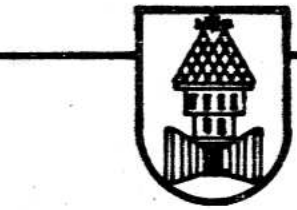
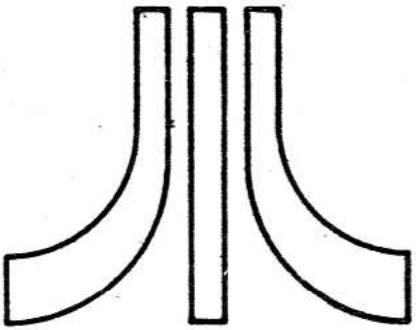




ZO SVAZARMU

ATARI KLUB

ZPRAVODAJ

2/89

Z OBSAHU:

CHANGE/FIND

Program je určen pro práci s jazykem Basic.Usnadňuje orientaci ve vytvářeném programu, simuluje funkci DUMP - výpis proměnných, určuje počet proměnných dosud v programu použitých. Trasuje výskyt proměnných v jednotlivých řádcích programu a zajišťuje změnu jejich názvu. Z těchto důvodů se jedná o výhodnou pomůcku hlavně těm, kteří se zabývají tvorbou vlastního programového vybavení v Basicu a lze ji doporučit obzvláště těm, kteří nevlastní tiskárnu.

Při tvorbě vlastního programu se často stává, že je nutno:

- trasovat výskyt proměnné v programu, např. při ladění
- ověřit, zda je nově uvažované jméno "volné" k použití
- ověřit počet dosud použitých jmen proměnných
- změnit název proměnné

Z těchto důvodů vznikla následující rutina, jež byla také s úspěchem vícekráte použita

Program začíná na ř.č. 32 700, aby jej bylo možno přihrát k většině programů, které vyšších čísel řádků málokdy používají.
Využívá se zde tří důležitých adres systému:

VNTP - adresa tabulky jmen proměnných (130,131 dec)
VNTD - adresa tabulky hodnot proměnných (132, 133 dec)
STMTAB- adresa tabulky příkazů (136, 137 dec)

a znalosti formy ukládání Basicovského programu v tokenizované formě.

Procedura se po natipování nahraje na pásek nebo disketu (LIST"C:" apod.).K hlavnímu (Vašemu) programu se přihraje (ENTER"C:") a v případě potřeby se

odstartuje z přímého režimu příkazem:

```
GOTO 32700
```

Na obrazovce se objeví nabídka-menu ve tvaru:

```
1 - FIND
2 - CHANGE
3 - List VNT
4 - END                    Cislo volby:
```

Po zadání čísla volby se provede příslušná činnost a dojde k opětovné nabídce menu až do zvolení činnosti číslo 4 - END.

Možnosti programu

1.FIND (hledání - trasování)

Na obrazovce se objeví:

```
FIND                [t.j. hledej]
```

a program očekává název proměnné, kterou má vyhledat. Po potvrzení jména pomocí <RETURN> dojde k:

a) vypisování čísel řádků, v nichž se proměnná vyskytuje (v hranatých závorkách)
b) hlášení "Neni v pameti", pokud proměnná není a nebyla v programu použita
c) hlášení "promenna pouzita jen pri ladeni programu", pokud název proměnné v tabulce jmen proměnných (VNT) existuje, ale ve vlastním textu programu není použit. Dochází k tomu vždy, jestliže použijete a později i smažete příkaz, obsahující toto jméno a program uchováte pomocí příkazu SAVE nebo CSAVE. Nevyčistíte-li VNT známým postupem LIST"C:" , NEW , ENTER"C:" , zůstane toto, nyní již nepotřebné, jméno přesto v tabulce.

Možnosti ad b) a c) potom signalizují, že zadané jméno proměnné můžete v dalších řádcích programu použít - a to i bez rizika nežádoucích komplikací přepisem její hodnoty.

2.CHANGE (změna názvu proměnné)

Na obrazovce se objeví:

```
CHANGE from:      [t.j.změň (jméno) z:]
```

a program očekává název proměnné ke změně, potvrzený

pomocí (RETURN).

Není-li toto jméno ve VNT, oznámí to na obrazovce a nabídne znovu menu. V opačném případě se zeptá:

to: (t. j. na:)

přičemž nové jméno musí mít stejný počet znaků jako jméno původní. Není-li tomu tak, program na to sám upozorní, udá požadovaný počet znaků a nedopustí jiné než správné zadání.

Důležité je pamatovat vždy na to, že:

a) název řetězcové proměnné musí končit znakem dolar

A$, JMENO$ apod.

b) název pole musí končit znakem levé okrouhlé závorky:

A(, POLE(apod.

3 - List VNT (výpis jmen proměnných)

Po této volbě dojde k výpisu seznamu jmen proměnných ("VN List") postupně po řádcích do tří sloupců a to v pořadí, v jakém byla použita.

Nakonec se vypíše celkový počet použitých proměnných ("VN Summary").

4 - END (konec)

Tato volba způsobí ukončení činnosti procedury a návrat do přímého režimu.

Poznámky na závěr

- v souladu se zvyklostmi a v zájmu splynutí formy procedury s příkazy a povely Basicu byly zvoleny anglické názvy funkcí tohoto programu

- kontrolovaný (trasovaný) program musí mít čísla řádků menší než 32700

- bližší rozbor nebyl podán. Zájemce o detaily odkazuji na publikace:

* Dočekal,P.:ABC o počítačích Atari, vydal KME ZO SSM SOUŽ Jimramov v roce 1986
* Bayer,M. a kol.:Atari 800XL- Důležité adresy systému a jejich použití při programování v Basicu, vydal AK Praha v roce 1988

- s ohledem na možné komplikace je nutno z procesu

CHANGE vyloučit řádky, obsahující DATA (hlavně alfanumerické znaky)

- zrychlení vyhledávání čísel řádků lze dosáhnout užitím procedury v TurboBasicu, popřípadě využitím strojového podprogramu (verze je k dispozici)

- vzhledem k tomu, že procedura je určena hlavně těm, kteří nemají možnost trasovat program pomocí výpisu (listingu) z tiskárny, vystupují údaje jen na obrazovku. Úprava pro tiskárnu je triviální záležitostí a nebyla proto zdokumentována.

- pomocí této procedury lze dosáhnout zajímavého efektu:
Hotový odladěný program můžeme formálně změnit tak, že bude obsahovat jen jednu proměnnou daného typu (numerickou, řetězcovou popř. pole). Poskytnete-li takto "upravený" program někomu dalšímu, jsou v něm jakékoliv úpravy (bez této procedury) natolik obtížné, že se jedná prakticky o "uzamčení" programu velmi elegantní formou. Pro nepříliš zkušeného programátora je nepochopitelný už ten fakt, že "to vůbec funguje". Listing takového programu je pochopitelně po natipování nefunkční i když originál pracuje dokonale.

Hodně zábavy a ulehčených chvil při práci Vám přeje

Ing. Aleš Oškera
Atari klub Gottwaldov
(c) 1989

```
32700 REM ****************************
32701 REM *         Change/Find      *
32702 REM *       ing.Oskera,1989    *
32703 REM *   Atari-klub Gottwaldov  *
32704 REM ****************************
32705 CLR :A=PEEK(130)+256*PEEK(131):Z=PEEK(132)+256*PEE
K(133)-1
32706 DIM X$(20),Y$(20),Z$(Z-A+1):FOR I=A TO Z:Z$(I-A+1,
I-A+1)=CHR$(PEEK(I)):NEXT I:LZ=LEN(Z$)
32707 ? :? "1-FIND":? "2-CHANGE":? "3-List VNT":? "4-END
","",:? "Cislo volby:";:INPUT #16,N:?
32708 IF N=1 THEN ? "FIND ";:GOTO 32711
32709 IF N=2 THEN ? "CHANGE from:";:GOTO 32711
32710 ON N-2 GOTO 32727,32731
32711 INPUT #16,X$:LX=LEN(X$):X$(LX,LX)=CHR$(ASC(X$(LX,L
X))+128):J=0:K=J:P=J:FOR I=1 TO LZ
```

```
32712 J=J+1:IF ASC(Z$(I,I))<128 THEN 32716
32713 K=K+1:IF J<>LX THEN 32715
32714 Y$=Z$(I-LX+1,I):IF X$=Y$ THEN POP :ON N GOTO 32717
,32724
32715 J=0
32716 NEXT I:? "Neni v pameti!":GOTO 32705
32717 A=PEEK(136)+256*PEEK(137):K=K+127
32718 Z=PEEK(A)+256*PEEK(A+1):L=PEEK(A+2):FOR I=(A+3) TO
 (A+L):IF PEEK(I)=K THEN 32723
32719 NEXT I
32720 A=A+L:IF Z<32700 THEN 32718
32721 ? :IF P=0 THEN ? "Promenna pouzita jen pri ladeni
pgm!"
32722 GOTO 32705
32723 P=1:? "[";Z;"] ";:GOTO 32720
32724 ? "          to:";:INPUT #16,Y$:LY=LEN(Y$):Y$(LY,LY
)=CHR$(ASC(Y$(LY,LY))+128)
32725 IF LX<>LY THEN ? "Pocet znaku musi byt ";LX;" !":G
OTO 32724
32726 Z$(I-LX+1,I)=Y$:FOR I=A TO Z:POKE I,ASC(Z$(I-A+1,I
-A+1)):NEXT I:GOTO 32705
32727 ? "VN-LIST :":L=1:J=L:Z=L:FOR I=1 TO LZ
32728 IF ASC(Z$(I,I))<128 THEN 32730
32729 K=I:X$=Z$(Z,K):K=K-Z+1:X$(K,K)=CHR$(ASC(X$(K,K))-1
28):? X$,:J=J+1:L=L+1:Z=I+1:IF L=4 THEN L=1:?
32730 NEXT I:? :? :? "VN-Summary = ";J-1:GOTO 32705
32731 END
```

Zrychlená verze programu se strojovou rutinou - jsou uvedeny jen změněné řádky :

```
32706 DIM X$(2 ),Y$(20),Z$(Z-A+1):FOR I=A TO Z:Z$(I-A+1,
I-A+1)=CHR$(PEEK(I)):NEXT I:LZ=LEN(Z$):GOSUB 32736

32718 Z=PEEK(A)+256*PEEK(A+1):L=PEEK(A+2):X=USR(1536,A+3
,L-2,K)
32719 IF PEEK(207)=1 THEN 32723

32732 DATA 104,104,133,205,104,133,204,104,104,133,206
32733 DATA 104,104,133,203,160,0,177,204,197,203,248,11
32734 DATA 200,152,197,206,144,244,169,0,133,207,96
32735 DATA 169,1,76,31,6
32736 RESTORE 32732:FOR I=1 TO 39:READ N:POKE 1535+I,N:N
EXT I:RESTORE :RETURN
```

Sada programů pro tiskárnu

BT 100

Sada obsahuje několik programů určených pro tiskárnu BT 100. Proramy BT100 HARDCOPY, GRAPHICS.MOZ umožňují tisk obrázků, BTV1.COM, BTV1R.COM, BTV1.BAS a ČAPEK BT 100 jsou určeny pro tisk textů. Programy BT100 HARDCOPY, BTV1.COM, BTV1R.COM, BTV1.BAS jsem vytvořil sám, ostatní (původně pracující s tiskárnou ATARI 1029) jsem předělal pro tiskárnu BT 100. Tiskárna musí být připojena přes joystickové porty. Programy ČAPEK BT 100, BTV1.BAS a BT 100 HARDCOPY jsou vytvořeny také v kazetové verzi.

BT 100 HARDCOPY

Celý program je napsán v assembleru a zkompilován. Do počítače se nahrává volbou L v menu DOSu. Umožňuje vytisknout obrázky z grafických programů pracujících v grafice 8 (DESIGN MASTER, MIKE DRAW, DIGITALE REDAKTEUR a další) na tiskárně BT 100

Po odstartování je v horní části obrazovky název, v dolní části jsou 4 řádky ,ve kterých se vypisují všechny potřebné údaje. Nyní je zde nápis Jméno souboru). Do této části programu se lze dostat kdykoliv stisknutím RESET. Jméno je třeba uvést i s názvem zařízení (D:,C:,...). Pokud je soubor nalezen, přepne se obraz do grafiky 8 a obrázek je načten do počítače a současně zobrazován. Program úmyslně ignoruje chybu 136 (konec souboru), aby bylo možné vytisknout i obrázky, které nezabírají plnou obrazovku (například takové, které vznikly v grafice 8 s textovým okénkem). Poté se zobrazí hlavní menu programu. Výběr lze provést pouhým stisknutím odpovídající klávesy.

<ESC>...Provede restart programu.Po jeho stisknutí se obrázek smaže a může být načten další soubor.

<RETURN>...Vlastní tisk obrazovky- vytiskne nahraný obrázek na tiskárně BT 100 na pozici určenou

levým okrajem (viz.volba <L>). Tisk se dá přerušit krátkým stisknutím BREAK. Tiskárna musí být připojena přes joystickové porty, jinak program ohlásí chybu 138.

<I>...Tímto povelem se celý obrázek "zinverzní". Druhé stisknutí vrací obrázek do původního stavu.

<R>...Orámování obrázku- kolem obrazovky se vytvoří rámeček, což obrázek po vytisknutí zvýrazní.

<L>...Levý okraj- umožňuje nastavit vzdálenost levého okraje obrázku od levého okraje tiskárny. Je určen číslem 0-9. Zvýšení levého okraje o 1 způsobí posunutí obrázku o 7 mm vpravo. Největší levý okraj je zvolen tak, aby tiskárna byla schopna vytisknout i obrázek sahající až na pravou stranu obrazovky.Po spuštění je hodnota levého okraje 0.

<SPACE>...Přepínání obrázek\text- Po stisknutí se objeví nahraný obrázek. Dalším stisknutím se lze dostat zpět do hlavního menu.

Program je uložen na adresách $8000-$86B0, jeho startovací adresa je $827B. Textová videoram je na adresách $9F60-$A000, obrazová na adresách $6100-$7F00.

GRAPHICS.MOZ pro BT 100

Jde o jeden z programů ze souboru MINI OFFICE II. Jeho popis je uveden v ATARI zpravodaji Tlmače 1/89.

Zvolený DATASET se dá vytisknout volbou P v režimu Edit data. Aby se vytisknul graf, je třeba pomocí kurzorových šipek najet na 3.okénko a stisknout RETURN. Nyní se zobrazí výběr I\O operací, ve kterém 3.okénko shora znamená tisk. Graf se začne tisknout okamžitě po najetí kurzorem na toto okénko a po stisknutí RETURN. Tisk se dá přerušit stisknutím ESC. Pro nedostatek paměti bylo nutné vynechat část programu, která "tiskne" (ne nahrává) graf na disketu.

* * * * *

Poznámka: Při tvorbě následujících programů (BTV1.COM, BTV1R.COM, BTV1.BAS) jsem vycházel z rutiny BT100 V1.0 KOMIN, která umožňuje tisk s různou výškou a šířkou písma. Ty jsou určeny bytem ICAX2 (v BASICu je to 3.parametr uváděný po OPEN) v IOCB podle vzorce X=16*V+S (V...násobek výšky, S...násobek šířky). V i S se mohou pohybovat v rozmezí 1-15.

BTV1.COM

Program se nahrává volbou L v DOSu. Po spuštění předá řízení BASICu. Tisk se provádí stejnými příkazy jako na firemní ATARI tiskárně. Na rozdíl od rutiny BT 100 V1 je program umístěn na jiném místě v paměti a přechod do BASICu je vyřešen tak, aby nebránil používání DOSu a disketové jednotky.

BTV1R.COM

Pracuje podobným způsobem jako BTV1.COM. Základní rutina je však přemístěna "pod" paměť ROM na místo mezinárodní znakové sady. Proto má uživatel BASICu k dispozici stejně velkou paměť jako bez této rutiny (ale za cenu ztráty možnosti používat mezinárodní znakovou sadu).

BTV1.BAS

Tato rutina umožňuje uživatelům BASICu používat ve svých programech tiskárnu BT 100 bez jakékoliv zaváděcí rutiny. Toho lze dosáhnout následujícím způsobem:

1) nahrát program BTV1.BAS příkazem LOAD (CLOAD) v BASICu do počítače a uložit jej příkazem LIST na disketu (nebo kazetu)
2) nahrát do počítače příkazem LOAD (CLOAD) upravovaný program
3) příkazem ENTER přihrát zpět program uložený v bodě 1
4) hotový program uložit na disketu (kazetu)

Upravovaný program nesmí obrahovat řádek 0 a řádky vyšší než 31849, protože na těchto řádcích se nachazí BTV1.BAS. Po spuštění upraveného programu se nejdříve inicializuje rutina pro tisk (nápis PROSÍM ČEKEJ) a dále už program pokračuje normálním způsobem. Takto upravené programy mohou být zkopírovány do TURBA 2000 programem BASIC>TURBO 2000.

ČAPEK BT 100

Je verze ČAPKA 2.0 pro tiskárnu BT 100. Je možné používat všechny funkce a povely jako v originální verzi pro tiskárnu ATARI 1029 (včetně tisku na obrazovku příkazem CONTROL P, přístroj S:). Rozdílný je jen počet písmen na řádek (54 v normální šířce, 27 při použití tučného tisku) a počet řádků na stránku.

Program se zavádí volbou L v menu DOSu. Je vytvořena také kazetová verze, která se zavádí přes TURBO 2000. Po spuštění se v textové části objeví

inverzní w, které automaticky zapíná mód tisku po jednotlivých listech papíru. V případě potřeby je možné tento mód zrušit (smazáním w), ale potom je třeba nastavit odpovídající počet řádků na stránku a další parametry pro kontinuální tisk.

Tisk se zapíná stejně jako v původní verzi příkazem CONTROL P, přístroj P:. Papír musí být založen tak, aby jeho horní strana byla kousek nad horním válcem tiskárny.

Verze ČAPEK BTZ má navíc možnost jemného řádkování, které se dá nastavit formátovacím povelem SELECT s (vytiskne inverzní s) následovaným číslem 0-15. Po příkazu s0 nebudou vynechávány žádné mezery mezi řádky, po príkazu s1 bude vynechána mezera o výšce 1/8 výšky znaku atd. Na rozdíl od originální verze tedy znamená vynechání řádku 8 krát menšího.

Pro tiskárnu BT 100 jsem upravil také následující programy:

ACTION! (kazetová i disketová verze)
DEBUG MONITOR (kazetová i disketová verze)
DIGITALE REDAKTEUR (disketová verze)

Ovládání všech programů je stejné jako u verze pro firemní ATARI tiskárnu.

Všechny uvedené upravené programy jsou volně k dispozici na referenčních kazetách a disketách Atari klubu Gottwaldov.

(c) 1989 Michal Kudr
AK Gottwaldov

=-=

--- LISTING 2 ----

```
10 ? CHR$(125)
20 OPEN #1,4,0,"K:":DIM A$(120):POKE 752,1
30 FOR I=1 TO 7:READ A1,A2,A3,A4:POKE 205,A3:POKE 206,A4
:CLOSE #2:OPEN #2,A1,A2,"O"
40    READ A$:? #2;CHR$(29);" ";A$:GET #1,A
50 NEXT I
60 FOR I=1 TO 2:READ A$:? #2;" ";A$:GET #1,A:NEXT I:END

100 DATA 3,4,24,7,Nejen u velkych pocitacu lze pouzivat
sys-  tem oken.
110 DATA 5,8,33,6,Rovnez na tvem osmibitovem ATARI je mo
zno ziskat podobne efekty.
120 DATA 2,15,30,6,Samozrejme nedosahuji te kvality  ale
 neni mozno mit vsechno.
130 DATA 18,3,21,8,Avsak nenarikej si - tento program us
pokoji tve sestnactibitove ambice.
140 DATA 10,13,21,7,Podivej se sam - copak to co mas na
obrazovce neni hezke?
150 DATA 8,9,27,7,Avsak pokud zjistis ze to neni ono tak
 tento program nemusis pouzivat.
160 DATA 4,1,23,10,Obsluha oken je velmi jednoducha - vy
uziva interni kanaly OICB.
170 DATA Souradnice leveho horniho okna se zadavaji jako
 parametry prikazu OPEN.
180 DATA Rozmery okna je treba nejprve zapsat do pameti
na adresu 205 (wx) a 206 (wy).
```

Okénka na Atari XL/XE

Jistě jste se již nejednou se závistí dívali na nové počítače firem Apple nebo Atari série ST pracující pod systémem GEM nebo podobnými. Nesmírně atraktivní forma zobrazování informací pomocí oken vedla k vyzkoušení této metody i na "vysloužilém" Atari 800XL (130XE).

Naštěstí projektanti relativně dobrého a téměř bezchybného operačního systému tohoto počítače ponechali uživateli ještě široké pole působnosti. Toho dokonalým příkladem je obsluha periferních zařízení (kterými u Atari jsou jak známo: tiskárna, magnetofon, editor, obrazovka, klávesnice a - po zavedení DOS - disketová jednotka). Každé zařízení má svůj zápis v tabulce HATABS (adresy 794-832, tedy prostor RAM), která však není vyplněna až do konce (je v ní místo na 13 zápisů). Jednotlivý zápis se skládá ze 3 bytů: jeden byte název zařízení v kódu ASCII a dva byty ukazující na počátek tabulky vektorů obsluhy procedury. V této tabulce jsou zapsány postupně vektory skoků pro příkazy OPEN, CLOSE, GET, PUT, STATUS, SPECIAL a sekvence skoku do inicializační procedury (JUMP TO).
UPOZORNĚNÍ: šest prvních zápisů má hodnotu zmenšenou o jednu proti skutečnosti. Tak např. příkaz Basicu OPEN #1,4,0,"C:" způsobí nejprve nalezení odpovídajícího zápisu v HATABS, dále 2 byty představované hodnotou 67 ("C" v kódu ASCII) jsou posuzovány jako adresa počátku tabulky vektorů procedury obsluhy magnetofonu. První dva byty této tabulky zvětšené o jednu stanoví vektor, který ukazuje místo v paměti, kde začíná procedura příkazu OPEN pro magnetofon.

Díky tomuto rozluštění obsluhy periferních zařízení je možno lehce vniknout do existujících procedur a také vytvořit vlastní zařízení. Takovým vlastním zařízením je v našem případě uvedené okénko. Je mu přidělen název "O:". Instrukce OPEN # č.kanálu,WX,WY,"O:" způsobí na obrazovce zobrazení okénka, jehož levý horní roh má souřadnice WX a WY. Protože je nutné zadat rozměry okna a pomocí instrukce OPEN je možno zadat pouze dva parametry, je třeba si pomoci příkazem POKE. Na adresu 205 je třeba uložit šířku a na 206 výšku okna, při čemž je třeba pamatovat na to, že je to nutno provést před otevřením kanálu. Instrukce OPEN zjišťuje, zda okénko je otevřeno v režimu GRAPHICS 0 (v opačném případě je hlášena chyba

č.255) a zda nezasahuje mimo obrazovku nebo není příliš malé (chyba 200).

Zobrazení textu v okénku je možno dosáhnout pomocí instrukce PUT # č.kanálu,A, kde A je kód ASCII tisknutého znaku nebo PRINT # č.kanálu,"řetězcová proměnná". Příkazy PUT # č.kanálu,125 nebo PRINT # č.kanálu,CHR$(125) způsobí vyčištění pouze části obrazovky uvnitř okna.

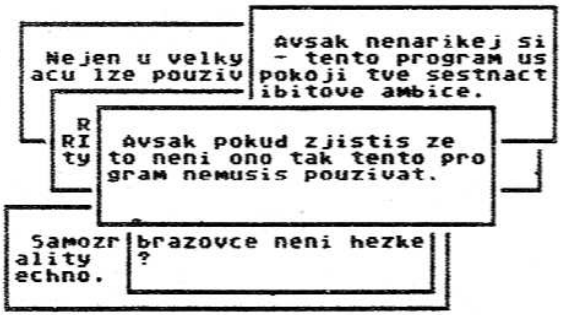
Současně po těchto příkazech, jako i bezprostředně po otevření okna je kurzor pro dané okno nastaven v jeho levém horním rohu. Na jeho polohu nemá vliv instrukce POSITION, ale mění ji příkazy přesouvající kurzor (kódy 28, 29, 30, 31). V případě, že tisknutý text je delší než "kapacita" okna, pak "přečnívající" část textu se objeví znovu začínajíc v jeho levém horním rohu. Je třeba si ještě pamatovat, že v případě otevření více než jednoho kanálu pro obsluhu oken, budou instrukce PRINT a PUT pracovat pouze v posledním okénku (tzn. v posledně otevřeném) nezávisle na tom, na který kanál se je budeme pokoušet poslat.

Program se nevešel na šestou stránku paměti, proto je umístěn na straně 158 (po přesunutí RAMTOP). Pokud si chce uživatel vybrat pro uložení programu jiné místo, je třeba změnit všechna čísla 158 v řádcích 100 a 101 a první číslo v řádku 129, a také zapsat jinou hodnotu na adresu 814.

K vyzkoušení a předvedení funkce programu je možno použít program č.2, který demonstruje okna (program č.1 musí být již předtím spuštěn).

Dle časopisu Bajtek přeložil
a upravil Ing. Libor Brázdil

Ukázka funkce programu :



--- LISTING 1 ----

```
1 REM ****************************
2 REM * Okna pro Atari 800/130 *
3 REM *      (C) 1988 Bajtek     *
4 REM ****************************
5 REM
10 AD=40448:RESTORE 100:POKE 106,158:GRAPHICS 8
20 FOR I=0 TO 37:SUMA=0:FOR J=0 TO 9
30      READ A:POKE AD+10*I+J,A:SUMA=SUMA+A:NEXT J:READ S

40    IF S<>SUMA THEN ? " CHYBA DAT V RADKU ";100+I:LIST
100+I:END
50 NEXT I:POKE 812,79:POKE 813,0:POKE 814,158
60 ? CHR$(125):POKE 205,14:POKE 206,4:OPEN #1,13,9,"D":?
 #1;CHR$(29);"    HOTOVO"
90 POSITION 2,20:END
100 DATA 14,158,248,158,248,158,251,158,248,158,1799
101 DATA 248,158,76,249,158,189,74,3,133,203,1491
102 DATA 189,75,3,133,204,165,87,240,3,160,1259
103 DATA 255,96,56,165,205,201,3,176,3,160,1320
104 DATA 200,96,165,206,201,3,176,3,160,200,1418
105 DATA 96,24,165,203,101,205,133,205,56,201,1389
106 DATA 40,144,3,160,200,96,165,204,101,206,1319
107 DATA 133,206,56,201,24,144,3,160,200,96,1223
108 DATA 165,88,133,207,165,89,133,206,162,0,1358
109 DATA 232,24,169,48,101,207,133,207,169,0,1262
110 DATA 101,208,133,208,228,204,208,238,169,0,1697
111 DATA 164,203,145,207,200,196,205,208,249,24,1881
112 DATA 169,48,101,207,133,207,169,0,101,208,1335
113 DATA 133,208,232,228,206,208,227,198,6,198,1844
114 DATA 205,165,88,133,207,165,89,133,208,162,1555
115 DATA 0,232,24,169,48,101,207,133,207,169,1282
116 DATA 0,101,208,133,208,228,204,208,238,169,1697
117 DATA 81,164,203,145,207,200,169,82,145,207,1683
118 DATA 200,196,205,208,249,169,69,145,207,24,1672
119 DATA 169,48,101,207,133,207,169,0,101,208,1335
120 DATA 133,208,169,124,164,203,145,207,164,205,1722
121 DATA 145,207,232,228,206,208,228,164,203,169,1998
122 DATA 98,145,207,200,169,82,145,207,200,196,1641
123 DATA 205,208,249,169,67,145,207,165,203,133,1751
124 DATA 4,165,204,133,5,238,4,230,5,160,1146
125 DATA 1,96,170,165,203,133,82,165,205,133,1353
126 DATA 83,165,84,72,165,85,72,165,4,133,1028
127 DATA 85,165,5,133,84,138,201,125,208,11,1155
128 DATA 104,133,85,104,133,84,230,205,76,88,1234
129 DATA 158,32,176,242,165,85,133,4,165,84,1244
130 DATA 133,5,165,4,197,203,208,8,165,205,1293
131 DATA 133,4,198,4,198,5,197,205,208,8,1168
132 DATA 165,203,133,4,230,4,230,5,165,5,1144
133 DATA 197,204,208,6,165,206,133,5,198,5,1327
134 DATA 197,206,208,6,165,204,133,5,230,5,1359
135 DATA 104,133,85,104,133,84,169,2,133,82,1029
136 DATA 169,39,133,83,169,30,32,176,242,169,1242
137 DATA 31,32,176,242,160,1,96,0,0,0,738
```

Pozor ! Listing programu 2 je na str.9

Zdroj pro BT 100

Výrobce jednojehličkové tiskárny BT 100 použil u svého výrobku ne zrovna standartní hodnotu napájecího napětí (22 V =). Protože nepatřím mezi šťastlivce, kteří mají k dispozici doporučený zdroj TESLA NZ 100 nebo zdroj FZ 1, byl jsem nucen příslušný zdroj vyrobit.

Pro zájemce o jeho výrobu předkládám schéma zapojení zdroje (obr.1), návrh plošného spoje (obr.2), chladič (obr.3) a seznam použitých součástek.

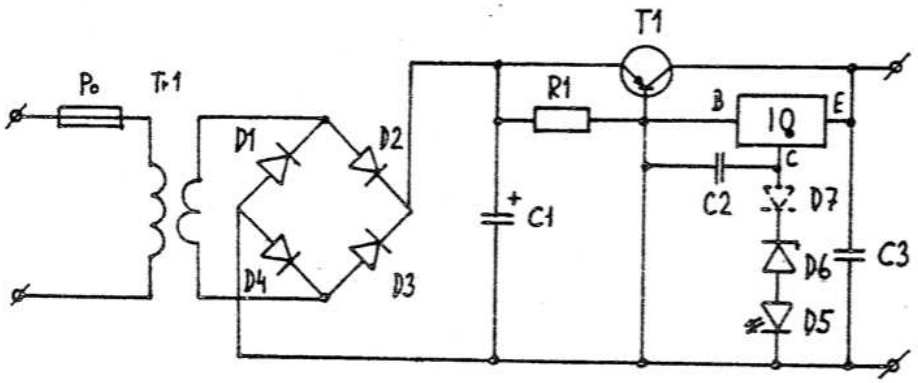
Výroba zdroje je velmi jednoduchá a při použití kvalitních součástek, by při jeho oživení nemělo dojít k problémům. Pouze v případě, že i při zatížení zdroje bude výstupní hodnota napětí vyšší než požadovaných 22 V =, doporučuji zapojit do serie s diodou D5 a D6 diodu D7 (ve schematu označeno čárkovaně a v seznamu součástek v závorce). Plošný spoj je potom nutno přerušit (naznačeno čárkovaně) a diodu vřadit do obvodu. Tranzistor T1 se musí od chladiče odizolovat slídovou podložkou. Při použití jiného typu kondenzátoru (C1) je třeba brát v úvahu výšku ohybu chladiče.

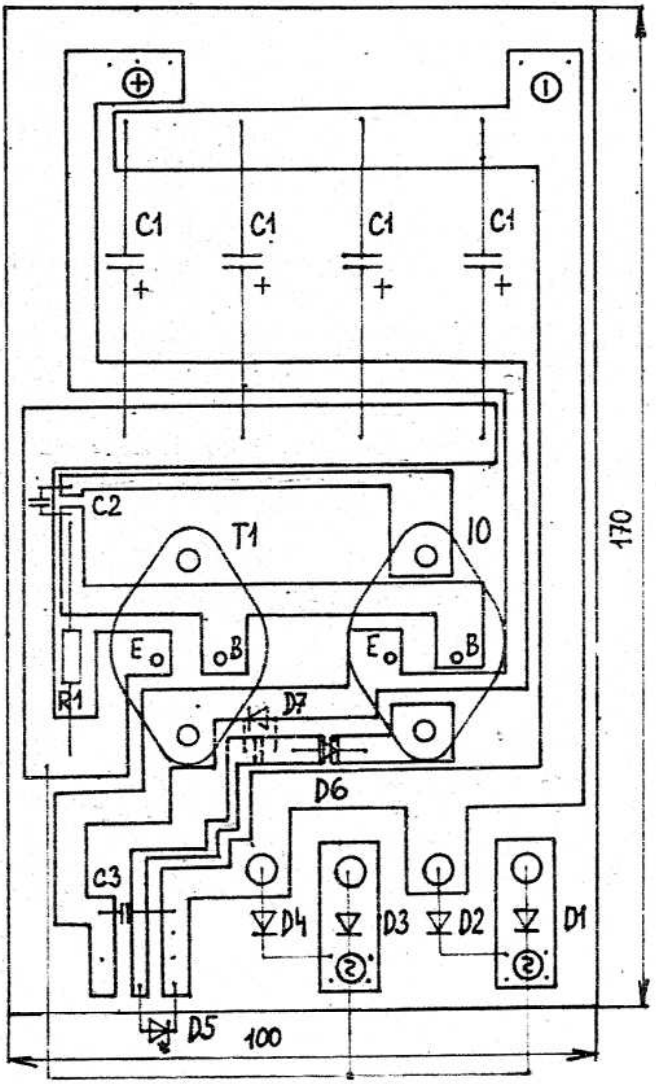
Celý zdroj umístíme ve vhodném krytu opatřeném větracími otvory. Zapojení a provedení zdroje musí odpovídat normě ČSN 341010.

Seznam součástek.

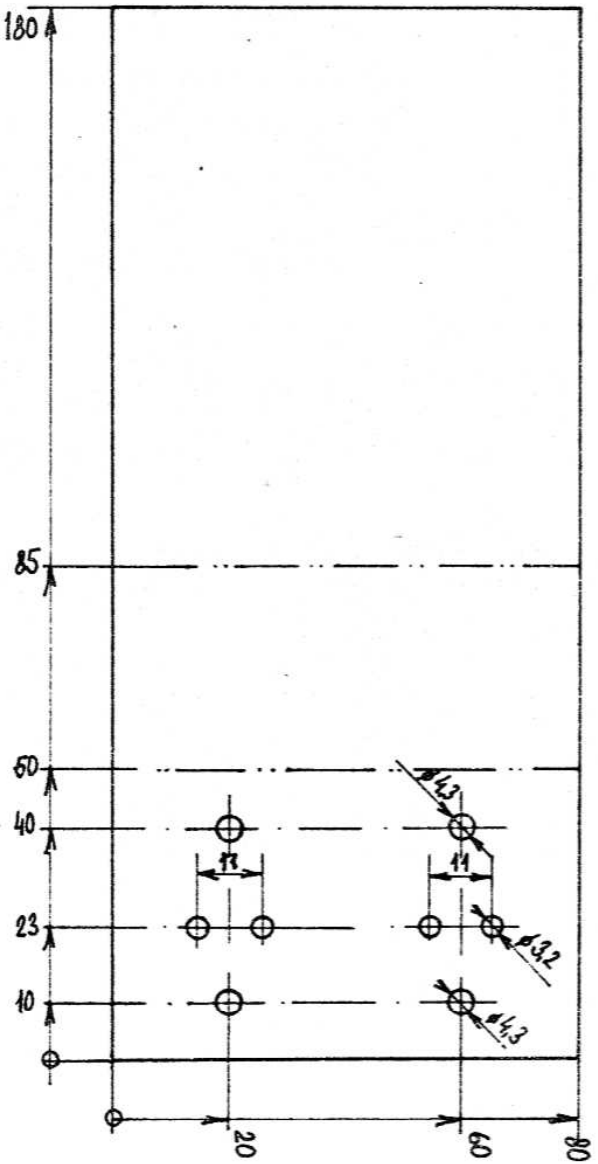
Tr1	transformátor 220 V/24 V 50 VA	
IO	MA 7815	1 ks
T1	KD 617	1 ks
D1-D4	KY 708	4 ks
D5	LQ 1731	1 ks
D6	KZ 260/5V1	1 ks
(D7	KY 130/80	1 ks)
C1	TE 677/500M	4 ks
C2,C3	TK 783 100n	2 ks
R1	TR 510 3J4	1 ks
Po	pojistka	

obr.1 schema zapojení zdroje





obr.2 návrh plošného spoje



obr.3 chladič

Připravil Ing. Stanislav Čechman
Atari klub Gottwaldov

Dělení s velkou přesností

Jedním z nedostatků programovacích jazyků je malá přesnost výpočtu. Převážně je to 6-9 desetinných míst. Avšak ne vždy je tato přesnost dostačující.

Uvedený program provádí dělení s přesností na n desetinných míst. Po spuštění programu je třeba zadat, s jakou přesností chceme provádět výpočty. Dále zadáme 2 čísla, která chceme dělit. Musí to být čísla různá od nuly.

Program je napsán v jazyku Turbo Basic XL, avšak prakticky beze změn (pouze příkazy s instrukcemi Input "...") může být použit i ve standartním Basicu. Avšak vzhledem k rychlosti výpočtu doporučujeme používat Turbo Basic XL nebo ještě lépe jeho kompilátor.

Dle časopisu Bajtek přeložil
a upravil Ing. Libor Brázdil

```
10 DIM D1$(100),D2$(100),A$(100)
20 INPUT "ZADEJ PRESNOST ",DOK
30 INPUT "ZADEJ DELENEC ",D1
40 INPUT "ZADEJ DELITEL ",D2
50 ? :? D1;"/";D2;"=";
60 IF D1>=D2 THEN 500
70 D1$=STR$(D1):D2$=STR$(D2)
80 DL1=LEN(D1$):DL2=LEN(D2$)
90 A$=D1$
100 ? "0.";
110 IF VAL(A$)<D2 THEN 150
120 A$(LEN(A$)+1)="0"
130 ? "0"
140 GOTO 110
150 REM
160 GOTO 710
500 D1$=STR$(D1):D2$=STR$(D2):A$=""
510 DL1=LEN(D1$):DL2=LEN(D2$)
520 A$=D1$(1,DL2):DLX=LEN(A$)
530 X1=VAL(A$)
540 IF X1<D2 THEN 800
545 DZ1=DL1-DLX
550 W1=INT(X1/D2)
560 ? W1;
570 A$=STR$(X1-W1*D2)
580 IF DZ1<1 THEN 700
590 A$(LEN(A$)+1)=D1$(DLX+1,DLX+1)
600 DZ1=DZ1-1:DLX=DLX+1
610 X1=VAL(A$)
620 GOTO 550
630 RUN
700 PRINT ".";:D=0
710 A$(LEN(A$)+1)="0"
720 X1=VAL(A$)
730 W1=INT(X1/D2)
740 ? W1;
750 A$=STR$(X1-W1*D2)
760 D=D+1:IF D=DOK THEN RUN
770 GOTO 710
800 REM
810 A$(LEN(A$)+1)=D1$(DLX+1,DLX+1)
820 DLX=DLX+1:X1=VAL(A$)
830 GOTO 545
```

BASIC XE

2.Část

8. Vstupně-výstupní operace

Basic XE pracuje se všemi periferiemi, t.j.:

C: - vstup a výstup, nelze však používat souběžně obě funkce

D1: až D8: - vstup a výstup, pracuje i simultánně

E: - (editor) - vstup a výstup. Default hodnota čísla kanálu pro E: je Basicem XE určena 0.

K: - jen vstup

P: - paralelní port modulu 850, většinou se využívá pro paralelní tiskárnu. Jen výstup

R1: až R4: - sériový port RS-232 modulu 850. Vstup a výstup. Na tyto porty lze připojit periferní zařízení s rozhraním RS-232 (V 24): terminály, plotry a modemy.

T: - vstup a výstup.

Pozor při využívání kanálu #7! Může to znemožnit funkci příkazu LPRINT, popř. jiných příkazů.

INPUT

Rozšířením oproti Atari Basicu je možnost tisku zprávy ????? u příkazu INPUT:

```
10 INPUT "Pořadí, 'jméno ))",A(x),Jméno(x;)
```

což je pochopitelně názornější než tisk standardního "?", a dále možnost čtení proměnné pole, popř. i dílčího řetězce:

```
15 INPUT #4,. A(5),X$(6,13)
```

LPRINT(LP.)

Příkaz LPRINT je určen pro výstup na tiskárnu, pro ni není nutno otevírat žádný kanál. Protože však u většiny tiskáren nefungují s příkazem LPRINT tiskové oddělovače (čárka, středník), doporučuje se používat raději příkaz PRINT#.

TAB

TAB[#kanál,]aexp?

Jak je vidět ve formátu, vztahuje se příkaz TAB k číslu kanálu, což je velmi výhodné.

Upozornění:
* není-li definován kanál, vztahuje se TAB k obrazovce
* aexp udává číslo sloupce, po němž se provádí TABelace; první sloupec je očíslován 0.
* je-li aexp menší než momentální pořadí (počet sloupců), vytiskne se (RETURN).
* počet sloupců se uchovává pro každé zařízení zvášť (v Aux 6 IOCB) a po každém CR (návrat vozíku) dojde k jeho vynulování.

fTAB

TAB(aexp)

Činnost funkce TAB je stejná jako u příkazu TAB, používá se v příkazech PRINT a PRINT USING. Zápis funkce musí být následován středníkem (;):

```
!éé PRINT#3;"source:"TAB(20);20;TAB(30);30
```

POZOR!: funkce TAB způsobí výstup příslušného počtu mezer na jakékoliv zařízení, proto ji používejte jen spolu s PRINT a PRINT USING.

9. Speciální příkazy I/O

Poté, co se seznámíte s dalšími příkazy, rozšířenějšími možnostmi vstupu a výstupu, doporučujeme Vám chvíli si s nimi "pohrát", abyste získali dokonalý přehled co dokáží a mohli je potom efektivně využívat v programech.

PRINT USING

PRINT[#kanál I,I]USING aexp, exp1[,exp2...]
[;]

Příkaz PRINT USING umožňuje výstup dat v požadovaném formátu, kde aexp je řetězec, specifikující tento formát pro výraz exp. Specifikátory formátování jsou

* + $, .

a musí být uzavřeny do uvozovek (") - musí být řetězeny.

Význam specifikátorů:

- # doplnění mezer
  doplnění nulami
- * doplnění hvězdičkami
- . poloha desetinné tečky (vkládá tečku)
- , poloha čárky (vkládá čárku)
- + tisk znaménka plus nebo minus
- - tisk znaménka minus
- $ tisk znaku $

Numerické formátování

* : Je-li ve výstupujícím čísle více číslic než udává formátové pole, umístí se číslo do formátového pole zprava a zbývající číslice se odřežou - viz příklad: (kolmé závorky v ukázkách tisku jsou použity pro názornost - určují hranici tiskové zóny, rezervované specifikátory formátovacího pole).

Hodnota	Formát	Výstup
123	######	| 123|
123		|0123|
123	****	|*123|
1234	######	|1234|
12345	######	|2345|

Nechcete-li dopustit situaci v posledním příkazu, použijte příkaz SET 14,1 - blíže viz příslušná kapitola.

. : Udává polohu desetinné tečky. Všechny další pozice budou obsazeny číslicemi. T.j. má-li číso méně číslic v desetinné části, než vyžaduje formát, jsou vyplněny nulami. Obsahuje-li číslo více desetinných míst, než připouští formát, dojde automaticky k zaokrouhlení. Případná druhá desetinná tečka není registrována jako formátovací specifikátor a ukončí formátovací pole.

Hodnota	Formát	Výstup
12.488	####.##	| 12.49|
123.4	####.##	|123.40|
2.35	**.**.	|*2.35.|

, : Čárka udává polohu čárky ve vystupujícím čísle. Specifikuje-li formát tisk čárky, před níž jsou doplňovací znaky (#, ,*), natisknou se příslušné znaky i místo čárky. Čárka je respektována pouze vlevo od desetinné tečky, v opačném případě ukončí formátové pole.

Hodnota	Formát	Výstup
5216	##,####	| 5,216|
3	*,****	|*****3|
4175	#,####.	|4,175.|

+ - : Znaménko plus udává, že se v každém případě má vytisknout znaménko (+ nebo -). Naopak specifikátor minus zapříčiní tisk ppuze znaménka "-" u záporného čísla a mezery u čísla kladného.

Hodnota	Formát	Výstup
43.7	+####.##	|+ 43.70|
-43.7	+####.##	|- 43.70|
23.58	-???.??	| 023.58|
-23.58	-???.??	|-023.58|

Lze využít i principu tzv. plovoucího znaménka. Znaménko se umístí na první pozici formátovacího pole a další pozice až po desetinnou tečku. To způsobí, že se příslušné znaménko vytiskne bezprostředně před první číslici a ostatní fungují jako doplňovací specifikátory.

Hodnota	Formát	Výstup
3.75	+++.##	| +3.75|
3.75	---.##	| 3.75|
-3.75	---.##	| -3.75|

Znaménko plus a minus lze použít v obdobném smyslu i na konci formátovacího pole (na poslední pozici). Ukončuje potom formát a tiskne skutečné znaménko:

```
Hodnota    Formát      Výstup
---------------------------
 43.17     ***.**+     |*43.17+|
 43.17     ???.??-     |043.17 |
-43.17     ####.##+     | 43.17-|
```

$: znak "dolar" indikuje potřebu tisku "$" před vystupující číslo a to buď jako pevný nebo tzv. plovoucí znak (obdoba znaménka). Vždy však musí být ve formátovacím poli použit jako první resp. druhý znak bezprostředně za znaménkem "+" nebo "-":

```
Hodnota    Formát      Výstup
---------------------------
 34.2       $###.##     |$34.20|
 34.2      +$###.##     |+$34.20|
 34.2      -$###.##     | $34.20|
-34.2      +$####.##    |-$ 34.20|
```

Plovoucí znak "$" je na první (resp. druhé) a dalších pozicích formátovacího pole až po desetinnou tečku:

```
Hodnota       Formát         Výstup
-----------------------------------
(4 )34.2        $$$$$.##     |  $34.20|
(4 )34.2       +$$$$$.##     |+  $34.20|
-72692.41     $$$,$$$.##+     |$72,692.41-|
```

Upozornění:
* v jednom formátovacím poli smí být pouze jen jeden plovoucí znak,
* nepoužívejte +,- a $ v jiných než popsaných situacích, jinak nelze zaručit očekávaný výsledek.

Formátování řetězců

Pro formátování řetězců jsou určeny tyto specifikátory:

% = zarovnání zprava
! = zarovnání zleva

Obsahuje-li řetězec více znaků než udává formátovací pole, dojde ke zkrácení řetězce.

```
Řetězec         Formát      Výstup
-----------------------------------
"Basic XE"      9*%         | Basic XE|
"Basic XE"      9*!         |Basic XE |
```

```
"Basic XE"      5*%          |Basic|
"Basic XE"      5*!          |Basic|
```

Vložené znaky

Do formátu lze vkládat na libovolné místo zvolený znak, a to pomocí specifikátoru "/". Tento znak neukončí formátovací pole ale umožní vytisknout na udané pozici vložený znak tak, jak je.

```
Hodnota          Formát                    Výstup
-------------------------------------------------
408446.3099      (####/)####/-#####        |(408)446-3099|
"OSS"            %/.%/.%/.                 |O.S.S|
```

Normal/inverse

Tyto dva příkazy umožní používat v programech obě formy zobrazení podle potřeby.

BPUT

```
BPUT#kanál,aexp1,aexp2[,bank]
```

Příkaz BPUT způsobí výstup datového bloku na zařízení, otevřené na kanálu #. Blok dat začíná na adrese aexp1 a je dlouhý aexp2 bytů. Nacházíte-li se v módu EXTEND, můžete si určit číslo paměťového bloku (bank).

Přitom aexp1 smí být paměťová adresa nebo adresa řetězce (viz ADR).

BGET

```
BGET # kanál,aexp1,aexp2 [,bank]
```

Příkaz BGET načte ze zařízení otevřeného na kanále # blok dat v aexp2 bytech a uloží je od adresy aexp1. Jako u BPUT smí být aexp1 adresa řetězce. V tomto případě BGET nemění délku tohoto řetězce.

RPUT

RPUT # kanal,exp.[,exp,..]

Příkaz RPUT umožní zapsat na zařízaní, otevřené na kanálu #, záznamy dané konstantní délky. Každé "exp" reprezentuje jeden prvek pole v záznamu. Aritmetické (číselné) pole sestává z 1 bytu, indikujícího typ číselného údaje, a 6 bytů čísla s plovoucí desetinnou tečkou. Retězcové pole sestává z 1 bytu, indikujícího typ řetězce, 2 bytů délky LEN, 2 bytů dimenzace DIM a dále DIM bytů dat.
Upozornění : Data, nahraná tímto způsobem nelze nechat vstoupit pomocí INPUT.

RGET

RGET # kanál, var[var...]

Příkaz RGET umožnuje čtení záznamů dané délky ze zařízení, otevřeného na kanálu #, přičemž vstupující údaje musí být téhož typu (číslo nebo řetězec).
Upozornění - příkaz RGET nelze použít pro proměnné typu nvar nebo svar. Prvek pole musí nejprve vstoupit do pomocné proměnné avar nebo svar a teprve potom jej lze přiřadit do příslušného prvku.

BSAVE

BSAVE aexp1, aexp2, "filespec"

Příkaz slouží k zápisu binární podoby záznamu ve standartním formátu Atari DOS (s hlavičkou), takže jej lze později nahrát na správné místo paměti.
Aexp1 je startovací adresa, aexp2 je konečná adresa ukládané pamětové oblasti, jež je zapsána bez vektoru RUN ani INIT s jednoduchou hlavičkou.

BLOAD

BLOAD "filepec"

Příkaz umožnuje načtení binárního záznamu zapsaného pomocí BSAVE. Lze jej využít i ke čtení podprogramů USR, vytvořených pomocí assembleru (MAC/65 apod.)
Upozornění : BLOAD nekontroluje adresy z hlavičky.

Tímto příkazem lze snadno vymazat důležité části paměti! Příkaz načte soubory libovolného počtu segmentů, ale ignoruje vektory RUN a INIT. Obsahuje-li soubor vektor RUN, lze jej spustit pomocí příkazů:

```
SET 8,0:A=USR(DPEEK($2E0)).
```

10. Tvorba disk. souborů

Příkazy této kapitoly rozšiřují možnosti diskových operací přímo z BASICu XE. Jsou to: DIR, PROTECT, UNPROTECT /UNP./, RENAME a ERASE.

DIR

```
DIR /"filespec"/
```

Příkaz DIR má podobný význam jako DIR v DOS XL. Nespecifikuje-li se filespec, znázorní se všechny soubory na D1: ve formě seznamu.

Příklad: DIR"D:*.COM" znázorní všechny soubory označené rozšiřujícím výrazem.COM

PROTECT

```
PROTECT "filespec"
```

Příkaz zabezpečí disketový soubor bez nutnosti přechodu do DOS. Význam PROTECT je stejný jako příkaz PRO v DOS XL a identický k LOCK v Atari DOS.

UNPROTECT (UNP.)

```
UNPROTECT "filespec"
```

Příkaz odjišťuje záznamy, zabezpečené pomocí PROTECT.

RENAME

RENAME "filespec,filename"

Příkaz k přejmenování souboru filespec, kde filename je nové jméno souboru. Čárka mezi filespec a filename je povinná!

ERASE

ERASE "filespec"

Příkaz vymaže všechny nechráněné soubory, které jsou označeny alespoň částečně shodně jako filespec. Např. ERASE"D:*.BAK"vymaže všechny soubory, jejichž označení končí na .BAK.

11. Cykly a skoky

Příkazy této kapitoly jsou: FOR, WHILE a GOTO, spolu s POP. Příkaz cyklu FOR/NEXT/STEP je znám z Atari Basicu, stejně jako GOTO (G).

WHILE/ENDWHILE

WHILE aexp
(instrukce)
ENDWHILE

Instrukce v těle cyklu WHILE jsou vykonávány tak dlouho, dokud je výraz aexp nenulový (t.j. kladný nebo záporný), tzn. že nemusí být vykonány ani jednou.

POP

Příkazu POP se používá, chcete-li:
1/ opustit předčasně smyčku
2/ opustit program(GOSUB)
3/ předefinovat předchozí hodnoty proměnných typu LOCAL bez EXIT.

Pozor: počet příkazů POP musí odpovídat počtu odstraňovaných hodnot ze zásobníku!

12. Stavové instrukce

Tyto instrukce připouštějí vykonání části programu pouze za určitých podmínek. Jsou to: IF/THEN, IF/ELSE/ENDIF a ON. První a poslední jsou totožné s Atari Basicem.

IF/ELSE/ENDIF

```
IF aexp
(instrukce)
(ELSE
(instrukce))
ENDIF
```

Je-li výraz aexp nenulový (true), vykonají se instrukce mezi aexp a ELSE, instrukce mezi ELSE a ENDIF budou přeskočeny a naopak. Výraz ELSE je nepovinný.

13. Ošetření chyb

Příkaz a funkce této kapitoly umožňují odhalit a odstranit důsledky chyb, vzniknuvších při běhu programu a nepřipustit tak zastavení programu. Jedná se o příkaz TRAP a funkci ERR. Příkaz TRAP je znám z Atari Basicu, lze pouze doplnit, že zmenšení TRAP se dosáhne nejen udáním čísla řádku >32767, ale i = 0 (např. TRAP 40000, TRAP 0).

f ERR

```
ERR(aexp)
```

Tato fce umožnuje nalezení čísla chyby a řádku, v němž se vyskytla. Zadá-li se aexp = 0, udá fce číslo chyby, zadá-li se aexp = 1, udá fce číslo řádku s chybou. Pro jiné hodnoty aexp není definována.

S výhodou lze vytvářet vlastní TRAPovací rutiny, kde po TRAP č.ř. lze použít v příslušném č.ř. příkaz

GOTO ERR(1)+40 a pod.

14. Operace s řetězci

V této kapitole jsou popsány funkce, umožňující oproti Atari Basicu operace s řetězci.

f FIND

FIND(sexp1, sexp2, aexp)

Funkce vyhledá dílčí řetězec sexp2 v řetězci sexp1 od jeho (aexp+1) pozice a udá jeho relativní pozici vzhledem k začátku sexp1. Nenalezne-li sexp2, výsledkem je 0.
Vtipné využití této funkce:

```
10 input "Change,Erase,        ",A$
20 On Find("CEL",A$(1,1),0)GOTO 100,200,300
30 Goto 10
```

f LEFT$

LEFT$(sexp,aexp)

Funkce vytvoří z daného řetězce sexp nový řetězec, obsahující aexp znaků řetězce sexp zleva. Např.:

A$=LEFT$("ABCDE",3) --> A$="ABC"

f RIGHT$

RIGHT$(sexp,aexp)

Vytvoří z původního řetězce sexp nový řetězec, obsahující aexp znaků řetězce sexp zprava.

f MID$

MID$(sexp, aexp1, aexp2)

Funkce vytvoří z řetězce sexp podřetězec, obsahující aexp2 znaků od pozice aexp1 původního řetězce. Např.

A$=MID$("ABCDEF",2,4) --> A$="BCDE"

f HEX$

HEX$(aexp)

Funkce má podobný význam jako STR$. Z daného čísla vytvoří řetězovou podobu čtyřmístného hexadecimálního čísla.

15. Funkce ovladačů

Kromě funkcí PADLE, PTRIG, STICK a STRIG je ve verzi Basic XE

f PEN

PEN(aexp)

Funkce, udávající pozici světelného pera. Je-li aexp=0, udává horizontální pozici, je-li aexp=1 udává pozici vertikální.

f HSTICK

HSTICK(aexp)

Funkce udává kód horizontálního pohybu udaného joystickem. Aexp udává číslo portu (0,1). Výsledkem může být:

-1 joystick doleva
 0 joystick v neutrální pozici
+1 joystick doprava

16. Grafika

Základní příkazy grafiky jsou totožné s Atari Basicem.
Navíc je příkaz

XIO(X.)

XIO 18,#6,0,0,"S:"

Tento speciální příkaz vyplní prostor na obrazovce mezi předem určenými hranicemi barvou nenulové hodnoty COLOR. Nuly ve formátu příkazu jsou povinné. Pro využití tohoto příkazu jsou nutné následující kroky:

1. Určete COLOR
2. PLOT pravý dolní roh
3. DRAWTO pravý horní roh
4. DRAWTO levý horní roh
5. POSITION kursor do levého dolního rohu
6. POKE 765, (hodnota COLOR)
7. vyvolejte příkaz XIO 18,,,,,

17. P/M grafika

Příkazy player Missile grafiky (dále PMG) umožňují využít hardware Atari způsobem nedostupným v Atari Basicu nebo Atari OS. U Basicu XE je pro tyto účely vytvořeno 7 PMG příkazů a 2 PMG funkce a dále 4 příkazy a 2 funkce podpůrné.

V PMG je každý hráč (player) ze 4 možných definován jako posloupnost paměťových míst, znázorňovaných na obrazovce jako vertikální pruh široký 8 pixelů a vysoký 128 resp. 256 pixelů ve všech 128 dostupných barvách. Každý bit nastavený na 1 je barevný a každý nulový bit je bezbarvý a proto neovlivňuje podklad - hrací pole. Pomocí PMMOVE lze hráčem libovolně pohybovat.

Střely (missile) jsou naproti tomu široké jen 2 pixely a všechny 4 střely tvoří jednoduchý paměťový blok. Každý 2 bitový pod-pruh střely má nezávislou horizontální pozici, ale stejnou barvu jako jeho rodič - hráč.

Zásady PMG

1. Hráči jsou číslováni od 0 do 3. Každý hráč má svou odpovídající střelu, jejíž číslo je o 4 větší než číslo hráče (t.j. od 4 do 7). Ve funkci BUMP jsou hrací pole barvy definované pomocí SETCOLOR, ale jsou o 8 větší než registry SETCOLOR - t.j. jsou číslovány 8 - 11.
2. Posun nahoru a dolů je u PLOT, DRAWTO a pod. určen zásadou, že souřadnice 0,0 je na obrazovce vlevo nahoře, takže zvyšování "+" hodnoty souřadnice vede k pohybu dolů. Naproti tomu PMMOVE a USTICK realizují pohyb relativní, proto "+" je posun nahoru a "-" je posun dolů.
3. " pmnum " je zkratka, označující číslo hráče (střely) a musí být číslem 0 - 3 (hráč) nebo 4 - 7 (střela).

PMGRAPHICS (PMG.)

PMGRAPHICS aexp.

Tento příkaz umožňuje popř. znemožňuje PMG. Hodnota aexp:

0 - vypnutí PMG
1 - umožňuje PMG s čarami jednoduché výšky (např. GR.15)
2 - dtto s čarami dvojité výšky (např. GR.7).

přičemž PMG.1 potřebuje dvakrát více paměti než PMG.2 (256 bytů na hráče oproti 128).

PMCOLOR (PMCO.)

PMCOLOR pmnum, aexp 1, aexp 2

Příkaz je identický k SETCOLOR. Tzn. že PMCOLOR 2,12,8 nastaví hráči č. 2 a střele 6 barvu střední zeleň (12,8). Stejně jak v normální grafice není žádný vztah mezi PMCO. a COLOR a PMG nemá defakto hodnotu barev po zapnutí resp. systémovém RESET.

PMMOVE

PMMOVE pmnum ;aexp 1 ;aexp 2

Poté, co byl hráč nebo střela definován pomocí POKE, MOVE, GET, BGET nebo MISSILE, lze je nezávisle umísťovat kamkoliv na obrazovku.
aexp 1 je absolutní hodnota pozice levého pruhu (0 - 255).
aexp 2 je specifikátor relativního vertikálního posunu. Pamatujte, že pruh hráče je 128 nebo 256 bytů paměti. Vertikální pohyb musí být doplněn o skutečný přesun bytů v pruhu - buď v paměti dál k vyšší adrese (po obrazovce dolů) nebo směrem k nižší adrese (po obrazovce nahoru). Vertikální posun je tedy v rozmezí -256 až +255 (dolů nebo nahoru o 255 pixelů), např. v závislosti na poloze joysticku:
PMMOVE 2; VSTICK (1) - pohyb hráče 2 nahoru/dolů.
Upozornění: SET 7, aexp sdělí příkazu PMMOVE, zda má hráč rolovat nebo zmizet po dosažení hranice hracího pole.

MISSILE (MIS.)

MISSILE pmnum, aexp 1, aexp 2

Příkaz MIS. způsobí výstřel střely z rodičovského hráče.
pmnum je číslo střely (4 - 7)
aexp 1 specifikuje absolutní vertikální pozici začátku střely (0 je vrchol paměti střel)
aexp 2 udává vertikální výšku střely v PMG.

Např.: MISSILE 4,40,1 :MISSILE 4,41,1

vytvoří nejprve střelu 1 pixel vysokou ve vertikální pozici 40 pixelů, ale potom ji smaže a vytvoří ji v pozici 41 pixelů, čímž se vytvoří zdání vertikálního pohybu.

PMWIDTH (PMW.)

PMWIDTH pmnum, aexp

Příkaz specifikuje šířku P/M na obrazovce. Zatímco PMG. platí pro všechny P/M, PMW. je pro každou P/M separátní. Aexp může být 1, 2 nebo 4 a určuje bitovou šířku (násobnost). Větší rozlišitelnost hráčů (šířku) lze dosáhnout i spojením dvou hráčů těsně vedle sebe.

PMCLR (PMC.)

PMCLR pmnum

Příkaz smaže P/M - t.j. vynuluje všechny byty. Upozornění: hodnoty pmnum 4 - 7 smaže VŠECHNY střely. K vymazání jen jedné z nich použijte:

SET 7,0 :PMMOVE N;Z55

f BUMP

BUMP(pmnum, aexp)

Funkce udává stav registrů srážek - t.j. 1 udává srážku a 0 indikuje, že ke srážce mezi párem udaných objektů nedošlo. Všimněte si, že aexp udává buď číslo hráče nebo hracího pole. Platné hodnoty BUMP jsou:

hráč - hráč: BUMP (0-3,0-3)
hráč - hrací pole: BUMP (0-3,8-11)
střela - hráč: BUMP (4-7,0-3)
střela - hr. pole: BUMP (4-7,8-11)

Hráč nemůže kolidovat sám se sebou. Před použitím BUMP v PGM je vhodné vynulovat registry srážek pomocí

HITCLR

jehož význam vyplývá z výše popsaného.

f PMADR

PMADR (pmnum)

Funkce PMADR udává paměťovou adresu kteréhokoliv hráče nebo střely. Používá se při MOVE, POKE, BGET a pod., popř. při získávání dat z hrací plochy. Funkce PMADR (m), kde m je číslo střely (4 - 7) dává pro všechny střely stejnou adresu.

Použití PEEK a POKE v PMG

Využívá se s výhodou k tvorbě hráčů a výběru dat z hracího pole nebo jeho části. Blíže vyplývá z ukázky

```
10 For Loc = 48 To 52
20 Read A: Poke Pmadr (0)+Loc,A
30 Next Loc
40 Data $99,$BB,$FF,$BB,$99
```

Použití MOVE v PMG

MOVE umožňuje velmi efektině tvořit velkého hráče, popř. pohybovat jím ve velkém rozsahu nehoru a dolů a tím tvořit zajímavé efekty. Vyzkoušejte si např.:

```
Move Adr(A$),Pmadr(2),128
     nebo
Poke Pmadr(1),$FF:Move Pmadr(1),Pmadr(1)+1,127
```

Použití BGET a BPUT v PMG

Používá se také pro rychlou tvorbu tvaru hráče, ale s výhodou lze obdržet příslušná data z diskety. Např.:
Bget # 3, Pmadr (0), $80 vytvoří hráče v PMG 2 ze souboru, otevřeného na # 3
Bget # 4, Pmadr (4), $500 vytvoří všechny střely a hráče v PMG 1 jediným povelem.
BPUT se používá běžněji při tvorbě pgm pro zápis tvaru hráče k pozdějšímu vyzvednutí pomocí BGET.

18. Zvuk

Operace se zvukem je totožná s verzí standartního Atari Basicu.

19. Třídění polí

Před vlastní popisem příkazů SORTUP a SORTDOWN je nutno vysvětlit několik věcí. Nejprve - těchto příkazů lze použít jen ke třídění polí - vektorů. Dále je nutno si uvědomit, že třídění probíhá podle tabulky ATASCII kódů. Tj. např.: že ve vzestupném třídění je " Zebra " před " antilopou " (kód Z=90, kód a=97), číslice jsou před velkými písmeny a znaménka jsou

vlastně uspořádána bez nějakého řádu. Pro možná nedorozumění doporučujeme změnit všechny znaky svar na inverzní a malé písmena na velká, např. pomocí této procedury:

```
800 Procedura " To Upeer Using! String$ "
810 Local I, Temp
820 For I=1 To Lem (String $)
830 Temp = Asc (String $ (I))&$7f
840 If Temp)$60 And Temp($7b Then Temp=Temp&$5f
850 String$(I,I)=Chr$(Temp)
860 Next F
870 Exit
```

Vhodným použitím středníku v definici Using lze docílit dokonce toho, že třídění proběhne jen podle zadaných pozic v řetězcích:

```
10 Dim Array$(5,20)
20 For I=1 To 5:Input "Řetězec)",Array$(I;):NextI
30 Sostup Array$ Using;3,5
40 For I=1 To 5:Print Array$(I;3,5),Array$(I;):Next
```

U numerických polí se používá za příkazem třídění označení pole následované dvěma závorkami (obdoba $ u řetězců):
Sortup A() způsobí vzestupné třídění všech prvků pole P(...)
Sortdown A() Using 3 To 5 způsobí vzestupné třídění prvků pole A o pořadových číslech 3 až 5 (včetně)

Pozor:

- Třídit lze pouze číselná pole typu vektoru, ne skutečné pole (matrice)
- číslo posledního tříděného prvku musí být)= číslu prvního tříděného prvku (nelze For I=3 To 1)
- pozor na DIMenzování

Formátování příkazu:

```
 SORTUP      array [USING[aexp 1 To aexp 2][;aexp 3,
 SORTDOWN    aexp 4]]
```

kde vysvětlivky byly podány výše

20. Pevná DATA v programu

Pro operace s pevnými daty slouží příkazy DATA(D.) , READ a RESTORE(RES.), jejichž použití je

totožné s Atari Basicem.

Poznámka: blok dat nesmí obsahovat čárku (,) a <RETURN>. Lze to ale obejít pomocí uzavření bloku do uvozovek ("..."):
DATA" data s , mezi nimi" - potom nelze použít v bloku uvozovek a <RETURN>.

21. Přímý přístup do paměti

je umožněn příkazy POKE, DPOKE, MOVE a funkcemi PEEK a DPEEK. Peek a Poke je totožné s Atari Basicem

f DPEEK

DPEEK (aexp[,bank])

Udává přímo dvoubytovou hodnotu z paměťových lokací aexp a aexp+1.

DPOKE

DPOKE aexp1,aexp2 [,bank]

Ukládá hodnotu aexpě (0 až 65535 včetně) dvoubytově do lokací s adresou aexp1 a aexp1+1.

MOVE

MOVE aexp1, aexp2, aexp3 [,bank]

Přesouvá blok dat z adresy na adresu rychlostí strojového programu. Aexp1 je počátební adresa přesouvaného bloku, aexp2 je počáteční adresa kam má být blok přesunut a aexp3 je délka bloku. Znaménko u aexp3 udává pořadí přesunu dat:

Kladné: (odkud) -> (kam)
(odkud+1) -> (kam+1)

```
          (odkud+len-1)->(kam+len-1)

Záporné:  (odkud+len-1)->(kam+len-1)
          (odkud+len-2)->(kam+len-2)
               .               .
               .               .
          (odkud)         ->(kam)
```

Je-li tedy znaménko kladné, může nastat přepsání spodní části původního bloku dat a u záporného znaménka lze přepsat horní část zdrojového, původního, bloku.

Pozor!
- během přesunu se nekontrolují adresy
- MOVE nefunguje pro přesun mezi paměťovými bloky. Nejprve je nutno blok přesunout do hlavní paměti a teprve potom do jiného bloku.

22. Aritmetické funkce

Jsou to: ABS, INT, SGN, SQR, EXP, LOG, CLOG, RND a RANDOM. Rozdíly a nové příkazy oproti Atari Basicu jsou popsány dále.

f RANDOM

```
RANDOM (aexp1 [,aexp2])
```

Tato funkce generuje náhodné celé číslo (typ INTEGER) od 0 do (aexp1-1) včetně (např. X=RANDOM (99)) popřípadě od aexp1 do aexp2 včetně (např. Y=RANDOM (10,20)).

23. Trigonometrické fce

Jsou totožné s Atari Basicem.

24. Podprogramy

Volání podprogramů je umožněno povely:
GOSUB - volá Basicovský podprogram číslem řádku
USR - volá strojový program pomocí adresy
PROCEDURE - volá Basicovský podprogram jménem

GOSUB (GOS.)

Příkaz volá nepojmenované podprogramy v Basicu, které musí končit EXIT nebo RETURN. Opouštíte-li tento podprogram (pomocí GOTO apod.) bez průchodu tímto ukončujícím příkazem, nezapomeňte užít POP!

PROCEDURE (PROC.)

PROCEDURE pname [USING rvar1 [,rvar2...]]

Označuje "hlavičku" podprogramu jménem pname (v uvozovkách!). Výraz USING umožňuje užití tzv. parametrů:

```
10 Konec=5
20 Call "Vysledek" Using 10
30 Print Konec
40 End
50 Procedure "Vysledek" Using Konec
60 Print Konec
70 Exit
```

Po spuštění tohoto programu se vytiskne nejprve 10 a potom 5. To z toho důvodu, že v podprogramu je proměnná Konec typu LOCAL (automaticky!).

POZOR:
- parametry v CALL a PROC. musí být stejného typu
- příkaz PROC. musí být jako první na řádku, jinak CALL selže! (nenajde se)
- automatické považování parametru za lokální proměnnou může způsobit potíže při operacích s řetězci:

```
10 Fun$="Plavani je sranda.":X$="Fakt?"
20 Call"Jaka sranda" Using!Fun$
30 Print Fun$,X$
40 End
50 Procedure"Jaka sranda" Using!X$
```

```
60 Print Fun$,X$
70 X$(1,5)="Choze"
80 Exit
```

Na obrazovce se objeví:

```
  Plavani je sranda.      Plavani je sranda.
  Chozeni je sranda.      Fakt?
```

Problém je v tom, že v ř.č.70 se změnou X$ změní obsah F$ ale nezmění se délka F$. Zadáte-li:

```
70 X$="XXX"
```

vytiskne se příkazem v ř.č.30:

```
  XXXvani je sranda.      Fakt?
```

Napravit to lze použitím TO:

```
20 Call"Jaka sranda" Using!Fun$ To !Fun$
80 Exit!X$
```

což způsobí kompletní okopírování obsahu X$ do F$.

EXIT

EXIT [pexp1 [,pexp2..]]

Příkaz provádí následující operace:
1) jsou-li v zásobníku nějaké proměnné (t.j. požili-li jste LOCAL nebo parametry), přemístí je EXIT do tabulky hodnot proměnných.
2) následují-li po EXIT nějaké pexp, umístí je do rvar umístěných po (za) TO v příkazu CALL.
3) prověří, zda podprogram byl volán pomocí CALL nebo GOSUB. V druhém případě simuluje činnost RETURN.

CALL

CALL pname [USING pexp1 [,pexp2...]][TO tvar[,rvar...]]

Příkaz volání procedury jménem pname. Toto jméno (na rozdíl od PROC) nemusí být řetězcová konstanta ale i proměnná, nikoli však podřetězec nebo prvek řetězcového pole (vektoru).

POZOR:
- četnost řetězení příkazů CALL je dána volnou kapacitou v zásobníku. CALL potřebuje 4 byty a

každý parametr 12 bytů.
- CALL je v režimu FAST pomalejší než GOSUB. V normálním režimu je (bez parametrů) o něco rychlejší.

F USR

je totožné s Atari Basicem. Za zmínku stojí jen otázka parametrů, jež dosud mnoha uživatelům činí potíže.

USR (aexp1 [,aexp2...]) ,kde:

aexp1 musí být celé číslo, neboť udává adresu rutiny aexp2, aexp3 ... jsou nepovinné parametry z intervalu (0,65535) a nemusí být celé - dojde automaticky k jejich zaokrouhlení.

Poté jsou ukládány do zásobníku v OBRÁCENÉM pořadí a jako poslední se uloží jejich celkový počet, jenž musí být vždy (je to 1 byte) odstraněn v rutině USR instrukcí PLA. Lze to změnit pomocí SET 8,aexp.

K o n e c

Přeložil a upravil
Ing. Aleš Oškera

O B S A H

Název : ZPRAVODAJ II/1989
Vydává : ATARI KLUB Gottwaldov
Odpovědný redaktor : Ing. Libor Brázdil
Počet stran : 40
Obálku navrhl : Ing. Aleš Oškera
Určeno : členům Atari klubů ZO Svazarmu a ČSUTS
NEPRODEJNÉ - odběr vázán na klubový příspěvek
Tisk : Rozmnožovna KR ČSUTS Gottwaldov
Předáno do tisku : prosinec 1989
Neprošlo jazykovou úpravou
370507289