

[illegible]

(C) Butamoff 1985 92 Oxford Road, Maseley
Birmingham

- Prehľad
- Úvod
- Editovanie
- Procedúry a parametre
- Štruktúrované programovanie
- Zjednodušený prístup k zariadeniam
- Prístup k dátam - polia, triedenie, prehliadanie
- Grafické príkazy
- TOOLKIT - Jednoduchá manipulácia s programami
- Syntax príkazov

- A Znakova mozzina
- B Rozsirena s nove chybove hlaseia
- C Kody chyb (pre ON ERROR)
- D Specialne s nove premenne
- E Pripojenia tlaciarne

PRÍKAZY:

ALTER	- Zmena atributov obrazovky
	- Zmena casti programu
AUTO	- Automaticke cislovanie riadkov
BREAK	- Vylepseny BREAK
CLEAR	- Presun RAMTOP-u bez straty premennych
CLOCK	- Hodiny s Alarmom: BEEP aj GOSUB
CLS	- Vymazanie okna (WINDOW)
CONTROL CODES	- Riedenie formatu a kurzora
COPY	- Kopirovanie casti poli a retazcov
CSIZE	- Nastavenie velkosti znakov a rozostupov
DEFAULT	- Nastavenie standartnych hodnot premennych
	- Vyber SAVE/LOAD zariadenia
DEF KEY	- Definicia funkcej klavesy
DEF PROC	- Definicia procedury
DELETE	- Vymazanie casti programu
	- Vymazanie casti poli a retazcov
DO	- Zaciatok cyklu DO - LOOP
DPOKE	- Dvojity poke (2 Byte)
DRAW TO	- Vykreslenie ciary do daneho bodu
EDIT	- Editovanie riadku programu
	- Editovanie obsahu premennej

ELSE	- Cast IF - THEN - ELSE struktury
END PROC	- Ukoncenie procedury
EXIT IF	- Podmienene opustenie cyklu DO - LOOP
FILL	- Vyplnenie oblasti obrazovky
GET	- Vstup stlacenej klavesy
JGJN	- Odloženie časti obrazu do premennej
KEYIN	- Spajanie riadkov programu
KEYWORDS	- Spajanie polí a reťazcov
LET	- Vstup reťazca ako programového riadku
LIST, LLIST	- Nastavenie formátu vstupu
LIST DATA	- Viacnásobne priradenie
LIST VAL	- Vypis časti programu
LIST VALS	- Vypis obsahu všetkých premenných
LIST DEF KEY	- Vypis obsahu číselných premenných
LIST FORMAT	- Vypis obsahu reťazcových premenných
LIST PROC	- Vypis obsahu definovaných klaves
LIST REF	- Listing programu v rôznych formátoch
LOCAL	- Vypis danej procedury
LOOP	- Vypis referencií na premennú
MERGE	- Definícia lokálnej premennej
MOVE	- Koniec cyklu DO - LOOP
ON	- Funguje autostart (Microdrive)
ON ERROR	- Pracuje pre všetky dátové typy (Microdrive)
OVER	- Voľba príkazu alebo čísla riadku
PLOT	- Spracovanie chyby
POKE	- Logický OR s obsahom obrazovky
POP	- Vypisovanie reťazcov od každého pixlu
PROC	- Ukladanie reťazcov do pamäte
READ LINE	- Vybratie BASIC stacku (GO SUB, DO-LOOP, PROC)
REF	- Volanie procedury (Mozno vypustiť)
RENUM	- Citanie reťazcov bez úvodzoviek
ROLL	- Vyhľadanie referencií v programe
SAVE	- Nastavenie premennej ako parametra
SCROLL	- Precisľovanie, presuny a kopírovanie
SHORT	- Posun časti obrazu
SPLIT	- Zapis časti programu, premenných
TRACE	- Posun časti obrazu
UNTIL	- Triedenie polí, reťazcov
USING	- Rozdelenie programového riadku
VERIFY	- Sledovanie behu programu
WINDOW	- Použitie v cykloch (Kým neplatí)
WHILE	- Formátovaný výstup dát
XOS YOS	- Overenie časti programu, premenných
XRG YRG	- Nastavenie formátov okna
	- Urcenie aktuálneho okna
	- Použitie v cykloch (Kým platí)
	- Nastavenie počiatku súradnicového systému,
	sírky a dĺžky obrazovky

FUNKCIE:

AND	- Bitový AND
BIN\$	- Konverzia Decimal - Bin
CHAR\$	- Konverzia Integer - String
COSE	- Rychlý kosínus
DEC	- Konverzia Hex - Decimal
DPEEK	- Dvojité PEEK
EOF	- Koniec súboru (Microdrive)
FILLED	- Veľkosť zaplneného úseku

HEX\$	- Konverzia Decimal - Hex
INARRAY	- Prehľadanie pola
INSTRING	- Prehľadanie retazca
ITEM	- Zistenie typu dat v READ
LENGHT	- Dĺžka a adresa pola
MEM	- Veľkosť voľnej pamäte
MEMORY\$	- Čela pamät je ako premenná
MOD	- Funkcia MODULO
NUMBER	- Konverzia String - Integer
OR	- Bitový OR
RNDM	- Vylepseny RND (Da aj celé čísla)
SCRN\$	- Vylepseny SCREEN\$ (Pozna aj UDG)
SHIFT\$	- Ďalšie kódy, písmena veľké / malé, atd.
SINE	- Rychlý sinus
STRING\$	- Viacnásobný retazec
TIME\$	- Aktuálny čas
USING\$	- Formát čísla
XOR	- Bitový XOR

U V O D

Vďaka BB03 sa stáva Vase 48k Spectrum vlastníkom najlepšieho 8-bitového BASIC-u. Pre jeho používanie nie je potrebné veľké školenie.

Najprv je potrebné prečítať si stat o editácii a potom v ľubovoľnom poradí ostatné príkazy.

Nahratie BB03: LOAD "" alebo LOAD "Bate Basic"

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

Vytváranie kopii:

Riadok 1 je SAVE-rutina, ktorá ukladá na pásku (Microdrive) najprv BASIC program a hneď za ním nasleduje binárny kód.

Bezpečnú kopiu možno urobiť nasledovne:

- 1: Vtiahnuť BASIC program BB03 cez MERGE
- 2: CLEAR rt; LOAD "" CODE
- 3: Ak chcete kopiu na Microdrive, treba zeditovať riadok 1
- 4: Po RUN sa uloží program aj kód

Pozn.: Kódová časť BB03 sa môže overiť pomocou VERIFY, pokiaľ je BB03 v činnosti. Ak to chceme naprieť tomu urobiť treba ho odpojiť pomocou RANDIMIZEUSR 59904.

Vlastnosti:

Riadok 0 obsahuje definície BB03 funkcií. Tento sa stáva súčasťou každého programu. Na stare programy, ktoré neboli nahraté v BB03, treba použiť MERGE, pretože pri LOAD by sa zotiel potrebný riadok 0. NEW máza len program a urobí CLEAR, riadok 0 je ponechaný.

Programy napísané v BB03 môžeme nahrávať cez SAVE a LOAD, pretože sa uloží aj riadok 0.

Predĺžene pípanie klávesnice signalizuje činnosť BB03. Da sa vypnúť pomocou POKE 23609,0. Ďalší príznak práce je invertovaný riadkový kurzor pri listingu.

Stare programy idu v BB03 rýchlejšie ako v pôvodnom Sinclair Basic-u. Je to spôsobené tým, že v cykloch a skokoch sa využívajú priamo adresy a nie čísla riadkov, ktoré sa hľadajú.

E D I T A C I A

Kľucové slova: 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 popisane. Podrobný popis je v abecednom zozname príkazov ďalej.

Aktuálny riadkový kurzor:

- je zobrazený inverzne
- rýchlejšie sa pohybuje kurzorovými tlačítkami. Niekedy sa stane, že pozícia nezhodí, 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 klavesy v 'K' režime, 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 vyplni úplne 'K' režim a chcete pri ALTER, REF alebo KEYIN vkladať 1-byte-tokens do reťazcov).

KEYWORDS - Riadenie pri vkladaní programu:

Tento príkaz slúži na nastavenie vkladacích režimov. Môžete si zvolit buď vkladanie príkazov a funkcií vo forme beznej pre Spectrum, alebo formou výpisu. Na začiatku je nastavený stav KEYWORDS 3, ktorý umožňuje výber ľubovoľnej formy prípadne vkladanie miesane. Pri KEYWORDS 0 sú vypnuté nové rozširujúce príkazy a zobrazujú sa grafické znaky.

Pre jednokľavesové vstupy v 'K' režime treba pri doplnujúcich kľúčových slovách zaradiť režim '0' a potom stlačiť odpovedajúcu klavesu s doplnkovým príkazom. Pri funkciách možno použiť buď štandardne FN + písmeno + zatvorky 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 23508.

Pri vkladaní programu ako textové reťazce, je vhodné vkladať príkazy v malej abecede. BB03 si rozpoznane príkazy prevádza do veľkej abecedy, ostatné nechať male.

LIST FORMAT:

Prikaz umoznuje robit vypis programu so zarazkami pri blokoch. Tym sa zvyisi citatel'nost programu, pripadne sa lahsie objavia chyby v strukture. (FOR - NEXT a pod.)

CSIZE:

Umoznuje menit velkost velkost pismen od 64 (s OVER 1 az 85) znakov na riadok az po velka pismena na nadpisy.

JOIN (Programove riadky) a SPLIT:

JOIN spoji 2 riadky programu do jedneho,
SPLIT rozlozi jeden riadok do dvoch.

Vlozenie 'Noveho riadku' do programoveho riadku:

V lubovolnom mieste je mozne vlozit do vstupneho riadku programu alebo INPUT riadku kod CHR\$ 13 = NL pomocou CAPS SHIFT + ENTER. Tym sa riadok neukonci, ale prejde sa na zaciatok dalsieho.

P R O C E D U R Y A P A R A M E T R E

Klucove slova: DEF PROC, PROC (pripadne), LOCAL, DEFAULT,
REF, READ LINE, LIST PROC.

Funkcia: ITEM()

Tato kapitola Vas strucne oboznami s moznostami vyuzivania procedur. Napisanu proceduru je mozne pomocou MERGE v pripade potreby nahrat z kniznice, ktoru si mozeme vytvorit.

V hlavnom programe sa vola procedura priamo menom a zadaju sa jej pripadne pozadovane parametre. Procedury ulozene v pameti je mozne volat aj v priamom rezime, co umoznuje rozsirit mnozinu prikazov BB03.

Kazda procedura ma byt len taka velka, aby bola lahko pochopitelna.

Procedura je vlastne podprogram, ktory zacina DEF PROC, meno a pripadne zoznamom parametrov, ktore su jej odovzdane pri volani. Na konci miesto RETURN vlozime END PROC. Procedura sa najde bez ohladu na to, ci je na zaciatku alebo na konci programu, neberie sa do uvahy ci program bezal alebo nie.

Napriklad: Urobme ekvivalent CP/M prikazu DIR

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

Keď vlozite RUN, nic sa nestane. Procedura nebola volana, podobne ako DEF FN.

Vlozte: 10 dir a RUN . Vypise sa obsah Microdrive.

Alebo: 'DIR' a procedura sa opat vykona.

Teda Spectrum BB03 ma o jeden prikaz viac.

Meno procedury musi zacinat pismenom a koncit

'SPACE', ':', 'REF', 'DATA', alebo 'ENTER'.

V menach je mozne pouzivat velku a malu abecedu (bez rozlisenia), cislice alebo podciarknutie. Je mozne pouzit vsak aj ine znaky.

Procedury musia vzdy zacinat DEF PROC a na koncit END PROC.

Ak chcete pouzit viac END PROC v 1 procedure (pred cim varujeme), potom nevie pocitac najat skutocny koniec procedury.

```
Priklad 2: 100 DEF PROC dir cislo
           110   CAT cislo
           120 ENDPROC
```

Premenna cislo sa vola formálny parameter. Cez formálne parametre sa pri volaní odovzdávajú skutočné hodnoty do procedury. Pri volaní je možné použiť potom tvar:

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

Tento spôsob sa volá odovzdávanie parametra hodnotou, pretože do procedury sa odovzdáva len hodnota.

Formálny parameter v definícii procedury musí byť premenná. Táto je však lokálna pre danú proceduru a BB03 ju mimo procedury nepozná.

Premenná použitá inde v tele procedury je systému známa a nazýva sa globálna. Aby aj takáto bola lokálna, treba použiť príkaz LOCAL.

```
Priklad: 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 procedura. Potom PRINT n,xkrat; vypíše sa premenná n, pretože bola globálna, ale xkrat bude neexistujúca premenná.

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

```
105 LOCAL n
```

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

```
Priklad: 100 DEF PROC box x,y,sirka,dlzska
           110   PLOT x,y: DRAW sirka,0
           120   DRAW 0,-dlzska: DRAW -sirka,0
           130   DRAW 0,dlzska
           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 dlzska=sirka
```

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

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 procedury, ale aj z procedury von (Obdoba VAR parametra v PASCAL-e). Pred formálnymi parametrami, ktoré budú tohto typu, treba vložiť kľúčové slovo REF.

```
Priklad: 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 procedure a vytlačene by boli pôvodné obsahy.

Na rozdiel od iných počítačov s procedurami v BASIC-u umožňuje BB30 ako premennú používať aj polia. Jediné obmedzenie je také, že ich možno prenášať len cez REF. Počas používania v procedure 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 Procedure počíta všetky 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+t(n)
        360   NEXT n
        370 END PROC
```

Tým, že bola použitá funkcia LENGTH, je možné v procedure počítať 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 procedure sa preberajú cez READ. Informácie o parametroch dáva funkcia ITEM(), ktorá nadeľuje 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 procedury je možné voľat samotnú proceduru.

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 size>4 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.: Vypis je v LIST FORMAT 2

STRUKTUROVANE PROGRAMOVANIE

Kľúčové slová: a,Procedury 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 PASCAL-a, ale sú
flexibilnejšie

c,IF - THEN - ELSE

d,ON - Vyberá čísla riadkov alebo príkazy zo
zoznamu, obdobou CASE

e,LIST FORMAT - vypisuje programy so záložkami,
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 máte DEFAULTami, 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 nashrať aj samostatujúce 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

Pristup k datam - polia triedenie, prehľadavanie

Kľúčové slova : JOIN, COPY, DELETE, SORT pre polia a reťazce
EDIT premennej, SAVE DATA, USING

Hľadacie funkcie: INARRAY, INSTRING

Funkcie: LENGTH, CHAR\$, NUMBER, EOF, SHIFT\$, USING\$

JOIN a COPY umožňujú polia alebo ich časti spájať a miesať.
Rozmery polia je možné zmeniť bez straty dát.

DELETE vymaze časť polia.

SORT triedi polia.

INARRAY, INSTRING veľmi rýchle prehľadávajú polia alebo reťazce.

Čez EDIT je možné zmeniť obsah premenných.

Čez SAVE DATA sa dajú uložiť premenne na pásku (Microdrive).

USING, USING\$ formátujú výstup dát.

Funkcia EOF udáva, či už boli z Microdrive prečítané všetky dáta.

GRAFIČKE PRIKAZY

Kľúčové slova: ALTER, CONTROL CODES, CSIZE, DRAW TO, FILL, GET
(oblast obrázkov), OVER 2, PLOT, POKE, ROLL,
SCROLL, WINDOW, XOS/YOS/XRG/YRG

Funkcie: SINE, COS, FILLED, MEMORY\$, SORN\$

TOOLKIT - Zjednodušená manipulácia s programami

Kľúčové slova: ALTER, AUTO, DEF KEY, DELETE, LIST ... TO ...
LIST DATA/VAL/VAL\$, LIST DEF KEY, LIST PROC,
LIST REF, REF, RENUM,

Funkcia: MEM()

SYNTAX PRIKAZOV

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

ALTER <Popis atributov> 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 atributov (Môže ich byť viac).

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

2: V celom programe sa vyskytí prvej referencie nahradi druhou referenciou.

- a) ALTER a\$ TO b\$ Zmeni v programe a\$ na b\$
- b) ALTER sum TO s Zmeni v programe meno sum na s
- c) ALTER 1 TO 23 Zmeni v programe na mieste 5 byte, kde je kod 1 na kod 23
- d) ALTER "stary" TO "novy" Toto zmeni retazce aj vo vnútri
- e) ALTER (a\$) TO "retazec" Zmeni obsah premennej

AUTO <cislo riadku> <,krok> (/'G'+6)

Pre režim vkládania textu programu sa generujú císla riadkov. Ukončenie režimu sa urobí stlačením BREAK viac ako 1s alebo vymaz císlo riadku a potom nejaký príkaz (LIST a pod.).

Ak nezadáte hodnoty, predpoklada sa 10,10.

BREAK (CAPS SHIFT + SPACE)

Príkaz funguje aj pre podprogramy v strojovom kóde. Treba ho podržať dlhšie ako 1s.

Pozor! BB03 pracuje Interrupt mode 2.

CLEAR bytes (/'K'+X)

Pre bytes < 767 sa presunie RANTOP o zadane číslo nadol. Možno zadať aj záporné číslo. Tento posuv nemá vplyv na obraz, premenne a stack GO SUB, DO ...

Definície funkčných kláves a Windows sú tiež posunuté.

CLOCK < číslo, alebo retazec > (/'G'+C)

Týmto príkazom má užívateľ prístup k Interruptom riadeným 24 hodinovým hodinám, ktoré môžu byť prípadne zobrazené v pravom hornom rohu obrazovky. Ku danému času je možné zadať budík, alebo prerušiť riešenie programu odskokom do podprogramu.

Pozor! Pčas BEEP a prístupu na periferie hodiny stoja.

a) CLOCK n = Nastavenie režimu.

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

Pri nastartovaní 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 zvonit. V režimoch 4-7 sa skoci 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 ukončí a prejde sa do podprogramu.

Rychlosť hodín je možné meniť na adrese 56866, kde je počet päťdesiatin sekundy, ktorá ubehne 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 zada 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	Zavislé od WINDOW
CHR# 9	Kurzor vpravo	Zavislé od WINDOW
CHR# 10	Kurzor nadol	Zavislé od WINDOW
CHR# 11	Kurzor nahor	Zavislé od WINDOW
CHR# 12	Vymaz znaku	Zavislé od WINDOW
CHR# 13	Zvláštny ENTER (CR/LF)	Zavislé od WINDOW

Riadiace znaky pre obrazové bloky:

CHR# 0 - Nasleduje 8 byte grafické informácie

CHR# 1 - Nasleduje 1 byte atribut 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 pixeloch. Normálna hodnota je 8,8.

Hodnoty 255,176 vedú k jednému znaku na obrazovke. Ak sa zada iba jedna hodnota, mení sa veľkosť znaku v oboch smeroch úmerne.

Pri malej rozteci je potrebné prejsť do OVER 1, aby sa znaky neprepisovali. Potom je možné dosiahnuť hustotu 25 znakov na 1 riadok príkazom CSIZE 3,7;OVER 1

Dobre čitateľný je ešte CSIZE 4,3.

BE03 PRINT rutina je velmi vsastrenna, pretoze moze tlacit na lubovolnu poziciu pixla. Cez CSIZE 0 sa da prejst do rychlejsiej ROM rutiny, ktora je po natiahnutí zavedena, ale nevie spracovavat CONTROL CODES 0 a 1.

Všetky možnosti riadenia pozície ('AT', 'TAB', ',', atd.) su automaticky korigovane na platnu veľkosť CSIZE.

UDG znaky su pre sirkú mensiu ako 8 pixlov zobrazene len ako prave polovice.

GET obrazove bloky su preberane ako cele bloky 8*8 pixlov. Podľa zodpovedajúceho CSIZE su teda úmerne zmenene. Je potrebné dodržať celé násobky (4,8,16,24, atd.) aby súhlasili hrany.

Da sa použiť lokálne pre 1 príkaz PRINT t.j.

PRINT CSIZE 16,16;" veľké"

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

DEFAULT premenhodnota <premenhodnota> ... ('G'+SHIFT 2)

DEFAULT = SAVE/LOAD zariadenie

1: Účinok sa podoba LET. Popis vid pri procedurách. Využíja sa v prípade, ak nebola danej premennej iným spôsobom priradená hodnota.

2: DEFAULT = m Microdrive 1
 DEFAULT = M1 Microdrive 1
 DEFAULT = n5 Siet
 DEFAULT = B RS 232, kanál B
 DEFAULT = t Paska (standartne)

DEF KEY 1-znak retazec ; retazec <+""> ('G'+SHIFT 1)

DEF KEY 1-znak retazec : príkaz : príkaz : ... <1>

Definuje sa retazec, ktorý bude klavessa generovať. Pri tom je možné, aby boli hneď vykonané, alebo len vložené do editačného riadku a až po ENTER odoslané. Ak je posledný znak retazca, alebo príkazov ':', potom sa ponechá v riadku a čaká sa na ukončenie.

Aby boli funkcie klavessy akceptované, treba prejsť do režimu '#' tlačítkami SYMBOL SHIFT + SPACE. Kedykoľvek je možné funkcie klavessy predefinovať. Ak sa vloží prázdny retazec, alebo zoznam príkazov je prázdny, potom sa klavessa rusí.

DEF KEY ERASE rusí všetky definície.

Definície ležia nad RAMTOP-om teda su chránené pred NEW.

SAVE rutina BE03 z riadku 1 odklada aj definície klavess.

LIST DEF KEY vypíše všetky definície.

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

DEF PROC menoproc DATA

Podrobnosti vit v stati o procedurách. Príkazom začína definícia procedury. Príkaz musí byť vždy sam v jednom riadku. Meno procedury moze byť rovnaké ako meno premennej a nebudu zamenené. Može obsahovať lubovolne znaky, ale musí začínať písmenom.

Ze menom nasleduje zoznam parametrov, alebo kľucové slovo DATA. V zozname môžu byť aj mena poli (b() a pod.), ale pred nimi musí byť REF.

DELETE <ciadoriadku 1> TO <ciadoriadku2> ('G'+7)

DELETE menopola (slicer)

DELETE menoretazca (slicer)

1: Vymaze sa usek programu od riadku 1 po riadok 2. Ak chyba cisloriadku 1 dosadi sa cislo prveho riadku vacsieho ako 0. Ak chyba cisloriadku 2, berie sa posledny riadok programu.

DELETE TO vymaze cely program okrem riadku 0, hodnoty premennych su zachovane.

Zadane riadky musia existovat, inac je chyba. Prikaz moze byt aj v programe. Nedoporučuje sa, aby bol v podprograme, procedurach, cykloch. Ak ma zrusit aj sam seba, musi byt poslednym prikazom bloku.

2: Prikaz vymaze cast pola, retazca alebo cele. Pre retazec sa nedostane dlzka 0, ale zrusi sa v tomto pripade uplne.

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

Pozri aj LOOP, EXIT IF.

DO a LOOP davaju moznost programovat cykly bez potreby skokov.

a: DO ... LOOP = Nekonecna slučka

b: DO WHILE podmienka ... LOOP = Cyklus sa vykonava len dovtedy, kym je podmienka pravdiva. Potom sa uz nevykona a pokracuje sa za LOOP.

c: DO UNTIL podmienka ... LOOP = Cyklus sa vykona len v pripade, ze podmienka nie je splnena. V opacnom pripade sa vykonaju prikazy po LOOP a pokracuje sa dalej.

Adresa kazdeho DO je ulozena do zasobnika, preto z tohto cyklu sa nesmie vyskocit cez GO TO, ale iba pouzitim EXIT IF popripade pomocou POP.

```
DPOKE adresa, cislo ('G'+P)
```

Pozri aj funkciu DPEEK.

DPOKE je dvojity POKE a zodpoveda pomalsiemu:

```
POKE adresa, cislo - INT(cislo/256)*256
```

```
POKE adresa+1, INT(cislo/256)
```

Dolny byta sa ulozi na adresu a horny na adresu+1. Je mozne spracovavav cisla 0 - 65535.

```
DRAW TO x,y <,uhol> ('K'+W SYMBOLSHIFT+F)
```

Na rozdiel od povodneho relativneho DRAW, vykresli sa ciara do bodu x,y.

```
EDIT <cisloriadku> (@ alebo 'G'+SHIFT 5)
```

EDIT retazcova premenna

EDIT ; ciselna premenna

1: Prikaz nezodpoveda staremu tlacitku EDIT (Stale sa da pouzivav). Je mozne zadat cislo riadku, ktorý ma byt editovany. Ak to nezadate, bude sa editovat riadok s programovym kurzorom. 2.3 Prikaz umoznuje menit hodnoty premennych, pricom povodna hodnota sa presunie do editacnej zony a po editovaní sa ulozi nova. Syntax editacie je podobna INPUT (co sa tyka riadiacich znakov a prispadneho vypisu).

```
ELSE prikazy ('G'+E)
```

Prikaz je sucastou struktury IF - THEN - ELSE. Ak je podmienka za IF nepravdiva, nevykonaju sa prikazy za THEN, ale pokracuje sa za najblizsim ELSE. Pred ELSE musi byt '!',

Priklady: (pismena su pre prehľad, nie v programe)

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

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

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

Ak bolo ELSE použité bez IF, funguje ako REM.

END PROC ('G'+B)

Pozri aj odsek o procedurach.

Prikaz ukoncuje definiciu procedury. Ak sa najde DEF PROC pri behu programu, preskoci sa po najblizsie END PROC.

EXIT IF podmienka ('G'+I)

Pozri aj DO - LOOP.

Prikaz je súčasťou štruktúry 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.

Priklad: (Vyplnenie kružnice)

CIRCLE 128,37,30: FILL 128,60

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 Ciselná alebo reťazcová premenná ('G'+G)

GET reťazcová premenná, x,y < ,sirka ,vyska>< ;typ>

1: Čítanie z klavesnice. Prečíta sa jeden znak s klavesnice, pričom sa čaka na stlačenie klavesy a nečaka 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 súradnicami ľavého horného rohu x,y do reťazcovej premennej. Táto sa dá zobrazit na inom mieste cez PRINT alebo PLOT. Sirka a výška sa udáva v print pozíciách, x,y sú štandardné súradnice PLOT.

Vzniknutý reťazec sa neda vytlačiť pri CSIZE 0, ale je možné meniť rozmery cez normálny príkaz CSIZE.

Pozn.: Retazec pozostava z 9 znakov: prvý je CONTROL CODE 0, udavajúci, že ďalších osem byte je kodovaných 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 ine efekty. Ak sa neuda typ, potom sa predpoklada 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 hardware obmedzenie Spectral

JOIN <cielo riadku> ('K'+SHIFT 6)
JOIN retazec alebo pole

Pozri aj SPLIT.
1: Prikaz spoji dany riadok s nasledujucim riadkom. Ak chyba číslo riadku, vezme sa riadok s programovým kurzorom. Touto operáciou sa ušetrí 4 byte, zvýši sa rýchlosť programu.

2: JOIN presuva retazce a polia. COPY ich kopiruje, ale v syntaxi sa nelisia, preto sú popisane spolu.

a) JOIN/COPY retazeci <slider> TO retazec2 <pozicia>

Prida sa retazeci k retazcu2.

Priklad: 10 LET a\$="12345",b\$="ABCDEFGH"

20 JOIN a\$ TO b\$

30 PRINT b\$: REM vypise ABCDEFGH12345

40 PRINT a\$: REM a\$ už neexistuje => chyba

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

b) JOIN/COPY pole1 <slider> TO pole2 <pozicia>

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

Priklad: Majme pole a\$(100,30), čo nestačí a potrebujeme ďalších 20 retazcov. Postup: DIM b\$(30,30): JOIN b\$ TO a\$. Funkcia LENGTH(1,"a\$") ukáže dĺžku 120.

Vkladaný retazec prispôbuje svoju dĺžku tomu, kde je vkladany. Ak je napríklad 30 znakov málo, potom utvoríme pomocné pole s potrebnou dĺžkou a pridáme k nemu krátke pole:

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

Teraz znovu 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 retazce:

DELETE a\$(1 TO 2)

Pozor: Spracovávané polia môžu mať maximálne 2 rozmery.

KEYIN retazec ('K'+SHIFT 4)

Prikazom 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.

Priklad: 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

ciarky

60 KEYIN a\$

Po RUN uvidíte, že program si sám vytvoril v riadku 100 príkaz DATA ...

Prikaz INPUT "Heslo:":a\$: KEYIN a\$ umiestnený na začiatku programu by mohol predstavovať ochranu programov, pretože nestačí heslo, ale treba vedieť aj číslo riadku.

V rezime KEYWORDS 3 alebo 4 sa klucove slova prevedu najprv na 1-byte tokeny.

KEYWORDS cislo ('G'+8)

Cisla 0 a 1 riadia vystup (PRINT, LIST)

2 az 4 riadia vkladanie programu.

Vyznamy: 0 - Zobrazuju sa graficke znaky
 1 - Zobrazuju sa nove klucove slova BB08. Nastavene po starte.
 2 - Klucove slova treba vkladat ako tokeny, t.j. jednou klavesou.
 3 - Po vstupe je riadok otestovany a vsetky klucove slova su prevedene na 1-byte. Akceptuje obidva druhy vstupu.
 4 - Neexistuje 'K' kurzor. Vsetky prikazy treba vypisovat.

LET premenna=vyraz <,premenna=vyraz>... ('K'+L)

Do jedneho prikazu je mozne vlozit viac priradeni.

LIST/LLIST <cisloriadku1> TO <cisloriadku2> ('K'+K, 'E'+V)

Vypise sa usak programu. Je pritom mozne niektoru z hodnot vynechat. Dosadia sa podobne ako pri DELETE.

LIST DATA -> Vsetky premenne ('K'+K 'E'+D)

LIST VAL -> Ciselne premenne ('K'+K 'E'+J)

LIST VAL\$ -> Retazcove premenne ('K'+K 'E'+SHIFT J)

Vypise sa aktualny obsah premennych.

2; vypise premenne v poradí: 1 - ciselne polia
 2 - riadice premenne FOR cyklov
 3 - premenne s 1 pismenovym menom
 4 - premenne s viacpismenovym menom
 3; vypise dalej: 5 - retazcove polia
 6 - retazcove premenne

Pre polia sa vypisu len rozmery, nie obsah. Pre retazce len prvych 15 znakov.

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

Prikaz vypise definicie vsetkych nadefinovanych funkcných klaves.

LIST FORMAT cislo ('K'+K 'E'+SHIFT 0)

Prikaz nastavi rezim, v ktorom bude robeny vypis programu. Na zaciatku je nastavene 0.

Vyznam cisla: 0 - Klasicky vypis IX Spectra, avsak riadky dlhsie ako 32 znakov pokracuju na dalsom riadku az od piatej pozicie.
 1 - Pre struktury robi odskok 1 medzera.
 2 - Pre struktury robi odskok 2 medzery.
 3 - Ako 0, ale bez cisel riadkov.
 4 - Ako 1, ale bez cisel riadkov.
 5 - Ako 2, ale bez cisel riadkov.

Aby bol vypis pekny, je dobre vlozit za kazdym THEN aj ELSE: .

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

Vypise sa text celej procedury s menom meno.

Pozn.: Po vypise obsahuje adresa 83825 cislo prveho a adresa 57358 cislo posledneho vypisovaneho riadku. Toto je mozne vyuzit pri programovo riadenom DELETE alebo RENUM. Hodnotu treba citat funkciou DPEEK.

LIST REF udaj ('K'+K 'G'+SHIFT 7)

Prikaz vypise cisla riadkov, na ktorych sa nachadza referencia na udaj. Toto moze byt meno premennej, cislo, postupnost znakov.

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

Vytvoria sa specialne premenne, ktore su znane len danej procedure. Formalne parametre su pre proceduru automaticky lokalne. V procedure sa moze vyskytovat viac prikazov LOCAL. Ak ma byt lokalne pole, musi byt deklarovane postupnostou prikazov:
LOCAL p(); DIM p(...)

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

Prikaz ukoncuje strukturu DO - LOOP. Podobne ako DO je mozne vlozit prikazy pre podmienene ukoncenie cyklu.

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

Pri Microdrive sa BB03 chova pre samostartujuci program vtiahnuty cez MERGE korektne.

MOVE ('E'+SHIFT G)

Moze na Microdrive teraz prenasat aj programy a ODDI subory.

GO TO/GO SUB ON vyrazi: riadok1, riadok2, ... ('K'+G/H, 'G'+0)
ON vyrazi: prikaz: prikaz: prikaz: ... prikaz ('G'+0)

1) Skoci sa na riadok, ktoreho cislo je v poradi na pozicii rovnaj hodnote vyrazu.
2) Vykona sa prikaz, ktoreho poradove cislo je rovne hodnote vyrazu.

Ak je hodnota vyrazu mimo poctu riadkov alebo prikazov, pokracuje sa nasledujucim prikazom alebo riadkom.

ON ERROR cialoriadku ('G'+N)
ON ERROR: prikaz: prikaz: ...

1) V pripade chyby (okrem 0 OK a 9 STOP Statement) sa zavola podprogram na uvedenom ciale riadku.

2) Vykona sa prikazy na riadku. Ak nie je chyba, preskocia sa. Vlastne spracovanie je vypnute pocas behu podprogramu, obnovi sa pri RETURN. CONTINUE vrati program na riadok, ktorý spôsobil chybu, ale nezapne sa spracovanie chyby, takže sa teraz vypise. (Pred CONTINUE treba vybrať návrat prikazom POP).

ON ERROR 0 vypne vlastne spracovanie chyby.

V podprograme existuju tri premenne:

line = cislo riadku s chybou
stat = poradove cislo prikazu v riadku
error = cislo chyby (vid priloha C)

Mozu sa pouzivat aj v programe, ale ak sa aktivuje ON ERROR alebo TRACE, su prepisane.

OVER cislo ('E'+SHIFT N)

Prikaz akceptuje okrem parametrov 0,1 aj 2. Jeho ucinok je OR. To znamena, ze znaky sa ani neprepisuju jeden druhym, ani sa nevykonava XOR po pixloch.

PLOT x,y <retazec> ('K'+Q)

Okrem zobrazenia bodu je mozne na mieste x,y zobrazit aj retazec. Moze sa jednat o normalny retazec alebo o retazec, vytvoreny pomocou GET. X, y urcuju lavy horny roh retazca.

Okrem INK, PAPER a pod. je mozne vlozit aj CSIZE na zvacsenie alebo zmensenie retazca.

Priklad: PLOT CSIZE 32; INK 2; 100,68;"HI!"

Vyhoda prikazu je rovnaky suradnicovy system pre grafiku aj znaky.

POKE adresa, retazec ('K'+O)

Okrem cisla 0 - 255 je mozne vkladat aj retazce. Ekvivalent PEEK nahrada funkcia MEMORY\$(slicer).

POP <ciselná premenna> ('G'+Q)

Prikaz vybera hodnotu zo stacku (GO SUB, DO, PROC). Ak je udana premenna, vlozi sa do nej cislo riadku.

Mozte teraz vyskocit z podprogramov, cyklov a procedur.

Priklad: 100 GOSUB 500

110 STOP

500 POP loc

510 PRINT "Podprogram volany z riadku ";loc

520 GO TO loc+1

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

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

Prikaz vola proceduru. Je to obdobie GO SUB, ale netreba poznat cislo riadku. Klucove slovo PROC mozno vynechat. Podrobnejši popis vid v stati o procedurach.

READ LINE <retazcova premenna> ... ('E'+A 'E'+SHIFT 3)

Prikaz umoznuje precitat do premennych hodnoty bez toho, aby museli byt v DATA-ch uvedene v " ".

REF referencia ('G'+SHIFT 7)

REF premenna

! Tymto prikazom sa prezrie cely program na vyskyt danej referencie.

Referencia je: premenna, cislo, postupnost znakov ... Ak sa najdu znaky, presunie sa dany riadok do editacnej oblasti s kurzorom za najdenou referenciou. Ak riadok nechcete opravit, treba odpovedat 'ENTER'. Ak stlacite ENTER este raz, hlada sa dalsi vyskyt referencie. Ak pocas hladania vlozite nejaky prikaz, hladanie sa prerusi.

Priklady: REF a\$ - hlada sa: a\$
 REF sum - hlada sa: -sum-
 REF "SUM" - hlada sa: SUM
 REF 1 - hlada sa: 1 (vratane neviditelnej 5-byte formy)
 REF "1" - hlada sa: 1
 REF 12*4 - hlada sa: 12 (vratane...)*4 (vratane...)
 REF (a\$) - hlada sa: obsah a\$
 REF (x) - hlada sa: obsah x vcitane...

Velka - mala abeceda nahra rolu.

2: V zozname formalnych parametrov urci premenne, ktorych hodnota je odovzdana adresou. Znamená to, že akakolvek 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)

Cely, alebo cast programu je mozne precislovat, presunut useky, alebo skopirovat na ine miesto programu.

a) RENUM Precisluje program tak, ze cisla riadkov budu v rozstupoch po 10, prve cislo je 10.
 b) RENUM od TO do Precislovanie useku.
 c) * Stary blok sa po presunuti nezmaže teda, skopiruje sa.

Priklady: RENUM - cely program
 RENUM LINE 100 STEP 20 - cely program 100,120,140
 RENUM 100 LINE 300 - riadok 100 na 300
 RENUM 1540 TO LINE 2000 - vsetko od 1540 sa presunie na 2000 a dalej

Prikaz precisluje vsetky referencie v GO TO, GO SUB, RESTORE, RUN, ON, ON ERROR, TRACE, LIST, LINE a DELETE. Pripadne cisla riadkov v CLOCK je treba zmenit rucne.

Ak sa stane, ze cislo riadku v GO TO, GO SUB je vyraz, napri: 100 PRINT: GO TO a vypise sa sprava: 'Failed at ...' (100:2)

Pozn.: RENUM vyuziva pre tabulky pamat obrazu. Preto je mozne spracovavat aj dlhe programy.

ROLL kodsmeru <,pixle><;x,y;sirka,vyska> ('G'+R)

Pozri aj SCROLL.

Prikaz presuva cely alebo cast obrazu o dany pocet pixlov v lubovolnom smere. To co z jednej strany vyjde von, na druhej strane sa objavi (na rozdiel od SCROLL).

Obraz sa posuva rychlejsie, ak sa zada vacsi pocet pixlov. Najrychlejsi presun je pri presune vo vodorovnom smere o 4, alebo 8 pixlov, ked sa vyuzivaju priamo instrukcie Z80.

kodsmeru	směr	pohybuje sa
1	vľavo	atributy
2	nadol	atributy
3	nahor	atributy
4	vľavo	atributy
5	vľavo	pixle
6	nadol	pixle
7	nahor	pixle
8	vľavo	pixle
9	vľavo	obidve
10	nadol	obidve
11	nahor	obidve
12	vľavo	obidve

Kedže sa atributy môžu presúvať len po 8 pixloch, ignoruje sa počet zadávaných pixlov.

V prípade presunov časti obrazu sa zadáva šírka s 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 naktahovaní 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. Prítom je vhodné pred uložením urobiť RENUM na vysoké čísla riadkov a po vŕtiahnutí RENUM na požadované miesto, aby sa dali vložiť ďalšie procedúry.

```
SCROLL <kodsmeru><,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.

```
matice      matice
SORT <pole znaku> nebo <pole čísel> nebo retezec
```

Kľavesy: M

SORT triedi reťazce, číselce alebo písmena v poradi v zostupnom alebo sestupnom. Nasled popísa spôsob použitia na matici znaku. Pohľadniete si nasledujúci program :

```
100 DIM A$(100,10)
110 FOR S = 1 TO 100 : FOR L = 1 TO 10
120 LET A$(S,L) = CHR$( RND*25 +65 )
130 NEXT L : NEXT S : GOTO 200
140 SORT A$
200 FOR S = 1 TO 100 : PRINT A$(S) : NEXT S
```

Tento program vyrobí matici 100 reťazcu o dĺžke po 10 znaku (čo trvá nejakou chvíľou) a nakoniec je matice demonštrovaná. Zadejte pak GOTO 140. reťazce sa roztriedi a nakoniec jsou znovu demonštrované.

Neleknete se - trideni zde trva petinu sekundy! U dalsich retezcu je zapotrebi casu jen neznatelne vice. Doba trideni je predevsim zavisla na poctu retezcu : 200 retezcu se roztridi za 0,7 sec. , 400 za 3 sec.

Retezce se tridi podle kodu ASCII, prazdne misto prijde pred "A", a toto zase pred "a" (viz priloha A k manualu spectra).

Pozmenite-li radek 140 na SORT INVERSE A\$, probiha trideni sestupne; pokuste se o to! Pomoci "sliceru" vlozky lze dej trideni omezit na urcity blok retezcu :

```
SORT A$(1 TO 20)
```

bude tridit pouze prvnych 20 retezcu a

```
SORT A$(30 TO)
```

bude tridit retezce od poradí 30 do konce. A je dokonce mozne trideni omezit jen na urcite casti retezcu:

```
SORT A$( ) ( 2TO)
```

tridi vsechny retezce podle druhého az posledního písmene, k prvému písmenu se nepřihlíží (zúčastní se však při pretridování). Váimnete si toho, že v tomto případě bylo třeba dvou "slicerů". Pokud máte vumyslu tridit jen podle dílu jednotlivých retezcu, musíte tyto díly definovat ve druhém "sliceru".

Předpokládejme, že jsme založili databanku se jmény, bydlistem a starím našich známých. V každém retezci (věta) tvoří prvých 20 znaků jméno a bydliste dalších 20 znaků. Starí se zajisté pohybuje mezi 0 a 255, může být proto odloženo v pozici 41 napr. takto :

```
LET A$(S,41) = CHR$ stari
```

u osoby s bezným číslem S (poradovým). Nyní postací zadat jediný příkaz, abyste všechny osoby roztridili podle jména, nebo podle bydliste, nebo podle stari.

Problem vyvstane tehdy, pokud máme v umyslu tridit podle číslic o různém počtu míst. Použijeme-li příkazu

```
LET A$(S,41 TO 46) = STR$ cislo
```

proto, abychom založili proměnou "cislo" do naší datové věty dojde k tomu, že "cislo" se přiřadí počínaje svým levým okrajem. Číslice 9 s číslem 9.00 DM se postaví tedy na totes místo jako číslice 1 s číslem DM 100.00. S takovým uspořádáním by dej SORT nemohl pracovat korektně. Musíme proto všechna čísla naformátovat tak, aby desetinné čárky či tečky stáli pod sebou. Toho se dosáhne jednoduše pomocí formátové funkce USING\$ (viz. tam v tom odstavci, jakoz i při USING a též u CHAR\$ pro čísla kodovaná podle znaku) :

```
LET A$(S,41 TO 46) = USING$("000.00",cislo)
```

Nyni lze zadat všechna čísla, třeba obraty zakazek, roztridit je a vytisknout prvních 20 největšími hodnotami alfabetycky. Všimnete si toho, že code pro "1" přijde před "2", takže SORT u čísel demonstruje napřed čísla menší. Pomocí SORT INVERSE seřadí podle velikosti čísla sestupně - když jsou čísla zde vyjádřena jako řetězce.

SORT pracuje též s řetězci jednoduše indexovanými i neindexovanými vůbec :

```
INPUT S$(pozice) = CHR$(data).
```

Týmž syntaxem pracuje SORT též s číselnou maticí dat o jedné nebo dvou dimenzích.

```
SORT B(1 TO 20) (2)
```

trídí prvních 20 řádků matice B podle čísel ve druhém sloupci (přitom se pohybuje s celými řádky). Použijte vždy nejméně jednoho "aliceru", když chcete roztrdit číselnou matici B() , aby rutina SORT ji uměla od proměnných B rozlišit. Druhý "alicer" smí u číselných polí mít délku 1 znaku - jinak než u polí řetězcových.

Uspořádání čísel vyžaduje cca čtyřnásobnou dobu než uspořádání řetězců právě pro rozdílné formáty a možný výskyt negativních čísel.

Poznámka : U verze 1.0 vyskytují se určité problémy, pokud počet elementů ke třídění je 256, 512, 768 a pod. Vyhnete se proto podobným násobkům 256 co do počtu tříděných jednotek.

SPLIT (SHIFT W)

Pracuje se s řádkami, které su v dolnej části obrazu. Přitom staci přesunout kurzor za :, stlačit <> a řádek vlevo sa po 'ENTER' uloží do programu. Část řádku vpravo zůstane s tým istým číslom v editacnej zone.

Příklad: 10 PRINT "hallo": GO TO 20: <> PRINT "Příklad"

Stlačit ENTER, do programu sa uloží:

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

V editacnom riadku zůstane:

```
10 (kurzor) PRINT "Příklad"
```

TRACE číslo řádku ('G'+T)

TRACE příkaz: příkaz: příkaz: ... :RETURN

Příkaz sa využívá při ladění programů. Umožňuje například vypis řádků počas spracovania.

Příklad: TRACE: LIST lino TO lino: PAUSE 0: RETURN

1: Před každým řádkom sa vykona na zadanom čísla řádku podprogram. Dalsie příkazy za TRACE sa na trasovanie nevstahujú.
2: Příkazy stojace za TRACE sa vykonajú před každým řádkom programu ako podprogram.

Pocas vykonavania trasovacieho podprogramu sa režim trasovania vypne, TRACE 0, RUN a CLEAR vypnu tento režim úplne.

Ako podprogram v trasovacom režime môže byť použitá rutina, určená aj na ine účely.

Príklad: Vypis čísla riadku a 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 formatovaní tlače:

PRINT USING formátovací reťazec; číselný výraz

Tu bude tiež popísaná funkcia USING#.

Príkaz USING a funkcia USING# umožňujú vytvárať čísla v zadanej forme. Prítom 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 desatinne miesta.

. - desatinná bodka

pišmeno - zobrazí sa

Príklady: "##.#" 12.3
 "###.#" 12.3
 "000.00" 012.35
 "00" 12
 "00.00K06" 12.35K06
 "0.00" 0.00

%..8 prekročený formát

Pozn.: USING nespracová čísla v exponenciálnom tvare.

Príkazy: PRINT USING A#; číslo
 PRINT USING#(a#, číslo)
 sú rovnocenne.

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' + S)

Pozri aj CLS, CSIZE

Príkaz vytvára pre zobrazenie výstupu pravouhlé okno. 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 okna majú svoje informácie odložené za RAMTOP, a sú chránené pred NEW.

Špeciálny prípad je okno nula, ktoré je aktívne po vtlačnutí. Je prvé určené ako celá obrazovka.

Priklad: WINDOW 1,0,175,128,176

Okno 1 bude mat lavy horny roh v pozicii 0,175, je siroke 128 pixlov a vysake 176 pixlov (lava polovica obrazu). Atributy su ako v 0.

Okno sa aktivuje prikazom WINDOW 1. Ked sa aktivne okno deaktivuje, je CSIZE a aktualna PRINT pozicia odlozena za RANTOP.

Prikaz WINDOW ERASE vymaze definicie okien.

XOS, YOS, XRG, YRG

Pozri prilohu D.

Uvedene slova nie su klucove, ale specialne premenne. Tieto umoznuju menit pre prikazy PLOT, DRAW a CIRCLE pociatok suradnicoveho systemu a rozmery obrazovky.

XOS,YOS - Suradnice pociatku

XRG,YRG - Dlzka usekov na osiach x, y

Pozor na obluky a kruznice pri DRAW a CIRCLE !

FUNKCIE

V BB03 mate k dispozicii 26 novych funkcii. Tieto su definovane v riadku 0. Ich skutočne definicie su vsak z dovodu rychlosti v strojovom kode BB03. Pri vypise su zobrazene ako klucove slova, takže namiesto FN 0\$ sa zobrezi STRING\$ a kurzor preskoci ako pri klucovom slove.

Ak mate programy, ktore nahodou pouzivaju tieto definicie, treba ich premenovat (Napr. ALTER).

Mena je mozne vkladat aj vypisanim alebo ako FN pismeno (alebo \$.

Pri vtiskovaní programov nenapisanych v BB03 treba pouzit MERGE, aby nebol vymazany riadok 0 s definiciami. Pre programy z BB03 je mozne pouzit LOAD, pretože SAVE odložil automaticky aj riadok 0.

Funkcia	Skutočne meno	Funkcia	Skutočne meno
AND	FN A(	MEM	FN M(
BIN\$	FN B\$	MEMORY\$	FN M\$
CHAR\$	FN C\$	MOD	FN V(
COS\$	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 (cislo,cislo)

FN A(cislo,cislo)

Funkcia robi bitovy OR medzi dvoma cislami 0 - 65535.

BIN\$ (cislo)

FN B\$(cislo)

Funkcia prevedie cislo 0 - 255 do 9- znakového reťazca, ktorý predstavuje dvojkovu 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# (cislo) ----- FN C#(cislo)

Pozri aj NUMBER (retazec).
Funkcia prevedie cislo 0 - 65535 na dvojznakovy retazec. Tak
je mozne ukadat cislo namiesto 5-byte na 2-byte.

Funkcia NUMBER je opacna a vytvara z retazca naspät cislo.

Priklad: 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 obsadi len 1000 byte oproti 2500 byte normalne. SORT
pracuje aj na takto skompromitovane pole.

COS# (cislo) ----- FN C(cislo)

Funkcia je ekvivalent COS, ale je cca 6-krat rychlejsia.

DEC (retazec) ----- FN D(retazec)

Pozri aj HEX# (cislo).

Funkcia vrati dekadicky ekvivalent i z 4-znakoveho hexa-
decimalneho retazca. Nerozoznava sa velka a mala abeceda.

DPEEK (adresa) ----- FN F(adresa)

Jedna sa o dvojity PEEK pre danu a nasledujucu adresu. Pomaly
v BASIC-u vyzerá takto: PEEK adresa+256*PEEK (adresa+1)

EOF (cislo kanala) ----- FN E(cislo kanala)

Funkcia dava pri citani dat z Microdrive 0, ale v pripade
posledneho udaja da 1.

FILLED () ----- FN F()

Vrati sa pocet pixlov, ktore boli nastavene poslednym prikazom
FILL.

HEX# (cislo) ----- FN H#(cislo)

Pozri aj DEC (retazec)

Vlozena hodnota sa vycisli a prevedie sa do hexadecimalneho
retazca.

Pre cisla -255 < x < 255 ma retazec 2 znaky

-65535 < x < 65535 ma retazec 4 znaky

INARRAY (retazcove pole(start<,alicer>),hladany retazec) FN U(.)

Pozri INSTRING.

Funkcia pretestuje retazcove pole na vyskyt retazca a vrati
jeho hodnotu alebo 0, ak ho nenaide.

V podstate sa jedna o INSTRING aplikovany na polia. V pripade,
ze niektore znaky retazca su nepodstatne, mozno na ich miesto
vlozit '#',

Pozn.: Mozna pouzit pre polia max 2 rozmerov.

INSTRING (start,retazec1,retazec2) FN I(...)

Funkcia prehliada retazec1 na vyskyt retazca2 pocnuc od pozicie start. Ak je retazec nájdený, funkcia vráti pozíciu prvého znaku, kde sa v retazci začína retazec2. Ak sa nenájde, vráti 0.

Znaky v retazci2, ktoré nie sú podstatné, je možné nahradiť znakom '#'.
Příklad: PRINT INSTRING (1,a#,"M##ER")

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

ITEM() FN T()

Pozri odsek o procedurách.

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 retazec

2 - Nasledujúca položka bude číselná

LENGTH (n,"menopola") FN L(n,"menopola")

Funkcia dá informáciu o dĺžke pola, ktorá je v BB03 veľmi dôležitá, pretože sa môže meniť bez straty informácií. Taktiež je schopná dať adresu, kde sa pole alebo retazec nachádza.

Pre n = 1 sa vráti veľkosť 1. rozmeru

2 sa vráti veľkosť 2. rozmeru

0 vráti adresu prvého prvku pola alebo retazca.

NEM () FN M()

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

MEMORY# () FN M#()

Pozri aj POKE retazcov.

Funkcia má k dispozícii celú pamäť od 0 do 65535 ako retazcovú 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)

Prehliadať celú pamäť na ľubovoľný retazec je možné:

10 REM aadfg

20 PRINT INSTRING (1,MEMORY#(), "aadfg")

MOD (cislo1,cislo2) FN V(cislo1,cislo2)

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

NUMBER (retazec) FN N(retazec)

Pozri aj CHAR# (cislo)

OR (cislol,cislo2) FN O(cislol,cislo2)

Urobi sa logicka bitova operacia medzi cislom1 a cislom2.

RNDM (cislo) FN R(cislo)

Vrati sa nahodne cislo 0 - 1 ak cislo = 0

Ak je cislo vacsia ako 0, vysledok je cele cislo 0 - cislo

RANDOMIZE (cislo) nastavi pevnu sekvenciu nahodnych cisel.

SCRN\$(riadok,stipec) FN K\$(riadok,stipec)

Pracuje podobne ako SCREEN\$, avsak pozna aj UDG znaky.

SHIFT\$(cislo,retazec) FN Z\$(cislo,retazec)

Vseobecne pouzitelna funkcia na konverziu retazcov. V zavislosti od cisla su robene zmeny v retazci.

- 1 - Vsetky znaky abecedy na velke
- 2 - Vsetky znaky abecedy na male
- 3 - Inverzia malych a velkych pismen
- 4 - Zo vsetkych riadiacich znakov okrem CHR\$ 13 " , "
- 5 - 4 + zmeni CHR\$ 129-255 na CHR\$ 0-127
(potlaci klucove slova)
- 6 - 5 aj pre CHR\$ 12
- 7 - Zmeni tokeny na vypisany tvar
- 8 - Inverzne 7
- 9 - 8, ale kazdy znak musi nasledovat klucove slovo
- 10 - 9, ale klucove slova pisane velkymi pismenami

SINE (cislo) FN S(cislo)

Rychly sinus, podobne ako COS\$,

STRING\$(cislo,retazec) FN S\$(cislo,retazec)

Vysledkom je cislokrat retazec.

Priklady: STRING\$(32,"-")

= -----

STRING\$(4,"AB")

= ABABABAB

STRING\$(3,"A"+CHR\$(13))

= A

A

A

TIME\$ () FN T\$()

Vysledkom je cas tak, ako bol ziskany prikazom CLOCK,

USING\$(formatovaci retazec,cislo) FN U\$(...)

Podla formatovacieho retazca sa vytvori retazec obsahujuci cislo v pozadovanom tvare.

XOR (cislol,cislo2) FN X(cislol,cislo2)

Funkcia urobi logicku operaciu XOR cislol a cislo2 po bitoch.

P R I L O H A A

Mnozina znakov:

Pri zapnutom rezime KEYWORDS 1 sa zmeni vyznam klaves
ZX-Spectra nasledovne:

kod	klavessa	znak
128	0	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 0	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 uzke písmo (CSIZE 4,8) sú v RAM od 51291 do 51626 (CHR# 32 až CHR# 127). Každá 8 znaky zaberajú 7 byta. Parne znaky obsadzujú 4 ľavé bity každého byte, nepárne 4 prave bity. Pre každý znak sa vytvorí 1 rada prázdných pixlov, takže vznikne 8 radov.

PRÍLOHA B

Rozsirujúce a nové chybové hlásenia:

Kod	Význam	Situácia
G	No room for line Nové precisovanie riadkov by viedlo k neocislovaným riadkom, alebo k riadkom s číslicami väčšími ako 9999.	RENUM

S	Missing LOOP	DO, EXIT IF
	Po EXIT IF, alebo podmienom DO (spolu s WHILE alebo UNTIL) sa nenasiel zodpovedajúci príkaz LOOP.	
T	LOOP without DO	LOOP
	Bol nájdený príkaz LOOP bez zodpovedajúceho DO.	
U	No such line	DELETE
	Riadok udaný v DELETE sa nenasiel.	
U	No POP data	POP
	Bol urobený pokus prečítať číslo z prázdneho stacku pre GO SUB, PROC a DO.	
	Znamená to, že neboli aktivované procedury, podprogramy ani DO cykly.	
W	Missing DEF PROC	PROC, END PROC
	Bola volaná procedura bez definovania, alebo bol nájdený príkaz END PROC bez zodpovedajúceho DEF PROC.	
X	No END PROC	DEF PROC
	Program sa pokúsil preskočiť definíciu procedury, ale nenasiel jej koniec.	

P R I L O H A C

Chybové kody:

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

ERRORR hodnota	Kod	Hlasenie
0	0	OK - nespracované
1	1	Next without for Ku NEXT sa nenaslo 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 Podprogram nebol volaný, nasiel sa RETURN
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 Číselná časť je mimo dovolený rozsah

12	C	Nonsense in BASIC Nesmysel v BASIC-u
13	D	BREAK - CONT repeats Prerušenie, CONTINUE bude opakovať
14	E	Out of DATA Prikaz READ nenasiel data
15	F	Invalid file name Nespravne meno suboru
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 nema zodpovedajúce NEXT
19	J	Invalid I/O device Nespravne zariadenie V/V
20	K	Invalid colour Nespravna farba
21	L	BREAK into program Prerušenie počas programu CONTINUE nebude opakovať, ale pokračovať
22	M	RAMTOP no good Nespravny vrchol pamate
23	N	Statement lost Skok na strateny prikaz
24	O	Invalid stream Nespravny prikaz pre INTERFACE 1
25	P	FN without DEF Volana funkcia bez definicie
26	Q	Parameter error Chyba v parametroch
27	R	Tape loading error Chyba pri citani z pasky
Bets Basic chyby:		
28	S	Missing LOOP Chybajuci LOOP
29	T	LOOP without DO Chybajuce DO
30	U	No such line Nenasiel sa riadok
31	V	No POP data Stack je prazdny, nebola volana struktura
32	W	Missing DEF PROC Chyba definicia procedury
33	X	No END PROC Procedura nema koniec
INTERFACE 1 chyby:		
43	b	Program finished Program skoncil
44	c	Nonsense in BASIC Nesmysel v BASIC-u, volana pri chybe v prikazoch INTERFACE 1
45	d	Invalid stream number Zle cislo kanala
46	e	Invalid device expression Zly vyraz pre zariadenie
47	f	Invalid name Zle meno

48	g	Invalid drive number Nepravne cislo zarariadenia
49	h	Invalid station number Nepravne cislo uzla v sieti
50	i	Missing name Chybajuce meno
51	j	Missing station number Chybajuce cislo stanice
52	k	Missing drive number Chybajuce cislo stanice
53	l	Missing band rate Nezadana rychlost prenosu
54	m	Header mismatch error Chyba v hlavicke
55	n	Stream already open Kanal je uz otvoreny
56	o	Writing to a "read" file Pokus o zapis do suboru urcneho pre citanie.
57	p	Reading a "write" file Pokus o citanie zo suboru urcneho pre zapis
58	q	Drive "write" protect Zakaz zapisu na dany mechanizmus
59	r	Microdrive full Kazeta microdrive je plna
60	s	Microdrive not present Microdrive nie je k dispozicii
61	t	File not found Subor sa nenasiel
62	u	Hook code error Chyba pri volani ROM v INTERFACE I
63	v	CODE error Chyba v kode
64	w	MERGE error Chyba pri MERGE
65	x	Verification failed Verifikacia sa nepodarila
66	y	Wrong file type Zly typ suboru

P R I L O H A D

Specialne premenne a nove systemove premenne:

a) Graficka: Nedaju sa vymazať CLEAR, RUN ale su nastavane na pociatocne hodnoty. Daju sa nastavit cez LET a maju vplyv na DRAW, DRAW TO, PLOT, CIRCLE, GET a FILL.

Meno	inicialna hodnota	vyznam
xos	0	PLOT x+XOS,y
xrg	256	@<=x<=xrg
yos	0	PLOT x,y+YOS
yrg	176	@<=y<=yrg

b) TRACE a ON ERROR premenne

Su vytvarane a specialnych podmienok. Mozte si vytvorit vlastne premenne a ich menami, ale ak pride k nejakej udalosti, budu prepisane a ich hodnotu nastavi system.

Meno	vyznam
error	Kod poslednej chyby (vid priloha C)
line	TRACE cislo nasledujuceho riadku
	ON ERROR cislo riadku s chybou
stat	TRACE poradove cislo nasledujuceho prikazu
	ON ERROR poradove cislo riadku s chybou

c: Dalsie systemove premenne:

29625	(2)	Prve cislo riadku procedury po LIST PROC
57358	(2)	Posledne cislo riadku procedury po LIST PROC
57362	OVER2T	Docasny OVER 2 status
57363	OVER2P	Permanentny OVER 2 status (1=OVER 1, 0=ina)
57364		X-surednica vystupu na obrazovku (nie pri CSIZE 0)
57365		Y ako hore
57366	XRIGHT	Max x-surednica (limit okna)
57367	XLEFT	Min x
57368	YTOP	Max y
57369	YBOT	Min y
57376	CURWIN	Aktualne okno, 0 pre 0, inak cislo okna + 127
57381	PRETTY	1 ak su v LIST odskoky, inak 0
57382	LNS	0 ak sa nemaju vypisovat cisla riadkov
57383	KYWD	Vstupny rezim (2,3,4)
57384		Pocet riadkov v hornej casti obrazovky
		Zavisí od CSIZE a WINDOW
57391		Pocet znakov na riadok
57395	DEFAULT	Microdrive mechanizmus, cislo stanice v sieti
57396	DEFAULT	Pismeno zaradenia (T,M,N,B)
51315		Dolny byte, pozicia pointera pre CLOCK ako adresa obrazovky (kazda hodnota POKE)
51316		Horny byte, detto. (POKE len vnuti obrazovej pamate)
55365		Pocet patdesiatin sekundy, ktore CLOCK posunu o 1 sekundu
55370		54 pre 60 s za minutu
		58 pre 100 s za minutu
53723		(Pre INTERFACE 1 s vyrobnym cislom < 87315)
		Pocet znakov v riadku pre kanal t
57500		(Pre INTERFACE 1 s vyrobnym cislom > 87316)
		Detto

P R I L O H A E

Pripojenie tlaciarne:

BB03 programy su spravne tlacene ZX-tlaciarnou alebo tlaciarnou priamo kompatibilnou, alebo tlaciarnou riadenou cez kanal "t" INTERFACE 1.

>>>>>>> E N D <<<<<<<

 * Tasword coding: (C) A.Mikus 27.2.1986 *

Matisteno pro IO SVAIARM Karolinka.