFB #02,výmaz	-
	DEFB #38,konec výpočtu	DE stále ukazuje na "1".
	POP HL	Ukazatel je obnoven a
	EX DE,HL	ukazatele zaměněny.
	LD C,#0A	Deset bajtů LIMITU a STEPu je
	LDIR	přesunuto.

Nyní jsou vložena čísla "smýčkovacího" řádku a příkazu.

LD HL,(#5C45) (PPC)	Číslo aktuálního řádku.
EX DE,HL	Zaměň registry
LD (HL),E	před přidáním
INC HL	čísla řádku
LD (HL),D	k řídící proměnné FOR.
LD D,(IY+13) (SUBPPC)	"Smýčkovací" příkaz je vždy
INC D	dalším příkazem
INC HL	ať již existuje,
LD (HL),D	nebo ne.

Podprogram NEXT-LOOP se volá aby testoval možnost "průběhu" a jestliže je to možné, provede se návrat. Jinak se identifikuje příkaz za smýčkou FOR-NEXT.

CALL #1DDA,NEXT-LOOP	Je "průběh" možný?
RET NC	Jestliže ano vrat se.
LD B,(IY+56) (STRLEN-lo)	Vyzvedni název proměnné.
LD HL,(#5C45) (PPC)	Kopíruj číslo aktuálního řádku
LD (#5C42),HL (NEWPPC)	do NEWPPC.
LD A,(#5C47) (SUBPPC)	Vyzvedni číslo aktuálního příkazu
NEG	a kompletuj ho.

LD	D,A	Převeď výsledek do D.
LD	HL,(#5C5D) (CH-ADD)	Vyzvedni hodnotu CH-ADD.
LD	E,#F3	Bude se hledat "NEXT".

Nyní se vzestupně prohledává programová oblast, dokud není nalezen první NEXT následovaný správnou proměnnou.

1D64	F-LOOP	PUSH BC	Uschovej název proměnné.
		LD BC,(#5C55) (NXTLIN)	Vyzvedni aktuální hodnotu NXTLIN.
		CALL #1D86,LOOK-PROG	Programová oblast je prohledávána
		LD (#5C55),BC (NXTLIN)	a BC se bude měnit s každým novým řádkem.
		POP BC	Obnov název proměnné.
		JR C,#1D84,REPORT-I	Nebylo-li <b>NEXT</b> nalezeno, skoč na chybové hlášení.
		RST #20,NEXT-CHAR	Přejdi za nalezené NEXT.
		OR #20	Akceptuj malá i velká písmena.
		CP B	Testuj název proměnné
		JR Z,#1D7C,F-FOUND	a skoč dopředu v případě shody.
		RST #20,NEXT-CHAR	Opět posuň <b>CH-ADD</b> a nejedná-li se o správnou proměnnou,
		JR #1D64,F-LOOP	skoč zpět.

NEWPPC obsahuje číslo řádku, ve kterém byl nalezen příkaz NEXT. Musí být nalezena pozice příkazu v řádku a uložena v NSPPC.

1D7C	F-FOUND	RST #20,NEXT-CHAR	Posuň CH-ADD.
		LD A,#01	Počítadlo příkazů v registru D počítalo příkazy od nuly
		SUB D	a proto musí být odečteno od jedné.
		LD (#5C44),A (NSPPC)	Výsledek je uschován.
		RET	Vrať se do STMT-RET.
1D84	REPORT-I	RST #08,ERROR-1	Ohlaš:
		DEFB #11	I-FOR without NEXT

#### PODPROGRAM "LOOK-PROG"

Tento podprogram se používá k vyhledávání příkazů DATA, DEF FN nebo NEXT. Na vstupu je příslušný kód tokenu v registru E a HL ukazuje na začátek prohledávané oblasti.

1D86	LOOK-PROG	LD A,(HL)	Vyzvedni aktuální znak,
		CP #3A	je-li to dvojtečka, (více příkazů v řádku)
		JR Z,#1DA3,LOOK-P-2	skoč dopředu.

Vstup do smyčky, která prozkoumá každý další řádek.

1D8B	LOOK-P-1	INC HL	Vyzvedni vyšší bajt čísla řádku a
		LD A,(HL)	nejsou-li v programu další řádky
		AND #C0	
		SCF	
		RET NZ	vrať se s CY=1.
		LD B,(HL)	Číslo řádku je vyzvednuto a
		INC HL	
		LD C,(HL)	
		LD (#5C42),BC (NEWPPC)	předáno do NEWPPC.
		INC HL	
		LD C,(HL)	
		INC HL	
		LD B,(HL)	Pak je vyzvednuta jeho délka.
		PUSH HL	Ukazatel je uschován
		ADD HL,BC	zatímco adresa konce řádku
		LD B,H	

	LD C,L	je vytvořena v BC.
	POP HL	Ukazatel je obnoven.
	LD D,#00	Nastav počítadlo příkazů na nulu.
1DA3 LOOK-P-2	PUSH BC	Ukazatel na konec řádku je uschován
	CALL #198B,EACH-STMT	zatímco jsou prozkoumávány příkazy v řádku.
	POP BC	<b>Obnovení ukazatele.</b>
	RET NC	Vrať se nebyl-li zde žádný výskyt,
	JR #1D8B,LOOK-P-1	jinak posuzuj další řádek.

#### PŘÍKAZ "NEXT"

Proměnná "v přidělení" byla již stanovena (viz TŘÍDA 04,#1C6C) a zbývá změnit její hodnotu, jak je požadováno.

1DAB NEXT	BIT 1,(IY+55) (FLAGX)	Nebyla-li proměnná nalezena
	JP NZ, #1C2E,REPORT-2	ohlaš chybu.
	LD HL,(#5C4D) (DEST)	Adresa proměnné je vyzvednuta a
	BIT 7,(HL)	její název je dále testován.
	JR Z, #1DDB,REPORT-1	<b>Pokud je špatný typ proměnné, ohlaš chybu.</b>

VALUE (hodnota) a STEP (krok) jsou upravovány pomocí kalkulátoru.

INC HL	Překroč jméno a
LD (#5C68),HL (MEM)	vytvoř z proměnné přechodnou "paměťovou oblast".
RST #28,FP-CALC	-
DEFB #E0,get-mem-0	v
DEFB #E2,get-mem-2	v,s
DEFB #0F,sčítání	v+s
DEFB #C0,st-mem-0	v+s
DEFB #02,výmaz	-
DEFB #38,konec výpočtu	-

Výsledek sečtení hodnot VALUE + STEP je nyní porovnáán s hodnotou LIMIT voláním podprogramu NEXT-LOOP.

CALL #1DDA,NEXT-LOOP	Testuj novou hodnotu VALUE proti hodnotě LIMIT.
RET C	Byla-li smyčka FOR-NEXT již dokončena, vrať se.

Jinak vyzvedni "číslo smyčkovacího řádku a příkazu".

LD HL,(#5C68) (MEM)	Nalezni adresu
LD DE,#000F	nižšího bajtu
ADD HL,DE	čísla "smyčkovacího řádku".
LD E,(HL)	Nyní toto číslo
INC HL	
LD D,(HL)	vyzvedni <b>do DE.</b>
INC HL	
LD H,(HL)	Vyzvedni také číslo příkazu.
EX DE,HL	Zaměň čísla před
JP #1E73,G0-T0-2	skokem dopředu, kde budou zpracovány jako G0-T0.

1DDB REPORT-1	RST #0B,ERROR-1	Ohlaš:
	DEFB #00	1-NEXT without FOR

## PODPROGRAM "NEXT-LOOP"

Tento podprogram testuje, byl-li překročen LIMIT hodnotou VALUE. Musí se přihlídnout ke znaménku hodnoty STEP. Tento podprogram nastaví CY flag na hodnotu 1, byl-li LIMIT překročen.

```
1DDA NEXT-LOOP  RST #28,FP-CALC      -
                 DEFB #E1,get-mem-1  l
                 DEFB #E0,get-mem-0  l,v
                 DEFB #E2,get-mem-2  l,v,s
                 DEFB #36,<0         l,v,(1 nebo 0)
                 DEFB #00,skok-pravda l,v,(1 nebo 0)
                 DEFB #02,na NEXT-1  l,v,(1 nebo 0)
                 DEFB #01,záměna     v,1

1DE2 NEXT-1     DEFB #03,odečítání   v-l nebo l-v
                 DEFB #37,>0         (1 nebo 0)
                 DEFB #00,skok-pravda (1 nebo 0)
                 DEFB #04,na NEXT-2  -
                 DEFB #38,konec výpočtu -
                 AND A               Vyčisti CY flag a
                 RET                 vrať se-smýčka je možná.
```

Ovšem, není-li možné smýčku provést, nastav CY flag.

```
1DE9 NEXT-2     DEFB #38,konec výpočtu -
                 SCF                 Nastav CY flag a
                 RET                 vrať se.
```

## PODPROGRAM PŘÍKAZU READ

Příkaz READ umožňuje načítání seznamů dat a ve skutečnosti je podobný sérii příkazů LET. Systémová proměnná X-PTR se využívá k úschově ukazatele na příkaz READ, zatímco CH-ADD slouží ke krokování seznamu DATA.

```
1DEC READ-3     RST #20,NEXT-CHAR    Posun na další znak a jeho vyzvednutí v A.
1DED READ       CALL #1C1F,CLASS-01  Zjištění zdali proměnná existuje a její vyhledání.
                 CALL #2530,SYNTAX-Z  Při kontrole syntaxe
                 JR Z,#1E1E,READ-2    skok dopředu.
                 RST #18,GET-CHAR     Vyzvedni znak.
                 LD (#5C5F),HL (X-PTR) Úschova CH-ADD v X-PTR.
                 LD HL,(#5C57) (DATADD) Vyzvedni ukazatel na seznam DATA,
                 LD A,(HL)            načti další znak
                 CP #2C               a pokud nebyl nalezen nový příkaz DATA,
                 JR Z,#1E0A,READ-1    skoč dopředu.
                 LD E,#E4             Hledá se
                 CALL #1D86,LOOK-PROG token DATA.
                 JR NC,#1E0A,READ-1   Skoč dopředu při úspěšném nález.
1E0B REPORT-E   RST #0B,ERROR-1     Ohlaš:
                 DEFB #0D             E-Out of DATA
```

Pokračování - vybírání hodnot ze seznamu DATA.

```
1E0A READ-1     CALL #0077,TEMP-PTR1   Posunuj ukazatel po seznamu a nastav CH-ADD.
                 CALL #1C56,VAL-FET-1  Vyzvedni hodnotu a přiděl jí proměnné.
                 RST #18,GET-CHAR     Vyzvedni aktuální hodnotu CH-ADD
                 LD (#5C57),HL (DATADD) a ulož ji v DATADD.
                 LD HL,(#5C5F) (X-PTR) Vyzvedni ukazatel příkazu READ
                 LD (IY+38),#00 (X-PTR-hi) a vynuluj X-PTR.
                 CALL #0078,TEMP-PTR2  CH-ADD ještě jednou ukazuje na příkaz READ.
```

1E1E	READ-2	RST #18,GET-CHAR	Vyzvedni aktuální znak
		CP #2C	a testuj na čárku.
		JR Z,#1DEC,READ-3	Je-li to čárka, skoč zpět, protože existují další položky.
		CALL #1BEE,CHECK-END	Při kontrole syntaxe se vrať přes CHECK-END,
		RET	anebo do STMT-RET.

#### PODPROGRAM PŘÍKAZU DATA

Při kontrole syntaxe je příkaz DATA testován, zdali obsahuje sérii platných výrazů, oddělených čárkami. Ale při běhu programu se tento příkaz vynechá.

1E27	DATA	CALL #2530,SYNTAX-Z	Pokud nekontroluješ syntaxi,
		JR NZ,#1E37,DATA-2	skoč dopředu.
1E2C	DATA-1	CALL #24FB,SCANNING	Kontroluj další výraz
		CP #2C	a správný oddělovač.
		CALL NZ,#1BEE,CHECK-END	Posuň se na další příkaz, pokud nesouhlasí.
		RST #20,NEXT-CHAR	Další znak.
		JR #1E2C,DATA-1	Pokračuj ve smyčce.

Při běhu programu se příkaz data překročí.

1E37	DATA-2	LD A,#E4	Token DATA.
------	--------	----------	-------------

#### PODPROGRAM VYNECHÁNÍ

Na vstupu je v registru A kód příkazu, který má být vynechán. (DATA nebo DEF FN, podle momentálního použití tohoto podprogramu).

1E39	PASS-BY	LD B,A	BC ať obsahuje nějaké dosti vysoké číslo.
		CPDR	Vzestupně hledání token.
		LD DE,#0200	Nyní najdi
		JP #198B,EACH-STMT	následující příkaz. (D mínus prvý).

#### PODPROGRAM PŘÍKAZU RESTORE

Operand pro RESTORE se považuje za číslo řádku a pokud není uveden, dosadí se nula. Vstupní bod REST-RUN se použije za chodu.

1E42	RESTORE	CALL #1E99,FIND-INT2	Kompresuj operand do BC.
1E45	REST-RUN	LD H,B	Převeď výsledek
		LD L,C	do HL.
		CALL #196E,LINE-ADDR	Najdi adresu tohoto nebo následujícího řádku.
		DEC HL	DATADD bude ukazovat
		LD (#5C57),HL (DATADD)	na jednu pozici před řádek.
		RET	Hotovo - vrať se.

#### PODPROGRAM PŘÍKAZU RANDOMIZE

Operand je kompresován do BC a uložen do příslušné systémové proměnné. Není-li operand uveden, použijí se hodnoty z FRAMES 1 a FRAMES 2.

1E4F	RANDOMIZE	CALL #1E99,FIND-INT2	Vyzvedni operand do BC
		LD A,B	a pokud se nejedná
		OR C	o nulovou hodnotu,
		JR NZ,#1E5A,RAND-1	skoč dopředu.
		LD BC,(#5C78) (FRAMES)	Vyzvedni dva nižší bajty FRAMES
1E5A	RAND-1	LD (#5C76),BC (SEED)	a ulož je do systémové proměnné SEED.

RET

Vrať se.

#### PODPROGRAM PŘÍKAZU CONTINUE

Příslušná čísla řádku a příkazu se stanou předmětem skoku.

1E5F	CONTINUE	LD HL,(#5C6E) (OLDPPC)	Číslo řádku.
		LD D,(IY+54) (OSPPC)	Číslo příkazu.
		JR #1E73,G0-T0-2	Skok dopředu.

#### PODPROGRAM PŘÍKAZU GO TO

Operandem musí být číslo řádku v rozsahu 1 až 9999. Ale vzhledem k "obrácené" logice číslování se testuje na číslo 61439.

1E67	G0-T0	CALL #1E99,FIND-INT2	Vyzvedni operand
		LD H,B	převed jej
		LD L,C	do HL.
		LD D,#00	Nastav číslo příkazu na nulu.
		LD A,H	Jestliže hodnota
		CP #F0	přesahuje 61439,
		JR NC,#1E9F,REPORT-B	skoč na chybové hlášení.

Vstupní bod G0-T0-2 se používá v četných případech při určování čísla následujícího řádku.

1E73	G0-T0-2	LD (#5C42),HL (NEWPPC)	Vlož číslo řádku
		LD (IY+10),D (NSPPC)	a číslo příkazu.
		RET	Vrať se.

#### PODPROGRAM PŘÍKAZU OUT

Oba parametry jsou vyzvednuty ze zásobníku kalkulátoru a vyslány.

1E7A	OUT	CALL #1E85,TWO-PARAM	Vyzvedni operandy.
		OUT (C),A	Skutečná instrukce OUT.
		RET	Hotovo.

#### PODPROGRAM PŘÍKAZU POKE

POKE se provede obdobně.

1E80	POKE	CALL #1E85,TWO-PARAM	Vyzvedni operandy.
		LD (BC),A	Skutečná instrukce POKE.
		RET	Hotovo.

#### PODPROGRAM TWO-PARAM

První parametr na zásobníku kalkulátoru musí být kompresovatelný do jednoduchého registru a je-li záporný, bude komplementován. Druhý parametr na zásobníku kalkulátoru musí být kompresovatelný do registrového páru.

1E85	TWO-PARAM	CALL #2DD5,FP-T0-A	Vyzvedni parametr
		JR C,#1E9F,REPORT-B	a skoč na chybu, je-li příliš vysoký.
		JR Z,#1E8E,TWO-P-1	Skoč dopředu s kladnými čísly,
		NEG	ale komplementuj záporná čísla.
1E8E	TWO-P-1	PUSH AF	Ušchovej první parametr
		CALL #1E99,FIND-INT2	a vyzvedni druhý.
		POP AF	Vyzvedni první parametr

RET

a vrať se.

### PODPROGRAMY NALEZENÍ INTEGERŮ

"Poslední hodnota" na zásobníku kalkulátoru je vyzvednuta a kompresována do jednoduchého registru nebo registrového páru.

1E94	FIND-INT1	CALL #2DD5,FP-TO-A	Vyzvedni hodnotu do A
		JR #1E9C,FIND-I-1	a skoč.
1E99	FIND-INT2	CALL #2DA2,FP-TO-BC	Vyzvedni hodnotu do BC
1E9C	FIND-I-1	JR C,#1E9F,REPORT-B	a v obou případech skoč na chybu, je-li CY = 1.
		RET Z	Vrať se s pozitivními čísly, která jsou v rozsahu.
1E9F	REPORT-B	RST #08,ERROR-1	Ohlaš:
		DEFB #0A	B-Integer out of range

### PODPROGRAM PŘÍKAZU RUN

Parametry příkazu RUN jsou předány do NEWPPC voláním příkazového podprogramu GO TO. Před návratem se provede RESTORE 0 a CLEAR 0.

1EA1	RUN	CALL #1E67,GO-TO	Nastav NEWPPC na hodnotu operandu.
		LD BC,#0000	
		CALL #1E45,REST-RUN	Proveď RESTORE 0.
		JR #1EAF,CLEAR-1	Výstup přes příkaz CLEAR.

### PODPROGRAM PŘÍKAZU CLEAR

Podprogram způsobí vyčištění oblasti proměnných, obrazovky a nastavení RAMTOP. Následkem této poslední operace je znovu nastaven zásobník a tím také vyčištěn zásobník GO SUB.

1EAC	CLEAR	CALL #1E99,FIND-INT2	Vyzvedni operand a použij 0, neexistuje-li.
1EAF	CLEAR-RUN	LD A,B	Pokud operand
		OR C	není nulový,
		JR NZ,#1EB7,CLEAR-1	skoč dopředu. (Při volání z RUN, ke skoku nedojde).
		LD BC,(#5CB2) (RAMTOP)	Použij stávající hodnotu RAMTOP.
1EB7	CLEAR-1	PUSH BC	Ušchovej ji.
		LD DE,(#5C4B) (VARS)	Zruš
		LD HL,(#5C59) (E-LINE)	oblast
		DEC HL	
		CALL #19E5,RECLAIM-1	proměnných.
		CALL #0D6B,CLS	Vyčisti obrazovku.

Hodnota v BC je testována na přípustné minimum a maximum.

		LD HL,(#5C65) (STKEND)	Aktuální hodnota zásobníku
		LD DE,#0032	je zvýšena
		ADD HL,DE	o 50 bajtů,
		POP DE	což bude představovat
		SBC HL,DE	dolní limit.
		JR NC,#1EDA,REPORT-M	RAMTOP je příliš nízký.
		LD HL,(#5CB4) (P-RAMT)	Je vyzvednuta hodnota P-RAMT
		AND A	a testována
		SBC HL,DE	proti RAMTOP.
		JR NC,#1EDC,CLEAR-2	Skok dopředu, je-li hodnota přijatelná.
1EDA	REPORT-M	RST #08,ERROR-1	Ohlaš:
		DEFB #15	M-RAMTOP no good

Pokračuj operací CLEAR.

1EDC	<b>CLEAR-2</b>	EX DE,HL	Skutečně
		LD (#5CB2),HL (RAMTOP)	nastavení RAMTOP.
		POP DE	Vyzvedni adresu STMT-RET.
		POP BC	Vyzvedni "chybovou" adresu.
		LD (HL),#3E	Vlož koncový znak pro zásobník GO SUB.
		DEC HL	Vynech jedno místo.
		LD SP,HL	Nastav ukazatel zásobníku na prázdný zásobník GO SUB.
		PUSH BC	Ulož "chybovou" adresu na zásobník
		LD (#5C3D),SP (ERR-SP)	a uschovej jeho adresu do ERR-SP.
		EX DE,HL	Zaměň adresy
		JP (HL)	a proveď nepřímý návrat do STMT-RET.

Poznámka: Je-li CLEAR volán příkazem RUN, je třeba najít další řádek skokem do STMT-RET, protože hodnoty NEWPPC a NSPPC byly již změněny.

#### PODPROGRAM PŘÍKAZU GO SUB

Aktuální hodnota PPC a zvýšená hodnota SUBPPC jsou uloženy na zásobník GO SUB.

1EED	GO-SUB	POP DE	Vyzvedni návratovou adresu k uschování v DE.
		LD H,(IY+13) (SUBPPC)	Vyzvedni číslo příkazu
		INC H	a zvětši jej.
		EX (SP),HL	Zaměň "chybovou adresu" s číslem příkazu
		INC SP	a reklamuj tuto lokaci.
		LD BC,(#5C45) (PPC)	Dále ulož
		PUSH BC	číslo aktuálního řádku.
		PUSH HL	Vrať "chybovou" adresu na zásobník
		LD (#5C3D),SP (ERR-SP)	a nastav ERR-SP, aby na ní ukazoval.
		PUSH DE	Vrať adresu STMT-RET.
		CALL #1E67,GO-T0	Nyní nastav NEWPPC & NSPPC na potřebné hodnoty.
		LD BC,#0014	Připrav hodnotu na testování prostoru.

#### PODPROGRAM TESTOVÁNÍ PROSTORU

Provádí řadu testů je-li dostatečný prostor volné paměti pro vykonání prováděného úkolu.

1F05	TEST-ROOM	LD HL,(#5C65) (STKEND)	Vezmi hodnotu STKEND
		ADD HL,BC	a zvyš ji o hodnotu vstupující v BC.
		JR C, #1F15,REPORT-4	Skoč na chybové hlášení, došlo-li k přetečení.
		EX DE,HL	Další pokus
		LD HL, #0050	s přičtením
		ADD HL,DE	80 bajtů.
		JR C, #1F15,REPORT-4	Skoč na chybové hlášení, došlo-li k přetečení.
		SBC HL,SP	Testuj proti adrese zásobníku
		RET C	a vrať se, pokud vyhovuje.
1F15	<b>REPORT-4</b>	LD L, #03	Toto je chyba za běhu a proto nikdy
		JP #0055,ERROR-3	nebude použit chybový znak (otazník)

#### ZJIŠTĚNÍ VOLNÉ PAMĚTI

SPECTRUM sice nemá basicový příkaz "FREE", ale protože zde existuje tento podprogram, lze velikost volné paměti zjistit použitím : PRINT 65536-USR 7962.

1F1A	FREE-MEM	LD BC, #0000	Nepovoluje se "nic navíc".
		CALL #1F05,TEST-ROOM	Vlastní test.
		LD B,H	Převezení



LD C,L	do BC
RET	a návrat.

#### PODPROGRAM PŘÍKAZU RETURN

Ze zásobníku GO SUB se vyzvedne číslo řádku a příkazu.

1F23 RETURN	POP BC	Vyzvedni adresu STMT-RET.
	POP HL	Vyzvedni "chybovou" adresu.
	POP DE	Vyzvedni poslední položku ze zásobníku GO SUB.
	LD A,D	Testuj, jestli se jedná
	CP #3E	o koncový znak zásobníku GO SUB,
	JR Z, #1F36,REPORT-7	a v tom případě skoč do podprogramu chybových hlášení.
	DEC SP	Celý údaj potřebuje tři bajty.
	EX (SP),HL	Zaměň číslo příkazu za "chybovou" adresu
	EX DE,HL	a převed do DE.
	LD (#5C3D),SP (ERR-SP)	Nastav chybový ukazatel.
	PUSH BC	Vrať adresu STMT-RET na zásobník.
	JP #1E73,GO-T0-2	Skok na změnu NEWPPC & NSPPC.
1F36 REPORT-7	PUSH DE	Vlož koncový znak
	PUSH HL	a chybovou adresu.
	RST #0B,ERROR-1	Ohlaš:
	DEFB #06	7-RETURN without GOSUB

#### PODPROGRAM PŘÍKAZU PAUSE

Trvání pauzy je dáno čítáním maskovaných přerušení, ke kterým dochází každou 1/50 tinu sekundy. PAUSE je ukončena buď po načtení příslušného počtu přerušení, nebo při stisku některé klávesy.

1F3A PAUSE	CALL #1E99,FIND-INT2	Vyzvedni operand.
1F3D PAUSE-1	HALT	Čekej na maskované přerušení.
	DEC BC	Zmenši čítač.
	LD A,B	Je-li nulový,
	OR C	příkaz PAUSE
	JR Z, #1F4F,PAUSE-END	končí.
	LD A,B	Jestliže byl operand nula,
	AND C	BC bude obsahovat #FFFF
	INC A	a tato je opět upravena na nulu.
	JR NZ,#1F49,PAUSE-2	Skok se všemi ostatními hodnotami.
	INC BC	
1F49 PAUSE-2	BIT 5,(IY+1) (FLAGS)	Nebyla-li stišťena žádná klávesa,
	JR Z,#1F3D,PAUSE-1	skok zpět.
1F4F PAUSE-END	RES 5,(IY+1) (FLAGS)	Signál: nebyla stišťena žádná klávesa.
	RET	Návrat do STMT-RET.

#### PODPROGRAM BREAK

Podprogram zjistí stišťení klávesy BREAK, což signalizuje nastavením CY=0.

1F54 BREAK-KEY	LD A, #7F	Vytvoř adresu portu #7FFE
	IN A,(#FE)	a načti bajt.
	RRA	Testuj pouze bit 0, jeho rotací do CY.
	RET C	Vrať se, pokud klávesa BREAK nebyla stišťena.
	LD A, #FE	Vytvoř adresu portu #FEFE
	IN A,(#FE)	a načti bajt.
	RRA	Testuj pouze bit 0, jeho rotací do CY.
	RET	Vrať se s CY=0, pokud došlo k současnému stišťení.

## PODPROGRAM PŘÍKAZU DEF FN

Při kontrole syntaxe se testuje, zdali má příkaz správnou formu. Také se vytvoří prostor pro výsledek ohodnocení funkce. Ale pokud je program v běhu, tento příkaz se vynechá.

1F60	DEF-FN	CALL #2530,SYNTAX-Z	Při kontrole syntaxe
		JR Z, #1F6A,DEF-FN-1	skok dopředu.
		LD A, #CE	Jinak
		JP #1E39,PASS-BY	vynechání příkazu.

Nejdříve se posuzuje proměnná této funkce.

1F6A	DEF-FN-1	SET 6,(IY+1) (FLAGS)	Signál: číselná proměnná.
		CALL #2C8D,ALPHA	Je aktuální znak v registru A písmeno?
		JR NC,#1F89,DEF-FN-4	Skok dopředu, jestliže ne.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CP #24	Nejedná-li se o \$,
		JR NZ,#1F7D,DEF-FN-2	skoč dopředu.
		RES 6,(IY+1) (FLAGS)	Změň bit 6, protože se jedná o řetězcovou proměnnou.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
1F7D	DEF-FN-2	CP #28	Závorka musí následovat za názvem proměnné. "(".
		JR NZ,#1FBD,DEF-FN-7	Skok na chybu.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CP #29	Test na závorku ")".
		JR Z,#1FA6,DEF-FN-6	Skoč, jestliže funkce nemá parametry.

Zde se vstupuje do smyčky, která postupně ošetří všechny parametry.

1F86	DEF-FN-3	CALL #2C8D,ALPHA	Je aktuální znak v registru A písmeno?
1F89	DEF-FN-4	JP NC,#1C8A,REPORT-C	Skok na chybu, jestliže ne.
		EX DE,HL	Uchovej ukazatel v DE.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CP #24	Nejedná-li se o \$,
		JR NZ,#1F94,DEF-FN-5	skoč dopředu.
		EX DE,HL	Uchovej nový ukazatel v DE.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
1F94	DEF-FN-5	EX DE,HL	Předej ukazatel na poslední písmeno názvu do HL
		LD BC,#0006	a vytvoř prostor 6 bajtů
		CALL #1655,MAKE-ROOM	za tímto posledním znakem názvu.
		INC HL	Do první
		INC HL	z nových lokací
		LD (HL),#0E	ulož znak uvádějící FP číslo.
		CP #2C	Není-li aktuální znak čárka,
		JR NZ,#1FA6,DEF-FN-6	opuť smyčku, protože nejsou další parametry.
		RST #20,NEXT-CHAR	Vyzvedni další znak
		JR #1F86,DEF-FN-3	a skoč zpět pro další parametry.

Dále je posouzena definice funkce.

1FA6	DEF-FN-6	CP #29	Kontrola, že závorka ")" skutečně existuje.
		JR NZ,#1FBD,DEF-FN-7	Skok na chybu.
		RST #20,NEXT-CHAR	Vyzvedni další znak
		CP #3D	a není-li to rovnítko,
		JR NZ,#1FBD,DEF-FN-7	skoč na chybu.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		LD A,(#5C3B) (FLAGS)	Vyzvedni jeho typ (číselná/řetězcová proměnná)
		PUSH AF	a uschovej.

	CALL #24FB,SCANNING	Posuď definici jako výraz.
	POP AF	Vyzvedni typ proměnné
	XOR (IY+1) (FLAGS)	a testuj je-li stejného typu
	AND #40	jako proměnná nalezená pro definici.
1FBF DEF-FN-7	JP NZ,#1C8A,REPORT-C	Pokud je to třeba, skoč na chybu.
	CALL #1BEE,CHECK-END	Výstup přes CHECK-END a tím posun na hodnocení dalšího příkazu.

#### PODPROGRAM UNSTACK-Z

Tento podprogram je volán v mnoha případech k provedení "dřívějšího" návratu z podprogramu při kontrole syntaxe. Tím se zabrání nežádoucím tiskům a přesouvání hodnot na zásobníku kalkulátoru.

1FC3 UNSTACK-Z	CALL #2530,SYNTAX-Z	Jde o kontrolu syntaxe?
	POP HL	Vyzvedni návratovou adresu
	RET Z	a ignoruj ji při kontrole syntaxe.
	JP (HL)	Za běhu programu proveď obyčejný návrat.

#### PODPROGRAMY PŘÍKAZU LPRINT & PRINT

Otevře se příslušný kanál a potom se postupně zkoumají jednotlivé položky, které mají být vytisknuty.

1FC9 LPRINT	LD A, #03	Příprava na otevření kanálu P.
	JR #1FCF,PRINT-1	Skok dopředu.
1FCD PRINT	LD A, #02	Příprava na otevření kanálu S.
1FCF PRINT-1	CALL #2530,SYNTAX-Z	Pokud nekontroluješ syntaxi,
	CALL NZ,#1601,CHAN-OPEN	skoč na otevření kanálu.
	CALL #0D4D,TEMPS	Nastav přechodné barvy.
	CALL #1FDF,PRINT-2	Volej tiskový podprogram.
	CALL #1BEE,CHECK-END	Při kontrole syntaxe se posuň na další příkaz.
	RET	Vrať se.

Tiskový podprogram je volán z programu PRINT, LPRINT & INPUT.

1FDF PRINT-2	RST #18,GET-CHAR	Vyzvedni první znak
	CALL #2045,PR-END-Z	a jsi-li již na konci seznamu položek,
	JR Z,#1FF2,PRINT-4	skoč dopředu.

Nyní vstup do smyčky, která obsluhuje poziční znaky a tiskové položky.





1FE5 PRINT-3	CALL #204E,PR-POSN-1	Obsluž všechny následovně poziční znaky.
	JR Z,#1FE5,PRINT-3	Obsluž jednoduché tiskové položky.
	CALL #1FFC,PR-ITEM-1	Testuj, jestli jsou další položky
	CALL #204E,PR-POSN-1	a tiskni všechny až do konce
	JR Z,#1FE5,PRINT-3	
1FF2 PRINT-4	CP #29	Je-li aktuální znak závorka ")",
	RET Z	vrať se. Jinak postup do podprogramu CR.

#### PODPROGRAM "CR - NÁVRAT VOZÍKU"

1FF5 PRINT-CR	CALL #1FC3,UNSTACK-Z	Vrať se při kontrole syntaxe.
	LD A,#0D	Proveď CR
	RST #10,PRINT-A-1	
	RET	a vrať se.

## PODPROGRAM TISKU POLOŽEK

Podprogram je volán z programu PRINT, LPRINT & INPUT.

	1FFC	PR-ITEM-1	RST #18,GET-CHAR	Vyzvedni první znak.
			CP #AC	Pokud to není token AT,
			JR NZ,#200E,PR-ITEM-2	skoč dopředu.
			CALL #1C79,NEXT-2NUM	Předej dva parametry na zásobník kalkulátoru.
			CALL #1FC3,UNSTACK-Z	Návrat při kontrole syntaxe.
			CALL #2307,STK-T0-BC	Parametry do BC.
			LD A,#16	Řídící znak AT do registru A
			JR #201E,PR-AT-TAB	a skok na provedení.
	200E	PR-ITEM-2	CP #AD	Pokud to není token <b>TAB</b> ,
			JR NZ,#2024,PR-ITEM-3	skoč dopředu.
			RST #20,NEXT-CHAR	Vyzvedni další znak.
			CALL #1C82,EXPT-1NUM	Předej jeden parametr na zásobník kalkulátoru.
			CALL #1FC3,UNSTACK-Z	Návrat při kontrole syntaxe.
			CALL #1E99,FIND-INT2	Parametr do BC.
			LD A,#17	Řídící znak TAB do registru A.

AT a TAB se provedou třemi voláními PRINT-OUT.

201E	PR-AT-TAB	RST #10, <b>PRINT-A-1</b>	Tiskni řídící znak.
		LD A,C	Následuje hodnota
		RST #10, <b>PRINT-A-1</b>	prvního parametru.
		LD A,B	A hodnota
		RST #10, <b>PRINT-A-1</b>	druhého parametru.
		RET	Vrať se.

Dále posuzuj "implantované informace" o barvách.

2024	PR-ITEM-3	CALL #21F2,CO-TEMP-3	Pokud je taková informace obsažena,
		RET NC	vrať se CY=0.
		CALL #2070,STR-ALTER	Posuzuj, má-li být změněn proud.
		RET NC	Pokračuj, pokud ne.

Tisková informace musí být číselný nebo řetězcový výraz.

CALL #24FB,SCANNING	Proveď hodnocení výrazu,
CALL #1FC3,UNSTACK-Z	ale vrať se, pokud provádíš kontrolu syntaxe.
BIT 6,(IY+1) (FLAGS)	Test typu výrazu.
CALL Z,#2BF1,STK-FETCH	Je-li to řetězec, vyzvedni potřebné parametry.
JP NZ,#2DE3,PRINT-FP	Je-li to číslo, vystup přes PRINT-FP.

Následující smyčka zpracuje postupně všechny znaky v řetězci.

203C	PR-STRING	LD A,B	Pokud jsou všechny
		OR C	znaky
		DEC BC	vypsány,
		RET Z	vrať se.
		LD A,(DE)	Vyzvedni kód
		INC DE	a posuň ukazatel.
		RST #10, <b>PRINT-A-1</b>	Tiskni znak.
		JR #203C,PR-STRING	A skoč na další posouzení.

## PODPROGRAM "KONEC TISKU"

Z flag = 1, není-li co tisknout.

2045	PR-END-Z	CP #29	Je-li znak závorka ")",
		RET Z	vrať se.
2048	PR-ST-END	CP #0D	Je-li znak CR,
		RET Z	vrať se.
		CP #3A	Před návratem testuj znak dvojtečky.
		RET	

#### PODPROGRAM "TISK POZICE"

Tento podprogram zpracuje různé znaky řídící tiskovou pozici.

204E	PR-POSN-1	RST #18,GET-CHAR	Vyzvedni aktuální znak.
		CP #3B	Je-li to středník,
		JR Z,#2067,PR-POSN-3	skoč dopředu.
		CP #2C	Je-li to čárka,
		JR NZ,#2061,PR-POSN-2	skoč dopředu.
		CALL #2530,SYNTAX-Z	Ale při kontrole syntaxe
		JR Z,#2067,PR-POSN-3	znak netiskni.
		LD A,#06	Dej do A řídící kód čárky
		RST #10,PRINT-A-1	vytiskni ji
		JR #2067,PR-POSN-3	a skoč dopředu.
2061	PR-POSN-2	CP #27	Testuj apostrof " ' "
		RET NZ	a vrať se pokud to nebyl některý z pozičních znaků.
		CALL #1FF5,PRINT-CR	Pokud nekontroluješ syntaxi, tiskni CR.
2067	PR-POSN-3	RST #20,NEXT-CHAR	Vyzvedni další znak.
		CALL #2045,PR-END-Z	Nejsi-li na konci tiskového příkazu,
		JR NZ,#206E,PR-POSN-4	skoč dopředu.
		POP BC	Jinak se vrať do volajícího podprogramu.
206E	PR-POSN-4	CP A	Nebylo-li dosaženo konce příkazu,
		RET	nastaví se Z flag=0.

#### PODPROGRAM "ZMĚNA PROUDU"

Kdykoliv je třeba posoudit, jestli uživatel chce použít odlišný proud, volá se tento podprogram.

2070	STR-ALTER	CP #23	Testuj znak " # "
		SCF	a pokud není,
		RET NZ	vrať se s CY=1.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CALL #1C82,EXPT-INUM	Parametr na zásobník kalkulátoru.
		AND A	CY=0.
		CALL #1FC3,UNSTACK-Z	Vrať se při kontrole syntaxe.
		CALL #1E94,FIND-INT1	Vyzvedni hodnotu do A.
		CP #10	Pokud je hodnota větší než #0F,
		JP NC,#160E,REPORT-0	skoč na chybové hlášení.
		CALL #1601,CHAN-OPEN	Otevři kanál pro daný proud.
		AND A	CY=0
		RET	a vrať se.

#### PODPROGRAM PŘÍKAZU INPUT

Tento podprogram umožňuje přidělení hodnot vstupujících z klávesnice do dané proměnné. Současně zajišťuje aby tisková informace se zobrazovala do dolní části obrazovky.

2089	INPUT	CALL #2530,SYNTAX-Z	Při kontrole syntaxe
		JR Z,#2096,INPUT-1	skoč dopředu.
		LD A,#01	Kanál K
		CALL #1601,CHAN-OPEN	a jeho otevření.

2096	INPUT-1	CALL #0D6E,CLS-LOWER	Smazání dolní části obrazovky.
		LD (IY+2),#01 (TV-FLAG)	Nastav dolní část obrazovky na velikost jednoho řádku.
		CALL #20C1,IN-ITEM-1	Volej obsluhu vložených tisků.
		CALL #1BEE,CHECK-END	Při kontrole syntaxe postup na další příkaz.
		LD BC,(#5C88) (S-POSN)	Vyzvedni aktuální tiskovou pozici
		LD A,(#5C6B) (DF-SZ)	a pokud se tato
		CP B	nachází nad dolní částí obrazovky,
		JR C,#20AD,INPUT-2	skoč dopředu.
		LD C,#21	Jinak nastav tiskovou pozici
		LD B,A	na nejvyšší řádek
20AD	INPUT-2	LD (#5C88),BC (S-POSN)	dolní části obrazovky.
		LD A,#19	Nyní
		SUB B	nastav
		LD (#5C8C),A (SCR-CT)	čítač rolování.
		RES 0,(IY+2) (TV-FLAG)	Signál: hlavní obrazovka.
		CALL #0DD9,CL-SET	Nastav systémové proměnné
		JP #0D6E,CLS-LOWER	a proved výstup přes CLS-LOWER.

Tato smyčka se zabývá položkami INPUT a PRINT.

20C1	IN-ITEM-1	CALL #204E,PR-POSN-1	Nejprve posuzuj poziční znaky.
		JR Z,#20C1,IN-ITEM-1	
		CP #2B	Jestliže aktuální znak není závorka " ( ",
		JR NZ,#20DB,IN-ITEM-2	skoč dopředu.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CALL #1FDF,PRINT-2	Vytiskni položky uvnitř závorek.
		RST #1B,GET-CHAR	Vyzvedni znak
		CP #29	a pokud to není závorka " ) ",
		JP NZ,#1C8A,REPORT-C	skoč na chybové hlášení.
		RST #20,NEXT-CHAR	Vyzvedni další znak
		JP #21B2,IN-NEXT-2	a skoč na posouzení dalších případných položek.

Nyní posuzuj, je-li použito INPUT LINE.

20DB	IN-ITEM-2	CP #CA	Nejedná-li se o LINE,
		JR NZ,#20ED,IN-ITEM-3	skoč dopředu.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CALL #1C1F,CLASS-01	Určení cílové adresy pro proměnnou.
		SET 7,(IY+55) (FLAGX)	Signál: INPUT LINE.
		BIT 6,(IY+1) (FLAGX)	Nejedná-li se o řetězcovou proměnnou,
		JP NZ,#1C8A,REPORT-C	skoč dopředu.
		JR #20FA,IN-PROMPT	Skok na vypsání náznamu.

Zpracování proměnných INPUT.

20ED	IN-ITEM-3	CALL #2C8D,ALPHA	Nejí-li aktuální znak písmeno,
		JP NC,#21AF,IN-NEXT-1	skoč zpět do smyčky.
		CALL #1C1F,CLASS-01	Určení cílové adresy proměnné.
		RES 7,(IY+55) (FLAGX)	Signál: Nejedná se o INPUT LINE.
20FA	IN-PROMPT	CALL #2530,SYNTAX-Z	Jestliže kontroluješ syntaxi,
		JP Z,#21B2,IN-NEXT-2	skoč dopředu.
		CALL #16BF,SET-WORK	Pracovní prostor je nastaven na nulu.
		LD HL,#5C71	Toto je FLAGX.
		RES 6,(HL)	Signál: řetězcový výsledek.
		SET 5,(HL)	Signál: INPUT mode.
		LD BC,#0001	Náznak zabere jedno místo.
		BIT 7,(HL)	Při použití LINE
		JR NZ,#211C,IN-PR-2	skoč dopředu.

	LD A,(#5C3B) (FLAGS)	Očekáváš-li
	AND #40	číselný vstup,
	JR NZ,#211A,IN-PR-1	skoč dopředu.
211A IN-PR-1	LD C,#03	Řetězcová <b>náznak</b> zabere tři místa.
	OR (HL)	Pro číselnou položku
	LD (HL),A	bude nastaven bit 6 FLAGX.
211C IN-PR-2	RST #30,BC-SPACES	Je vytvořen požadovaný prostor.
	LD (HL),#0D	Znak CR na poslední místo.
	LD A,C	Testováním bitu 6
	RRCA	registru C se zjistí
	RRCA	bylo-li vytvořeno pouze jedno místo
	JR NC,#2129,IN-PR-3	a v tom případě se provede skok.
	LD A,#22	Znak uvozovek
	LD (DE),A	přijde na první
	DEC HL	a druhé
	LD (HL),A	místo.
2129 IN-PR-3	LD (#5C5B),HL (K-CUR)	Je uschována pozice kurzoru.

V případě INPUT LINE může být editor volán bez další přípravy, ale pro ostatní typy INPUTů musí být pozměněn chybový zásobník, aby zachytával případně chyby.

	BIT 7,(IY+55) (FLAGX)	Pokud jde o INPUT LINE,
	JR NZ,#215E,IN-VAR-3	skok dopředu.
	LD HL,(#5C5D) (CH-ADD)	Uchovej aktuální hodnotu CH-ADD
	PUSH HL	na zásobníku.
	LD HL,(#5C3D) (ERR-SP)	Uchovej aktuální hodnotu ERR-SP
	PUSH HL	na zásobníku.
213A IN-VAR-1	LD HL,#213A	Toto bude bod návratu
	PUSH HL	v případě výskytu chyby.
	BIT 4,(IY+4B) (FLAGS2)	Pokud používáš kanál K,
	JR Z,#214B,IN-VAR-2	přeskoč dopředu.
	LD (#5C3D),SP (ERR-SP)	Nastav ukazatel chyby.
214B IN-VAR-2	LD HL,(#5C61) (WORKSP)	Nastav HL na začátek řádku INPUT
	CALL #11A7,REMOVE-FP	a odstraň FP formy.
	LD (IY+0),#FF (ERR-NR)	Signál: zatím žádná chyba.
	CALL #0F2C,EDITOR	Nyní vezmi INPUT
	RES 7,(IY+1) (FLAGS)	a nastav signál : kontrola syntaxe.
	CALL #21B9,IN-ASSIGN	Testuj INPUT na chyby
	JR #2161,IN-VAR-4	a skoč, je-li vše v pořádku.
215E IN-VAR-3	CALL #0F2C,EDITOR	Vem LINE.

Všechny systémové proměnné musí být nastaveny, než se provede skutečné přidělení hodnoty.

2161 IN-VAR-4	LD (IY+34),#00 (K-CUR-hi)	Adresa kurzoru je resetována.
	CALL #21D6,IN-CHAN-K	Pokud se používá jiný kanál než K
	JR NZ,#2174,IN-VAR-5	provede se skok.
	CALL #111D,ED-COPY	Inputový řádek je vypsán na obrazovku
	LD BC,(#5C82) (ECHO-E)	a pozice z ECHO-E se stane aktuální tiskovou pozicí
	CALL #0DD9,CL-SET	v dolní části obrazovky.
2174 IN-VAR-5	LD HL,#5C71	Toto je FLAGX.
	RES 5,(HL)	Signál: <b>edit mode</b> .
	BIT 7,(HL)	Pokud se jedná
	RES 7,(HL)	o INPUT LINE,
	JR NZ,#219B,IN-VAR-6	skoč dopředu.
	POP HL	Zahod adresu IN-VAR-1.
	POP HL	Vyzvedni
	LD (#5C3D),HL (ERR-SP)	a obnov původní adresu pro ERR-SP.
	POP HL	Vyzvedni původní adresu CH-ADD

	LD (#5C5F),HL (X-PTR)	a uschovej ji v X-PTR.
	SET 7,(IY+1) (FLAGS)	Signál: běh programu.
	CALL #21B9,IN-ASSIGN	Proveď přidělení.
	LD HL,(#5C5F) (X-PTR)	Původní adresa CH-ADD do HL.
	LD (IY+3B),#00 (X-PTR-hi)	Vynulování X-PTR-hi
	LD (#5C5D),HL (CH-ADD)	a obnovení CH-ADD.
	JR #21B2,IN-NEXT-2	Skok na posouzení případných dalších položek.
219B IN-VAR-6	LD HL,(#5C63) (STKBOT)	Konec LINE do HL.
	LD DE,(#5C61) (WORKSP)	Začátek LINE do DE.
	SCF	
	SBC HL,DE	Výpočet délky
	LD B,H	a převod
	LD C,L	do BC.
	CALL #2AB2,STK-STOP	Tyto parametry jdou na zásobník
	CALL #2AFF,LET	a provede se přidělení.
	JR #21B2,IN-NEXT-2	Skok na posouzení případných dalších položek.

Jsou posuzovány další položky.

21AF IN-NEXT-1	CALL #1FFC,PR-ITEM-1	Ošetři jakoukoliv tiskovou položku.
21B2 IN-NEXT-2	CALL #204E,PR-POSN-1	Ošetři jakoukoliv poziční položku.
	JP Z,#20C1,IN-ITEM-1	Případný skok na posouzení dalších položek.
	RET	Návrat.

#### PODPROGRAM "IN ASIGN"

Tento podprogram je volán dvakrát pro každý INPUT. Jednou při "kontrolě syntaxe" a jednou za běhu programu.

21B9 IN-ASSIGN	LD HL,(#5C61) (WORKSP)	Nastav CH-ADD, aby ukazovala
	LD (#5C5D),HL (CH-ADD)	na první místo pracovního prostoru.
	RST #1B,GET-CHAR	Vyzvedni znak.
	CP #E2	Je to token STOP?
	JR Z,#21D0,IN-STOP	Skoč, jestliže ano.
	LD A,(#5C71) (FLAGX)	Jinak přiřaď
	CALL #1C59,VAL-FET-2	proměnné její hodnotu.
	RST #1B,GET-CHAR	Vyzvedni znak
	CP #0D	a pokud se jedná o CR,
	RET Z	vrať se.
21CE REPORT-C	RST #0B,ERROR-1	Ohlaš:
	DEFB #0B	C-Nonsense in BASIC.

Do tohoto bodu se skočí v případě, že INPUT začíná příkazem STOP.

21D0 IN-STOP	CALL #2530,SYNTAX-Z	Ale při kontrolě syntaxe
	RET Z	nebude ohlášena "chyba".
21D4 REPORT-H	RST #0B,ERROR-1	Ohlaš:
	DEFB #10	H-Stop in INPUT.

#### PODPROGRAM "IN-CHAN-K"

Podprogram vrací Z flag=0, jedině když se používá kanál K.

21D6 IN-CHAN-K	LD HL,(#5C51) (CURCHL)	Je vyzvednuta
	INC HL	bázová adresa
	INC HL	kanálových informací
	INC HL	a kód aktuálního
	INC HL	kanálu



LD	A,(HL)	je testován
CP	#4B	proti znaku K.
RET		Návrat.

#### PODPROGRAMY "BAREVNÝCH POLOŽEK"

Tento soubor podprogramů lze rozdělit na dvě části:

- Část obsluhující "implementované" barevné informace.
- Část obsluhující systémové proměnné ATTR-T, MASK-T a P-FLAG.

**Barevné informace budou zpracovány postupně. Vstupní bod je CO-TEMP-2.**

21E1	CO-TEMP-1	RST #20,NEXT-CHAR	Vyzvedni další znak.
21E2	CO-TEMP-2	CALL #21F2,CO-TEMP-3	Posuď, je-li tento znak vloženou barevnou informací.
		RET C	Vrať se s CY=1, pokud není.
		RST #18,GET-CHAR	Vyzvedni aktuální znak.
		CP #2C	Je-li to čárka,
		JR Z,#21E1,CO-TEMP-1	skoč zpět.
		CP #3B	Je-li to středník,
		JR Z,#21E1,CO-TEMP-1	skoč zpět.
		JP #1C8A,REPORT-C	Jinak ohlaš chybu.
21F2	CO-TEMP-3	CP #D9	Jestliže testovaný kód není
		RET C	v rozsahu
		CP #DF	#D9 až #DD (INK až OVER),
		CCF	vrať se
		RET C	s CY=1.
		PUSH AF	Ušchvej kód barevné informace
		RST #20,NEXT-CHAR	a posuň ukazatel na parametr.
		POP AF	Obnov kód.

Kód barevné informace a jeho parametry jsou nyní vytisknuty voláním podprogramu PRINT-OUT.

21FC	CO-TEMP-4	SUB #C9	Kód token je snížen na hodnotu řídicího znaku (#10 až #15).
		PUSH AF	Ušchvej kód.
		CALL #1C82,EXPT-1NUM	Předej parametr na zásobník kalkulátoru.
		POP AF	Obnov kód
		AND A	a pokud kontroluješ syntaxi,
		CALL #1FC3,UNSTACK-Z	vrať se pomocí UNSTACK-Z.
		PUSH AF	Ušchvej kód.
		CALL #1E94,FIND-INT1	Vyzvedni parametr
		LD D,A	a převed do D.
		POP AF	Obnov kód
		RST #10,PRINT-A-1	a vytiskni jej.
		LD A,D	Parametr do A
		RST #10,PRINT-A-1	a vytiskni jej.
		RET	Vrať se.

B) Barevné systémové proměnné (ATTR-T, MASK-T & P-FLAG) jsou upraveny na požadované hodnoty. Podprogram je volán z podprogramu PRINT-OUT. Na vstupu je kód řídicího znaku v registru A a jeho parametr v registru D. Všechny změny **barev** jsou pouze "přechodné".

2211	CO-TEMP-5	SUB #11	Snižuj rozsah
		ADC A,#00	a skoč dopředu
		JR Z,#2234,CO-TEMP-7	při INK & PAPER.
		SUB #02	Ještě sniž rozsah
		ADC A,#00	a skoč dopředu
		JR Z,#2273,CO-TEMP-C	při FLASH & BRIGHT.

Kód řídicího znaku je nyní #01 pro INVERSE a #02 pro OVER. Podle toho bude nastavena systémová proměnné P-FLAG.

	CP #01	Příprava na skok pro OVER.	
	LD A,D	Vyzvedni parametr.	
	LD B,#01	Příprav masku pro OVER.	
	JR NZ,#222B,CO-TEMP-6	Skoč, jde-li o OVER.	
	RLCA	Pro INVERSE 0 musí být bit 2 registru A nastaven na nulu.	
	RLCA	Pro INVERSE 1 musí být bit 2 reg. A nastaven na jedničku.	
	LD B,#04	Maska má bit 2 nastaven na jedničku.	
2228	CO-TEMP-6	LD C,A	Ušchovej obsah A.
	LD A,D	Povolený rozsah parametru	
	CP #02	je pouze #00 až #01.	
	JR NC,#2244,REPORT-K	Skok na chybu, jeli mimo rozsah.	
	LD A,C	Vyzvedni zpět A.	
	LD HL,#5C91	Toto je P-FLAG, který se bude měnit.	
	JR #226C,CO-CHANGE	Výstup přes CO-CHANGE se změnou P-FLAG maskou v registru B.	

V této části se zpracovuje PAPER a INK. Na vstupu je CY=1 pro INK.

2234	CO-TEMP-7	LD A,D	Vyzvedni parametr.
		LD B,#07	Příprav masku pro INK.
		JR C,#223E,CO-TEMP-8	Skoč dopředu s INK.
		RLCA	Vynásob parametr osmi.
		RLCA	
		LD B,#38	Příprav masku pro PAPER.
223E	CO-TEMP-8	LD C,A	Ušchovej parametr v C.
		LD A,D	Povolený rozsah parametru.
		CP #0A	je pouze #00 až #09.
		JR C,#2246,CO-TEMP-9	Skoč, je-li rozsah dodržen.
2244	REPORT-K	RST #0B,ERROR-1	Ohlaš:
		DEFB #13	K-Invalid colour

Pokračuj zpracováním řádek PAPER & INK.

2246	CO-TEMP-9	LD HL,#5CBF	Příprava na změnu ATTR-T, MASK-T & P-FLAG.
		CP #08	Skoč dopředu
		JR C,#2258,CO-TEMP-B	s PAPER/INK 0 až 7.
		LD A,(HL)	Vyzvedni hodnotu ATTR-T a pokud je PAPER/INK 8,
		JR Z,#2257,CO-TEMP-A	skoč dopředu (barva se nemění).
		OR B	Ale pro PAPER/INK 9
		CPL	budou barvy černá a bílá.
		AND #24	
		JR Z,#2257,CO-TEMP-A	Skoč s černými INK/PAPER.
		LD A,B	Pokračuj s bílými INK/PAPER.
2257	CO-TEMP-A	LD C,A	Převed hodnotu do C.

Nyní jsou použity maska B a hodnota C ke změně ATTR-T.

2258	CO-TEMP-B	LD A,C	Hodnota zpět do A.
		CALL #226C,CO-CHANGE	Změna ATTR-T.
		LD A,#07	Bity MASK-T
		CP D	se budou měnit
		SBC A,A	pouze pro
		CALL #226C,CO-CHANGE	PAPER/INK 8 nebo 9.





RLCA	Pro P-FLAG
RLCA	je příslušná maska
AND #50	vytvořena
LD B,A	v registru B.
LD A,#08	Bit 4 a 6 bude změněn
CP D	jedině v případě že
SBC A,A	PAPER/INK je 9. <b>Pokračuj přes CO-CHANGE na zpracování P-FLAG.</b>

#### PODPROGRAM "CO-CHANGE"

Tento podprogram předá systémové proměnné bity vstupující v registru A. Maska v B ukazuje, které bity z A mají být kopírovány **na (HL)**.

226C CO-CHANGE	XOR (HL)	Bity specifikované
	AND B	registrem B
	XOR (HL)	jsou změněny
	LD (HL),A	a výsledek jde do systémové proměnné.
	INC HL	Přejdi na adresu další systémové proměnné.
	LD A,B	Vrať se s maskou
	RET	v registru A.

Nyní se zpracuje FLASH & BRIGHT.

2273 CO-TEMP-C	SBC A,A	Pro BRIGHT bude Z - flag = 0.
	LD A,D	Parametr je vyzvednut
	RRCA	a rotován.
	LD B,#80	Příprav masku pro FLASH.
	JR NZ,#227D,CO-TEMP-D	Skoč dopředu jde-li o FLASH.
	RRCA	Další rotace
	LD B,#40	a příprava masky pro BRIGHT.
227D CO-TEMP-D	LD C,A	Uschovej hodnotu v C.
	LD A,D	Vyzvedni parametr a testuj jeho rozsah.
	CP #08	Povoleno je pouze #08,
	JR Z,#2287,CO-TEMP-E	#01
	CP #02	a #02.
	JR NC,#2244,REPORT-K	Skoč na chybové hlášení <b>pokud tomu tak není.</b>
2287 CO-TEMP-E	LD A,C	Obnov uschovanou hodnotu do A.
	LD HL,#5C8F (ATTR-T)	Toto je ATTR-T.
	CALL #226C,CO-CHANGE	Proveď změnu této systémové proměnné.
	LD A,C	Obnov uschovanou hodnotu do A.
	RRCA	Rotuj, aby bit nastavený
	RRCA	pro FLASH/BRIGHT 8 přešel na místo
	RRCA	bitu 7 pro FLASH nebo bitu 6 pro BRIGHT.
	JR #226C,CO-CHANGE	Vystup přes CO-CHANGE.



#### PODPROGRAM PŘÍKAZU BORDER

Parametr příkazu BORDER je proveden instrukcí OUT a uschován v systémové proměnné BORDCR.

2294 BORDER	CALL #1E94,FIND-INT1	Vyzvedni parametr
	CP #08	a testuj jeho rozsah.
	JR NC,#2244,REPORT-K	Skok na chybu, je-li parametr větší než 7.
	OUT (#FE),A	Nastav barvu BORDER
	RLCA	násob
	RLCA	parametr
	RLCA	osmi.
	BIT 5,A	Je-li barva BORDER světlá,
	JR NZ,#22A6,BORDER-1	skoč dopředu (INK bude černý).

	XOR	#07	Změna barvy INK.
22A6 BORDER-1	LD	(#5C48),A (BORDCR)	Nastav systémovou proměnnou.
	RET		Vrať se.

#### PODPROGRAM "ADRESA OBRAZOVÉHO BODU"

Je používán podprogramy POINT a PLOT. Na vstupu obsahuje BC souřadnice x,y. Na výstupu je v HL adresa bajtu obrazovky, ve kterém je daný bod obsažen a registr A ukazuje na jeho pozici uvnitř tohoto bajtu.

22AA PIXEL-ADD	LD	A,#AF	Testuj, zdali souřadnice y
	SUB	B	není větší než 175
	JP	C,#24F9,REPORT-B	a jestliže ano, skoč na ohlášený chyby.
	LD	B,A	B nyní obsahuje 175-y.
	AND	A	A obsahuje bity: 7 6 5 4 3 2 1 0 bajtu z <b>reg. B + CY=0.</b>
	RRA		A obsahuje: 00 a bity: 7 6 5 4 3 2 1.
	SCF		<b>CY=1</b>
	RRA		A obsahuje: 01 00 a bity 7 6 5 4 3 2.
	AND	A	<b>CY=0</b>
	RRA		A obsahuje: 00 01 00 a bity: 7 6 5 4 3.
	XOR	B	
	AND	#FB	A obsahuje: 00 01 00 a bity: 7 6 2 1 0.
	XOR	B	Takže H bude obsahovat $64+8*INT(b/64)+(b \text{ mod } 8)$
	LD	H,A	což je vyšší bajt adresy bodu.
	LD	A,C	A obsahuje bity: 7 6 5 4 3 2 1 0 bajtu z <b>reg. C.</b>
	RLCA		
	RLCA		
	RLCA		A obsahuje bity: 2 1 0 7 6 5 4 3.
	XOR	B	
	AND	#C7	
	XOR	B	A obsahuje bity: <b>2 1 5 4 3 5 4 3.</b>
	RLCA		
	RLCA		A obsahuje bity: <b>5 4 3 7 6 5 4 3.</b>
	LD	L,A	Takže L bude $32*INT(b/(b \text{ mod } 64)/8)+INT(x/8)$ .
	LD	A,C	A obsahuje x ( <b>mod 8</b> ).
	AND	#07	Takže bod je bit ( <b>0 až 7</b> ) uvnitř daného bajtu.
	RET		

#### PODPROGRAM POINT

Je používán podprogramem příkazu POINT při prohledávání. Na vstupu jsou souřadnice x, y uloženy na zásobníku kalkulátoru a na výstupu je na zásobníku 1, pokud má bod barvu INK a 0, je-li to barva PAPER.

22CB POINT-SUB	CALL	#2307,STK-T0-BC	Y do reg.B a x do reg. C.
	CALL	#22AA,PIXEL-ADD	Adresa bodu do HL.
	LD	B,A	B bude počítat A+1 krát
	INC	B	aby se požadovaný bit převedl do bitu 0.
	LD	A,(HL)	<b>Vyzvední bajt z videoram.</b>
22D4 POINT-LP	RLCA		Rotuj doleva a zároveň do CY.
	DJNZ	#22D4,POINT-LP	
	AND	#01	Bit=1 pro inkoust, 0 pro papír.
	JP	#2D28,STACK-A	Výsledek je uložen na zásobník kalkulátoru.

## PODPROGRAM PŘÍKAZU PLOT

Je využíván také programy CIRCLE a DRAW. Na vstupu jsou souřadnice x, y uloženy na zásobníku kalkulátoru. Po nalezení adresy se bere v úvahu stav INVERSE a OVER a nakreslí příslušný bod.

22DC	PLOT	CALL #2307,STK-T0-BC	Y do reg. B a x do reg. C.
		CALL #22E5,PLOT-SUB	Volání podprogramu.
		JP #0D4D,TEMPS	Výstup s nastavením přechodných barev.
22E5	PLOT-SUB	LD (#5C7D),BC (COORDS)	Nastav systémovou proměnnou.
		CALL #22AA,PIXEL-ADD	Adresa bodu do HL.
		LD B,A	B bude počítat A+1 krát,
		INC B	aby se log. 0 dostala na správné místo v A.
		LD A,#FE	Vložení nuly
22F0	PLOT-LOOP	RRCA	a její nastavení
		DJNZ #22F0,PLOT-LOOP	do bitu, který odpovídá pozici bodu
		LD B,A	kopie do B.
		LD A,(HL)	Bajt obrazovky do <b>A</b> .
		LD C,(IY+87) (P-FLAG)	
		BIT 0,C	Test na OVER 1
		JR NZ,#22FD,PL-TST-IN	a přeskok, když ano
		AND B	OVER 0 způsobí, že bod bude mít hodnotu 0.
22FD	PL-TST-IN	BIT 2,C	Test na INVERSE 1
		JR NZ,#2303,PLOT-END	a přeskok, když ano
		XOR B	INVERSE 0 překlopí
		CPL	logickou hodnotu bodu.
2303	PLOT-END	LD (HL),A	Bajt je vložen na obrazovku.
		JP #0BDB,PO-ATTR	Výstup s nastavením atributu.

## PODPROGRAM STK-T0-BC

Tento podprogram uloží dvě FP čísla do registrového páru BC a je používán k vyzvednutí parametrů v rozsahu #00 až #FF. Zároveň však ukládá do registrového páru DE tzv. diagonální hodnoty (+-1,+-1), které jsou používány v podprogramu DRAW.

2307	STK-T0-BC	CALL #2314,STK-T0-A	První číslo do A.
		LD B,A	Potom do B.
		PUSH BC	Chvilková úschova.
		CALL #2314,STK-T0-A	Druhé číslo do A a
		LD E,C	jeho znaménko do E.
		POP BC	Obnov první číslo.
		LD D,C	Jeho znaménko do D.
		LD C,A	Druhé číslo do C.
		RET	BC a <b>DE</b> jsou nastaveny tak, jak bylo požadováno.

## PODPROGRAM STK-T0-A

Tento podprogram uloží do registru A FP číslo z vrcholu kalkulátorového zásobníku. Číslo musí být v rozsahu #00 až #FF.

2314	STK-T0-A	CALL #2DD5,FP-T0-A	Modul zaokrouhlené poslední hodnoty do registru A,
		JP C,#24F9,REPORT-B	není-li to možné, ohlaš chybu.
		LD C,#01	#01 do C jako signál: pozitivní hodnota.
		RET Z	Vrať se, byla-li hodnota pozitivní.
		LD C,#FF	V ostatních případech změň hodnotu v C na #FF (= -1).
		RET	Hotovo.

## PŘÍKAZ CIRCLE

Tento podprogram narýsuje přibližnou kružnici se středem v bodě X,Y o poloměru Z. Parametry jsou před použitím převedeny na tvar INTEGER. Proto Z musí být menší než 87.5, i když střed kružnice leží ve středu obrazovky. Použitá metoda nakreslí sérii přímků aproximujících oblouk. Program CIRCLE má čtyři části:

- Test poloměru. Je-li poloměr menší než jedna, je narýsován pouze bod o souřadnicích X,Y.
- Je zavolána CD-PRMS1 na adrese #247D, která se používá k nastavení počátečních parametrů pro CIRCLE a DRAW.
- Jsou nastaveny zbývající parametry pro CIRCLE včetně počátečního umístění prvního "oblouku".
- Skok do DRAW, kde je využito programu pro narýsování oblouku (na adrese #2420).

Nyní budou popsány postupně části a) až c).

ad a) #2320 až #233A. Poloměr Z je vyzvednut z kalkulátorového zásobníku a jeho hodnota je převedena na vhodný tvar. Je-li Z menší než jedna, je vymazáno ze zásobníku a je proveden odskok pro narýsování bodu o souřadnicích X,Y.

```
2320 CIRCLE      RST #18,GET-CHAR      Vyzvedni aktuální znak a
                  CP #2C              otestuj ho.
                  JP  NZ,#1C8A,REPORT-C  Ohlaš chybu, není-li to čárka.
                  RST #20,NEXT-CHAR      Vyzvedni další znak (poloměr) a
                  CALL #1C82,EXPT-1NUM    ulož ho na zásobník kalkulátoru.
                  CALL #1BEE,CHECK-END    Přejdi na další příkaz při kontrole syntaxe.
                  RST #28,FP-CALC        Kalkulátor.
                  DEFB #2A,abs           X,Y,Z
                  DEFB #3D,re-stack      Tvar Z je převeden z INTEGERu na FP a ponechán na
                                          zásobníku;
                  DEFB #38,konec výpočtu  proto je jeho exponent k dispozici.
                  LD  A,(HL)             Vezmi exponent poloměru.
                  CP  #81                Testuj, jestli je menší než jedna.
                  JR  NC,#233B,C-R-GRE-1  Jestliže ne, skoč.
                  RST #28,FP-CALC        Je-li menší,
                  DEFB #02,vymaz         vymaž ho ze zásobníku.
                  DEFB #38,konec výpočtu  Zásobník nyní obsahuje X,Y.
                  JR  #22DC,PLOT         Narýsuj bod na souřadnicích X,Y.
```

ad b) #233B až #2346 a volání CD-PRMS1.  $2*PI$  je uloženo do paměti č.5 a je zavoláno CD-PRMS1. Tento program uloží do registru B počet křivek, které jsou potřebné pro kruh viz.  $A=4*INT(PI*SQR(Z/4))+4$ , což je 4,8,12 ... až do maxima 32. Také ukládá do paměti č.0 až č.4 hodnoty  $2*PI/A$ ,  $SIN(PI/A)$ , 0,  $COS(2*PI/A)$  a  $SIN(2*PI/A)$ .

```
233B C-R-GRE-1  RST #28,FP-CALC      X,Y,Z,PI/2
                  DEFB #A3,stk-pi/2
                  DEFB #38,konec výpočtu  Nyní dochází ke změně exponentu
                  LD  (HL),#83           na #83, čímž dochází ke změně PI/2 na 2*PI.
                  RST #28,FP-CALC      X,Y,Z,2*PI.
                  DEFB #C5,st-mem-5    2*PI do mem-5.
                  DEFB #02,vymaz       X,Y,Z
                  DEFB #38,konec výpočtu
                  CALL #247D,CD-PRMS1  Nastav počáteční parametry.
```

ad c) #2347 až #2381: zbývající parametry a skok na DRAW. Testuje se, jestli je délka počátečního oblouku menší než jedna. Je-li tomu tak, provede se skok na prosté narýsování bodu o souřadnicích X,Y. Jinak jsou nastaveny tyto parametry:  $X=Z$  a  $Y=Z*SIN(PI/A)$  jako počáteční a koncový bod, a okopírovány do COORDS. Nula a hodnota  $2*Z*SIN(PI/A)$  jsou uschovány v mem-1 a mem-2 jako počáteční přírůstky. Smyčka příkazu DRAW, která zajišťuje rýsování oblouku, také zajistí, že všechny narýsované body zůstanou na stejné kružnici, a přírůstkový úhel bude  $2*PI/A$ .

2347	PUSH BC	Úschova počítadla oblouků do B.
	RST #28,FP-CALC	X,Y,Z
	DEFB #31,zdvojení	X,Y,Z,Z
	DEFB #E1,get-mem-1	X,Y,Z,Z,SIN (PI/A)
	DEFB #04,násobení	X,Y,Z,Z*SIN (PI/A)
	DEFB #38,konec výpočtu	Z*SIN (PI/A) je polovinou délky počátečního oblouku,
	LD A,(HL)	a tato hodnota je testována, aby se zjistilo,
	CP #80	jestli je menší než 0.5.
	JR NC,#235A,C-ARC-GE1	Když ne, provede se skok.
	RST #28,FP-CALC	Jinak jsou Z a "polooblouk" vymazány
	DEFB #02,vymaz	ze zásobníku kalkulátoru.
	DEFB #02,vymaz	
	DEFB #38,konec výpočtu	
	POP BC	Strojový zásobník je také vyčištěn a
	JP #22DC,PLOT	provede se skok na vytištěný bodu o souřadnicích X,Y.
235A C-ARC-GE1	RST #28,FP-CALC	X,Y,Z,Z*SIN (PI/A)
	DEFB #C2,st-mem-2	Z * SIN (PI/A) do mem-2.
	DEFB #01,záměna	X,Y,Z*SIN (PI/A),Z
	DEFB #C0,st-mem-0	X,Y,Z*SIN (PI/A),Z
	DEFB #02,vymaz	X,Y,Z*SIN (PI/A)
	DEFB #03,odečítání	X,Y-Z*SIN (PI/A)
	DEFB #01,záměna	Y-Z*SIN (PI/A),X
	DEFB #E0,get-mem-0	Y-Z*SIN (PI/A),X,Z
	DEFB #0F,sčítání	Y-Z*SIN (PI/A),X+Z
	DEFB #C0,st-mem-0	X+Z do mem-0.
	DEFB #01,záměna	X+Z,Y-Z*SIN (PI/A)
	DEFB #31,zdvojení	X+Z,Y-Z*SIN (PI/A), Y-Z*SIN (PI/A)
	DEFB #E0,get-mem-0	sa, sb, sb, sa
	DEFB #01,záměna	sa, sb, sa, sb
	DEFB #31,zdvojení	sa, sb, sa, sb, sb
	DEFB #E0,get-mem-0	sa, sb, sa, sb, sb, sa
	DEFB #A0,stk-nula	sa, sb, sa, sb, sb, sa, 0
	DEFB #C1,st-mem-1	mem-1 nastavena na nulu.
	DEFB #02,vymaz	sa, sb, sa, sb, sb, sa
	DEFB #38,konec výpočtu	

Zde "sa" znamená X+Z a "sb" znamená Y-Z\*SIN (PI/A).

INC (IY+28) (mem-2-1st)	Exponent v mem-2 je změněn-vzniká tvar 2*Z*SIN (PI/A).
CALL #1E94,FIND-INT1	Poslední hodnota X + Z je
LD L,A	předána ze zásobníku do registru A, a
PUSH HL	je uschována v HL.
CALL #1E94,FIND-INT1	Y-Z*SIN (PI/A) jde ze zásobníku do A
POP HL	
LD H,A	a je kopírováno do H.
LD (#5C7D),HL (COORDS)	HL ukazuje na počáteční bod a ten jde do COORDS.
POP BC	Je obnoveno počítadlo oblouků a
JP #2420,DRW-STEPS	provede se skok na DRAW.

Nyní jsou na zásobníku hodnoty X+Z, Y-Z\*SIN (PI/A), Y-Z\*SIN (PI/A), X+Z.

## PODPROGRAM PŘÍKAZU DRAW

Do tohoto podprogramu se vstupuje se souřadnicemi bodů, řekněme  $X_0, Y_0$  v systémové proměnné COORDS. Jsou-li pro příkaz DRAW udány pouze dva parametry  $X, Y$ , potom podprogram narýsuje přímkou z bodu  $X_0, Y_0$  do bodu  $X_0+X, Y_0+Y$ . Je-li udán třetí parametr  $G$ , potom podprogram rýsuje aproximaci kruhového oblouku z bodu o souřadnicích  $X_0, Y_0$  do bodu  $X_0+X, Y_0+Y$ , jehož středový úhel je  $G$  rad.

Podprogram má čtyři části:

- jsou-li udány pouze dva parametry nebo parametr poloměru je menší než jedna, je narýsována prostá přímka.
- volá CD-PRMS1 na #247D k nastavení počátečních parametrů.
- nastavuje zbývající parametry a to včetně počátečních rozmístění pro první oblouk.
- vstupuje do smyčky pro rýsování oblouků na #24B7.

Dva podprogramy CD-PRMS1 a DRAW-LINE následují za tímto hlavním podprogramem. Nyní budou postupně popsány všechny čtyři části hlavního podprogramu.

ad a) Jsou-li udány pouze dva parametry provede se skok na LINE-DRAW na adrese #2477. Přímka se rýsuje také v tom případě, je-li hodnota  $Z$  menší než jedna.  $Z$  leží jedním až jedenapůltým násobkem průměru zamýšlené kružnice. V této části je nastavena mem-0 na  $\text{SIN}(G/2)$ , mem-1 na  $Y$  a mem-5 na  $G$ .

2382	DRAW	RST #18,GET-CHAR	Vyzvedni aktuální znak.
		CP #2C	Je-li to čárka,
		JR Z,#238D,DR-3-PRMS	skoč.
		CALL #1BEE,CHECK-END	Při kontrole syntaxe se posuň na další příkaz.
		JP #2477,LINE-DRAW	Skoč na pouhé rýsování přímky.
238D	DR-3-PRMS	RST #20,NEXT-CHAR	Vyzvedni další znak (úhel),
		CALL #1C82,EXPT-1NUM	který je uložen na zásobníku kalkulátoru.
		CALL #1BEE,CHECK-END	Při kontrole syntaxe se posuň na další příkaz.
		RST #28,FP-CALC	$X, Y, G$ jsou na zásobníku.
		DEFB #C5,st-mem-5	$G$ jde do mem-5
		DEFB #A2,stk-polovinu	$X, Y, G, 0.5$
		DEFB #04,násobení	$X, Y, G/2$
		DEFB #1F,sin	$X, Y, \text{SIN}(G/2)$
		DEFB #31,zdvojení	$X, Y, \text{SIN}(G/2), \text{SIN}(G/2)$
		DEFB #30,not	$X, Y, \text{SIN}(G/2), (0 \text{ nebo } 1)$
		DEFB #30,not	$X, Y, \text{SIN}(G/2), (1 \text{ nebo } 0)$
		DEFB #00,skok-pravda	$X, Y, \text{SIN}(G/2)$
		DEFB #06,na DR-SIN-NZ	Jestli $\text{SIN}(G/2) \neq 0$ tj. $G \neq 2 * \pi$
		DEFB #02,výmaz	narýsuj přímkou.
		DEFB #38,konec výpočtu	$X, Y$
		JP #2477,LINE-DRAW	Přímka z bodu $X_0, Y_0$ do $X_0+X, Y_0+Y$ .
23A3	DR-SIN-NZ	DEFB #C0,st-mem-0	$\text{SIN}(G/2)$ jde do mem-0.
		DEFB #02,výmaz	$X, Y$ jsou nyní na zásobníku
		DEFB #C1,st-mem-1	$Y$ jde do mem-1
		DEFB #02,výmaz	$X$
		DEFB #31,zdvojení	$X, X$
		DEFB #2A,abs	$X, X' (X'+\text{ABS } X)$
		DEFB #E1,get-mem-1	$X, X', Y$
		DEFB #01,záměna	$X, Y, X'$
		DEFB #E1,get-mem-1	$X, Y, X', Y$
		DEFB #2A,abs	$X, Y, X', Y' (Y'+\text{ABS } Y)$
		DEFB #0F,sčítání	$X, Y, X'+Y'$
		DEFB #E0,get-mem-0	$X, Y, X'+Y', \text{SIN}(G/2)$
		DEFB #05,dělení	$X, Y, (X'+Y') * \text{SIN}(G/2)$ (dále pouze $Z'$ )
		DEFB #2A,abs	$X, Y, Z (Z=\text{ABS } Z')$
		DEFB #E0,get-mem-0	$X, Y, Z, \text{SIN}(G/2)$
		DEFB #01,záměna	$X, Y, \text{SIN}(G/2), Z$



DEFB #3D,re-stack	Z je v zásobníku převedeno tak,
DEFB #38,konec výpočtu	aby byl k dispozici jeho exponent.
LD A,(HL)	Vyzvedni exponent Z.
CP #81	Je-li Z větší nebo rovno jedné,
JR NC,#23C1,DR-PRMS	skoč.
RST #28,FP-CALC	X,Y,SIN (G/2),Z
DEFB #02,delete	X,Y,SIN (G/2)
DEFB #02,delete	X,Y
DEFB #38,konec výpočtu	
JP #2477,LINE-DRAW	Narýsuj přímku z bodu X0,Y0 do X0+X,Y0+Y.

ad b) Pouhé volání CD-PRMS1. Tento podprogram uchová v registru B počet kratších oblouků, které jsou potřeba pro kompletní oblouk. Podprogram také ukládá v mem-0 až v mem-4 hodnoty G/A, SIN (G/2\*A), 0, COS (G/A), SIN (G/A).

23C1 DR-PRMS CALL #247D,CD-PRMS1 Podprogram je volán.

ad c) Nastaví se zbytek parametrů jak je třeba. Zásobník bude obsahovat tyto čtyři položky čteno zdola: X0+X a Y0+Y jako koncový bod posledního oblouku, pak X0 a Y0 jako počáteční bod prvního oblouku. Mem-0 bude obsahovat X0 a mem-5 Y0. Mem-1 a mem-2 budou obsahovat počáteční umístění pro první oblouk U a V a mem-3 a mem-4 budou obsahovat COS (G/A) a SIN (G/A) pro potřeby smyčky rýsujeící oblouky. Formule pro výpočet U a V se dá vysvětlit následovně: Místo aby se postupovalo po závěrečné tětivě řekněme délky L, s přírůstky X a Y, chceme postupovat po počáteční tětivě (která může být delší) délky L\*W, kde W=SIN (G/2\*A)/SIN (G/2), s přírůstky X\*W a Y\*W, ale pootočené o úhel (G/2-G/2\*A), tedy s pravdivými přírůstky:

$$U=Y*W*SIN (G/2-G/2*A)+X*W*COS (G/2-G/2*A)$$

$$V=Y*W*COS (G/2-G/2*A)-X*W*SIN (G/2-G/2*A)$$

Tyto formule mohou být kontrolovány z diagramu, při použití normálních rozkladů pro COS (P-Q) a SIN (P-Q), kde Q=G/2-G/2\*A.

23C4	PUSH BC	Uschovej počítadlo oblouků v regist. B.
	RST #28,FP-CALC	X,Y,SIN (G/2),Z
	DEFB #02,výmaz	X,Y,SIN (G/2)
	DEFB #E1,get-mem-1	X,Y,SIN (G/2),SIN (G/2*A)
	DEFB #01,záměna	X,Y,SIN (G/2*A),SIN (G/2)
	DEFB #05,dělení	X,Y,SIN (G/2*A)/SIN (G/2)+W
	DEFB #C1,st-mem-1	W do mem-1.
	DEFB #02,výmaz	X,Y
	DEFB #01,záměna	Y,X
	DEFB #31,zdvojení	Y,X,X
	DEFB #E1,get-mem-1	Y,X,X,W
	DEFB #04,násobení	Y,X,X*W
	DEFB #C2,st-mem-2	X*W do mem-2.
	DEFB #02,výmaz	Y,X
	DEFB #01,záměna	X,Y
	DEFB #31,zdvojení	X,Y,Y
	DEFB #E1,get-mem-1	X,Y,Y,W
	DEFB #04,násobení	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
	DEFB #E0,get-mem-0	X,Y,Y*W,X*W,G/G/A
	DEFB #03,odčítání	X,Y,Y*W,X*W,G-G/A
	DEFB #A2,stk-polovina	X,Y,Y*W,X*W,G-G/A,1/2
	DEFB #04,násobení	X,Y,Y*W,X*W,G/2-G/2*A (dále pouze F)
	DEFB #31,zdvojení	X,Y,Y*W,X*W,F,F
	DEFB #1F,sin	X,Y,Y*W,X*W,F,SIN F

DEFB #C5, st-mem-5	SIN F do mem-5.
DEFB #02, výmaz	X, Y, Y*W, X*W, F
DEFB #20, cos	X, Y, Y*W, X*W, COS F
DEFB #C0, st-mem-0	COS F do mem-0.
DEFB #02, výmaz	X, Y, Y*W, X*W
DEFB #C2, st-mem-2	X*W do mem-2.
DEFB #02, výmaz	X, Y, Y*W
DEFB #C1, st-mem-1	Y*W do mem-1.
DEFB #E5, get-mem-5	X, Y, Y*W, SIN F
DEFB #04, násobení	X, Y, Y*W*SIN F
DEFB #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, násobení	X, Y, Y*W*SIN F, X*W*COS F
DEFB #0F, sčítání	X, Y, Y*W*SIN F + X*W*COS F (dále U)
DEFB #E1, get-mem-1	X, Y, U, Y*W
DEFB #01, záměna	X, Y, Y*W, U
DEFB #C1, st-mem-1	U do mem-1.
DEFB #02, výmaz	X, Y, Y*W
DEFB #E0, get-mem-0	X, Y, Y*W, COS F
DEFB #04, násobení	X, Y, Y*W*COS F
DEFB #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, násobení	X, Y, Y*W*COS F, X*W*SIN F
DEFB #03, odečítání	X, Y, Y*W*COS F - X*W*SIN F (dále V)
DEFB #C2, st-mem-2	V do 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)
DEFB #0F, sčítání	X, Y, V' + U'
DEFB #02, výmaz	X, Y
DEFB #38, konec výpočtu	DE nyní ukazuje na U' + V'.
LD A, (DE)	Vem exponent U' + V'.
CP #81	Jestliže platí U' + V' < 1, pak
POP BC	pouze uklidí zásobník.
JP C, #2477, LINE-DRAW	Přímka z bodu X0, Y0 do X0 + X, Y0 + Y.
PUSH BC	Jinak pokračuj s parametry X, Y
RST #28, FP-CALC	na zásobníku.
DEFB #01, záměna	Y, X
DEFB #38, konec výpočtu	
LD A, (#5C7D) (COORDS-lo)	Dej X0 do A a
CALL #2D28, STACK-A	tím dále na zásobník.
RST #28, FP-CALC	Y, X, X0
DEFB #C0, st-mem-0	X0 do mem-0.
DEFB #0F, sčítání	Y, X0 + X
DEFB #01, záměna	X0 + X, Y
DEFB #38, konec výpočtu	
LD A, (#5C7E) (COORDS-hi)	Dej Y0 do A a
CALL #2D28, STACK-A	tím i na zásobník.
RST #28, FP-CALC	X0 + X, Y, Y0
DEFB #C5, st-mem-5	Y0 do mem-5.
DEFB #0F, sčítání	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, konec výpočtu	
POP BC	Obnov počítadlo oblouků v registru B.

ad d) Smyčka pro rýsování oblouků. Do této smyčky se vstupuje na adrese #2439 s souřadnicemi počátečního bodu na vrcholu zásobníku a s umístěním prvního oblouku v mem-1 a mem-2. Je zajištěno pomocí jednoduché trigonometrie, že všechny následující body X1, Y1 a X2, Y2 leží na kružnici a jsou těmito středového úhlu N, který je též původem těchto souřadnic, potom  $X2=X1*\cos(N)-Y1*\sin(N)$ ,  $Y2=X1*\sin(N)+Y1*\cos(N)$ . Ale protože původ je zde ve spodním levém rohu obrazovky, musí se pro dosažení požadovaného efektu zahrnout tyto vztahy do přírůstků, řekněme  $Un=Xn+1-Xn$  a  $Vn=Yn+1-Yn$ , ve smyčce pro rýsování oblouků. V následujícím komentáři jsou ukázány hodnoty zásobníku pro n+1 průběh smyčky, kdy  $Xn$  a  $Yn$  jsou už zvětšeny o hodnoty  $Un$  a  $Vn$  poté, co byly obdrženy z  $Un-1$  a  $Vn-1$ . Čtyři hodnoty na vrcholu zásobníku na #2425 jsou u DRAW, čteno **zdola**,  $X0+X$ ,  $Y0+Y$ ,  $Xn$ ,  $Yn$  ale pro ušetření místa nejsou tyto hodnoty ukázány až do bodu #2439. Zajímají-li vás hodnoty pro příkaz CIRCLE, jsou vidět na konci podprogramu CIRCLE. Pro CIRCLE platí, že úhel G musí mít hodnotu  $2*PI$ .

2420	DRW-STEPS	DEC B	B počítá průchody smyčkou.
		JR Z,#245F,ARC-END	Skok, když B dosáhlo nuly.
		JR #2439,ARC-START	Skok na začátek smyčky.
2425	ARC-LOOP	RST #28,FP-CALC	Viz. text o zásobníku.
		DEFB #E1,get-mem-1	Un-1
		DEFB #31,zdvojenf	Un-1,Un-1
		DEFB #E3,get-mem-3	Un-1,Un-1,COS (G/A)
		DEFB #04,násobenf	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,násobenf	Un-1,Un-1*COS (G/A),Vn-1*SIN (G/A)
		DEFB #03,odčítánf	Un-1,Un-1*COS (G/A)-Vn-1*SIN (G/A)-Un
		DEFB #C1,st-mem-1	Un do mem-1.
		DEFB #02,výmaz	Un-1
		DEFB #E4,get-mem-4	Un-1,SIN (G/A)
		DEFB #04,násobenf	Un-1*SIN (G/A)
		DEFB #E2,get-mem-2	Un-1*SIN (G/A),Vn-1
		DEFB #E3,get-mem-3	Un-1*SIN (G/A),Vn-1,COS (G/A)
		DEFB #04,násobenf	Un-1*SIN (G/A),Vn-1*COS (G/A)
		DEFB #0F,sčítánf	Un-1*SIN (G/A)+Vn-1*COS (G/A)-Vn
		DEFB #C2,st-mem-2	Vn do mem-2.
		DEFB #02,výmaz	Jak bylo řečeno v textu, zásobník ve skutečnosti obsahuje hodnoty $X0+X$ , $Y0+Y$ , $Xn$ a $Yn$ .
		DEFB #38,konec výpočtu	Uschovej řídtač oblouků.
2439	ARC-START	PUSH BC	
		RST #28,FP-CALC	$X0+X$ , $Y0+Y$ , $Xn$ , $Yn$
		DEFB #C0,st-mem-0	$Yn$ do mem-0.
		DEFB #02,výmaz	$X0+X$ , $Y0+Y$ , $Xn$
		DEFB #E1,get-mem-1	$X0+X$ , $Y0+Y$ , $Xn$ , $Un$
		DEFB #0F,sčítánf	$X0+X$ , $Y0+Y$ , $Xn+Un$ ( $Xn+Un=Xn+1$ )
		DEFB #31,zdvojenf	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Xn+1$
		DEFB #38,konec výpočtu	Dále $Xn'$ příbližná <b>hodnota</b> dosažená podprogramem
		LD A,(#5C7D) {COORDS-lo}	pro rýsování přímek je uložena do registru A a uložena na zásobník.
		CALL #2D28,STACK-A	
		RST #28,FP-CALC	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Xn+1$ , $Xn'$
		DEFB #03,odčítánf	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Xn+1-Xn'$ = $Un'$
		DEFB #E0,get-mem-0	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Un'$ , $Yn$
		DEFB #E2,get-mem-2	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Un'$ , $Yn$ , $Vn$
		DEFB #0F,sčítánf	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Un'$ , $Yn+Vn=Yn+1$
		DEFB #C0,st-mem-0	$Yn+1$ do mem-0.
		DEFB #01,záměna	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Yn+1$ , $Un'$
		DEFB #E0,get-mem-0	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Yn+1$ , $Un'$ , $Yn+1$
		DEFB #38,konec výpočtu	$Yn'$ je aproximováno podobně jako $Xn'$
		LD A,(#5C7E) {COORDS-hi}	je okopírováno do A
		CALL #2D28,STACK-A	a uloženo na zásobník.
		RST #28,FP-CALC	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Yn+1$ , $Un'$ , $Yn+1$ , $Yn'$
		DEFB #03,odčítánf	$X0+X$ , $Y0+Y$ , $Xn+1$ , $Yn+1$ , $Un'$ , $Vn'$

	DEFB #38,konec výpočtu	
	CALL #24B7, DRAW-LINE	Je naryšován další oblouk.
	POP BC	Obnovení počítadla oblouků.
	RJNZ #2425, ARC-LOOP	Mají-li se rýsovat ještě další oblouky provede se skok.
245F	ARC-END	Souřadnice konce oblouku, který byl naryšován,
	DST #28, FP-CALC	jsou nyní
	DEFB #02, výmaz	vymazány ze zásobníku.
	DEFB #02, výmaz	
	DEFB #01, záměna	Y0+Y, X0+X
	DEFB #38, konec výpočtu	
	LD A, (#5C7D) {COORDS-lo}	X-ová souřadnice posledního oblouku, řekněme Xz',
	CALL #2D28, STACK-A	je nyní okopírována na zásobník.
	RST #28, FP-CALC	Y0+Y, X0+X, Xz'
	DEFB #03, odčítání	Y0+Y, X0+X-Xz'
	DEFB #01, záměna	X0+X-Xz', Y0+Y
	DEFB #38, konec výpočtu	
	LD A, (#5C7E) {COORDS-hi}	Obdrží se Y-ová souřadnice.
	CALL #2D28, STACK-A	
	RST #28, FP-CALC	X0+X-Xz', Y0+Y, Yz'
	DEFB #03, odčítání	X0+X-Xz', Y0+Y-Yz'
	DEFB #38, konec výpočtu	
2477	LINE-DRAW	Závěrečný oblouk je naryšován do bodu X0+X, Y0+Y,
	CALL #24B7, DRAW-LINE	nebo uzavření kružnice. Nastav barvy.
	JP #0D4D, TEMPS	

#### PODPROGRAM PRO NASTAVENÍ PARAMETRŮ

Tento podprogram je využíván jak příkazem CIRCLE, tak příkazem DRAW, aby jim nastavila počáteční parametry. Příkaz CIRCLE ji volá s parametry X, Y a poloměrem Z na vrcholu zásobníku, čteno **zdola**. Příkaz DRAW má své X, Y, SIN (G/2) a Z, (jak je definováno v DRAW ad a)), na vrcholu zásobníku. V následujícím komentáři je obsah zásobníku ukázán pouze od Z dále. Podprogram vrací v registru B počet oblouků, registr A byl již popsán v příkazech CIRCLE a DRAW, a v mem-0 až mem-5 jsou hodnoty G/A, SIN (G/2\*A), 0, COS (G/A), SIN (G/A) a G. Pro kružnici se musí G rovnat hodnotě 2\*PI.

247D	CD-PRMS1	RST #28, FP-CALC	Z
		DEFB #31, zdvojení	Z, Z
		DEFB #28, <b>odmocnina</b>	Z, SQR Z
		DEFB #34, stk-data	Z, SQR Z, 2
		DEFB #32, exponent #82	
		DEFB #00, #00, #00, #00	
		DEFB #01, záměna	Z, 2, SQR Z
		DEFB #05, dělení	Z, 2/SQR Z
		DEFB #E5, get-mem-5	Z, 2/SQR Z, G
		DEFB #01, záměna	Z, G, 2/SQR Z
		DEFB #05, dělení	Z, G*SQR Z/2
		DEFB #2A, abs	Z, G*SQR Z/2 (G!*MOD G)
		DEFB #38, konec výpočtu	Z, G*SQR Z/2 = A1
		CALL #2DD5, FP-T0-A	Hodnota A1 je uložena ze zásobníku do A, je-li to možno.
		JR C, #2495, USE-252	Je-li A1 > 256, použij 252.
		AND #FC	4*INT (A1/4) do registru A.
		ADD A, #04	Přičti k počtu oblouků v A čtyři.
		JR NC, #2497, DRAW-SAVE	Je-li to stále méně než 256, skoč.
2495	USE-252	LD A, #FC	Zde použij 252.
2497	DRAW-SAVE	PUSH AF	Nyní uschovej počet oblouků.
		CALL #2D28, STACK-A	A okopíruj tuto hodnotu také na zásobník kalkulátoru.
		RST #28, FP-CALC	Z, A
		DEFB #E5, get-mem-5	Z, A, G
		DEFB #01, záměna	Z, G, A
		DEFB #05, dělení	Z, G/A
		DEFB #31, zdvojení	Z, G/A, G/A

DEFB #1F,sin	Z,G/A,SIN (G/A)
DEFB #C4,st-mem-4	SIN (G/A) do mem-4.
DEFB #02,vymaz	Z,G/A
DEFB #31,zdvojenf	Z,G/A,G/A
DEFB #A2,stk-polovina	Z,G/A,G/A,0.5
DEFB #04,násobenf	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) do mem-1.
DEFB #01,záměna	Z,SIN (G/2*A),G/A
DEFB #C0,st-mem-0	G/A do mem-0.
DEFB #02,vymaz	Z,SIN (G/2*A)=S
DEFB #31,zdvojenf	Z,S,S
DEFB #04,násobenf	Z,S*S
DEFB #31,zdvojenf	Z,S*S,S*S
DEFB #0F,sčítánf	Z,2*S*S
DEFB #A1,stk-1	Z,2*S*S,1
DEFB #03,odčítánf	Z,2*S*S,-1
DEFB #1B,negace	Z,1-2*S*S=COS (G/A)
DEFB #C3,st-mem-3	COS (G/A) do mem-3.
DEFB #02,vymaz	Z
DEFB #3B,konec výpočtu	
POP BC	Obnov počet oblouků do B.
RET	Konec.

#### PODPROGRAM "NARÝSUJ PŘÍMKU"

Tento podprogram je volán příkazem DRAW k narysování přímky vedoucí z bodu X0,Y0 (v COORDS) do bodu X0+X,Y0+Y, kde přírůstky X a Y jsou uloženy na vrcholu kalkulátoru. Podstatou této metody je rozmístit co možná nejvíce horizontálních nebo vertikálních kroků mezi základní rozsah diagonálních kroků, s využitím algoritmu, který rozmisťuje **vertikální** kroky co možná nejrovnoměrněji.

24B7	DRAW-LINE	CALL #2307,STK-T0-BC	ABS Y do B;ABS X do C; SGN Y do D;SGN X do E.
		LD A,C	Je-li ABS X > =
		CP B	ABS Y,
		JR NC,#24C4,DL-X-GE-Y	skoč dopředu.
		LD L,C	Menší jde do L a větší jde do H.
		PUSH DE	Uchovej diagonální krok (+-1,+-1).
		XOR A	Vlož vertikální krok (+-1,0)
		LD E,A	do DE a ( <b>D-SGN Y</b> )
		JR #24CB,DL-LARGER	skoč pro nastavení registru H.
24C4	DL-X-GE-Y	OR C	Jsou-li ABS X i ABS Y nulové,
		RET Z	vrať se.
		LD L,B	Menší (zde je to ABS Y) jde do L.
		LD B,C	ABS X jde do B pro H.
		PUSH DE	Uchovej diagonální krok (+-1,+-1).
		LD D,#00	Vlož horizontální krok (0,+1) do DE.
24CB	DL-LARGER	LD H,B	Větší z ABS X,ABS Y jde do H.

Zde začíná algoritmus. Větší z ABS X a ABS Y, řekněme H, je uloženo do A a redukováno na INT (H/2). Nyní se provede H-L horizontálních nebo vertikálních kroků a L diagonálních kroků, kdy L je menší z ABS X a ABS Y, tímto způsobem: L se přičte do A; jestliže nyní je A >= H, je sníženo o H a provedou se diagonální kroky, jinak se provedou horizontální nebo vertikální kroky. Toto se opakuje H-krát (B také obsahuje H). Pověsimněte si, že se též využívají zrcadlové registry HL k uschování hodnoty COORDS.

	LD A,B	B jde do A stejně jako do H.	
	RRA	A začíná s hodnotou INT (H/2).	
24CE	D-L-LOOP	ADD A,L	L se přičte do A.

	JR C,#24D4,D-L-DIAG	Je-li to víc než 256 provede se skok na diagonální krok.
	CP H	Jestliže A < H provede se skok na horizontální, nebo
	JR C,#24DB,D-L-HR-VT	vertikální krok.
24D4	D-L-DIAG	
	SUB H	Sniž A o H a
	LD C,A	obnov jej do C.
	EXX	Nyní použij zrcadlové registry.
	POP BC	Diagonální krok do B'C'.
	PUSH BC	A úschova.
	JR #24DF,D-L-STEP	Skok k provedení kroku.
24DB	D-L-HR-VT	
	LD C,A	Ušchovej A (nesníženo) v C.
	PUSH DE	<b>Ušchovej</b> krátce na zásobníku.
	EXX	Zaměň registrové páry.
	POP BC	Vyzvedni krok.
24DF	<b>D-L-STEP</b>	
	LD HL,(#5C7D) (COORDS)	Nyní prováděj krok, nejdříve dej COORDS do H'L'.
	LD A,B	Krok Y z B' do A.
	ADD A,H	Přičti H'.
	LD B,A	Výsledek ulož do <b>B</b> .
	LD A,C	Nyní bude testován na rozsah X-ový
	INC A	krok (Y-ový krok se otestuje v podprogramu PLOT).
	ADD A,L	Přičti <b>L</b> do A a
	JR C,#24F7,D-L-RANGE	proved skok při CY=1 na další test.
	JR Z,#24F9,REPORT-B	Z=0 a CY=0 X-ová pozice -1 je mimo rozsah.
24EC	D-L-PLOT	
	DEC A	Obnov pravdivou hodnotu v A.
	LD C,A	Hodnota do C' pro rýsování.
	CALL #22E5,PLOT-SUB	Narýsuj tento krok.
	EXX	Obnov hlavní registry.
	LD A,C	C jde zpět do A pro pokračování algoritmu.
	DJNZ #24CE,D-L-LOOP	Skok na start smyčky po B-kroku (tedy po H-kroku).
	POP DE	Vyčisti strojový zásobník.
	RET	Konec.
24F7	D-L-RANGE	
	JR Z,#24EC,D-L-PLOT	Z=0 a CY=1 znamená X-ová souřadnice 255, v rozsahu.
24F9	REPORT-B	
	RST #08,ERROR-1	Ohlaš:
	DEFB #0A	B-Integer out of range.

## HODNOCENÍ VÝRAZŮ

### PODPROGRAM SCANNING

Tento podprogram se používá, aby ohodnotil výsledky dalšího výrazu. Výsledek se vrací jako "poslední hodnota" na zásobníku kalkulátoru. Pro numerické výsledky bude "poslední hodnota" skutečné FP číslo, nicméně výsledkem hodnocení řetězců je soubor parametrů představující poslední hodnotu. První z pěti bajtů není specifikován, druhý a třetí bajt obsahují adresu začátku řetězce a čtvrtý a pátý bajt jeho délku. Bit 6 systémové proměnné FLAGS je nastaven pro číselné výsledky a je nulový pro řetězce. Jestliže se výraz skládá pouze z jediného operandu, např. A ... RND ... A\$(4,3 TO 7) ..., potom "poslední hodnota" představuje hodnotu, která se získá ohodnocením daného operandu. V případě, že výraz obsahuje **jako operand funkci**, např. CHR\$ A, ... NOT A, ... SIN 1, je operační kód této funkce uložen na zásobníku, dokud není vypočtena poslední hodnota operandu. Potom se tato hodnota použije v příslušné operaci, aby se získala nová "poslední hodnota". V případě, že se mají vykonat aritmetické nebo logické operace, např. A+B, ..., A\*B, ..., A/B, potom se musí uschovat operační kód a poslední hodnota prvního argumentu, dokud není nalezena poslední hodnota druhého argumentu. A vskutku výpočet poslední hodnoty druhého argumentu v sobě může zahrnovat uložení posledních hodnot a operačních kódů na zásobník po dobu průběhu vlastního výpočtu. Proto lze zpracovat i složitější hierarchii operací. Například CHR\$(T+A)-26\*INT((T+A)/26+65) se bude provádět tak dlouho, až se dosáhne bodu, z kterého již lze mezivýsledky "demontovat", a takto může být teprve získána konečná "poslední" hodnota. Každý operační kód má přidělenou prioritu a operace s vyšší prioritou jsou vždy uskutečněny před operacemi s prioritou nižší. Podprogram začíná s registrem A obsahujícím první znak souboru a počáteční prioritou nula, která je uložena na zásobníku.

24FB	SCANNING	RST #18,GET-CHAR	Vyzvedni první znak.
		LD B,#00	Číslo priority
		PUSH BC	jde na zásobník.
24FF	S-LOOP-1	LD C,A	Hlavní bod opětovného vstupu.
		LD HL,#2596	
		CALL #16DC,INDEXER	Hledej v tabulce funkcí, kdy kód je v registru C.
		LD A,C	Obnov kód do A.
		JP NC,#2684,S-ALPHNUM	Skoč, nebyl-li kód nalezen v tabulce.
		LD B,#00	
		LD C,(HL)	Použij položku nalezenou v tabulce
		ADD HL,BC	k vytvoření požadované adresy a skoč
		JP (HL)	na tuto adresu.

Následují čtyři podprogramy, které jsou volány podprogramy pro tabulku funkcí. První z nich s názvem "SCANNING **QUOTE SUBROUTINE**" ("TESTOVÁNÍ UVOZOVEK") je použita podprogramem S-QUOTE aby zjistil, jestli každá řetězcová uvozovka má další uvozovku do páru.

250F	S-QUOTE-S	CALL #0074,CH-ADD+1	Ukazuj na další znak.
		INC BC	Zvyš délku o jednu.
		CP #0D	Je to CR? (-ENTER?)
		JP Z,#1C8A,REPORT-C	Ohlaš chybu, jestliže ano.
		CP #22	Existuje další uvozovka?
		JR NZ,#250F,S-QUOTE-S	Skoč zpět, jestliže ne.
		CALL #0074,CH-ADD+1	Ukazuj na další znak a
		CP #22	nastav Z flag jestliže se jedná o další uvozovku.
		RET	Konec.

Další podprogram, který lze nazvat: "testování dvou souřadnic", je používán programy **S-SCREEN\$**, S-ATTR a S-POINT, aby se zajistilo, že požadované souřadnice budou udány v náležitě formě.

2522	S-2-COORD	RST #20,NEXT-CHAR	Vyzvedni další znak.
		CP #28	Je to závorka "("?
		JR NZ,#252D,S-RPORT-C	Ohlaš chybu, jestliže ne.
		CALL #1C79,NEXT-2NUM	Souřadnice na zásobníku kalkulátoru.
		RST #18,GET-CHAR	Vyzvedni aktuální znak.

	CP	#29	Je to závorka ")"?
252D	S-RPORT-C	JP NZ,#1C8A,REPORT-C	Ohlaš chybu, jestliže ne.

### PODPROGRAM "SYNTAX-Z"

V tomto bodě je vložen podprogram SYNTAX-Z, který je z jiných částí programu volán 32 krát. A protože instrukce BIT 7,(IY+1) má oproti instrukci CALL o jeden bajt více, ušetří se na každém volání této instrukce jeden bajt. Jednoduchý test bitu 7 FLAGS nastaví Z flag na nulu při běhu programu a na jedničku při kontrole syntaxe.

2530	SYNTAX-Z	BIT 7,(IY+1) (FLAGS)	Testuj bit 7 FLAGS.
		RET	Konec.

Další podprogram je testovací podprogram SCREEN\$, který je používán z podprogramu S-SCREEN\$ pro nalezení znaku na souřadnicích řádek X a sloupec Y obrazovky. Prohledává pouze znaky, na které je nastavena systémová proměnná CHARS.

Poznámka: Běžně se jedná o znaky od #20 (mezera) do #7F. Uživatel však může změnou systémové proměnné CHARS testovat také další znaky (včetně uživatelské grafiky).

2535	S-SCRN%-S	CALL #2307,STK-T0-BC	X do C a Y do B (kde $0 < X < 23$ a $0 < Y < 31$ dekadicky)
		LD HL,(#5C36) (CHARS)	CHARS+256 dekadicky
		LD DE,#0100	
		ADD HL,DE	vytvoří v HL básovou adresu znakového souboru.
		LD A,C	X je okopírována do A.
		RRCA	Číslo 32 dekadicky $\times X$ mod 8+Y se formuje v A a
		RRCA	je kopírováno do E.
		RRCA	
		AND #E0	Toto je nižší bajt požadované adresy obrazovky.
		XOR B	
		LD E,A	
		LD A,C	X je opět okopírována do A.
		AND #18	Nyní je opět dekadicky $64+8*INT(X/8)$
		XOR #40	uloženo do D.
		LD D,A	DE nyní obsahuje požadovanou adresu na obrazovce.
		LD B,#60	B spočítá 96 znaků.
254F	S-SCRN-LP	PUSH BC	Ušchvej počet.
		PUSH DE	Ušchvej ukazatel obrazovky a
		PUSH HL	ukazatel na znakový soubor.
		LD A,(DE)	Vyzvedni první bitový řádek znaku z obrazovky.
		XOR (HL)	Testuj s řádkem ze znakového souboru.
		JR Z,#255A,S-SC-MTCH	Byla-li nalezena přímá shoda, skoč.
		INC A	Testuj shodu s inverzním znakem (z #FF vytvoř v A #00).
		JR NZ,#2573,S-SCR-NXT	Skoč, nebyla-li ani tato shoda nalezena.
		DEC A	Obnov #FF v A.
255A	S-SC-MTCH	LD C,A	Inverzní status (#00 nebo #FF) do C.
		LD B,#07	B počítá dalších 7 bitových řádků.
255D	S-SC-ROWS	INC D	Posuň DE na další bitový řádek (přičti 256 dekadicky).
		INC HL	Posuň HL na další řádek (tedy na další bajt).
		LD A,(DE)	Vyzvedni řádek z obrazovky,
		XOR (HL)	porovnej jej s řádkem v ROM.
		XOR C	Přidej inverzní status a
		JR NZ,#2573,S-SCR-NXT	skoč, nedochází-li k celkové shodě.
		DJNZ #255D,S-SC-ROWS	Skoč zpět, pokud nejsou porovnány všechny bitové řádky.
		POP BC	Zbav se ukazatele na soubor znakových matic a
		POP BC	ukazatele obrazovky.
		POP BC	Závěrečný počet do BC.
		LD A,#80	Kód posledního znaku +1.
		SUB B	A nyní obsahuje požadovaný kód.



	LD	BC,#0001	Nyní bude potřeba jedno místo v pracovním prostoru.
	RST	#30,BC-SPACES	Vytvoř toto místo.
	LD	(DE),A	Vlož do něj znak.
	JR	#257D,S-SCR-STO	Skoč na uložení znaku na zásobník.
2573	S-SCR-NXT	POP HL	Obnov ukazatel na znakový soubor.
	LD	DE,#0008	
	ADD	HL,DE	Posuň jej o 8 bajtů tedy na další matici tohoto souboru.
	POP	DE	Obnov ukazatel obrazovky.
	POP	BC	Obnov čítač.
	DJNZ	#254F,S-SCRN-LP	Skok zpět, protože znaků je 96.
	LD	C,B	Signál: prázdný řetězec. Nebyl-li znak nalezen,
257D	S-SCR-STO	JP #2AB2,STK-STO-\$	skoč k uložení nulového řetězce. Jinak ulož nalezený znak

Poznámka: Výstup přes STK-STO-\$ je chybou, která způsobí, že nalezený výsledek je uložený dvakrát. Zde měla být instrukce RET.

Poslední z těchto čtyř podprogramů je podprogram prohledávající atributy. Je volán programem S-ATTR, aby vrátil hodnotu ATTR (X,Y), udávající hodnotu atributu pro znak nacházející se na souřadnicích X,Y obrazovky.

2580	S-ATTR-S	CALL #2307,STK-T0-BC	X do C, Y do B (opět $0 < X < 23$ a $0 < Y < 31$ dekadicky)
		LD A,C	X je okopírováno do A a čísla
		RRCA	$32 * (X \text{ mod } 8) + Y$
		RRCA	
		RRCA	je formováno v A a okopírováno do L.
		LD C,A	Hodnota $32 * (X \text{ mod } 8) + \text{INT}(X/8)$ je také v C
		AND #E0	
		XOR B	
		LD L,A	L obsahuje nižší bajt atributové adresy.
		LD A,C	Hodnota $32*(X \text{ mod } 8)+\text{INT}(X/8)$ je zpět okopírována do A.
		AND #03	Hodnota $\text{INT}(X/8)+\#58$ je
		XOR #58	formována v A a okopírována do H,
		LD H,A	kteřé pak obsahuje vyšší bajt adresy atributu.
		LD A,(HL)	Atributový bajt je okopírován do A a
		JP #2D28,STACK-A	je proveden odskok a uložení daného bajtu na zásobník.

#### TABULKA TESTOVACÍCH FUNKCÍ

Tato tabulka obsahuje 8 funkcí a 4 operátory. Zahrnuje v sobě tedy pět nových funkcí SPEKTRA a umožňuje elegantním způsobem dosahovat některých funkcí a operátorů, které již existovaly v počítači ZX-81.

adresa	kód	doplňk	jméno	adresa prováděcího podprogramu
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]	268D
25A8	AA	BF	S-SCREEN\$	2668
25AA	AB	C7	S-ATTR	2672
25AC	A9	CE	S-POINT	267B
25AE	00		Koncový znak	

## PODPROGRAM PRO VYHODNOCENÍ FUNKCÍ

25AF S-U-PLUS RST #20,NEXT-CHAR Pro obyčejné plus se pouze přesuň na další znak a  
JP #24FF,S-LOOP-1 skoč zpět na hlavní opětovný vstup programu SCANNING.

Podprogram na prohledávání uvozovek: Tento podprogram zpracovává řetězce s uvozovkami, např. obyčejně jako "jméno", nebo složitější "téměř ""nevinná"" lež" nebo viditelně nadbytečně jako VAL\$ ""A"".

25B3 S-QUOTE RST #18,GET-CHAR Vyzvedni aktuální znak.  
INC HL Ukazuj na začátek řetězce.  
PUSH HL Ušchovej počáteční adresu.  
LD BC,#0000 Nastav délku na nulu.  
CALL #250F,S-QUOTE-S Volej porovnávací podprogram.  
JR NZ,#25D9,S-Q-PRMS Skoč, jestliže je Z = 0, což znamená: další uvozovka není

25BE S-Q-AGAIN CALL #250F,S-QUOTE-S Znovu volej pro třetí uvozovku.  
JR Z,#25BE,S-Q-AGAIN A dále pro pátou sedmou a tak dále.  
CALL #2530,SYNTAX-Z Testuješ-li syntax,  
JR Z,#25D9,S-Q-PRMS skoč na reset bitu č. 6 FLAGS a pokračuj v prohledávání.  
RST #30,BC-SPACES Vytvoř místo pro řetězec a závěrečnou uvozovku.  
POP HL Vyzvedni ukazatel na start a  
PUSH DE ulož ukazatel na první místo.

25CB S-Q-COPY LD A,(HL) Vyzvedni znak z řetězce,  
INC HL posuň ukazatel na další.  
LD (DE),A Okopíruj poslední znak do pracovního prostoru a  
INC DE posuň ukazatel na další místo.  
CP #22 Je poslední znak uvozovka?  
JR NZ,#25CB,S-Q-COPY Jestliže ne, skok na přenos další uvozovky.  
LD A,(HL) Je-li však další znak uvozovka, okopíruj až tu další  
INC HL  
CP #22

25D9 S-Q-PRMS JR Z,#25CB,S-Q-COPY Jinak ukonči kopírování.  
DEC BC Vytvoř v BC skutečnou délku.

Poznámka: Povšimněte si, že první uvozovka nebyla započítána do délky. Koncová uvozovka ale byla a proto je nyní i tato vynechána. První, třetí, pátá, ... uvozovka uvnitř řetězce byla počítána, ale druhá, čtvrtá, šestá, ... počítány nebyly.

25DB S-STRING POP DE Obnov start kopírovaného řetězce.  
LD HL,#5C3B Toto je FLAGS a tento vstupní bod se používá,  
RES 6,(HL) kdykoliv má být bit 6 FLAGS resetován a  
BIT 7,(HL) parametry řetězce  
CALL NZ,#2AB2,STK-STO-\$ uloženy na zásobník pokud se provádí řádek.  
JP #2712,S-CONT-2 Skoč na pokračování prohledávání řádku.

Poznámka: Povšimněte si, že při kopírování řetězců do pracovního prostoru každý pár uvozovek uvnitř řetězce ("" ) byl snížen na jednu uvozovku (").

25EB S-BRACKET RST #20,NEXT-CHAR Při prohledávání závorek vyzvedne další znak a  
CALL #24FB,SCANNING volá rekurzivně SCANNING.  
CP #29 Nebyla-li nalezena správná závorka,  
JP NZ,#1C8A,REPORT-C bude ohlášena chyba.  
RST #20,NEXT-CHAR Jinak se pokračuje v prohledávání.  
JP #2712,S-CONT-2

25F5 S-FN JP #27BD,S-FN-SBRN Podprogram prohledávání funkcí.

Tento podprogram pro uživatelem definované funkce pokračuje do podprogramu prohledávání funkcí.

25F8 S-RND	CALL #2530,SYNTAX-Z JR Z,#2625,S-RND-END LD BC,(#5C76) (SEED) CALL #2D2B,STACK-BC RST #2B,FP-CALC DEFB #A1,stk-jedna DEFB #0F,přičtení DEFB #34,stk-data DEFB #37,exponent #87 DEFB #16,(#00,#00,#00) DEFB #04,násobení DEFB #34,stk-data DEFB #80,(čtyři bajty) DEFB #41,exponent #91 DEFB #00,#00,#80,(#00) DEFB #32,n-mod-m DEFB #02,výmaz DEFB #A1,stk-jedna DEFB #03,odečtení DEFB #31,zdvojení DEFB #3B,konec výpočtu CALL #2DA2,FP-T0-BC LD (#5C76),BC (SEED) LD A,(HL) AND A JR Z,#2625,S-RND-END SUB #10 LD (HL),A	Pokud se nekontroluje syntaxe, skoč na výpočet náhodného čísla. Vyzvedni aktuální hodnotu SEED a ulož ji na zásobník kalkulátoru. Nyní použij kalkulátoru. Poslední hodnotou je nyní SEED + 1. Ulož na zásobník číslo 75 (dekadicky).  (SEED + 1) * 75 Nyní jsou bajty expandovány, takže na zásobník jde číslo 65537.  (SEED+1)*75/65537 abys dosáhl "zbytku" a "odpovědi". Vymaž odpověď.  Poslední hodnota je nyní zbytek - 1. Zdvoj tuto poslední hodnotu a ukonči výpočet. Užij poslední hodnotu jako nový údaj pro SEED. Vyzvedni exponent poslední hodnoty a jestliže je nulový skoč dopředu. Sniž exponent, což znamená: vyděl hodnotou 65536, aby se dosáhlo požadovaného rozsahu hodnot. Odskoč přes podprogram PI.
2625 S-RND-END	JR #2630,S-PI-END	

Podprogram "hledání PI": nalezne výraz PI a pokud se nekontroluje syntaxe, uloží na zásobník hodnotu PI jako "poslední hodnotu".

2627 S-PI	CALL #2530,SYNTAX-Z JR Z,#2630,S-PI-END RST #2B,FP-CALC DEFB #A3,stk-pi/2 DEFB #3B,konec výpočtu INC (HL)	Test na kontrolu syntaxe. Skoč, jestliže kontroluješ syntaxi. Nyní použij kalkulátor. PI/2 je uloženo na <b>zás. kalk.</b> jako "poslední hodnota".  Exponent je zvětšen, čímž se dosáhne výsledku PI.
2630 S-PI-END	RST #20,NEXT-CHAR JP #26C3,S-NUMERIC	Přesun na další znak. Skok dopředu.
2634 S-INKEY\$	LD BC,#105A RST #20,NEXT-CHAR CP #23 JP Z,#270D,S-PUSH-PO LD HL,#5C3B RES 6,(HL) BIT 7,(HL) JR Z,#2665,S-INK\$-EN CALL #028E,KEY-SCAN LD C,#00 JR NZ,#2660,S-INK\$-STK CALL #031E,K-TEST JR NC,#2660,S-INK\$-STK DEC D LD E,A	Priorita #10 operační kód #5A pro podprogram READ-IN.  Je-li další znak <b>klíčezek</b> , skoč, neboť bude následovat číselný argument. Toto je FLAGS. Resetuj bit 6 pro řetězcový výsledek a testuj bit 7 na kontrolu syntaxe. Skoč, je-li to potřeba. Vyzvedni <b>hodnotu klávesy</b> do DE. Připrav prázdný řetězec a vlož jej na zásobník, bylo-li stisknuto více kláves. Testuj hodnotu klávesy a ulož hodnoty prázdného řetězce, jestliže hodnoty kláves nevyhovují. #FF do D pro L mod (set bit 3). Hodnota klávesy do registru E pro dekodování.

	CALL #0333,K-DECODE	Dekóduj hodnotu klávesy.
	PUSH AF	Uschovej hodnotu ASCII.
	LD BC,#0001	Je potřeba jedno místo v pracovním prostoru.
	RST #30,BC-SPACES	Vytvoř toto místo.
	POP AF	Obnov hodnotu ASCII.
	LD (DE),A	Příprav jeho uschování na zásobník jakožto řetězce.
	LD C,#01	Jeho délka je 1.
2660 S-IK\$-STK	LD B,#00	Dokončí parametr délky a
	CALL #2AB2,STK-STO-\$	ulož požadovaný řetězec.
2665 S-INK\$-EN	JP #2712,S-CONT-2	Skoč dopředu.
2668 S-SCREEN\$	CALL #2522,S-2-COORD	Testuj, zda byly udány 2 souřadnice.
	CALL NZ,#2535,S-SCRN\$-S	Pokud nekontroluješ syntaxi, volej podprogram.
	RST #20,NEXT-CHAR	Vyzvedni další znak
	JP #25DB,S-STRING	a skoč zpět.
2672 S-ATTR	CALL #2522,5-2-COORD	Testuj, zda byly udány 2 souřadnice.
	CALL NZ,#2580,S-ATTR-S	Pokud nekontroluješ syntaxi, volej podprogram.
	RST #20,NEXT-CHAR	Vyzvedni další znak
	JR #26C3,S-NUMERIC	a skoč zpět.
267B S-POINT	CALL #2522,S-2-COORD	Testuj, zda byly udány 2 souřadnice.
	CALL NZ,#22CB,POINT-SUB	Pokud nekontroluješ syntaxi, volej podprogram.
	RST #20,NEXT-CHAR	Vyzvedni další znak
	JR #26C3,S-NUMERIC	a skoč zpět.
2684 S-ALPHNUM	CALL #2C88,ALPHANUM	Je znak alfanumerický ?
	JR NC,#26DF,S-NEGATE	Skoč, jestliže ne.
	CP #41	
	JR NC,#26C9,S-LETTER	Nyní skoč, je-li to písmeno, jinak pokračuj do S-DECIMAL.

Tento podprogram zpracovává číselné výrazy začínající desetinnou čárkou nebo číslicí. Také se stará o příkaz BIN, který je zpracováván podprogramem "decimal na FP".

268D S-DECIMAL CALL #2530,SYNTAX-Z  
(EQU.S-BIN) JR NZ,#26B5,S-STK-DEC      Skoč dopředu, jestliže se provádí řádek.

Akce, která se nyní zahájí se značně odlišuje pro kontrolu syntaxe a pro běh programu. Kontroluje se syntaxe, je vypočítána FP forma a okopírována do aktuálního basicovského řádku. Provádí-li se program, bude FP vždy k dispozici, takže je okopírována na zásobník kalkulátoru aby vytvořila "poslední hodnotu".

Během kontroly syntaxe:

CALL #2C9B,DEC-T0-FP	Je nalezena FP forma.
RST #18,GET-CHAR	HL se nastaví, aby ukazoval za poslední číslici.
LD BC,#0006	Je potřeba 6 míst.
CALL #1655,MAKE-ROOM	Vytváří se prostor v basicovém řádku.
INC HL	HL ukazuje na první volné místo.
LD (HL),#0E	Vloží se značka FP čísla a
INC HL	postoupí se na další číslo.
EX DE,HL	Nyní bude ukazatel potřeba v DE.
LD HL,(#5C65) (STKEND)	Vyzvedni STKEND.
LD C,#05	Jedná se o 5 bajtů, které budou přeneseny.
AND A	Vyčistí Cy flag.
SBC HL,BC	Nový STKEND = starý STKEND - 5.
LD (#5C65),HL (STKEND)	Přenes FP číslo ze zásobníku kalkulátoru
LDIR	do basicového řádku.

EX	DE,HL	Obnov ukazatel v HL
DEC	HL	a ukazuj na poslední přidany bajt.
CALL	#0077,TEMP-PTR1	Tento podprogram nastaví CH-ADD.
JR	#26C3,S-NUMERIC	Skoč dopředu.

Za chodu programu:

26B5	S-STK-DEC	RST #18,GET-CHAR	Vyzvedni aktuální znak.
26B6	S-SD-SKIP	INC HL	Nyní se přesuň na další znak, pokud
		LD A,(HL)	není nalezena
		CP #0E	značka pro FP číslo.
		JR NZ,#26B6,S-SD-SKIP	
		INC HL	Ukazuj na 1. bajt čísla a
		CALL #33B4,STACK-NUM	přenes toto číslo na zásobník.
		LD (#5C5D),HL (CH-ADD)	Nastav CH-ADD.

Číselný výsledek musí být identifikován, když přichází z RND, PI, ATTR, POINT nebo dekadického čísla, nastavením bitu 6 ve FLAGS.

26C3	S-NUMERIC	SET 6,(IY+1) (FLAGS)	Nastav signál: numerická hodnota.
		JR #26DD,S-CONT-1	Skoč dopředu.

#### PODPROGRAM HODNOCENÍ PROMĚNNÝCH

Po identifikaci názvu proměnné a zavolání LOOK-VARS, kde byla zjištěna existence proměnné v oblasti proměnných nebo programové oblasti (DEF FN pro uživatelem definované funkce FN), uloží se nalezená číselná hodnota na zásobník kalkulátoru pomocí podprogramu STACK-NUM. Řetězcové pole však musí být předáno na zásobník podprogramem STK-VAR, nebo při uživatelem definované funkci podprogramem STK-F-ARG (který je volán z podprogramu LOOK-VARS).

26C9	S-LETTER	CALL #28B2,LOOK-VARS	V případě, že proměnná neexistuje,
		JP C,#1C2E,REPORT-2	vypiš chybové hlášení.
		CALL Z,#2996,STK-VAR	Jinak ulož par. řet. a vrať adresu číselného elementu.
		LD A,(#5C3B) (FLAGS)	Vyzvedni FLAGS.
		CP #C0	Testuj bit 6 a 7 současně.
		JR C,#26DD,S-CONT-1	Jeden nebo oba jsou vynulovány.
		INC HL	Číselná hodnota musí být uložena na zásobník.
		CALL #33B4,STACK-NUM	Přenes tam číslo.
26DD	S-CONT-1	JR #2712,S-CONT-2	Skoč dopředu.

Znak je testován na kód pro "-", čímž se identifikuje "unární mínus". Před provedením testu je registr B nastaven na prioritu #09 a registr C na hodnotu #DB, což je kód této operace.

26DF	S-NEGATE	LD BC,#09DB	Priorita #09, operační kód #DB.
		CP #2D	Je to "-" ?
		JR Z,#270D,S-PUSH-PO	Skoč dopředu, když ano.

Nyní se testuje znak na kód "VAL\$" s prioritou #10 a operačním kódem #18.

		LD BC,#1018	Priorita #10, operační kód #18.
		CP #AE	Je to VAL\$ ?
		JR Z,#270D,S-PUSH-PO	Skoč dopředu je-li to VAL\$.

Aktuální znak musí nyní představovat některou z funkcí CODE až NOT s kódem #AF až #C3.

SUB	#AF	Rozsah funkcí je změněn a #AF až #C3 na #00 až #14.
JP	C,#1C8A,REPORT-C	Ohlaš chybu, jsou-li mimo rozsah.

Funkce NOT je identifikována a zpracována odděleně od ostatních.

LD	BC,#04F0	Priorita #04, operační kód #F0.
CP	#14	Je to funkce NOT?
JR	Z,#270D,S-PUSH-PO	Skoč, jestliže ano.
JP	NC,#1C8A,REPORT-C	Znovu testuj rozsah.

Zbývající funkce mají prioritu 16. Operační kódy pro tyto funkce jsou nyní vypočítány. Funkce, které operují na řetězcích potřebují mít bit 6 nulový a funkce, které vracejí řetězcové výsledky potřebují mít bit 7 nulový ve svých operačních kódech.

LD	B,#10	Priorita 16.
ADD	A,#DC	Funkční rozsah je nyní #DC až #EF.
LD	C,A	Převed operační kód do C.
CP	#DF	Odděl CODE, VAL a LEN, které operují
JR	NC,#2707,S-NO-T0-\$	na řetězcích pro dosažení numerických hodnot.
RES	6,C	
2707 S-NO-T0-\$	CP #EE	Odděl STR\$ a CHR\$, které operují na číslech a dávají
	JR C,#270D,S-PUSH-PO	řetězcové výsledky.
	RES 7,C	Poznač operační kód. Ostatní operační kódy mají
		bit 7 a 6 = 1.

Prioritní a operační kódy pro funkci, která se právě posuzuje, jsou nyní uloženy na strojový zásobník, kde se vytváří určitá posloupnost těchto operací.

270D S-PUSH-PO	PUSH BC	Ulož na zásobník prioritní a operační kód
	RST #20,NEXT-CHAR	
	JP #24FF,S-LOOP-1	než se posuneš na posouzení další části výrazu.

Zde pokračuje hodnocení řádku. Aktuální argument může být následován "(", binárním operátorem, nebo je-li to konec výrazu, pak znakem CR, oddělovačem nebo THEN.

2712 S-CONT-2	RST #18,GET-CHAR	Vyzvedni aktuální
2713 S-CONT-3	CP #28	znak a
	JR NZ,#2723,S-OPERTR	skoč, není-li to "(", která indikuje výraz se závorkami.

Jestliže je poslední hodnota číselná, potom výraz v závorkách je pravdivým podvýrazem a musí být ohodnocen samostatně. Je-li však poslední hodnota řetězec, potom výraz v závorkách představuje element pole nebo část řetězce. Potom zavoláním podprogramu SLICING je tento výraz upraven tak jak je potřeba.

BIT	6,(IY+1) (FLAGS)	
JR	NZ,#2734,S-LOOP	Skoč dopředu, když obsluhuje číselný výraz v závorkách.
CALL	#2A52,SLICING	Modifikuj parametry "poslední hodnoty".
RST	#20,NEXT-CHAR	
JR	#2713,S-CONT-3	Přesuň se k posouzení dalšího znaku.

Jestliže aktuální znak je skutečně binární operátor, bude mít operační kód v rozsahu #C3 až #CF a příslušný kód priority.

2723 S-OPERTR	LD B,#00	Původní kód do BC pro hledání v tabulce operátorů.
	LD C,A	
	LD HL,#2795	Ukazatel na tabulku.
	CALL #16DC,INDEXER	Hledej v tabulce.
	JR NC,#2734,S-LOOP	Skoč dopředu, nebyla-li operace nalezena.
	LD C,(HL)	Vyzvedni příslušný kód z tabulky.
	LD HL,#26ED	Ukazatel na tabulku priorit: #26ED+#C3=#27B0 je
		první adresa.

ADD HL,BC	Hledej v tabulce.
LD B,(HL)	Vyzvedni příslušnou prioritu.

Nyní se vstupuje do hlavní smyčky tohoto programu. V této fázi již je:

a) Poslední hodnota na zásobníku kalkulátoru.

b) Počáteční prioritní znak na zásobníku pod "hromadou" **funkcí a binárních operačních kódů neznámé velikosti**. Ovšem tato "hromada" může být též nulová.

c) BC obsahuje aktuální operaci a prioritu, která při dosažení konce bude mít hodnotu nula.

Konečně jsou ze zásobníku sejmuty poslední hodnoty a porovnány oproti aktuální operaci a prioritě. Jestliže aktuální priorita je vyšší než poslední priorita, potom se provede výstup ze smyčky, protože aktuální priorita je **považována za vyšší než priorita operace poslední. Takže je-li aktuální priorita považována za nižší, potom** je provedena tato poslední operace. Aktuální operace a priorita jde zpátky na zásobník, aby mohla být dále zpracována smyčkou. Tímto způsobem se hierarchie funkcí a binárních operací, které čekají v jakési frontě, zpracovává ve správném pořadí.

2734 S-LOOP	POP DE	Vyzvedni poslední operaci a její prioritu.
	LD A,D	Priorita jde do registru A.
	CP B	Porovnej "poslední" oproti "aktuálnímu".
	JR C,#2773,S-TIGHTER	Výstup na čekání na argument.
	AND A	Jsou obě priority nulové? Jestliže ano
	JP Z,#0018,GET-CHAR	výstup přes GET-CHAR, čímž se "poslední hodnota" <b>stane</b> výslednou.

Než je provedena poslední operace, je funkce USR rozdělena na "USR číslo" nebo "USR řetězec" podle logické hodnoty bitu 6 FLAGS, nastaveného již dříve při ukládání argumentu funkce USR jako "poslední hodnoty".

	PUSH BC	Ulož aktuální hodnotu na zásobník.
	LD HL,#5C3B	Toto je FLAGS.
	LD A,E	Poslední operace se testuje na kód pro USR,
	CP #ED	dá USR číslo, nebylo-li modifikováno.
	JR NZ,#274C,S-STK-LST	Skoč, nebylo-li to USR.
	BIT 6,(HL)	Testuj bit 6 FLAGS.
	JR NZ,#274C,S-STK-LST	Skoč, byl-li nastaven (USR číslo).
	LD E, #99	Změň poslední kód: doplněk #19+#80 pro <b>řetězec či číslo</b>
274C S-STK-LST	PUSH DE	Ulož přechodně poslední hodnoty
	CALL #2530,SYNTAX-Z	
	JR Z,#275B,S-SYNTEST	a nedělej nic, kontroluješ-li syntax.
	LD A,E	Poslední <b>oper.kód</b> .
	AND #3F	Vymaž bity 6 a 7,
	LD B,A	aby se převedlo <b>op.kód</b> na kalkulátorový doplněk.
	RST #2B,FP-CALC	Užij kalkulátor.
	DEFB #3B,FP-calc-2	Proveď aktuální operaci.
	DEFB #3B,Konec výpočtu	Konec.
	JR #2764,S-RUNTEST	Skoč dopředu.

Důležitou částí kontroly syntaxe je též testování operace aby se zajistilo, že povaha poslední hodnoty je správného typu pro posuzovanou operaci.

275B S-SYNTEST	LD A,E	Vyzvedni poslední operační kód.
	XOR (IY+1) (FLAGS)	Testuj typ poslední hodnoty oproti požadavkům na funkci.
	AND #40	Aby byla syntaxe správná, musí se shodovat.
2761 S-RPORT-C	JP NZ,#1C8A,REPORT-C	Skoč, jestliže syntaxe selhala.

Před návratem projdi smyčkou, aby se povaha poslední hodnoty zaznamenala do systémové proměnné FLAGS.

2764	S-RUNTEST	POP DE	Vyzvedni poslední operační kód.
		LD HL,#5C3B	Toto je FLAGS.
		SET 6,(HL)	Předpokládej, že výsledek je číselný.
		BIT 7,E	Je-li povaha poslední hodnoty číselná,
		JR NZ,#2770,S-LOOPEND	skoč dopředu.
		RES 6,(HL)	Je to řetězec.
2770	S-LOOPEND	POP BC	Vyzvedni aktuální hodnotu do BC a
		JR #2734,S-LOOP	Skoč zpět.

Kdykoliv je "aktuální" operace" s vyšší prioritou, potom poslední a aktuální hodnoty jdou zpátky na zásobník. Nicméně, když aktuální operace **pracuje s řetězcí**, je operační kód modifikován aby se signalizoval tento požadavek.

2773	S-TIGHTER	PUSH DE	Poslední hodnoty jdou na zásobník.
		LD A,C	Vyzvedni aktuální operační kód.
		BIT 6,(IY+1) (FLAGS)	Jedná-li se o číselný operand,
		JR NZ,#2790,S-NEXT	neupravuj <b>tento</b> kód.
		AND #3F	Nuluj bit 6 a 7.
		ADD A,#08	Zvyš kód o #08 a
		LD C,A	vrať ho v registru C.
		CP #10	Je to operace AND?
		JR NZ,#278B,S-NOT-AND	Skoč, jestliže ne.
		SET 6,C	AND vyžaduje číselný operand.
		JR #2790,S-NEXT	Skoč dopředu.
2788	S-NOT-AND	JR C,#2761,S-RPORT-C	Operace -,*,/,^,OR nelze provádět mezi řetězci.
		CP #17	Jedná se o "+"?
		JR Z,#2790,S-NEXT	Skoč, jestliže ano.
		SET 7,C	Další operace produkují číselný výsledek.
2790	S-NEXT	PUSH BC	Aktuální hodnoty jdou na zásobník.
		RST #20,NEXT-CHAR	Posuzuj další znak
		JP #24FF,S-LOOP-1	a skoč zpět do smyčky.

## TABULKA OPERÁTORŮ

lokace	kód op.	kód operátor	lokace	kód op.	kód operátor
2795	2B	CF +	27A3	3C	CD <
2797	2D	C3 -	27A5	C7	C9 <=
2799	2A	C4 *	27A7	C8	CA >=
279B	2F	C5 /	27A9	C9	CB <>
279D	5E	C6 ^	27AB	C5	C7 OR
279F	3D	CE =	27AD	C6	C8 AND
27A1	3E	CC >	27AF	00	koncový znak

## TABULKA PRIORIT

lokace	priorita	operátor	lokace	priorita	operátor
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	<=			



## PODPROGRAM HODNOCENÍ FUNKCÍ

Tento podprogram je volán z podprogramu "vyhodnocení FN", aby vyhodnotila funkce definované uživatelem v basicovém řádku. Podprogram lze rozdělit na čtyři části :

a) Kontrola syntaxe příkazu FN během kontroly syntaxe.

b) Při běhu programu se provede hledání příkazu DEF FN v programové oblasti a porovnávají se jména funkcí dokud se nenalezne shoda. Jinak je ohlášena chyba.

c) Jsou ohodnoceny argumenty funkce voláním podprogramu SCANNING.

d) Je ohodnocena samotná funkce (SCANNINGem, který zase volá LOOK-VARS a tím i "SF-ARGMTS").

27BD	S-FN-SBRN	CALL #2530,SYNTAX-Z	Pokud se nekontroluje
		JR NZ,#27F7,SF-RUN	syntaxe, skoč na SF-RUN.
		RST #20,NEXT-CHAR	Vyzvedni první znak názvu.
		CALL #2C8D,ALPHA	Není-li alfanumerický,
		JP NC,#1C8A,REPORT-C	ohlaš chybu.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CP #24	Je to "\$" ?
		PUSH AF	Uschovej Z flag na zásobník
		JR NZ,#27D0,SF-BRKT-1	a skoč, nebyl-li to "\$".
		RST #20,NEXT-CHAR	Jinak vyzvedni další znak.
27D0	SF-BRKT-1	CP #28	Nejedná se o závorku
		JR NZ,#27E6,SF-RPRT-C	ohlaš chybu.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CP #29	Je to závorka ")" ?
		JR Z,#27E9,SF-FLAG-6	Skoč jestliže ano, neboť nejsou žádné argumenty.
27D9	SF-ARGMTS	CALL #24FB,SCANNING	Votej SCANNING, který kontroluje syntax arg, a vloží FP.
		RST #18,GET-CHAR	Vyzvedni znak, který následuje za argumentem.
		CP #2C	Není-li to cárka ",",
		JR NZ,#27E4,SF-BRKT-2	skoč, neboť nejsou žádné argumenty.
		RST #20,NEXT-CHAR	První znak dalšího argumentu.
		JR #27D9,SF-ARGMTS	Zpátky do smyčky k prozkoumání dalšího argumentu.
27E4	SF-BRKT-2	CP #29	Je současný znak závorka ")" ?
27E6	SF-RPRT-C	JP NZ,#1C8A,REPORT-C	Ohlaš chybu, není-li to závorka ")".
27E9	SF-FLAG-6	RST #20,NEXT-CHAR	Ukazuj na další znak v basicovém řádku.
		LD HL,#5C3B	Toto je FLAGS.
		RES 6,(HL)	Předpokládá se řetězcová funkce.
		POP AF	Obnov Z flag a skoč
		JR Z,#27F4,SF-SYN-EN	je-li výsledek funkce řetězcového typu.
		SET 6,(HL)	Jinak nastav bit 6 ve FLAGS.
27F4	SF-SYN-EN	JP #2712,S-CONT-2	Skoč zpět při vyhodnocování řádku.

ad b) Při běhu programu se musí najít příslušný příkaz DEF FN.

27F7	SF-RUN	RST #20,NEXT-CHAR	Vyzvedni první znak jména.
		AND #DF	Resetuj bit 5 pro velká písmena.
		LD B,A	Okopíruj jméno do B.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		SUB #24	Odečti #24, což je kód pro znak "\$".
		LD C,A	Výsledek ulož do C (nula pro řetězce, jinak číselná fce).
		JR NZ,#2802,SF-ARGMT-1	Skoč, jedná-li se o číselnou funkci.
		RST #20,NEXT-CHAR	Vyzvedni další znak. Je to závorka "(".
2802	SF-ARGMT-1	RST #20,NEXT-CHAR	Vezmi první znak prvního argumentu.
		PUSH HL	Uschovej ukazatel na zásobník a

	LD HL,(#5C53) (PROG)	ukazuj na začátek programu
	DEC HL	Zpět o jedno místo.
2808 SF-FND-DF	LD DE,#00CE	Hledat se bude DEF FN.
	PUSH BC	Ušchovej název a řetězcový statut.
	CALL #1D86,LOOK-PROG	Nyní prohledávej program.
	POP BC	Obnov název a řetězcový statut.
	JR NC,#2814,SF-CP-DEF	Skoč, byl-li příkaz DEF FN nalezen.
2812 REPORT-P	RST #08,ERROR-1	Jinak ohlaš:
	DEFB #18	P-FN without DEF

Po nalezení příkazu DEF FN je testováno jméno a status obou **funkcí**. Neshodují-li se, pokračuje se v hledání.

2814 SF-CP-DEF	PUSH HL	Má-li se hledat dál, uschovej ukazatel na znak DEF FN.
	CALL #28AB, FN-SKPOVR	Vyzvedni název funkce DEF FN.
	AND #DF	Resetuj bit 5 pro velká písmena.
	CP B	Shoduje se název pro hledanou FN?
	JR NZ,#2825,SF-NOT-FD	Skoč, jestliže ne.
	CALL #28AB, FN-SKPOVR	Vyzvedni další znak v příkazu DEF FN.
	SUB #24	Odečti #24 což je znak "\$".
	CP C	Porovnej status <b>funkce</b> .
	JR Z,#2831,SF-VALUES	Při kompletní shodě skoč.
2825 SF-NOT-FD	POP HL	Obnov ukazatel na DEF FN.
	DEC HL	Krok zpět.
	LD DE,#0200	
	PUSH BC	Příprav se na další prohledávání a k
	CALL #198B,EACH-STMT	nalezení konce příkazu použij porovnávací podprogram.
	POP BC	Mezitím bylo uloženo jméno a status.
	JR #2808,SF-FND-DF	Skoč zpět na další hledání.

ad c) Nyní je nalezen správný příkaz DEF FN. Argumenty příkazu FN budou ohodnoceny opakovaným voláním SCANNINGU a **5 bajtů jejich hodnoty** (nebo parametry pro řetězce) budou uloženy do příkazu DEF FN, do míst vytvořených při kontrole syntaxe. HL bude použit jako ukazatel na příkaz DEF FN (v případě volání podprogramu FN-SKPOVR), zatímco CH-ADD ukazuje na příkaz FN (a volá si RST #20 v případě potřeby).

2831 SF-VALUES	AND A	Ukazuje-li HL na znak "\$", přesuň se
	CALL Z,#28AB, FN-SKPOVR	na znak "(".
	POP DE	Odhoď ukazatel na DEF FN.
	POP DE	Vyzvedni ukazatel na první argument
	LD (#5C5D),DE (CH-ADD)	FN a okopíruj ho do CH-ADD.
	CALL #28AB, FN-SKPOVR	Přeskoč "(",
	PUSH HL	uschovej tento ukazatel na zásobník.
	CP #29	Ukazuje na <b>závorku</b> ?
	JR Z,#2885,SF-R-BR-2	Jestliže ano, skoč, protože funkce nemá argumenty.
2843 SF-ARG-LP	INC HL	Ukazuj na další
	LD A,(HL)	kód a dej ho do A.
	CP #0E	Je to značka pro FP číslo?
	LD D,#40	Nastav bit 6 v registru D pro numerický argument.
	JR Z,#2852,SF-ARG-VL	Skoč, jednalo-li se o <b>numer.</b> argument.
	DEC HL	HL bude ukazovat na znak "\$" a ne na řídicí kód.
	CALL #28AB, FN-SKPOVR	
	INC HL	HL nyní ukazuje na značku FP čísla.
	LD D,#00	Bit 6 je vynulován, signál: řetězec.
2852 SF-ARG-VL	INC HL	Ukazuj na první z pěti bajtů v DEF FN.
	PUSH HL	Ušchovej tento ukazatel na zásobník.
	PUSH DE	Ušchovej "řetězcový status" argumentu.
	CALL #24FB, SCANNING	Ohodnoť argument.

	POP AF	Dej <b>v lajku</b> "číslo/fetězec" do A.	
	XOR (IY+1) (FLAGS)	Testuj bit 6 oproti	
	AND #40	výsledku ze SCANNINGu	
	JR NZ,#288B,REPORT-Q	a vypiš hlášení Q jestliže se neshodují.	
	POP HL	Vyzvedni ukazatel na první z pěti <b>bajtů</b> v DEF FN a	
	EX DE,HL	dej ho do DE.	
	LD HL,(#5C65) (STKEND)	HL nechť ukazuje na STKEND.	
	LD BC,#0005	BC je použito jako počítadlo 5 bajtů které se přenesou.	
	SBC HL,BC	Zmenši STKEND o 5 a	
	LD (#5C65),HL (STKEND)	tak vymaž poslední hodnotu ze zásobníku.	
	LDIR	Okopíruj pět bajtů do míst v DEF FN.	
	EX DE,HL	HL ukazuje na další kód.	
	DEC HL	Zajisti, aby HL ukazovalo na znak nad pěti bajty.	
	CALL #28AB, FN-SKPOVR		
	CP #29	Je to závorka ")"?	
	JR Z,#2885,SF-R-BR-2	Skoč jestliže ano, neboť nejsou další argumenty pro DEFFN	
	PUSH HL	Je to čárka ",", uschovej <b>tedy její ukazatel</b> .	
	RST #18,GET-CHAR	Vyzvedni znak za posledním argumentem <b>příkazů</b> FN.	
	CP #2C	Není-li to čárka,	
	JR NZ,#288B,REPORT-Q	skoč, neboť jsou to popletené argumenty pro FN a DEF FN.	
	RST #20,NEXT-CHAR	Posuň CH-ADD na další argument FN.	
	POP HL	Vyzvedni ukazatel na ",", v příkazu DEF FN.	
	CALL #28AB, FN-SKPOVR	Posuň HL na další argument v DEF FN.	
	JR #2843,SF-ARG-LP	Skoč zpět k posouzení tohoto argumentu.	
2885	SF-R-BR-2	PUSH HL	Uschovej ukazatel na závorku ")" v DEF FN.
	RST #18,GET-CHAR	Vyzvedni znak po posledním argumentu v FN.	
	CP #29	Je to závorka ")" ?	
	JR Z,#288D,SF-VALUE	Jestliže ano, skoč k ohodnocení <b>funkce</b> ,	
288B	REPORT-Q	RST #08,ERROR-1	jinak ohlaš:
	DEFB #19	Q-Parameter error	

d) Konečně je funkce sama ohodnocena voláním SCANNING, když předtím DEFADD obsahovala adresu argumentů tak, jak se vyskytovaly v příkazu DEF FN. Tím je zajištěno, že při zavolání **SCANNINGU** LOOK-VARS budou prohledány nejprve tyto argumenty a pak teprve proměnné.

288D	SF-VALUE	POP DE	Obnov ukazatel na ")" v DEF FN.
		EX DE,HL	Dej tento ukazatel
		LD (#5C5D),HL (CH-ADD)	do HL a uschovej ho v CH-ADD.
		LD HL,(#5C0B) (DEFADD)	Vyzvedni starou hodnotu DEFADD.
		EX (SP),HL	Ulož ji na zásobník a současně vyzvedni
		LD (#5C0B),HL (DEFADD)	adresu <b>DEF FN</b> a ulož ji do DEFADD.
		PUSH DE	Uschovej adresu ")" v příkazu FN.
		RST #20,NEXT-CHAR	Posuň CH-ADD na poslední ")" a "-"
		RST #20,NEXT-CHAR	na začátku výrazu pro DEF FN.
		CALL #24FB,SCANNING	Nyní ohodnot <b>funkce</b> .
		POP HL	Obnov adresu závorky v příkazu FN a
		LD (#5C5D),HL (CH-ADD)	uschovej ji v CH-ADD.
		POP HL	Obnov původní hodnotu
		LD (#5C0B),HL (DEFADD)	DEFADD a ulož ji zpět.
		RST #20,NEXT-CHAR	Vyzvedni další znak v basicovém řádku.
		JP #2712,S-CONT-2	Skoč zpět do SCANNINGu.

#### PODPROGRAM "FN-SKPOVR" (SKOKY VE FUNKCI)

Tento podprogram se používá z podprogramů FN a STK-F-ARG aby posouval HL po příkazu DEF FN, aniž by byla CH-ADD změněna, neboť tato ukazuje na příkaz FN.

28AB	FN-SKPOVR	INC HL	Ukazuj na další kód v příkazu.
------	-----------	--------	--------------------------------

LD	A,(HL)	Dej ho do A a je-li
CP	#21	to řídící znak nebo
JR	C,#28AB, FN-SKPOVR	mezera, přeskoč ho.
RET		Jinak se vrať.

### PODPROGRAM \*PROHLEDEJ PROMĚNNÉ\*

Tento podprogram se volá kdykoliv je potřeba prohledávat oblast proměnných, nebo argumentů DEF FN, jak je třeba. Vstupuje se do ní se systémovou proměnnou CH-ADD nastavenou na první písmeno názvu proměnné, jejíž umístění se hledá. Jméno se bude nacházet v programové nebo pracovní oblasti. Nejprve bude v registru C vytvořen označovací bajt jehož povaha je založena na prvním písmenu názvu proměnné. Bity 5 a 6 tohoto bajtu indikují typ použité proměnné. Registr B je používán jako bitová vlajka (flag).

28B2	LOOK-VARS	SET 6,(IY+1) (FLAGS)	Předpokládej číselnou proměnnou.
		RST #18,GET-CHAR	Vyzvedni první znak do A.
		CALL #2C8D,ALPHA	Je alfanumerický ?
		JP NC,#1C8A,REPORT-C	Ohlaš chybu jestliže ne.
		PUSH HL	Ušchovej ukazatel na první znak.
		AND #1F	Převed bity 0-4 do registru C.
		LD C,A	Bity 5-7 jsou vždy nulové.
		RST #20,NEXT-CHAR	Vyzvedni další znak do A.
		PUSH HL	Ušchovej jeho ukazatel.
		CP #28	Je druhým znakem závorka "(" ?
		JR Z,#28EF,V-RUN/SYN	Odděl pole a čísla.
		SET 6,C	Nyní nastav bit 6.
		CP #24	Je druhým znakem "\$" ?
		JR Z,#28DE,V-STR-VAR	Odděl všechny řetězce.
		SET 5,C	Nyní nastav bit 5.
		CALL #2C88,ALPHANUM	Jestliže délka názvu proměnné je pouze jeden znak, skoč dopředu.
		JR NC,#28E3,V-TEST-FN	

Nyní bude nalezen koncový znak názvů majících více než jeden znak.

28D4	V-CHAR	CALL #2C88,ALPHANUM	Je znak alfanumerický ?
		JR NC,#28EF,V-RUN/SYN	Vyskoč ze smyčky jestliže byl nalezen konec názvu.
		RES 6,C	Označ rozlišovací bajt.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		JR #28D4,V-CHAR	Jdi zpět a testuj jej.

Obyčejné řetězce a řetězcová pole vyžadují, aby bit 6 systémové proměnné FLAGS byl vynulován.

28DE	V-STR-VAR	RST #20,NEXT-CHAR	Posuň CH-ADD za značku "\$".
		RES 6,(IY+1) (FLAGS)	Nuluj bit 6 jako signál:řetězec.

Není-li DEFADD-HI nula, znamená to, že se bude zpracovávat fce ("FN"), je-li však program v běhu, budou se hledat argumenty příkazu DEF FN.

28E3	V-TEST-FN	LD A,(#5C0C) (DEFADD-HI)	Je DEFADD-HI nula?
		AND A	
		JR Z,#28EF,V-RUN/SYN	Jestliže ano, skoč dopředu.
		CALL #2530,SYNTAX-Z	Je program v běhu?
		JP NZ,#2951,STK-F-ARG	Jestliže ano, skoč dopředu k prohledání příkazu DEF FN.

Jinak (nebo když proměnná nebyla nalezena v příkazu DEF FN) se provede hledání v oblasti proměnných, pokud se ovšem nekontroluje syntaxe.

28EF	V-RUN/SYN	LD B,C	Kopíruj rozlišovací bajt do registru B.
------	-----------	--------	---

CALL #2530,SYNTAX-Z		
JR NZ,#28FD,V-RUN		Je-li program v běhu, skoč dopředu.
LD A,C		Převeď rozlišovací bajt do A.
AND #E0		Vymaž část znakového kódu.
SET 7,A		Indikuj syntaxi nastavením bitu 7.
LD C,A		Obnov značkový bajt
JR #2934,V-SYNTAX		a pokračuj.

Provádí se basicový řádek a proto se provede prohledání oblasti proměnných.

28FD V-RUN	LD HL,(#5V4B) (VARS)	Vyzvedni ukazatel VARS.
------------	----------------------	-------------------------

Nyní vstup do smyčky, která posoudí názvy existujících proměnných.

2900 V-EACH	LD A,(HL)	První písmeno každé existující proměnné
	AND #7F	porovnávej na bitech 0-6.
	JR Z,#2932,V-80-BYTE	Skoč při bajtu #80 (koncový znak oblasti proměnných).
	CP C	Vlastní porovnání.
	JR NZ,#292A,V-NEXT	Jestliže se první znaky neshodují, skoč dopředu.
	RLA	Rotuj A doleva a
	ADD A,A	pak je zdvoj, abys otestoval bity 5,6.
	JP P,#293F,V-FOUND-2	Řetězcové proměnné a pole.
	JR C,#293F,V-FOUND-2	Jednoduché číselné a "FOR NEXT" proměnné.

Dlouhé názvy vyžadují aby byly porovnány celé.

	POP DE	Poříd kopii ukazatele na druhý znak.
	PUSH DE	
	PUSH HL	Uchovej ukazatel na první znak.
2912 V-MATCHES	INC HL	Posuzuj další znak.
2913 V-SPACES	LD A,(DE)	Vyzvedni postupně všechny znaky
	INC DE	a ukazuj na další znak.
	CP #20	Je to mezera?
	JR Z,#2913,V-SPACES	Ignoruj mezery.
	OR #20	Nastav bit 5 tak,
	CP (HL)	abys testoval velká i malá písmena.
	JR Z,#2912,V-MATCHES	Jestliže se shodují, skoč zpět pro další znak.
	OR #80	Budou se shodovat při nastaveném bitu 7 ?
	CP (HL)	Zkus to.
	JR NZ,#2929,V-GET-PTR	Jestliže se poslední znaky neshodují, skoč dopředu.
	LD A,(DE)	Kontrola konce
	CALL #2C88,ALPHANUM	názvu před odskokem
	JR NC,#293E,V-FOUND-1	dopředu.

Ve všech případech neshody názvů se musí HL upravit tak, aby ukazoval na další proměnnou v oblasti proměnných.

2929 V-GET-PTR	POP HL	Vyzvedni ukazatel.
292A V-NEXT	PUSH BC	Krátce uschovej B & C.
	CALL #19B8,NEXT-ONE	Nyní DE ukazuje na další proměnnou.
	EX DE,HL	Zaměň oba ukazatele
	POP BC	vyzvedni zpět B&C a
	JR #2900,V-EACH	skoč znovu do smyčky.

Nebyla-li nalezena žádná položka se správným názvem, skoč sem.

2932 V-80-BYTE	SET 7,B	Signál: proměnná nenalezena.
----------------	---------	------------------------------

A při kontrole syntaxe skočíš sem.

2934	V-SYNTAX	POP DE	Znič ukazatel na druhý znak.
		RST #18,GET-CHAR	Vyzvedni aktuální znak.
		CP #28	Je to závorka "(" ?
		JR Z,#2943,V-PASS	Skoč dopředu.
		SET 5,B	Signál:nejedná se o pole.
		JR #294B,V-END	Skoč dopředu.

Sem přijdeš po nalezení položky s odpovídajícím jménem.

293E	V-FOUND-1	POP DE	Odhoď uschovaný ukazatel na proměnnou.
293F	V-FOUND-2	POP DE	Znič ukazatel na 2. znak proměnné.
		POP DE	Znič ukazatel na 1. znak proměnné.
		PUSH HL	Uschovej ukazatel na poslední znak názvu proměnné.
		RST #18,GET-CHAR	Vyzvedni aktuální znak.

Má-li jméno proměnné více než jeden znak, musí být převedeny i další znaky.

Poznámka: Zdá se však, že to už bylo uděláno v části V-CHAR.

2943	V-PASS	CALL #2C88,ALPHANUM	Je znak alfanumerický?
		JR NC,#294B,V-END	Byl-li nalezen konec názvu proměnné, skoč.
		RST #20,NEXT-CHAR	Vyzvedni další znak
		JR #2943,V-PASS	a jdi zpět na jeho test.

Nyní jsou nastaveny výstupní parametry.

294B	V-END	POP HL	HL ukazuje <b>na</b> poslední znak názvu.
		RL B	Rotuj celý registr,
		BIT 6,B	otestuj jeho bit 6.
		RET	Hotovo.

Výstupní parametry tohoto podprogramu mohou být shrnuty následovně: Systémová proměnná CH-ADD ukazuje na první místo za názvem proměnné v basicovém řádku.

Pokud proměnná nebyla nalezena:

- CY=1
- Z flag je nastaven **jenom** při hledání pole.
- Registrový pár HL ukazuje na první znak názvu proměnné tak, jak se vyskytuje v basicovém řádku.

Pokud proměnná byla nalezena:

- CY=0
- Z flag=1 jak pro jednoduché řetězcové proměnné, tak pro pole.
- Registrový pár HL ukazuje na písmeno krátkého jména, nebo na poslední znak u vícepísmenných názvů, existující položky v oblasti proměnných.

Ve všech případech bity 5 a 6 registru C indikují typ zpracovávané proměnné. Bit 7 je komplementem "SYNTAX/RUN flag". Bity 0-4 budou obsahovat kód znaku nalezené proměnné pouze tehdy, byl-li program v běhu.

Pokud byl podprogram volán při kontrole syntaxe, bude se vždy vracet s CY=0. Z flag bude nastaven na 1 pro pole, nebo na 0 pro všechny proměnné kromě řetězcových názvů nesprávně následovaných závorkou "(", což nastaví Z flag na jedna, ale v případě že se bude provádět SAVE "jméno" DATA a\$(), bude se syntaxe považovat za správnou.

## PODPROGRAM "ULOŽ ARGUMENTY FUNKCÍ NA ZÁSOBNÍK"

Podprogram je volán z LOOK-VARS když není DEFADD-hí nula, aby v oblasti argumentů byl nalezen příkaz DEF FN, před prohledáním **programové oblasti**. Byla-li proměnná nalezena v příkaze DEF FN, budou parametry řetězcové proměnné uloženy na zásobník a bude dán signál: nevolej STK/VAR. Ale to už zůstává na programu SCANNING, který uloží hodnoty číselné proměnné obvyklým způsobem (viz. adresa #26DA).

2951	STK-F-ARG	LD HL,(#5C0B) (DEFADD)	Ukazuj na 1.znak oblasti argumentů
		LD A,(HL)	a převed jej do A.
		CP #29	Je to závorka ")" ?
		JP Z,#28EF,V-RUN/SYN	Jestliže ano, skoč k prohledání oblasti proměnných.
295A	SFA-LOOP	LD A,(HL)	Vyzvedni další argument ve smyčce.
		OR #60	Nastav bity 5 a 6, jako signál: obvyčejná číselná proměnná.
		LD B,A	Okopíruj je do B.
		INC HL	Ukazuj na další kód.
		LD A,(HL)	Vyzvedni jej do A.
		CP #0E	Je to #0E (značka FP čísla) ?
		JR Z,#296B,SFA-CP-VR	Jestliže ano, skoč (je to číselná proměnná).
		DEC HL	Nastav HL aby ukazovalo na značku
		CALL #28AB, FN-SKPOVR	"\$" a ne na mezeru nebo řídící znak.
		INC HL	HL nyní ukazuje na značku FP čísla.
		RES 5,B	Nuluj bit 5 v B což je signál: řetězcová proměnná.
296B	SFA-CP-VR	LD A,B	Vyzvedni název proměnné do A.
		CP C	Je to ta, kterou hledáme ?
		JR Z,#2981,SFA-MATCH	Jestli ano, skoč.
		INC HL	Nyní překroč 5 bajtů
		INC HL	FP čísla nebo
		INC HL	parametru řetězce,
		INC HL	aby mohl být nalezen
		INC HL	další argument.
		CALL #28AB, FN-SKPOVR	Posuň se na další znak.
		CP #29	Je to závorka ")" ?
		JP Z,#28EF,V-RUN/SYN	Jestliže ano, skoč na prohledání oblasti proměnných.
		CALL #28AB, FN-SKPOVR	Ukazuj na další
		JR #295A,SFA-LOOP	argument a skoč zpět na jeho posouzení.

Byla nalezena shoda. Parametry řetězcové proměnné jdou nyní na zásobník, čímž se předejde potřebě volání programu STK-VAR.

2981	SFA-MATCH	BIT 5,C	Jedná se o číselnou
		JR NZ,#2991,SFA-END	proměnnou, skoč, a SCANNING už ji uloží na zásobník.
		INC HL	Ukazuj na 1. z pěti ukládaných bajtů.
		LD DE,(#5C65) (STKEND)	DE ukazuje na <b>STKEND</b> .
		CALL #33C0, MOVE-FP	Ulož těchto 5 bajtů.
		EX DE,HL	HL ukazuje na novou pozici STKEND, jejíž
		LD (#5C65),HL (STKEND)	hodnota je uložena do STKEND.
2991	SFA-END	POP DE	Odhoď ukazatele pro
		POP DE	LOOK-VARS (1. a 2. znakový ukazatel).
		XOR A	Vrať se s CY i Z flag nulovými, což
		INC A	je signál: nevolat STK-VAR.
		RET	Hotovo.

## PODPROGRAM STK-VAR

Tento podprogram se obvykle používá k nalezení parametrů, které definují existující řetězec v oblasti proměnných nebo vrací v HL základovou adresu určitého elementu nebo **polem**. Je-li volána z DIM slouží pouze ke kontrole syntaxe basicového příkazu. Parametry definující řetězec ale mohou být pozměněny voláním programu SLICING, je-li to potřeba.

<b>2996</b>	STK-VAR	XOR A	Nastav vlnku: pole.
		LD B,A	Vynuluj B.
		BIT 7,C	Skoč dopředu při
		JR NZ,#29E7,SV-COUNT	kontrole syntaxe.

Dále odděl jednoduché řetězce od polí.

		BIT 7,(HL)	Jedná-li se o pole,
		JR NZ,#29AE,SV-ARRAYS	skoč dopředu.
		INC A	Signál: jednoduchý řetězec.
<b>29A1</b>	SV-SIMPLE\$	INC HL	Posuň ukazatel.
		LD C,(HL)	Vyzvedni
		INC HL	délku
		LD B,(HL)	řetězce.
		INC HL	Posuň ukazatel.
		EX DE,HL	Převeď ukazatel na aktuální řetězec.
		CALL #2AB2,STK-STO-\$	Předej parametry na zásobník kalkulátoru.
		RST #1B,GET-CHAR	Vyzvedni aktuální
		JP #2A49,SV-SLICE?	znak a skoč na test "krácení".
<b>29AE</b>	SV-ARRAYS	INC HL	Posuň ukazatel
		INC HL	za délku
		INC HL	pole.
		LD B,(HL)	Vyzvedni počet dimenzí.
		BIT 6,C	Jedná se o číselné pole
		JR Z,#29C0,SV-PTR	skoč dopředu.

Jestliže řetězcové pole má počet dimenzí 1, lze jej považovat za obyčejný řetězec.

	DEC B	Zmenši počet
	JR Z,#29A1,SV-SIMPLE\$	dimenzí a skoč, byl-li počet 1.

Dále se provede kontrola indexu.

	EX DE,HL	Uchovej ukazatel v DE.
	RST #1B,GET-CHAR	Vyzvedni aktuální znak.
	CP #2B	Je to závorka "(" ?
	JR NZ,#2A20,REPORT-3	Skoč podat chybové hlášení, jestliže ne.
	EX DE,HL	Obnov ukazatel v HL.

Pro oba typy polí se nyní ohodnotí index.

29C0	SV-PTR	EX DE,HL	Dej ukazatel do DE.
		JR #29E7,SV-COUNT	Skoč dopředu.

Následující smyčka má za úkol nalézt parametry daného elementu pole. Do smyčky se vstupuje v bodě **SV-COUNT**. Smyčka se vykoná B krát, což u číselných polí odpovídá počtu použitých dimenzí. Pro řetězcová pole se smyčka vykoná B-1 krát, protože poslední index se použije ke stanovení "výřezu" z řetězce.

29C3	SV-COMMA	PUSH HL	Uchovej čítač.
		RST #1B,GET-CHAR	Vyzvedni aktuální znak.
		POP HL	Obnov čítač.



CP	#2C	Je aktuální znak čárka "," ?
JR	Z,#29EA,SV-LOOP	Skoč dopředu na posouzení dalšího indexu.
BIT	7,C	Při běhu programu
JR	Z,#2A20,REPORT-3	skoč ohlásit chybu.
BIT	6,C	Jedná-li se o
JR	NZ,#29DB,SV-CLOSE	řetězcové pole, skoč dopředu.
CP	#29	Je aktuální znak závorka ")" ?
JR	NZ,#2A12,SV-RPT-C	Skoč ohlásit chybu, jestliže ne.
RST	#20,NEXT-CHAR	Vyzvedni další znak.
RET		Syntaxe je v pořádku - vrať se.

Pro řetězcové pole může již tento index znamenat "řez", anebo je ještě dále v basicovém řádku.

29DB	SV-CLOSE	CP	#29	Je aktuální znak závorka ")"?
		JR	Z,#2A48,SV-DIM	Jestliže ano, skoč na kontrolu dalšího indexu.
		CP	#CC	Je aktuální znak TO ?
		JR	NZ,#2A12,SV-RPT-C	Skoč ohlásit chybu, jestliže ne.
29E0	SV-CH-ADD	RST	#18,GET-CHAR	Vyzvedni aktuální znak.
		DEC	HL	Ukazuj na předchozí
		LD	(#5C5D),HL (CH-ADD)	znak a nastav CH-ADD.
		JR	#2A45,SV-SLICE	Skoč na hodnocení "řezu".

Zde je vstup do smyčky.

29E7	SV-COUNT	LD	HL,#0000	Nastav čítač elementů na nulu.
29EA	SV-LOOP	PUSH	HL	Krátce jej uschovej.
		RST	#20,NEXT-CHAR	Vyzvedni další znak.
		POP	HL	Obnov čítač.
		LD	A,C	Vyzvedni rozlišovací bajt.
		CP	#C0	Pokud nekontroluješ
		JR	NZ,#29FB,SV-MULT	<b>syntaxi</b> , skoč vpřed.
		RST	#18,GET-CHAR	Vyzvedni aktuální znak.
		CP	#29	Je aktuální znak závorka ")" ?
		JR	Z,#2A48,SV-DIM	Čítání elementů skončeno - skoč dopředu.
		CP	#CC	Je aktuální znak TO ?
		JR	Z,#29E0,SV-CH-ADD	Skoč zpět - jedná se o "řez".
29FB	SV-MULT	PUSH	BC	Uschovej počet dimenzí a rozlišovací bajt.
		PUSH	HL	Uschovej čítač elementů.
		CALL	#2AEE,DE,(DE+1)	Vyzvedni velikost dimenze do DE.
		EX	(SP),HL	Čítač do HL a <b>ukazatel</b> na zásobník.
		EX	DE,HL	<b>Záměna</b> .
		CALL	#2ACC,INT-EXP1	Ohodnoť další index.
		JR	C,#2A20,REPORT-3	Je-li mimo rozsah, ohlaš chybu.
		DEC	BC	Dekrementuj výsledek ohodnocení, protože čítač má počítat elementy vyskytující se před specifikovaným elementem.
		CALL	#2AF4,GET-HL*DE	Vynásob čítač velikostí dimenze.
		ADD	HL,BC	Přičti výsledek z INT-EXP1 k aktuálnímu čítači.
		POP	DE	Vyzvedni ukazatel proměnné.
		POP	BC	Vyzvedni počet dimenzí a rozlišovací bajt.
		DJNZ	#29C3,SV-COMMA	Pokračuj v průchodech smyčkou, dokud B není nula.

Než se odliší číselná a řetězcová pole, testuje se **vlažka** SYNTAX/RUN.

2A12	SV-RPT-C	BIT	7,C	Jestliže kontroluješ
		JR	NZ,#2A7A,SL-RPT-C	syntaxi, skoč ohlásit chybu.
		PUSH	HL	Uschovej čítač.
		BIT	6,C	Jedná-li se o řetězcové

JR NZ,#2A2C,SV-ELEM\$ pole, skoč dopředu.

Při zpracování číselného pole musí být aktuální znak závorka ")".

	LD B,D	Převed ukazatel	
	LD C,E	proměnné do BC.	
	RST #18,GET-CHAR	Vyzvedni aktuální znak.	
	CP #29	Je aktuální znak závorka ")" ?	
	JR Z,#2A22,SV-NUMBER	Přeskoč chybové hlášení, jestliže jo.	
2A20	REPORT-3	RST #08,ERROR-1	Ohlaš:
	DEFB #02	3-Subscript out of range.	

Nyní může být vypočtena adresa skutečné FP formy.

2A22	SV-NUMBER	RST #20,NEXT-CHAR	Vyzvedni další znak.
		POP HL	Vyzvedni čítač.
		LD DE,#0005	Číselný element má délku 5 bajtů.
		CALL #2AF4,GET-HL*DE	Vypočti celkový počet bajtů před požadovaným elementem.
		ADD HL,BC	HL nyní ukazuje před požadovaný element.
		RET	Vrať se s touto adresou.

Při zpracování řetězcového pole je velikost elementu dána hodnotou "poslední dimenze". Jsou vypočteny příslušné parametry a předány na zásobník kalkulátoru.

2A2C	SV-ELEM\$	CALL #2AEE,DE,(DE+1)	Vyzvedni rozměr poslední dimenze.
		EX (SP),HL	Ukazatel proměnné jde na zásobník a čítač do HL.
		CALL #2AF4,GET-HL*DE	Vynásob čítač velikostí dimenze.
		POP BC	Vyzvedni ukazatel proměnné.
		ADD HL,BC	HL nyní ukazuje na jedno místo před element.
		INC HL	A nyní na začátek elementu.
		LD B,D	Převed velikost
		LD C,E	poslední dimenze do BC, což bude délka.
		EX DE,HL	Začátek do DE.
		CALL #2AB1,STK-ST-0	Parametry na zásobník kalkulátoru.

Poznámka: První parametr je nula, což indikuje element pole a tedy signál : nemazat existující element.

Zápis indexu je možný třemi způsoby: A\$(2,4 TO 8) nebo A\$(2)(4 TO 8) anebo pokud se požaduje celý řetězec A\$(2).

	RST #18,GET-CHAR	Vyzvedni aktuální znak.	
	CP #29	Je to závorka ")" ?	
	JR Z,#2A48,SV-DIM	Skoč, jestliže ano.	
	CP #2C	Je to čárka "," ?	
	JR NZ,#2A20,REPORT-3	Skoč na ohlášení chyby, jestliže ne.	
2A45	SV-SLICE	CALL #2A52,SLICING	Použij SLICING pro úpravu souboru parametrů.
2A48	SV-DIM	RST #20,NEXT-CHAR	Vyzvedni další znak.
2A49	SV-SLICE?	CP #2B	Je to závorka "(" ?
		JR Z,#2A45,SV-SLICE	Skoč zpět, je-li třeba posoudit "řez".

Po posouzení posledního indexu se provede návrat, přičemž parametry požadovaného řetězce jsou uloženy na zásobníku kalkulátoru.

	RES 6,(IY+1) (FLAGS)	Signál: řetězcový výsledek.
	RET	Návrat.

## PODPROGRAM "SLICING"

Tento podprogram provede "Řez" z aktuálního řetězce. Na vstupu jsou parametry řetězce uloženy na zásobníku kalkulátoru a v registrech A B C D E. Parametry jsou vyzvednuty pouze při exekuci řádku.

2A52	SLICING	CALL #2530,SYNTAX-Z	Testuj <b>vlažku</b> .
		CALL NZ,#2BF1,STK-FETCH	Při běhu programu vyzvedni parametry.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CP #29	Je to závorka ")" ?
		JR Z,#2AAD,SL-STORE	Skoč dopředu, jestli ano.
		PUSH DE	Začátek jde na zásobník.
		XOR A	Nuluj registr A a
		PUSH AF	uschovej.
		PUSH BC	uschovej délku.
		LD DE,#0001	Předpoklad, že "Řez" začne 1.znakem.
		RST #18,GET-CHAR	Vyzvedni 1.znak.
		POP HL	Délka do HL.



Nyní se ohodnotí první parametr pro řez.

		CP #CC	Je aktuální znak TO ?
		JR Z,#2A81,SL-SECOND	První parametr je tedy považován za 1 a skoč testovat 2.
		POP AF	Obnov A, které je #00.
		CALL #2ACD, <b>INT-EXP2</b>	1.parametr do BC, A bude #FF, byla-li chyba v rozsahu.
		PUSH AF	uschovej tuto hodnotu.
		LD D,B	Převed 1.parametr
		LD E,C	do DE.
		PUSH HL	uschovej délku.
		RST #18,GET-CHAR	Vyzvedni aktuální znak.
		POP HL	Obnov délku.
		CP #CC	Je aktuální znak TO ?
		JR Z,#2A81,SL-SECOND	Skoč na test 2.parametru.
		CP #29	Je to závorka ")" ?
2A7A	SL-RPT-C	JP NZ,#1C8A,REPORT-C	Skoč ohlásit chybu, jestliže ne.

V tomto bodě již byl identifikován řez o délce jednoho znaku. Například A\$(4).

LD H,D	Poslední znak řezu
LD L,E	je tedy také prvním znakem.
JR #2A94,SL-DEFINE	Skoč dopředu.

Zde se hodnotí druhý parametr.

2A81	SL-SECOND	PUSH HL	uschovej délku.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		POP HL	Obnov délku.
		CP #29	Je to závorka ")" ?
		JR Z,#2A94,SL-DEFINE	Skoč, jestliže není druhý parametr.
		POP AF	Jestliže 1.parametr byl v rozsahu, A je #00, jinak #FF.
		CALL #2ACD, <b>INT-EXP2</b>	BC obsahuje 2.parametr.
		PUSH AF	uschovej "chybový registr".
		RST #18,GET-CHAR	Vyzvedni aktuální znak.
		LD H,B	Předej výsledek
		LD L,C	z <b>INT-EXP2</b> do HL.
		CP #29	Je to závorka ")" ?
		JR NZ,#2A7A,SL-RPT-C	Jestliže ne, skoč ohlásit chybu.

Budou definovány "nové" parametry.

2A94	SL-DEFINE	POP AF	Vyzvedni "chybový" registr.
		EX (SP),HL	Druhý parametr jde na zásobník a současně start je v HL.
		ADD HL,DE	Přičti první parametr ke startu.
		DEC HL	Jdi zpět o jedno místo, aby hodnota v HL byla správná.
		EX (SP),HL	Nový start jde na zásobník a 2.parametr zpět do HL.
		AND A	Odečti první parametry od druhých k
		SBC HL,DE	nalezení délky řezu.
		LD BC,#0000	Inicializuj novou délku.
		JR C,#2AAB,SL-OVER	Negativní řez nezpůsobí chybu, ale nulový řetězec.
		INC HL	Umožni jedno místo pro uzavírací bajt.
		AND A	Nyní testuj chybový registr.
		JP M,#2A20,REPORT-3	Byl-li některý parametr mimo rozsah, skoč.
		LD B,H	Převed novou délku
		LD C,L	do BC.
2AA8	SL-OVER	POP DE	Vyzvedni nový start
		RES 6,(IY+1) (FLAGS)	a zajisti, že se stále signalizuje řetězec.
2AAD	SL-STORE	CALL #2530,SYNTAX-Z	V tomto místě se vrať, jestliže kontroluješ syntaxi,
		RET Z	jinak pokračuj do podprogramu STK-STORE.

#### PODPROGRAM "STK-STORE"

Tento podprogram předává hodnoty v registrech A B C D E na zásobník kalkulátoru. S každým zavoláním tohoto programu se tedy zásobník zvětší o 5 bajtů. Podprogram se normálně používá k předání řetězcových parametrů, ale je i využíván podprogramy STACK-BC a LOG (2^A) k ukládání "malých celých čísel" na zásobník kalkulátoru. Všimněte si, že při ukládání parametrů řetězce první hodnota (přichází v registru A) bude nulová, jestliže řetězec je součástí pole nebo je řezem nějakého delšího řetězce. Pro kompletní jednoduchý řetězec bude hodnota #01. **Tato vlnka** se používá v příkazu LET, kde #01 signalizuje, že stará kopie má být zničena.

2AB1	STK-ST-0	XOR A	Signál: řetězec jako část pole nebo jeho řez.
2AB2	STK-STO-\$	RES 6,(IY+1) (FLAGS)	Zajisti, že <b>vlnka</b> indikuje řetězcový výsledek.
2AB6	STK-STORE	PUSH BC	Ušchovej krátce BC.
		CALL #33A9,TEST-5-SP	Je místo 5 bajtů? Nevrať se pokud není toto místo.
		POP BC	Obnov BC.
		LD HL,(#5C65) (STKEND)	Vyzvedni adresu prvního místa nad aktuálním zásobníkem.
		LD (HL),A	Převed 1.bajt.
		INC HL	Postup.
		LD (HL),E	Převed 2. a 3.bajt, což je u řetězců start.
		INC HL	
		LD (HL),D	
		INC HL	Postup.
		LD (HL),C	Převed 4. a 5.bajt, což je u řetězců délka.
		INC HL	
		LD (HL),B	
		INC HL	Posuň ukazatel tak, aby ukazoval nad zásobník.
		LD (#5C65),HL (STKEND)	Ulož tuto adresu do STKEND a
		RET	vrať se.

#### PODPROGRAM "INT-EXP"

Tento podprogram vrací výsledek ohodnocení "dalšího výrazu", jako například hodnotu integer v registrovém páru BC. Také testuje výsledek proti limitujícím hodnotám předávaným v registrovém páru HL. CY se bude rovnat 1, jestliže došlo k chybě "out of range" (mimo rozsah). Registr A se používá jako "chybový registr". Obsahuje #00 nedošlo-li k chybě a #FF došlo-li k chybě.

2ACC	INT-EXP1	XOR A	Nuluj chybový registr.
2ACD	INT-EXP2	PUSH DE	Ušchovej registr DE,
		PUSH HL	HL a také

PUSH AF		chybový registr.
CALL #1C82,EXPT-1NUM		Hodnota dalšího výrazu je přenesena na zásobník <b>ka lk.</b>
POP AF		Obnov chybový registr.
CALL #2530,SYNTAX-Z		
JR Z,#2AEB,I-RESTORE		Při kontrole syntaxe skoč dopředu.
PUSH AF		Uschovej opět chybový registr.
CALL #1E99,FIND-INT2		Poslední hodnota je kompresována do BC.
POP DE		Chybový registr do D.
LD A,B		Další výraz, který dává nulu je vždy chybový
OR C		
SCF		
JR Z,#2AEB,I-CARRY		takže je-li tomu tak, skoč.
POP HL		Udělej kopii limitu. Toto bude <b>rozměr</b> , limit pro DIM
PUSH HL		nebo délka řetězce.
AND A		Nyní porovnej
SBC HL,BC		výsledek oproti tomuto limitu.

Stav CY a hodnota v registru D jsou nyní manipulovány tak, aby daly příslušnou hodnotu pro chybový registr.

2AEB I-CARRY	LD A,D	Vyzvedni hodnotu "staré chyby", vytvoř "novou" chybu.
	SBC A,#00	#00 = nedošlo k chybě. #FF nebo méně je mimo rozsah.
2AEB I-RESTORE	POP HL	Obnov registry před návratem.
	POP DE	
	RET	Vrať se: chybový registr je A.

#### PODPROGRAM DE, (DE+1)

Tento podprogram provádí pseudoinstrukci LD DE,(DE+1) a vrací HL jako ukazatel na místo adresované hodnotou (DE+2).

2AEE DE,(DE+1)	EX DE,HL	Použij HL pro konstrukci.
	INC HL	Ukazuj na "DE+1".
	LD E,(HL)	Ve skutečnosti LD E,(DE+1).
	INC HL	Ukazuj na "DE+2".
	LD D,(HL)	Ve skutečnosti LD D,(DE+2).
	RET	Hotovo.

#### PODPROGRAM "GET-HL\*DE"

Tento podprogram **volá** HL=HL\*DE, pokud se nekontroluje syntaxe, čímž se dosahuje požadované funkce. Přetečení ze **16-tého bitu** dává hlášení "out of memory". Není to sice úplně pravda, ale předpokládá se, že není dost paměti pro úkol, který je uvažován.

2AF4 GET-HL*DE	CALL #2530,SYNTAX-Z	
	RET Z	Při kontrole syntaxe se vrať okamžitě.
	CALL #30A9,HL=HL*DE	Proveď násobení.
	JP C,#1F15,REPORT-4	Hlášení "out of memory".
	RET	Hotovo.

## PODPROGRAM PŘÍKAZU LET

Toto je skutečný podprogram přiřídlování, používaný příkazy LET, READ a INPUT. Jestliže cílová proměnná je nově deklarovaná proměnná, potom systémová proměnná DEST bude ukazovat na první písmeno názvu proměnné, tak jak se vyskytuje v basicovém řádku. Bit 1 systémové proměnné FLAGX bude nastaven na jedničku. Jestliže však cílová proměnná již existuje, potom bit 1 ve FLAGX bude nulový a systémová proměnná DEST bude ukazovat na místo před pěti bajty starého čísla; a pro řetězec na první místo starého řetězce. Použití systémové proměnné DEST tímto způsobem se vztahuje na jednoduché proměnné a části polí. Bit 0 systémové proměnné FLAGX je roven jedné, jestliže cílová proměnná je kompletní jednoduchá řetězcová proměnná (což je signál: smaž starou kopii). Na začátku se vyzvedne aktuální hodnota v systémové proměnné DEST a testuje se bit 1 proměnné FLAGS.

```
2AFF LET      LD HL,(#5C4D) (DEST)      Vyzvedni aktuální adresu v DEST.
              BIT 1,(IY+55) (FLAGX)
              JR Z,#2B66,L-EXISTS      Obsluhuješ-li proměnnou, která již existuje, skoč.
```

Jde o nově deklarovanou proměnnou. Proto musí být nejdříve nalezena délka jejího názvu.

```
LD BC,#0005      Předpokládej, že se jedná o číselnou proměnnou-5 bajtů.
```

Zde je vstup do smyčky, která obsluhuje znaky dlouhého názvu. Jakékoliv mezery nebo barevné kódy v názvu jsou ignorovány.

```
2B0B L-EACH-CH INC BC      Přičti jedničku do čítače za znak názvu.
2B0C L-NO-SP   INC HL      Posuň se po názvu proměnné.
              LD A,(HL)     Vyzvedni aktuální kód.
              CP #20        Je-li to mezera,
              JR Z,#2B0C,L-NO-SP      skoč zpět. Takto jsou ignorovány mezery.
              JR NC,#2B1F,L-TEST-CH   Je-li kód v rozsahu #21 až #FF, skoč dopředu.
              CP #10
              JR C,#2B29,L-SPACES     Vezmi jako výsledný kód ty, které jsou v rozsahu
              #00 až #0F.
              CP #16
              JR NC,#2B29,L-SPACES     Také akceptuj rozsah #16 až #1F.
              INC HL      Překroč řídící kód za jakýmkoliv znakem pro INK až OVER.
              JR #2B0C,L-NO-SP      Skoč zpět, protože tyto kódy jsou považovány za mezery.
```

Odděl "číselné" a "řetězcové" názvy.

```
2B1F L-TEST-CH CALL #2C88,ALPHANUM   Je znak alfanumerický?
              JR C,#2B0B,L-EACH-CH   Je-li tomu tak, akceptuj ho jako kód dlouhého názvu.
              CP #24                 Je to znak "$"?
              JP Z,#2BC0,L-NEW$      Obsluhuješ-li nově deklarovaný obyčejný řetězec,
              skoč dopředu.
```

Nově deklarovaná **proměnná**, která byla výše zpracována, potřebuje nyní prostor o velikosti BC v oblasti proměnných pro uložení svého názvu a své hodnoty. Tento prostor se vytvoří, potom se do něj okopíruje název proměnné, přičemž znaky tohoto názvu jsou už označeny tak, jak je potřeba.

```
2B29 L-SPACES LD A,C      Okopíruj délku do A.
              LD HL,(#5C59) (E-LINE) Nastav HL tak, aby
              DEC HL      ukazovalo na #80-kový bajt na konci oblasti proměnných.
              CALL #1655,MAKE-ROOM   Nyní otevři oblast proměnných.
```

Je vytvořen prostor o velikosti "BC" před #80-kovým bajtem.

```
INC HL      Ukazuj na první
INC HL      nový bajt.
EX DE,HL    DE bude ukazovat na druhý nový bajt.
```

PUSH DE	Uschovej tento ukazatel.
LD HL,(#5C4D) (DEST)	Vyzvedni ukazatel na začátek jména.
DEC DE	DE ukazuje na první nový bajt.
SUB #06	B bude obsahovat
LD B,A	počet znaků, které byly nalezeny v dlouhém názvu navíc.
JR Z,#2B4F,L-SINGLE	Obsluhuješ-li proměnnou s krátkým názvem, skoč dopředu.

Znaky navíc dlouhého názvu jsou předány do oblasti proměnných.

2B3E L-CHAR	INC HL	Ukazuj na každý znak "navíc".
	LD A,(HL)	Vyzvedni jej.
	CP #21	
	JR C,#2B3E,L-CHAR	Akceptuj znaky od #21 do #FF a ignoruj od #00 do #20.
	OR #20	Nastav bit 5 pro malá písmena.
	INC DE	Převáděj znaky
	LD (DE),A	postupně od druhého nového bajtu dále.
	DJNZ #2B3E,L-CHAR	Pokračuj znovu do smyčky pro všechny znaky "navíc".

Poslední znak dlouhého názvu musí být ORován hodnotou #80.

OR #80	Označ znak tak jak
LD (DE),A	je třeba a přepiš poslední kód.

První znak názvu zpracovávané proměnné je nyní posouzen.

	LD A,#C0	Připrav se na označení "dlouhý název".
2B4F L-SINGLE	LD HL,(#5C4D) (DEST)	Vyzvedni ukazatel na znak.
	XOR (HL)	A obsahuje #80 pro krátký název a #C0 pro dlouhý název.
	OR #20	Nastav bit 5 pro malá písmena.
	POP HL	Nyní odhod ukazatel.

Podprogram L-FIRST je nyní volán pro vložení "písmene" na příslušné místo.

CALL #2BEA,L-FIRST	Vlož "písmeno" - HL ukazuje na nový #80-kový bajt.
--------------------	--

Poslední hodnota může být nyní převedena do oblasti proměnných. Všimněte si, že v tomto bodě HL vždy ukazuje na pozici za pěti bajty, které jsou přiděleny tomuto číslu. Instrukce RST #28 je použita k zavolání kalkulátoru a poslední hodnota je pak "vymazána", ale nepřepsána.

2B59 L-NUMERIC	PUSH HL	Uschovej cílový ukazatel.
	RST #28,FP-CALC	Použij kalkulátor.
	DEFB #02,výmaz	Takto je posunut STKEND zpět o pět bajtů.
	DEFB #38,konec výpočtu	
	POP HL	Obnov ukazatel.
	LD BC,#0005	Nechť číslo je dlouhé pět bajtů.
	AND A	
	SBC HL,BC	HL ukazuje na první z pěti lokací.
	JR #2BA6,L-ENTER	Skoč dopředu k provedení skutečného převedení.

Při zjištění, že proměnná již existuje, vstup do tohoto bodu. Nejprve otestuj bit 6 ve FLAGS, abys oddělil číselné proměnné od řetězcových nebo řetězcová pole.

2B66 L-EXISTS	BIT 6,(IY+1) (FLAGS)	Při zpracování jakéhokoliv druhu
	JR Z,#2B72,L-DELETE\$	řetězcové proměnné skoč dopředu.

Pro číselné proměnné "nové" číslo přepíše "staré" číslo. Takže HL musí ukazovat na místo za pět bajtů existující položky. Nyní ovšem HL ukazuje na místo před těmito pěti bajty.

LD	DE,#0006	Pět bajtů pro číslo + jeden pro znak FP čísla.
ADD	HL,DE	Nyní HL ukazuje na místo "za".
JR	#2B59,L-NUMERIC	Skoč <b>zpět</b> k provedení skutečného přenesení.

Parametry řetězcové proměnné jsou vyzvednuty a kompletní jednoduché řetězce jsou odděleny od krácených řetězců a polí.

2B72	L-DELETE\$	LD HL,(#5C4D) (DEST)	Vyzvedni start. Poznámka: tento řádek je nadbytečný.
		LD BC,(#5C72) (STRLEN)	Vyzvedni délku.
		BIT 0,(IY+55) (FLAGX)	Při zpracování kompletního obyčejného
		JR NZ,#2BAF,L-ADD\$	řetězce skoč dopředu. Starý řetězec bude potřeba vymazat pouze v tomto případě.

Při zpracování řezu existující řetězcové proměnné, nebo řezu nebo kompletní části řetězcové pole, prochází program dvěma fázemi. Nejprve se v pracovním prostoru musí vytvořit nový řetězec, který je podle potřeby zkrácen nebo prodloužen, a v druhé fázi je pak tento řetězec okopírován na své místo v oblasti proměnných. Ovšem, pokud má řetězec nulovou délku, neděje se nic.

LD	A,B	
OR	C	
RET	Z	Jedná-li se o nulový řetězec, vrať se.

Jinak vytvoř požadovaný počet míst v pracovním prostoru.

PUSH	HL	Uschovej start (DEST).
RST	#30,BC-SPACES	Vytvoř potřebné místo v pracovním prostoru.
PUSH	DE	Uschovej ukazatel na první místo.
PUSH	BC	Uschovej délku pro pozdější použití.
LD	D,H	DE ukazuje na
LD	E,L	poslední místo a
INC	HL	HL ukazuje o bajt dále.
LD	(HL),#20	Je vložen znak mezery a po jeho
LDDR		okopírování do všech nových pozic ukazuje HL na první nový znak.

Nyní jsou ze zásobníku kalkulátoru vyzvednuty parametry právě zpracovávaného řetězce.

PUSH	HL	Krátce uschovej ukazatel.
CALL	#2BF1,STK-FETCH	Vyzvedni nové parametry
POP	HL	a obnov ukazatel.

Poznámka: V tomto bodě již bylo v pracovním prostoru vytvořeno místo pro právě "přidělovanou proměnnou", například při příkazu LET a\$(4 TO 8)="abcdefg" bylo vytvořeno pět míst. Parametry vyzvednuté jako poslední hodnota reprezentují řetězec, který má být kopírován do nových míst s použitím Prokrustova pravidla krácení nebo prodlužování podle potřeby. Délka nového řetězce je porovnávána s délkou možného prostoru.

EX	(SP),HL	Délka nové oblasti do HL a ukazatel nové oblasti na zásobník.	
AND	A	Porovnej dvě délky	
SBC	HL,BC	a pasuje-li nový řetězec do prostoru	
ADD	HL,BC	(a tedy není-li potřeba žádného krácení),	
JR	NC,#2B9B,L-LENGTH	skoč dopředu.	
LD	B,H	Jinak uprav novou	
LD	C,L	délku neboť je příliš dlouhá.	
2B9B	L-LENGTH	EX (SP),HL	Délka nové oblasti na zásobník a ukazatel nové oblasti do HL.



Pokud nemá nový řetězec nulovou délku je kopírován do pracovního prostoru. Prokrustovo pravidlo se provádí automaticky, jestliže je nový řetězec kratší, než pro něj vytvořené místo, a je ho třeba "natáhnout".

EX	DE,HL	Start nového řetězce do HL a ukazatel nové oblasti do DE.
LD	A,B	
OR	C	
JR	Z,#2BA3,L-IN-W/S	Je-li nový řetězec nulový, skoč dopředu.
LDIR		Jinak přeasuň nový řetězec do pracovního prostoru.

Hodnoty, které byly uschovány na zásobníku, jsou nyní obnoveny.

2BA3	L-IN-W/S	POP	BC	Délka nové oblasti.
		POP	DE	Ukazatel nové oblasti.
		POP	HL	Start, ukazatel na proměnnou (původně v DEST), nyní je v L-ENTER použit k předání "nového" řetězce do oblasti proměnných.

#### PODPROGRAM "L-ENTER"

Tento krátký podprogram se používá buď k předání číselné hodnoty ze zásobníku kalkulátoru, nebo řetězce z pracovního prostoru, na své nové místo v oblasti proměnných. Podprogram je tedy používán pro všechny řetězce kromě nově deklarovaných a existujících kompletních řetězců.

2BA6	L-ENTER	EX	DE,HL	Zaměň ukazatele.
		LD	A,B	
		OR	C	Opět kontroluj nulovou délku.
		RET	Z	Vrať se při nulovém výsledku.
		PUSH	DE	Ušchověj cílový ukazatel.
		LDIR		Přenes číselnou hodnotu nebo řetězec.
		POP	HL	HL ukazuje na první bajt číselné hodnoty nebo řetězce.
		RET		Vrať se.

#### PODPROGRAM "LET" POKRAČUJE ZDE"

Při zpracování kompletních a existujících řetězců vstupuje nový řetězec "pod maskou" nově deklarovaného před tím, než je zničena jeho původní verze.

2BAF	L-ADD\$	DEC	HL	HL ukazuje na znak
		DEC	HL	písmene názvu proměnné.
		DEC	HL	Tzn. DEST-3.
		LD	A,(HL)	Vyzvedni znak.
		PUSH	HL	Ušchověj ukazatel na existující verzi.
		PUSH	BC	Ušchověj délku existujícího řetězce.
		CALL	#2BC6,L-STRING	Použij L-STRING k přičtení nového řetězce do oblasti proměnných.
		POP	BC	Obnov délku a
		POP	HL	ukazatel.
		INC	BC	Přidej jeden bajt pro písmeno a
		INC	BC	
		INC	BC	dva bajty pro délku.
		JP	#19E8,RECLAIM-2	Odejdí přes "zničení" celé původně existující verze.

Nově deklarované jednoduché řetězce jsou zpracovány následovně:

2BC0	L-NEW\$	LD	A,#DF	Připrav se na označení názvu proměnné.
		LD	HL,(#5C4D) (DEST)	Vyzvedni ukazatel na písmeno a
		AND	(HL)	patříčně jej poznač. Pak použij L-STRING (viz dále).

### PODPROGRAM "L-STRING"

Parametry nového řetězce jsou vyzvednuty. Pak se vytvoří potřebný prostor a řetězec je přenesen.

2BC6	L-STRING	PUSH AF	Uschovej znak názvu proměnné.
		CALL #2BF1,STK-FETCH	Vyzvedni start a délku nového řetězce.
		EX DE,HL	Převeď start do HL.
		ADD HL,BC	HL at ukazuje jedno místo za řetězec.
		PUSH BC	Uschovej délku.
		DEC HL	HL ukazuje na konec řetězce.
		LD (#5C4D),HL (DEST)	Krátce uschovej ukazatel.
		INC BC	Přidej jeden bajt pro písmeno a
		INC BC	
		INC BC	dva bajty pro délku.
		LD HL,(#5C59) (E-LINE)	HL ukazuje na #80-kový bajt
		DEC HL	konce proměnných.
		CALL #1655,MAKE-ROOM	Nyní vytvoř prostor v oblasti proměnných.

Poznámka: Ve skutečnosti je #80-kový bajt posunut (a s ním všechny důležité ukazatele) tak, že vznikne BC míst.

LD HL,(#5C4D) (DEST)	Obnov ukazatel na konec nového řetězce.
POP BC	Proveď kopii délky nového řetězce,
PUSH BC	
INC BC	přidej jednu na délku, i v případě délky řetězce 0.
LDDR	Pak okopíruj nový řetězec plus jeden bajt.
EX DE,HL	HL nechť ukazuje na vyšší bajt délky.
INC HL	
POP BC	Vyzvedni délku.
LD (HL),B	Vlož její vyšší a
DEC HL	
LD (HL),C	pak nižší bajt.
POP AF	Vyzvedni znak názvu proměnné.

### PODPROGRAM "L-FIRST"

Do tohoto podprogramu se vstupuje s již patřičně označeným písmenem názvu proměnné v registru A. Tento znak přepíše starý #80-kový bajt v oblasti proměnných. Při výstupu z tohoto podprogramu je registrový pár HL nastaven na nový #80-kový bajt.

2BEA	L-FIRST	DEC HL	Nechť HL ukazuje na starý #80-kový bajt.
		LD (HL),A	Ulož tam písmeno názvu proměnné.
		LD HL,(#5C59) (E-LINE)	Nechť HL ukazuje
		DEC HL	na nový #80-kový bajt.
		RET	Skončeno s nově deklarovanou proměnnou.

### PODPROGRAM "STK-FETCH"

Tento důležitý podprogram vyzvedne poslední hodnotu ze zásobníku kalkulátoru. Těchto pět bajtů může vyjadřovat buď FP formu, "krátkou" nebo "dlouhou" formu, nebo to mohou být parametry řetězce.

2BF1	STK-FETCH	LD HL,(#5C65) (STKEND)	Vyzvedni STKEND.
		DEC HL	Zpět o jedno místo.
		LD B,(HL)	5. hodnota.
		DEC HL	Zpět o jedno místo.
		LD C,(HL)	4. hodnota.
		DEC HL	Zpět o jedno místo.

LD D,(HL)	3. hodnota.
DEC HL	Zpět o jedno místo.
LD E,(HL)	2. hodnota.
DEC HL	Zpět o jedno místo.
LD A,(HL)	1. hodnota.
LD (#5C65),HL (STKEND)	Obnov hodnotu STKEND na novou pozici.
RET	Hotovo.

## PODPROGRAM PŘÍKAZU DIM

Tento podprogram vytváří nová pole v oblasti proměnných. Podprogram začíná vyhledáním existujících proměnných v oblasti proměnných, aby se zjistilo zda pole s tímto názvem již existuje, a případně bylo toto pole zničeno před vytvořením pole nového. Nové číselné pole bude mít všechny prvky nastaveny na nulu a nové řetězové pole bude vyplněno mezerami.

2C02 DIM	CALL #28B2,LOOK-VARS	Prohledej oblast proměnných.
2C05 <b>D-REPORT-C</b>	JP NZ,#1C8A,REPORT-C	Ohlaš chybu C, pokud se nejedná o pole.
	CALL #2530,SYNTAX-Z	
	JR NZ,#2C15,D-RUN	Při běhu programu skoč dopředu.
	RES 6,C	Testuj syntaxi <b>řetězových polí jako by byli číselné</b> .
	CALL #2996,STK-VAR	Kontroluj syntaxi výrazů v závorkách a posuzuj
	CALL #1BEE,CHECK-END	další výrazy, <b>když je syntaxe</b> v pořádku.

Existující pole je zničeno.

2C15 D-RUN	JR C,#2C1F,D-LETTER	Neexistuje-li žádné pole skoč dopředu.
	PUSH BC	Uchovej rozlišovací bajt.
	CALL #19B8,NEXT-ONE	Nalezni začátek další proměnné a
	CALL #19EB,RECLAIM-2	znič existující pole.
	POP BC	Obnov rozlišovací bajt.

Jsou nalezeny počáteční parametry nového pole.

2C1F D-LETTER	SET 7,C	Nastav bit 7 rozlišovacího bajtu.
	LD B,#00	Čítač rozměrů na nulu.
	PUSH BC	Uchovej čítač a rozlišovací bajt.
	LD HL,#0001	Registr HL bude obsahovat délku elementu v poli.
	BIT 6,C	(1 pro řetězec a
	JR NZ,#2C2D,D-SIZE	
	LD L,#05	5 pro číslo).
2C2D D-SIZE	EX DE,HL	Délka elementu do DE.

Do následující smyčky se vstoupí pro každý rozměr, který je specifikován v závorkách po příkazu DIM. Celkový počet **bajtů pro elementy** pole bude vytvořen v registrovém páru DE.

2C2E D-NO-LOOP	RST #20,NEXT-CHAR	Posuň CH-ADD při každém průchodu.
	LD H,#FF	Nastav limitní hodnotu.
	CALL #2ACC,INT-EXP1	Ohodnot parametr.
	JP C,#2A20,REPORT-3	Ohlaš chybu je-li parametr mimo rozsah.
	POP HL	Vyzvedni čítač dimenzí a rozlišovací bajt.
	PUSH BC	Uchovej parametr při každém průchodu smyčkou.
	INC H	Inkrementuj čítač rozměrů při každém průběhu.
	PUSH HL	Obnov čítač dimenzí a rozlišovací bajt.
	LD H,B	Parametry jsou přeneseny do registrového páru HL.
	LD L,C	
	CALL #2AF4,GET-HL*DE	Celkový počet bajtů se vytváří v registrovém páru HL.
	EX DE,HL	Potom je převeden do DE.
	RST #18,GET-CHAR	Vyzvedni aktuální znak a

CP #2C  
 JR Z,#2C2E,D-N0-LOOP skoč znovu do smyčky, je-li to další rozměr.

Poznámka: V tomto bodě obsahuje registrový pár DE počet bajtů požadovaných pro elementy nového pole. Velikost každého rozměru je uschována na zásobník.

Nyní zkontroluj, že skutečně existuje závorka ")" za danými výrazy.

CP #29 Je to ")"?  
 JR NZ,#2C05,D-REPORT-C Jestliže ne, skoč zpět.  
 RST #20,NEXT-CHAR Posuň CH-ADD za ní.

Nyní budou rozměrům přiděleny velikosti.

POP BC Vyzvedni čítač rozměrů a rozlišovací bajt.  
 LD A,C Pro pozdější použití předej rozlišovací bajt do reg. A.  
 LD L,B Přenes čítač do L.  
 LD H,#00 Vyčisti registr H.  
 INC HL  
 INC HL Zvětši čítač rozměrů o dvě a  
 ADD HL,HL vynásob ho dvěma. Vytvoř správnou délku pro proměnnou  
 ADD HL,DE přičtením celkového počtu bajtů.  
 JP C,#1F15,REPORT-4 Ohlaš "out of memory" je-li třeba.  
 PUSH DE Uchovej celkový počet bajtů pole,  
 PUSH BC čítač dimenzí a rozlišovací bajt.  
 PUSH HL Uchovej také počet všech bajtů nutných k vytvoření pole  
 LD B,H a zároveň ho přenes  
 LD C,L do BC.

Požadovaný prostor pro nové pole je vytvořen na konci oblasti proměnných.

LD HL,(#5C59) (E-LINE) HL ukazuje na  
 DEC HL #80-kový bajt.  
 CALL #1655,MAKE-ROOM Je vytvořen prostor a  
 INC HL HL ukazuje na jeho první místo.

Nyní jsou vloženy parametry.

LD (HL),A Řádně označené písmeno je vloženo první.  
 POP BC Je vyzvednuta celková délka a  
 DEC BC  
 DEC BC  
 DEC BC je snížena o tři.  
 INC HL Posuň HL.  
 LD (HL),C Vlož nižší a  
 INC HL  
 LD (HL),B vyšší bajt délky.  
 POP BC Vyzvedni čítač rozměrů.  
 LD A,B Převeď jej do registru A.  
 INC HL Posuň HL.  
 LD (HL),A Vlož počet rozměrů.

Elementy nového pole jsou nyní vyčištěny.

LD H,D  
 LD L,E HL ukazuje na poslední místo pole a  
 DEC DE DE o jedno místo před ním.  
 LD (HL),#00 Vlož nulu na poslední místo.

	BIT 6,C	Ale pokud se jedná o řetězcové pole
	JR Z,#2C7C,DIM-CLEAR	
	LD (HL),#20	budou místo nul vloženy "mezery".
2C7C DIM-CLEAR	POP BC	Vyzvedni celkový počet bajtů pole a
	LDDR	vyčistí pole plus jedno místo navíc.

Nyní jsou vloženy velikosti rozměrů.

2C7F DIM-SIZES	POP BC	Vyzvedni velikost rozměru.
	LD (HL),B	Vlož vyšší a
	DEC HL	
	LD (HL),C	nižší bajt.
	DEC HL	
	DEC A	Zmenši čítač dimenzí a
	JR NZ,#2C7F,DIM-SIZES	opakuji operaci, dokud nebyly posouzeny všechny <b>dimenze</b>
	RET	Jinak se vrať.

#### PODPROGRAM "ALFANUM"

Je-li v registru A uložen znak platné číslice nebo písmene, vrací tento podprogram CY flag nastaven na hodnotu jedna.

2C8B ALPHANUM	CALL #2D1B,NUMERIC	<b>Test na číslici.</b>
	CCF	CY bude jedna pro platnou číslici a
	RET C	<b>v tom případě se vrať.</b>

#### PODPROGRAM "ALFA"

Je-li v registru A uložen znak platného písmene vrací tento podprogram CY flag nastaven na hodnotu jedna.

2C8D ALPHA	CP #41	Testuj oproti #41, což je kód pro písmeno "A".
	CCF	Komplementuj CY a
	RET NC	vrať se nebyl-li to platný kód znaku.
	CP #5B	Testuj oproti #5B, což je o jednu více než kód písmene Z.
	RET C	Vrať se jestliže se jedná o <b>znak</b> .
	CP #61	Testuj oproti #61, což je kód pro písmeno "a".
	CCF	Komplementuj CY a
	RET NC	vrať se nebyl-li to platný kód znaku.
	CP #7B	Testuj oproti #7B, což je o jednu více než kód písmene z.
	RET	Hotovo.

#### PODPROGRAM "DECIMAL TO FLOATING POINT"

Součástí kontroly syntaxe je převádění dekadických čísel uložených v basicovém řádku na floating point formu. Tento podprogram načte dekadické číslo (číslíci za číslíci) a výsledek zpracování uloží na zásobník kalkulátoru. Nejdříve je však obsloužen případný výskyt příkazu BIN, který je následován **sérií jedniček a nul**.

2C9B DEC-TO-FP	CP #C4	Jedná se o znak BIN?
	JR NZ,#2CBB,NOT-BIN	Jestliže ne, skoč.
	LD DE,#0000	Nastav hodnotu výsledku na hodnotu 0.
2CA2 BIN-DIGIT	RST #20,NEXT-CHAR	Vyzvedni další znak.
	SUB #31	Odečti znakový kód pro jedničku.
	ADC A,#00	Nula nyní dává nulu <b>s CY=1</b> a <b>jednička dává nulu s CY=0</b> .
		Jakýkoliv jiný znak
	JR NZ,#2CB3,BIN-END	vyvolá skok na BIN-END, a zkontroluje se znovu syntaxe.
	EX DE,HL	Výsledek do HL.

	CCF	Komplementuj CY.	
	ADC HL,HL	Posuň výsledek do leva s načtením CY do bitu 0.	
	JP C,#31AD,REPORT-6	Ohlaš přetečení, je-li výsledek větší než 65535.	
	EX DE,HL	Vrať výsledek do DE.	
	JR #2CA2,BIN-DIGIT	Skoč zpět na další nulu nebo jedničku.	
2CB3	BIN-END	LD B,D	Okopíruj výsledek
	LD C,E	do BC	
	JP #2D2B,STACK-BC	pro uložení na zásobník.	

Pro ostatní čísla se nejdříve převede celá část a jestliže je další znak **desetinná tečka**, posoudí se i zlomková část.

2CB8	NOT-BIN	CP #2E	Je první znak "."?
		JR Z,#2CCB,DECIMAL	Jestliže ano, skoč dopředu.
		CALL #2D3B,INT-T0-FP	Jinak vytvoř poslední hodnotu jako typ integer.
		CP #2E	Je další znak "."?
		JR NZ,#2CEB,E-FORMAT	Jestliže ano, skoč dopředu, neboť to může být exponent.
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CALL #2D1B,NUMERIC	Jestliže to není číslice,
		JR C,#2CEB,E-FORMAT	skoč (např 1.E4 je povoleno).
		JR #2CD5,DEC-STO-1	Skoč dopředu na obsluhu číslic za desetinnou tečkou.
2CCB	DECIMAL	RST #20,NEXT-CHAR	Jestliže číslo začínalo desetinnou tečkou,
		CALL #2D1B,NUMERIC	podívej se zda je následující znak číslice.
2CCF	DEC-RPT-C	JP C,#1CBA,REPORT-C	Není-li tomu tak, ohlaš chybu.
		RST #28,FP-CALC	Použij kalkulátor a
		DEFB #A0,stk-nula	ulož nulu jako celou část těchto čísel.
		DEFB #38,konec výpočtu	
2CD5	DEC-STO-1	RST #28,FP-CALC	Použij opět kalkulátor.
		DEFB #A1,stk-jedna	Nalezni FP formu dekadického čísla jedna a
		DEFB #C0,st-mem-0	uschovej ji v paměťové oblasti číslo nula.
		DEFB #02,vymaz	
		DEFB #38,konec výpočtu	
2CDA	NXT-DGT-1	RST #18,GET-CHAR	Vyzvedni aktuální znak.
		CALL #2D22,STK-DIGIT	Je-li to číslice ulož ji na zásobník.
		JR C,#2CEB,E-FORMAT	Jinak skoč dopředu.
		RST #28,FP-CALC	Nyní použij kalkulátor.
		DEFB #E0,get-mem-0	Při každém průchodu smyčkou je číslo vyzvednuto z paměti
		DEFB #A4,stk-deset	
		DEFB #05,dělení	vyděleno deseti
		DEFB #C0,st-mem-0	a obnoveno v paměti (asi takto 0.1,0.01,0.001 atd.).
		DEFN #04,násobení	Aktuální číslice je násobena aktuálním číslem
		DEFB #0F,sčítání	a přičtena k poslední hodnotě.
		DEFB #38,konec výpočtu	
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		JR #2CDA,NXT-DGT-1	Skoč zpět (o jeden bajt více než je třeba) k posouzení.

Dále posuzuj jakoukoliv notaci E, např. formátu x E m nebo x e m, kde m je kladné nebo záporné celé číslo.

2CEB	E-FORMAT	CP #45	Je aktuálním znakem "E"?
		JR Z,#2CF2,SIGN-FLAG	Jestliže ano, pak skoč dopředu.
		CP #65	Je to "e"?
		RET NZ	Konec jestliže ne.
2CF2	SIGN-FLAG	LD B,#FF	Použij B jako <b>vlažku</b> pro znaménko (#FF je plus "+").
		RST #20,NEXT-CHAR	Vyzvedni další znak.
		CP #2B	Je-li to plus,
		JR Z,#2CFE,SIGN-DONE	skoč dopředu.
		CP #2D	Není-li to ani mínus,

	JR	NZ,#2CCF,ST-E-PART	skoč dopředu.
	INC	B	Změň znaménko.
2CFE	ST-E-PART	RST #20,NEXT-CHAR	Ukazuj na první číslici.
2CCF	ST-E-PART	CALL #2D1B,NUMERIC	Není-li to číslice
	JR	C,#2CCF,DEC-RPT-C	ohlaš chybu.
	PUSH	BC	Ušchovej <b>vlažku</b> v registru B.
	CALL	#2D3B,INT-T0-FP	Ulož na zásobník hodnotu ABS m, kde m je exponent.
	CALL	#2DD5,FP-T0-A	Převeď ABS m do A.
	POP	BC	Obnov znaménkovou vložku do B.
	JP	C,#31AD,REPORT-6	Ohlaš přetečení je-li ABS m>255
	AND	A	nebo skutečně větší než 127
	JP	M,#31AD,REPORT-6	(hodnoty větší než 39 budou vyloučeny později).
	INC	B	Testuj znaménkovou vložku v registru B.
	JR	Z,#2D18,E-FP-JUMP	Plus (tedy #FF) nastaví Z flag a pak se odskočí.
	NEG		Neguj m je-li znaménko minus.
2D18	E-FP-JUMP	JP #2D4F,E-T0-FP	Skoč k přidělení poslední hodnoty, výsledek $x*10^m$ .

#### PODPROGRAM "NUMERIC"

Představuje-li aktuální hodnota v registru A znak platné číslice vrací tento podprogram CY flag nastaven na **nulu**.

2D1B	NUMERIC	CP #30	Testuj oproti #30 což je kód pro nulu.
		RET C	Jestliže se nejedná o platný znakový kód, vrať se.
		CP #3A	Testuj oproti hornímu limitu.
		CCF	Komplementuj CY a
		RET	hotovo.

#### PODPROGRAM "STK-DIGIT"

Nepředstavuje-li hodnota v registru A platnou číslici, podprogram se prostě vrátí, jinak je znaku přiřazena jeho hodnota a uložena na zásobník kalkulátoru.

2D22	STK-DIGIT	CALL #2D1B,NUMERIC	Není-li znak číslo,
		RET C	vrať se.
		SUB #30	Nahraď kód aktuální číslicí.

#### PODPROGRAM "STACK-A"

Tento podprogram vytváří z absolutní binární hodnoty v registru A hodnotu ve tvaru floating point a ukládá ji na zásobník.

2D28	STACK-A	LD C,A	Převeď hodnotu do registru C a
		LD B,#00	vynuluj registr B.

#### PODPROGRAM "STACK-BC"

Tento podprogram vytváří z absolutní binární hodnoty v registru BC hodnotu ve tvaru floating point a ukládá ji na zásobník. Forma používaná v tomto a tím pádem i v předchozích dvou pod programech, je forma **malých celých čísel**. První a pátý bajt jsou nulové, třetí a čtvrtý bajt je nižší a vyšší bajt **16-ti bitové** čísla v komplementovaném tvaru (tedy  $-n = 65536 + n$ ) a druhý bajt je znaménkový, tedy #00 pro plus a #FF pro minus.

2D2B	STACK-BC	LD IY,#5C3A	Nastav IY na ERR-NR.
		XOR A	Vynuluj registr A a
		LD E,A	registr E, aby indikoval +.
		LD D,C	Kopíruj méně významný bajt do D a
		LD C,B	významnější bajt do C.
		LD B,A	Vyčisti registr B.

CALL #2AB6,STK-STORE	Nyní ulož číslo.
RST #28,FP-CALC	HL nechť ukazuje na STKEND -5.
DEFB #38,konec výpočtu	
AND A	Vynuluj CY flag.
RET	Hotovo.

#### PODPROGRAM "INTEGER TO FLOATING POINT"

Tento podprogram převede číslo z basicového řádku, tedy celou část dekadického čísla nebo čísla řádku, na zásobník kalkulátoru jako poslední hodnotu v FP tvaru. Opakované volání na CH-ADD+1 vyzvedne postupně všechny číslice celočíselné části. Výstup se provede v případě nalezení nenumerického znaku.

2D3B INT-T0-FP	PUSH AF	Uschovej první číslici v A.
	RST #28,FP-CALC	Použij kalkulátor.
	DEFB #A0,stk-nula	Poslední hodnota je nyní nula.
	DEFB #38,konec výpočtu	
	POP AF	Obnov původní číslici.

Nyní se vstupuje do smyčky. Představuje-li kód číslici, nalezne se její FP forma uloží se pod poslední položku. Ta je potom vynásobena deseti a přičtena k číslici, aby byla vytvořena nová poslední hodnota a ta pak přenesena opět na začátek smyčky.

2D40 NXT-DGT-2	CALL #2D22,STK-DIGIT	Představuje-li kód číslici, ulož její FP formu na <b>zás.</b>
	RET C	kalkulátoru. Jinak se vrať.
	RST #28,FP-CALC	Použij kalkulátor.
	DEFB #01,záměna	Číslice jde pod poslední hodnotu.
	DEFB #A4,stk-deset	Je uloženo číslo 10.
	DEFB #04,násobení	Poslední hodnota = poslední hodnota * 10.
	DEFB #0F,sčítání	Poslední hodnota = poslední hodnota + číslice.
	DEFB #38,konec výpočtu	
	CALL #0074,CH-ADD+1	Další znak jde do A.
	JR #2D40,NXT-DGT-2	Skok zpět do smyčky s tímto znakem.



## ARITMETICKÉ PODPROGRAMY

### E-FORMAT TO FLOATING POINT FORMAT

(Doplňek: #3C - "e-to-fp")

Tento podprogram převádí číslo zapsané ve tvaru xEm, kde m je kladné nebo záporné celé číslo, na zásobník kalkulátoru tak, že hodnota x je poslední hodnotou a hodnota m je v registru A. Metoda používaná k nalezení absolutní hodnoty m, řekněme čísla p, je dána dělením nebo násobením čísla x hodnotou  $10^p$  podle toho, je-li m kladné nebo záporné. Pro dosažení tohoto výsledku se postupuje následovně: p je posouváno doprava dokud není nulové a x je násobeno nebo děleno hodnotou  $10^{(2^n)}$  pro každý nastavený bit hodnoty p. Jelikož p nikdy nepřesáhne hodnotu větší než 39, budou bity 6 a 7 hodnoty p normálně nulové.

2D4F	E-TO-FP	RLCA RRCA JR NC,#2D55,E-SAVE CPL INC A	Testuj znaménko m rotacemi bitu 7 do CY a zpět aniž by došlo ke změně A. Je-li m pozitivní, skoč. Neguj m v registru A bez porušení CY.
2D55	E-SAVE	PUSH AF LD HL,#5C92 CALL #350B,FP-0/1  RST #28,FP-CALC DEFB #A4,stk-deset DEFB #38,konec výpočtu POP AF	Uschovej krátce registr A. Toto je MEMBOT. Znaménkový bajt je nyní uložen v 1. bajtu mem-0,#00= +, #01= -. Na zásobníku je hodnota x. x,10 x,10 Obnov m v registru A.
2D60	E-LOOP	SRL A JR NC,#2D71,E-TST-END PUSH AF RST #28,FP-CALC	Ve smyčce posunuj další bit m - nastaví CY a Z jak třeba. Je-li CY rovno nule skoč. Uschovej zbytek m a <b>v lajky</b> .

Na zásobníku je nyní  $x'$  (což je přechodná fáze násobení  $x*10^m$ ) a  $10^{(2^n)}$ , kde  $n=0,1,...,5$ .

		DEFB #C1,stk-mem-1 DEFB #E0,get-mem-0 DEFB #00,skoč-pravda DEFB #04,na E-DIVSN DEFB #04,násobení DEFB #33,skoč DEFB #02,na E-FETCH	$10^{(2^n)}$ do mem-1 $x',10^{(2^n)},1/0$ $x',10^{(2^n)}$ $x',10^{(2^n)}$ $x'*10^{(2^n)} = x'$ <b><math>x'</math></b> $x''$
2D6D	E-DIVSN	DEFB #05,dělení	$x'/10^{(2^n)} = x''$ ( $x*10^{(2^n)}$ či $x'/10^{(2^n)}$ )
2D6E	E-FETCH	DEFB #E1,get-mem-1 DEFB #38,konec výpočtu POP AF	$x'',10^{(2^n)}$ $x'',10^{(2^n)}$ Obnov zbytek m v registru A a podmínkové <b>v lajky</b> .
2D71	E-TST-END	JR Z,#2D7B,E-END PUSH AF RST #28,FP-CALC DEFB #31,zdvojení DEFB #04,násobení DEFB #38,konec výpočtu POP AF	Bylo-li m sníženo na nulu, skoč. Uschovej zbytek m v registru A. $x'',10^{(2^n)}$ $x'',10^{(2^n)},10^{(2^n)}$ $x'',10^{(2^{(n+1)})}$ $x'',10^{(2^{(n+1)})}$ Obnov zbytek m v registru A.
2D7B	E-END	JR #2D60,E-LOOP RST #28,FP-CALC DEFB #02,výmaz DEFB #38,konec výpočtu RET	Skoč zpět pro všechny bity hodnoty m. Použij kalkulátor k vymazání <b>výsledné</b> mocniny deseti. Poslední hodnota je tedy $x*10^m$ .

### PODPROGRAM "VYZVEDNUTÍ CELÉHO ČÍSLA"

Tento podprogram načte do DE hodnotu "malého" celého čísla z místa adresovaného registrovým párem HL. Tedy n je běžně prvním (nebo druhým) číslem na zásobníku kalkulátoru, ale HL může (po záměně s DE) dosáhnout i čísla, které už bylo ze zásobníku kalkulátoru "vymazáno". Podprogram sám o sobě nevymaže číslo ze zásobníku a vrací HL tak, že ukazuje na čtvrtý bajt čísla v jeho původní pozici.

Poznámka: "malé celé číslo" je každé číslo n, pro které platí  $-65535 < n < 65535$ .

2D7F	INT-FETCH	INC	HL	Ukazuj na znaménkový bajt čísla a
		LD	C,(HL)	vyzvedni ho do C.

Následující mechanismus provede dvojkový doplněk čísla jestliže je záporné (C obsahuje #FF). Jinak ho nechá být (C obsahuje #00).

INC	HL	Ukazuj na méně významný bajt a
LD	A,(HL)	vyzvedni ho do A.
XOR	C	Je-li číslo záporné, proved jedničkový doplněk.
SUB	C	Přičti 1 pro záporná čísla a nastav CY, pokud není nula.
LD	E,A	Méně významný bajt do E.
INC	HL	Ukazuj na významnější bajt
LD	A,(HL)	a vyzvedni jej do A.
ADC	A,C	Dokonči dvojkovou komplementaci v
XOR	C	případě negativního čísla. CY je vždy rovno nule.
LD	D,A	Významnější bajt do D a
RET		hotovo.

### PODPROGRAM "ULOŽENÍ CELÉHO ČÍSLA"

Tento podprogram ukládá hodnotu "malého" celého čísla na místo adresované registrovým párem HL a čtyři další místa. Tedy n nahradí první (nebo druhé) číslo na zásobníku kalkulátoru. HL se vrací s hodnotou adresy prvního bajtu čísla n na zásobníku kalkulátoru.

2DBC	P-INT-STO	LD	C,#00	Tento vstupní bod je použit pro ukládání celých čísel.
2DBE	INT-STORE	PUSH	HL	Ukazatel na první místo je uschován.
		LD	(HL),#00	První bajt je nastaven na nulu.
		INC	HL	Ukazuj na druhé místo a
		LD	(HL),C	vlož znaménkový bajt.

Nyní je použit stejný způsob vytváření dvojkových doplňků pro negativní čísla jako v podprogramu "ULOŽENÍ CELÉHO ČÍSLA". Toto je zapotřebí například před nebo po násobení malých celých čísel. Ovšem sčítání se provádí bez jakýchkoliv úprav.

INC	HL	Ukazuj na třetí místo.
LD	A,E	Vyzvedni nižší bajt,
XOR	C	proved jeho dvojkový doplněk, je-li to třeba.
SUB	C	
LD	(HL),A	Ulož tento bajt.
INC	HL	Ukazuj na čtvrté místo,
LD	A,D	vyzvedni vyšší bajt,
ADC	A,C	opět proved odpovídající úpravy.
XOR	C	
LD	(HL),A	Ulož i tento bajt.
INC	HL	Ukazuj na páté místo a
LD	(HL),#00	vlož tam nulu.
POP	HL	Vrať se s adresou prvního bajtů ukládaného čísla v HL.
RET		

## PODPROGRAM "FP DO BC"

Tento podprogram se volá ze čtyř různých míst pro různé účely a je používán ke kompresi FP hodnoty na zásobníku kalkulátoru do registrového páru BC. Je-li výsledek větší než 65535, potom program nastaví CY na hodnotu jedna. Je-li poslední hodnota negativní, potom je Z flag roven nule. Nižší bajt výsledku je také okopírováno do registru A.

2DA2	FP-T0-BC	RST #28,FP-CALC	Použij kalkulátor, aby HL ukazovalo na STKEND-5.
		DEFB #38,konec výpočtu	
		LD A,(HL)	Vyzvedni exponentový bajt.
		AND A	Je-li nulový,
		JR Z,#2DAD,FP-DELETE	skoč, neboť se jedná o malé číslo.
		RST #28,FP-CALC	Použij kalkulátor
		DEFB #A2,stk-polovina	k zaokrouhlení poslední hodnoty na nejbližší
		DEFB #0F,sčítání	
		DEFB #27,int	celé číslo, což ho převede na malé celé číslo,
		DEFB #38,konec výpočtu	je-li to možné.
2DAD	FP-DELETE	RST #28,FP-CALC	Použij kalkulátor
		DEFB #02,delete	k vymazání čísla ze zásobníku.
		DEFB #38,konec výpočtu	DE stále ukazuje na číslo v paměti (na STKEND).
		PUSH HL	Ušchovej oba zásobníkové ukazatele.
		PUSH DE	
		EX DE,HL	HL nyní ukazuje na číslo.
		LD B,(HL)	Kopíruj první bajt do B.
		CALL #2D7F,INT-FETCH	Kopíruj bajty 2,3 a 4 do C, E a D.
		XOR A	Vynuluj registr A.
		SUB B	CY je nula pokud je B různé od nuly.
		BIT 7,C	Z flag je nastaven, jestliže číslo je kladné.
		LD B,D	Vyšší bajt do B.
		LD C,E	Nižší bajt do C a
		LD A,E	do A.
		POP DE	Obnov ukazatele.
		POP HL	
		RET	Hotovo.

## PODPROGRAM "LOG(2\*A)"

Tento podprogram se volá z podprogramu PRINT-FP aby vypočítal přibližný počet číslic před desetinnou tečkou pro x (číslo, které má být vytištěno). Nejsou-li před desetinnou tečkou žádné číslice, spočte přibližný počet úvodních nul za desetinnou tečkou. Do podprogramu se vstupuje s číslem e<sup>x</sup> v registru A, což je pravý exponent x, nebo e<sup>x-2</sup> a počítá se z LOG při základu 10 z hodnoty (2\*A). A se pak nastaví na hodnotu ABS INT (z+0.5) jak je požadováno. Pro tento účel se použije FP-T0-A.

2DC1	LOG(2*A)	LD D,A	Hodnota A je uložena ve tvaru: 00 00 A 00 00 pro A>0
		RLA	nebo 00 FF A FF 00 pro A<0.
		SBC A,A	
		LD E,A	Tyto bajty jsou nejprve uloženy do A E D C B a později
		LD C,A	
		XOR A	
		LD B,A	
		CALL #2AB6,STK-STORE	na zásobník kalkulátoru.
		RST #28,FP-CALC	Použij kalkulátor.
		DEFB #34,stk-data	LOG 2 je nyní uložen na zásobník.
		DEFB #EF,exponent #7F	
		DEFB #1A,#20,#9A,#85	Zásobník obsahuje A a LOG 2
		DEFB #04,násoben	A*LOG 2, čili LOG (2*A)
		DEFB #27,int	INT LOG(2*A)



**PODPROGRAM "FLOATING POINT TO A"**

Tento krátký ale velice důležitý podprogram je volán v osmi případech pro různé účely. Používá mimo jiné, FP-T0-BC k převedení poslední hodnoty do registru A kdykoliv je to možné. Proto také testuje **modulus** čísla a jestliže je toto číslo větší než 255, je nastaveno CY. Jinak se vrací modul čísla zaokrouhlený na nejbližší **celé číslo** v registru A. **Z** flag se používá k rozlišení kladného nebo záporného čísla.

2DD5	FP-T0-A	CALL #2DA2,FP-T0-BC	Kompresuj poslední hodnotu do BC.
		RET C	Bylo-li to mimo rozsah, vrať se.
		PUSH AF	Ušchvej výsledek <b>a podmínkové příznaky</b> a
		DEC B	opět je chyba,
		INC B	není-li v registru B nula.
		JR Z,#2DE1,FP-A-END	Je-li vše O.K., skoč.
		POP AF	Vyzvedni výsledek <b>a podmínkové příznaky</b> .
		SCF	Signál, že výsledek je mimo rozsah.
		RET	Konec-neúspěšně.
2DE1	FP-A-END	POP AF	Vyzvedni výsledek <b>a podmínkové příznaky</b> .
		RET	Konec-úspěšně.

**PODPROGRAM "VYPSÁNÍ FP ČÍSLA"**

Tento podprogram se volá podprogramem příkazu PRINT na adrese #2039 a STR\$ na adrese #3630, které převádí číslo na řetězec tak, jako kdyby byl vypsán. Tento podprogram vytiskne x, které je uloženo jako poslední hodnota na zásobníku kalkulátoru. Tiskový formát nikdy nezabere více než čtrnáct míst. Osm nejvýznamnějších číslic x správně zaokrouhlených je uschováno v tiskovém bafru vytvořeném speciálně pro tento účel v pamětech zásobníkové paměti mem-3 a mem-4. Malá čísla numericky menší než jedna a čísla větší než  $2^{27}$  jsou zpracována samostatně. Menší jsou násobena hodnotou  $10^n$ , kde n je přibližný počet úvodních nul za desetinnou čárkou, zatímco větší čísla jsou dělena hodnotou  $10^{(n-7)}$ , kde n je přibližný počet číslic před desetinnou čárkou. Tímto se všechna čísla dostávají do středního rozsahu a počet číslic požadovaných před desetinnou čárkou je uložen v druhém bajtu paměti mem-5. Nakonec je vše vytisknuto s použitím E-formátu, jestliže bylo více než osm číslic před desetinnou tečkou nebo **více než čtyři nuly za desetinnou tečkou**.

Následující program v basicu ukazuje na rozsah tiskových formátů:

```
10 FOR a=1 TO 12: PRINT SGN a*9^a: NEXT a
```

a) Nejprve je ošetřeno znaménko čísla x:

x<0 - podprogram skočí na **PF-NEGTV**, kde se před hodnotu ABS x vytiskne znaménko mínus.

x=0 - x je vymazáno ze zásobníku kalkulátoru, je vytištěna nula a převeden návrat.

x>0 - podprogram **tenom** pokračuje.

2DE3	PRINT-FP	RST #28,FP-CALC	Použij kalkulátor.
		DEFB #31,zdvojení	x,x
		DEFB #36, <0	x, (1/0) (logická hodnota x)
		DEFB #00,skok-pravda	x
		DEFB #0B,na PF-NEGTV	x
		DEFB #31,zdvojení	x,x
		DEFB #37, >0	x, (1/0) (logická hodnota x)
		DEFB #00,skok-pravda	x
		DEFB #0D,na PF-POSTVE	x kde již x' bude ABS x
		DEFB #02,výmaz	-
		DEFB #38,konec výpočtu	-
		LD A,#30	Vlož znak nula "0" a
		RST #10,PRINT-A-1	vytiskni ho.
		RET	Konec a poslední hodnota je nula.

2DF2	PF-NEGTV	DEFB #2A,abs DEFB #38,konec výpočtu LD A,#2D RST #10,PRINT-A-1 RST #28,FP-CALC	x'=ABS x x' Vlož znak "-" a vytiskni ho. Opět použij kalkulátor.
2DF8	PF-POSTVE	DEFB #A0,stk-nula DEFB #C3,st-mem-3 DEFB #C4,st-mem-4 DEFB #C5,st-mem-5 DEFB #02,výmaz DEFB #38,konec výpočtu EXX PUSH HL EXX	15 bajtů paměti mem-3 až 5 je nastaveno na nuly potřebné pro tiskový bafr a dvě počítadla. Zásobník je vyčištěn až na x'. x' HL', který bude obsahovat kalkulátorové doplňky (např. pro STR\$), je uložen na zásobník procesoru (SP).

b) Toto je start smyčky, která zpracovává větší čísla. Nicméně každé číslo x je rozděleno na svou celou část (i) a zlomkovou část (f). Jedná-li se o malé číslo, tedy -65535 až 65535, je uloženo v registrovém páru DE' před vložením do tiskového bafru.

2E01	PF-LOOP	RST #28,FP-CALC DEFB #31,zdvojení DEFB #27,int DEFB #C2,st-mem-2 DEFB #03,odečet DEFB #E2,get-mem-2 DEFB #01,záměna DEFB #C2,st-mem-2 DEFB #02,výmaz DEFB #38,konec výpočtu LD A,(HL) AND A JR NZ,#2E56,PF-LARGE CALL #2D7F,INT-FETCH LD B,#10 LD A,D AND A JR NZ,#2E1E,PF-SAVE OR E JR Z,#2E24,PF-SMALL LD D,E LD B,#08	Opět použij kalkulátor. x',x' x',INT(x')=i x',i do mem-2 x'-i+f f,i i,f i,f do mem-2 i i Jestliže i je malé číslo (1.bajt) je nulový a tedy ABS i<=65535, pak neskákej a okopíruj i do DE (i jako x'>=0). B je nastaveno jako počítadlo 16-ti bitů. D je okopírováno do A. Není-li D nulové, skoč. Nyní testuj E a jestliže DE je nula, skoč, neboť x je čistý zlomek. Přenes E do D. Nastav B na osm bitů.
2E1E	PF-SAVE	PUSH DE EXX POP DE EXX JR #2E7B,PF-BITS	Přenes za pomoci zásobníku DE do DE', aby jeho hodnoty mohly být přeneseny do tiskového bafru v bodě PF-BITS. Skoč dopředu.

c) Čisté zlomky jsou násobeny hodnotou  $10^n$ , kde n je přibližný počet úvodních nul za desetinnou čárkou, -n je přičteno k druhému bajtu mem-5, který obsahuje počet číslic před desetinnou čárkou. Záporná čísla indikují počet nul za desetinnou čárkou.

2E24	PF-SMALL	RST #28,FP-CALC DEFB #E2,get-mem-2 DEFB #38,konec výpočtu	i (i je zde nula) i,f i,f
------	----------	---	---------------------------------

Poznámka: Povšimněte si, že zásobník je nyní nevyvážen. K tomu by bylo potřeba další bajt DEFB #02, výmaz na adrese #2E25 hned za instrukcí RST #28. Proto nyní výraz jako např. "2"+STR\$ 0.5 je nesprávně ohodnocen jako 0.5: nula ponechaná na zásobníku nahradí dvojku a vše je považováno za nulový

řetězec. Tím pádem i řetězcová porovnání mohou vést k nesprávným hodnotám, když druhý řetězec má formu STR\$ x a x<1, tedy např. výraz "50"<STR\$ 0.1 dává logickou hodnotu pravda, neboť místo "50" je opět použit nulový řetězec.

LD A,(HL)	Exponentový bajt E čísla f je kopírován do A.
SUB #7E	A obsahuje E-126 tedy E'+2 kde E' je skutečný exponent f.
CALL #2DC1,LOG(2^A)	Konstrukce A=ABS INT(LOG(2^A)) je nyní vytvořena.
LD D,A	A=n takže: n okopírováno z A do D.
LD A,(#5CAC) (mem-5-2nd)	Aktuální počet je vyzvednut z druhého bajtu mem-5 a
SUB D	
LD (#5CAC),A (mem-5-2nd)	je od něj odečteno n.
LD A,D	n je okopírováno z D do A.
CALL #2D4F,E-TO-FP	Na zásobník je uložena hodnota y=f*10^n.
RST #28,FP-CALC	i,y
DEFB #31,zdvojení	i,y,y
DEFB #27,int	i,y,INT y-i2
DEFB #C1,st-mem-1	i2 do mem-1
DEFB #03,odečet	i,y-i2
DEFB #E1,get-mem-1	i,y-i2,i2
DEFB #38,konec výpočtu	i,f2,i2 (f2*y-i2)
CALL #2DD5,FP-TO-A	i2 je převedeno ze zásobníku do A.
PUSH HL	Ukazatel na f2 je uschován.
LD (#5CA1),A (mem-3-1st)	i2 je uschováno v prvním bajtu mem-3. Tedy pro tisk.
DEC A	Ovšem je-li i2 nula, nebude se počítat jako číslice pro
RLA	tisk. A je manipulováno tak, že nula způsobí nulu,
SBC A,A	ale nenulová hodnota způsobí jedničku.
INC A	
LD HL,#5CAB	Toto je 1.bajt mem-5, kam je vložena hodnota
LD (HL),A	nula nebo jedna (tedy počet číslic pro tisk)
INC HL	a přičtena k druhému bajtu mem-5,
ADD A,(HL)	což je počet číslic před desetinnou čárkou.
LD (HL),A	
POP HL	Ukazatel na f2 je obnoven.
JP #2ECF,PF-FRACTN	Skok na umístění f2 do bafru (HL ukazuje na f2,DE na i2).

d) Čísla větší než  $2^{*27}$  jsou násobena hodnotou  $2^{*(-n+7)}$  což snižuje počet číslic před desetinnou čárkou na osm a je proveden opětovný vstup do smyčky v bodě PF-LOOP.

2E56 PF-LARGE	SUB #80	E-#80=E' je skutečný exponent i.
	CP #1C	je-li E' menší než 28
	JR C,#2E6F,PF-MEDIUM	skoč.
	CALL #2DC1,LOG(2^A)	V registru A je vytvořena hodnota n a
	SUB #07	snížena na n-7.
	LD B,A	Pak je okopírována do B a
	LD HL,#5CAC	
	ADD A,(HL)	přičtena k druhému bajtu mem-5, což je počet číslic
	LD (HL),A	požadovaný před desetinnou čárkou pro tisk čísla x.
	LD A,B	Potom je i vynásobeno hodnotou $10^{*(-n+7)}$ a tím
	NEG	převedeno pro střední rozsah.
	CALL #2D4F,E-TO-FP	
	JR #2E01,PF-LOOP	Skok zpět do smyčky pro zpracování středně velkého čís.

e) Celá část hodnoty x je uložena v tiskovém bafru, který je tvořen 10 bajty mem-3 a mem-4.

2E6F PF-MEDIUM	EX DE,HL	DE nyní ukazuje na i, HL na f.
	CALL #2FBA,FETCH-TWO	Mantisa i je nyní DE', DE.
	EXX	Vyzvedni zrcadlové registry.
	SET 7,D	Pravdivý numerický bit 7 do D'.

LD	A,L	Exponentový bajt E čísla i do registru A.
EXX		Zpět na hlavní registry.
SUB	#80	Skutečný exponent do registru A.
LD	B,A	Tím se dosáhne požadovaného počtu bitů.

V případě, že i je malé celé číslo (menší než 65535), provede se opětovný vstup v tomto bodě.

2E7B	PF-BITS	SLA E	Mantisa i je nyní rotována doleva a
		RL D	všechny bity posouvány do mem-4 a
		EXX	každý bajt mem-4 je dekadicky upraven
		RL E	po každém posunu.
		RL D	Všechny čtyři bajty čísla i.
		EXX	Zpět na hlavní registry.
		LD HL,#5CAA	Adresa 5.bajtu mem-4 do HL.
		LD C,#05	Počet bajtů do C.
2E8A	PF-BYTES	LD A,(HL)	Vyzvedni bajt z mem-4.
		ADC A,A	Posuň jej doleva a přiber i nový bit.
		DAA	Pak proved dekadickou korekci a
		LD (HL),A	výsledek opět ulož do mem-4.
		DEC HL	Ukazuj na další bajt mem-4.
		DEC C	Sniž počítadlo bajtů o jeden.
		JR NZ,#2E8A,PF-BYTES	Skoč pro každý bajt v mem-4.
		DJNZ #2E7B,PF-BITS	Skoč pro každý bit hodnoty INT x.

Dekadická korekce každého bajtu mem-4 dává dvě desítkové číslice na bajt, čímž vznikne nejvíce 9 číslic. Vzniklá číslice budou nyní převedeny, jedna na bajt, v pamětech mem-3 a mem-4 použitím instrukce RLD.

		XOR A	Registr A je vyčištěn, aby mohl převzít číslice.
		LD HL,#5CA6	Zdrojová adresa 1.bajtu mem-4.
		LD DE,#5CA1	Cílová adresa 1.bajtu mem-3.
		LD B,#09	Existuje maximálně 9 číslic.
		RLD	Levá slabika mem-4 je odhozena.
		LD C,#FF	Hodnota #FF v C je signál: úvodní nula, #00 bez úvodní.
2EA1	PF-DIGITS	RLD	Levá slabika hodnoty v (HL) do A, pravá sl. (HL) doleva.
		JR NZ,#2EA9,PF-INSERT	Skoč, jestliže číslice v A není nulová.
		DEC C	Test na úvodní nulu.
		INC C	
		JR NZ,#2EB3,PF-TEST-2	Jednalo-li se o úvodní nulu, skoč.
2EA9	PF-INSERT	LD (DE),A	Vlož číslci.
		INC DE	Ukazuj na další cílovou adresu.
		INC (IY+113) (mem-5-1st)	Další číslice pro tisk a
		INC (IY+114) (mem-5-2nd)	další před desetinnou tečkou.
		LD C,#00	Změň signál z úvodní nuly na jinou nulu.
2EB3	PF-TEST-2	BIT 0,B	Zdrojový ukazatel má být inkrementován při každém druhém
		JR Z,#2EB8,PF-ALL-9	průchodu smyčkou, kdy B je liché.
		INC HL	
2EB8	PF-ALL-9	DJNZ #2EA1,PF-DIGITS	Skoč zpět pro všech devět číslic.
		LD A,(#5CAB) (mem-5-1st)	Vyzvedni čítač.
		SUB #09	Nebylo-li zde devět číslic bez úvodních nul,
		JR C,#2ECB,PF-MORE	skoč pro další čísla.
		DEC (IY+113) (mem-5-1st)	Příprava na zaokrouhlení: sniž počet na osm.
		LD A,#04	Porovnej devátou číslicí, se čtyřkou
		CP (IY+111) (mem-4-4th)	k nastavení CY pro zaokrouhlení.
		JR #2F0C,PF-ROUND	Skoč na zaokrouhlení.
2ECB	PF-MORE	RST #2B,FP-CALC	Použij opět kalkulátor.
		DEFB #02,výmaz	i je nyní vymazáno.
		DEFB #E2,get-mem-2	f

DEFB #38,konec výpočtu f

f) Zlomková část K je nyní převedena do tiskového bafru.

2ECF	PF-FRACTN	EX DE,HL	DE nyní ukazuje na f.
		CALL #2FBA,FETCH-TWO	Mantisa f je převedena DE', DE.
		EXX	Přepni na zrcadlové registry.
		LD A,#80	Exponent f je snižen na nulu,
		SUB L	posunutím bitů hodnoty f, #80-E posune doprava
		LD L,#00	hodnotu E uloženou v registru (L).
		SET 7,D	Skutečný numerický bit 7 do (D).
		EXX	Obnov hlavní registry.
		CALL #2FDD,SHIFT-FP	Nyní proved posun.
2EDF	PF-FRN-LP	LD A,(IY+113) (mem-5-1st)	Vyzvedni počet číslic.
		CP #08	Nebylo-li osm číslic,
		JR C,#2EEC,PF-FR-DGT	skoč dopředu.
		EXX	Bylo-li osm číslic, použij k zaokrouhlení
		RL D	D' doleva, čímž se nastaví CY.
		EXX	Obnov hlavní registry a
		JR #2FOC,PF-ROUND	skoč dopředu k zaokrouhlení.
2EEC	PF-FR-DGT	LD BC,#0200	Počáteční nula do C a počet dvou do B.
2EEF	PF-FR-EXX	LD A,E	D',E',D a E se násobí deseti
		CALL #2F8B,CA=10*A+C	ve dvou fázích nejprve DE a pak D'E',
		LD E,A	každý bajt bajtem ve dvou krocích.
		LD A,D	
		CALL #2F8B,CA=10*A+C	Celočíselná část výsledku je potom v C aby mohla být
		LD D,A	převedena do tiskového bafru.
		PUSH BC	Počet a výsledek jsou uloženy do obou párů BC a B'C'.
		EXX	
		POP BC	
		DJNZ #2EEF,PF-FR-EXX	Jednou skoč zpět přes výměnu registrů.
		LD HL,#5CA1	Start 1.bajt mem-3.
		LD A,C	Výsledek do A pro uložení.
		LD C,(IY+113) (mem-5-1st)	Počet číslic do C.
		ADD HL,BC	Adresuj první prázdný bajt.
		LD (HL),A	Ulož další číslici.
		INC (IY+113) (mem-5-1st)	Zvyš počet číslic.
		JR #2EDF,PF-FRN-LP	Dokud není všech osm číslic, skákej zpět.

g) Číslice uložené v tiskovém bafru jsou zaokrouhleny na maximum osmi číslic pro tisk.

2FOC	PF-ROUND	PUSH AF	Ušchovej CY pro zaokrouhlení.
		LD HL,#5CA1	Bázová adresa čísla mem-3 1.bajt.
		LD C,(IY+113) (mem-5-1st)	Doplňek (počet číslic v čísle) do BC.
		LD B,#00	
		ADD HL,BC	Adresuj poslední bajt čísla.
		LD B,C	Kopíruj C do B jako čítač.
		POP AF	Obnov CY.
2F18	PF-RND-LP	DEC HL	Toto je poslední bajt čísla.
		LD A,(HL)	Vyzvedni ho do A.
		ADC A,#00	Přičti CY, tedy zaokrouhli.
		LD (HL),A	Ulož zaokrouhlený bajt do bafru.
		AND A	Byla-li to nula, nebo desítka,
		JR Z,#2F25,PF-R-BACK	bude B zaokrouhleno a závěrečná nula nebo desítka,
		CP #0A	nebude započtena do tisku.
		CCF	Nuluj CY pro platnou číslici.
		JR NC,#2F2D,PF-COUNT	Je-li CY=0, skoč.
2F25	PF-R-BACK	DJNZ #2F18,PF-RND-LP	Skoč zpět pro další zaokr. nebo další závěrečné nuly.



	LD (HL),#01	Toto je přetečení doleva.
	INC B	Zde je zapotřebí další jedničky.
	INC (IY+114) (mem-5-2nd)	Je to také další extra číslice před desetinnou tečkou.
2F2D PF-COUNT	LD (IY+113),B (mem-5-1st)	B nyní nastaví počet tištěných číslic mimo záv. nul.
	RST #2B,FP-CALC	f musí být o vymazáno.
	DEFB #02,vymaz	-
	DEFB #3B,konec výpočtu	-
	EXX	Kalkulátorový doplněk, který byl uschován na
	POP HL	zásobníku je obnoven v zrcadlovém HL.
	EXX	

h) Nyní může být číslo vytištěno. Nejdříve bude do C uložen počet tištěných číslic, mimo závěrečných nul, a B bude obsahovat počet číslic před desetinnou tečkou.

	LD BC,(#5CAB) (mem-5-1st)	Čítače jsou nastaveny.
	LD HL,#5CA1	Adresa první číslice.
	LD A,B	Jestliže je více než 9 nebo méně než -4
	CP #09	číslíc před desetinnou tečkou,
	JR C,#2F46,PF-NOT-E	je použit formát E.
	CP #FC	Méně než -4 před desetinnou tečkou
	JR C,#2F6C,PF-E-FRMT	znamená více než čtyři nuly za desetinnou tečkou.
2F46 PF-NOT-E	AND A	Jestliže nejsou žádná čísla před desetinnou tečkou,
	CALL Z,#15EF,OUT-CODE	vytiskni počáteční nulu.

Další vstupní bod se používá k vytištění čísla ve formátu E.

2F4A PF-E-SBRN	XOR A	Nastav A na nulu.
	SUB B	Odečti B. Míinus znamená, že existují číslice
	JR M,2F52,PF-OUT-LP	před desetinnou tečkou, a proto je vytiskni.
	LD B,A	A je nyní použito jako čítač.
	JR #2F5E,PF-DC-OUT	Skoč dopředu k vytištění části za desetinnou tečkou.
2F52 PF-OUT-LP	LD A,C	Kopíruj počet číslic, které mají být vytištěny, do A.
	AND A	Pokud A=0, pak ještě existují závěrečné nuly, které
	JR Z,#2F59,PF-OUT-DT	je třeba vytisknout (skoč neboť B je nenulové).
	LD A,(HL)	Vyzvedni číslici z tiskového bufru.
	INC HL	Ukazuj na další číslici.
	DEC C	Dekrementuj počet o jednu.
2F59 PF-OUT-DT	CALL #15EF,OUT-CODE	Vytiskni příslušnou číslici.
	DJNZ #2F52,PF-OUT-LP	Skákej zpět,dokud B není nulové.
2F5E PF-DC-OUT	LD A,C	Je třeba vytisknout desetinnou tečku.
	AND A	Pokud ovšem C není nulové.
	RET Z	V tom případě se vrať, neboť je všechno hotovo.
	INC B	Přičti jedničku do B (toto přičte desetinnou tečku).
	LD A,#2E	Dej kód pro "." do A.
2F64 PF-DEC-OS	RST #10,PRINT-A-1	Vytiskni tečku.
	LD A,#30	Vlož kód pro nulu.
	DJNZ #2F64,PF-DEC-OS	Skoč zpět k vytištění všech potřebných nul.
	LD B,C	Nastav čítač pro všechny zbývající číslice.
	JR #2F52,PF-OUT-LP	Proveď jejich vytištění.
2F6C PF-E-FRMT	LD D,B	Počet číslic je kopírován do D a
	DEC D	okamžitě zmenšen, aby byl dán i exponent.
	LD B,#01	Je zapotřebí jedna číslice před des. tečkou pro E formát
	CALL #2F4A,PF-E-SBRN	Celá část čísla před E je nyní vytištěna.
	LD A,#45	Vlož kód pro E a
	RST #10,PRINT-A-1	vytiskni ho.
	LD C,D	Exponent do C pro vytištění a
	LD A,C	do A pro otestování.
	AND A	Je testováno znaménko a

	JP	P,#2F83,PF-E-POS	Je-li kladné, skoč.	
	NEG		Jinak neguj A.	
	LD	C,A	Kopíruj A zpět do C pro tisk.	
	LD	A,#2D	Vlož kód pro mínus a	
	JR	#2F85,PF-E-SIGN	vytiskni znaménko.	
2F83	PF-E-POS	LD	A,#2B	Vlož kód pro "+".
2F85	PF-E-SIGN	RST	#10,PRINT-A-1	Nyní vytiskni znaménko ("+" nebo "-")
	LD	B,#00	BC obsahuje exponent;	
	JP	#1A1B,OUT-NUM-1	Vytiskni ho.	

#### PODPROGRAM "CA=10\*A+C"

Tento program je volán podprogramem PRINT-FP k vynásobení každého bajtu D',E',D a E **deseti** tak, **že vrací** celočíselnou část výsledku v registru C. Na vstupu registr A obsahuje bajt, který má být násoben deseti a registr C obsahuje CY z předchozího bajtu. Při návratu obsahuje A výsledný bajt a v registru C je logická hodnota CY flag pro další bajt.

2F8B	CA=10*A+C	PUSH DE	Uschovej kterýkoliv pár DE (může to být D'E'nebo DE).
		LD L,A	Kopíruj násobence do HL.
		LD H,#00	
		LD E,L	Okopíruj jej také
		LD D,H	do DE.
		ADD HL,HL	Dvakrát zdvoj HL.
		ADD HL,HL	
		ADD HL,DE	Přičti DE (což už 5*A) a
		ADD HL,HL	nakonec zdvoj HL (konečně 10*a).
		LD E,C	Kopíruj C do DE (D je nulové) pro sčítání.
		ADD HL,DE	Nyní se HL rovná 10*A+C.
		LD C,H	H je kopírováno do C a
		LD A,L	L je kopírováno do A, čímž je úkol hotov.
		POP DE	Obnov DE a vrať se.
		RET	

#### PODPROGRAM "PŘÍPRAVA NA SČÍTÁNÍ"

Tento podprogram je první ze čtyř podprogramů používaných hlavními aritmetickými podprogramy pro: odčítání, sčítání, násobení a dělení. Podprogram připraví FP-ČÍSLO pro sčítání tím, že nahradí jeho znaménkový bit skutečným číselným bitem logické úrovně 1 a provede negaci čísla (dvojkovou komplementací) je-li záporné. Exponent se vrací v registru A a první bajt je nastaven na #00 pro kladné číslo a #FF pro záporné číslo.

2F9B	PREP-ADD	LD	A,(HL)	Převed exponent do A.
		LD	(HL),#00	Předpokládej kladné číslo.
		AND	A	Je-li číslo nulové
		RET	Z	je příprava skončena.
		INC	HL	Nyní ukazuj na znaménkový bajt.
		BIT	7,(HL)	Nastav Z flag pro pozitivní číslo.
		SET	7,(HL)	Obnov skutečný číselný bit 7.
		DEC	HL	Ukazuj znovu na první bajt.
		RET	Z	Kladná čísla jsou o.k., ale záporná se musí invertovat.
		PUSH	BC	Uschovej jakýkoliv dřívější doplněk.
		LD	BC,#0005	Bude zpracováno pět bajtů.
		ADD	HL,BC	Ukazuj jedno místo za poslední bajt.
		LD	B,C	Převed 5 do B.
		LD	C,A	Uschovej exponent v C.
		SCF		Nastav CY=1.
2FAF	NEG-BYTE	DEC	HL	Ukazuj postupně na všechny bajty.
		LD	A,(HL)	Vyzvedni každý bajt.
		CPL		<b>Jedničkový doplněk</b> bajtu.

ADC	A,#00	Přičti CY pro negaci.
LD	(HL),A	Obnov bajty.
DJNZ	#2FAF,NEG-BYTE	Smyčka proběhne 5-krát.
LD	A,C	Obnov exponent v A.
POP	BC	Obnov jakékoliv dřívější exponenty.
RET		Hotovo.

#### PODPROGRAM "FETCH TWO NUMBERS"

Podprogram je volán z programů ADDITION MULTIPLICATION a DIVISION k vyzvednutí dvou čísel ze zásobníku kalkulátoru a jejich uložení do registrů včetně zrcadlových. Na vstupu ukazuje HL na první bajt prvního čísla a DE ukazuje na první bajt druhého čísla. Při násobení nebo dělení je znaménko výsledku umístěno v druhém bajtu prvního čísla.

2FBA	FETCH-TWO	PUSH HL	Uschovej HL.
		PUSH AF	Uschovej AF.

Pět bajtů prvního čísla bude:  
M1, M2, M3, M4 a M5.

Pět bajtů druhého čísla bude:  
N1, N2, N3, N4 a N5.

LD	C,(HL)	M1 do C.
INC	HL	Další.
LD	B,(HL)	M2 do B.
LD	(HL),A	Kopíruj znaménko výsledku do (HL).
INC	HL	Další.
LD	A,C	M1 do A.
LD	C,(HL)	M3 do C.
PUSH	BC	Uschovej M2 a M3 na zásobník.
INC	HL	Další.
LD	C,(HL)	M4 do C.
INC	HL	Další.
LD	B,(HL)	M5 do B.
EX	DE,HL	HL nyní ukazuje na N1.
LD	D,A	M1 do D.
LD	E,(HL)	N1 do E.
PUSH	DE	Uschovej M1 a N1 na zásobník.
INC	HL	Další.
LD	D,(HL)	N2 do D.
INC	HL	Další.
LD	E,(HL)	N3 do E.
PUSH	DE	Uschovej N2 a N3 na zásobník.
EXX		Přepni na zrcadlové registry.
POP	DE	N2 do D' N3 do E'
POP	HL	M1 do H' N1 do L'
POP	BC	M2 do B' M3 do C'
EXX		Přepni na původní registry.
INC	HL	Další.
LD	D,(HL)	N4 do D.
INC	HL	Další.
LD	E,(HL)	N5 do E.
POP	AF	Obnov AF.
POP	HL	Obnov HL.
RET		Vrať se.

Shrnutí: M1 - M5 jsou v H',B',C',C,B  
N1 - N5 jsou v L',D',E',D,E  
HL ukazuje na první bajt prvního čísla.

## PODPROGRAM "SHIFTOVANÝ SČÍTANEC"

Tento podprogram posunuje FP-číslo až o 32 míst doprava, čímž ho správně nastaví do pozice potřebné pro sčítání. Číslo s menším exponentem již bylo takto ošetřeno před zavoláním tohoto podprogramu. Jakékoliv přetečení doprava (do CY) je přičteno zpět k číslu. Je-li rozdíl exponentů větší než 32, nebo CY přeteká na začátek čísla, je toto číslo nastaveno na nulu tak, aby sčítání nezměnilo druhého sčítance.

2FDD	SHIFT-FP	AND A	Při nulovém rozdílu exponentů se vrať.
		RET Z	
		CP #21	Je-li rozdíl exponentů větší než #20, skoč dopředu.
		JR NC,#2FF9,ADDEND-0	
		PUSH BC	Krátce uschovej BC.
		LD B,A	Rozdíl exponentů do B jako čítač posunů.
2FE5	ONE-SHIFT	EXX	Přepni zrcadlové registry.
		SRA L	Posuň aritmeticky doprava L' a zachovej znaménkový bit.
		RR D	Rotuj doprava s CY
		RR E	registry D',E'
		EXX	
		RR D	a dále D a E.
		RR E	Tímto se všech pět bajtů čísla posune B-krát doprava.
		DJNZ #2FE5,ONE-SHIFT	Skákej zpět do vynulování B.
		POP BC	Obnov původní BC.
		RET NC	Hotovo, nedošlo-li k přetečení.
		CALL #3004,ADD-BACK	Přičti zpět CY.
		RET NZ	Vrať se, pokud CY nepřepadlo hned zpět.
2FF9	ADDEND-0	EXX	Přepni zrcadlové registry.
		XOR A	Vynuluj A.
2FFB	ZEROS-4/5	LD L,#00	Nastav sčítance na nulu v
		LD D,A	D',E',D a E včetně znaménkového bajtu L'.
		LD E,L	Při zavolání z #3160 se vynulují
		EXX	pouze čtyři bajty.
		LD DE,#0000	
		RET	

## PODPROGRAM "ADD BACK" (-ZPĚTNÉ PŘÍČTENÍ)

Podprogram přičítá zpět k číslu každé CY, které přeteklo doprava. V extrémním případě se CY vrací nalevo do čísla. Pokud je podprogram volán během sčítání, znamená tento extrém, že mantisa 0.5 byla posunuta o 32 míst doprava a sčítanec bude nastaven na nulu. Při zavolání z podprogramu MULTIPLICATION znamená tento extrém, že exponent musí být inkrementován, což může způsobit přetečení.

3004	ADD-BACK	INC E	Přičti CY k poslednímu bajtu.
		RET NZ	Vrať se, nedošlo-li k přetečení doleva.
		INC D	Pokračuj na další bajt.
		RET NZ	Vrať se, nedošlo-li k přetečení doleva.
		EXX	Pokračuj na další bajt.
		INC E	Také jej inkrementuj.
		JR NZ,#300D,ALL-ADDED	Skoč nedošlo-li k přetečení doleva.
		INC D	Inkrementuj poslední bajt.
300D	ALL-ADDED	EXX	Obnov hlavní registry.
		RET	Vrať se.

## OPERACE SUBTRACTION (-ODEČÍTÁNÍ)

Doplňák: #03 "subtract"

Tento podprogram jednoduše mění znaménko menšitele a pokračuje dále do ADDITION. Povšimněte si, že HL ukazuje na menšence a DE na menšitele.

300F SUBTRACT	EX	DE,HL	Zaměň ukazatele.
	CALL	#346E,NEGATE	Změň znaménko menšitele.
	EX	DE,HL	Zaměň ukazatele zpět a pokračuj do ADDITION.


## OPERACE ADDITION (-SČÍTÁNÍ)

Doplňák: #0F "addition"

První ze tří hlavních aritmetických podprogramů provádí sčítání dvou čísel v FP formě, každé s čtyřbajtovou adresou a jednobajtovým exponentem. V těchto třech pod programech jsou dvě čísla na vrcholku zásobníku kalkulátoru sečtena, násobena nebo dělena. Výsledkem je pak jedno číslo na vrcholku zásobníku kalkulátoru, čili "poslední hodnota". HL ukazuje na druhé číslo odshora, což je buďto sčítanec, **násobitel** nebo dělenec. DE ukazuje na číslo na vrcholu zásobníku, což je sčítanec, násobenec nebo dělitel. Po ukončení podprogramu HL ukazuje na výsledek, což je poslední hodnota, jejíž hodnota může být považována za STKEND-5. Ale podprogram sčítání nejdříve testuje, zda následující dvě sčítaná čísla jsou "malá celá číslíce". Je-li tomu tak, je jednoduše sečteno HL a BC, a výsledek je uložen na zásobník. Před nebo po sčítání není prováděna dvojková komplementace, neboť čísla, která tuto komplementaci vyžadují, už jsou uložena na zásobníku v potřebném tvaru a připravená ke sčítání.

3014 ADDITION	LD	A,(DE)	Nejsou-li oba první bajty
	OR	(HL)	sčítaných čísel nulové,
	JR	NZ,#303E,FULL-ADDN	skoč na plné sčítání.
	PUSH	DE	Ušchovej ukazatel na druhé číslo.
	INC	HL	Ukazuj na druhý bajt prvního čísla a
	PUSH	HL	uschovej i tento ukazatel.
	INC	HL	Ukazuj na méně významný bajt.
	LD	E,(HL)	Vyzvedni ho do E a
	INC	HL	
	LD	D,(HL)	Vyzvedni významnější bajt.
	INC	HL	
	INC	HL	
	INC	HL	Posuň se na další bajt druhého čísla
	LD	A,(HL)	Vyzvedni ho do A (znaménkový bajt).
	INC	HL	
	LD	C,(HL)	Vyzvedni méně a
	INC	HL	
	LD	B,(HL)	více významné bajty.
	POP	HL	Vyzvedni ukazatel na znaménkový bajt prvního čísla.
	EX	DE,HL	Vlož jej do DE a číslo do HL.
	ADD	HL,BC	Proveď sečtení s výsledkem v HL.
	EX	DE,HL	Výsledek do DE a adresa znaménkového bajtu do HL.
	ADC	A,(HL)	Přičti znaménkové bajty a CY do registru A.
	RRCA		
	ADC	A,#00	Nenulové A indikuje přetečení.
	JR	NZ,#303C,ADDN-OFLW	Obnov ukazatele proveď plné sečtení.
	SBC	A,A	Nastav správný znaménkový bajt pro výsledek.
	LD	(HL),A	Ulož jej do zásobníku.
	INC	HL	Ukazuj na další místo.
	LD	(HL),E	Ulož nižší a
	INC	HL	
	LD	(HL),D	vyšší bajt výsledku.
	DEC	HL	Přesuň ukazatel zpět na pozici <b>prvního bajtu</b> .
	DEC	HL	
	DEC	HL	

POP DE	Obnov STKEND do registru DE
RET	Hotovo.

Poznámka: Povšimněte si, že číslo -65536 zde může nabýt tvaru 00 FF 00 00 00 jako výsledek sčítání dvou menších záporných celých čísel, např: -65000-536 je jednoduše uložen v této formě, což je chyba. Systém SPECTRA nemůže zpracovat toto číslo. Většina funkcí je považuje za nulu a číslo je vypsané jako -1E-38 nebo-li jako -0 v tomto nevhodném formátu. 

303C ADDN-OFLW	DEC HL	Obnov ukazatel na první číslo.
	POP DE	Obnov ukazatel na druhé číslo.
303E FULL-ADDN	CALL #3293,RE-ST-TWO	Proveď opětovné uložení obou čísel na zásobník.

Podprogram "plného sčítání" nejprve volá **PREP-ADD** pro každé číslo, potom vezme dvě čísla ze zásobníku kalkulátoru a uloží číslo s menším exponentem na pozici sčítance, dále volá SHIFT-FP k posunutí sčítance až o 32 míst doprava pro nastavení správné pozice ke sčítání. Vlastní součet proběhne v několika bajtech a při přetečení je proveden jednoduchý posun. Výsledek je **dvojkově** komplementován při záporném čísle a v případě, že nebylo hlášeno aritmetické přeplnění, jinak podprogram pokračuje do TEST-NORM, aby byl normalizován výsledek před jeho uložení na zásobník se správným znaménkovým bitem vloženým do druhého bajtu.

	EXX	Zaměň registry.
	PUSH HL	Ušchovej adresu dalšího literálu.
	EXX	Zaměň registry.
	PUSH DE	Ušchovej ukazatel na 2.sčítanec.
	PUSH HL	Ušchovej ukazatel na 1.sčítanec.
	CALL #2F9B,PREP-ADD	Příprav 1.sčítanec
	LD B,A	a uschovej jeho exponent v B.
	EX DE,HL	Zaměň ukazatele.
	CALL #2F9B,PREP-ADD	Příprav 2.sčítanec
	LD C,A	a uschovej jeho exponent v C.
	CP B	Jestliže první exponent je menší
	JR NC,#3055,SHIFT-LEN	ponech první číslo na pozici 2.sčítance, jinak
	LD A,B	zaměň exponenty a
	LD B,C	
	EX DE,HL	ukazatele.
3055	SHIFT-LEN	
	PUSH AF	Ušchovej větší z exponentů v A.
	SUB B	Rozdíl mezi exponenty je délkou pro posun doprava.
	CALL #2FBA,FETCH-TWO	Vyzvedni dvě čísla ze zásobníku.
	CALL #2FDD,SHIFT-FP	Posuň 2.sčítanec doprava.
	POP AF	Obnov větší exponent a
	POP HL	HL ukazuje na výsledek.
	LD (HL),A	Ulož exponent výsledku.
	PUSH HL	Opět uschovej ukazatel.
	LD L,B	M4 do H a
	LD H,C	M5 do L (viz FETCH-TWO).
	ADD HL,DE	Přičti dva pravé bajty
	EXX	N2 do H' a N3 do L' (viz FETCH-TWO).
	EX DE,HL	
	ADC HL,BC	Přičti levé bajty plus CY.
	EX DE,HL	Výsledek zpět do D',E'.
	LD A,H	Sečti H'L' a CY: tento mechanismus zajistí,
	ADC A,L	že bude zavolán jeden posuv doprava
	LD L,A	pokud součet kladných čísel způsobil přetečení doleva,
	RRA	nebo součet dvou záporných čísel nezpůsobil přetečení
	XOR L	doleva.
	EXX	
	EX DE,HL	Výsledek jde nyní do DE a <b>DE</b> .
	POP HL	Vyzvedni ukazatel na exponent.

	RRA	Test na posun (HL) byly #00 pro kladná čísla a #FF pro záporná čísla).
	JR NC,#307C,TEST-NEG	A je čítač jednoho posunu doprava.
	LD A,#01	Zavolej posun.
	CALL #2FDD,SHIFT-FP	Přičti jedničku k exponentu; toto může způsobit aritmetické přetečení.
	INC (HL)	Test na negativní výsledek.
307C	TEST-NEG	Vyzvedni znaménkový bit z L do A (což správně indikuje znaménko výsledku).
	JR Z,#309F,ADD-REP-6	
	EXX	
	LD A,L	
	AND #80	
	EXX	
	INC HL	
	LD (HL),A	Ulož jej do druhého bajtu výsledku na zásobníku kalkulátoru.
	DEC HL	
	JR Z,#30A5,G0-NC-MLT	Byl-li nulový neprováděj dvojkový doplněk výsledku.
	LD A,E	Vyzvedni 1.bajt.
	NEG	Neguj jej.
	CCF	Komplementuj CY pro pokračující negaci a uschovej tento bajt.
	LD E,A	
	LD A,D	Vezmi další bajt a proveď dvojkový doplněk.
	CPL	Přičti CY pro negaci.
	ADC A,#00	Ušchovej tento bajt.
	LD D,A	Vyzvedni další bajt do A
	EXX	
	LD A,E	E.
	CPL	Proveď komplementaci.
	ADC A,#00	Přičti CY pro negaci.
	LD E,A	Ušchovej bajt.
	LD A,D	Vyzvedni poslední bajt a komplementuj ho.
	CPL	Přičti CY pro negaci.
	ADC A,#00	Pokud nedošlo k přetečení, je hotovo.
	JR NC,#30A3,END-COMPL	Dej 0.5 do mantisy
	RRA	a přičti jedničku k exponentu (to je třeba tehdy, je-li výsledek součtu dvou čísel přesný násobek dvou).
	EXX	Ohlaš chybu, je-li třeba.
309F	ADD-REP-6	
	INC (HL)	
	JP Z,#31AD,REPORT-6	
	EXX	
30A3	END-COMPL	Ušchovej poslední bajt.
	LD D,A	
	EXX	
30A5	G0-NC-MLT	Vynuluj CY.
	XOR A	Vrať se přes TEST-NORM.
	JP #3155,TEST-NORM	

#### PODPROGRAM "HL=HL\*DE"

Podprogram je volán podprogramem "GET-HL\*DE" a MULTIPLICATION, aby provedl 16-ti bitové násobení. Každé přetečení na 16.bitu se zpracuje po návratu z tohoto podprogramu.

30A9	HL=HL*DE	PUSH BC	Ušchovej BC.
		LD B,#10	Jedná se o 16-bitové násobení.
		LD A,H	A obsahuje vyšší bajt.
		LD C,L	C obsahuje nižší bajt.
		LD HL,#0000	Nastav výsledek na nulu.
30B1	HL-LOOP	ADD HL,HL	Zdvojení výsledku.
		JR C,#30BE,HL-END	Skok při přetečení.
		RL C	Rotuj bit 7 registru C do CY.
		RLA	Rotuj CY do bitu 0 registru A a bit 7 do CY.
		JR NC,#30BC,HL-AGAIN	Při nulovém CY skoč.
		ADD HL,DE	Jinak přičti DE a
		JR C,#30BE,HL-END	skoč při přetečení.

30BC	HL-AGAIN	DJNZ #30B1,HL-LOOP	Dokud neproběhlo 16 průchodů, pokračuj.
30BE	HL-END	POP BC	Obnov BC.
		RET	Hotovo.

### PODPROGRAM PŘÍPRAVY NA NÁSOBENÍ NEBO DĚLENÍ

Tento podprogram připravuje FP-číslo pro násobení nebo dělení a vrací CY=1 bylo-li číslo nulové. Znaménko výsledku je v registru A a znaménkový bit čísla je nahrazen skutečným numerickým bitem s hodnotou 1.

30C0	PREP-M/D	CALL #34E9,TEST-ZERO	Jestli-že číslo je nulové,
		RET C	vrať se s CY=1.
		INC HL	Ukazuj na znaménkový bajt,
		XOR (HL)	vyzvedni znaménko výsledku do A a současně nuluj CY.
		SET 7,(HL)	Nastav pravdivý numerický bit.
		DEC HL	Opět ukazuj na exponent.
		RET	Vrať se s nulovým CY.

### OPERACE NÁSOBENÍ

Doplňěk: #04 "multiply"

Tento podprogram nejprve zjistí, zda násobená čísla jsou malá celá čísla, a je-li to pravda, použije se k vyzvednutí těchto čísel ze zásobníku podprogram INT-FETCH. Pak se provede HL=HL\*DE a INT-STORE k uložení výsledku na zásobník. Každé přetečení při tomto "krátkém" násobení (tedy pokud již výsledek není "malé" celé číslo) způsobí skok na násobení v pětibajtové FP formě.

30CA	MULTIPLY	LD A,(DE)	Nejsou-li první bajty obou čísel nulové,
		OR (HL)	
		JR NZ,#30F0,MULT-LONG	skoč na "dlouhé" násobení.
		PUSH DE	Ušchovej ukazatel na druhé,
		PUSH HL	první a
		PUSH DE	opět na druhé číslo.
		CALL #2D7F,INT-FETCH	Vyzvedni znaménko v registru C a číslo v registru DE.
		EX DE,HL	Číslo do HL a
		EX (SP),HL	hned na zásobník, druhý ukazatel do HL.
		LD B,C	Ušchovej první znaménko v B.
		CALL #2D7F,INT-FETCH	Vyzvedni druhé znaménko do C a číslo v registru DE.
		LD A,B	Vytvoř znaménko výsledku v registru A.
		<b>XOR C</b>	
		LD C,A	Ulož je do C.
		POP HL	Obnov první číslo v HL.
		CALL #30A9,HL=HL*DE	Proveď skutečné násobení.
		EX DE,HL	Ušchovej výsledek v DE.
		POP HL	Obnov ukazatel na první číslo.
		JR C,#30EF,MULT-OFLW	Pokud došlo k přetečení, skoč na plné násobení.
30E5		LD A,D	Těchto 5 bajtů zajistí, že 00 FF 00 00 00
		OR E	je nahrazeno nulou (nebylo v třeba, pokud by toto číslo
		JR NZ,#30EA,MULT-RSLT	bylo <b>vyloučeno</b> ze systému).
		LD C,A	
30EA	MULT-RSLT	CALL #2D8E,INT-STORE	Nyní ulož výsledek na zásobník.
		POP DE	Obnov STKEND do DE.
		RET	Hotovo.
30EF	MULT-OFLW	POP DE	Obnov ukazatele na druhé číslo.
30F0	MULT-LONG	CALL #3293,RE-ST-TWO	Proveď "znovu uložení" čísel v piné FP formě (5 bajtů).

Podprogram "plného násobení připraví první číslo pro násobení voláním PREP-M/D, s návratem v případě nuly. Jinak se také připraví druhé číslo voláním PREP-M/D a jestliže je nulové, podprogram nastaví hodnotu výsledku na nulu. Dále se vyzvednou dvě čísla ze zásobníku kalkulátoru a jejich



mantisy jsou vynásobeny obvyklým způsobem, tj. rotací prvního čísla (což je násobitel) doprava a přičtením druhého čísla (což je násobenec) k výsledku v případě, že bit násobitele byl nastaven. Potom jsou sečteny exponenty a testuje se přetečení a podtečení (což dává výsledek nula). Na závěr je výsledek normalizován a uložen na zásobník kalkulátoru se správným znaménkovým bitem v druhém bajtu.

XOR A	A=0, takže znaménko prvního čísla jde do A.
CALL #30C0,PREP-M/D	Příprav první číslo a jestliže je nulové,
RET C	vrať se (výsledek je již nula).
EXX	Zaměň registry.
PUSH HL	Ušchovej adresu dalšího literálu.
EXX	Zaměň registry.
PUSH DE	Ušchovej ukazatel na násobenec.
EX DE,HL	Zaměň ukazatele.
CALL #30C0,PREP-M/D	Příprav druhé číslo
EX DE,HL	a opět zaměň ukazatele.
JR C,#315D,ZERO-RSLT	Byl-li výsledek nula, skoč dopředu.
PUSH HL	Ušchovej ukazatel na výsledek.
CALL #2FBA,FETCH-TWO	Vyzvedni dvě čísla ze zásobníku.
LD A,B	M5 do A (viz FETCH-TWO).
AND A	Příprava na odečítání.
SBC HL,HL	Nuluj HL pro výsledek.
EXX	Zaměň registry.
PUSH HL	Ušchovej M1 a N1 (viz FETCH-TWO).
SBC HL,HL	Nuluj H'L' pro výsledek.
EXX	Zaměň registry.
LD B,#21	B obsahuje #21 pro 33 posunů.
JR #3125,STRT-MLT	Proveď smyčku.

Zde začíná smyčka pro násobitel.

3114	MLT-LOOP	JR NC,#311B,NO-ADD	Nedošlo-li k přetečení, skoč na NO-ADD.
		ADD HL,DE	Jinak přičti násobenec D'E',DE (FETCH-TWO) do výsledku
		EXX	kteřý je vytvářen v H'L' a HL.
		ADC HL,DE	
		EXX	
311B	NO-ADD	EXX	Ať už byl nebo nebyl násobenec přičten,
		RR H	posuň výsledek v H'L'HL doprava tak, že každý
		RR L	bit, který ukápně do CY je okamžitě přijat
		EXX	dalším bajtem a
		RR H	
		RR L	
3125	STRT-MLT	EXX	posun pokračuje do B'C' a CA (viz FETCH-TWO a dále).
		RR B	
		RR C	
		EXX	
		RR C	Poslední bit, který zůstane v CY vyvolá další přičtení
		RRA	násobence k výsledku.
		DJNZ #3114,MLT-LOOP	Proveď 33-krát tuto smyčku čímž se posoudí všechny bity.
		EX DE,HL	Přenes výsledek z
		EXX	
		EX DE,HL	H'L'HL do D'E'DE.
		EXX	

Sečti oba exponenty.

POP BC	Obnov exponenty M1 a N1.
POP HL	Obnov ukazatel na exponentový bajt.
LD A,B	Vytvoř součet dvou exponentů v A a

	ADD	A,C	oprav CY tak,	
	JR	NZ,#313B,MAKE-EXPT	Že nebyl-li součet nulový, nech CY být.	
	AND	A	Jinak CY vynuluj.	
313B	MAKE-EXPT	DEC	A	Připrav se na
	CCF		zvětšení exponentu o #80.	

Zbytek podprogramu je společný jak pro násobení tak pro dělení.

313D	DIVN-EXPT	RLA	Těchto pár bajtů rafinovaně vytváří správný exponentový bajt. Rotací registru A doleva a	
	CCF		doprava vznikne exponent (skutečný exponent + #80) v A.	
	RRA		Je-li P flag nulový není třeba hlásit žádnou chybu.	
	JP	P,#3146,OFLW-1-CLR	Ohlaš přetečení, jestliže CY=0.	
	JR	NC,#31AD,REPORT-6	Vynuluj CY.	
	AND	A	Exponentový bajt je kompletní.	
3146	OFLW-1-CLR	INC	A	Je-li A nulové, je potřebný další test na přetečení.
	JR	NZ,#3151,OFLW-2-CLR		
	JR	C,#3151,OFLW-2-CLR	Je-li CY=0 a výsledek je již v norm. formě (bit 7 D' je nastaven), pak došlo k přet., které je nutno ohlásit	
	EXX		Ale když bit 7 registru 0-0,	
	BIT	7,D	pak je výsledek v rozsahu, tj. pod hodnotou $2^{127}$ .	
	EXX		Ulož exponentový bajt.	
	JR	NZ,#31AD,REPORT-6	Předej pátý bajt výsledku pro normal. sekvenci, tedy přetečení z L do B'.	
3151	OFLW-2-CLR	LD	(HL),A	
	EXX			
	LD	A,B		
	EXX			

Zbytek tohoto podprogramu ošetřuje normalizaci a je společný pro všechny aritmetické podprogramy.

3155	TEST-NORM	JR	NC,#316C,NORMALISE	Je-li CY=0, normalizuj.
	LD	A,(HL)		Jinak ošetři podtečení (nulový výsledek) nebo téměř podtečení (výsledek $2^{128}$ ).
	AND	A		Vrať exponent do A a testuj, je-li
3159	NEAR-ZERO	LD	A,#80	registr A nulový (pro případ $2^{128}$ ) Je-li to pravda,
	JR	Z,#315E,SKIP-ZERO		vytvoř $2^{128}$ v případě normálního čísla, nebo
315D	ZERO-RSLT	XOR	A	produkuj nuly.
315E	SKIP-ZERO	AND	D	Exponent musí být nulový (pro nulu) nebo 1 (pro $2^{128}$ ).
	CALL	#2FFB,ZEROS-4/5		
	RLCA			
	LD	(HL),A		Obnov exponentový bajt.
	JR	C,#3195,OFLOW-CLR		Skoč v případě $2^{128}$ .
	INC	HL		Jinak do druhého bajtu výsledku na kalkulátorovém zásobníku vlož nulu
	LD	(HL),A		
	DEC	HL		
	JR	#3195,OFLOW-CLR		a převed' výsledek.

Aktuální normalizační operace.

316C	NORMALISE	LD	B,#20	Normalizuj výsledek až 32 posuny doleva v D'E'D E
316E	SHIFT-ONE	EXX		přidáním A dokud
	BIT	7,D		bit 7 v registru D' není nastaven na nulu.
	EXX			Registr A obsahuje nulu po sčítání a proto
	JR	NZ,#3186,NORML-NOW		nedojde k žádné změně přesnosti.
	RLCA			A obsahuje 5.bajt z registru B1
	RL	E		po násobení nebo dělení,
	RL	D		ale protože se bere v úvahu vždy jen 32 bitů,
	EXX			nedojde k žádné změně přesnosti.
	RL	E		Povšimněte si, že

RL	D	A je rotováno kruhově s odbočkou při přetečení ....
EXX		eventuálně náhodný proces.
DEC	(HL)	Při každém posunu je dekrementován exponent.
JR	Z,#3159,NEAR-ZERO	Je-li exponent=0, je 2 <sup>-129</sup> zaokrouhleno na 2 <sup>-128</sup> .
DJNZ	#316E,SHIFT-ONE	32-krát do smyčky.
JR	#315D,ZERO-RSLT	Nebyl-li bit 7 registru D' nikdy 1 je celý výsledek nula.

Ukonči normalizaci posouzením CY.

3186	NORML-NOW	RLA	Po normalizaci přičti zpět jakékoliv závěrečné CY,
		JR NC,#3195,OFLOW-CLR	které vstoupilo do A.
		CALL #3004,ADD-BACK	Nepřetéká-li CY hned zpět,
		JR NZ,#3195,OFLOW-CLR	skoč dopředu.
		EXX	V opačném případě nastav mantisu na 0.5
		LD D,#80	
		EXX	
		INC (HL)	a inkrementuj exponent.
		JR Z,#31AD,REPORT-6	Tato akce může vést k aritmetickému přetečení.

Závěrečná část tohoto podprogramu slouží k předání výsledku do bajtů rezervovaných pro výsledek na zásobníku kalkulátoru a k opětovnému nastavení ukazatelů.

3195	OFLOW-CLR	PUSH HL	Ušchovej ukazatel výsledku.
		INC HL	Ukazuj na znaménkový bajt výsledku.
		EXX	Výsledek je přesunut z registrů D'E'D E
		PUSH DE	
		EXX	
		POP BC	do BCDE a pak
		LD A,B	do ACDE.
		RLA	Znaménkový bit je vyzvednut ze svého předchozího uložení
		RL (HL)	a přenese na správnou pozici v bitu 7 1. bajtu mantisy.
		RRA	
		LD (HL),A	Je vložen 1.bajt.
		INC HL	Další.
		LD (HL),C	Je vložen 2.bajt.
		INC HL	Další.
		LD (HL),D	Je vložen 3.bajt.
		INC HL	Další.
		LD (HL),E	Je vložen 4.bajt.
		POP HL	Obnov ukazatel na výsledek.
		POP DE	Obnov ukazatel na druhé číslo.
		EXX	Zaměň registry.
		POP HL	Obnov adresu dalšího literálu.
		EXX	Zaměň registry.
		RET	Hotovo.
31AD	REPORT-6	RST #08,ERROR-1	Ohlaš:
		DEFB #05	6-Number too big

## OPERACE DĚLENÍ

Doplňk: #05 "division"

Tento podprogram nejprve připraví dělitel voláním PREP-M/D a hlásí aritmetické přeplnění, je-li dělitel nulový, **dale** připraví dělenec opětovným voláním PREP-M/D, s návratem v případě nuly. Dále vyzvedne dvě čísla ze zásobníku kalkulátoru a dělí jejich mantisy pomocí běžného dělení t.j. zkušebním odečtem dělitele od dělence a jejich obnovením při přetečení. Jinak se přičte jednička k podílu. Maximální přesnost se dosahuje u čtyřbajtového dělení. Po odečtení exponentu se podprogram napojuje v místě, které je společné pro dělení a násobení a je na adrese #313D DIVN-EXPT.

31AF	DIVISION	CALL #3293,RE-ST-TWO EX DE,HL XOR A CALL #30C0,PREP-M/D JR C,#31AD,REPORT-6 EX DE,HL CALL #30C0,PREP-M/D RET C EXX PUSH HL EXX PUSH DE PUSH HL CALL #2FBA,FETCH-TWO EXX PUSH HL LD H,B LD L,C EXX LD H,C LD L,B XOR A LD B,#DF JR #31E2,DIV-START	Použij FP formy. Zaměň ukazatele. A je vynulováno a znaménko 1.čísla jde do A. Příprav dělitele a ohlaš aritmetické přeplnění v případě, že dělitel je nulový. Zaměň ukazatele. Příprav dělence a je-li nulový, vrať se. Zaměň registry. Ušchovej adresu dalšího literálu. Zaměň registry. Ušchovej ukazatel na dělitel. Ušchovej ukazatel na dělenec. Vyzvedni dvě čísla ze zásobníku. Zaměň registry. Ušchovej H1 a N1 na zásobník. Kopíruj čtyři bajty dělence z registrů B'C'C B (tedy M2,M3,M4 a M5) do registrů H'L'H L.  Nuluj registry A a CY. B bude počítat od -33 do -1 dvojkové doplňky. Skoč do dělicí smyčky pro první zkušební odečet.
		Zde začíná dělicí smyčka.	
31D2	DIV-LOOP	RLA RL C EXX RL C RL B EXX	Posuň výsledek doleva v B'C'C A za pomoci CY.
31DB	DIV-34TH	ADD HL,HL EXX ADC HL,HL EXX	Posuň zbytek dělence zprava doleva v H'L'HL před dalším pokusným odečtem. Jestliže bit přetekl do CY,
31E2	DIV-START	JR C,#31F2,SUBN-ONLY SBC HL,DE EXX SBC HL,DE EXX JR NC,#31F9,N0-RSTORE ADD HL,DE EXX ADC HL,DE EXX AND A JR #31FA,COUNT-ONE	skoč dopředu. Zkušební odečet dělitele v D'E'DE od zbytku dělence v H'L'HL (je zajištěno CY=0 před prvním odečtem, viz předchozí krok).  Nedošlo-li k přetečením, skoč dopředu. Jinak obnov tj. přičti zpět dělitel
31F2	SUBN-ONLY	AND A SBC HL,DE EXX SBC HL,DE EXX	Protože dělitel nepasuje, vynuluj CY, aby nebyl žádný bit pro podíl. Skoč dopředu. Proveď pouhé odečtení bez obnovení a pokračuj na nastavení CY, protože ztracený bit dělence musí být obnoven a použit pro podíl.
31F9	N0-RSTORE	SCF	Jednička pro podíl B'C'C A.
31FA	COUNT-ONE	INC B JP M,#31D2,DIV-LOOP PUSH AF JR Z,#31E2,DIV-START	Inkrementace čítače pro každý průchod smyčkou, které je provedena 32-krát pro všechny bity. Ušchovej každý 33.bit pro přesnost (aktuální CY). Ještě jeden zkušební odečet pro každý 34.bit

Poznámka: Skok je proveden do špatného místa. 34.bit se nikdy nezíská pokud se nebude posouvat dělenec. Proto například důležité výsledky jako 1/10 a 1/1000 nejsou zaokrouhleny tak, jak by měli být. Zaokrouhlení se nikdy neprovede pokud záleží na 34.bitu. Skok měl být na #31DB DIV-34TH tedy bajt #3200 měl, obsahovat **hodnotu #DA místo hodnoty #E**.

LD	E,A	Nyní přesuň čtyři bajty tvořící mantisu výsledku
LD	D,C	z B'C'C A do D'E'D E.
EXX		
LD	E,C	
LD	D,B	
POP	AF	Potom ulož 34. a 33. bit do B' aby
RR	B	byly později při normalizaci vyzvednuty.
POP	AF	
RR	B	
EXX		
POP	BC	Obnov exponentové bajty M1 a N1.
POP	HL	Obnov ukazatel na výsledek.
LD	A,B	Vlož do A rozdíl mezi dvěma exponentovými
SUB	C	bajty s nastavením CY v případě potřeby.
JP	#313D, DIVN-EXPT	<b>Výstup</b> přes DIVN-EXPT.

### ODŘÍZNUTÍ DESETINNÉ ČÁSTI ČÍSLA

Doplňěk: #3A "truncate"

Podprogram vrací výsledek po odříznutí desetinné části čísla x jako poslední hodnotu na zásobníku kalkulátoru. Například T(2.4)=2 a také T(-2.4)=-2. Podprogram se okamžitě vrací v tom případě, kdy x je ve formě "malého" celého čísla. Nula je použita jako výsledek, jestliže exponentový bajt je menší než #81 (neboť  $ABS\ x < 1$ ). Jestliže x je "malé" celé číslo, nebo exponentový bajt je větší než #A0, pak je **x podprogramem vráceno ve své původní formě**. V ostatních případech je nastaven správný počet bajtů čísla x na nulu a v případě potřeby je další bajt rozdělen maskou.

3214	TRUNCATE	LD	A,(HL)	Vyzvedni exponentový bajt čísla x do registru A.
		AND	A	Je-li A nula,
		RET	Z	vrať se, neboť x je již "malé" celé číslo.
		CP	#81	Je-li exponent <b>větší</b> než #80,
		JR	NC,#3221,T-GR-ZERO	skoč.
		LD	(HL),#00	Jinak nastav exponent na nulu.
		LD	A,#20	Ulož 32 do registru A.
		JR	#3272,NIL-BYTES	Za pomoci NIL-BYTES vynuľuj všechny bajty čísla x.
3221	T-GR-ZERO	CP	#91	Jestliže se exponent nerovná #91,
		JR	NZ,#323F,T-SMALL	skoč.

Dalších 26 bajtů je zde proto, aby otestovalo, zda x náhodou není rovno číslu -65536 (#: 91 80 00 00 00). V případě pozitivního výsledku porovnání jsou tyto bajty přepsány na #: 00 FF 00 00 00. A to je CHYBA! Jak již bylo řečeno okolo adresy #303B, není systém SPECTRA schopen takové číslo zpracovat. Zde je vlastně provedeno INT(-65536) a jako výsledek je vytvořena hodnota -1. To je škoda, protože kdyby se s číslem nic nedělo, bylo by vše v perfektním pořádku. Jednoduchou nápravou by bylo vynechání 28 bajtů od adresy #3223 do adresy #323E.

INC	HL	HL ukazuje na 4. bajt x kde končí 17 bitů celočíselné
INC	HL	části čísla x za prvním bitem.
INC	HL	
LD	A,#80	První bit je vyzvednut do registru A s použitím
AND	(HL)	masky #80.
DEC	HL	Tento a předchozích 5 bitů je společně testováno na nulu.
OR	(HL)	
DEC	HL	HL ukazuje na druhý bajt x.

	JR	NZ,#3233,T-FIRST	Test končíf, nebyla-li nalezena nula.
	LD	A,#80	Jinak (test na
	XOR	(HL)	je ukončen a ponechá Z flag nastaven.
3233	T-FIRST	DEC	HL ukazuje na první bajt x.
	JR	NZ,#326C,T-EXPONENT	Je-li Z flag nulový, provede se skok.
	LD	(HL),A	První bajt je nastaven na nulu a
	INC	HL	HL ukazuje na druhý bajt.
	LD	(HL),#FF	Druhý bajt bude #FF a
	DEC	HL	HL opět ukazuje na první bajt.
	LD	A,#18	Je vynulováno posledních 24 bitů
	JR	#3272,NIL-BYTES	pomocí NIL-BYTES, se dokončí stvoření 00 FF 00 00 00.

Leží-li exponentový bajt čísla x v rozsahu #81 až #90 včetně, potom T(x) je "malým" celým číslem a bude kompresováno do jednoho nebo dvou bajtů. Ale nejdříve se otestuje velikost x.

323F	T-SMALL	JR	NC,#326D,X-LARGE	Skoč s exponentovým bajtem #92 a více (bylo by lepší testovat hodnotu #91).
		PUSH	DE	Uschovej STKEND v DE.
		CPL		Rozsah 129<=A<=144 je převeden na 126>=A>=111.
		ADD	A,#91	Rozsah je nyní 15>=A>=0.
		INC	HL	HL ukazuje na druhý bajt a
		LD	D,(HL)	ten je přenesen do D.
		INC	HL	
		LD	E,(HL)	Třetí bajt do E.
		DEC	HL	
		DEC	HL	HL opět ukazuje na první bajt.
		LD	C,#00	Předpokládej kladné číslo.
		BIT	7,D	Testuj záporné číslo (bit 7 registru D je 1)
		JR	Z,#3252,T-NUMERIC	Při kladném čísle skoč.
		DEC	C	Změň znaménko.
3252	T-NUMERIC	SET	7,D	Vlož skutečný numerický bit do registru D.
		LD	B,#08	Testuj, zda A>=8 (jen jeden bajt), nebo je-li třeba dvou
		SUB	B	bajtů.
		ADD	A,B	Ponech A nezměněné.
		JR	C,#325E,T-TEST	Skoč, jsou-li potřeba dva bajty.
		LD	E,D	Dej jeden bajt do E
		LD	D,#00	Nastav D na nulu.
		SUB	B	Nyní 1<=A<=7 počítá potřebné posuny.
325E	T-TEST	JR	Z,#3267,T-STORE	Není-li nutný posun skoč.
		LD	B,A	B bude počítat posuvy.
3261	T-SHIFT	SRL	D	Posunuj B-krát doprava D a E.
		RR	E	aby vzniklo správné číslo.
		DJNZ	#3261,T-SHIFT	Opakuj do vynulování B.
3267	T-STORE	CALL	#2DBE,INT-STORE	Uschovej výsledek na zásobník.
		POP	DE	Obnov STKEND v DE.
		RET		Hotovo.

Zbývá posoudit velké hodnoty čísla x.

326C	T-EXPONENT	LD	A,(HL)	Vyzvedni exponentový bajt čísla x do registru A.
326D	X-LARGE	SUB	#A0	Odečti 160=#A0 od exponentu.
		RET	P	Při plus se vrať - x má nevýznamnou celočíselnou část.
		NEG		Jinak neguj zbytek, čímž získáš počet nulových bitů.

Zde budou vynulovány bity mantisy.

3272	NIL-BYTES	PUSH	DE	Uschovej aktuální hodnotu DE (STKEND).
		EX	DE,HL	HL ukazuje jedno místo za pátý bajt.

	DEC HL	a následovně přímo na pátý bajt čísla x.
	LD B,A	Vezmi do registru B počet bitů, které mají být
	SRL B	vynulovány, a vyděl jej osmi, abys získal počet bajtů,
	SRL B	kterých se to týká.
	SRL B	Jestliže výsledek je nulový,
	JR Z,#3283,BITS-ZERO	skoč dopředu.
327E	LD (HL),#00	Nastav B bajtů na nulu.
	DEC HL	
	DJNZ #327E,BYTE-ZERO	
3283	AND #07	Vezmi A (mod 8) - počet bitů, které mají být nulovány.
	JR Z,#3290,IX-END	Ale v případě, že se nemá už nic dělat, skoč na konec.
	LD B,A	B bude počítat bity.
	LD A,#FF	Připrav masku.
328A	SRL A	Při každém průchodu smyčkou je zprava do A vsunuta nula,
	DJNZ #328A,LESS-MASK	čímž se vytváří správná maska.
	AND (HL)	Nepotřebné bity
	LD (HL),A	jsou pomocí masky vynulovány.
3290	EX DE,HL	Vrať ukazatel do HL.
	POP DE	Vrať STKEND do DE.
	RET	Hotovo.

#### PODPROGRAM "RE-STACK TWO"

Tento podprogram vytváří ze dvou "malých" celých čísel plnohodnotná čísla v FP formě. Toho je dosaženo voláním následujícího podprogramu dvakrát.

3293	RE-ST-TWO	CALL #3269,RESTK-SUB	Volej podprogram a pokračuj do něj jako druhé zavolání
3296	RESTK-SUB	EX DE,HL	Při každém volání zaměň ukazatele.

#### PODPROGRAM "RE-STACK"

Doplňěk: #3D "re-stack"

Tento podprogram vyzvedne a opět uloží jedno číslo, které se ovšem změní z "malého" celého čísla na číslo v plnohodnotném pětibajtovém FP vyjádření. Pro jedno číslo je používán programem ARCTAN a přes kalkulátorový doplňěk funkcemi EXP, LN a "get-argt".

3297	RE-STACK	LD A,(HL)	Neří-li první bajt nulový
		AND A	
		RET NZ	vrať se neboť číslo určitě není "malé" celé číslo.
		PUSH DE	Ušchovej ukazatel z DE.
		CALL #2D7F,INT-FETCH	Vyzvedni znaménko do C a číslo do DE.
		XOR A	Vyčisti registr A.
		INC HL	Ukazuj na 5.místo.
		LD (HL),A	Nastav 5.bajt na nulu.
		DEC HL	Ukazuj na 4.místo.
		LD (HL),A	Nastav 4.bajt=0. Bajty 2 a 3 budou obsahovat mantisu.
		LD B,#91	Nastav B=145 pro exponent, tedy až šestnáct bitů čísla.
		LD A,D	Jestliže je D nulové,
		AND A	bude potřeba maximálně 8 bitů.
		JR NZ,#32B1,RS-NRMLSE	Je zapotřebí více než 8 bitů.
		OR E	Je otestováno také E.
		LD B,D	Nuluj B (=nulový exponent v případě, že E je také nula)
		JR Z,#32BD,RS-STORE	Je-li E skutečně nulové, skoč.
		LD D,E	Přenes E do D (D bylo nulové, ale E ne).
		LD E,B	Nastav E na nulu.
		LD B,#89	Dej do B exponent 137 - ne více než 8 bitů.
32B1	RS-NRMLSE	EX DE,HL	Ukazatel do DE, číslo do HL.
32B2	RSTK-LOOP	DEC B	Dekrementuj exponent při každém posunu.

	ADD	HL,HL	Posuň číslo doprava o jednu pozici.	
	JR	NC,#32B2,RSTK-LOOP	Dokud CY=0, skákej zpět do smyčky.	
	RRC	C	Dej znaménkový bit do CY a	
	RR	H	vlož jej na své místo do čísla,	
	RR	L	které je právě posunuto o jedno místo.	
	EX	DE,HL	Ukazatel na čtvrtý bajt zpět do HL.	
32BD	RS-STORE	DEC	HL	Ukazuj na 3.místo.
		LD	(HL),E	Uschovej 3.bajt.
		DEC	HL	<b>Ukazuj na 2.místo.</b>
		LD	(HL),D	<b>Uschovej 2.bajt.</b>
		DEC	HL	Ukazuj na 1.místo.
		LD	(HL),B	Uschovej exponentový bajt.
		POP	DE	Obnov původní ukazatel v DE.
		RET		Hotovo.



## FLOATING POINT KALKULÁTOR

### TABULKA KONSTANT

Tato první tabulka obsahuje pět užitečných a často používaných čísel: nula, jedna, polovina, polovina PI a deset. Čísla jsou uložena ve staženém tvaru, který je expandován podprogramem STACK LITERALS na požadovaný tvar floating-point (=FP tvar).

	data	konstanta	expandovaný tvar
32C5	stk-zero	DEFB #00 DEFB #B0 DEFB #00	nula 00 00 00 00 00
32C8	stk-one	DEFB #40 DEFB #B0 DEFB #00 DEFB #01	jedna 00 00 01 00 00
32CC	stk-half	DEFB #30 DEFB #00	polovina 80 00 00 00 00
32CE	stk-pi/2	DEFB #F1 DEFB #49 DEFB #0F DEFB #DA DEFB #A2	polovina pi 81 49 0F DA A2
32D3	stk-ten	DEFB #40 DEFB #B0 DEFB #00 DEFB #0A	deset 00 00 0A 00 00

### TABULKA ADRES

Druhá tabulka slouží k vyhledávání adres 66ti operačních podprogramů kalkulátoru. Doplnky, které umožňují indexování v této tabulce jsou odvozeny buď z řídicích kódů, použitých v podprogramu SCANNING viz. #2734, nebo z literálů, které následují po instrukci RST #28.

doplněk	název	adresa	doplněk	název	adresa
32D7 00	skok-pravda	368F	3319 21	tan	37DA
32D9 01	záměna	343C	331B 22	asn	3833
32DB 02	výmaz	33A1	331D 23	acs	3843
32DD 03	odečtení	300F	331F 24	atn	37E2
32DF 04	násobení	30CA	3321 25	ln	3713
32E1 05	dělení	31AF	3323 26	exp	36C4
32E3 06	umocnění	3851	3325 27	int	36AF
32E5 07	číslo OR číslo	351B	3327 28	sqr	384A
32E7 08	číslo AND číslo	3524	3329 29	sgn	3492
32E9 09	číslo <= číslo	353B	332B 2A	abs	346A
32EB 0A	číslo >= číslo	353B	332D 2B	peek	34AC
32ED 0B	číslo <> číslo	353B	332F 2C	in	34A5
32EF 0C	číslo > číslo	353B	3331 2D	usr-no	34B3
32F1 0D	číslo < číslo	353B	3333 2E	str\$	361F
32F3 0E	číslo = číslo	353B	3335 2F	chr\$	35C9
32F5 0F	sčítání	3014	3337 30	not	3501
32F7 10	řetězec AND číslo	352D	3339 31	zdvojení	33C0
32F9 11	řetězec <= řetězec	353B	333B 32	n-mod-m	36A0
32FB 12	řetězec >= řetězec	353B	333D 33	relativní skok	3686
32FD 13	řetězec <> řetězec	353B	333F 34	stk-data	33C6
32FF 14	řetězec > řetězec	353B	3341 35	dec-jr-nz	367A

3301	15	řetězec	<	řetězec	353B	3343	36	číslo	<	0	3506
3303	16	řetězec	=	řetězec	353B	3345	37	číslo	>	0	34F9
3305	17	řetězec	+	řetězec	359C	3347	38	konec	výpočtu		369B
3307	18	val\$			35DE	3349	39	get-argument			3783
3309	19	usr\$			34BC	3348	3A	int	se	zaokr.	3214
330B	1A	načtení	do	A	3645	334D	3B	fp-calc-2			33A2
330D	1B	negace			346E	334F	3C	e-to-fp			2D4F
330F	1C	code			3669	3351	3D	re-stack			3297
3311	1D	val			35DE	3353	3E	serie	06	atd.	3449
3313	1E	len			3674	3355	3F	stk-nula	atd.		341B
3315	1F	sin			37B5	3357	40	st-mem-0	atd.		342D
3317	20	cos			37AA	3359	41	get-mem-0	atd.		340F

Poznámka: Poslední čtyři podprogramy jsou víceúčelové a vstupuje se do nich s parametrem, který je kopíř bitů 0 až 4 původního literálu:

Doplňěk #3E: série#06, série#08 a série#0C odpovídá literálům: #86,#88 a #8C.

Doplňěk #3F: stk-nula, stk-jedna, stk-polovina, stk-pi/2, stk-deset odpovídá literálům #A0 až #A4.

Doplňěk #40: st-mem-0 až st-mem-5 odpovídá literálům #C0 až #C5.

Doplňěk #41: get-mem-0 až get-mem-5 odpovídá literálům #E0 až #E5.

#### PODPROGRAM "CALCULATE"

Tento podprogram provádí výpočty s čísly ve formě s pohyblivou řádovou tečkou. (Dále jen FP). Výpočty lze rozdělit do tří skupin:

- Binární operace například sčítání, kdy dvě FP čísla jsou sečtena a výsledek pak tvoří "poslední hodnotu".
- Unární operace například sin, kdy "poslední hodnota" je změněna tak, že je výsledkem dané operace.
- Manipulační operace například st-mem-0, kde "poslední hodnota" je okopířována do prvních pěti bajtů paměti kalkulátoru.

Operace, které se mají provést jsou specifikovány sledem datových bajtů takzvaných literálů, které následují po instrukci RST#28 a tento sled musí být zakončen literálem #38 - konec výpočtu. V případě potřeby vykonání jediné početní operace je možné vložit doplněk do registru B a vyvolat operaci jediného výpočtu pomocí literálu #3B.

Také je možné volat tento podprogram rekurzivně, tedy ze sama sebe, a v takovém případě lze použít systémovou proměnnou BREG jako čítač, který řídí počet operací před návratem. První část tohoto podprogramu nastavuje registry na požadované hodnoty, vypočítává doplňky a případné parametry z literálů, které jsou testovány. Doplněk slouží k vyhledání adresy příslušného podprogramu v tabulce konstant. Parametry jsou použity při volání víceúčelových podprogramů.

Poznámka: FP číslo může být ve skutečnosti také soubor řetězcových parametrů.

335B	CALCULATE	CALL	#35BF,STK-PNTRS	Předpokládej unární operaci a proto nastav HL tak, aby ukazovalo na 1.bajt "poslední hodnoty" a DE na STKEND.
335E	GEN-ENT-1	LD	A,B	Jinak předej doplněk jednoduché operace do BREG, při rekurzivním
		LD	(#5C67),A (BREG)	volání se nejedná o doplněk, ale o parametr (=čítač).
3362	GEN-ENT-2	EXX		Návř. adr. podprogramu je uložena v zrcadlovém HL. Toto číslo je také ukaz. na 1.literál. Vstup v tomto místě
		EX	(SP),HL	

	EXX		se používá, když BREG je čítač a nemá se zničit.
3365	RE-ENTRY	LD (#5C65),DE (STKEND)	Zde začíná smyčka <b>obsl.</b> všechny literály následující
	EXX		za volací instrukcí. Přepni zrcadlové registry a
	LD	A,(HL)	vyzvedni literál pro tuto smyčku.
	INC	HL	H'L' ukazuje na další literál a
336C	SCAN-ENT	PUSH HL	je krátce uschováno na zásobníku. Tento bod je používán
			pro podprogram SINGLE CALCULATION nalezení <b>pož. podpr.</b>
	AND	A	Testuj registr A a odděl
	JP	P,#3380,FIRST-3D	<b>jed. liter.</b> od víceúčelových. Skoč s literálem #00-#3D
	LD	D,A	Uschovej literál v registru D.
	AND	#60	Pokračuj pouze s bity 5 a 6 a
	RRCA		čtyřmi posuny z nich udělej bity 1 a 2.
	RRCA		
	RRCA		
	RRCA		
	ADD	A,#7C	Požadované doplňky jsou <b>#3E</b> -#41 a
	LD	L,A	L obsahuje požadovaný doplněk.
	LD	A,D	Vytvoř parametr M tak, že
	AND	#1F	použiješ bity 1,2,3 a 4 z původního literálu.
	JR	#338E,ENT-TABLE	Najdi adresu požadovaného programu.
3380	FIRST-3D	CP #18	
	JR	NC,#338C,DOUBLE-A	Při odskoku proveď unární operaci.
	EXX		Všechny <b>podpr.</b> , které jsou binárními operacemi vyžadují,
	LD	BC,#FFFB	aby HL ukazovalo na první operand a
	LD	D,H	DE na druhý ("poslední hodnotu") tak, jak jsou na <b>zásob.</b>
	LD	E,L	kalkulátoru.
	ADD	HL,BC	
	EXX		
338C	DOUBLE-A	RLCA	Protože každá položka v tabulce je dvoubajtová,
	LD	L,A	je vytvořený doplněk zdvojnásoben.
338E	ENT-TABLE	LD DE,#32D7	Bázová adresa tabulky.
	LD	H,#00	Adresa požadované položky v tabulce
	ADD	HL,DE	je vytvořena v HL a
	LD	E,(HL)	adresa požadovaného podprogramu
	INC	HL	
	LD	D,(HL)	je vyzvednuta do DE.
	LD	HL,#3365	Adresa opětovného vstupu
	EX	(SP),HL	je uložena na zásobníku pod
	PUSH	DE	adresu požadovaného podprogramu.
	EXX		Přepni na hlavní registry.
	LD	BC,(#5C66)(STKEND-hi)	BREG je uložena do B, čímž se vrací <b>dopl.</b> jediné operace.
33A1	delete	RET	Nyní je proveden nepřímý skok do požadovaného <b>podpr.</b>

#### PODPROGRAM "DELETE"

Doplňk: #02 "delete"

Tento podprogram obsahuje pouze instrukci RET. Literál #02 způsobí, že tento program je považován za binární operaci. Proto se do něj vstupuje s prvním číslem adresovaným HL a druhým číslem adresovaným DE a vytvořený výsledek bude opět adresován registrovým párem HL. Instrukce RET tedy způsobí, že první číslo je považováno za "poslední hodnotu" a druhé číslo je vymazáno. Druhé číslo samozřejmě není vymazáno z paměti, ale stává se pouze neaktivním a pravděpodobně bude brzy přepsáno.

#### PODPROGRAM "SINGLE OPERATION"

Doplňk: #3B "fp-calc-2"

Tento podprogram je volán z podprogramu SCANNING na adrese #2757 a používá se k provedení jednoduché aritmetické operace. Doplněk uvažované operace se předává kalkulátoru v registru B odkud je předán do systémové proměnné BREG.

Výsledkem provedení tohoto podprogramu je zejména provedení skoku do příslušného podprogramu pro jedinou operaci.

33A2	FP-CALC-2	POP AF	Odhoď návratovou adresu.
		LD A,(#5C67) (BREG)	Převeď doplněk do A.
		EXX	Přepni na zrcadlové registry.
		JR #336C,SCAN-ENT	Nalezni požadovanou adresu, uloži ji na zásobník a proved příslušnou operaci.

#### PODPROGRAM "TEST 5-SPACES"

Podprogram testuje, zda je v paměti dostatek místa pro pět bajtů FP čísla, které má být přidáno na zásobník kalkulátoru.

33A9	TEST-5-SP	PUSH DE	Krátce uschovej DE
		PUSH HL	a také HL.
		LD BC,#0005	Test bude na 5 bajtů
		CALL #1F05,TEST-ROOM	Proveď test.
		POP HL	Obnov HL a
		POP DE	také DE.
		RET	Hotovo.

#### PODPROGRAM "STACK NUMBER"

Tento podprogram je volán z příkazu BEEP a z podprogramu SCANNING dvakrát, aby okopíroval STKEND do DE, přenesl FP číslo na zásobník kalkulátoru a opět nastavil STKEND na hodnotu v DE. Pro skutečný přesun je použit podprogram MOVE-FP.

33B4	STACK-NUM	LD DE,(#5C65) (STKEND)	Okopíruj STKEND do DE jako cílovou adresu.
		CALL #33C0,MOVE-FP	Přesuň číslo.
		LD (#5C65),DE (STKEND)	Obnov hodnotu STKEND.
		RET	Hotovo.

#### PODPROGRAM "MOVE A FLOATING - POINT NUMBER"

Doplněk: #31 "duplicate"

Tento program přenesle FP číslo na vrchol zásobníku kalkulátoru (ve třech případech) nebo z vrcholu kalkulátoru do paměti kalkulátoru (v jednom případě).

Je také používán kalkulátorem pro obyčejné zdvojení čísla na vrcholu kalkulátoru, čímž se zásobník kalkulátoru rozšíří o pět bajtů.

33C0	MOVE-FP	CALL #33A9,TEST-5-SP	Testuje se místo.
		LDIR	Přenes.
		RET	Hotovo.

#### PODPROGRAM "STACK LITERALS"

Doplněk: #34 "stk-data"

Tento podprogram vytvoří z dodaného čísla 2,3,4 nebo 5 literálů a umístí je jako poslední hodnotu na zásobník kalkulátoru. Je-li tento podprogram vyvolán literálem #34, jsou uvažované literály uloženy za touto konstantou, je-li volán podprogramem SERIES GENERATOR, jsou literály dodány podprogramem vytvářejícím postoupnosti čísel a je-li volán podprogramy SKIP KONSTANTS nebo STACK A KONSTANTS, jsou literály získány z tabulky konstant na adrese #32C5-#32D6. V každém případě je první literál vydělen hodnotou #40 a celočíselná hodnota podílu plus 1 určí zda bude použito 1, 2, 3 nebo 4 následujících literálů jako čísla mantisy. Každý nevyplněný bajt z pěti bajtů pro FP číslo je nastaven na nulu.

První literál se také používá k určení exponentu po snížení modulu #40 pokud ovšem zbytek není nulový.

V tom případě se použije druhý literál tak jak je, bez snižování modulu #40. V každém případě se k literálu přičte #50, čímž se vytvoří zvětšený exponentový bajt e (skutečný exponent e' plus #80).

Zbývajících pět bajtů je uloženo na zásobník včetně potřebných nul a podprogram **Vrací zpět**.

33C6	STK-DATA	LD H,D LD L,E	Tento podprogram <b>vykoná</b> manipulační operaci přičtení poslední hodnoty na zásobník kalkulátoru a HL ukazuje za poslední hodnotu (výsledek).
33C8	STK-CONST	CALL #33A9,TEST-5-SP EXX PUSH HL EXX EX (SP),HL PUSH BC LD A,(HL) AND #C0 RLCA RLCA LD C,A	Testuj potřebný prostor. Přepni na zrcadlové registry. Ušchovej ukazatel na další literál. Přepni zpět. Zaměň výsledný ukazatel a ukazatel na další literál. Ušchovej krátce BC. Vyzvedni první literál do A. Vyděl ho hodnotou #40 pro dosažení hodnot 0,1,2 nebo <b>3</b> .
		INC C LD A,(HL) AND #3F JR NZ,#33DE,FORM-EXP INC HL LD A,(HL)	Celočíselná část e je předána do C a zvětšena pro dosažení hodnoty 1,2,3 nebo 4 (= počet liter. mantisy). Opět je vyzvednut literál, snížen o hodnotu modulu #40 a bude použit při nenulovém zbytku.
33DE	FORM-EXP	ADD A,#50 LD (DE),A LD A,#05 SUB C INC HL INC DE LD B,#00 LDIR POP BC EX (SP),HL EXX POP HL EXX LD B,A XOR A	Jinak bude použit další neredukovaný literál. Exponent e je formován přičtením hodnoty #50 a předán na zásobník DE kalkulátoru jako 1. z 5 hodnot výsledku. C literálů je vyzvednuto ze zdroje a uloženo do bajtů výsledku. Obnov BC. Zaměň zpět ukazatel <b>výsledku</b> a ukazatel literálu ze zásobníku <b>do</b> H'L'.
33F1	STK-ZEROS	DEC B RET Z LD (DE),A INC DE JR #33F1,STK-ZEROS	Počet nulových bajtů je udán hodnotou 5-C-1 a tento počet nul je přidán k výsledku, aby byl doplněn požadovaný počet bajtů.

#### PODPROGRAM "SKIP KONSTANTS"

Do tohoto podprogramu se vstupuje s registrovým párem HL obsahujícím **bázovou adresu kalkulátorové tabulky konstant a registr A obsahuje** parametr určující jednu z pěti konstant. Podprogram provádí nulové operace přenesením pěti bajtů každé nepotřebné konstanty na adresy #0000, #0001, #0002, #0003 a #0004 v paměti ROM až do nalezení požadované konstanty. Na výstupu obsahuje HL bázovou adresu požadované konstanty v tabulce konstant.

33F7	SKIP-CONS	AND A	Je-li parametr nula
33F8	SKIP-NEXT	RET Z PUSH AF	nebo byla-li nalezena potřebná konstanta, vrať se. Ušchovej parametr

PUSH DE	a ukazatel výsledku.
LD DE,#0000	Mrtvá adresa.
CALL #33C8,STK-CONST	Proveď imaginární uložení expandované konstanty.
POP DE	Obnov ukazatel
POP AF	výsledku a parametr.
DEC A	Počítej smyčky a posuď další
JR #33F8,SKIP-NEXT	hodnoty čítače.

#### PODPROGRAM "MEMORY LOCATION"

Podprogram nalezne básovou adresu každých pěti bajtů v paměťové oblasti kalkulátoru na kterou nebo z které má být přeneseno FP číslo do nebo ze zásobníku kalkulátoru. Tato operace se provede pětinasobným přičtením **dodaného parametru k** básově adrese pro oblast jejíž adresa je v registrovém páru HL. Pověšiměte si, že při zpracování proměnné FOR-NEXT jsou ukazatele zaměněny tak, aby proměnná byla považována za oblast kalkulátoru (viz #1D20).

3406 LOC-MEM	LD C,A	Kopíruj parametr do C.
	RLCA	Zdvoj parametr.
	RLCA	Zdvoj výsledek.
	ADD A,C	Přičti hodnotu parametru, což dá 5 násobek původní <b>hodn.</b>
	LD C,A	Převeď výsledek do registrového páru BC.
	LD B,#00	
	ADD HL,BC	Vytvoř novou básovou adresu.
	RET	Hotovo.

#### PODPROGRAM "GET FROM MEMORY AREA"

Doplňky: #E0 až #E5 "get-mem-0" až "get-mem-5"

Podprogram je volán použitím literálů #E0 až #E5 a parametry odvozené z těchto literálů jsou v registru A. Podprogram volá MEMORY LOCATION k uložení požadované zdrojové adresy do HL a MOVE A FLOATING-POINT NUMBER k okopírování příslušných pěti bajtů z paměti **kalkulátoru**, aby vytvořil novou "poslední hodnotu".

340F GET-MEM-0	PUSH DE	Uschovej ukazatel výsledku.
etc.	LD HL,(#5C68) (MEM)	Vyzvedni ukazatel na aktuální paměťovou oblast.
	CALL #3406,LOC-MEM	Bázová adresa je nalezena.
	CALL #33C0,MOVE-FP	Pět bajtů přeneseno.
	POP HL	Ukazatel výsledku nastaven.
	RET	Hotovo.

#### PODPROGRAM "STACK A (KONSTANTS"

Doplňky: #A0 až #A4 "stk-nula" , "stk-jedna" , "stk-půl"  
"stk-pi/2" , "stk-deset".

Tento podprogram používá SKIP KONSTANTS k nalezení básově adresy požadované konstanty z tabulky konstant a potom **STACK LITERALS**, aby se dosáhlo expandované formy konstanty jako "poslední **hod.**" na zásobníku kalkulátoru.

341B STK-ZERO	LD H,D	Nastav HL jako ukazatel výsledku.
etc.	LD L,E	
	EXX	Přepni na zrcadlové registry.
	PUSH HL	Uschovej adresu dalšího literálu.
	LD HL,#32C5	Bázová adresa tabulky konstant.
	EXX	Zpět na hlavní registry.
	CALL #33F7,SKIP-CONS	Nalezni požadovanou básovou adresu.
	CALL #33C8,STK-CONST	Expanduj konstantu.
	EXX	
	POP HL	Obnov ukazatel na další literál.

EXX  
RET Hotovo.

#### PODPROGRAM "STORE IN MEMORY AREA"

Doplňk: #C0 až #C5 "st-mem-0" až "st-mem-5"

Tento podprogram se volá použitím literálů #C0 až #C5 a parametry odvozené z těchto literálů jsou v registru A. Podprogram je velmi podobný podprogramu GET FROM MEMORY, ale zdrojové a cílové ukazatele jsou zaměněny.

342D	ST-MEM-0	PUSH HL	Uschovej ukazatel výsledku.
	etc.	EX DE,HL	Zdroj krátce do DE.
		LD HL,(#5C68) (MEM)	Vyzvedni ukazatel na aktuální paměťovou oblast.
		CALL #3406,LOC-MEM	Je nalezena bázeová adresa.
		EX DE,HL	Je zaměněn zdrojový a cílový ukazatel.
		CALL #33C0,MOVE-FP	Přenes pět bajtů.
		EX DE,HL	STKEND do DE.
		POP HL	Ukazatel výsledku do HL.
		RET	Hotovo.

Povšimněte si, že ukazatele HL a DE zůstávají tak jak byly, ukazujíce na STKEND-5 a STKEND, takže poslední hodnota zůstává na zásobníku kalkulátoru. Je-li to třeba, může být vyřazena použitím "delete".

#### PODPROGRAM "EXCHANGE" (ZÁMĚNA)

Doplňk: #01 "exchange"

Tato binární operace zamění první a druhé číslo na vrcholu zásobníku kalkulátoru.

343C	EXCHANGE	LD B,#05	Jde o pět bajtů.
343E	SWAP-BYTE	LD A,(DE)	Každý bajt druhého
		LD C,(HL)	čísla a každý bajt prvního čísla.
		EX DE,HL	Přepni zdroj na cíl.
		LD (DE),A	Na první a
		LD (HL),C	na druhé číslo.
		INC HL	Přesuň se na
		INC DE	posouzení dalšího páru bajtů.
		DJNZ #343E,SWAP-BYTE	Zaměň pět bajtů.
		EX DE,HL	Dej do pořádku ukazatele, protože číslo 5 je liché.
		RET	Hotovo.

#### PODPROGRAM "SERIES GENERATOR" (GENERÁTOR ŘAD)

Doplňky: #86, #88 a #8C "series-06", "series-08" a "series-0C"

Tento důležitý podprogram generuje řady rozvojem podle Čebyševových polynomů a jejich hodnoty jsou pak použity k výpočtu hodnot funkcí SIN, ATN, LN a EXP a tím pádem i dalších aritmetických funkcí, které jsou s předchozími v nějakém aritmetickém vztahu (COS, TAN, ASN, ACS ...SQR).

Polynomy jsou generovány pro  $n=1,2,\dots$ , rekurentním vztahem:  $T_{n+1}(z)=2*z*T_n(z)-T_{n-1}(z)$ , kde  $T_n(z)$  je  $n$ -tý prvek Čebyševova polynomu argumentu ( $z$ ). Řada ve skutečnosti vypadá takto:  $T_0, 2z, 1-2z^2, \dots, 2*T_n-1$ , kde  $n$  je 6 pro SIN, 8 pro EXP a 12 pro LN a ATN.

Koeficienty mocnin hodnoty ( $z$ ) v těchto polynomech mohou být nalezeny v knize Handbook of Mathematical Functions od M. Abramoviče a I. A. Steguna (V Doveru 1965), strana 795. Jednoduše řečeno, podprogram je volán s poslední hodnotou na zásobníku kalkulátoru, řekněme  $Z$ , což je číslo mající nějaký jednoduchý vztah k argumentu, řekněme  $X$ , např. pro funkci SIN  $X$ . Volající podprogram také dodává seznam požadovaných konstant (např. 6 konstant pro SIN).

SERIES GENERÁTOR pak pracuje s těmito daty a vrací volajícímu podprogramu "poslední hodnotu", která obsahuje jednoduchý výsledek pro požadovanou funkci např. SIN  $X$ .

Podprogram může být rozdělen na čtyři části :

### 1) Nastavení čítače průchodů smyček :

Volající podprogram předá své parametry v registru A pro použití jako čítač. Do kalkulátoru se vstupují v bodě GEN-ENT-1, takže čítač může být nastaven.

```
3449 SERIES-06 LD B,A Předej parametr do B.
etc. CALL #335E,GEN-ENT-1 Toto je ve skutečnosti RST #28, ale s nastavením čítače.
```

### 2) Práce s poslední hodnotou Z :

Smyčka generátoru vyžaduje, aby hodnota  $2*Z$  byla uložena v mem-0, v mem-2 má být uložena nula, která má být zároveň poslední hodnotou.

```
DEFB #31,zdvojení Z,Z
DEFB #0F,součet 2*Z
DEFB #C0,st-mem-0 2*Z mem-0 obsahuje 2*Z.
DEFB #02,výmaz -
DEFB #A0,stk-nula 0
DEFB #C2,st-mem-2 0 mem-2 obsahuje 0.
```

### 3) Hlavní smyčka :

Řada je generována ve smyčce s použitím BREG jako čítače. Konstanty z volajícího podprogramu jsou postupně ukládány na zásobník podprogramem STK-DATA. Je proveden opětovný vstup do kalkulátoru v bodě GEN-ENT-2, tak aby se neznížila hodnota BREG a řada se vytváří v této formě:  $B(R)-2*Z*B(R-1)-B(R-2)+A(R)$ , pro  $R=1,2,\dots,N$ , kde  $A(1),A(2),\dots,A(N)$  jsou konstanty dodané volajícím podprogramem (SIN,ATN,LN a EXP) a  $B(0)=0=B(-1)$ .  $(R+1)$ ná smyčka začíná s  $B(R)$  na zásobníku a s  $2*Z$ ,  $B(R-2)$  a  $B(R-1)$  v mem-0, mem-1 a mem-2.

```
3453 G-LOOP DEFB #31,zdvojení B(R),B(R)
DEFB #E0,get-mem-0 B(R),B(R),2*Z
DEFB #04,násobení B(R),2*B(R)*Z
DEFB #E2,get-mem-2 B(R),2*B(R)*Z,B(R-1)
DEFB #C1,st-mem-1 B(R-1) do mem-1
DEFB #03,odečet B(R),2*B(R)*Z-B(R-1)
DEFB #38,konec výpočtu
```

Na zásobník kalkulátoru je umístěna další konstanta.

```
CALL #33C6,STK-DATA B(R),2*B(R)*Z-B(R-1),A(R+1)
```

Toto je opětovný vstup do kalkulátoru bez poškození BREG.

```
CALL #3362,GEN-ENT-2
DEFB #0F,součet B(R),2*B(R)*Z-B(R-1)+A(R+1)
DEFB #01,záměna 2*B(R)*Z-B(R-1)+A(R+1),B(R)
DEFB #C2,st-mem-2 B(R) do mem-2
DEFB #02,výmaz 2*B(R)*Z-B(R-1)+A(R+1)=B(R+1)
DEFB #35,dec-jr-nz B(R+1)
DEFB #EE,na #3453,G-LOOP
```

### 4) Odečtení čísla $B(N-2)$ :

Smyčka ponechává  $B(N)$  na zásobníku a požadovaný výsledek je dán hodnotou  $B(N)-B(N-2)$ .

```
DEFB #E1,get-mem-1 B(N),B(N-2)
DEFB #03,odečet B(N)-B(N-2)
DEFB #38,konec výpočtu
RET Hotovo.
```



## FUNKCE "ABSOLUTE MAGNITUDE"

Doplňk: #2A "abs"

Tento podprogram provede unární operaci tím, že zajistí, aby znaménkový bit FP čísla byl vynulován. "Malá celá čísla" **musí** být posouzena odděleně. Většinu práce však vykoná následující podprogram "unární mínus".

346A	ABS	LD	B,#FF	B je nastaveno na #FF.
		JR	#3474,NEG-TEST	Je provedeno unární mínus.

## OPERACE UNÁRY MINUS

Doplňk: #1B "negate"

Tento podprogram provede unární operaci tím, že změní znaménko poslední hodnoty na zásobníku kalkulátoru. Nula zůstává nezměněna. FP čísla jsou zpracována tak, že jejich znaménkový bit je nulován pro "abs" nebo překlopen pro "negate". "Malá celá čísla" mají vynulován znaménkový bajt pro "abs" nebo změněn pro "negate".

346E	NEGATE	CALL	#34E9,TEST-ZERO	Je-li číslo nula,
		RET	C	návrat s bajty 00 00 00 00 00 zůstanou nezměněny.
		LD	B,#00	B je nastaveno na #00 pro "negate".

Zde je vstup pro "ABS".

3474	NEG-TEST	LD	A,(HL)	Jestliže první bajt je nulový, provede
		AND	A	se
		JR	Z,#3483,INT-CASE	zpracování "malého celého čísla".
		INC	HL	Ukazuj na druhý bajt.
		LD	A,B	Vyzvedni #FF pro "abs" a #00 pro "negate".
		AND	#80	Změň na #80 pro "abs" a #00 pro "negate".
		OR	(HL)	Nastaví se bit 7 pro "abs", nic se nezmění pro "negate".
		RLA		Nyní je změněn
		CCF		bit 7, což způsobí, že bit 7 2.bajtu je <b>nulován</b> .
		RRA		
		LD	(HL),A	Nový bajt je uložen.
		DEC	HL	HL ukazuje opět na první bajt.
		RET		Hotovo.

V případě celého čísla dochází k podobné operaci se znaménkovým bajtem.

3483	INT-CASE	PUSH	DE	Ulož STKEND v DE.
		PUSH	HL	Ušchovej ukazatel na číslo v HL.
		CALL	#2D7F,INT-FETCH	Vyzvedni znaménko do C a číslo do DE.
		POP	HL	Obnov ukazatel na číslo v HL.
		LD	A,B	Vlož #FF pro "abs" a #00 pro "negate".
		OR	C	#FF pro "abs",žádná změna pro "negate".
		CPL		#00 pro "abs" a změněný bajt pro
		LD	C,A	"negate" uschovej v C.
		CALL	#2D8E,INT-STORE	Ušchovej výsledek na zásobníku a
		POP	DE	vrať STKEND do DE.
		RET		Hotovo.

### FUNKCE "SIGNUM" (ZNAMÉNKO)

Doplňák: #29 "sgn"

Tento podprogram provádí funkci SGN X a proto vrací jako poslední hodnotu 1 když X>0, 0 když X=0 a -1 když X<0.

3492	SGN	CALL #34E9,TEST-ZERO	Je-li X nulové
	RET	C	vrať se s poslední hodnotou nula.
	PUSH	DE	Uschovej ukazatel na STKEND.
	LD	DE,#0001	Vlož do DE jedničku.
	INC	HL	Ukazuj na další bajt čísla X.
	RL	(HL)	Rotuj bit 7 do CY a
	DEC	HL	ukazuj na původní adresu.
	SBC	A,A	C bude obsahovat #00 pro kladné X a
	LD	C,A	#FF pro záporné X.
	CALL	#2D8E,INT-STORE	Ulož 1 nebo -1 podle potřeby.
	POP	DE	Obnov ukazatel na STKEND.
	RET		Hotovo.

### FUNKCE "IN"

Doplňák: #2C "in"

Tento podprogram provádí funkci IN X. Proveďte vstup na úrovni procesoru z portu X, jehož hodnota je v BC, a proveďte instrukci IN A,(C).

34A5	IN	CALL #1E99,FIND-INT2	"Poslední hodnota" X <b>kompresována</b> do BC.
		IN A,(C)	Signál je přijat.
		JR #34B0,IN-PK-STK	Skok na uložení výsledku na zásobník kalkulátoru.

### FUNKCE "PEEK"

Doplňák: #2B "peek"

Tento podprogram provádí funkci PEEK X. "Poslední hodnota" je odebrána ze zásobníku za pomoci FIND-INT2 a nahrazena hodnotou uloženou na příslušném paměťovém místě.

34AC	PEEK	CALL #1E99,FIND-INT2	"Poslední hodnota" je <b>kompresována</b> do BC.
		LD A,(BC)	Vyzvedni požadovaný bajt.
34B0	IN-PK-STK	JP #2D28,STACK-A	Výstup přes stack A.

### FUNKCE "USR"

Doplňák: #2D "usr-no"

Tento podprogram ("USR číslo" je něco jiného než "USR řetězec"), obsluhuje funkci USR X, kde X je číslo. Hodnota čísla X je uložena do BC, návratová adresa je uložena na zásobník a program je proveden od adresy X.

34B3	USR-NO	CALL #1E99,FIND-INT2	"Poslední hodnota" je kompresována do BC.
		LD HL,#2D2B	Vlož návratovou adresu podprogramu STACK-BC.
		PUSH HL	Ulož <b>návratovou adresu</b> a
		PUSH BC	proved nepřířímý skok na tento podprogram.
		RET	Hotovo.

Poznámka: Je zajímavé, že **registrový** pár IY je znovu nastaven na svou původní hodnotu po návratu přes STACK-BC, ale důležitý ukazatel na poslední literál, který je v **HL**, obnoven není. Proto pro úspěšný návrat do BASICu musí zrcadlově HL obsahovat adresu instrukce "end-calc" v podprogramu SCANNING, která je na adrese #2758.

## FUNKCE "USR STRING"

Doplňák: #19 "usr-\$"

Podprogram provede funkci USR X\$, kde X\$ je řetězec, a v BC vrací adresu bitové matice znaku uživatelsky definované grafiky, odpovídající znaku uloženému v X\$. Podprogram hlásí chybu A, když X\$ není jediné písmeno v rozsahu od "a" do "u" nebo znak uživatelsky definované grafiky.

34BC	USR-\$	CALL #2BF1,STK-FETCH	Vyzvedni parametry řetězce X\$.
		DEC BC	Sniž délku o jedna pro test.
		LD A,B	Nebyla-li délka
		OR C	jedna,
		JR NZ,#34E7,REPORT-A	skoč a ohlaš chybu.
		LD A,(DE)	Jinak vyzvedni jediné kód řetězce.
		CALL #2C8D,ALPHA	Jestliže představuje písmeno,
		JR C,#34D3,USR-RANGE	skoč na vyzvednutí jeho adresy.
		SUB #90	Sniž adresu na skutečných 0 - 20 a
		JR C,#34E7,REPORT-A	ohlaš chybu A, je-li znak mimo rozsah.
		CP #15	Opět testuj rozsah
		JR NC,#34E7,REPORT-A	a ohlaš chybu A, je-li znak mimo rozsah.
		INC A	Uprav rozsah na 1 - 21, jako rozsah pro "a" až "u".
34D3	USR-RANGE	DEC A	Nyní je rozsah 0-20
		ADD A,A	Vynásobením osmi
		ADD A,A	se získá doplněk
		ADD A,A	pro adresu.
		CP #A8	Testuj rozsah doplňku a
		JR NC,#34E7,REPORT-A	ohlaš chybu A, jeli mimo rozsah.
		LD BC,(#5C7B) (UDG)	Vyzvedni adresu 1. bajtu uživ. definované grafiky do
		ADD A,C	BC a přičti C k doplňku.
		LD C,A	Výsledek vlož zpět do C.
		JR NC,#34E4,USR-STACK	Nedošlo-li k přenosu, skoč,
		INC B	jinak zvětší B.
34E4	USR-STACK	JP #2D2B,STACK-BC	Ulož adresu.
34E7	REPORT-A	RST #08,ERROR-1	Ohlaš:
		DEFB #09	A-Invalid argument.

## PODPROGRAM "TEST-ZERO" (TEST-NULA)

Podprogram je volán přinejmenším devětkrát k otestování FP čísla je-li nulové. Tento test vyžaduje, aby první čtyři bajty čísla byly nulové. Podprogram se vrací s CY=1 bylo-li číslo skutečně nula.

34E9	TEST-ZERO	PUSH HL	Uschovej HL a
		PUSH BC	také BC.
		LD B,A	Uschovej hodnotu A v registru B.
		LD A,(HL)	Vyzvedni 1. bajt.
		INC HL	Ukazuj na 2. bajt.
		OR (HL)	Proveď logické OR.
		INC HL	Ukazuj na 3. bajt.
		OR (HL)	Proveď logické OR.
		INC HL	Ukazuj na 4. bajt.
		OR (HL)	Proveď logické OR.
		LD A,B	Obnov původní hodnotu v registru A,
		POP BC	v registrech BC a
		POP HL	v registrech HL.
		RET NZ	Vrať se s CY=0, pokud byl kterýkoliv ze čtyř bajtů >0.
		SCF	Nastav CY jako signál, že číslo bylo nula a
		RET	vrať se.

### OPERACE "GREATER THAN ZERO" (VĚTŠÍ NEŽ NULA)

Doplňk: #37 "greater-0"

Podprogram vrací jako "poslední hodnotu" číslo 1, byla-li aktuální "poslední hodnota" větší než nula, jinak vrací číslo 0. Je také využíván jinými podprogramy, při skoku "při plus".

34F9	GREATER-0	CALL	#34E9,TEST-ZERO	Je poslední hodnota nula ?
		RET	C	Vrať jestliže ano.
		LD	A,#FF	Proveď LESS THAN ZERO, ale
		JR	#3507,SIGN-TO-C	signalizuj, že je potřeba provést opačnou akci.

### FUNKCE "NOT"

Doplňk: #30 "not"

Podprogram vrací jako "poslední hodnotu" číslo 1 jestliže byla aktuální "poslední hodnota" nulová, jinak vrací číslo nula. Je také využíván jinými podprogramy pro "skok při nule".

3501	NOT	CALL	#34E9,TEST-ZERO	CY bude 1 jedině když poslední hodnota byla 0, což
		JR	#350B,FP-0/1	už je správný výsledek.

### OPERACE "LESS THAN ZERO" (MENŠÍ NEŽ NULA)

Doplňk: #36 "less-0"

Podprogram vrací jako "poslední hodnotu" číslo 1, byla-li aktuální "poslední hodnota" menší než nula, jinak vrací číslo 0. Je také využíván jinými podprogramy při skoku "při mfnus".

3506	LESS-0	XOR	A	Vynuluj registr A.
3507	SIGN-TO-C	INC	HL	Ukazuj na znaménkový bajt.
		XOR	(HL)	CY=0 pro kladná čísla a CY=1 pro záporná čísla.
		DEC	HL	
		RLCA		Při vstupu z GREATER-0 jde do CY opačné znaménko.

### PODPROGRAM "ZERO OR ONE" (NULA NEBO JEDNA)

Tento podprogram nastaví poslední hodnotu na nulu jestliže CY=0 nebo na jedničku jestliže CY=1. Při zavolání z "E-TO-FP" vytváří tuto nulu nebo jedničku v mem-0.

350B	FP-0/1	PUSH	HL	Uchovej ukazatel na výsledek.
		LD	A,#00	Vynuluj A bez porušení CY.
		LD	(HL),A	Nastav 1.bajt na 0.
		INC	HL	Ukazuj na 2. bajt.
		LD	(HL),A	Nastav 2.bajt na 0.
		INC	HL	Ukazuj na 3. bajt.
		RLA		Rotuj CY do A, takže A bude 1, bylo-li CY=1, jinak A=0.
		LD	(HL),A	Nastav třetí bajt na tuto hodnotu a
		RRA		zajisti opět nulové A.
		INC	HL	Ukazuj na 4. bajt.
		LD	(HL),A	Nastav jej na nulu.
		INC	HL	Ukazuj na 5. bajt.
		LD	(HL),A	Nastav jej na nulu.
		POP	HL	Obnov ukazatel výsledku.
		RET		Hotovo.

### OPERACE "OR"

Doplňk: #07 "or"

Podprogram provádí binární operaci "X OR Y" a vrací X, jestliže Y bylo nulové, a jinak hodnotu nula.

351B	OR	EX	DE,HL	HL ukazuje na Y, tedy druhé číslo.
		CALL	#34E9,TEST-ZERO	Testuj jestli je Y nulové.
		EX	DE,HL	Obnov ukazatele.
		RET	C	Jestliže Y=0, vrať se; X je nyní poslední hodnotou.
		SCF		Nastav CY a
		JR	#350B,FP-0/1	nastav "poslední hodnotu" na jedna.

### OPERACE "NUMBER AND NUMBER"

Doplňk: #08 "no-&-no"

Podprogram provádí binární operaci "X AND Y" a vrací X jestliže Y není nula, v ostatních případech hodnotu nula.

3524	NO-&-NO	EX	DE,HL	HL ukazuje na Y, tedy druhé číslo.
		CALL	#34E9,TEST-ZERO	Testuj, zda Y je nulové.
		EX	DE,HL	Zaměň ukazatele zpět.
		RET	NO	Vrať se s X jako poslední hodnotou, nebylo-li Y nula.
		AND	A	Nuluj CY a
		JR	#350B,FP-0/1	nastav poslední hodnotu na nulu.

### OPERACE "STRING AND NUMBER"

Doplňk: #10 "str-&-no"

Podprogram provádí binární operaci "X\$ AND Y" a vrací X\$ jestliže Y je nenulové, jinak vrací nulový řetězec.

352D	STR-&-NO	EX	DE,HL	HL ukazuje na Y, DE na X\$.
		CALL	#34E9,TEST-ZERO	Testuj, zda Y je nulové.
		EX	DE,HL	Zaměň ukazatele zpět.
		RET	NC	Vrať se s X\$ jako poslední hodnotou, když Y nebylo nula.
		PUSH	DE	Ušchovej ukazatel na číslo.
		DEC	DE	Ukazuj na 5. bajt param. řetěz. = vyšší bajt délky.
		XOR	A	Nuluj registr A.
		LD	(DE),A	Vyšší bajt délky je nyní nastaven na 0.
		DEC	DE	Ukazuj na nižší bajt délky.
		LD	(DE),A	Také jej nastav na nulu.
		POP	DE	Obnov ukazatele.
		RET		Vrať se s parametry řetězce jako "poslední hodnotou".

### OPERACE "COMPARISON"

Doplňky: #09 až #0E a #11 až #16 "no-l-eq", "no-gr-eq", "nos-neq", "no-grtr", "no-less", "nos-eq", "str-l-eq", "str-gr-eq", "strs-neq", "str-grtr", "str-less" a "strs-eq".

3520	NO-L-EQL	LD	A,B	Doplňk do registru A.
	etc.	SUB	#08	Rozsah je #01 - #06 a #09 - #0E.
		BIT	2,A	Tento rozsah je změněn na #00 - #02,#04 - #06,#08 - #0A
		JR	NZ,#3543,EX-OR-NOT	a #0C - #0E.
		DEC	A	
3543	EX-OR-NOT	RRCA		Dále je snižen na #00 - #07 a CY=1 pro ">=" a "<".
		JR	NC,#354E,NU-OR-STR	Operace s CY=1

	PUSH AF	jsou považovány za své komplementární operace
	PUSH HL	
	CALL #343C,EXCHANGE	jakmile jsou hodnoty zaměněny.
	POP DE	
	EX DE,HL	
	POP AF	
354E NU-OR-STR	BIT 2,A	Číselná porovnání jsou oddělena od řetězcových porovnání otestováním bitu 2.
	JR NZ,#3559,STRINGS	
	RRCA	Číselné <b>oper.</b> mají rozsah #00-#01 s CY=1 pro "=" a "<>".
	PUSH AF	Ušchovej doplněk.
	CALL #300F,SUBTRACT	Čísla jsou odečtena
	JR #358C,END-TESTS	pro závěrečné testy.
3559 STRINGS	RRCA	<b>Řetězc. porov.</b> mají rozsah #02-#03 s CY=1 pro "=" a "<>".
	PUSH AF	Ušchovej doplněk.
	CALL #2BF1,STK-FETCH	Délky a počáteční adresy
	PUSH DE	řetězců jsou vyzvednuty ze zásobníku kalkulátoru.
	PUSH BC	
	CALL #2BF1,STK-FETCH	
	POP HL	Délka 2. řetězce.
3564 BYTE-COMP	LD A,H	
	OR L	
	EX (SP),HL	
	LD A,B	
	JR NZ,#3575,SEC-PLUS	Skoč pokud druhý řetězec není nulový.
	OR C	
356B SECND-LOW	POP BC	Zde je 2. řetězec buď nulový nebo menší než 1. řetězec.
	JR Z,#3572,BOTH-NULL	
	POP AF	
	CCF	CY je komplementováno
	JR #3588,STR-TEST	k dosažení správného výsledku testu.
3572 BOTH-NULL	POP AF	Zde je CY použito tak, jak je.
	JR #3588,STR-TEST	
3575 SEC-PLUS	OR C	První řetězec je nyní nulový, 2. ne.
	JR Z,#3585,FRST-LESS	Ani jeden řetězec není nulový
	LD A,(DE)	takže mohou být porovnány jejich další bajty.
	SUB (HL)	
	JR C,#3585,FRST-LESS	První bajt je menší.
	JR NZ,#356B,SECND-LOW	Druhý bajt je menší.
	DEC BC	Bajty se rovnají,
	INC DE	takže délky jsou dekrementovány a
	INC HL	
	EX (SP),HL	
	DEC HL	
	JR #3564,BYTE-COMP	provede se skok na BYTE-COMP k porovnání dalších bajtů.
3585 FRST-LESS	POP BC	
	POP AF	
	AND A	CY je vynulováno pro dosažení správných výsledků testu.
3588 STR-TEST	PUSH AF	Pro řetězcové testy je na <b>Zás.</b> kalkulátoru uložena 0.
	RST #28,FP-CALC	
	DEFB #A0,stk-nula	
	DEFB #38,konec výpočtu	
358C END-TESTS	POP AF	Tyto tři testy, dávají správné výsledky
	PUSH AF	pro všech dvanáct porovnání.
	CALL C,#3501,NOT	
	POP AF	Počáteční CY je =1 pro "<>" a "=" a výsledné
	PUSH AF	
	CALL NC,#34F9,GREATER-0	
	POP AF	

RRCA		CY je nastaveno pro ">", "<" a "=".
CALL	NC,#3501,NOT	
RET		Hotovo.

### OPERACE "STRING CONCATENATION"

Doplňk: #17 "strs-add"

Tento podprogram provádí binární operaci A\$+B\$. Parametry obou řetězců jsou vyzvednuty a je vypočtena celková délka. V pracovním prostoru je vytvořen dostatečný prostor pro oba řetězce, které jsou pak do něj překopírovány. Výsledkem tohoto podprogramu je proto vytvoření přechodné proměnné A\$+B\$, která se nachází v pracovním prostoru.

359C	STRS-ADD	CALL #2BF1,STK-FETCH	Parametry druhého řetězce jsou vyzvednuty a uschovány.
		PUSH DE	
		PUSH BC	
		CALL #2BF1,STK-FETCH	Parametry prvního řetězce jsou vyzvednuty.
		POP HL	Délky jsou nyní v registrových párech HL a BC.
		PUSH HL	
		PUSH DE	Parametry 1. řetězce jsou uschovány.
		PUSH BC	
		ADD HL,BC	Je vypočtena celková délka obou řetězců a
		LD B,H	
		LD C,L	uložena do BC.
		RST #30,BC-SPACES	Je vytvořen pracovní prostor.
		CALL #2AB2,STK-STO-\$	Param. nového řetězce jsou předány na zásobník kalkul.
		POP BC	Jsou obnoveny parametry 1. řetězce a
		POP HL	
		LD A,B	
		OR C	
		JR Z,#35B7,OTHER-STR	pokud není nulový,
		LDIR	je okopírován do pracovního prostoru.
35B7	OTHER-STR	POP BC	Identická procedura následuje pro druhý řetězec,
		POP HL	čímž se dosáhne
		LD A,B	výsledku A\$+B\$.
		OR C	
		JR Z,#35BF,STK-PNTRS	
		LDIR	

### PODPROGRAM "STK-PNTRS"

Podprogram vrací HL jako ukazatel na první bajt poslední hodnoty, tedy STKEND-5 a DE ukazuje jedno místo za poslední hodnotu, tedy na STKEND.

35BF	STK-PNTRS	LD HL,(#5C65) (STKEND)	Vyzvedni aktuální hodnotu STKEND.
		LD DE,#FFFB	Nastav DE na -5.
		PUSH HL	Uschovej STKEND.
		ADD HL,DE	Vypočti STKEND - 5.
		POP DE	Vyzvedni STKEND do DE.
		RET	Vrať se.

### FUNKCE "CHR\$"

Doplňk #2F: "chrs"

Podprogram zpracuje funkci CHR\$ X a vytvoří příslušný znak v pracovním prostoru.

35C9	CHRS	CALL #2DD5,FP-TO-A	Poslední hodnota do A.
		JR C,#35DC,REPORT-B	Je-li x>255, ohlaš chybu.
		JR NZ,#35DC,REPORT-B	Je-li x záporné číslo, ohlaš chybu.

	PUSH AF	Uschovej hodnotu x	
	LD BC,#0001	a vytvoř jedno místo	
	RST #30,BC-SPACES	v pracovním prostoru.	
	POP AF	Vyzvedni x	
	LD (DE),A	a vlož jej do místa v pracovním prostoru.	
	CALL #2AB2,STK-STO-3	ULOŽ parametry tohoto řetězce na zásobník kalkulátoru.	
	EX DE,HL	Zaměň ukazatele	
	RET	a vrať se.	
35DC	REPORT-B	RST #08,ERROR-1	Ohlaš:
		DEFB #0A	B-Integer out of range

#### FUNKCE "VAL" A "VAL\$"

Doplňěk #1D: "val".                      Doplněk #18: "val\$".

Podprogram provádí funkce VAL X\$ a VAL\$ X\$. Při obsluze VAL X\$ se provede číselné ohodnocení výrazu a výsledek se uloží jako poslední hodnota na zásobník kalkulátoru. Při obsluze VAL\$ X\$ se provede řetězcové ohodnocení výrazu a parametry řetězce se uloží jako poslední hodnota na zásobník kalkulátoru. Doplněk vstupuje v registru B.

35DE	VAL	LD HL,(#5C5D) (CH-ADD)	Aktuální hodnota CH-ADD
		PUSH HL	je uschována na zásobníku.
		LD A,B	Doplňěk do A.
		ADD A,#E3	Vytvoř #00 a CY=1 pro "val" nebo #FB a CY=0 pro "val\$".
		SBC A,A	Vytvoř #FF (bit 6=1) pro "val" nebo #00 (bit 6=0) "val\$"
		PUSH AF	Uschovej <b>vla jku</b> .
		CALL #2BF1,STK-FETCH	Vyzvedni parametry řetězce,
		PUSH DE	uschovej jeho začátek,
		INC BC	zvětši délku
		RST #30,BC-SPACES	a vytvoř místo v pracovním prostoru.
		POP HL	Obnov začátek řetězce v HL.
		LD (#5C5D),DE (CH-ADD)	Ukazatel na první nově vytvořené místo jde do CH-ADD
		PUSH DE	a na zásobník.
		LDIR	Kopíruj řetězec do vytvořeného prostoru.
		EX DE,HL	Zaměň ukazatele.
		DEC HL	Bajt "navíc"
		LD (HL),#0D	je nahrazen znakem "CR" (ENTER).
		RES 7,(IY+1) (FLAGS)	Nuluj <b>vla jku</b> "syntax/run"
		CALL #24FB,SCANNING	a proved kontrolu správnosti syntaxe.
		RST #18,GET-CHAR	Vyzvedni znak za řetězcem.
		CP #0D	Pokud to není znak "CR",
		JR NZ,#360C,V-RPORT-C	ohlaš chybu.
		POP HL	Obnov začátek řetězce v HL.
		POP AF	Vyzvedni <b>vla jku</b> "val/val\$"
		XOR (IY+1) (FLAGS)	a testuj bit 6 oproti
		AND #40	výsledku kontroly syntaxe.
360C	V-RPORT-C	JP NZ,#1C8A,REPORT-C	Ohlaš chybu, pokud se neshodují.
		LD (#5C5D),HL (CH-ADD)	Začátek řetězce do CH-ADD.
		SET 7,(IY+1) (FLAGS)	Signál: program v běhu.
		CALL #24FB,SCANNING	Řetěz je <b>zprac.</b> jako "další výraz" a jeho <b>hod.</b> uložena.
		POP HL	Vyzvedni
		LD (#5C5D),HL (CH-ADD)	a obnov původní hodnotu v CH-ADD.
		JR #35BF,STK-PNTRS	Vrať se přes STK-PNTRS, což nastaví systémové ukazatele.



## FUNKCE "STR\$"

Doplňk: #2E "str\$"

Podprogram provede funkci STR\$ X a vrací parametry řetězce, obsahujícího ten samý text, který by se objevil na obrazovce při provedení příkazu PRINT X.

361F STR\$	LD BC,#0001	Vytvoř jedno místo
	RST #30,BC-SPACES	v pracovního prostoru,
	LD (#5C5B),HL (K-CUR)	kopíruj jeho adresu do K-CUR
	PUSH HL	a také ji ulož na zásobník.
	LD HL,(#5C51) (CURCHL)	Aktuální kanálová adresa
	PUSH HL	je uložena na zásobník.
	LD A,#FF	Otevři kanál "R", což umožní
	CALL #1601,CHAN-OPEN	výstup řetězce do pracovní oblasti.
	CALL #2DE3,PRINT-FP	Poslední hodnota X, je vytištěna do pracovního prostoru.
	POP HL	Obnov adresu aktuálního kanálu do HL a
	CALL #1615,CHAN-FLAG	obnov také příznaky, které mu přináležejí.
	POP DE	Obnov počáteční adresu řetězce.
	LD HL,(#5C5B) (K-CUR)	Adresa kurzoru ukazuje jedno místo za řetězec
	AND A	a proto
	SBC HL,DE	rozdíl ukazatelů je délkou řetězce,
	LD B,H	kteřá je
	LD C,L	převedená do BC.
	CALL #2AB2,STK-STO-\$	Parametry řetězce jsou uloženy na zásobník kalkulátoru.
	EXX DE,HL	Zaměň ukazatele
	RET	a vrať se.

## PODPROGRAM "READ-IN"

Doplňk: #1A "read-in"

Tento podprogram je volán přes kalkulátor pomocí doplňku z prvního řádku podprogramu INKEY\$. Umožňuje načítání dat přes různé proudy, které jsou ve standardním SPECTRU. Stejně jako INKEY\$, vrací i tento podprogram parametry řetězce.

3645 READ-IN	CALL #1E94,FIND-INT1	Kompresuj číselný parametr do A
	CP #10	a jestliže není menší než 16,
	JP NC,#1E9F,REPORT-B	ohlaš chybu.
	LD HL,(#5C51) (CURCHL)	Aktuální kanálová adresa
	PUSH HL	je uložena na zásobník.
	CALL #1601,CHAN-OPEN	Otevři kanál specifikovaný parametrem
	CALL #15E6,INPUT-AD	a přijmi jakoby "hodnotu klávesy".
	LD BC,#0000	Délka řetězce při "neplnění" je 0.
	JR NC,#365F,R-I-STORE	Skoč, jestliže nebyl žádný signál.
	INC C	Nastav délku na 1,
	RST #30,BC-SPACES	vytvoř tento prostor,
	LD (DE),A	a vlož do něj řetězec.
365F R-I-STORE	CALL #2AB2,STK-STO-\$	Předej parametry řetězce na zásobník kalkulátoru.
	POP HL	Vyzvedni kanálovou adresu
	CALL #1615,CHAN-FLAG	a obnov příslušné vlajky.
	JP #35BF,STK-PNTRS	Vrať se přes STK-PNTRS, což nastaví systémové ukazatele.

## FUNKCE "CODE"

Doplňk: #1C "CODE"

Podprogram provede funkci CODE A\$ a vrací kód prvního znaku z řetězce A\$ nebo 0, pokud byl řetězec A\$ nulový.

3669 CODE	CALL #2BF1,STK-FETCH	Vyzvedni parametry řetězce.
	LD A,B	Testuj délku

	OR	C	a pokud je nulové,
	JR	Z,#3671,STK-CODE	skoč dopředu.
	LD	A,(DE)	Vyzvedni kód prvního znaku
3671	STK-CODE	JP	#2D2B,STACK-A a před návratem jej uloží na zásobník kalkulátoru.

#### FUNKCE "LEN"

Doplňk: #1E "len"

Podprogram provede funkci LEN A\$ a vrací délku řetězce jako poslední hodnotu na zásobníku kalkulátoru.

3674	LEN	CALL	#2BF1,STK-FETCH	Vyzvedni parametry řetězce
		JP	#2D2B,STACK-BC	a proveď návrat přes STACK-BC, který uloží <b>posl.</b> hodnotu.

#### PODPROGRAM "DECREASE THE COUNTER"

Doplňk: #35 "dec-jr-nz"

Podprogram je využíván pouze podprogramem SERIES GENERATOR a ve skutečnosti provádí instrukci DJNZ, ale čítačem je proměnná BREG a ne registr **B**.

367A	DEC-JR-NZ	EXX	Přepni na zrcadlové registry
		PUSH	HL a uschovej ukazatel na další literál.
		LD	HL,#5C67 HL ukazuje na BREG.
		DEC	(HL) Dekrementace BREG.
		POP	HL Obnovení ukazatele na další literál.
		JR	NZ,#3687,JUMP-2 Skoč při nenulovém BREG.
		INC	HL Překroč další literál.
		EXX	Přepni na hlavní registry
		RET	a vrať se.

#### PODPROGRAM "JUMP"

Doplňk: #33 "jump"

Podprogram provede nepodmíněný skok při zavolání literálem #33. Je také používán podprogramy DECREASE THE COUNTER a JUMP ON TRUE.

3686	JUMP	EXX	Přepni na zrcadlové registry.
3687	JUMP-2	LD	E,(HL) Vyzvedni délku skoku do registru <b>E</b> .
		LD	A,E V registru A se vytvoří
		RLA	hodnota #00 pro kladná čísla nebo
		SBC	A,A #FF pro záporná čísla
		LD	D,A a je převedena do <b>D</b> .
		ADD	HL,DE H'L' obsahuje cílovou adresu.
		EXX	Přepni na hlavní registry.
		RET	Hotovo.

#### PODPROGRAM "JUMP ON TRUE"

Doplňk: #00 "jump-true"

Podprogram provede podmíněný skok, jestliže "poslední hodnota" na zásobníku kalkulátoru je pravdivá. Její adresa je v DE.

368F	JUMP-TRUE	INC	DE Posuň ukazatel na třetí bajt čísla,
		INC	DE který je buď nula nebo jedna
		LD	A,(DE) a vyzvedni je do A.
		DEC	DE Opět ukazuj
		DEC	DE na první bajt.

AND	A	Je třetí bajt nula?
JR	NZ,#3686,JUMP	Proveď skok, jestliže není nulový (=pravda).
EXX		Přepni na zrcadlové registry.
INC	HL	Překroč literál udávající délku skoku.
EXX		Přepni na hlavní registry.
RET		Hotovo.

#### PODPROGRAM "END-CALC"

Doplňk: #38 "end-calc"

Podprogram ukončí operaci RST #28.

369B	END-CALC	POP	AF	Zahoď adresu zpětných návratů do kalkulátoru.
		EXX		Přepni na zrcadlové registry
		EX	(SP).HL	a ulož adresu z HL na zásobník.
		EXX		Přepni na hlavní registry.
		RET		Skoč na dřívější adresu dodanou z HL.

#### PODPROGRAM "MODULUS"

Doplňk: #32 "n-mod-m"

Podprogram počítá  $N \pmod{M}$ , kde M je kladné celé číslo na vrcholu zásobníku kalkulátoru a N je celé číslo pod vrcholem zásobníku. Podprogram vrací celočíselnou část podílu  $\text{INT}(N/M)$  na vrcholu zásobníku (tedy jako poslední hodnotu) a pod vrchol zásobníku uloží zbytek  $N - \text{INT}(N/M)$ . Podprogram se používá při výpočtu náhodných čísel ke snížení  $N \pmod{65537}$ .

36A0	N-MOD-M	RST	#28,FP-CALC	N,M
		DEFB	#C0,st-mem-0	N,M (M do mem-0)
		DEFB	#02,vymaz	N
		DEFB	#31,zdvojení	N,N
		DEFB	#E0,get-mem-0	N,N,M
		DEFB	#05,dělení	N,N/M
		DEFB	#27,int	N,INT(N/M)
		DEFB	#E0,get-mem-0	N,INT(N/M),M
		DEFB	#01,záměna	N,M,INT(N/M)
		DEFB	#C0,st-mem-0	INT(N/M) do mem-0.
		DEFB	#04,násobení	N,M*INT(N/M)
		DEFB	#03,odečítání	N-M*INT(N/M)
		DEFB	#E0,get-mem-0	N-M*INT(N/M),INT(N/M)
		DEFB	#38,konec výpočtu	
		RET		Hotovo.

#### FUNKCE "INT"

Doplňk: #27 "int"

Podprogram vrací jako poslední hodnotu celočíselnou část čísla, které je při vstupu na vrcholu zásobníku kalkulátoru. Tedy  $\text{INT } 2.4$  dává 2, ale protože podprogram zaokrouhuje směrem dolů, bude  $\text{INT } -2.4$  roven -3.

36AF	INT	RST	#28,FP-CALC	X
		DEFB	#31,zdvojení	X,X
		DEFB	#36,<0	X,(1/0) (logická hodnota)
		DEFB	#00,skok-pravda	X
		DEFB	#04,na #36B7,X-NEG	X

Pro případ, že  $X > 0$ , se nyní nalezne celočíselná část X.

```

DEFB #3A,truncate      I(X)
DEFB #38,konec výpočtu
RET                    Hotovo.

```

Pro záporné celé číslo se vrací hodnota I(X), pro ostatní záporná čísla se vrací I(X)-1.

```

36B7 X-NEG      DEFB #31,zdvojení      X,X
                DEFB #3A,truncate      X,I(X)
                DEFB #C0,st-mem-0      I(X) jde do mem-0.
                DEFB #03,odečítání      X-I(X)
                DEFB #E0,get-mem-0      X-I(X),I(X)
                DEFB #01,záměna         I(X),X-I(X)
                DEFB #30,not            I(X),(1/0) (logická hodnota).
                DEFB #00,skok-pravda    I(X)
                DEFB #03,na #36C2,EXIT  I(X)

```

Pro záporná celá čísla se provede přeskok.

```

                DEFB #A1,stk-jedna      I(X),1
                DEFB #03,odečítání      I(X)-1
36C2 EXIT      DEFB #38,konec výpočtu    I(X) nebo I(X)-1
                RET

```

## FUNKCE "EXPONENTIAL"

Doplňák: #26 "exp"

Podprogram provádí funkci EXP X a je prvním ze čtyř podprogramů, které používají SERIES GENERATOR k vytváření Čebyševových polynomů. Aproximace INT X je nalezena takto:

- X je děleno LN 2 k získání Y, takže 2 na Y-tou dává požadovaný výsledek.
- Je nalezena hodnota N pro kterou platí, že  $N \approx \text{INT } Y$ .
- Je nalezena hodnota W pro kterou platí, že  $W = Y - N$ , kde  $0 < W < 1$ .
- Je vytvořen argument Z takový, že  $Z = 2 * W - 1$ .
- Použitím SERIES GENERATOR se získá hodnota  $2^W$ .
- Konečně se přičte N k exponentu, což dává  $2^{(N+W)}$ , neboli  $2^Y$  a tedy požadovaný výsledek **EXP** X.

```

36C4 EXP      RST #28,FP-CALC      X
                DEFB #3D,re-stack      X (X ve formě FP).
                DEFB #34,stk-data      X,1/LN2
                DEFB #F1,exponent #81
                DEFB #38,AA,#3B,#29
                DEFB #04,násobení      X/LN2 (dále jen Y).
                DEFB #31,zdvojení      Y,Y
                DEFB #27,int 1C46      Y,INT Y (dále N).
                DEFB #C3,st-mem-3      N do mem-3.
                DEFB #03,odečítání      Y-N (dále jen W).
                DEFB #31,zdvojení      W,W
                DEFB #0F,sčítání      2*W
                DEFB #A1,stk-jedna      2*W,1
                DEFB #03,odečítání      2*W-1 (dále jen Z).
                DEFB #88,serie-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,#B0,#B0,#14
7.  DEFB #EE,exponent #7E
    DEFB #7E,#BB,#94,#58
8.  DEFB #F1,exponent #81
    DEFB #3A,#7E,#F8,#CF      2^W
    DEFB #E3,get-mem-3        2^W,N
    DEFB #38,konec výpočtu
CALL #2DD5,FP-T0-A
JR   NZ,#3705,N-NEGTV
JR   C,#3703,REPORT-6
ADD  A,(HL)
JR   NC,#370C,RESULT-OK
3703 REPORT-6 RST #08,ERROR-1
DEFB #05
3705 N-NEGTV JR   C,#370E,RSLT-ZERO
SUB  (HL)
JR   NC,#370E,RSLT-ZERO
NEG  NEG
370C RESULT-OK LD  (HL),A
RET  RET
370E RSLT-ZERO RST #28,FP-CALC
DEFB #02,výmaz
DEFB #A0,stk-nula
DEFB #38,konec výpočtu
RET  RET

```

Do registru A je vložena absolutní hodnota N mod 256.  
Skoč dopředu při záporném N.  
Ohlaš chybu, je-li ABS N > 255.  
Přičti ABS N k exponentu.  
Skoč, je-li e>255.  
Ohlaš:  
6-Number too big.  
Výsledek má být 0, jestliže n < -255.  
Odečti ABS N od exponentu, protože N bylo záporné.  
Výsledek bude nula pro e < 0.  
Mínus e je změněno na plus e.  
Vlož exponent e.  
Hotovo.  
Použij kalkulátor  
a vlož  
exponent e = 0.  
Hotovo.

### FUNKCE PŘIROZENÉHO LOGARITMU

Doplňk: #25 "ln"

Tento podprogram provádí funkci LN x a je druhým ze čtyř podprogramů, které používají podprogram SERIES GENERATOR k vytváření Čebyševových polynomů.

Aproximace LN x je nalezena následovně:

- Testuje se x a je ohlášena chyba A, pokud x není kladné.
- Dále je x rozděleno na exponent  $e^1$  a mantisu  $x^1 = x / (2^e)$ , kde  $x^1 \geq .5$  ale  $x^1 < 1$ .
- Jsou formovány požadované hodnoty y1 nebo y2. Je-li  $x^1 > .8$ , pak  $y1 = e^1 * LN2$ , jinak  $y2 = (e^1 - 1) * LN2$ .
- Jestliže  $x^1 > .8$ , pak je na zásobník uložena hodnota  $x^1 - 1$ , jinak  $2 * x^1 - 1$ .
- Jestliže  $x^1 > .8$ , pak bude argument  $z = 2.5 * x^1 - 3$ , jinak  $z = 5 * x^1 - 3$ . Vždy pak musí platit že  $-1 < z < 1$  (což je podmínka pro konvergenci řady).
- K vytvoření požadované funkce se použije podprogram "SERIES GENERATOR".
- Nakonec se užije obvyklé násobení a sčítání a LN x se vrací jako poslední hodnota na zásobníku kalkulátoru.

```
3713 LN      RST #28,FP-CALC      x
```

Proveď krok a).

```

DEFB #3D,re-stack      x (v plné FP formě)
DEFB #31,zdvojení     x,x
DEFB #37,>0           x,(1/0) (logická hodnota).
DEFB #00,skok-pravda  x
DEFB #04,na 371C,VALID x
DEFB #38,konec výpočtu

```

371A REPORT-A RST #08,ERROR-1 Ohlaš:  
 DEFB #09 A-Invalid argument

Proveď krok b).

371C VALID DEFB #A0,stk-nula x,0  
 DEFB #02,výmaz x  
 DEFB #38,konec výpočtu x  
 LD A,(HL) Exponent e jde do registru A.  
 LD (HL),#80 x je sníženo na x'.  
 CALL #2D28,STACK-A Na zásobníku je:x',e.  
 RST #28,FP-CALC x',e  
 DEFB #34,stk-data x',e,128 (dekadicky)  
 DEFB #38,exponent #88 x',e  
 DEFB #00,(#00,#00,#00)  
 DEFB #03,odečítání x',e

Proveď krok c).

	DEFB #01,záměna	e',x'	
	DEFB #31,zdvojení	e',x',x'	
	DEFB #34,stk-data	e',x',x',0.8	
	DEFB #F0,exponent #80		
	DEFB #4C,#CC,#CC,#CD		
	DEFB #03,odečítání	e',x',x'-0.8	
	DEFB #37,>0	e',x',(1/0)	
	DEFB #00,skok-pravda	e',x'	
	DEFB #08,na 373D,GRE.8	e',x'	
	DEFB #01,záměna	x',e'	
	DEFB #A1,stk-jedna	x',e',1	
	DEFB #03,odečítání	x',e'-1	
	DEFB #01,záměna	e'-1,x'	
	DEFB #38,konec výpočtu	e'-1,x'	
	INC (HL)	e'-1,2*x'	
	RST #28,FP-CALC	postup pro x' větší	postup pro x' menší
373D GRE.8	DEFB #01,záměna	x',e'	2*x',e'-1
	DEFB #34,stk-data	x',e',LN2	2*x',e'-1,LN2
	DEFB #F0,exponent #80		
	DEFB #31,#72,#17,#F8		
	DEFB #04,násobení	x',e'*LN2 (+y1)	2*x',(e'-1)*LN2 (+y2)

Proveď krok d).

DEFB #01,záměna	y1,x'	y2,2*x'
DEFB #A2,stk-polovina	y1,x',.5	y2,2*x',.5
DEFB #03,odečítání	y1,x'-.5	y2,2*x'-.5
DEFB #A2,stk-polovina	y1,x'-.5,.5	y2,2*x'-.5,.5
DEFB #03,odečítání	y1,x'-1	y2,2*x'-1

Proveď krok e).

DEFB #31,zdvojení	y1,x'-1,x'-1	y2,2*x'-1,2*x'-1
DEFB #34,stk-data	y1,x'-1,x'-1,2.5	y2,2*x'-1,2*x'-1,2.5
DEFB #32,exponent #82		
DEFB #20,(#00,#00,#00)		
DEFB #04,násobení	y1,x'-1,2.5*x'-2.5	y2,2*x'-1,5*x'-2.5
DEFB #A2,stk-polovina	y1,x'-1,2.5*x'-2.5,.5	y2,2*x'-1,5*x'-2.5,.5
DEFB #03,odečítání	y1,x'-1,2.5*x'-3 (+z)	y2,2*x'-1,5*x'-3 (=z)

**Proveď krok f).**

	DEFB #8C,série-0C	y1,x <sup>-1</sup> ,z	y2,2*x <sup>-1</sup> ,z
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		
	DEFB #90,#AA,(#00,#00)		
6.	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		

Nyní je poslední hodnota:  $\text{LN}x'/(x'-1)$   $\text{LN}(2*x')/(2*x'-1)$

**Proveď krok g).**

DEFB #04,násobení	$\text{LN}(2^*e')$ , $\text{LN}x'$	$\text{LN}(2^*(e'-1))$ , $\text{LN}(2*x')$
DEFB #0F,sčítání	$\text{LN}((2^*e')*x')$	$\text{LN}(2^*(e'-1)*2*x')$
DEFB #38,konec výpočtu	LN x	
RET	Poslední hodnota je LN x.	

**PODPROGRAM "REDUKCE ARGUMENTU"**

Doplňek #39: "get-argt"

Podprogram převede argument funkce SIN nebo COS na hodnotu v. Program nejprve nalezne hodnotu y, pro které platí:  $y=x/(2*PI)-\text{INT}(x/2*PI)+.5$ , kde  $.5 > y >= -.5$ .

Podprogram vrací:

- a)  $v=4*y$  pokud  $-1 <= 4*y <= 1$
  - b)  $v=2$  až  $4*y$  pokud  $1 < 4*y < 2$
  - c)  $v=4*y-2$  pokud  $-2 <= 4*y < -1$
- Ve všech případech  $-1 <= v <= 1$  a  $\text{SIN}(PI* v/2)=\text{SIN} x$ .

3783	<b>GET-ARGT</b>	RST #28,FP-CALC	x
		DEFB #3D,re-stack	x ( <b>v plné FP formě</b> )
		DEFB #34,stk-data	x,1/(2*PI)
		DEFB #EE,exponent #7E	
		DEFB #22,#F9,#83,#6E	
		DEFB #04,násobení	x/(2*PI)
		DEFB #31,zdvojení	x/(2*PI),x/(2*PI)
		DEFB #A2,stk-polovina	x/(2*PI),x/(2*PI),.5
		DEFB #0F,sčítání	x/(2*PI), <b>x/(2*PI)+.5</b>

```

DEFB #27,int 1C46          x/(2*PI) ,INT(x/(2*PI)+.5)
DEFB #03,odečítání       x/(2*PI)-INT(x/(2*PI)+.5)*y

```

Poznámka: přičtení 0.5 a provedení INT zaokrouhlí výsledek na nejbližší celé číslo.

```

DEFB #31,zdvojení       y,y
DEFB #0F,sčítání        2*y
DEFB #31,zdvojení       2*y,2*y
DEFB #0F,sčítání        4*y
DEFB #31,zdvojení       4*y,4*y
DEFB #2A,abs            4*y,ABS(4*y)
DEFB #A1,stk-jedna      4*y,ABS(4*y),1
DEFB #03,odečítání      4*y,ABS(4*y)-1=z
DEFB #31,zdvojení       4*y,z,z
DEFB #37,>0             4*y,z,(1/0)
DEFB #C0,st-mem-0       Mem-0 obsahuje výsledek testu.
DEFB #00,skok-pravda    4*y,z
DEFB #04,na #37A1,ZPLUS 4*y,z
DEFB #02,výmaz          4*y
DEFB #38,konec výpočtu  4*y=v (ad a)
RET

```

Po provedení skoku pokračuj zde.

```

37A1 ZPLUS  DEFB #A1,stk-jedna  4*y,z,1
              DEFB #03,odečítání  4*x,z-1
              DEFB #01,záměna     z-1,4*y
              DEFB #36,<0         z-1,(1/0)
              DEFB #00,skok-pravda z-1
              DEFB #02,na #37A8,YNEG z-1
              DEFB #1B,negace     1-z
37A8 YNEG   DEFB #38,konec výpočtu 1-z=v (ad b)
              RET                 z-1=v (ad c)

```

## FUNKCE "COS"

Doplňěk #20: "cos"

Podprogram vrací poslední hodnotu jako aproximaci COS x. Používá výraz: COS x-SIN (PI\*W/2), kde  $-1 \leq W \leq 1$ . Při derivaci W podle x používá podprogram výsledek testu z předchozího podprogramu. Tento je uložen v mem-0. Podprogram pokračuje skokem do podprogramu SINE v bodě C-ENT a vrací poslední hodnotu COS x.

```

37AA COS    RST #28,FP-CALC      x
              DEFB #39,get-arg1  v
              DEFB #2A,abs       ABS v
              DEFB #A1,stk-jedna  ABS v,1
              DEFB #03,odečítání  ABS v-1
              DEFB #E0,get-mem-0  ABS v-1,(1/0)
              DEFB #00,skok-pravda ABS v-1
              DEFB #06,na #37B7,C-ENT  ABS v-1 =W

```

Pokud nedošlo ke skoku, pokračuj zde.

```

DEFB #1B,negace  1-ABS v
DEFB #33,skok   1-ABS v
DEFB #03,na #37B7,C-ENT  1-ABS v =W

```



## FUNKCE "SINUS"

Doplňák #1F: "sin"

Podprogram je třetím ze čtyř, které používají podprogram "SERIES GENERATOR" k vytváření Čebyševových polynomů. Aproximace  $\sin x$  je nalezena následovně:

a) Argument  $x$  je redukován a v tomto případě se provede přímo  $W=v$ . Pověšimněte si, že  $-1 \leq W \leq 1$  (pro konvergenci řady).

b) Je vytvořen argument  $Z$ , takový, že  $Z=2*W*W-1$ .

c) Použije se "SERIES GENERATOR" k získání  $(\sin(\pi*W/2))/W$ .

d) Konečně prostým násobením se získá hodnota  $\sin x$ .

```
37B5 SIN      RST  #28 FP-CALC      x
```

Proveď krok a).

```
DEFB #39, get-argt      W
```

Proveď krok b).

```
37B7 C-ENT    DEFB #31, zdvojení      W, W
              DEFB #31, zdvojení      W, W, W
              DEFB #04, násobení      W, W*W
              DEFB #31, zdvojení      W, W*W, W*W
              DEFB #0F, sčítání       W, 2*W*W
              DEFB #A1, stk-jedna     W, 2*W*W, 1
              DEFB #03, odečítání     W, 2*W*W-1 = Z
```

Proveď krok c).

```
DEFB #86, série-06      W, Z
1. 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, #BB, #23
5. DEFB #EE, exponent #7E
DEFB #92, #0D, #CD, #ED
6. DEFB #F1, exponent #81
DEFB #23, #5D, #1B, #EA
```

Po posledním průchodu smyčkou je poslední hodnota  $(\sin(\pi*W/2))/W$ .



```
DEFB #04, násobení      SIN(PI*W/2)-SIN x (nebo COS x)
DEFB #38, konec výpočtu
RET  Poslední hodnota na zázobníku kalkulátoru je SIN x (nebo COS x).
```

## FUNKCE "TAN"

Doplňák #21: "tan"

Podprogram vrací poslední hodnotu  $\sin x/\cos x$  s aritmetickým přetečením, je-li  $\cos x=0$ .

```
37DA TAN      RST  #28, FP-CALC      x
              DEFB #31, zdvojení      x, x
              DEFB #1F, sin           x, SIN x
              DEFB #01, záměna       SIN x, 1/x
              DEFB #20, cos          SIN x, COS x
```

DEFB #05,dělení

DEFB #38,konec výpočtu

RET

Je-li to potřeba, ohlaš aritmetické přetečení.

TAN x

Poslední hodnota na zásobníku kalkulátoru je TAN X.

## FUNKCE "ARCTAN"

Doplňěk #24: "atn"

Podprogram je posledním ze čtyř, které používají podprogram "SERIES GENERATOR" k vytváření Čebyševových polynomů. Podprogram vrací reálné číslo mezi  $-\pi/2$  a  $\pi/2$ , které se rovná hodnotě v radianech úhlu, jehož TAN je x. Aproximace ATN x je nalezena takto:

Hodnoty W a Y jsou nalezeny pro tyto tři případy hodnoty x a argument Z je pak formován takto:

a1) Jestliže  $-1 < x < 1$  pak  $W=0$  &  $Y=x$  b1) argument  $Z=2*Y*Y-1 = 2*x*x-1$

a2) Jestliže  $1 <= x$  pak  $W=\pi/2$  &  $Y=-1/x$  b2) argument  $Z=2*Y*Y-1 = 2*(x*x)-1$

a1) Jestliže  $x < -1$  pak  $W=-\pi/2$  &  $Y=-1/x$  b3) argument  $Z=2*Y*Y-1 = 2*(x*x)-1$

Ve všech případech musí být splněna podmínka pro konvergenci řad  $y: -1 <= Y <= 1$ .

c) Použije se "SERIES GENERATOR" k získání požadované funkce.

d) Konečně prostým násobením a sčítáním se získá hodnota ATN x.

Proveď krok a).

37E2 ATN	CALL #3297,RE-STACK	Použij FP formu x.
	LD A,(HL)	Vyzvedni exponent x.
	CP #81	Jestliže $x=Y$
	JR C,#37F8,SMALL	skoč dopředu v případě 1.
	RST #28,FP-CALC	x
	DEFB #A1,stk-jedna	x,1
	DEFB #1B,negace	x,-1
	DEFB #01,záměna	-1,x
	DEFB #05,dělení	-1/x
	DEFB #31,zdvojení	-1/x,-1/x
	DEFB #36,<0	-1/x,(1/0)
	DEFB #A3,stk-pi/2	-1/x,(1/0),PI/2
	DEFB #01,záměna	-1/x,PI/2,(1/0)
	DEFB #00,skok-pravda	-1/x,PI/2
	DEFB #06,na #37FA,CASES	Skok vpřed v případě 2.
	DEFB #1B,negace	-1/x,-PI/2
	DEFB #33,skok	
	DEFB #03,na #37FA,CASES	Skok vpřed v případě 3.
37F8 SMALL	RST #28,FP-CALC	Y
	DEFB #A0,stk-nula	Y,0

Proveď krok b).

37FA CASES	DEFB #01,záměna	W,Y
	DEFB #31,zdvojení	W,Y,Y
	DEFB #31,zdvojení	W,Y,Y,Y
	DEFB #04,násobení	W,Y,Y*Y
	DEFB #31,zdvojení	W,Y,Y*Y,Y*Y
	DEFB #0F,sčítání	W,Y,2*Y*Y
	DEFB #A1,stk-jedna	W,Y,2*Y*Y,1
	DEFB #03,odečítání	W,Y,2*Y*Y-1=Z



1. DEFB #8C,série-0C W,Y,Z  
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

```

Po posledním průchodu smyčkou je poslední hodnota:

```

1.) ATN X/X      2.) ATN (-1/X)/(-1/X)      3.) ATN(-1/X)/(-1/X)

```

Proveď krok d).

```

DEFB #04,násobení      1.) W,ATN X      2.) W,ATN(-1/X)      3.) W,ATN(-1/X)
DEFB #0F,sčítání      ATN X
DEFB #38,konec výpočtu
RET                    Poslední hodnota na zásobníku kalkulátoru je ATN X.

```

## FUNKCE "ARCSIN"

Doplněk #22: "asn"

Podprogram vrací reálné číslo mezi  $-\pi/2$  a  $\pi/2$ , které se rovná hodnotě úhlu v rad, jehož SIN je x. Jestliže  $y = \text{ASN } x$ , potom  $x = \text{SIN } y$ . Podprogram používá trigonometrickou identitu:

$$\text{TAN}(y/2) = \text{SIN } y / (1 + \text{COS } y)$$

```

3833 ASN      RST #28,FP-CALC      x
              DEFB #31,zdvojení    x,x
              DEFB #31, "          x,x,x
              DEFB #04,násobení    x,x*x
              DEFB #A1,stk-jedna   x,x*x,1
              DEFB #03,odčítání    x,x*x-1
              DEFB #1B,negace      x,1-x*x
              DEFB #2B,sqr         x,SQR(1-x*x)
              DEFB #A1,stk-jedna   x,SQR(1-x*x),1
              DEFB #0F,sčítání     x,1+SQR(1-x*x)
              DEFB #05,dělení     x/(1+SQR(1-x*x)) =TAN(y/2)
              DEFB #24,atn        y/2
              DEFB #31,zdvojení    y/2,y/2
              DEFB #0F,sčítání     y=ASN x
              DEFB #38,konec výpočtu
              RET

```

Poslední hodnota na zásobníku kalkulátoru je  $\text{ASN } x$ .

### FUNKCE "ARCCOS"

Doplněk #23: "acs"

Podprogram provede  $\text{ACS } x$  a vrací reálné číslo od 0 do  $\text{PI}$  včetně, které je rovno velikosti úhlu  $v$  rad, jehož  $\text{COS}$  je  $x$ . Podprogram používá vztah:  $\text{ACS } x = \text{PI}/2 - \text{ASN } x$ .

```
3843 ACS      RST #28,FP-CALC      x
              DEFB #22,asn        ASN x
              DEFB #A3,stk-pi/2   ASN x,PI/2
              DEFN #03,odčítání   ASN x-PI/2
              DEFB #1B,negace     PI/2-ASN x+ACS x
              DEFB #38,konec výpočtu
              RET
```

### FUNKCE "SQUARE ROOT" (=druhá odmocnina)

Doplněk #28: "sqr"

Podprogram provede  $\text{SQR } x$  a vrací kladnou druhou odmocninu reál. čísla  $x$ , je-li  $x$  kladné a nulu, je-li  $x$  nula. Záporné  $x$  vyvolá chybové hlášení A-Invalid argument (v podpr.EXPONENTIATION). Podprogram provádí  $\text{SQR}$  jako  $x^{0.5}$  a proto po uložení konstanty 0.5 pokračuje přímo podprogram EXPONENTIATION.

```
384A SQR      RST #28,FP-CALC      x
              DEFB #31,zdvojení   x,x
              DEFB #30,not        x,(1/0)
              DEFB #00,skok-pravda x
              DEFB #1E,na #386C,LAST x
              DEFB #A2,stk-polovina x,.5
              DEFB #38,konec výpočtu
```



### OPERACE "EXPONENTIATION"

Doplněk #06: "to-power"

Podprogram provádí binární operaci  $x$  na  $y$ , kterou považuje za ekvivalentní  $\text{EXP}(Y * \text{LN } x)$ . Pokud  $x=0$ , výsledek je 1 (pokud  $y=0$  ( $0^0=1$ ), vrací 0 když je  $y$  kladné). Při záporném  $y$  je hlášena chyba.

```
3851 TO-POWER RST #28,FP-CALC      x,y
              DEFB #01,záměna     y,x
              DEFB #31,zdvojení   y,x,x
              DEFB #30,not        y,x,(1/0)
              DEFB #00,skok-pravda y,x
              DEFB #07,na #385D,XISO y,x
```

Je-li  $x=0$ , provede se skok.

```
DEFB #25,ln          y,LN x (pro  $x<0$  chybové hlášení)
DEFB #04,násobení   y*LN x
DEFB #38,konec výpočtu
JP #36C4,EXP
```

Zde se posoudí 3 případy pro nulové  $x$ .

```
385D XISO      DEFB #02,výmaz      y
              DEFB #31,zdvojení   y,y
              DEFB #30,not        y,(1/0)
```

```

DEFB #00,skok-pravda      y
DEFB #09,na #386A,ONE     y

```

Skoč je-li x=0 a y=0.

```

DEFB #A0,stk-nula         y,0
DEFB #01,záměna           0,y
DEFB #37,>0                0,(1/0)
DEFB #00,skok-pravda     0
DEFB #06,na #386C,LAST

```

Skoč je-li x=0 a y kladné.

```

DEFB #A1,stk-jedna        0,1
DEFB #01,záměna           1,0
DEFB #05,dělení           Dělení nulou-chyba.
DEFB #02,výmaz

```



386C LAST

```

DEFB #A1,stk-jedna
DEFB #38,konec výpočtu
RET

```



386E ..... 3CFF DEFB #FF,#FF,.....

ROM Spectrum 48 nevyužívá

3D00 až 3FFF DEFB matice znakového souboru ASCII #20-#7F.

Př. písmeno "A" je zde uloženo takto:

```

DEFB #00                    00000000
DEFB #3C                    00111100
DEFB #42                    01000010
DEFB #42                    01000010
DEFB #7E                    01111110
DEFB #42                    01000010
DEFB #42                    01000010
DEFB #00                    00000000

```

### Přehled systémových proměnných:

Písmena ve sloupci Pozn. mají následující význam:

**X** Pokud změníte obsah této proměnné, systém se může zhroutit.

**M** Změnou hodnoty této proměnné nezískáte žádný efekt.

Čísllice ve sloupci **Pozn.** udávají velikost proměnné v bajtech.

Ve sloupci **Obsah** je uvedena hodnota proměnné po zapnutí počítače. Pokud není hodnota uvedena, nemá tato proměnná po zapnutí rozumný obsah.

Pozn.	Adresa	Jméno	Obsah	Význam	
N8	5C00	23552	KSTATE	DEFS 8	Pracovní místo je využíváno při čtení kláves.
N1	5C08	23560	LAST-K	DEFB 0	Zde je uložen kód naposledy stisknuté klávesy.
1	5C09	23561	REPDEL	DEFB 35	Čas (v 1/50 sekundy), po kterém se začne klávesa opakovat,
1	5C0A	23562	REPPER	DEFB 5	Čas (v 1/50 sekundy), mezi opakováním kláves.
N2	5C0B	23563	DEFADD	DEFW 0	Adresa argumentu uživatelské funkce.
N1	5C0D	23565	K-DATA	DEFB 0	Druhý bajt řízení barev (vložený z klávesnice).
N2	5C0F	23566	TVDATA	DEFB 0,0	Bajty barvy, AT a TAB při výstupu na obrazovku.
X3B	5C10	23568	STRMS	DEFS 2*28	Adresy kanálů.
2	5C36	23606	CHARS	DEFW #3C00	Adresa generátoru znaků zmenšená o 256.
1	5C38	23608	RASP	DEFB 64	Délka varovného bzučáku.
1	5C39	23609	PIP	DEFB 0	Délka pípnutí klávesnice.
1	5C3A	23610	ERR-NR	DEFB 255	0 jednu méně než číslo hlášení (chyby).
X1	5C3B	23611	FLAGS	DEFS 1	Různé příznaky pro řízení systému BASIC.
X2	5C3D	23613	ERR-SP	DEFW #FF54	Adresa položky na zásobníku, která se používá při chybě.
N2	5C3F	23615	LIST-SP	DEFW #FF54	Adresa návratu z automatického výpisu programu.
N1	5C41	23617	MODE	DEFB 0	Určuje režim kurzoru (K, L, C, E nebo G).
2	5C42	23618	NEWPPC	DEFS 2	Číslo řádky, na kterou se má skočit.
1	5C44	23620	NSPPC	DEFS 2	Číslo příkazu v řádce určené NEWPPC,
2	5C45	23621	PPC	DEFS 2	Číslo řádky, která se právě vykonává.
1	5C47	23623	SUBPPC	DEFS 2	Číslo právě vykonávaného příkazu.
1	5C48	23624	BORDCR	DEFB #00111000	Barva pozadí*8; dále obsahuje atributy pro dolní část obr.
2	5C49	23625	E-PPC	DEFS 2	Číslo aktivní řádky (ta na které je kurzor).
X2	5C4B	23627	VARS	DEFW #5CCB	Adresa začátku proměnných.
N2	5C4D	23629	DEST	DEFS 2	Adresa proměnné, do které se přiřazuje.
X2	5C4F	23631	CHANS	DEFW #5CB6	Adresa, kde jsou uloženy informace o kanálech.
X2	5C51	23633	KRCHL	DEFW #5CBB	Adresa kanálu, který používán pro vstup a výstup.
X2	5C53	23635	PROG	DEFW #5CCB	Adresa začátku programu v BASICu.
X2	5C55	23637	NXTLIN	DEFS 2	Adresa další řádky v programu.
X2	5C57	23639	DATADD	DEFS 2	Adresa ukončení poslední položky v příkazu DATA.
X2	5C59	23641	E-LINE	DEFW #5CCC	Adresa právě napsaného příkazu.
2	5C5B	23643	K-CUR	DEFS 2	Adresa kurzoru.
X2	5C5D	23645	CH-ADD	DEFS 2	Adresa znaku, který se bude interpretovat.
2	5C5F	23647	X-PTR	DEFS 2	Adresa výskytu syntaktické chyby (t.j. před znakem ?).
X2	5C61	23649	WORKSP	DEFW #5D08	Adresa dočasné pracovní paměti.
X2	5C63	23651	STKBOT	DEFW #5D08	Adresa dna zásobníku kalkulátoru.
X2	5C65	23653	STKEND	DEFW #5D08	Adresa vrcholu zásobníku a začátek volné paměti.
N1	5C67	23655	BREG	DEFS 1	Registr b kalkulátoru.
N2	5C68	23656	MEM	DEFW #5C92	Adresa paměť kalkulátoru.
1	5C6A	23658	FLAGS2	DEFS 1	Další příznaky.
X1	5C6B	23659	DF-SZ	DEFB 2	Počet řádek v dolní oblasti obrazovky (editační řádek).
2	5C6C	23660	S-TOP	DEFS 2	Počet řádek při automatickém výpisu programu.
2	5C6E	23662	OLDPPC	DEFS 2	Číslo řádky, na kterou skočí příkaz CONTINUE.
1	5C70	23664	OSPPC	DEFS 1	Číslo příkazu, na který skočí příkaz CONTINUE.

N1	5C71	23665	FLAGX	DEFS	1	Různé příznaky.
N2	5C72	23666	STRLFN	DEFS	2	Délka pravě vyhodnocovaného řetězce (před přiřazením).
N2	5C74	23668	T-ADDR	DEFS	2	Adresa další položky v tabulce syntaxe.
2	5C76	23670	SEED	DEFW	0	Násada pro generování náhodných čísel.
3	5C78	23672	FRAMES	DEFS	3	Čítač snímků. Je zvětšen každých 20ms.
2	5C7B	23675	UDG	DEFW	##FF58	Adresa prvního uživatelsky definovaného znaku.
1	5C7D	23677	COORDS	DEFS	1	Souřadnice x posledně vykresleného bodu.
1	5C7E	23678		DEFS	1	Souřadnice y posledně vykresleného bodu.
1	5C7F	23679	P-POSN	DEFS	1	Pozice tisku na ZX-Printer.
1	5C80	23680	PR-CC	DEFS	1	Dolní bajt adresy bajtu při výstupu řádky na ZX-Printer
1	5C81	23681		DEFS	1	Není využito.
2	5C82	23682	ECHO-E	DEFW	23*256+33	Adresa ve spodní části obr., za kterou nelze dát kurzor
2	5C84	23684	DF-CC	DEFS	2	Adresa výstupu znaku na obrazovku.
2	5C86	23686	DF-CCL	DEFS	2	Jako DF CC, ale pro dolní část obrazovky.
X1	5C88	23688	S-POSN	DEFS	1	Pozice sloupce pro výstup znaku příkazem PRINT.
X1	5C89	23689		DEFS	1	Pozice řádky pro výstup znaku příkazem PRINT.
X1	5C8A	23690	S-POSNL	DEFS	1	Jako S-POSN, ale pro dolní část obrazovky.
X1	5C8B	23691		DEFB	23	Pozice řádky jako S-POSN, ale pro dolní část obrazovky.
1	5C8C	23692	SCR-CT	DEFS	1	Počet řádků+1, po jejichž vypsání se výpis zastaví.
1	5C8D	23693	ATTR-P	DEFB	00111000	Nastavené barvy.
1	5C8E	23694	MASK-P	DEFB	0	Příznak transparentních barev.
N1	5C8F	23695	ATTR-T	DEFB	00111000	Dočasně platné barvy.
N1	5C90	23696	MASK-T	DEFB	0	Jako MASK-P, ale dočasné.
1	5C91	23697	P-FLAG	DEFS	1	Další příznaky.
N30	5C92	23698	MEMBOT	DEFS	30	Oblast vyhrazená pro kalkulátor.
2	5CB0	23728	NMIADD	DEFS	2	Používá podprogram pro obsluhu NMI.
2	5CB2	23730	RAMTOP	DEFW	##FF57	Adresa posledního bajtu, který používá BASIC.
2	5CB4	23732	P-RAMT	DEFW	##FFFF	Adresa posledního bajtu fyzické paměti.

## REJSTRÍK

<b>ÚVOD</b> . . . . .	5
<b>PODPROGRAMY RESTARTŮ A TABULKY</b> . . . . .	7
0000 START . . . . .	7
0008 ERROR-1 . . . . .	7
0010 PRINT-A-1 . . . . .	7
0018 SET-CHAR . . . . .	7
0020 NEXT-CHAR . . . . .	7
0028 FP-CALC . . . . .	7
0030 BC-SPACES . . . . .	8
0038 MASK-INT . . . . .	8
0053 ERROR-2 . . . . .	8
0066 RESET . . . . .	8
0074 CH-ADD+1 . . . . .	9
007D SKIP-OVER . . . . .	9
<b>KLÁVESNICOVÉ PODPROGRAMY</b> . . . . .	12
028E KEY-SCAN . . . . .	12
02BF KEYBOARD . . . . .	13
0310 <b>K-REPEAT</b> . . . . .	14
031E K-TEST . . . . .	14
0333 K-DECODE . . . . .	15
<b>PODPROGRAMY PRO OVLÁDÁNÍ REPRODUKTORU</b> . . . . .	18
03B5 BEEPER . . . . .	18
03FB BEEP . . . . .	19
<b>PODPROGRAMY SAVE - LOAD - VERIFY</b> . . . . .	22
04C2 SA-BYTES . . . . .	22
053F SA/LD-RET . . . . .	24
0556 LD-BYTES . . . . .	24
05E3 LD-EDGE-2 . . . . .	26
0605 SAVE-ETC . . . . .	27
07CB VR-CONTRL . . . . .	32
0802 LD-BLOCK . . . . .	33
0808 LD-CONTRL . . . . .	33
08B6 ME-CONTRL . . . . .	35
092C ME-ENTER . . . . .	37
0970 SA-CONTRL . . . . .	38
<b>PODPROGRAMY OBSLUHUJÍCÍ TISK NA OBRAZOVKU A TISKÁRNU</b> . . . . .	40
09F4 PRINT-OUT . . . . .	40
0A23 PO-BACK-1 . . . . .	40
0A3D PO-RIGHT . . . . .	41
0A4F PO-ENTER . . . . .	41
0A5F PO-COMMA . . . . .	41
0A69 PO-QUEST . . . . .	41
0A6D PO-TV-2 . . . . .	41
0AD9 PO-ABLE . . . . .	43
0ADC PO-STORE . . . . .	43
0B03 PO-FETCH . . . . .	43
0B24 PO-ANY . . . . .	43
0B7F PR-ALL . . . . .	44
0BDB PO-ATTR . . . . .	46
0C0A PO-MSG . . . . .	46



0C3B	PO-SAVE . . . . .	47
0C41	PO-SEARCH . . . . .	47
0C55	PO-SCR . . . . .	48
0D4D	TEMPS . . . . .	50
0D6B	CLS . . . . .	51
0DAF	CL-ALL . . . . .	52
0DD9	CL-SET . . . . .	52
0DFE	CL-SC-ALL . . . . .	53
0E44	CL-LINE . . . . .	54
0E88	CL-ATTR . . . . .	55
0E9B	CL-ADDR . . . . .	55
0EAC	COPY . . . . .	56
0ECD	COPY-BUFF . . . . .	56
0EDF	CLEAR-PRB . . . . .	57
0EF4	COPY-LINE . . . . .	57
<b><u>PODPROGRAMY EDITORU</u></b> . . . . .		59
0F2C	EDITOR . . . . .	59
0F81	ADD-CHAR . . . . .	60
0FA9	ED-EDIT . . . . .	60
0FF3	ED-DOWN . . . . .	61
1007	ED-LEFT . . . . .	61
100C	ED-RIGHT . . . . .	61
1015	ED-DELETE . . . . .	62
101E	ED-IGNORE . . . . .	62
1024	ED-ENTER . . . . .	62
1031	ED-EDGE . . . . .	62
1059	ED-UP . . . . .	63
1076	ED-SYMBOL . . . . .	63
107F	ED-ERROR . . . . .	63
1097	CLEAR-SP . . . . .	64
10A8	KEY-INPUT . . . . .	64
111D	ED-COPY . . . . .	65
1190	SET-HL . . . . .	67
11A7	REMOVE-FP . . . . .	67
<b><u>PROVÁDĚČÍ PODPROGRAMY</u></b> . . . . .		68
11B7	NEW . . . . .	68
12A2	MAIN-EXEC . . . . .	70
155D	MAIN-ADD . . . . .	73
15D4	WAIT-KEY . . . . .	74
15E6	INPUT-AD . . . . .	75
15EF	OUT-CODE . . . . .	75
1601	CHAN-OPEN . . . . .	75
1615	CHAN-FLAG . . . . .	76
1634	CHAN-K . . . . .	76
1642	CHAN-S . . . . .	76
164D	CHAN-P . . . . .	76
1652	ONE-SPACE . . . . .	77
1664	POINTERS . . . . .	77
168F	LINE-ZERO . . . . .	78
169E	RESERVE . . . . .	78
16B0	SET-MIN . . . . .	79
16D4	REC-EDIT . . . . .	79
16DB	INDEXER-1 . . . . .	79
16E5	CLOSE . . . . .	79
1701	CLOSE-2 . . . . .	80
171C	CLOSE-STR . . . . .	80

171E	STR-DATA	80
1736	OPEN	81
175D	OPEN-2	81
1781	OPEN-K	82
1785	OPEN-S	82
1789	OPEN-P	82
1793	CAT-ETC	83
1795	AUTO-LIST	83
17F5	LLIST	84
17F9	LIST	84
1855	OUT-LINE	85
18B6	NUMBER	86
18C1	OUT-FLASH	86
18E1	OUT-CURS	86
190F	LN-FETCH	87
1925	OUT-SP-2	87
196E	LINE-ADDR	88
1980	CP-LINES	89
198B	EACH-STMT	89
19B9	NEXT-ONE	90
19DD	DIFFER	90
19E5	RECLAIM-1	90
19FB	E-LINE-NO	91
1A1B	OUT-NUM-1	91

**INTERPRETACE PŘÍKAZŮ V BASICOVÉM ŘÁDKU** . . . . . 93

1B17	LINE-SCAN	96
1B28	STMT-LOOP	96
1B6F	SEPARATOR	97
1B76	STMT-RET	97
1B8A	LINE-RUN	97
1B9E	LINE-NEW	98
1BB2	REM	98
1BB3	LINE-END	98
1BBF	LINE-USE	98
1BD1	NEXT-LINE	99
1BEE	CHECK-END	99
1BF4	STMT-NEXT	99
1C0D	CLASS-03	100
1C10	CLASS-00	100
1C16	JUMP-C-R	100
1C1F	CLASS-01	100
1C22	VAR-A-1	100
1C56	VAL-FET-1	101
1C6C	CLASS-04	101
1C79	NEXT-2NUM	102
1C96	PERMS	102
1CBE	CLASS-09	103
1CDB	CLASS-0B	103
1CDE	FETCH-NUM	103

**PŘÍKAZOVÉ PODPROGRAMY** . . . . . 104

1CEE	STOP	104
1CF0	IF	104
1D03	FOR	104
1DB6	LOOK-PROG	106
1DAB	NEXT	107
1DDA	NEXT-LOOP	108

1DED	READ	108
1E27	DATA	109
1E39	PASS-BY	109
1E42	RESTORE	109
1E4F	RANDOMIZE	109
1E5F	CONTINUE	110
1E67	GO-TO	110
1E7A	OUT	110
1E80	POKE	110
1E85	TWO-PARAM	110
1E94	FIND-INT1	111
1EA1	RUN	111
1EAC	CLEAR	111
1EED	GO-SUB	112
1F05	TEST-ROOM	112
1F1A	FREE-MEM	112
1F23	RETURN	113
1F3A	PAUSE	113
1F54	BREAK-KEY	113
1F60	DEF-FN	114
1FC3	UNSTACK-Z	115
1FC9	LPRINT	115
1FCD	PRINT	115
1FF5	PRINT-CR	115
1FFC	PR-ITEM-1	116
2045	PR-END-Z	117
204E	PR-POSN-1	117
2070	STR-ALTER	117
2089	INPUT	117
21B9	IN-ASSIGN	120
21D0	IN-STOP	120
21D6	IN-CHAN-K	120
21E1	CO-TEMP-1	121
226C	CO-CHANGE	123
2294	BORDER	123
22AA	PIXEL-ADD	124
22CB	POINT-SUB	124
22DC	PLOT	125
2307	STK-TO-BC	125
2314	STK-TO-A	125
2320	CIRCLE	126
2382	DRAW	128
247D	CD-PRMS1	132
24B7	DRAW-LINE	133
<b><u>HODNOCENÍ VÝRAZŮ</u></b>		
24FB	SCANNING	135
2530	SYNTAX-Z	136
25AF	S-U-PLUS	138
25B3	S-QUOTE	138
25E8	S-BRACKET	138
25F8	S-RND	139
2627	S-PI	139
2634	S-INKEY\$	139
2668	S-SCREEN\$	140
2672	S-ATTR	140
267B	S-POINT	140
26C9	S-LETTER	141

27BD	S-FN-SBRN	145
28AB	FN-SKPOVR	147
28B2	LOOK-VARS	148
2951	STK-F-ARG	151
2996	STK-VAR	152
2A52	SLICING	155
2AB6	STK-STORE	156
2ACC	INT-EXP1	156
2AEE	DE, (DE+1)	157
2AF4	GET-HL*DE	157
2AFF	LET	158
2BA6	L-ENTER	161
2BAF	L-ADD\$	161
2BC6	L-STRING	162
2BEA	L-FIRST	162
2BF1	STK-FETCH	162
2C02	DIM	163
2C88	ALPHANUM	165
2C8D	ALPHA	165
2C9B	DEC-TO-FP	165
2D1B	NUMERIC	167
2D22	STK-DIGIT	167
2D28	STACK-A	167
2D2B	STACK-BC	167
2D3B	INT-TO-FP	168

#### **ARITMETICKÉ PODRPOGRAMY**

2D4F	E-TO-FP	169
2D7F	INT-FETCH	170
2D8C	P-INT-STO	170
2DA2	FP-TO-BC	171
2DC1	LOG(2^A)	171
2DD5	FP-TO-A	172
2DE3	PRINT-FP	172
2F8B	CA=10*A+C	178
2F9B	PREP-ADD	178
2FBA	FETCH-TWO	179
2FDD	SHIFT-FP	180
3004	ADD-BACK	180
300F	SUBTRACT	181
3014	ADDITION	181
30A9	HL=HL*DE	183
30C0	PREP-M/D	184
30CA	MULTIPLY	184
31AF	DIVISION	188
3214	TRUNCATE	189
3293	RE-ST-TWO	191
3297	RE-STACK	191

#### **FLOATING POINT KALKULÁTOR**

335B	CALCULATE	193
33A2	FP-CALC-2	196
33A9	TEST-5-SP	196
33C0	MOVE-FP	196
33C6	STK-DATA	197
33F7	SKIP-CONS	197
3406	LOC-MEM	198
340F	GET-MEM-0	198

341B	STK-ZERO	198
342D	ST-MEM-0	199
343C	EXCHANGE	199
3449	SERIES-06	200
346A	ABS	201
346E	NEGATE	201
3492	SGN	202
34A5	IN	202
34AC	PEEK	202
34B3	USR-NO	202
34BC	USR-\$	203
34E9	TEST-ZERO	203
34F9	GREATER-0	204
3501	NOT	204
3506	LESS-0	204
350B	FP-0/1	204
351B	OR	205
3524	NO-&-NO	205
352D	STR-&-NO	205
353B	NO-L-EQL	205
359C	STRS-ADD	207
35B7	STK-PNTRS	207
35C9	CHRS	207
35DE	VAL	208
361F	STR\$	209
3645	READ-IN	209
3669	CODE	209
3674	LEN	210
367A	DEC-JR-NZ	210
3686	JUMP	210
368F	JUMP-TRUE	210
369B	END-CALC	211
36A0	N-MOD-M	211
36AF	INT	211
36C4	EXP	212
3713	LN	213
3783	GET-ARGT	215
37AA	COS	216
37B5	SIN	217
37DA	TAN	217
37E2	ATN	218
3833	ASN	219
3843	ACS	220
384A	SQR	220
3851	TO-POWER	220

**Přehled systémových proměnných** . . . . . 222

**REJSTŘÍK** . . . . . 224

Elektronická verzia: ).1.20%%  
Peter Turányi alias Softhouse  
<http://softhouse.speccy.cz>

Ďakujem

Lukášovi Macurovi za pomoc s OCR a korektúrami  
Zdeňkovi Starému za skeny