BETA BASIC 3.0

(C) Betasoft 1985, 92 Oxford Road, Masley, Birmingham

PREHĽAD 2
PRÍKAZY: 2
FUNKCIE: 3
ÚVOD 4
EDITÁCIA 4
PROCEDÚRY A PARAMETRE 5
Referencie, alebo odovzdávanie parametra adresou: 7
Zoznamy parametrov: 8
Rekurzia: 8
ŠTRUKTÚROVANÉ PROGRAMOVANIE 9
Zjednodušený prístup na periférne zariadenia: 9
Prístup k dátam - polia, triedenie, prehľadávanie 9
GRAFICKÉ PRÍKAZY 9
TOOLKIT - Zjednodušená manipulácia s programami 9
SYNTAX PRÍKAZOV 10
ALTER, AUTO, BREAK, CLEAR, CLOCK 10
CLS, CONTROL CODES 11
COPY, CSIZE, DEFAULT, DEF KEY 12
DEF PROC, DELETE, DO UNTIL, DPOKE 13
DRAW TO, EDIT, ELSE, END PROC, EXIT IF, FILL 14
GET, JOIN 15
KEYIN, KEYWORDS, LET, LIST/LLIST 16
LIST VAL$, LIST DEF KEY, LIST FORMAT, LIST PROC, LIST REF, LOCAL 17
LOOP UNTIL, MERGE, MOVE, ON, ON ERROR, OVER, PLOT 18
POKE, POP, PROC, READ LINE, REF 19
RENUM, ROLL 20
SAVE DATA, SCROLL, SORT, SPLIT, TRACE 21
UNTIL, USING, VERIFY, WHILE 22
WINDOW, XOS, YOS, XRG, YRG 23
FUNKCIE 24
AND, BIN$, CHAR$ 24
COSE, DEC, DPEEK, EOF, FILLED, HEX$, INARRAY, INSTRING 25
ITEM, LENGTH, MEM, MEMORY$, MOD, NUMBER, OR, RNDM, SCRN$ 26
SHIFT$, SINE, STRING$, TIME$, USING$, XOR 27
PRÍLOHA A: Množina znakov 28
PRÍLOHA B: Rozširujúce a nové chybové hlásenia: 29
PRÍLOHA C: Chybové kódy: 30

PREHĽAD

PRÍKAZY:

ALTER	- Zmena atribútov obrazovky
	- Zmena časti programu
AUTO	- Automatické číslovanie riadkov
BREAK	- Vylepšený BREAK
CLEAR	- Presun RAMTOP-u bez straty premenných
CLOCK	- Hodiny s Alarmom: BEEP aj GOSUB
CLS	- Vymazanie okna (WINDOW)
CONTROL CODES	- Riadenie formátu a kurzora
COPY	- Kopírovanie častí polí a reťazcov
CSIZE	- Nastavenie veľkosti znakov a rozostupov
DEFAULT	- Nastavenie štandartných hodnôt premenných
	- Výber SAVE/LOAD zariadenia
DEF KEY	- Definícia funkčnej klávesy
DEF PROC	- Definícia procedúry
DELETE	- Vymazanie časti programu
	- Vymazanie častí polí a reťazcov
DO	- Začiatok cyklu DO - LOOP
DPOKE	- Dvojitý poke (2 Byte)
DRAW TO	- Vykreslenie čiary do daného bodu
EDIT	- Editovanie riadku programu
	- Editovanie obsahu premennej
ELSE	- Časť IF - THEN - ELSE štruktúry
END PROC	- Ukončenie procedúry
EXIT IF	- Podmienené opustenie cyklu DO - LOOP
FILL	- Vyplnenie oblasti obrazovky
GET	- Vstup stlačenej klávesy
	- Odloženie časti obrazu do premennej
JOIN	- Spájanie riadkov programu
	- Spájanie polí a reťazcov
KEYIN	- Vstup reťazca ako programového riadku
KEYWORDS	- Nastavenie formátu vstupu
LET	- Viacnásobné priradenie
LIST, LLIST	- Výpis časti programu
LIST DATA	- Výpis obsahu všetkých premenných
LIST VAL	- Výpis obsahu číselných premenných
LIST VAL$	- Výpis obsahu reťazcových premenných
LIST DEF KEY	- Výpis obsahu definovaných kláves
LIST FORMAT	- Listing programu v rôznych formátoch
LIST PROC	- Výpis danej procedúry
LIST REF	- Výpis referencií na premennú
LOCAL	- Definícia lokálnej premennej
LOOP	- Koniec cyklu DO - LOOP
MERGE	- Funguje autostart (Microdrive)
MOVE	- Pracuje pre všetky dátové typy (Microdrive)
ON	- Voľba príkazu alebo čísla riadku
ON ERROR	- Spracovanie chyby
OVER	- Logický OR s obsahom obrazovky
PLOT	- Vypisovanie reťazcov od každého pixlu
POKE	- Ukladanie reťazcov do pamäti
POP	- Vybratie BASIC stacku (GO SUB, DO-LOOP, PROC)

PROC	- Volanie procedúry (Možno vypustiť)
READ LINE	- Čítanie reťazcov bez úvodzoviek
REF	- Vyhľadanie referencií v programe
	- Nastavenie premennej ako parametra
RENUM	- Prečíslovanie, presuny a kopírovanie
ROLL	- Posun časti obrazu
SAVE	- Zápis časti programu, premenných
SCROLL	- Posun časti obrazu
SORT	- Triedenie polí, reťazcov
SPLIT	- Rozdelenie programového riadku
TRACE	- Sledovanie behu programu
UNTIL	- Použitie v cykloch (Kým neplatí)
USING	- Formátovaný výstup dát
VERIFY	- Overenie časti programu, premenných
WINDOW	- Nastavenie rozmerov okna
	- Určenie aktuálneho okna
WHILE	- Použitie v cykloch (Kým platí)
XOS YOS	- Nastavenie počiatku súradnicového systému,
XRG YRG	- Nastavenie šírky a dĺžky obrazovky

FUNKCIE:

AND	- Bitový AND
BIN$	- Konverzia Decimal - Bin
CHAR$	- Konverzia Integer - String
COSE	- Rýchly kosínus
DEC	- Konverzia Hex - Decimal
DPEEK	- Dvojitý PEEK
EOF	- Koniec súboru (Microdrive)
FILLED	- Veľkosť zaplneného úseku
HEX$	- Konverzia Decimal - Hex
INARRAY	- Prehľadanie poľa
INSTRING	- Prehľadanie reťazca
ITEM	- Zistenie typu dát v READ
LENGHT	- Dĺžka a adresa poľa
MEM	- Veľkosť voľnej pamäte
MEMORY$	- Celá pamäť je ako premenná
MOD	- Funkcia MODULO
NUMBER	- Konverzia String - Integer
OR	- Bitový OR
RNDM	- Vylepšený RND (Dá aj celé čísla)
SCRN$	- Vylepšený SCREEN$ (Pozná aj UDG)
SHIFT$	- Ďalšie kódy, písmená veľké /malé, atď.
SINE	- Rýchly sínus
STRING$	- Viacnásobný reťazec
TIME$	- Aktuálny čas
USING$	- Formát čísla
XOR	- Bitový XOR

ÚVOD

Vďaka BB03 sa stáva Vaše 48k Spectrum vlastníkom najlepšieho 8-bitového BASIC-u. Pre jeho používanie nie je potrebné veľké školenie. Najprv je potrebné prečítať si stať o editácii a potom v ľubovoľnom poradí ostatné príkazy.

Nahranie BB03: LOAD "" alebo LOAD "Beta Basic"

BB03 si vtiahne basic riadky 0,1,2 a odštartuje sa na 2. Binárna časť má 18k. RAMTOP sa nastaví na 47070. Potom sa riadky 1 a 2 vymažú a zostane len 0. Tým je BB03 pripravený.

Vlastnosti:

Riadok 0 obsahuje definície BB03 funkcií. Tento sa stáva súčasťou každého programu. Na staré programy, ktoré neboli nahraté v BB03, treba použiť MERGE, pretože pri LOAD by sa zotrel potrebný riadok 0. NEW maže len program a urobí CLEAR, riadok 0 je ponechaný. Programy napísane v BB03 môžeme nahrávať cez SAVE a LOAD, pretože sa uloží aj riadok 0.

Predĺžené pípanie kláves signalizuje činnosť BB03. Dá sa vypnúť pomocou POKE 23609,0. Ďalší príznak práce je invertovaný riadkový kurzor pri listingu.

Staré programy bežia v BB03 rýchlejšie ako v pôvodnom Sinclair Basicu. Je to preto, že v cykloch a skokoch sa využívajú priamo adresy a nie čísla riadkov, ktoré sa hľadajú.

EDITÁCIA

Kľúčové slová: **EDIT, KEYWORDS, LIST FORMAT, CSIZE, JOIN, SPLIT**

Starý editor Spectra je k dispozícii tak ako bol, ale sú pridané ďalšie príkazy, ktoré sú teraz v krátkosti popísane. Podrobný popis je v abecednom zozname príkazov ďalej.

Aktuálny riadkový kurzor: je zobrazený inverzne, rýchlejšie sa pohybuje kurzorovými tlačítkami. Ak pozícia nesúhlasí, treba stlačiť ENTER.

Editácia riadku: Tlačítko '0' funguje ako príkaz 'EDIT'. Ak ho stlačíte po 'ENTER', príkaz 'EDIT' sa vypíše. '0' číslo riadku 'ENTER' alebo '0' 'ENTER' spôsobí výpis riadku do editačnej oblasti. Ak nezadáte číslo riadku, bude editovaný riadok s programovým kurzorom. EDIT sa dá vypísať ešte ako Graphics, Caps shift + 5.

Pohyb kurzora v riadku: V editovanom riadku je možný pohyb všetkými kurzorovými šípkami. Kurzor skáče medzi riadkami.

Kurzorové režimy: Za normálnych okolností sa kódy príkazov dajú vložiť v režime **K**. Text iba v **L/C**. V BB03 si môžete vybrať, či chcete vkladať jedným stlačením klávesy v režime **K**, alebo vypísaním v **L/C** režime. Ak je kurzor v režime **K** prechod do **L/C** je cez SPACE. Naopak z **L/C** do **K** sa dostaneme cez Symbol Shift + ENTER. (Užitočné ak ste vypli úplne **K** režim a chcete pri ALTER, REF alebo KEYIN vkladať 1-byte-tokeny do reťazcov).

KEYWORDS - Riadenie pri vkladaní programu:
Tento príkaz slúži na nastavenie vkladacích režimov. Môžete si zvoliť vkladanie príkazov a funkcií vo forme bežnej pre Spectrum alebo formou výpisu. Na začiatku je nastavený stav **KEYWORDS 3**, ktorý umožňuje výber ľubovoľnej formy prípadne vkladanie miešane. Pri **KEYWORDS 0** sú vypnuté nové rozširujúce príkazy a zobrazujú sa grafické znaky. Pre jednoklávesové vstupy v **K** režime treba pri doplňujúcich kľúčových slovách zaradiť režim **G** a potom stlačiť odpovedajúcu

klávesu s doplnkovým príkazom. Pri funkciách môžeme použiť štandardne FN+písmeno+zátvorky s argumentom, alebo vypísať ' ','F','N',' ' ..., alebo môžeme vypísať priamo názov funkcie BB03.

Test vstupného príkazu: Po stlačení 'ENTER' za príkazom sa otestuje jeho správnosť. Pri chybe sa ozve BEEP. Jeho dĺžka sa nastavuje na adrese 23608. Pri vkladaní programu ako textové reťazce, je vhodné vkladať príkazy v malej abecede. BB03 si rozpoznané príkazy prevedie do veľkej abecedy, ostatné nechá malé.

LIST FORMAT: Príkaz umožňuje robiť výpis programu so zarážkami pri blokoch. Tým sa zvýši čitateľnosť programu, prípadne sa ľahšie objavia chyby v štruktúre. (FOR-NEXT a.p.)

CSIZE: Umožňuje meniť veľkosť písmen od 64 (s OVER 1 až 85) znakov na riadok až po veľké písmená na nadpisy.

JOIN (Programové riadky) a SPLIT: JOIN spojí 2 riadky programu do jedného, SPLIT rozloží jeden riadok do dvoch.

Vloženie 'Nového riadku' do programového riadku: V ľubovoľnom mieste je možné vložiť do vstupného riadku programu alebo INPUT riadku kód CHR$ 13 = NL pomocou CAPS SHIFT+ENTER. Tým sa riadok neukončí, ale prejde sa na začiatok ďalšieho.

PROCEDÚRY A PARAMETRE

Kľúčové slová: **DEF PROC**, **PROC** (prip.), **LOCAL**, **DEFAULT**, **REF**, **READ LINE**, **LIST PROC**
Funkcia: **ITEM()**

Táto kapitola Vás oboznámi s možnosťami využívania procedúr. Napísanú procedúru je možné pomocou MERGE v prípade potreby nahrať z knižnice, ktorú si môžeme vytvoriť.

V hlavnom programe sa volá procedúra priamo menom a zadajú sa jej prípadné požadované parametre. Procedúry uložené v pamäti je možné volať aj v priamom režime, čo umožňuje rozšíriť množinu príkazov BB03. Každá procedúra má byť len taká veľká, aby bola ľahko pochopiteľná.

Procedúra je vlastne podprogram, ktorý začína DEF PROC, meno a prípadne zoznamom parametrov, ktoré sú jej odovzdané pri volaní a konci miesto RETURN vložíme END PROC. Procedúra sa nájde bez ohľadu na to, či je na začiatku alebo na konci programu, neberie sa do úvahy či program bežal alebo nie.

Napríklad: Urobme ekvivalent CP/M príkazu DIR

```
100 DEF PROC DIR
110 CAT 1
120 END PROC
```

Keď vložíte RUN, nič sa nestane. Procedúra nebola volaná, podobne ako DEF FN.
Vložte: 10 dir a RUN. Vypíše sa obsah Microdrive.
Alebo: 'DIR' a procedúra sa opäť vykoná. Teda Spectrum BB03 má o príkaz viac.

Meno procedúry musí začínať písmenom a končiť **'SPACE',':','REF','DATA', alebo 'ENTER'.**

V menách je možné používať veľkú aj malú abecedu (bez rozlíšenia), číslice alebo podčiarknutie. Je možné použiť však aj iné znaky.

Procedúry musia vždy začínať **DEF PROC** a na končiť **END PROC.** Ak chcete použiť viac END PROC v 1 procedúre (pred čím varujeme), potom nevie počítač nájsť skutočný koniec procedúry.

Príklad 2:

```
100 DEF PROC dir číslo
110 CAT číslo
120 ENDPROC
```

Premenná číslo sa volá formálny parameter. Cez formálne parametre sa pri volaní načítajú skutočné hodnoty do procedúry. Pri volaní je možné použiť potom tvar:

```
dir 1,
```
alebo
```
LET x=1:dir x,
```
alebo
```
dir ľubovoľný numerický výraz.
```

Tento spôsob sa volá načítanie parametra hodnotou, lebo do procedúry sa načíta len hodnota. Formálny parameter v definícii procedúry musí byť premenná. Táto je však lokálna pre danú procedúru a BB03 ju mimo procedúry nepozná. Premenná použitá inde v tele procedúry je systému známa a nazýva sa globálna. Aby aj takáto bola lokálna, treba použiť príkaz **LOCAL**.

Príklad:

```
100 DEF PROC hallo xkrat
110 FOR n=1 TO xkrat
120 PRINT "HALLO"
130 NEXT n
140 END PROC
```

Ak zadáte HALLO 4, vyvolá sa procedúra. Potom PRINT n,xkrat; vypíše sa premenná n, pretože bola globálna, ale xkrat bude neexistujúca premenná.

Je dobrou programátorskou praxou všetky premenné v procedúre mať lokálne, teda aj 'n'. To sa zabezpečí príkazom:

```
105 LOCAL n
```

Ak existovala globálna premenná v programe s názvom 'n', nemá teraz volanie procedúry na ňu vplyv.

Príklad:

```
100 DEF PROC box x,y,sirka,dlzka
110 PLOT x,y: DRAW sirka,0
120 DRAW 0,-dlzka: DRAW -sirka,0
130 DRAW 0,dlzka
140 END PROC
```

Príkaz BOX 100,100,10,40 vykreslí obdĺžnik s ľavým rohom v bode 100,100 šírkou 10 a dĺžkou 40.

Ak by sme chceli niektorý parameter vynechať, objaví sa chyba. Potom je potrebné použiť príkaz **DEFAULT**, napríklad:

```
105 DEFAULT dlzka=sirka
```

Teraz je možné volať BOX 100,100,50 a vykreslí sa štvorec.

Príkaz DEFAULT nastaví parametre v takom prípade, ak neboli nastavené cez skutočné parametre.

Referencie, alebo odovzdávanie parametra adresou:

Tento spôsob odovzdávania parametrov umožňuje prenos údajov nielen do procedúry, ale aj z procedúry von (Obdoba VAR parametra v PASCALe). Pred formálne parametre, ktoré budú tohto typu, treba vložiť kľúčové slovo **REF**.

Príklad:

```
200 DEF PROC vymena REF a$,REF b$
210 LOCAL t$
220 LET t$=a$,a$=b$,b$=t$
230 END PROC
```

Program:

```
10 LET x$="BASIC",y$="BETA "
20 vymena x$,y$
30 PRINT x$,y$
```

Uvidíte, že x$ a y$ sú vymenené. V prípade, že by nebolo použité REF, výmena by sa urobila len v procedúre a vytlačené by boli pôvodné obsahy.

Na rozdiel od iných počítačov s procedúrami v BASICu umožňuje BB03 ako premennú používať aj polia. Jediné obmedzenie je také, že ich možno prenášať len cez REF. Počas používania v procedúre sa im len zmení meno bez kopírovania obsahu. Tým sa šetrí čas a miesto. Ak to potrebujete, je možné vytvoriť lokálne pole a cez rozšírený príkaz **COPY** urobiť skopírovanie.

Príklad:

```
100 DIM t(10)
110 FOR n=1 TO 10
120 LET t(n)=n
130 NEXT n
140 SUMA t(),vysledok
150 PRINT vysledok

300 REM Procedura scita vsetky prvky pola
310 DEF PROC SUMA REF a(), REF sum
320 LOCAL n
330 LET sum=0
340 FOR n=1 TO LENGTH (1,"a()")
350 LET sum=sum+a(n)
360 NEXT n
370 END PROC
```

Tým, že bola použitá funkcia **LENGTH**, je možné v procedúre sčítať jednorozmerné pole ľubovoľnej dĺžky.

Zoznamy parametrov:

Ako viete, môže za volaním procedúry nasledovať 0 alebo viac parametrov. Ale môže nastať taký prípad, že nevieme vopred, koľko bude mať daná premenná parametrov. V tom prípade použijeme na mieste formálnych parametrov kľúčové slovo **DATA**, a v procedúre sa preberajú cez **READ**. Informácie o parametroch dáva funkcia **ITEM()**, ktorá nadobúda nasledovné hodnoty:

0: Nie je k dispozícii žiadny parameter
1: K dispozícii je číselný parameter
2: Môžeme prečítať reťazec

Príklad:
```
100 DEF PROC era DATA
110  DO UNTIL ITEM()=0
120   READ a$
130   ERASE 1,a$
140  LOOP
150 END PROc
```

Volanie môže byť:
```
era "file 1"
era "file 1","file 2","file 3",...,"file n"
```

Ak bude 120 READ LINE a$, potom je možné aj volanie:
```
era file 1
era file 1,file 2,...,file 3
```

Rekurzia:

V definícii procedúry je možné volať samotnú procedúru.

Príklad:
```
100 DEF PROC diamant x,y,size,diff
    DEFAULT diff=15
    PLOT x,y-size
    DRAW -size,size
    DRAW size,size
    DRAW size,-size
    DRAW -size,-size
110  IF size4 THEN
        diamant x,y+size,size-diff
        diamant x,y-size,size-diff
        diamant x-size,y,size-diff
        diamant x+size,y,size-diff
120 END PROC
```

Volanie:
```
diamant 128,88,40
```

Pozn.: Výpis je v LIST FORMAT 2

ŠTRUKTÚROVANÉ PROGRAMOVANIE

Kľúčové slová:
a) Procedúry **DEF PROC, END PROC, LOCAL, REF** (Parameter), **DEFAULT** (Hodnota premennej), **ITEM()** (Funkcia)
b) **DO, LOOP, EXIT IF, WHILE, UNTIL,** slúžia ako REPEAT a WHILE v PASCALe, ale sú flexibilnejšie
c) **IF - THEN - ELSE**
d) **ON** - Vyberá čísla riadkov alebo príkazy zo zoznamu, obdoba **CASE**
e) **LIST FORMAT** - vypisuje programy so zarážkami, tak ako sú vytvárané štruktúry.

Zjednodušený prístup na periférne zariadenia:

Kľúčové slová: **DEFAULT (SAVE/LOAD), SAVE, MERGE, MOVE,**
Funkcia: **LENGTH**

Pre Microdrive je možné vynechať *m a teda príkaz bude: LOAD 1;"PROGRAM".
Ak ešte DEFAULT=m1, stačí pre uloženie na Microdrive 1 príkaz: SAVE "PROGRAM".

SAVE a VERIFY umožňujú pracovať aj s časťou programu. Tiež je možný zápis samotných premenných. **MERGE** môže z Microdrive nahrať aj samoštartujúce programy. **MOVE** prenáša okrem dátových súborov aj programy, CODE a polia.

Funkcia **LENGTH** dá ako výsledok dĺžku a adresu poľa, takže je možné manipulovať s poľom pomocou LOAD "meno" CODE

Prístup k dátam - polia, triedenie, prehľadávanie

Kľúčové slová :
JOIN, COPY, DELETE, SORT pre polia a reťazce **EDIT** premenná, **SAVE DATA, USING**
Hľadacie funkcie: **INARRAY, INSTRING**
Funkcie: **LENGTH, CHAR$, NUMBER, EOF, SHIFT$, USING$**

JOIN a **COPY** umožňujú polia alebo ich časti spájať a miešať. Rozmery poľa je možné zmeniť bez straty dát. **DELETE** vymaže časť poľa. **SORT** triedi polia. **INARRAY, INSTRING** veľmi rýchle prehľadávajú polia alebo reťazce. Cez EDIT je možné zmeniť obsah premenných. Cez SAVE DATA sa dajú uložiť premenné na pásku (Microdrive). **USING, USING$** formátujú výstup dát. Funkcia **EOF** udáva, či už boli z Microdrive prečítané všetky dáta.

GRAFICKÉ PRÍKAZY

Kľúčové slová: **ALTER, CONTROL CODES, CSIZE, DRAW TO, FILL, GET** (Oblasť obrazovky), **OVER 2, PLOT, POKE, ROLL, SCROLL, WINDOW, XOS/YOS/XRG/YRG**
Funkcie: **SINE, COSE, FILLED, MEMORY$, SCRN$**

TOOLKIT - Zjednodušená manipulácia s programami

Kľúčové slová: **ALTER, AUTO, DEF KEY, DELETE, LIST ... TO ..., LIST REF, REF, RENUM, LIST DATA/VAL/VAL$, LIST DEF KEY, LIST PROC,**
Funkcia: **MEM()**

SYNTAX PRÍKAZOV

To čo je v zátvorke <>, nie je povinné.

ALTER <Popis atribútov> TO Popis atributov ('G'+A)
ALTER Referencia TO Referencia

1) Umožňuje efektívne manipulovať s atribútmi. Jednoduchá forma zmení atribúty celej obrazovky bez vymazania textu. Zložitejšia forma zmení len tie atribúty, ktoré spĺňajú podmienku prvého popisu atribútov (Môže ich byť viac).

Príklad:
```
ALTER INK 3, BRIGHT 1, PAPER 7 TO INK 5, FLASH 1
```

2) V celom programe sa výskyt prvej referencie nahradí druhou referenciou.

a; ALTER a$ TO b$	Zmení v programe a$ na b$
b; ALTER sum TO s	Zmení v programe mená sum na s
c; ALTER 1 TO 23	Zmení v programe na mieste 5 byte, kde je kód 1 na kód 23
d; ALTER "stary" TO "novy"	Zmení reťazce aj vo vnútri
e; ALTER (s$) TO "retazec"	Zmení obsah premennej

AUTO <číslo riadku><,krok> ('G'+6)

Pre režim vkladania textu programu sa generujú čísla riadkov. Ukončenie režimu sa urobí stlačením BREAK viac ako 1s alebo vymaž čísla riadku a potom nejaký príkaz (LIST a pod.).
Ak nezadáte hodnoty, predpokladá sa 10,10.

BREAK (CAPS SHIFT + SPACE)

Príkaz funguje aj pre podprogramy v strojovom kóde. Treba ho podržať dlhšie ako 1s.
Pozor! BB03 pracuje v Inrerrupt mode 2.

CLEAR bytes ('K'+X)

Pre bytes <767 sa presunie RAMTOP o zadané číslo nadol. Možno zadať aj záporné číslo. Tento posuv nemá vplyv na obraz, premenné a stack GO SUB, DO . . . Definície funkčných kláves a Windows sú tiež posunuté. >

CLOCK <číslo, alebo reťazec > ('G'+C)

Týmto príkazom má užívateľ prístup k Interruptom riadeným 24 hodinovým hodinám, ktoré môžu byť prípadne zobrazene v pravom hornom rohu obrazovky. Ku danému času je možné zadať budík, alebo prerušiť riešenie programu odskokom do podprogramu.
Pozor! Počas BEEP a pri prístupe na periférie hodiny stoja.

a) **CLOCK n** = Nastavenie režimu.

n	*Alarm GO SUB*	*Alarm BEEP*	*Zobrazenie*
0	-	-	-
1	-	-	ANO
2	-	ANO	-
3	-	ANO	ANO
4	ANO	-	-
5	ANO	-	ANO
6	ANO	ANO	-
7	ANO	ANO	ANO

Pri naštartovaní sú hodiny v režime 0.

b) **CLOCK "HH:MM:SS"** = Nastavenie času
':' sa môže vynechať, pretože akceptovaných je len prvých šesť číslic. Písmená okrem 'a' a 'A' sa ignorujú.

c) **CLOCK "aHH:MM:SS"** = Nastavenie Alarmu.
Pri dosiahnutí času bude počítač v režimoch 2,3,6,7 zvoniť. V režimoch 4-7 sa skočí do podprogramu, ktorý začína na riadku (8-9999) alebo je prvý riadok za CLOCK.

d) **CLOCK číslo riadku** alebo **CLOCK : príkaz : príkaz : príkaz : ... : RETURN**
Po dosiahnutí času sa programový riadok ukonči a prejde sa do podprogramu.
Rýchlosť hodín je možné meniť na adrese 56866, kde je počet päťdesiatin sekundy, ktoré ubehnú za 1s času. Na adrese 56870 je 54 pre 60 s na 1 minútu, 58 pre 100 s na 1 minútu.

CLS <číslo okna > ('K'+V)

Vymaže sa aktuálne okno (WINDOW). Ak sa zadá číslo, vymaže sa okno, ak bolo definované. CLS 0 vymaže vždy celú obrazovku bez ohľadu na aktuálne okno.

CONTROL CODES

Sú zvláštne znaky, ktoré sa priamo na obrazovke nezobrazujú, avšak uskutočňujú na nej rôzne akcie.

Riadenie kurzora:

CHR$ 2	Kurzor vľavo	Nezávislé od WINDOW
CHR$ 3	Kurzor vpravo	Nezávislé od WINDOW
CHR$ 4	Kurzor nadol	Nezávislé od WINDOW
CHR$ 5	Kurzor nahor	Nezávislé od WINDOW
CHR$ 8	Kurzor vľavo	Závislé od WINDOW
CHR$ 9	Kurzor vpravo	Závislé od WINDOW
CHR$ 10	Kurzor nadol	Závislé od WINDOW
CHR$ 11	Kurzor nahor	Závislé od WINDOW
CHR$ 12	Vymaž znak	Závislé od WINDOW
CHR$ 15	Zvláštny ENTER (CR/LF)	Závislé od WINDOW

Riadiace znaky pre obrazové bloky:

CHR$ 0 Nasleduje 8 byte grafické informácie
CHR$ 1 Nasleduje 1 byte atribút a 8 byte graf. informácie

COPY pre reťazce a polia ('K'+Z)

Veľmi úzko súvisí s príkazom JOIN. Preto bude popísaný až tam.

CSIZE šírka <,výška> ('G'+SHIFT 8)

Príkaz riadi veľkosť písma pre príkazy PRINT, PLOT, LIST. Hodnoty sú v pixloch. Normálna hodnota je 8,8. Hodnoty 255, 176 vedú k jednému znaku na obrazovke. Ak sa zadá iba jedna hodnota, mení sa veľkosť znaku v oboch smeroch úmerne.
Pri malej rozteči je potrebné prejsť do OVER 1, aby sa znaky neprepisovali. Potom je možné dosiahnuť hustotu 85 znakov na 1 riadok príkazom CSIZE 3,7:OVER 1 Dobre čitateľný je ešte CSIZE 4,8.
BB03 PRINT rutina je veľmi všestranná, pretože môže tlačiť na ľubovolnú pozíciu pixla. Cez CSIZE 0 sa dá prejsť do rýchlejšej ROM rutiny, ktorá je po natiahnutí zavedená, ale nevie spracovávať CONTROL CODES 0 a 1. Všetky možnosti riadenia pozície ('AT', 'TAB', ',' atď.) sú automaticky korigované na platnú veľkosť CSIZE.
UDG znaky sú pre šírku menšiu ako 6 pixlov zobrazene len ako práve polovice.
GET obrazové bloky sú preberané ako celé bloky 8*8 pixlov. Podľa zodpovedajúceho CSIZE sú teda úmerne zmenené. Je potrebné dodržať celé násobky (4,8,16,24, atd.) aby súhlasili hrany. Dá sa použiť lokálne pre 1 príkaz PRINT t.j.

```
PRINT CSIZE 16,16;" velke"
```

vytlačí text väčšími písmenami, pričom ďalšie výstupy sú ako predtým.

DEFAULT prem=hodnota <,prem=hodnota>. . . ('G'+SHIFT 2)
DEFAULT = SAVE/LOAD zariadenie

1) Účinok sa podobá LET. Popis viď pri procedúrach. Využije sa v prípade, ak nebola danej premennej iným spôsobom priradená hodnota.

2)
DEFAULT = m Microdrive 1
DEFAULT = M1 Microdrive 1
DEFAULT = n5 Sieť
DEFAULT = B RS 232, kanál B
DEFAULT = t Páska (štandartne)

DEF KEY 1-znak reťazec ; reťazec <+":"> ('G'+SHIFT 1)
DEF KEY 1-znak reťazec : príkaz : príkaz : . . . <:>

Definuje sa reťazec, ktorý bude klávesa generovať. Pritom je možné, aby boli hneď vykonané, alebo len vložené do editačného riadku a až po ENTER odoslané. Ak je posledný znak reťazca, alebo príkazov ':', potom sa ponechá v riadku a čaká sa na ukončenie.
Aby boli funkčné klávesy akceptované, treba prejsť do režimu '*' tlačítkami SYMBOL SHIFT + SPACE. Kedykoľvek je možné funkčné klávesy predefinovať. Ak sa vloží prázdny reťazec, alebo zoznam príkazov je prázdny, potom sa klávesa ruší.

DEF KEY ERASE ruší všetky definície.

Definície ležia nad RAMTOP-om teda sú chránené pred NEW.
SAVE rutina BB03 z riadku 1 odkladá aj definície kláves.

LIST DEF KEY vypíše všetky definície.

DEF PROC menoproc <parameter><,REF parameter>. . . **('G'+1)**
DEF PROC menoproc DATA

Podrobnosti viď v stati o procedúrach. Príkazom začína definícia procedúry. Príkaz musí byť vždy sám v jednom riadku. Meno procedúry môže byť rovnaké ako meno premennej a nebudú zamenené. Môže obsahovať ľubovoľné znaky, ale musí začínať písmenom.
Za menom nasleduje zoznam parametrov, alebo kľúčové slovo DATA. V zozname môžu byť aj mená polí (b() a pod.), ale pred nimi musí byť REF.

DELETE <číslo riadku 1>TO <číslo riadku 2> **('G'+7)**
DELETE menopoľa (slicer)
DELETE menoreťazca (slicer)

1) Vymaže sa úsek programu od riadku 1 po riadok 2. Ak chýba číslo riadku 1 dosadí sa číslo prvého riadku väčšieho ako 0. Ak chýba číslo riadku 2, berie sa posledný riadok programu.
DELETE TO vymaže celý program okrem riadku 0, hodnoty premenných sú zachované. Zadané riadky musia existovať, ináč je chyba. Príkaz môže byť aj v programe. Nedoporučuje sa, aby bol v podprograme, procedúrach, cykloch. Ak má zrušiť aj sám seba, musí byť posledným príkazom bloku.
2) Príkaz vymaže časť poľa, reťazca alebo celé. Pre reťazec sa nedostane dĺžka 0, ale zruší sa v tomto prípade úplne.

DO ('G'+D) DO WHILE podmienka **('G'+J)**
DO UNTIL podmienka **('G'+K)**

Pozri aj LOOP, EXIT IF. DO a LOOP dávajú možnosť programovať cykly bez potreby skokov.
a) **DO . . . LOOP** = Nekonečná slučka

b) **DO WHILE podmienka . . . LOOP** = Cyklus sa vykonáva len dovtedy, kým je podmienka pravdivá. Potom sa už nevykoná a pokračuje sa za LOOP.

c) **DO UNTIL podmienka . . . LOOP** = Cyklus sa vykoná len v prípade, že podmienka nie je splnená. V opačnom prípade sa vykonajú príkazy po LOOP a pokračuje sa ďalej. Adresa každého DO je uložená do zásobníka, preto z tohto cyklu sa nesmie vyskočiť GO TO, ale iba použitím EXIT IF poprípade pomocou POP.

DPOKE adresa, číslo **('G'+P)**

Pozri aj funkciu DPEEK. DPOKE je dvojitý POKE a zodpovedá pomalšiemu:

```
POKE adresa, číslo - INT(číslo/256)*256
POKE adresa+1, INT(číslo/256)
```

Dolný byte sa uloží na adresu a horný na adresu+1. Je možné spracovávať čísla 0 - 65535.

DRAW TO x,y <,uhol> **('K'+W SYMBOL SHIFT+F)**

Na rozdiel od pôvodného relatívneho DRAW, vykreslí sa čiara do bodu x,y.

EDIT <čísloriadku> **(0 alebo 'G'+SHIFT 5)**
EDIT reťazcová premenná
EDIT ; číselná premenná

1) Príkaz nezodpovedá starému tlačítku EDIT (Stále sa dá používať). Je možné zadať číslo riadku, ktorý má byť editovaný. Ak ho nezadáte, bude sa editovať riadok s programovým kurzorom.
2,3) Príkaz umožňuje meniť hodnoty premenných, pričom pôvodná hodnota sa presunie do editačnej zóny a po editovaní sa uloží nová. Syntax editácie je podobná INPUT (čo sa týka riadiacich znakov a prípadného výpisu).

ELSE príkazy **('G'+E)**

Príkaz je súčasťou štruktúry IF - THEN - ELSE. Ak je podmienka za IF nepravdivá, nevykonajú sa príkazy za THEN, ale pokračuje sa za najbližším ELSE. Pred ELSE musí byť ':'.

Príklady: (písmená sú pre prehľad, nie v programe)
a)
```
IF (a)-THEN (a) príkazy:
IF (b)-THEN (b) príkazy: ELSE (b) príkazy
```

b)
```
IF (a)-THEN (a) príkazy:
IF (b)-THEN (b) príkazy: ELSE (b) príkazy:
ELSE (a) príkazy
```

c)
```
IF (a)-THEN (a) príkazy: ELSE (a) príkazy:
IF (b)-THEN (b) príkazy: ELSE (b) príkazy
```

Ak bolo ELSE použité bez IF, funguje ako REM.

END PROC **('G'+3)**

Pozri aj odsek o procedúrach. Príkaz ukončuje definíciu procedúry. Ak sa nájde DEF PROC pri behu programu, preskočí sa po najbližšie END PROC.

EXIT IF podmienka **('G'+I)**

Pozri aj DO - LOOP. Príkaz je súčasťou štruktúr DO - LOOP. Pri splnenej podmienke program pokračuje za zodpovedajúcim LOOP.

FILL x,y **('G'+F)**
FILL INK farba; x,y
FILL PAPER farba; x,y

Pozri funkciu FILLED()
1) alebo 2) Vyplní súvislú plochu bodov farby PAPER farbou INK.
3) Vyplní súvislú plochu farbou PAPER. X,y je jeden bod súvislej oblasti. Ak už má farbu INK, nič sa nestane.

Príklad: (Vyplnenie kružnice)
```
CIRCLE 128,87,30: FILL 128,80
```

Treba však dať pozor na obmedzenie hardwarom Spectra, kde 8*8 pixlov môže mať len 2 farby (PAPER a INK). Počet pixlov nastavených cez FILL je možné zistiť funkciou FILLED().

GET Číselná alebo reťazcová premenná ('G'+G)
GET reťazcová premenná, x,y <,šírka ,výška><;typ>

1) Čítanie z klávesnice. Prečíta sa jeden znak s klávesnice, pričom sa čaká na stlačenie klávesy a nečaká sa na 'ENTER'. Ak je daná reťazcová premenná, vloží sa znak. Do číselnej sa vloží pre čísla 0-9, potom A=10, B=11, atď. Vhodné pre programy s MENU riadením.
2) Z obrazovky sa odloží obdĺžniková oblasť sú súradnicami ľavého horného rohu x,y do reťazcovej premennej. Táto sa dá zobraziť na inom mieste cez PRINT alebo PLOT. Šírka a výška sa udáva v print pozíciách, x,y sú štandartné súradnice PLOT. Vzniknutý reťazec sa nedá vytlačiť pri CSIZE 0, ale je možné meniť rozmery cez normálny príkaz CSIZE.

Pozn.:
Reťazec pozostáva z 9 znakov: prvý je CONTROL CODE 0, udávajúci, že ďalších osem byte je kódovaných ako UDG. V prípade väčších úsekov sa automaticky vkladajú kódy riadenia kurzora.
Pre OVER 0 sa PAPER okraje vzorom prepisujú, pre OVER 1 a 2 vznikajú iné efekty. Ak sa neudá typ, potom sa predpokladá 0, t.j. bezfarebný. Preto je vždy vyfarbený na aktuálne farby INK a PAPER. Typ 1 odloží aj atribúty, preto sú pri výpise dodržané. Pozor však na hardware obmedzenie Spectra!

JOIN <číslo riadku> ('K'+SHIFT 6)
JOIN reťazec alebo pole

Pozri aj SPLIT.
1) Príkaz spojí daný riadok s nasledujúcim riadkom. Ak chýba číslo riadku, vezme sa riadok s programovým kurzorom. Touto operáciou sa ušetria 4 byte, zvýši sa rýchlosť programu.
2) JOIN presúva reťazce a polia. COPY ich kopíruje, ale v syntaxi sa nelíšia, preto sú popísané spolu.

a) **JOIN/COPY reťazec1 <slicer>TO reťazec2 <pozícia>** Pridá sa reťazec1 k reťazcu2.

Príklad:
```
10 LET a$="12345",b$="ABCDEFG"
20 JOIN a$ TO b$
30 PRINT b$: REM vypise ABCDEFG12345
40 PRINT a$: REM a$ už neexistuje = chyba
```

Ak sa v riadku 20 zmení JOIN na COPY, reťazec a$ ostane zachovaný.

b) **JOIN/COPY pole1 <slicer>TO pole2 <pozícia>**

Výhodné v prípade, ak sa pole naplnilo a je potrebné aby bolo dimenzované na väčší rozmer.

Príklad: Majme pole a$(100,30), čo nestačí a potrebujeme ďalších 20 reťazcov.
Postup: DIM b$(20,30):JOIN b$ TO a$ funkcia LENGHT(1,"a$") ukáže dĺžku 120.

Vkladaný reťazec prispôsobuje svoju dĺžku tomu, kde je vkladaný. Ak je napr. 30 znakov málo, potom utvoríme pomocné pole s potrebnou dĺžkou a pridáme k nemu krátke pole:

```
DIM b$(1,40): JOIN a$ TO b$
```

Vytvoríme pole s pôvodným názvom, ale potrebnou dĺžkou a pridáme k nemu pomocné pole:

```
DIM a$(1,40): JOIN b$ TO a$
```

Nakoniec treba vymazať vzniknuté nadbytočné reťazce:

```
DELETE a$(1 TO 2)
```

Pozor! Spracovávané polia môžu mať maximálne 2 rozmery.

KEYIN reťazec **('K'+SHIFT 4)**

Príkazom KEYIN a$ sa do programu zoberie obsah a$ ako príkazový riadok. Tak sa môže program sám modifikovať. Dôsledky tohto príkazu ležia mimo obzor autora tejto príručky.

Príklad:

```
10 LET a$="100 DATA"
20 FOR n=0 TO 9
30 LET a$=a$+STR$(PEEK n)+","
40 NEXT n
50 LET a$=a$( TO LEN a$-1):REM odobratie poslednej čiarky
60 KEYIN a$
```

Po RUN uvidíte, že program si sám vytvoril v riadku 100 príkaz DATA . . .
Príkaz INPUT "Heslo:";a$: KEYIN a$ umiestnený na začiatku programu by mohol predstavovať ochranu programov, pretože nestačí heslo, ale treba vedieť aj číslo riadku. V režime KEYWORDS 3 alebo 4 sa kľúčové slová prevedú najprv na 1-byte tokeny.

KEYWORDS číslo **('G'+8)**

Čísla 0 a 1 riadia výstup (PRINT, LIST) 2 až 4 riadia vkladanie programu.

Významy:

0 - Zobrazujú sa grafické znaky
1 - Zobrazujú sa nové kľúčové slová BB03. Nastavené po štarte.
2 - Kľúčové slová treba vkladať ako tokeny, t.j. jednou klávesou.
3 - Po vstupe je riadok otestovaný a všetky kľúčové slová sú prevedené na 1-byte. Akceptuje obidva druhy vstupu.
4 - Neexistuje 'K' kurzor. Všetky príkazy treba vypisovať.

LET premenná=výraz <,premenná=výraz>. . . **('K'+L)**

Do jedného príkazu je možné vložiť viac priradení.

LIST/LLIST <číslo riadku1>TO <číslo riadku2> **('K'+K, 'E'+V)**

Vypíše sa úsek programu. Je pritom možné niektorú z hodnôt vynechať. Dosadia sa podobne ako pri DELETE.

LIST DATA - Všetky premenné **('K'+K 'E'+D)**
LIST VAL - Číselné premenné **('K'+K 'E'+J)**
LIST VAL$ - Reťazcové premenné **('K'+K 'E'+SHIFT J)**

1) Vypíše sa aktuálny obsah premenných.
2) Vypíše premenné v poradí:
   1 - číselné polia
   2 - riadiace premenné FOR cyklov
   3 - premenné s 1 písmenovým menom
   4 - premenné s viacpísmenovým menom
3) Vypíše ďalej:
   5 - reťazcové polia
   6 - reťazcové premenné

Pre polia sa vypíšu len rozmery, nie obsah. Pre reťazce len prvých 15 znakov.

LIST DEF KEY **('K'+K, 'G'+SHIFT 1)**

Príkaz vypíše definície všetkých nadefinovaných funkčných kláves.

LIST FORMAT číslo **('K'+K 'E'+SHIFT 0)**

Príkaz nastaví režim, v ktorom bude robený výpis programu. Na začiatku je nastavené 0.

Význam čísel:
0 - Klasický výpis ZX Spectra, avšak riadky dlhšie ako 32 znakov pokračujú na ďalšom riadku až od piatej pozície.
1 - Pre štruktúry robí odskok 1 medzera.
2 - Pre štruktúry robí odskok 2 medzery.
3 - Ako 0, ale bez čísel riadkov.
4 - Ako 1, ale bez čísel riadkov.
5 - Ako 2, ale bez čísel riadkov.

Aby bol výpis pekný, je dobre vložiť za každým THEN aj ELSE: .

LIST PROC meno **('K'+K 'G'+2)**

Vypíše sa text celej procedúry s menom meno.
Pozn.: Po výpise obsahuje adresa 23625 číslo prvého a adresa 57358 číslo posledného vypisovaného riadku. Toto je možné využiť pri programovo riadenom DELETE alebo RENUM. Hodnotu treba čítať funkciou DPEEK.

LIST REF údaj **('K'+K 'G'+SHIFT 7)**

Príkaz vypíše čísla riadkov, na ktorých sa nachádza referencia na údaj. Môže to byť meno premennej, číslo, postupnosť znakov.

LOCAL premenná <,premenná>. . . **('G'+SHIFT 3)**

Vytvoria sa špeciálne premenné, ktoré sú známe len danej procedúre. Formálne parametre sú pre procedúru automaticky lokálne. V procedúre sa môže vyskytovať viac príkazov LOCAL. Ak má byť lokálne pole, musí byť deklarované postupnosťou príkazov:

```
LOCAL p(): DIM p(...)
```

LOOP **('G'+L)**
LOOP WHILE podmienka **('G'+L 'G'+J)**
LOOP UNTIL podmienka **('G'+L 'G'+K)**

Príkaz ukončuje štruktúru DO - LOOP. Podobne ako DO je možné vložiť príkazy pre podmienené ukončenie cyklu.

MERGE <mechanizmus; "meno" **('E'+SHIFT T)**

Pri Microdrive sa BB03 chová pre samoštartujúci program vtiahnutý cez MERGE korektne.

MOVE **('E'+SHIFT 6)**

Môže na Microdrive teraz prenášať aj programy a CODE súbory.

GO TO/GO SUB ON výraz; riadok1, riadok2, . . . **('K'+G/H, 'G'+O)**
ON výraz: príkaz: príkaz: príkaz: . . . príkaz **('G'+O)**

1) Skočí sa na riadok, ktorého číslo je v poradí na pozícii rovnej hodnote výrazu.
2) Vykoná sa príkaz, ktorého poradové číslo je rovné hodnote výrazu.
Ak je hodnota výrazu mimo počtu riadkov alebo príkazov, pokračuje sa nasledujúcim príkazom alebo riadkom.

ON ERROR číslo riadku **('G'+N)**
ON ERROR: príkaz: príkaz: . . .

1) V prípade chyby (okrem 0 OK a 9 STOP Statement) sa zavolá podprogram na uvedenom čísle riadku.
2) Vykonajú sa príkazy na riadku. Ak nie je chyba, preskočia sa. Vlastné spracovanie je vypnuté počas behu podprogramu, obnoví sa pri RETURN. CONTINUE vráti program na riadok, ktorý spôsobil chybu, ale nezapne sa spracovanie chyby, takže sa teraz vypíše. (Pred CONTINUE treba vybrať návrat príkazom POP).
ON ERROR 0 vypne vlastné spracovanie chyby. V podprograme existujú tri premenné:
line = číslo riadku s chybou
stat = poradové číslo príkazu v riadku
error = číslo chyby (viď príloha C)
Môžu sa používať aj v programe, ale ak sa aktivuje ON ERROR alebo TRACE, sú prepísané.

OVER číslo **('E'+SHIFT N)**

Príkaz akceptuje okrem parametrov 0,1 aj 2. Jeho účinok je OR. To znamená, že znaky sa ani neprepisujú jeden druhým, ani sa nevykonáva XOR po pixloch.

PLOT x,y <;reťazec> **('K'+Q)**

Okrem zobrazenia bodu je možné na mieste x,y zobraziť aj reťazec. Môže sa jednať o normálny reťazec alebo o reťazec, vytvorený pomocou GET. X,y určujú ľavý horný roh reťazca. Okrem INK, PAPER a pod. je možné vložiť aj CSIZE na zväčšenie alebo zmenšenie reťazca.

Príklad:
```
PLOT CSIZE 32; INK 2; 100,88;"HI!"
```

Výhoda príkazu je rovnaký súradnicový systém pre grafiku aj znaky.

POKE adresa, reťazec **('K'+O)**

Okrem čísla 0 - 255 je možné vkladať aj reťazce. Ekvivalent PEEK nahrádza funkcia **MEMORY$()(slicer).**

POP <číselná premenná> **('G'+Q)**

Príkaz vyberá hodnotu zo stacku (GO SUB, DO, PROC). Ak je udaná premenná, vloží sa do nej číslo riadku. Môžete teraz vyskočiť z podprogramov, cyklov a procedúr.

Príklad:

```
100 GOSUB 500
110 STOP
500 POP loc
510 PRINT "Podprogram volaný z riadku ";loc
520 GO TO loc+1
```

Ak by na riadku 520 bol príkaz RETURN, objavila by sa chyba 7 RETURN without GOSUB.

PROC meno <parameter><,parameter>. . . **('G'+2)**

Príkaz volá procedúru. Je to obdoba GO SUB, ale netreba poznať číslo riadku. Kľúčové slovo PROC možno vynechať. Podrobnejší popis viď v stati o procedúrach.

READ LINE <reťazcová premenná>. . . **('E'+A 'E'+SHIFT 3)**

Príkaz umožňuje prečítať do premenných hodnoty bez toho, aby museli byť v DATA-ch uvedené v " ".

REF referencia **('G'+SHIFT 7)**
REF premenná

1) Týmto príkazom sa prezrie celý program na výskyt danej referencie. Referencia je: premenná, číslo, postupnosť znakov . . . Ak sa nájdu znaky, presunie sa daný riadok do editačnej oblasti s kurzom za nájdenou referenciou. Ak riadok nechcete opraviť, treba odpovedať 'ENTER'. Ak stlačíte ENTER ešte raz, hľadá sa ďalší výskyt referencie. Ak počas hľadania vložíte nejaký príkaz, hľadanie sa preruší.

Príklady:

REF a$ - hľadá sa: a$
REF sum - hľadá sa: -sum-
REF "SUM" - hľadá sa: SUM
REF 1 - hľadá sa: 1 (vrátane neviditeľnej) 5-byte formy
REF "1" - hľadá sa: 1
REF 12*4 - hľadá sa: 12 (vrátane...)*4 (vrátane...)
REF (a$) - hľadá sa: obsah a$
REF (x) - hľadá sa: obsah x včítane...

Veľká - malá abeceda nehrá rolu.

2) V zozname formálnych parametrov určí premenné, ktorých hodnota je odovzdaná adresou. Znamená to, že akákoľvek zmena obsahu danej premennej vo vnútri procedúry bude známa aj po jej ukončení. Polia možno odovzdávať len cez REF.

RENUM <*><riadok1 TO riadok2><LINE start><STEP krok> **('G'+4)**

Celý, alebo časť programu je možné prečíslovať, presunúť úseky, alebo skopírovať na iné miesto programu.
a) RENUM Prečísluje program tak, že čísla riadkov budú v rozostupoch po 10, prvé číslo je 10.
b) RENUM od TO do Prečíslovanie úseku.
c) * Starý blok sa po presunutí nemaže teda, skopíruje sa.

Príklady:

RENUM	- celý program
RENUM LINE 100 STEP 20	- celý program 100,120,140
RENUM 100 LINE 300	- riadok 100 na 300
RENUM 1540 TO LINE 2000	- všetko od 1540 sa presunie na 2000 a ďalej

Príkaz prečísluje všetky referencie v GO TO, GO SUB, RESTORE, RUN, ON, ON ERROR, TRACE, LIST, LINE a DELETE. Prípadne čísla riadkov v CLOCK je treba zmeniť ručne. Ak sa stane, že číslo riadku v GO TO, GO SUB je výraz, napr:

```
100 PRINT: GO TO a
```

výpise sa správa: 'Failed at . . : . .' (100:2)
Pozn.: RENUM využíva pre tabuľky pamäť obrazu. Preto je možné spracovávať aj dlhé programy.

ROLL kód smeru <,pixle><;x,y;šírka,výška> **('G'+R)**

Pozri aj SCROLL.
Príkaz presúva celý alebo časť obrazu o daný počet pixlov v ľubovoľnom smere. To čo z jednej strany vyjde von, na druhej strane sa objaví (na rozdiel od SCROLL). Obraz sa posúva rýchlejšie, ak sa zadá väčší počet pixlov. Najrýchlejší presun je pri presune vo vodorovnom smere o 4, alebo 8 pixlov, keď sa využívajú priamo inštrukcie Z80.

kódsmeru	*smer*	*pohybuje sa*
1	vľavo	atribúty
2	nadol	atribúty
3	nahor	atribúty
4	vpravo	atribúty
5	vľavo	pixle
6	nadol	pixle
7	nahor	pixle
8	vpravo	pixle
9	vľavo	obidve
10	nadol	obidve
11	nahor	obidve
12	vpravo	obidve

Keďže sa atribúty môžu presúvať len po 8 pixloch, ignoruje sa počet zadaných pixlov.
V prípade presunov častí obrazu sa zadáva šírka v PRINT pozíciách (1-32), výška v pixloch (1-176). Ak sa má presúvať v aktuálnom okne, stačí iba jednoduchá forma príkazu.

SAVE <riadok1>TO<riadok2;><mechanizmus;> "meno" **('K'+S)**
SAVE DATA <mechanizmus; > "meno"

Odložiť je možné nielen celý, ale aj časť programu, prípadne len premenné (DATA). Pri naťahovaní cez LOAD treba dať pozor na to, že sa vymaže pôvodný program (aj riadok 0).
Takto je možné zapísať aj samotné procedúry do formy knižnice. Pritom je vhodné pred uložením urobiť RENUM na vysoké čísla riadkov a po vtiahnutí RENUM na požadované miesto, aby sa dali vložiť ďalšie procedúry.

SCROLL <kódsmeru><,pixle><;x,y;šírka,výška> **('G'+S)**

Pozri aj ROLL.
Pri volaní bez parametra sa posúva celý obraz o riadok nahor. Ak sa použije kód 5-8, presúva sa celé aktuálne okno o 1 pixel. Informácia vysunutá z okna sa stráca a na druhom konci vstupujú prázdne riadky alebo stĺpce.

SORT pole alebo reťazec **('G'+M)**
SORT INVERSE pole alebo reťazec **('G'+M 'E'+SHIFT M)**

Príkaz usporiada polia, alebo reťazce vo vzostupnom, alebo zostupnom poradí podľa čísel, alebo podľa abecedy.
Pre polia je možné použiť jeden alebo dva rozmery. Pritom numerické sú usporiadané od najväčšieho prvku k najmenšiemu.

SPLIT **(SHIFT W)**

Pracuje sa s riadkami, ktoré sú v dolnej časti obrazu. Pritom stačí presunúť kurzor za :, stlačiť <> a riadok vľavo sa po 'ENTER' uloží do programu. Časť riadku vpravo zostane s tým istým číslom v editačnej zóne.

Príklad:
```
10 PRINT "hallo": GO TO 20: <> PRINT "Príklad"
```

Stlačiť ENTER, do programu sa uloží:

```
10 PRINT "hallo": GO TO 20
```

V editačnom riadku zostane:

10 (kurzor) PRINT "Príklad"

TRACE číslo riadku **('G'+T)**
TRACE príkaz: príkaz: príkaz: . . . :RETURN

Príkaz sa využíva pri ladení programov. Umožňuje napr. výpis riadkov počas spracovania.

Príklad:
```
TRACE: LIST lino TO lino: PAUSE 0: RETURN
```

1) Pred každým riadkom sa vykoná na zadanom čísle riadku podprogram. Ďalšie príkazy za TRACE sa na trasovanie nevzťahujú.
2) Príkazy stojace za TRACE sa vykonajú pred každým riadkom programu ako podprogram. Počas vykonávania trasovacieho podprogramu sa režim trasovania vypne. TRACE 0, RUN a CLEAR vypnú tento režim úplne.
Ako podprogram v trasovacom režime môže byť použitá rutina, určená aj na iné úlohy.

Príklad: Výpis čísla riadku, príkazu a premennej a$ na určité miesto.

```
9000 LET stlpec=PEEK 23688,riadok=PEEK 23689:REM odloženie
pôvodnej pozície tlače
9010 PRINT AT 0,0; INVERSE 1; lino;":";stat,"A$= ";a$
9020 POKE 23688, stlpec: POKE 23689, riadok:RETURN
```

UNTIL podmienka **('G'+K)**

Pozri DO - LOOP
Využíva sa ako podmienka riadenia cyklu DO LOOP.

USING **('G'+U)**

Využíva sa pri formátovaní tlače:
PRINT USING formátovací reťazec; číselný výraz

Tu bude tiež popísaná funkcia USING$.
Príkaz USING a funkcia USING$ umožňujú vytvárať čísla v zadanej forme. Pritom USING je možné použiť len spolu s PRINT a USING$ má širšiu aplikáciu. Výsledkom USING je výstup na obrazovku, výsledkom USING$ je reťazec.

Vo formátovacom reťazci sa využíva:
- medzera alebo číslica pred desatinnou bodkou
0 - nula alebo číslica pred desatinnou bodkou
Za desatinnou bodkou sú obe správne zaokrúhlené na desatinné miesta.
. - desatinná bodka
písmeno - zobrazí sa

Príklady:
"##.#" 12.3
"###.#" 12.3
"000.00" 012.35
"00" 12
"00.00KCS" 12.35KCS
"0.00" %..3 prekročený formát

Pozn.: USING nespracúva čísla v exponenciálnom tvare.

Príkazy:
```
PRINT USING A$; číslo
PRINT USING$(a$,číslo) sú rovnocenné.
```

VERIFY <riadok1 TO riadok2;><mechanizmus;> meno **(E'+SHIFT R)**
VERIFY DATA <mechanizmus;> meno

Popísané v SAVE.

WHILE **('G'+L)**

Príkaz určuje podmienku ukončenia cyklu DO - LOOP. Pozri tam.

WINDOW číslo <,x,y,šírka,výška> **('G'+5)**

Pozri aj CLS, CSIZE

Príkaz vytvára pre zobrazenie výstupu pravouhlé okná. Príkazy PRINT a LIST je možné smerovať len do nich bez vplyvu na ostatnú časť obrazu. Každé okno má svoju vlastnú pozíciu PRINT, OVER, BRIGHT, FLASH, CSIZE, INK a PAPER.

Ako číslo okna treba zadať 1 - 127. Takto vytvorené okná majú svoje informácie odložené za RAMTOP, a sú chránené pred NEW. Špeciálny prípad je okno nula, ktoré je aktívne po vtiahnutí. Je pevne určené ako celá obrazovka.

Príklad:

```
WINDOW 1,0,175,128,176
```

Okno 1 bude mať ľavý horný roh v pozícii 0,175, je široké 128 pixlov a vysoké 176 pixlov (ľavá polovica obrazu). Atribúty sú ako v 0.

Okno sa aktivuje príkazom WINDOW 1. Keď sa aktívne okno deaktivuje, je CSIZE a aktuálna PRINT pozícia odložená za RAMTOP. Príkaz **WINDOW ERASE** vymaže definície okien.

XOS, YOS, XRG, YRG

Pozri prílohu D.

Uvedené slová nie sú kľúčové, ale špeciálne premenné. Tieto umožňujú meniť pre príkazy PLOT, DRAW a CIRCLE počiatok súradnicového systému a rozmery obrazovky.

XOS,YOS - Súradnice počiatku
XRG,YRG - Dĺžka úsekov na osiach x,y

Pozor na oblúky a kružnice pri DRAW a CIRCLE !

FUNKCIE

V BB03 máte k dispozícii 26 nových funkcií. Tieto sú definované v riadku 0. Ich skutočné definície sú však z dôvodu rýchlosti v strojovom kóde BB03. Pri výpise sú zobrazené ako kľúčové slová, t.j. miesto FN S$ sa zobrazí STRING$ a kurzor preskočí ako pri kľúčovom slove Ak máte programy, ktoré náhodou používajú tieto definície, treba ich premenovať (Napr. ALTER). Mená je možné vkladať aj vypísaním alebo ako FN písmeno (alebo $.

Pri vťahovaní programov nenapísaných v BB03 treba použiť MERGE, aby nebol vymazaný riadok 0 s definíciami. Pre programy z BB03 je možné použiť LOAD, pretože SAVE odložil automaticky aj riadok 0.

Funkcia	*Skutočné meno*	*Funkcia*	*Skutočné meno*
AND	FN A()	MEM	FN M()
BIN$	FN B$	MEMORY$	FN M$
CHAR$	FN C$	MOD	FN V()
COSE	FN C()	NUMBER	FN N()
DEC	FN D()	OR	FN O()
DPEEK	FN P()	RNDM	FN R()
EOF	FN E()	SCRN$	FN K()
FILLED	FN F()	SHIFT$	FN Z$
HEX$	FN H$	SINE	FN S()
INARRAY	FN U()	STRING$	FN S$
INSTRING	FN I()	TIME$	FN T$
ITEM	FN U()	USING$	FN U$
LENGTH	FN L()	XOR	FN X()

AND (číslo,číslo) **FN A(číslo,číslo)**

Funkcia robí bitový OR medzi dvoma číslami 0 - 65535.

BIN$ (číslo) **FN B$(číslo)**

Funkcia prevedie číslo 0 - 255 do 8-znakového reťazca, ktorý predstavuje dvojkovú reprezentáciu čísla a číslo 256 - 65535 do 16 znakového.
Ak nechcete 0 a 1, vložte na adresu 62865 znak za 1, a na 62869 znak miesto 0.

CHAR$ (číslo) **FN C$(číslo)**

Pozri aj NUMBER (reťazec).
Funkcia prevedie číslo 0 - 65535 na dvojznakový reťazec. Tak je možné ukladať číslo namiesto 5-byte na 2-byte.
Funkcia NUMBER je opačná a vytvára z reťazca naspäť číslo.

Príklad:

```
100 DIM a$(500,2)
110 FOR e=1 TO 500: LET a$(e)=CHAR$(e*10): NEXT e
120 PRINT "Pole je vytvorene"
130 PAUSE 0
140 FOR e=1 TO 500: PRINT e,NUMBER(a$(e)): NEXT e
```

Pole obsadí len 1000 byte oproti 2500 byte normálne. SORT pracuje aj na takto skompresované pole.

COSE (číslo) **FN C(číslo)**

Funkcia je ekvivalent COS, ale je cca 6-krát rýchlejšia.

DEC (reťazec) **FN D(reťazec)**

Pozri aj HEX$ (cislo).
Funkcia vráti dekadický ekvivalent 1 až 4-znakového hexadecimálneho reťazca. Nerozoznáva sa veľká a malá abeceda.

DPEEK (adresa) **FN P(adresa)**

Dvojitý PEEK pre danú a nasledujúcu adresu. Pomaly v BASIC-u vyzerá takto:

```
PEEK adresa+256*PEEK (adresa+1)
```

EOF (číslo kánala) **FN E(číslo kánala)**

Funkcia dáva pri čítaní dát z Microdrive 0, ale v prípade posledného údaja dá 1.

FILLED () **FN F()**

Vráti sa počet pixlov, ktoré boli nastavené posledným príkazom FILL.

HEX$ (číslo) **FN H$(číslo)**

Pozri aj DEC (reťazec)
Vložená hodnota sa vyčísli a prevedie sa do hexadecimálneho reťazca.
Pre čísla -256 <x <256 má reťazec 2 znaky
-65536 <x <65536 má reťazec 4 znaky

INARRAY (reťazcové pole(štart<,slicer>),hľadaný reťazec) **FN U(.)**

Pozri INSTRING.
Funkcia pretestuje reťazcové pole na výskyt reťazca a vráti jeho hodnotu alebo 0, ak ho nenájde.
V podstate sa jedná o INSTRING aplikovaný na polia. V prípade, že niektoré znaky reťazca sú nepodstatné, možno na ich miesto vložiť '#'.
Pozn.: Možné použiť pre polia max 2 rozmerov.

INSTRING (štart,reťazec1,reťazec2) **FN I(...)**

Funkcia prehľadá reťazec1 na výskyt reťazca2 počnúc od pozície štart. Ak je reťazec nájdený, funkcia vráti pozíciu prvého znaku, kde sa v reťazci1 začína reťazec2. Ak sa nenájde, vráti 0. Znaky v reťazci2, ktoré nie sú podstatne, je možné nahradiť znakom '#'.

Príklad:

```
PRINT INSTRING (1,a$,"M##ER")
```

Vytlačí sa pozícia ľubovoľného slova tvaru MAYER, MILER, MEIER atď.

ITEM() **FN T()**

Pozri odsek o procedúrach.
Funkcia dáva informáciu o ďalšej položke zo zoznamu, ktorá sa má čítať READ. Je určená predovšetkým pre procedúry, avšak dá sa použiť aj pre normálne príkazy DATA, READ.
Vracia hodnotu
0 - Všetky položky boli prečítané
1 - Nasledujúca položka bude reťazec
2 - Nasledujúca položka bude číselná

LENGTH (n,"menopoľa") **FN L(n,"menopoľa)**

Funkcia dá informáciu o dĺžke poľa, ktorá je v BB03 veľmi dôležitá, pretože sa môže meniť bez straty informácií. Taktiež je schopná dať adresu, kde sa pole alebo reťazec nachádza.
Pre n = 1 sa vráti veľkosť 1. rozmeru
2 sa vráti veľkosť 2. rozmeru
0 vráti adresu prvého prvku poľa alebo reťazca.

MEM () **FN M()**

Vráti sa počet voľných byte v pamäti. Zátvorky musia zostať prázdne.

MEMORY$ () **FN M$()**

Pozri aj POKE reťazcov.
Funkcia má k dispozícii celú pamäť od 0 do 65535 ako reťazcovú premennú. V skutočnosti však z technických dôvodov nie je prístupná adresa 0 a posledné 3 bunky.
Takže LEN (MEMORY$()) = 65532.
Obrazovku je možné odložiť príkazom: LET a$=MEMORY$ ()(16384 TO 22527)
Prehľadá celú pamäť na ľubovoľný reťazec: PRINT INSTRING(1,MEMORY$(),"asdfg")

MOD (číslo1,číslo2) **FN V(číslo1,číslo2)**

Výsledok je modulo delenie čísla 1 číslom 2 (zvyšok).

NUMBER (reťazec) **FN N(reťazec)**

Pozri aj CHAR$ (číslo)
Prevedie 2-znakový reťazec na číslo 0-65535. Ak argument nemá 2 znaky, vznikne chyba.

OR (číslo1,číslo2) **FN O(číslo1,číslo2)**

Urobí sa logická bitová operácia medzi číslom1 a číslom2.

RNDM (číslo) **FN R(číslo)**

Vráti sa náhodné číslo 0 - 1 ak číslo = 0
Ak je číslo väčšie ako 0, výsledok je celé číslo 0 - číslo

RANDOMIZE (číslo) nastaví pevnú sekvenciu náhodných čísel.

SCRN$(riadok,stlpec) **FN K$(riadok,stlpec)**

Pracuje podobne ako SCREEN$, avšak pozná aj UDG znaky.

SHIFT$ (číslo,reťazec) **FN Z$(číslo,reťazec)**

Všeobecne použiteľná funkcia na konverziu reťazcov. V závislosti od čísla sú robené zmeny v reťazci.

1 - Všetky znaky abecedy na veľké
2 - Všetky znaky abecedy na malé
3 - Inverzia malých a veľkých písmen
4 - Zo všetkých riadiacich znakov okrem CHR$ 13 "."
5 - 4 + zmení CHR$ 128-255 na CHR$ 0-127 (potlačí kľúčové slová)
6 - 5 aj pre CHR$ 13
7 - Zmení tokeny na vypísaný tvar
8 - Inverzne 7
9 - 8, ale každý znak musí nasledovať kľúčové slovo
10 - 9, ale kľúčové slová písane veľkými písmenami

SINE (číslo) **FN S(číslo)**

Rýchly sínus, podobne ako COSE.

STRING$ (číslo,reťazec) **FN S$(číslo,reťazec)**

Výsledkom je číslo krát reťazec.

Príklady:

```
STRING$ (32,"-") = ----------.....---
STRING$ (4,"AB") = ABABABAB
STRING$ (3,"A"+CHR$ 13) = A
A
A
```

TIME$ () **FN T$()**

Výsledkom je čas tak, ako bol získaný príkazom CLOCK.

USING$ (formátovací reťazec,číslo) **FN U$(...)**

Podľa formátovacieho reťazca sa vytvorí reťazec obsahujúci číslo v požadovanom tvare.

XOR (číslo1,číslo2) **FN X(číslo1,číslo2)**

Funkcia urobí logickú operáciu XOR čísla1 a čísla2 po bitoch.

PRÍLOHA A
Množina znakov

Pri zapnutom režime KEYWORDS 1 sa zmení význam kláves ZX-Spectra nasledovné:

Kód	*Klávesa*	*Znak*
128	8	KEYWORDS
129	1	DEF PROC
130	2	PROC
131	3	END PROC
132	4	RENUM
133	5	WINDOW
134	6	AUTO
135	7	DELETE
136	SHIFT 7	REF
137	SHIFT 6	JOIN
138	SHIFT 5	EDIT
139	SHIFT 4	KEYIN
140	SHIFT 3	LOCAL
141	SHIFT 2	DEFAULT
142	SHIFT 1	DEF KEY
143	SHIFT 8	CSIZE
144	A	ALTER
145	B	BLANK
146	C	CLOCK
147	D	DO
148	E	ELSE
149	F	FILL
150	G	GET
151	H	BLANK
152	I	EXIT IF
153	J	WHILE
154	K	UNTIL
155	L	LOOP
156	N	SORT
157	M	ON ERROR
158	O	ON
159	P	DPOKE
160	Q	POP
161	R	ROLL
162	S	SCROLL
163	T	TRACE
164	U	USING

Znaky pre úzke písmo (CSIZE 4,8) sú v RAM od 51291 do 51626 (CHR$ 32 až CHR$ 127). Každé 2 znaky zaberajú 7 byte. Párne znaky obsadzujú 4 ľavé bity každého byte, nepárne 4 párne bity. Pre každý znak sa vytvorí 1 rada prázdnych pixlov, takže vznikne 8 radov.

PRÍLOHA B
Rozširujúce a nové chybové hlásenia:

Kód Význam	*Situácia*	
G No room for line	RENUM	Nové prečíslovanie riadkov by viedlo k neočíslovaným riadkom, alebo k riadkom s číslami väčšími ako 9999.
S Missing LOOP	DO, EXIT IF	Po EXIT IF, alebo podmienenom DO (spolu s WHILE alebo UNTIL) sa nenašiel zodpovedajúci príkaz LOOP.
T LOOP without DO	LOOP	Bol nájdený príkaz LOOP bez DO.
U No such line	DELETE	Riadok udaný v DELETE sa nenašiel.
V No POP data	POP	Bol urobený pokus prečítať číslo z prázdneho stacku pre GO SUB, PROC a DO. Znamená to, že neboli aktivované procedúry, podprogramy ani DO cykly.
W Missing DEF PROC	PROC,END PROC	Bola volaná procedúra bez definovania alebo bol nájdený príkaz END PROC bez zodpovedajúceho DEF PROC.
X No END PROC	DEF PROC	Program sa pokúsil preskočiť definíciu procedúry, ale nenašiel jej koniec.

PRÍLOHA C
Chybové kódy:

Nasledujúci zoznam obsahuje hodnoty premennej error, ktorá je vytvorená príkazom ON ERROR. Chyby 0 až 9 nie sú spracované.

Č.	*Kód*	*Hlásenie*	*Chyba*
0	0	OK	- nespracované
1	1	Next without for	Ku NEXT sa nenašlo FOR
2	2	Variable not found	Premenná neexistuje
3	3	Subscript wrong	Nesprávny index
4	4	Out of memory	Preplnená pamäť
5	5	Out of screen	Mimo obrazovky
6	6	Number too big	Príliš veľké číslo
7	7	RETURN without GOSUB	Našiel sa RETURN bez volania GO SUB
8	8	End of file	Koniec súboru
9	9	STOP statement	- nespracované Príkaz STOP
10	A	Invalid argument	Nesprávny argument
11	B	Integer out of range	Celá časť je mimo dovolený rozsah
12	C	Nonsence in BASIC	Nezmysel v BASIC-u
13	D	BREAK - CONT repeats	Prerušenie, CONTINUE bude opakovať
14	E	Out of DATA	Príkaz READ nenašiel data
15	F	Invalid file name	Nesprávne meno súboru
16	G	No room for line	Niet miesto pre riadok
17	H	STOP in INPUT	Na INPUT bol vložený STOP
18	I	FOR without NEXT	FOR nemá zodpovedajúce NEXT
19	J	Invalid I/O device	Nesprávne zariadenie V/V
20	K	Invalid colour	Nesprávna farba
21	L	BREAK into program	Prerušenie počas programu CONTINUE bude opakovať, ale pokračovať
22	M	RAMTOP no good	Nesprávny vrchol pamäte
23	N	Statement lost	Skok na stratený príkaz
24	O	Invalid stream	Nesprávny príkaz pre INTERFACE 1
25	P	FN without DEF	Volaná funkcia bez definície
26	Q	Parameter error	Chyba v parametroch
27	R	Tape loading error	Chyba pri čítaní z pásky

Beta Basic chyby:

28	S	Missing LOOP	Chýbajúci LOOP
29	T	LOOP without DO	Chýbajúce DO
30	U	No such line	Nenašiel sa riadok
31	V	No POP data	Stack je prázdny
32	W	Missing DEF PROC	Chýba definícia procedúry
33	X	No END PROC	Procedúra nemá koniec

INTERFACE 1 chyby:

43	b	Program finished	Program skončil
44	c	Nonsence in BASIC	Nezmysel v BASICu, volané pri chybe v príkazoch INTERFACE 1
45	d	Invalid stream number	Zlé číslo kanála
46	e	Invalid device expression	Zlý výraz pre zariadenie
47	f	Invalid name	Zlé meno
48	g	Invalid drive number	Nesprávne číslo zariadenia
49	h	Invalid station number	Nesprávne číslo uzla v sieti
50	i	Missing name	Chýbajúce meno
51	j	Missing station number	Chýbajúce číslo stanice
52	k	Missing drive number	Chýbajúce číslo zariadenia
53	l	Missing band rate	Nezadaná rýchlosť prenosu
54	m	Header mismatch error	Chyba v hlavičke
55	n	Stream already open	Kanál je už otvorený
56	o	Writing to a "read" file	Pokus o zápis do súboru určeného pre čítanie.
57	p	Reading a "write" file	Pokus o čítanie zo súboru určeného pre zápis
58	q	Drive "write" protect	Zákaz zápisu na daný mechanizmus
59	r	Microdrive full	Kazeta microdrive je plná