zpravodaj atari klubu

příloha

1 – 4

ZPRAVODAJ ATARI klub Praha

Vydává 487. ZO Svazarmu — ATARI KLUB v Praze 4.
Šéfredaktor a vedoucí redakční rady JUDR. Jan Hlaváček.
Zástupce šéfredaktora ing. Stanislav Borský.
Obálku navrhl RNDr. J. Tamchyna.
Adresa redakce:
487. ZO Svazarmu - ATARI KLUB Praha
REDAKCE
poštovní přihrádka 51
100 00 Praha 10
Řídí redakční rada: V. Bílek, ing. J. Biskup, RNDr. J. Bok, CSc., ing. S. Borský, ing. V. Friedrich, ing. O. Hanuš, RNDr. L. Hejna, CSc., Z. Lazar, prom. fyz., CSc., F. Tvrdek, ing. M. Vavrda.
Technická redakce Otilie Strnadová.
Otisk povolen se souhlasem redakce při zachování autorských práv a s uvedením pramene. Rukopisy nevyžádané redakcí se nevracejí. Za původnost a věcnou správnost ručí autor.
Vychází šestkrát ročně. Neprodejné. Členům klubu distribuováno zdarma. Nepravidelné přílohy na objednávku jsou kompenzovány zvláštním klubovým příspěvkem.
Rozsah čísla 116 stran. Neprošlo jazykovou úpravou.
Tiskne ČTK - repro, Brandýs n/Labem
Do tisku předáno v VII/88

Vydávání schváleno OV Svazarmu Praha 4 a OŠK ONV Praha 4.
Evidenční číslo ÚVTEI 86 042.

příloha I - 1

Miloš Bayer a kolektiv

ATARI 800 XL

DŮLEŽITÉ ADRESY SYSTÉMU A JEJICH POUŽITÍ PŘI PROGRAMOVÁNÍ V BASICU

(1. část)

Recenze
RNDr. Jiří Bok, CSc.

1.0 ATARI BASIC, PEEK & POKE

Programovaci jazyk BASIC je sestaven z rady prikazovych slov, s jejichz pomoci se vytvari program podle urcitych pravidel. Pocitac prikazy vykonava za pomoci prekladace (interpret), ktery prevadi slova BASICu do strojoveho kodu. Pouziti BASICu zpomaluje rychlost zpracovani. Vetsine uzivatelu to nevadi, kratsi pracovni cas nema pri programovani vyznam. Kdo vsak ma zkusenosti s programovanim v BASICu prijde na to, ze s pouzivanou zasobou BASICovych povelu se casto dostane na hranice moznosti.Vse co by mohl realizovat s narustajicimi zkusenostmi, se v BASICu neda napsat. S prikazy PEEK a POKE mame v BASIC programu primy vliv na pametove bunky pocitace. Tak lze tvorit programy primym adresovanim.

ATARI ridi 8-bitovy procesor, ktery muze adresovat 65536 (0 - 65535) adres. Kazda z techto pametovych bunek obsahuje 8 bitu. Jedna bitova informace odpovida rozhodnuti ano-ne. Matematicky to chapeme jako 1 nebo 0. Osm bitu tvori jeden byte. Byte je nejmensi pametova jednotka kterou muzeme adresovat.Kazde pametove misto obsahuje jeden byte. Uvnitr bytu dostane kazdy bit cislo od 0 do 7. Tak vznikne 8-mistne dvojkove cislo.

Stavba bytu :

cislo bitu	7	6	5	4	3	2	1	0
bity(prikl)	1	0	0	1	0	1	1	0
vaha	2↑7 +	2↑6 +	.....	+				2↑0

Dvojkove cislo se da prepocitat na decimalni (desitkovou) hodnotu tak, ze se sectou vahy vsech bitu, ktere jsou ve stavu 1. Vahy jednotlivych bitu :-

cislo bitu	7	6	5	4	3	2	1	0
bity(prikl)	1	0	0	1	0	1	1	0
vaha	128	64	32	16	8	4	2	1

V tomto priklade bude decimalni hodnota = 128+0+0+16+0+4+2+0=150. Jestlize se prikazem POKE n,150 dosadi na adresu n hodnota 150, pak se tam dosadi nas ukazkovy byte.Jestlize se potom budeme ptat prikazem PEEK(n), najdeme hodnotu decimalni, tedy 150.
V ATARI BASICu stale pracujeme s decimalnimi cisly. Musime vsak o binarnich (dvojkovych) vedet tolik, ze pozice v byte znaci jeho hodnotu.
Bit n ma tedy vahu 2↑n. Jestlize vime, ze bit 2↑0=1, muze se pomoci PEEK a POKE vybavit. Pro programovani ve stroj. kodu jsou decimalni hodnoty mene vhodne, protoze pri hodnote napr. 150 kazdy hned nevi, ktere bity jsou nastaveny a ktere ne. Binarni zpusob psani neni svymi nekonecnymi radami nul a jednicek take zvlast vhodny. Proto se casto sdruzuji ctyri bity do jedine hodnoty. Ctyri bity mohou predstavovat decimalni hodnoty od 0(0000) do 15 (1111)

Cislo 16 je zaklad hexadecimalni (sestnactkove) soustavy:

decimalne	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
hexadecimalne	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Priklad:

binarne	0000	0010	0101	1000	1010	1111
decimalne	000	002	005	008	010	015
hexadecimalne	0	2	5	8	A	F

Nyni mohou byt hex. cislice stejne jako hex. cisla razena jedno za druhym jako binarni cisla k binarnim cislicim, nebo dec. cislice k dec. cislum.

Binarne	0000	0000	0001	0010	0100	1101	1111	1111
decimalne		000		18		77		255
hexadecimalne	0	0	1	2	4	D	F	F

Jeden byte muze obsahovat decimalni hodnoty od 0 do 255. Tyto se daji hex. zpusobem napsat dvema cislicemi(00-FF). Dvojkove cislo muzeme snadno prevest na sestnactkove a zpet, nebot ctverici dvojkovych cislic odpovida jednoznacne jedna cislice sestnactkova; u desitkoveho cisla je prevod slozitejsi. Kdo pracuje s ATARI BASICem, nemusi se o hexadecimalni soustavu nijak zvlast starat, vystaci s malymi znalostmi binarni soustavy. Orientace v pameti je vsak v hexadecimalni notaci snazsi.

Drive nez se podivame na zaklady pameti, budeme si trochu hrat s cisly a sou stavami, ktere jsme si prave predstavili. Ciselna soustava se sklada z mnoziny cislic, ktere lze radit do libovolnych kombinaci.Postaveni cislice v cisle udava jeji hodnotu. Obycejne pocitame s decimalni soustavou. Je zalozena na cislicich 0-9. Kazde misto v decimalni soustave odpovida jedne mocnine desiti. Posledni nebo nulte misto (jednotky) ma hodnotu 10↑0 = 1, druhe misto (desitky) ma hodnotu 10↑1 = 10, treti misto (stovky) ma hodnotu 10↑2= 100. Cislo 417 tedy znamena 4*10↑2 + 1*10↑1 + 7*10↑0 = 400+10+7 = 417. Obdobne take pracuji ostatni ciselne soustavy. Tak pouziva binarni soustava jen dve cislice (0,1), kazde misto v urcitem binarnim cisle vyjadruje hodnotu odpovidajici mocnine zakladu (2). Hexadecimalni soustava pouziva 16 cislic (0-F). Stejne tak kazde misto v hexadecimalnim cisle znamena odpovidajici hodnotu mocniny 16.

Cislo 3E predstavuje hodnotu 3E = 3*16↑1 + E*16↑0 = 3*16 + 14*1 .V dec. soustave 62, binarni 0011 1110. Mezera mezi bloky slouzi jen k lepsimu prehledu a nuly na vyssich mistech lze vypustit jako v dec. soustave.
8 bitu jednoho bytu lze delit na dve casti(nibble).Jeden nibble muze obsahovat dec. hodnoty 0-15, hex. 0-F.
Jedna hexadecimalni cislice odpovida jednomu ctyrmistnemu binarnimu cislu.

Pro nektere ucely se pouzivaji jeste jine ciselne soustavy, napr duodecimalni. Zaklad je 12, obsahuje cislice 0-B. Stare merici a pocetni systemy tuto bazi casto pouzivaly (cas, clo). Obliba se zakladala na tom, ze 12 lze delit 2, 3, 4. Jelikoz prevody mezi soustavami jsou namahave a protoze pocitac byl postaven proto, aby neprijemne prace delal, sestavime program. Program DEZXBIN prevadi decimalni cislo <0-255> na 8mi mistne binarni cislo. Neni zabezpecen proti chybnym udajum.

```
0 REM DEZXBIN.
1 REM ******************************
2 REM *  PREVOD  DEC. - BIN        *
3 REM ******************************
10 DIM B(7)
20 FOR J=0 TO 7 :B(J)=0: NEXT J
30 ? CHR$(125):? " ZADEJ DECIMALNI CISLO "
40 ?:? " OD 0 DO 255 A >RETURN< " :?
50 INPUT D
60 R=D
90 IF D>127 THEN D=D-128: B(7)=1
100 IF D>63 THEN D=D-64 : B(6)=1
110 IF D>31 THEN D=D-32 : B(5)=1
120 IF D>15 THEN D=D-16 : B(4)=1
130 IF D> 7 THEN D=D-8  : B(3)=1
140 IF D> 3 THEN D=D-4  : B(2)=1
150 IF D> 1 THEN D=D-2  : B(1)=1
160 B(0)=D
170 ?:? "    " ;R; "    =
 ";B(7);B(6);B(5);B(4);B(3);B(2);B(1);B(0)
180 ?:? "DALSI CISLO? J/N"
190 IF PEEK (764)=1 THEN POKE 764,255: GOTO 20
200 IF PEEK (764)=350 THEN POKE 764,255: END
210 GOTO 190
```

```
10      Promenna B je dimenzovana na 8 bitu (0-7), prijme 8 bitu
20      CHR$(125) smaze(clear) obrazovku
90-150  Zde se zjistuje, ktere bity jsou nastaveny.  Je-li
        zadane dec.  cislo vetsi nez 127, musi byt nasazen bit
        7, tj.  hodnota 128.  Je-li to tak, bude zadane cislo D
        zmenseno  o 128.  Je-li stale >63, pak je nutne nastavit
        bit 6, tj.  hodnota 64, atd..
160     tento radek vytiskne vysledek
180     Zde zadame dalsi cislo
190-200 Dotaz na adresu 764, funkce bude vysvetlena pozdeji.
210     Vraci na 190.  Tim vznikne smycka, kterou ukonci  pouze
        povel J/N.
```

Protoze program neni dale chranen, lze ho prerusit povelem BREAK. Muzeme si povsimnout, ze radky 90-150 lze efektivneji sestavit pomoci smycky FOR - NEXT. Nahradte tedy 90-150 radkami 90-130 podle programu DEM0001. Kratsi program nemusi byt rychlejsi. Vypocet druhe mocniny v radku 110 je dlouhy.

```
0       REM DEM0001.
1       REM ******************************
2       REM * POMALEJI S FOR-NEXT        *
3       REM ******************************
110     FOR J=7 TO 1 STEP -1: BW=2↑J
120     IF D>BW-1 THEN D=D-BW: B(J)=1
130     NEXT J
```

Nasledujici program pracuje opacne. Uzivatel zada 8-mistne bin. cislo a program ho prevede na decimalni.

```
0       REM BINXDEC
1       REM ********************************
2       REM * PREVOD BIN. NA DEC.          *
3       REM ********************************
10      DIM B$(8), A(8)
20      A(8)=1: A(7)=2: A(6)=4: A(5)=8: A(4)=16: A(3)=32:
21      A(2)=64: A(1)=128
30      ? CHR$(125): ? "  ZADEJ BINARNI CISLO  "
40      ?: ?  " S OSMI CISLICEMI A RETURN ":?
50      INPUT B$
60      FOR J=1 TO 8
70      IF B$(J,J)= "1" THEN D=D+A(J)
80      NEXT J
100     ?: ? " CISLO ";B$; " = "; D
110     ?: ? " DALSI CISLO?  J/N "
120     IF PEEK(764)=1 THEN POKE 764,255: D=0: GOTO 30
130     IF PEEK(764)=35 THEN POKE 764,255: END
140     GOTO 120
```

10 B$ dimenzovan na 8 mist, A$ je dimenzovan na 9 mist(0-8), ale pouziva se jen (1-8), tim budou vztahy s A$ jasnejsi. Lze take pracovat s A(0) - A(7).

20 Osmi promennym A(8) - A(1) budou prirazeny decimalni hodnoty, ktere odpovidaji hodnotam odpovidajicich binarnich cisel

60-80 Tato smycka cte jednotlive znaky v zadanem osmimistnem bin. cisle, ktere budou uvedeny jako B$. Prikaz B$(n,m) vyjme z B$ cast zacinajici na n-tem miste. Prikaz B$(J,J) vyjme tedy jeden znak z B$ a to ten, ktery je na J-tem miste. Znaky v retezci se ctou (pocitaji) zleva doprava.Prvni znak v B$ je bit 7 s decimalni hodnotou 128. Dec. hodnota prislusna kazdemu bitu se priradi k promenne A(J) na radku 20.

Pri prevodu na hex. cisla je jeste dalsi potiz. K numerickym hodnotam (0-9) pristupuji cislice A-F, ktere pocitac muze zpracovat jako retezec. Proto je bezne hexadecimalni cisla zadavat jako predem stanoveny retezec (243=$F3).

```
1       REM **********************************
2       REM * PREVOD DEC. NA HEX.            *
3       REM **********************************
10      DIM HEX$(16),H$(1),L$(1)
20      HEX$="0123456789ABCDEF"
30      ? CHR$(125):? " ZADEJ DEC. CISLO "
40      ?: ? " OD 0 DO 255 A RETURN ":?
50      INPUT D
60      R=D
70      A=INT(D/16): H$=HEX$(A+1,A+1)
80      B=D-A*16: L$=HEX$(B+1,B+1)
170     ?: ?"    ";R;"  =  ";H$;L$
180     ?: ?" DALSI CISLO  J/N ? "
190     IF PEEK (764)=1 THEN POKE 764,255: GOTO 30
200     IF PEEK (764)=35 THEN POKE 764,255: END
210     GOTO 190
```

10 HEX$ ma prijmout 16 cisel, ktera budou pouzivana.

Hledane cislo je slozeno ze 2 cislic. Horni misto(HI) je v H$, dolni(LO) v L$

20 Zde se ulozi cislice 0-F do HEX$

50 Zadane cislo(dec 0-255) se priradi do D

70 Jestlize D/16, odpovida celociselna hodnota (INTeger) vysledku na hornim miste HEX. cisla. Protoze je tento vysledek dimenzovan, musime jeste udat HEX$(A+1,A+1). Hexadecimalni cislice 0 je prvni znak HEX$, musi se k vysledku pricist 1.

80 Zbytek horniho deleni odpovida dolnimu hex. cislu. D-A*16 udava zbytek v dec. forme, ktery se prevede na hex. cislici stejnym zpusobem

Pri prevodu HEX. na DEC cisla mame stejny problem s cislicemi A-F a musime je opet prevadet.

```
0       REM HEXXDEC
1       REM ************************
2       REM * HEX NA DEC CISLO      *
3       REM ************************
10      DIM HEX$(2),H$(1),L$(1)
30      ? CHR$(125):? "ZADEJ HEX. CISLO "
40      ?: ?" O DVOU MISTECH A RETURN ":?
50      INPUT HEX$
60      H$=HEX$(1,1): L$=HEX$(2,2)
70      IF H$="A" THEN H=10: GOTO 150
80      IF H$="B" THEN H=11: GOTO 150
90      IF H$="C" THEN H=12: GOTO 150
100     IF H$="D" THEN H=13: GOTO 150
110     IF H$="E" THEN H=14: GOTO 150
120     IF H$="F" THEN H=15: GOTO 150
130     H=VAL(H$)
150     IF L$="A" THEN L=10: GOTO 290
160     IF L$="B" THEN L=11: GOTO 290
170     IF L$="C" THEN L=12: GOTO 290
180     IF L$="D" THEN L=13: GOTO 290
190     IF L$="E" THEN L=14: GOTO 290
200     IF L$="F" THEN L=15: GOTO 290
210     L=VAL(L$)
290     D=H*16+L
300     ?:? "   "; HEX$ ; "  =  " ;D
310     ?:? " DALSI CISLO J/N ? "
320     IF PEEK (764)=1 THEN POKE 764,255: GOTO 30
330     IF PEEK (764)=35 THEN POKE 764,255: END
340     GOTO 320
```

10 Zadane HEX. cislo muze byt dvoumistne

60 H$ a L$ se nactou z HEX$ - INPUT

70-120 V H se zaznamena hodnota dec. cisla z H$, jestlize H$ je cislo A-F

130 Neni-li, pak lze prikazem VAL cislici v H$ prevest na numerickou hodnotu

150-210 Stejne se prevede L$

290 Hledana dec. hodnota se pak muze jednoduse vypocitat. Horni misto HEX. cisla ma hodnotu 16 (tedy H*16), dolni pak 1 (H*16+L) LM 0

Prevod binarniho cisla na hex. je kombinace jiz znamych programu.

```
0       REM BINXHEX
1       REM ***********************
2       REM * BIN NA HEX          *
3       REM ***********************
10      DIM HEX$(16),H$(1),L$(1),B$(8),A(8)
20      HEX$= "0123456789ABCDEF"
30      A(8)=1:A(7)=2:A(6)=4:A(5)=8:A(4)=1:A(3)=2:A(2)=4:
A(1)=8
40      ?CHR$(125):? " ZADEJ BIN CISLO "
50      ?:?" OSMIMISTNE A RETURN ":?
60      INPUT B$
70      FOR J=1 TO 4
80      IF B$(J,J)= "1" THEN H=H+A(J)
90      NEXT J
100     H$=HEX$(H+1,H+1)
110     FOR J = 5 TO (
120     IF B$(J,J)="1" THEN L=L+A(J)
130     NEXT J
140     L$= HEX$(L+1,L+1)
155     ?:? B$;"=";" ";H$;L$
170     ?:?" DALSI CISLO J/N ?"
190     IF PEEK (764)=1 THEN POKE 764,255: GOTO 30
200     IF PEEK (764)=35 THEN POKE 764,255: END
210     GOTO 190


0       REM HEXXBIN
1       REM ***********************
2       REM * HEX NA BIN          *
3       REM ***********************
10      DIM HEX$(2), H$(1), L$(1),B(7)
20      FOR J=0 TO 7: B(J)=0:NEXT J
30      ?CHR$(125): ? " ZADEJ HEX. CISLO "
40      ?:?" DVOUMISTNE A RETURN ":?
50      INPUT HEX$
60      H$=HEX$(1,1): L$=HEX$(2,2)
70      IF H$="A" THEN H=10: GOTO 150
80      IF H$="B" THEN H=11: GOTO 150
90      IF H$="C" THEN H=12: GOTO 150
100     IF H$="D" THEN H=13: GOTO 150
110     IF H$="E" THEN H=14: GOTO 150
120     IF H$="F" THEN H=15: GOTO 150
130     H=VAL(H$)
150     IF L$="A" THEN L=10: GOTO 220
160     IF L$="B" THEN L=11: GOTO 220
170     IF L$="C" THEN L=12: GOTO 220
180     IF L$="D" THEN L=13: GOTO 220
190     IF L$="E" THEN L=14: GOTO 220
200     IF L$="F" THEN L=15: GOTO 220
210     L=VAL(L$)
220     IF H>7 THEN H=H-8: B(7)=1
230     IF H>3 THEN H=H-4: B(6)=1
240     IF H>1 THEN H=H-2: B(5)=1
250     B(4)=H
260     IF L>7 THEN L=L-8: B(3)=1
270     IF L>3 THEN L=L-4: B(2)=1
280     IF L>1 THEN L=L-2: B(1)=1
290     B(0)=L
300     ?:?"   "; HEX$ ; " =
```

```
";B(7);B(6);B(5);B(4);B(3);B(2);B(1);B(0)
310     ?:?" DALSI CISLO J/N ? "
320     IF PEEK (764)=1 THEN POKE 764,255: GOTO 20
330     IF PEEK (764)=35 THEN POKE 764,255: END
340     GOTO 320
```

Take k obracenemu prevodu z hex. do bin. cisla nam zbyva vysvetleni:

```
1       REM **********************
2       REM * PREVOD  CISEL      *
3       REM **********************
10      DIM HEX$(16),H$(1),L$(1),B$(8),B(7),A(8)
20      HEX$="0123456789ABCDEF"
100     ? CHR$(125): POKE 82,0: POKE 752,1
110     ?:?"    PREVODY CISEL    "
120     ?"                       "
130     ?:?" DEC. NA BIN.  >1< "
140     ?:?" DEC. NA HEX.  >2< "
150     ?:?:?:? "BIN NA DEC  >3< "
160     ?:?" BIN NA HEX       >4< "
170     ?:?:?:? "HEX NA DEC >5< "
180     ?:?" HEX NA BIN >6< "
200     OPEN#1,4,0 "K:":GET#1,T: CLOSE#1
210     IF T<49 THEN 200
220     IF T>54 THEN 200
230     GOTO (T-48)*1000
1000    REM ZACATEK PROGRAMU : DEC NA BIN
2000    REM   "          "   : DEC NA HEX
3000    REM   "          "   : BIN NA DEC
4000    REM   "          "   : BIN NA HEX
5000    REM   "          "   : HEX NA DEC
6000    REM   "          "   : HEX NA BIN
```

Jestlize se chcete pocvicit v opisovani, muzete techto 6 malych programu sdruzit do jednoho vetsiho. Na obrazovce se nejprve objevi nabidka sesti moznych prevodu. Zvolenim odpovidajiciho cisla naskoci potrebna cast programu. Je dobre ho pojistit proti chybnym udajum uzivatele (TRAP 1000 = v pripade chyby jdi na radek 1000)

200 zde se otevre cteci kanal(4) ke klavesnici(K=keyboard), ktery odlozi ATASCII-hodnotu nactene klavesy do promenne T a opet uzavre

210-220 prezkouseni, zda je stisknuta klavesa 1-6(ATASCII-hodnoty 49-54)

230 a je-li to tak, pak se zde vypocita odpovidajici cil skoku

2.0 PREHLED O BITECH

U decimalni hodnoty (napr. 191), bez zkusenosti nepozname , ktere bity jsou nebo nejsou nastaveny. Neexistuji take BASICove prikazy, ktere by nam primo daly zpravu o okamzitem stavu urcitych bitu i kdyz nektere prikazy primo pusobi na urcite bity. Mnohdy je ale nutne nahlednout do nektereho bytu a v nem bity nastavit, nebo se na jejich stav zeptat. Prikaz PEEK(n) umi najit pouze decimalni hodnotu, PRINT PEEK(n) vypise na monitoru datovy byte, ktery je na adrese n. Prikaz B=PEEK(n) priradi tuto hodnotu promenne B. Abychom tedy ziskali obraz jedne pametove bunky (byte), mame jen jednu moznost: rozlozit databyte na jednotlive bity. Jak to lze provest nam ukazuje drive uvedeny program DECXBIN. Dale jeste jednou koncentrovana forma:

```
1       REM ***************************
2       REM * ROZKLAD  BYTE NA  BITY *
3       REM ***************************
10      DIMB(7)
20      X=INT(RND(0)*8)
30      R=PEEK(53770):D=R
40      FOR J=0 TO 7:B(J)=0:NEXTJ
50      IF D>127 THEN D=D-128:B(7)=1
60      IF D>63  THEN D=D-64 :B(6)=1
70      IF D>31  THEN D=D-32 :B(5)=1
80      IF D>15  THEN D=D-16 :B(4)=1
90      IF D>7   THEN D=D-8  :B(3)=1
100     IF D>3   THEN D=D-4  :B(2)=1
110     IF D>1   THEN D=D-2  :B(1)=1
120     B(0)=D
130     ?:?" V BYTU " ;R; " JE BIT   ";X; "  =  "; B(X)
140     GOTO 20
```

```
20      urci bit, ktery ma byt vyhledan
30      databyte zde dostane z R prislusnou hodnotu v rozsahu
        0-255.          Muzeme take pouzit R= INT(RND(0)*256)
40-120  vlastni rozlozeni na bity
```

Tento program obsazuje relativne velkou cast pameti. Jestlize ma byt zjisten obsah byte, lze pouzit kratsi formu programu, ktera vsak nemusi byt rychlejsi. Pro pochopeni si ozrejmime pomer hodnot bitu. Hodnota bitu n+1 je prave 2* tak velika jako hodnota n. Jestlize se vyjadruje hodnota bitu vyssiho hodnotou nizsiho, pak je vysledek VZDY sude cislo.
Priklad:

Tento datovy byte by mel byt 128:

cislo bitu	6	5	4	3	2	1	0
vaha(W)	64	32	16	8	4	2	1
kvocient (Q=128/W)	2	4	8	16	32	64	128

Jestlize se datovy byte vyjadruje hodnotou jednoho bitu a vysledek je sude cislo, pak odpovidajici bit NENI nastaven(0). Jestlize dostaneme liche cislo, bit JE nastaven(1).
Priklad:

Tento datovy byte by mel byt 133:

cislo bitu		7	6	5	4	3	2	1	0
vaha		128	64	32	16	8	4	2	1
INT-kvocient	1	2	4	8	16	33	66	133
stav bity		1	0	0	0	0	1	0	1

Take takto se muzeme dozvedet, ze v datovem bytu 138 je ulozena bitova kombinace 10000101.

Urcite jste zpozorovali, ze pri deleni byte bitovou hodnotou nevychazi vzdy cele cislo (integer). Ma-li se zjistit, zda je urcity bit nastaven, jsou nutne nasledujici kroky:

1/ vypocitat hodnotu x-teho bitu (2↑x)
2/ datovy byte R delit touto hodnotou
3/ brat ohled jen na cast s celym cislem = INT(R/2↑x)
4/ urcit, zda je tato hodnota suda nebo licha. Sude cislo je delitelne dvema. Kvocient je tedy cele (integer) cislo. Je-li nejake cislo n delitelne dvema a INT(n/2)=n/2, pak je n sude cislo. Je-li INT(n/2) <> n/2, pak n je liche cislo. Uplny vzorec zni :

```
0       REM   VZOREC 1
10      REM   INT(INT(R/2↑X)/2) <> INT (R/2↑X)/2
```

Jestlize IF tyto podminky splnuje THEN je bit X=1(NASTAVEN) Nyni jeste maly program, ktery takto pracne sestaveny vzorec pouziva:

```
1       REM ******************
2       REM * NASTAVENI BITU *
3       REM ******************
10      B=0
20      X=INT(RND(0)*8)
30      R=PEEK(53770)
40      IF INT(INT(R/2↑X)/2) <> INT((R/2↑X)/2) THEN B=1
50      ?:? " V DATABYTE " ;R; " JE BIT# " ;X;"  =  ";B
60      GOTO 10
```

Prirozene by se dala do radku 40-50 vlozit smycka FOR/NEXT, ktera by mela 8 cislic(0-7); B by pak muselo byt dimenzovano na 8 (DIM B(7)). Tim by se program zkratil a zabral mene pameti, ale je zretelne pomalejsi. Proto je tento vzorec zajimavy pro zjisteni jen jednoho bitu. Jak se s binarnimi cisly pocita, nepotrebujeme v ATARI-BASICu vedet.

PEEK a POKE se uzivaji s decimalnimi cisly. Ma-li se zmenit hodnota urcite adresy v pameti, je nutne nejdrive precist pomoci PEEK(n) aktualni hodnotu v adrese n, tu prictenim nebo odectenim decimalni hodnoty zmenit a pomoci prikazu POKE adresu obsadit novou hodnotou [POKE(n, PEEK(n+/-m)]. Je-li n-m<0 nebo

n+m>255, objevi se chybove hlaseni ERROR-3 (cislicovy rozsah prekrocen).

Nakonec se jeste musime zminit o jedne nutne malickosti. Pro mnohe ucely se pouziva doplnkoveho (komplementarniho) bytu. Tam, kde byla 1, bude 0; kde byla 0, bude 1. Tento proces nazyvame komplementovani. Velmi casto se pouziva v grafice

```
puvodni byte: 1   0   0   1   0   1   1   0        = 150
+ komplemen-
tovany:       0   1   1   0   1   0   0   1        = 105
= soucet:     1   1   1   1   1   1   1   1        = 255
```

Inversni proces pracuje rovnocenne:

```
vsech 8 bitu
nastaveno:    1   1   1   1   1   1   1   1        = 255
- puvodni
  byte:       1   0   0   1   0   1   1   0        = 150

= komplementovany
  byte:       0   1   1   0   1   0   0   1        = 105
```

Abychom nejaky byte komplementovali, potrebujeme jeho decimalni hodnotu odecist od 255 a tak dostaneme decimalni hodnotu komplementovaneho bytu.

POKE n , 255 - PEEK(n) INVERTUJE BYTE NA ADRESE n.

3.0 ROM A RAM

Konstrukce mikroprocesoru urcuje, kolik pametovych mist muzeme pouzivat. Srdcem ATARI je mikroprocesor 6502, ktery muze adresovat 65536 bunek pameti. Pro znazorneni hodnot 0-65535 pouzivame 2 byty. (dva 8-bitove byty). Kazdych 256 bytu se sdruzuje do jedne stranky. Celkem je stranek 256. Horni hodnota bytu (HI) adresuje pametovou stranku. Dolni hodnota bytu (LO) adresuje misto na pametove strance. Tak lze jednim cislem (adresou) usporadat 256*256=65536 pametovych bunek.

Abychom registrovali nejakou urcitou adresu, pouzivame 2 byty, ktere jsou ulozene ZASADNE v poradi LO,HI. Adresa se pak vyhleda podle vzorce:

LO+HI*256.

Jestlize je napr. v adresach 88 a 89 ulozen ukazatel(vektor) na urcitou adresu, pak tuto adresu najdeme nasledujicim prikazem:

ADR = PEEK(88) + PEEK(89)*256

Vedle dvou bytovych vektoru existuji i ukazatele, ktere obsazuji jen 1 byte. Ty ovsem mohou ukazat jen na zacatek stranky. V takovem pripade se adresa najde podle:
ADRESS=PEEK(106)*256
Ctyri stranky, kazda po 256 bytech (neboli 1024 bytu), se oznacuji jako 1 kbyte. Jednotka kilo je zde tedy pouzivana v ponekud jinem vyznamu, (ne 1000, ale 1024).
V mnoha pripadech je treba dat pozor na 4-kilobytovou hranici. Jestlize se do pameti adresuje hex. cislo, pak se pocita nejvyssi cislice tohoto hex. cisla se ctyrmi kbyty. Dohromady tedy pocitac pouziva pres 64 kbytu (65536 bytu) pametovych mist.V zadnem pripade vsak neni uzivateli k dispozici cely tento prostor. Asi polovina vsech adres se pouziva pro provoz pocitace, periferie a ty musi byt stale k dispozici.
Tato data jsou zcasti strukturalne v pameti zakotvena, takze jsou zachovana i ve vypnutem pocitaci, nemohou byt uzivatelem zmenena. Takove pametove oblasti se nazyvaji ROM (read only memory).
Pametova mista, ktera mohou prijimat ruzne udaje uzivatele nazyvame RAM (random access memory).

System pouziva celou skalu RAM adres, aby zachytil prubezne hodnoty (napr. urcite ukazatele). S mnohymi temito registry muze uzivatel manipulovat, napr. sem pomoci POKE ukladat urcite hodnoty. Ovsem mnoho registru system velmi rychle obnovuje a to tak rychle, ze zasah pomoci POKE zustava neucinny.
V nekterych registrech maji jednotlive bity urcite funkce v jinych zase bity na sebe samy berou urcitou ulohu. Nikdy nemusi byt pouzito vsech osm bitu v jedne pametove bunce.
Po zapnuti ATARI 800 XL je uzivateli k dispozici 37 kbytu, 148 stranek nebo presne 37 902 bitu.
Je-li pripojena disketova jednotka, zmensi se tyto hodnoty asi na 31,5 kbytu 126 stranek nebo 32 274 kbytu, protoze odpovidajici misto zabere DOS (diskovy operacni system). Prikazem PRINT FRE(0) muzeme zjistit kolik mista mame k dispozici.

4.0 SITUACNI PLAN PAMETI

d = $ - adresa : funkce

---

0 = $ 0000 nejnizsi adresa (bottom of memory)

0 = $ 0000 OS nulta stranka RAM

128 = $ 0080 BASIC - nulta stranka RAM

256 = $ 0100 zasobnik (stack)

512 = $ 0200 ukazatel (preruseni), paddles, barvove registry

768 = $ 0300 IOCB 0-7, buffer tiskarny

1024 = $ 0400 cazetovy buffer

1652 = $ 0480 OS - RAM

1536 = $ 0600 volne, pro kratke rutiny ve strojovem kodu

1792 = $ 0700 BOOT - RAM pro uzivatele nebo DOS

2048 = $ 0800 DOS - RAM, je-li nahran

11008 = $ 2B00 RAM, konec DOS neni pevny

32768 = $ 8000 Display-list, pamet obrazovky, (pocatecni adresa je dana graf. modem), koncova adresa textoveho okenka je vzdy 40 959 = $9FFF.

40960 = $ A000 BASIC-ROM

49152 = $ C000 OS-ROM

53248 = $ D000 GTIA

53760 = $ D200 POKEY

54016 = $ D300 PIA

54272 = $ D400 ANTIC

54528 = $ D500 interface modulu

55296 = $ D800 floating point ROM (matematicke podprogramy pro praci s cisly s plovouci desetinnou carkou).

57344 = $ E000 ROM se standartni sadou znaku, zacatek OS-ROM (operacni system); dosahuje az na 65535=$FFFF

58368 = $ E400 editor

58384 = $ E410 vektory obrazovky

58400 = $ E420 vektory klavesnice

58416 = $ E430 vektory tiskarny

58432 = $ E440 vektory kazety

58448 = $ E450 JMP vektory (skoky na adresy)

58496 = $ E480 RAM vektory pro powerup

58534 = $ E483 CIO (central input/output)

59093 = $ E605 ridici program preruseni (interrupt)

59716 = $ EC00 SIO (serial input/output)

60906 = $ EDEC ridici program disku

61048 = $ EE78 ridici program tiskarny

61249 = $ EF41 ridici program kazety

61667 = $ F0E3 monitorove rutiny

62436 = $ F3E4 ridici program obrazovky a klavesnice

65535 = $ FFFF nejvyssi adresa (top of memory)

5.0 MAPA PAMETI

0,1 $0, 1 LINZBS

Je pouzivan RESET rutinou pri testovani pameti. Obnovuje monitorovou RAM a pokud je to mozne, pamatuje hodnotu casovace (timeru) pro VBLANK

2,3 $2,$3 CASINI

Vektor pro inicializaci kazetoveho bootu.Kdyz byl tento boot ucinny, nasleduje skok (JSR) na zde udanou adresu. Paty (LO) a sesty (HI) byte prvniho zaznamu na kazete obsahuji inicializacni adresu.

4,5 $4,$5 RAMLO

RAM vektor pro test velikosti pameti pri powerup. Take se pouziva jako adresa pro diskovy boot (1798=$706).

6 $6 TRAMSZ

Docasny registr pro test velikosti pameti. Pouziva se pri powerup a odevzdava svou hodnotu do RAMTOP(106=$6A). Jestlize se zasune modul(cardridge), cte se zde 1.

7 $7 TSTDAT

Datovy registr pro RAM test, prebira hodnotu prave testovaneho mista.

8 $8 WARMST

Indikator tepleho startu(flag). Powerup zde inicializuje nulu(0), ktera zde zustane zachovana, dokud neni stisknuto RESET nebo neni zmenena POKE. RESET zde dosazuje 255(vsechny bity nastaveny)

9 $9 BOOT?

Jeden uspesny diskovy boot zde zanecha 1 (bit 0 je nastaven), jeden uspesny kazetovy boot zde zanecha 2 (bit 1 je nastaven). BOOT? obsahuje nulu, jeslize zadna z obou periferii nebyla "bootovana". Zalezi na tom, zda se pri RESET pouziva vektor kazety (CASINI 2,3=$2$3) nebo DOS vektor(DOSVEC10,11 =AB). Studeny start "zaopatri" oba bootingy a nastavi odpovidajici bit. Je-li v BOOT? ulozeno 255(vsechny bity nastaveny), zalezi na systemu,co se stane jestlize se skiskne RESET. Po provedeni bootu muze byt tlacitko zablokovano (ochrana software)

10,11 $A,$B DOSVEC

Pocatecni vektor pro diskovy software. Prikaz BASICu "DOS" umoznuje skok na tuto adresu. Ukazatel se muze menit, ale muze byt take obnoven pomoci RESET. Aby se tomu predeslo, musi byt zmeneny adresy 5446(LO) a 5450(HI)= $1546 a $154A, ktere obycejne na DOSVEC ukazuji.

Nasledujici program ukazuje, jak se pomoci DOSVEC muze nahradit BASIC- prikaz DOS:

```
0       REM  ADR10
1       REM  ****************************
2       REM  *  DOS STISKNUTIM KLAVESY  *
3       REM  ****************************
10      REM  PROGRAM
20      REM  PROGRAM
60      IF PEEK(764)=255 THEN 10
70      OPEN#1,4,0,"K": GET#1,D: CLOSE#1
80      IF D=68 THEN 100
90      GOTO 10
100     LET DOS=USR(PEEK(10)+PEEK(11)*256)
```

10-20 symbolizuje nejaky program
60 kontroluje stisknuti libovolne klavesy
70 jestlize ano, precte se otevrenym kanalem z klavesnice ATASCII
hodnota teto klavesy do promenne D
80 je-li D=68, pak stisknutim klavesy "D" program pokracuje na radce 100
90 zpetny skok, pokud nebyla stisknuta "D"

100 BASIC prikaz (DOS) je zde pouzit jako promenna. Musime na to vyrazne poukazat pomoci LET..

Prikaz USR skoci do programu ve strojovem kodu, ktery zacina v pametove bunce uvedene za USR. Je-li zde jako pocatecni adresa nasazen DOSVEC, skoci USR do DOS. Prikaz USR musi byt veden jako promenna (zde DOS), ale dale nema zadnou funkci.

12,13 $C,$D DOSINI

Inicializacni vektor pro disk-boot. "Prevezme" adresu, ktera po nahrani DOS-u zahaji program uzivatele. Zde muze byt spusten ucastnicky software pomoci neprimeho JSR.

14,15 $E,$F APPMHI

Vektor prvnich volnych pametovych mist v uzivatelske RAM ukazuje, az kam je pamet obsazena BASIC programem. Od adresy, na kterou ukazuje tento vektor, az po RAMTOP(40959=$9FFF) pak mohou byt ulozena obrazovkova pamet a dsplay list.

16 $10 POKMSK

Interruptova maska pro obvod POKEY. Stinovy registr IRQEN(53774=$D20E). Kazdy z osmi bitu se vztahuje k jednomu moznemu preruseni(interrupt).Nastaveni urciteho bitu zpusobi, ze odpovidajici interrupt je mozny:

bit 7 (128) klavesa BREAK
bit 6 (64) ostatni klavesy
bit 5 (32) seriove vstupy dat pripraveny
bit 4 (16) seriove vystupy dat
bit 3 (8) seriovy vystup ukoncen
bit 2 (4) POKEY timer 4
bit 1 (2) POKEY timer 2
bit 0 (1) POKEY timer 1

Hodnoty je nutne zavest do IRQEN(53774=$D20E) pomoci POKE. Kazda hodnota <128 blokuje klavesu BREAK. Se 127-mi se sice BREAK program zastavuje, ale zaroven se stava zavislym na systemu. RESET je pak jedinou moznosti. A jak je tlacitko RESET zajisteno proti nepovolanym se ukaze pri BOOT?. Vyzkousejte si tu nekolik hodnot, abyste to poznali. V kazdem pripade nastavi kazdy prikaz, ktery probehne na obrazovce (screen"S:" nebo editor"E:") bitu 7 jednicku a tim opet uvolni interrupt pro BREAK. Po kazdem prikazu PRINT, GRAPHICS nebo OPEN ("S:" nebo "E:") se musi klavesa BREAK znova zajistit. To se provede nejlepe timto programem:

```
0       REM  ADR16BRK.
1       REM  ************************************
2       REM  * PODPROGRAM PRO ZAJISTENI BREAK *
3       REM  ************************************
```

```
10      GOSUB 5000
5000    BRK=PEEK(16)
5010    IF BRK>127 THEN BRK=BRK-128
5020    POKE 16,BRK
5030    POKE 53774,BRK
5040    RETURN
```

ROM nove "B" verze OS ma zvlastni vektor pro BREAK-interrupt. K tomu viz BRKKEY(566,567=$236,237)

17 $11 BRKKEY

Indikator klavesy BREAK. Je-li BREAK stisknut je zde 0. Kazda jina hodnota znamena, ze stisknut neni.

18,19,20 $12,$13,$14 RTLOK

Vnitrni hodiny ATARI. Tikaji frekvenci podle dodaneho napeti. Zatimco "rytmus americkych aparatu je 60Hz, zde prodavane stroje pouzivaji 50Hz. Jestlize pracujete s US softwarem, pri kterem je treba cist z vnitrnich hodin, musi se hodnoty prizpusobit faktorem 5/6.
Pri zapnuti pocitace zacnou vnitrni hodiny bezet od hodnoty 0. Registr 20 pocita v 50-ti Hz taktu od 0-255, tedy v jedne sekunde od 0-49. Jakmile se dosahne teto hodnoty, zvysi registr 20 hodnotu registru 19 o jednicku a zacne opet u nuly. Ve chvili, kdy registr 19 prekroci hodnotu 255, zvysi se zase hodnota registru 18. Tyto tri registry mohou dohromady pocitat az do 16 777 215.

Takto muze vypadat program na hodiny:

```
0       REM ADR18
1       REM  ******************
2       REM  * VNITRNI HODINY *
3       REM  ******************
10      FOR J=0 TO 2: POKE 18+J,0: NEXT J
20      ? CHR$(125): POKE 752,1
30      T= INT((PEEK(18)*65535+PEEK(19)*256+PEEK(20))/50)
40      D= INT(T/86400)
50      HH= INT(T/3600):H=HH-D*24
60      MM= INT(T/60): M=MM-D*1440-H*60
70      S=T-D*86400-H*3600-M*60
80      POSITION 10,11
90      ? D; "d :"; H; "h:"; M; "' :"; S; CHR$(34)
100     GOTO 30
```

10- nastaveni vnitrnich hodin na nulu
20- smaze obrazovku a potlaci kurzor
30- vypocita okamzitou hodnotu registru 18-20, deli padesati a dosadi celou cast kvocientu do T
40- sleduje jak hodnota T odpovida celym dnum a ulozi vysledek do do promenne D
50- urci stejnym zpusobem cele minuty z T. HH zahrnuje

vysledek. Jestlize od HH odecteme pomoci vypocitanych dnu "nepokryte hodiny" (D*24), zustanou nam bezici hodiny. H pak prevezme tento udaj.

60- analogicky vypocet pro bezici minuty

70- bezici sekundy S dostaneme, jestlize od celeho souctu sekund T odecteme sekundove ekvivalenty z D, H, M.

80, 90- vytisteni

100- novy beh programu

Misto toho, abyste dny, hodiny, min.a sek. vypocitavali a pak cekali, az bude registr plny a preskoci, muzete nechat hodiny bezet 24hod, aby pocitaly pak RTCLOK nastavit na nulu a zapocitat jako promennou 1 cely den. Tak mohou hodiny bezet nekonecne. 24 hodin je plnych kdyz RTCLOK ma hodnotu 24*60*60*50=4*320 000=65536+235+235+0. Tento prikaz tedy zni:
IF PEEK(20)=65 THEN IF PEEK(19)=235 THEN IF PEEK(18)=0 THEN POKE 18,0 : POKE 19,0
Take program na vnitrni hodiny potrebuje urcity zpracovatelsky cas, behem nehoz vnitrni hodiny bezi dal. Pri kratkem programu, jako je vyse uvedeny, je pocitac dostatecne rychly a zobrazi kazdou sekundu, i kdyz je registr 18-20 stale znovu cten, aby se mohlo zpracovat 30 radek programu. Pro ATARI je sekunda velmi dlouha doba. Behem ni vyridi asi 40 000 strojovych cyklu, tedy behem toho, co registr 20 pocita o jednu dal a na to potrebuje 1/50s, pocitac absolvuje 8000 taktu. Tento program lze vestavet do jineho, pak muzete dostat urcite potrebne vypocty (funkce uhlu, mocniny..) drive, nez se sekundy vymeni. Vnitrni hodiny nemusi byt bezpodminecne prepocitany do realneho casu. Muzete se ptat jen na registr 19 (desetiny sek.) a nalezenou hodnotu dosadit do pomalu rostouci graficke krivky. Nebo lze zadat hodnotu po jejimz nasazeni se neco stane:
IF PEEK(18)*65536 + PEEK(19)*256 + PEEK(20) THEN IF PEEK(18)*65536 + PEEK(19)*256 + PEEK(20)= X ...THEN

21,22 $15,$16 BUFADR

Momentalni ukazatel SIO pri disketovych operacich. Registr pro neprime buffer-adresy

23 $17 ICCOMT

Prikaz pro vektor CIO. Pouziva se k nalezeni relativni adresy (offset) v tabulce prikazu.

24,25 $18$19 DSKUTL

Vektor pro FMS (file management system - cast DOSu)

26,27 $1A,$1B DSKUTL

Vektor pro DUP (disk utilities package)

28 $1C PTIMOT

Casovy prodleva (timeout) tiskarny

29 $1D PBPNT

Ukazatel buffru tiskarny. Udava probihajici pozici(byte) v buffru. Hodnoty od 0 do hodnoty PBUFSZ

30 $1E PBUFSZ

Velikost buffru tiskarny v okamzitem modu. Normalni velikost buffru je 40 bytu.

31 $1F PTEMP

Pouziva se s manipulacnim programem tiskarny, aby se prednostne pamatovala hodnota znaku, ktery ma byt vytisten.

32 $20 ICHIDZ

Index manipulacniho programu.Je nastaven operacnim systemem jako index v tabulce nazvu zarizeni pro momentalne otevreny soubor. Neni-li zadny soubor otevren, je zde 255.

33 $21 ICDNOZ

Maximalni pocet disketovych jednotek. Zacina 1.

34 $22 ICCOMZ

Uzivatelem dany prikazovy byte, ktery urcuje jak bude formatovan zbytek IOCB a ktery I/O krok ma byt proveden.

35 $23 ICSTAZ

Status byte pro naposledy zminenou periferii

36,37 $24,$25 ICBALZ/HZ

Adresa buffru pro prenos dat nebo adresa prijmu dat pro prikazy jako STATUS, OPEN..

38,39 $26,$27 ICPTLZ/HZ

Adresa rutiny "put-byte". Prikaz CLOSE nastavuje tento ukazatel na CIO "IOCB not OPEN"

40,41 $28,$29 ICBLLZ/HZ

Citac pro delku buffru pri PUT a GET. Pri kazdem prenesenem

bytu zmensen o jednicku.

42 $2A ICX1Z

Prvni byte pomocne informace, ktera pro OPEN specifikuje typ pristupu k souboru.

43 $2B ICAX2Z

Druhy byte pomocne informace, CIO pracovni promenne

44, 45 $2C, $2D ICAX3Z/4Z

Pouziva se pri prikazech BASICu "NOTE" a "POINT", aby se prenesla cisla sektoru.

46 $2E ICAX5Z

Urcuje byte v sektoru prvne stanovenem ICAX3Z/4Z, ktery ma byt dosazen.

47 $2F ICAX6Z

Volny byte. Pouziva se jako vhodne misto pro odlozeni znakoveho bytu v probihajici operaci PUT.

48 $30 STATUS

Lokace pro vnitrni status. SIO rutiny pouzivaji tento byte k uschove statusu probihajici SIO operace.

49 $31 CHKSUM

Zkusebni (kontrolni) soucet; pouziva se pri SIO

50, 51 $32, $33 BUFRLO

Ukazatel v datovem buffru. Urcuje byte, ktery ma byt vyslan nebo prijat.

52, 53 $34$35 BFENLO/HI

Ukazatel nejblizsiho bytu za koncem data-buffru

54 $36 CRETRY

Udava, kolikrat bude operacni system opakovat povel, pokud se uspesne nevykonal (napr. formatovat disketu, popsat sektor..). Default hodnota je 13.

55 $37 DRETRY

Pocet opakovani, jestlize pristroj (napr. disk) uspesne nevykonal. Default hodnota chyby je 1.

56 $38 BUFRFL

Stav datoveho buffru, 255 znamena plno.

57 $39 RECVDN

Prijem dat. 255 znamena prijato.

58 $3A XMTDON

Prenos dat, 255 znamena prijato.

59 $3B CHKSNT

Kontrolni soucty. 255 znamena zaslano.

60 $3C NOCKSM

Udaj o nepritomnosti kontrolniho souctu. 0 znamena, ze prenosu dat nasleduje kontrolni soucet. Jakakoliv jina hodnota udava, ze zadny kontrolni soucet nenasleduje.

61 $3D BPTR

Ukazatel kazetoveho bufru, index prave zpracovavanych (psanych nebo ctenych) dat v datasouboru.

62 $3E FTYPE

Urcuje odstup mezi data bloky

63 $3F FEOF

End-of-file (EOF). Flag pro kazetu. Nenulovy udaj znamena, ze je zjisten konec souboru.

64 $40 FREQ

Pocitadlo "pipnuti" pro operaci OPEN kazety: 1*pri cteni, 2*pri zapisu.

65 $41 SOUNDR

Indikator sumoveho signalu pri prenosu dat. Nula v tomto

registru zpusobi bezsumovy provoz. Jina hodnota umoznuje akustickou zpetnou vazbau. Napr. u pasku se synchronni zvukovou a datovou stopou muze byt datova stopa zatlumena.

66 $42 CRITIC

Jina hodnota nez 0 vyradi cast OS. Muze to byt nutne, kdyz se od interruptu ocekava vysoka frekvence. Vedlejsi efekt spociva v tom, ze je znemozneno opakovaci funkce (repeat). Pak neni mozne stisknout tutez klavesu opakovane. Dale se meni ton bzucaku (CONTROL a 2). Nedoporucuje se nasadit CRITIC na nenulovou hodnotu.

67-73 $43,$49 FMZSPG

Registry nulte stranky diskoveho FMS. Sedm bytu.

74 $4A CKEY

Flag pro vyzadany kazetovy boot pri studenem startu. Zkousi, je-li pri zapnuti stisknuta klavesa START. Jestlize ano, nastavi se CKEY. Tim je umoznen auto boot z kazety.

75 $4B CASSBT

Indikator pro kazetovy boot. Pri zapnuti(powerup) se OS pokusi bootovat kazetu i disketu. Pokud kazetovy boot neni mozny, ulozi se zde 0. Blize viz BOOT?(9=$9)

76 $4C DSTAT

Registr pro status obrazovky a klavesnice. Je pouzivan monitorem, ukaze se zde chybna pozice kurzoru, malo mista v pameti pro graficky mod nebo stav preruseni pomoci BREAK.

77 $4D ATRACT

Kdyz TV obraz dlouho stoji, muze se vypalit aktivni vrstva obrazovky. Konstrukteri ATARI doplnili ATRACT mod tak, aby se toto nemohlo stat. Tedy, bude-li obraz dlouho stat, zmeni v kratkych intervalech tento mod hodnoty v barvovych registrech a redukuje tak celkovy jas obrazovky. ATRACT funguje jako indikator a casovac attract-modu. Jakmile se stiskne libovolna klavesa, registr se vzdy z IRQ nastavi na 0. Vyjimkou jsou klavesy SELECT, OPTION, START. Pokud neni stisknuta jakakoliv klavesa, kazdych 5 sekund (taktovano VBLANK) se zvysi hodnota v tomto registru. Ve chvili kdy prekroci hodnota ATRACT 127, preklopi timer(bit0-6) a nastavi flag(bit7). Decimalni hodnota registru jde pak az na 254 a nastavi se ATRACT mod.

Nasledujici program ukazuje zpusob prace tohoto registru. Zaroven se pouzivaji vnitrni hodiny(RTCLOK), aby se stopovala

doba ATRACT modu:

```
0       REM **********************
1       REM * OCHRANA OBRAZOVKY  *
2       REM **********************
10      DIM B(6)
20      POKE 19,0: POKE 20,0
30      ?CHR$(125): POKE 752,1
40      T=INT(PEEK(19)*256 + PEEK(20)/50)
50      X=PEEK(77)
60      IF X>127 THEN R=10: POSITION 3,6: ? "STOPPED"
70      FOR J=0 TO 6: B(J)=0: NEXT J
80      M=INT(T/60)
90      S=T-M*60
100     Y=X
110     IF X>127 THEN X=X-128: B(6)=1
120     IF X>63  THEN X=X-64 : B(5)=1
130     IF X>31  THEN X=X-32 : B(4)=1
140     IF X>15  THEN X=X-16 : B(3)=1
150     IF X>7   THEN X=X-8  : B(2)=1
160     IF X>3   THEN X=X-4  : B(1)=1
170     IF X>1   THEN X=X-2  : B(0)=1
200     POSITION 2,2: ?
 CHR$(17);CHR$(18);CHR$(18);CHR$(18);CHR$(18)
        CHR$(18);CHR$(18);CHR$(18);CHR$(5)
210     POSITION 2,3: ? CHR$(124);  "       ";CHR$(124)
220     POSITION 4,3: ? M;"  "
230     POSITION 6,3: ? " : "
240     POSITION 7,3: ? S;"  "
250     POSITION 2,4: ?
 CHR$(26);CHR$(18);CHR$(18);CHR$(18);CHR$(18)
        CHR$(18);CHR$(18);CHR$(18)CHR$(3)
260     POSITION 0,16: ? "                                          "
270     POSITION 3,20:
280     ? " ";B(6);" ";B(5);" ";B(4);" ";B(3);" ";B(2);" ";B(1);
        " ";B(0);" "; X ; "  =  "
290     POSITION 34,20: ? Y;  "  "
300     GOTO 40+R
```

10- B$ ma prebrat bity z ATRACTu. String je dimenzovan na 6, aby se usetrilo misto. Proc to staci se dozvite dale.

20- pro tento ucel staci z RTCLOK registry 19, 20. Bezi spolecne asi 20 min. Zde se nastavi na nulu.

40- spolecny celkovy cas T(sek.) se vypocita znamym zpusobem.

50- do X se zapise hodnota registru 77

60- jestlize X prekroci kritickou hodnotu 127, stopnou se vnitrni hodiny. To proto, ze promenna R byla nastavena na 10, takze zpetny krok na konci programu uz nevyjde na radek 40(aby si precetl RTCLOK), ale na radek 50. Zaroven se pod stopkami objevi napis "STOPPED"

70- sest prvku indexovane promenne B se nastavi na nulu

80,90- vypocita cas v min. a sek.

100- hodnota X se priradi do Y

110-170 Z bytu X se vypocitaji bity 7-0 a ulozi se v B(J). Protoze se odpovidajici hodnoty vzdy odectou od X,

odpovida X podle radku 170 stavu bitu 0. To vysvetluje radek 10. Promenna X nesmi byt pro kazdy prubeh znovu nastavena na 0, protoze X se vzdy plni hodnotou registru 77

200-260 obrazova grafika-PRINT. Protoze jehlove pero nebo jina pseudograficka malba nema specialni prizpusobeni softwaru, musi se pro prislusne graf. znaky pouzivat CHR$.

220-240 ve vyznacenem ramecku se pise bezici cas v min a sek.

280,290 vyda aktualni byte v ATRACTu a odpovidajici dec. hodnotu

300 nekonecna smycka

Pokud se v prubehu programu stiskne klavesa, vrati se sice hodnota registru 77, ale stopky bezi dal. Dotazem na CH(764=$2FC) lze zare gistrovat stisk klavesy a zpetnym skokem na radek 20 se stopky opet nastavi na 0. Jak dlouho je nutne pouzivat ATRACT, aby se atrtact mod vypnul, muzete sami timto programem urcit. Hodnota je zretelne vetsi, nez se v ruznych publikacich udava. Jestlize pisete herni nebo graficky program, ktery casovou hranici prekroci, aniz by nasledovalo zadani klavesou, meli byste zamezit tomu, aby attract mod menil barvy. K tomu staci POKE 77,0.

78 $4E DRKMSK

Zde je 254 do te doby, nez je attract-mod aktivni a pak se nastavi 246, coz znemozni, aby barevna svetlost prekrocila 50%

79 $4F COLRSH

Maska pro zmenu barev. Barvove registry (od 708=$264) se spoji s adresami 78,79 pomoci EOR-operace(exkluziv OR)

80 $50 TMPCHR

Docasny registr, ktery pouziva ridici program obrazovky, aby vyslal data na obrazovku, nebo je odtud precetl.

81 $51 HOLD1

Jako TMPCHR. Pouziva se take, aby "uchoval" pocet odkazu na display list

82 $52 LMARGN

Hodnota sloupce leveho okraje pri vystupu na obrazovku. Nula odpovida levemu okraji obrazovky.

83 $53 RMARGN

Hodnota sloupce praveho okraje obrazovky. 39 je pravy okraj, coz je take default hodnota. Oba registry mohou byt (s urcitym

omezenim) nastaveny na libovolne hodnoty. Tak to bez dalsiho dela obrazovkovy editor, jestlize je pomoci POKE za 82,83 ulozena stejna hodnota.

Nahrajte do pameti libovolny program, zadejte POKE 83,n a POKE 83,n (n=0-39) a dejte LIST. Co se stane?
Pravy okraj stoji vlevo od leveho okraje; napr. POKE 82,37 a POKE 83,47. Jestlize nyni nastavite levy okraj na hodnotu >39, jeste radi stisknete RESET.(obnovite default-hodnoty). Mazani nebo vkladani radku neni ovlivneno zmenou vystupu. Insert-line vlozi urcity fyzicky radek stejne delky a delete-line maze vzdy logicky radek. Take akusticke znameni pro konec logickeho radku je ovlivneno temito registry. Zrusi se na konci programoveho radku, ktery je zkracen pomoci kratsiho fyzickeho radku. Bzucak tedy zazni vzdy pred koncem treti radky.

84 $54 ROWCRS

Okamzita poloha grafickeho nebo textoveho kurzoru v radku. Obsah teto adresy odpovida radku obrazovky od kterehо(na kterem) bude psan nebo cten nasledujici znak. Minimalni hodnota PEEK (84)=0. Podle pouziteho grafickeho modu jsou pripustne hodnoty od 0(nahore) do 19(dole)

85,86 $55$56 COLORS

Okamzita sloupcova pozice grafickeho nebo textoveho kurzoru. Podle pouziteho grafickeho modu jsou mozne hodnoty od 0(vlevo) do 39 nebo 319(vpravo).

V nasledujicim programu nastavuje prikaz POSITION kurzor na ruzna mista obrazovky a vytiskne jeho okamzitou polohu:

```
0       REM ADR84
1       REM  ************************
2       REM  * DEMO- POLOHA KURZORU *
3       REM  ************************
10      ? CHR$(125): POKE 82,0
20      FOR X=0 TO 30 STEP 6
30      FOR Y=0 TO 22 STEP 2
40      POSITION X,Y: ? PEEK(85); ","; PEEK(84);
50      NEXT Y
60      NEXT X
70      GOTO 70
```

40- adresa 86 obsahuje HI-byte sloupcove polohy kurzoru. Je vzdy 0, mimo GR.8, kde ma hodnotu 1. Proto se zde staci zeptat na 85. Obsahy registru jsou vzdy vytisteny v neprevracenem tvaru, protoze vsechny prikazy, ktere se vztahuji na pozici na obrazovce, davaji hodnoty v poradi sloupec-radek, tedy : PLOT sloupec, radek nebo POSITION x,y. Protoze registr 85 obsahuje hodnotu sloupce, vytiskne se pred carkou

70- zabrani ukonceni programu pomoci READY.

Je nesmyslne zadat nasledujici povel: POSITION n,m: ?PEEK(85); " "; PEEK(84).

Tento prikaz nenajde hodnoty n,m, protoze, po zadani RETURN, skoci kurzor na zacatek pristiho fyzickeho radku a stejnou pozici najde take prikaz BASICu LOCATE, protoze prikaz PRINT pohne kurzorem a zmeni hodnotu v ROWCRS a COLORS.

Ostatne CHR$(253) nebo signalni ton(BELL) take nejsou zadne tisknutelne znaky a nepusobi proto na posici kurzoru.
Protoze prikaz LOCATE ma sve zvlastnosti, pripojujeme program:

```
0       REM  ADR85
1       REM  ************************
2       REM  * VLIV LOCATE NA KURZOR *
3       REM  ************************
10      GRAPHICS 0: ?  CHR$(125)
20      POSITION 30,20
30      ? PEEK(85); " , " ; PEEK(84)
40      LOCATE 4,4,D
50      ? PEEK(58); " , " ; PEEK(84),
60      LOCATE 0,0,X
70      POSITION 10,10 : ? "A"
80      LOCATE 10,10,D : ? D
90      FOR Z=0 TO 1000: NEXT Z
100     LOCATE 10,10,D : ? D;"  "
110     FOR Z=0 TO 1000: NEXT Z
120     LOCATE 10,10,D : ? D;"  "
200     GOTO 90
```

10- ackoli GR.0 nemusi byt zvlast volan, je zde nutny protoze LOCATE pusobi jen tehdy, kdyz mu predchazi prikaz GR..

20- naskoci poloha 20 a 30. Tim se v kurzorovych registrech nachazeji odpovidajici hodnoty, ktere jsou cteny a vydany na obrazovku v radku 30

30- dejte pozor na signalni ton (BELL) na konci prikazu. Brani tomu, aby kurzor po vytisteni skocil na zacatek nasledujiciho radku a drzi ho za vyrazem.

40- protoze, ale v radku 40 probehne LOCATE, zustane opticky kurzor za pozici 30,20, ackoli ve skutecnosti se nachazi na miste 4,4. LOCATE najde hodnotu COLORu(graf. bod) nebo hodnotu ATASCII a zada ji do stanovenych promennych.

50- opet vytisten kurzor. registr. Carka na konci povelu ovlivnuje skok tabelatoru.

60- povelem LOCATE, bude opticky kurzor zpet

70- na miste 10,10 se vytiskne A

80- LOCATE najde na 10,10 ATASCII A = 65 a hned ji vytiskne. PRINT za LOCATE zmeni hodnotu na lokalizovanem miste

90- kratka pauza

100- LOCATE uz na miste 10,10 nenajde 65(A), ale 32 = prazdny znak

110- kratka pauza

120- hodnota se opet v tomto miste zmeni pomoci LOCATE a vytiskne. Vidime prazdny invertovany znak a najde se hodnota 160

200- pokracovani az do BREAK

Trik LOCATE-PRINT pracuje jen v GR.0. AT-BASIC-Reference Cards tvrdi, ze podobne pracuje GR.9,10,11. Uvedeny priklad nedava vzdy vysledky, protoze kdyz je zapojen GR. bez textoveho okenka, PRINT prepne okamzite na GR.0.

87 $57 DINDEX

Obsahuje v dolnich 4 bitech rozlisovaci cislo aktivizovaneho graf. modu. Protoze se OS vztahuje v tomto registru na vice grafickych funkci, lze to vyuzit k vymene na zcela jiny mod nez ten, co byl vyvolan pomoci GR.. Registr muze nabyvat hodnotu 0-15, ktere odpovidaji GR.-prikazum. (nikoliv ANTIC-viz display-list)
Nasledujici program ukaze, jak lze pomoci DINDEX simulovat novy GR. mod. Vytvari graficka reseni o 160(0-159)*192(0-191) obrazovych bodech. Kazdy pixel je pak siroky jako dva obrazove body a kazdy radek modu je vysoky jako jedna TV radka. Barvy budou definovany podle GR.8 (popr.24), ale rozdil je jen ve svetlosti:

```
0       REM ADR87
1       REM  ****************
2       REM  *  GR. 7  1/2  *
3       REM  ****************
10      GR.24
20      POKE 87,7
30      COLOR 3
40      FOR X=0 TO 159: PLOT X,0: NEXT X
50      COLOR 2
60      FOR Y=0 TO 95: PLOT 159,Y: NEXT Y
70      COLOR 3
80      PLOT 0,0: DRAWTO 159,95
90      COLOR 2
100     PLOT 0,95: DRAWTO 159,0
110     COLOR 1
120     FOR X=0 TO 159: PLOT X,95: NEXT X
200     POKE 89,PEEK(89)+15
210     COLOR 3
220     FOR X=0 TO 159: PLOT X,0: NEXT X
230     COLOR 2
240     FOR Y=0 TO 95: PLOT 0,Y:PLOT 159,Y: NEXT Y
250     COLOR 1
260     PLOT 0,0: DRAWTO 79,95: DRAWTO 159,0
270     PLOT 0,95: DRAWTO 79,0: DRAWTO 159,95
280     COLOR 2
290     FOR X=0 TO 159: PLOT X,95: NEXT X
300     GOTO 300
```

```
10-     graf.  mod bez text okenka
20-     v DINDEX je ulozena hodnota 7
30-120  kresli ruzne linie v barvach.
```

Misto FOR-NEXT-PLOT lze pouzit PLOT-DRAWTO. Vyvolany GR.8 dava k dispozici 320 PPL (pixel per line=obrazove body pro jeden radek), ovsem sedm ulozenych do 87 pomoci POKE naplni radek jiz [illegible] obrazovych bodu. To jsou PPL GR.7. Protoze

pro kazdy pixel jsou zde k dispozici dva bity barevne informace, muzeme vyzadovat tri barevne hodnoty(viz obrazova pamet). Prikazem GR. je vyvolan display-list. Pri GR.8 je kazdy radek modu vysoky jako TV radek. Obrazovka ma 191 modus- radku. Je-li na obrazovku vyvolana pomoci PLOT nebo DRAWTO pozice ktera presahuje hodnotu GR.7, nasleduje ERROR-141, protoze se zde uplatnuje ulozena hodnota v DINDEX pomoci POKE. S hodnotami v GR.7 tady muze byt zpracovana jen horni polovina obrazovky.

Pomoci GR.8 se v pameti rezervuje dostatek mista pro prijem obrazovych dat pro cele stinitko. Pri zminenych manipulacich nelze popsat dolni polovinu graficky pusobicimi BASIC prikazy. Vsechny prikazy se vypocitavaji v offsetu SM-bytu, ktery obsahuje data hledaneho bodu a to podle vzorce :
pocatecni adresaSM + hodnota sloupce/PPB + hodnota radku * BPL(byte per line). Se souradnicemi GR.7 je tedy mozny offset 40+95*40

200- abychom dostali obrazova data na dolni polovinu, musime zvysit vektor SM-startadresy o vyse spocitanych 3840 bytu
210-290 zbyle hodnoty pozic a pixelu na dolni polovine se "odPLOTuji"
300- brani ukonceni programu. Protoze neni k dispozici textove okenko, nastavi se GR.0; tedy nastavena GR. se smaze, jakmile se da READY

Jestlize chcete videt moznosti grafickych modu, zmente ADR87 v nasledujicich radkach:

```
0       REM ADR87A
1       REM ************
2       REM * CHUTOVKA *
3       REM ************
10      GR.24: A=2
15      TRAP 15: A=A+1
20      POKE 87,A
300     GOTO 15
```

88,89 $58,$59 SAVMSC

Ukazatel start adresy obrazove pameti(SM= screen memory). V tomto bytu lezi data pro pixely, ktere budou zobrazeny v levem hornim rohu obrazovky. Graficka data obsazuji pamet od SAVMSC po RAMTOP (40959=$C000). Vyvolanim GR. prikazu se ukazatel nastavi na odpovidajici pametove misto, ktere je k dispozici. Ukazatel muze byt v pripade potreby uzivatelem zmenen. Je mozne ulozit do pameti data nekolika obrazovkovych obsahu a pak presouvanim SM-ukazatele vyvolavat a menit start adresy jednotlivych obsahu. To sice jde jen po kazdem VBLANK, kdyz je display-list cten opet zepredu, a to 50-krat za sekundu.
V nasledujicim programu byl zvolen GR.3 (popr.19), protoze tento mod ma tu vlastnost, ze SM je schopen ovladat jen 240 bytu, tedy mene nez jednu stranku. Tak lze pro obrazovku pohodlne ulozit do pod sebou lezicich pametovych stranek nejaka data a zmenou HI bytu nebo SM ukazatele menit grafiku na monitoru skokove. Ostatne zde nema cenu menit SAVMSC, protoze SM ukazatel je take soucast display-list a odkud z pameti si ma videoprocesor

vyzvednout data urcuje tento ukazatel v DL:

```
1       REM  ***************************
2       REM  * SKOKOVA ZMENA GRAFIKY  *
3       REM  ***************************
10      D=1
20      GR.19
30      POKE 708,50
40      POKE 709,52
50      POKE 710,54
60      FOR P=0 TO 20
70      COLOR C+1
80      X=INT(RND(0)*12)
90      Y=INT(RND(0)*12)
100     PLOT 19-X,11-Y
110     PLOT 19+X,11-Y
120     PLOT 19+X,11+Y
130     PLOT 19-X,11+Y
140     PLOT 19-Y,11-X
150     PLOT 19+Y,11-X
160     PLOT 19+Y,11+X
170     PLOT 19-Y,11+X
180     NEXT P
200     D=D+1:C=C+1:IF C=3 THEN C=0
210     IF D=7 THEN 300
220     SM=PEEK(88)+PEEK(89)*256
230     FOR K=0 TO 239
240     POKE SM-D*256+K,PEEK(SM+K)
250     NEXT K
260     GOTO 20
300     DL=PEEK(560)+PEEK(561)*256
310     Z=PEEK(DL+5)
320     FOR J=0 TO 6
330     POKE DL+5,Z-J
340     FOR W=0 TO 50:NEXT W
350     NEXT J
360     FOR J=5 TO 2 STEP -1
370     POKE DL+5,Z-J
380     FOR W=0 TO 50:NEXT W
390     NEXT J
400     GOTO 320
```

30-50 barvove hodnoty pro prikaz COLOR jsou ulozeny do barvovych registru(COLOR 0, 708=$2C4)

60-180 nekolik nahodnych grafik v barvach

220-250 postupne jsou cteny datove byty obrazovkove pameti(PEEK SM+K) ve stejnem poradi (POKE SM-D*256+K). Pritom D urcuje o kolik stranek vyse se ted maji data ulozit (POKE) a K pocita 240 bytu GR.3.

300- vypocita start adresu DL (SDL STL, 560,561 = $230, $231)

310- HI-byte SM ukazatele v DL je paty byte (nulty se pocita take)

320-390 SM vektor se presouva vzdy po kratke cekaci smycce. Na obrazovce se tak strida skokem pet grafik vytvorenych v horni casti programu

90 $5A OLDROW

Predchozi radek grafickeho kurzoru. Obnovuje se ROWCRS(84=$54) pred tim, nez ten prijme novou hodnotu. Tak jsou pro DRAWTO nebo FILL (XIO 18) k dispozici:
1/ v OLDROW pocatecni souradnice
2/ v ROWCRS konecne(cilove) souradnice

91,92 $5B,$5C OLDCOL

Analogicka funkce k OLDROW pouze pro sloupcovou pozici a proto dva byty velika (320 sloupcu v GR. 8)

93 $5D OLDCHR

Obsahuje hodnotu znaku pod kurzorem a pouziva se k obnoveni znaku, kdyz je kurzor v pohybu.

94,95 $5E,$5F OLDADR

Obsahuje adresu obrazove pameti prave probihajici kurzorove pozice. Pouziva se spolecne s OLDCHR, aby se znak obnovil, je-li kurzor dale v pohybu.

96 $60 NEWROW

Radkova souradnice bodu, ke kterym ma dosahnout DRAWTO nebo FILL (XIO 18)

97,98 $61,$62 NEWCOL

Sloupcova souradnice bodu, ke kteremu ma byt DRAWTO nebo FILL(XIO 18). NEWROW a NEWCOL nabyvaji svych hodnot z ROWCRS a COLCLR (84-86=$54-$56), aby tyto behem rutiny DRAWTO (FILL) mohly prijmout jiz dalsi kurzorove souradnice.

99 $63 LOGCOL

Udava pozici kurzoru v logickem radku, ktery muze byt u ATARI dlouhy v GR.0 skoro tri radky tj. 3*40=120 znaku. LOGCOL pak muze mit hodnotu 0-119. S timto registrem pracuje ridici program displeje, aby napr. vydal signalni ton "radek plny".

100-105 $64-$69 //////

Misto, ktere pouziva ridici program displeje k zapamatovani ruznych hodnot.

106 $6A RAMTOP

Ukazuje ve strankach (=265 bytu) uzivatelskou RAM, ktera je k

dispozici. PEEK(106)*256 zprostredkuje nejvyssi volnou adresu. To se muze pouzit k vytvoreni ochranneho pametoveho rozsahu, napr. pro strojove rutiny, volne definovane rady znaku, player-missile data nebo alternativni obrazova pamet. Nasledujici prikaz zajistuje zadany rozsah:

POKE(106),PEEK(106)-n

n- je pocet pametovych stranek, ktere se maji rezervovat. Hodnota PEEK(106)*256 NESMI byt MENSI nez PEEK(144) + PEEK(145) * 256(MEMTOP).
Je-li pro rezervovana mista v pameti zmenen RAMTOP, mel by hned nasledovat GR. povel. Pak muze byt klidne vyvolan graf. mod, ktery je prave zapnut. Tim spolupracuje DL a graficka data soubezne s novym RAMTOP.

Prikaz GR. nebo CLEAR maze hornich 64 bytu od RAMTOP. Rolovanim textoveho okenka v graf. modu se prepise 800 bytu v horni polovine RAMTOP, protoze textove okenko roluje ve skutecnosti veskera obrazova data GR. 0, tzn. ze mimo ctyr radku textoveho okenka take dalsich 20 radku po 40 bytech=800 bytu. Na tyto efekty se musi pri rezervovani mista v pameti brat ohled.
Kdo bude pracovat s GR.7,8,9,10,11,14 nebo 15 zjisti, ze dojde k porucham, jestlize bude RAMTOP posunut tak, ze DL a SM-data krizuji 4-kilobitovou hranici. Kdo pri zobrazeni (obrazku) dostane podivne vysledky, muze si pomoci snizenim RAMTOP o nasobek 16-ti (4*4 stranky=4k). Dalsi moznost zajistit si chranene misto v pameti je pomoci MEMLO(743,744=$2E7,$2E8).

107 $6B BUFCNT

Je pouzivan obrazovkovym editorem jako citac zbylych logickych radku

108,109 $6C,$6D BUFSTR

Obcas misto pro odlozeni. Vraci znak na ktery ukazuje BUFSTR

110 $6E BITMSK

Je pouzivan ridicim programem displeje pri bit-mapping rutine jako maska pro grafiku nebo docasna pametova bunka.

111 $6F SHFAMT

Pocita pozici pixelu uvnitr bytu graficke informace.

112,113 $70,$71 ROWAC

Akumulator (stradac) pro radkovou kontrolu pri PLOTovani.

114,115 $72,$73 COLAC

Stradac pro sloupcovou kontrolu

116,117 $74,$75 ENDPT

Koncovy bod linie, ktery ma byt dokreslen. Kontroluje PLOTovani bodu na jednu radku

118 $76 DELTAR

Radkovy rozdil = absolutni hodnota NEWROW-ROWCRS

119,120 $77$78 DELTAC

Sloupcovy rozdil = abs. hodnota NEWCOL-COLCRS. DELTAR a DELTAC se pouzivaji k vypoctu stoupani linie, ktera ma byt PLOTovana.

121,122 $79,$7A KEYDEF

Ukazatel na tabulku definici k prevodu klavesnicoveho kodu na ATASCII hodnotu(#:760 a 761=$2F8,$2F9)

123 $7B SWPFLG

Indikator pro kontrolu kurzoru u delene obrazovky (s textovym okenkem).

124 $7C HOLDCH

Zde je odlozena hodnota jednoho znaku pred provedenim CONTROL nebo SHIFT logiky.

125 $7D INSDAT

Pomocna pamet, kterou pouziva ridici program displeje pro znak pod kurzorem a pro rozeznani konce radku.

126,127 $7E,$7F COUNTR

Zacina s vetsi hodnotou z DELTAR nebo DELTAC. Odpovida poctu bodu, ktere je treba PLOTovat, aby se dotahla nejaka linie. Je pro kazdy vytvoreny bod zmensen o 1. Je-li tento byte=0, pak je linie ukoncena.

128,129 $80,$81 LOMEM

Ukazatel na nejnizsi adresu, ktera je k dispozici pro BASIC-program. Prvnich 256 bytu slouzi jako buffer pro preklad BASIC radku. Tokeny (prvni stupen prekladu, forma urcena vyrobcem pred ulozenim do RAM) jsou 1byte velke numericke hodnoty pro prikazy, operatory, funkce.

Hodnota je zde ovlivnena z MEMLO pri inicializaci pocitace nebo pri NEW, nikoli vsak pri RESET. Kdyz se zmeni MEMLO, musi se take prizpusobit hodnota v LOMEM. Je-li program uchovan pomoci SAVE, nejdrive se zaznamena od LOMEM do STARP 7 ukazatelu. Hodnota LOMEM je pritom odectena od vsech techto dvoubytovych ukazatelu. Oba prvni zaznamenane byty jsou tedy nuly. Po ukazatelich se zapisi tabulky jmen a hodnot promennych a potom tokenizovany BASIC program.
Pri LOAD bude hodnota MEMLO prictena ke kazdemu ze sedmi ukazatelu a pak zapsana do adres ukazatelu na nultou stranku. Pak bude v horni polovine MEMLO rezervovano 256 bytu pro vystupni buffer a zaroven bude v napojeni na tento buffer BASIC program ulozen.
S temito sedmi BASIC ukazateli se da udelat mnoho nepravosti. Pokud se obavame pristupu nekoho nepovolaneho do naseho programu, muzeme ho do jiste miry zabezpecit. Program pujde bez problemu kopirovat, ale po LIST dostane exoticky vzhled.

Nasledujici program ukazuje jaky zmatek zpusobi jeden radek:

```
0       REM  ADR128
1       REM  **********************
2       REM  * OCHRANA PRED LIST *
3       REM  **********************
10      A=A+10
20      B=10:C=5:D=7.5
30      F=A+B: G=A+C: H=A/D
40      ?F;G;H
50      GOTO 10
100     REM XIO33,#1,0,0,"D:SECRETE"
110     REM Y=PEEK(128)+PEEK(129)*256+3: POKE 128,Y-INT(Y/256)*
        *256: POKE 129,INT(Y/256): SAVE "D:SECRETE"
120     REM RUN "D:SECRETE"
```

10-50 obsahuji demo-program, ktery budeme chranit

110- odstrante REM a dejte radek jako prime zadani. Na disketu bude napsan soubor se jmenem "SECRETTE". Pritom bude, od vsech ukazatelu, odectena hodnota LOMEM.

120- jeste zde odstrante cislo radku a REM pro prime zadani. Prirozene muzete nejdriv zaradit LOAD a pak RUN. Vysledek na obrazovce zretelne ukazuje, ze program pracuje bez chyby. Ted zkuste dat LIST, LIST"C:".

100- k vycisteni diskety od blazniveho programu staci zadat primo tento radek. (bez cisla radku, bez REM), coz nam usetri vstup do DOS, vyvolani OPTION B a pokud na diskete neni zadny MEM.SVE, po OPTION B jeste znovu nahrat program z "D" do pocitace.

130,131 $82,$83 VNTP

Ukazuje na prvni byte tabulky jmen promennych. Jmena jsou pamatovana v tom poradi, v jakem poradi jsou do programu

zadana.Take vzdy, kdyz se preklepnete a chyba bude mit nahodnou podobu, kterou by mohla mit promenna. Jmena jsou ulozena ve forme ATASCII. Dotycne promenne jsou zaznamenavany az k otevrene zavorce a podle toho se rozeznavaji. Retezcove promenne konci znakem $. Jmena mohou byt dlouha (teoreticky) jako logicky radek. Pocitac je schopen rozeznat vsechny tyto promenne.
Aby se poznalo, kde konci jmeno promenne a zacina dalsi, nastavuje se u posledniho znaku jmena MSB(most significant bit=bit 7) s decimalni hodnotou 128. Pro obrazovy editor to znamena, ze vyda znak inverzne. Do tabulky lze zanest az 128 jmen promennych.

Kdyz pisete dlouhy program, muzete se dostat do uzkych. K tomu nekolik rad:

1- volte co nejkratsi jmena promennych. Setri se tim pametove misto a urychluje beh programu.

2- neni-li uz v tabulce jmen promennych misto, pomuze casto mala oklika. Je velmi pravdepodobne, ze pri programovani pouzijete jmena promennych, ktere nakonec nebudou pouzity. Nahrajte pomoci LIST na disketu, dejte NEW a znovu nahrajte pomoci ENTER. Nyni jsou vsechna nepotrebna jmena promennych smazana. Toto neplati u povelu SAVE, ten si tabulku promennych pamatuje.

3- pouzivejte casteji pomocne promenne. Nezalezi na tom, zda je v prvni smycce FOR-NEXT jmenovana jedna promenna a v dalsi jina. Muzete pro vsechny FOR-NEXT smycky pouzit stejny citac smycek(mimo ty, ktere jsou jedna ve druhe).

Je-li dost mista pro promenne, meli byste pro vsechny numericke hodnoty, ktere se vyskytnou vice nez 3*, pouzit promenne. Promenna potrebuje jedno misto v tabulce jmen a jedno v tabulce hodnot. Ve vlastnim BASICovem programu obsadi jedna promenna pouze tolik bytu, kolik zabere nazev(v nejlepsim pripade jeden byte). Kazda numericka hodnota zabira vzdy 6 bytu. Piste tedy GOTO J misto GOTO 300...V programu, kde je pouzito vice numerickych hodnot, lze pomoci promennych usetrit dost mista. Potize nastanou, pokud budete chtit dodatecne zmenit cisla radku pomoci obsluzneho programu (utility) "renumber", protoze ten neumi pocitat nenumericke skokove cile.

Ted se podivejte na tabulku jmen promennych:

```
0       REM  ************************
1       REM  * TABULKA PROMENNYCH  *
2       REM  ************************
10      A=PEEK(130) + PEEK(131) * 256: E=PEEK(132) + PEEK(133) *
        *256: FOR I=4 TO E-A-1: O=PEEK(A+1): IF O>128 THEN
        O=O-128: GOTO 30
20      ? CHR$(O);: GOTO 40
30      ? CHR$(O),: GOTO 40
40      NEXT I: END
100     B=1: C=2: D=3: DIM F$(1),G(2),H$(3)
200     REM V=PEEK(130) + PEEK(131)*256 + 4: POKE V+X,CHR + 128
300     REM X=0: CHR=84: REM B BUDE T
```

10- promenna A vyzvedne ukazatel na zacatku tabulky jmen promennych (adresy 130,131). E vyzvedne ukazatel na jejim konci (adresy 132,133). Nasledujici FOR-NEXT pak cte vsechny hodnoty mezi A a E. Protoze tento maly program sam obsahuje promenne (A,E,I,O), zacina smycka hodnotou 4. Zkuste dat jednicku nebo nulu. Podminka IF prezkousi, zda prave cteny byte je poslednim znakem jmena promenne. Jestlize ano, skoci na radek 30, kde carka slouzi tabulatorovemu skoku a tim promenne rozdeli.

20- protoze jsou jmena promennych v tabulce v ATASCII forme, staci prikaz CH$, aby byly citelne. Strednik radi znaky vedle sebe.

30- jako radek 20, jen se skokem tabulatoru

100- reprezentuje BASICovy program, ktery je zde ze 6-ti promennych

300- je-li ted zjistena pozice promenne v tabulce, lze pomoci POKE zmenit hodnotu ATASCII nektere pametove bunky a zmenit jmeno promenne. Prvni (resp. nulta) promenna na radku 100 je B. B ted bude T, jehoz ATASCII hodnota je 84. Nyni odstrante cislo a prvni REM radku a potvrdte. X znaci misto, ktere ma byt zmeneno. Pritom se musi pocitat kazdy znak, ne jen promenne. B je nulty znak tabulky, proto X dostane hodnotu 0. CHR prevezme ATASCII hodnotu noveho znaku.

200- zde se provede vymena promennych. V prevezme ukazatel(VNTP) na zacátek tabulky promennych. Pricte se 4, protoze predchozi program obsahuje 4 promenne (radka 10-40) a ktere se prezkoci. Prvni byte na ktery VTNP ukaze je vzdy 0. Do adresy V (prvni znak prvni promenne v tabulce)+X ofset znaku, ktery se ma zmenit, se pomoci POKE ulozi ATASCII-hodnota CHR noveho znaku. Protoze toto je v prikladu posledni znak promenne, musi se k rozeznani konce promenne nasadit jeste bit 7(CHR+128). Odstrante cislo radku a REM (pro prime zadani) a dejte RUN, aby se tabulka jmen znovu vytiskla a uvidite... Kdyz jeste chcete zacarovat F$ na V$, dejte za X trojku a za CHR 86, RETURN.

Na zaklade techto programovych radku se da sestavit obsluzny program (utility), ktery s komfortnim zadanim pro X a CHR a s dotazem, zda jde o koncovy znak umoznuje v programech zmenu jmen promennych. Je dobre psat takovy pomocny program s vysokymi cisly radku, (32700-32767), aby se v pripade potreby mohl nahrat nad stavajici BASIC program. Pritom se musi dat pozor, aby mezi zdrojovym a pomocnym programem nedoslo ke krizeni nazvu promennych.

Zamer prodlouzit jmeno promennych by se pri reseni setkal s velkymi potizemi, protoze by to znamenalo odsunout vsechny promenne, ktere za nim stoji v tabulce o odpovidajici pametove misto.

Snazsi je jmena promennych zkratit. Ma-li byt smazan posledni znak jmena promenne, vlozte na odpovidajici misto(napr. 128) pomoci POKE nulu. Tim se sice naroky na tabulku nezmensuji, je opet obsazeno dost bytu, ale da se ocistit pomoci

LIST"C:",NEW,ENTER"C:".
Pokud chcete chranit svuj programovy listing, muzete pritom vyuzit tabulku promennych:

```
0       REM  ADR131
1       REM  ******************************
2       REM  * ROZRUSENI TAB. PROMENNYCH  *
3       REM  ******************************
10      A=PEEK(130) + PEEK(131)*256
20      Z=PEEK(132) + PEEK(133)*256
30      FOR J=A TO Z: POKE J,129: NEXT J
100     B=10: C=30: D=2.5
110     F=C-B: G=B/D: H=D*C
120     ?F, G, H
```

10- ukazatel VNTP na zacatek tabulky jmen promennych je ulozen v A
20- VNTD na konec tab. jmen promennych je ulozen v Z
30- ted potrebujete naplnit vsechny byty mezi A a Z stejnou ATASCII hodnotou. Aby se vsechny promenne v listingu takto naplnily, zvolte ATASCII hodnotu + 128. Lze ulozit hodnotu 128 pomoci POKE, protoze pak vsechny promenne pri LIST zmizi. Zertovne jsou i hodnoty < 128, protoze pak neexistuje konec promennych. Korunou substituce je 155, tedy ATASCII hodnota pro RETURN. Kazda promenna je pak nahrazena RETURNem, coz vyda na dlouhy listing bez jmen promennych.

132,133 $84$85 VNTD

Ukazatel na tabulku hodnot promennych. Zde se zaznamenavaji vsechny okamzite hodnoty vsech promennych, ktere jsou v tabulce jmen. Kazda promenna zde zabere 8 bytu.

Prvni byte urcuje druh promenne.

Poznavaci cislo 00 znaci skalar (nedimenzovana numericka hodnota)
Poznavaci cislo 65 znaci dimenzovanou promennou (indexovana promenna)
Poznavaci cislo 129 znaci string (retezec)

Druhy byte obsazuje poradove cislo promenne, tj. 0-127.

Pri skalaru obsazuje poslednich 6 bytu BCD-konstantu (binary coded decimal). Pri dimenzovane promenne udavaji byty 3,4 offset ze STARP($140,141=$8C,$8D), tedy polohu v tabulce stringu

a poli. Byty 5,6 obsahuji prvni dimenzovani +1, byty 7 a 8 druhe + 1. Je-li promenna jednorozmerna, maji byty 7 a 8 hodnotu 0,0. U stringu udavaji byty 3 a 4 take offset ze STARP. Byty 5 a 6 obsahuji okamzitou delku retezce a byty 7 a 8 dimenzovanou delku.

Jestlize chcete naplnit string stejnymi znaky, lze toho dosahnout docela jednoduse. Staci zadat na prvni misto zadany znak, pak tentyz znak na posledni misto , a na druhe misto string samotny:

```
0       REM  ADR135
1       REM  ********************
2       REM  * NAPLNENI STRINGU *
3       REM  ********************
10      DIM TX$(266)
20      TX$(1) = "?"
30      TX$(266) = TX$
40      TX$(2) = TX$
100     ? TX$
```

```
10-     dimenzovani
20-     ulozeni prvniho znaku
30-     posledni, musite zadat TX$, ne "?".
40-     jeste urcit druhe misto a naplnit string
100-    to je vysledek
```

Neni nutne timto zpusobem naplnit cely string, muzete take zadat jen cast:

```
0       REM  ADR135X
1       REM  ********************
2       REM  * VYHOZENI PAMETI *
3       REM  ********************
10      DIM TX$(32767)
20      TX$(1)="!"
30      TX$(9999)=TX$
40      TX$(2)=TX$
100     ?TX$
```

```
10-     zde se rezervuje veskere volne misto v pameti pro TX$
30-     pak je naplneno prvnich 9999 mist "!". Je tu ale maly
        problem. Protoze se TX$ rozsiri na celou pamet, neni uz
        mozne zobrazeni. Musime se proto v radku 10 trochu
        omezit. Vyse popsany efekt funguje nejen s jednotlivymi
        znaky, ale i s celymi radami znaku. K tomu se na prvni
        misto ve stringu ulozi rada znaku, na poslednim miste
        retez sam a sice tak, ze posledni znak rady znaku obsadi
        posledni misto, ktere ma byt naplneno; tedy ma-li byt
        string zcela naplnen, pak DELKA STRINGU minus DELKA RADY
        ZNAKU. Konecne se musi na miste, ktere nasleduje za
        prvnim mistem ulozene rady znaku (na miste 1+ delka rady
```

znaku) ulozit string sam:

```
0       REM  ADR135A
1       REM ****************************
2       REM * PLNENI STRINGU PERIODOU *
3       REM ****************************
10      DIM TX$(264)
20      TX$(1)="!#?@ "
30      TX$(261)=TX$
40      TX$(5)=TX$
50      ? TX$
```

```
10-     dimenzovani na 264
20-     na mistech 1-4 se ulozi rada znaku
30-     pak se na mistech 261-264 ulozi TX$
40-     na miste 5, popr.  miste 5-8
```

136,137 $88$89 STMTAB

Ukazatel na tabulku prikazu, tedy vlastni program v BASICu v tokenizovane forme. (tokenizovana forma= kazde klicove slovo BASICu (vcetne operatoru) je vyjadreno jednobytovym symbolem = tokenem) Oba prvni byty nektereho tokenizovaneho BASICoveho radku obsahuji cislo radku (LO,HI*256). Nasleduje ciselny byte, ktery udava pocet pametovych bunek, ktere obsazuje BASICovy radek. Hodnota ciselneho bytu je nastavena az pote, co je ukoncen prevod do tokenu ve vystupnim buffru. (256 za LOMEM). Byte 3 take obsahuje pocet bytu, neboli offset (posunuti) od zacatku daneho radku programu k zacatku dalsiho radku.

Chcete-li se dozvedet, jake pametove bunky obsadi vas program, zkuste priklad:

```
0       REM  ADR136
1       REM  ****************************************
2       REM  * STARTOVACI ADRESY/CISLO BASIC RADKY *
3       REM  ****************************************
10      A=10
20      B=30
30      C=7.5
40      F=A*B
50      G=B/C
60      H=C-A
70      ? F,G,H
100     T=PEEK(136) + PEEK(137) *256
110     ZN=PEEK(T) + PEEK(T+1) *256
120     IF ZN>32767 THEN END
130     ? " BASIC-RADEK# "; ZN ; " MA START-ADRESU "; T;,
140     T=T+PEEK(T+2)
150     GOTO 110
```

```
10-70   obsahuji demo program
```

100- prepocita numerickou hodnotu STMTAB
110- tady se vypocita cislo radku ZN z bytu 0 a 1 prvniho tokenizovaneho radku
130- na obrazovce se ukaze cislo radku ZN a start adresa T.
140- nyni se pouzije 3.byte (T+2), aby se nasla start adresa nasledujiciho BASICoveho radku.
150- pristi radek muze byt zpracovan

Pro toho, kdo tohle prokoukne neni problem napsat rutinu na precislovani radku v programu. K tomu je treba vyhledat kazdy nulty nebo prvni byte kazdeho radku a ulozit tam zadane nove cislo radku.

```
0       REM  ADR137
1       REM  ************************************
2       REM  * ZMENA CISLOVANI BASICOVEHO RADKU *
3       REM  ************************************
10      A=25
20      B=7.5
30      C=17
40      D=A/B
50      G=B*C
60      H=C-A
70      ?F,G,H
1000    ZN=500: Z=PEEK(136) + PEEK(137)*256: SW=25
30 000  HI=INT(ZN/256): POKE Z,ZN-HI*256: POKE Z+1,HI:
        ZN=ZN+SW: Z=Z+PEEK(Z+2): GOTO 30 000
```

10-70 demo program
1000- ZN vyzvedne cislo prvniho radku, SW po krocich cislovani. Z vypocita ukazatel na prvni BASICovy radek.
30 000- zde se provedou premeny. HI vypocita ze ZN podil HI bytu. Pak se pomoci POKE ulozi zbytek LO byte (ZN-HI*256) do adresy Z a do adresy Z+1 hodnota HI bytu. Jako dalsi se po krocich SW zvysi cislo nasledujiciho radku. Ted se ukazatel Z zvysi o offset(PEEK(4+2)). Konecne nasleduje skok na 30 000, takze lze zpracovat dalsi radek.

Program konci hlasenim chyby. Jako posledni se zmenilo cislo radku 30 000, GOTO nenajde dalsi radek. Jestlize ted predem zadane hodnoty odstranite a date LIST, posledni radek ma ted cislo 850.
Kdyz urcujete hodnoty pro ZN a SW, musite dat pozor, aby nejvyssi cislo radku neprekrocilo 32767. Budete-li psat renumber-utility program, mel by se na to zeptat, protoze nikdy neni uplne jasne, az do ktereho cisla radku ma program

dosahnout.

Take zmenou cisla radku lze ochranit program pred LIST.(vsechny budou mit stejne cislo). Probehne to na stejnem principu jako premena cisel radku.
Kdyz uz je program napsan a BASICove radky jsou v pameti ve spravnem poradi, jsou cisla radku lhostejna. Presto, ze vsechny radky maji stejna cisla, prinasi program spravne vysledky. Normalne se da pouzivat SAVE,LOAD nebo LIST. Pouze v pripade zaznamu na perferii pomoci LIST zustane pri dalsim nahrani(ENTER) z programu jen posledni radek.

```
0       REM  ADR137A
1       REM  *************************
2       REM  * ZMENA   CISEL   RADKU *
3       REM  *************************
10      A=25
20      B=7.5
30      C=17
40      F=A/B
50      G=B*C
60      H=C-A
70      ?F,G,H
1000    Z=PEEK(136) + PEEK(137) *256
30 000  POKE Z,244: POKE Z+1, 1: Z=Z+PEEK(Z+2): GOTO 30 000
```

```
10-70-  demo program
1000-   vypocita ukazatel STMTAB
30 000- meni vsechna cisla radku na 500(244+1*256). Lze tady
        nastavit libovolnou hodnotu. Jestlize tato hodnota
        presahuje rozsah(LO=255,HI=255), nasleduje ERROR a
        ztrata programu.
```

```
138,139              $8A,$8B              STMCUR
```

Ukazatel probihajiciho prikazu BASICu. Pouziva se jako pristup na tokeny v prave zpracovavanem BASIC radku. Ve chvili, kdyz BASIC radek ceka na urcitou ulohu, tento ukazatel miri na radek primeho zadani (32768) a to umoznuje zvlast delikatni ochranu:

```
0       REM ADR138
1       REM ******************
2       REM * OCHRANA S RUN *
3       REM ******************
32766   POKE PEEK(138) + PEEK(139)*256+2,0:
        SAVE "D:FILENAME.EXT":
        NEW
```

```
32767   REM startujte s GOTO 32766
```

32766- pridejte tento prikaz k vasemu programu jako posledni radek a oznacte si program GOTO 32766. Drive nez ulozite program pomoci SAVE, znemozni se pomoci STMCUR zpetny skok na radek primeho zadani. Pak je program uchovan v tokenech, vsechny ukazatele (take STMCUR)jsou opatreny aktualnimi hodnotami. Pak se program zrusi NEW.

Tato ochrana pracuje tez s kazetovym magnetofonem. Zadejte SAVE"C:". Ale pozor, drive nez sestavite chranenou verzi programu touto metodou, zajistete si nechranenou, protoze puvodni program bude v pocitaci smazan.

Protoze bude program ulozen pomoci prikazu SAVE, muze se nahrat jen prikazem LOAD. Po primem zadani LOAD neni mozne zadne dalsi prime zadani, napr. RUN, LIST. Takovy program lze nahrat a bezprostredne spustit jen RUN"C:". Program tedy pracuje, ale nelze ho vylistovat.

Takovy program musi byt bezchybny, aby nedoslo k preruseni chybovym hlasenim. Tim by se uvolnila klavesnice. Jak takovy program sestavit je uvedeno v STOPLN(186,187=$BA,$BB). Mimoto musi byt blokovan RESET a BREAK, aby neslo program prerusit.

K overeni pouzijte libovolny program, do kterehо ulozte REM radek(radky) s tajnou zpravou.

```
0       REM ADR.138S
1       REM ****************
2       REM * TAJNA ZPRAVA *
2       REM ****************
100     REM .....
200     REM .....
300     REM .....
400     REM .....
800     ?"NOVA HRA PRO ZACATECNIKY,"
810     ?:?" ZMACKNI <START>-"
820     IF PEEK(53279)=6 THEN RUN
830     IF PEEK(53279)=1 THEN GR.0: GOTO 1000
840     GOTO 820
1000    ? CHR$(125):?:?"PO VLOZENI KOD. SLOVA"
1010    ?: ?"SE NECHRANENA VERZE PROGRAMU"
1020    ?: ?"ZAPISE NA DISKETU"
1030    ?: ?"PROGRAM VAS VYZVE S"
1040    ?"ENTER";CHR$(34);"D:CIA";CH$(34)
1050    DIM CW$(5)
1060    TRAP 1060
1070    INPUT CW$
1080    IF CW$="QRYXT" THEN LIST "D:CIA",0 1999
1090    GOTO 1060
20000   POKE PEEK(138)+PEEK(139)*256+2,0: SAVE"D:BOTSCHFT.MAD"
        :NEW
```

100-700 reprezentuji jednoduchy program, do kterehо bude vlozena tajna zprava

800-810 program konci zadosti zacit novou hru prez START

820 je-li stisknut START, bude odstartovan pomoci RUN

830 jen pouceny prijemce vi, ze nema stisknout START,

zmackne soucasne START SELECT a skoci tak na radek 1000
840- uzavre dotazovaci retezec
1000-1050-zde zacina druha ochranna rutina, ktera zada kodove slovo
1060- zadani(vlozeni) se musi pojistit proti chybe
1080- spravne slovo umozni LIST na disketu, posledni radek s ochrannou rutinou nebude napsan.
20000- LIST/LOAD ochrana. Nez program poslete adresatovi, poznacte mu GOTO 20 000.

140,141 $8C,$8D STARP

Ukazatel na prvni byte tabulky stringu a poli. Prvky poli jsou ulozeny v sestibytovem BCD formatu. Kdyz dimenzujete promennou na (9,9), bude obsazeno 10*10*6=600 bytu. Stringy naopak obsadi pro kazdy prvek 1 byte, tedy DIMG$(20) obsadi 20 bytu. Velkou cast pameti usetrime, kdyz cisla (numericke hodnoty) prevedeme do stringu.

```
1       REM  ADR.140
2       REM  ***********************
3       REM  * STRING MISTO ARRAY *
4       REM  ***********************
10      DIM A(10),Z$(20)
20      A(0)=9: A(1)=11: A(2)=1984: A(3)=140
30      Z$="09111985ADR140"
40      ? A(0); "."; A(1); "."; A(2); ":"; A(3)
50      ?:? Z$(1,2); Z$(3,4); "."; Z$(5,8); ":"; Z$(9,14)
```

Listing radku 30 obsahuje pseudograficke znaky.

10- zde se casto zbytecne predimenzuji pole a retezy.
20- A(0)-A(3) prijimaji data a poznavaci cislo. Efektivni spotreba pameti je 4*6=24 bytu
30- zde jsou vsechny znaky zavedeny do stringu. Spotreba pameti je 14 bytu, ackoli je zde jeste nekolik dodatecnych informaci.
40- vytiskne se datum a poznavaci cislo
50- stejneho cile se dosahne s castmi stringu. Numericke hodnote nesmi predchazet 0. Oznaceni dne 09(misto 9) lze vest jako string a jak ukazuje poznavaci cislo, je mozne michat pismena a cislice. Protoze stringy lze v BASICu prevest prikazem VAL na numericke hodnoty, daji se take s takto ulozenymi hodnotami provadet ruzne vypocty.

Pohled do tabulky poli promennych:

```
1       REM  ADR140A
2       REM  ****************************
3       REM  * STRING A ARRAY TABULKA *
4       REM  ****************************
10      DIM A(9), Z$(20)
20      REM FOR J=0 TO 9: A(J)=0: NEXT J
30      REM Z$="            "
40      A(0)=9: A(1)=11: A(2)=1984: A(3)=140
```

```
50      Z$="09111985ADR140"
60      ? A(0); "."; A(1); "."; A(2); ":"; A(3)
70      ?:? Z$(1,2); "."; Z$(3,4); "."; Z$(5,8); " : "; Z$(9,14)
100     ?CHR$(125)
110     SP=PEEK(140) + PEEK(141) * 256
120     FOR J=0 TO 9
130     FOR I=0 TO 5
140     ? PEEK(SP+J*6+J); "  ";
150     NEXT I
160     ?
170     NEXT J
200     FOR J=60 TO 79
210     ? PEEK(SP+J); "  "
220     NEXT J
230     ?
240     FOR J=60 TO 73
250     ? CHR$(PEEK(SP+J));
260     NEXT J
```

10-70- obsahuji vyse vysvetleny program
100- maze obrazovku
110- ukazatelem STARP vypocita pocatecni adresu tabulky stringu a poli
120- cte byty deseti promennych A
130- cte 6 bytu kazde promenne A
140- vytiskne promennou A ve forme byte-mezera...
150- zacne s dalsi promennou A na dalsim radku, aby odpovidajici byty vytvarely sloupce
200- protoze pole A obsadi 60 bytu, lezi(ATASCII) hodnoty Z stringu od 60 do 73
210- zde jsou cteny (PEEK) a vytisteny

Pri behu programu vidime, ze v tabulce je dira u promennych, ktere sice byly dimenzovany, ale dosud nebyla stanovena jejich hodnota. U poli to vede k neblahym chybam. Odstrante v radku 20 a 30 REM, nahradte v radku 240 hodnotu 73 hodnotou 79 a rozbehnete program znovu. Ted je tabulka docela pekna, protoze prvky poli, jejichz hodnota jeste nebyla stanovena, jsou vyplneny nulami. String byl v radku 30 zaplnen sledem prazdnych znaku a srdicek. Ted vidite, ze prvky byly obsazeny ve stringu v ATASCII-formatu, tj. prazdne znaky jsou jako 32, srdicka jako 0.
Pokud vite, na kterem miste tabulky hledana promenna stoji, muzete hodnoty prvku poli a stringu ukladat pomoci POKE. Prvky stringu se zaznamenavaji v ATASCII formatu, numericke hodnoty v sestibytovem formatu BCD.Tento format cisel zabere sice vice mista a operace s nim trvaji dele ,ale umoznuje pracovat s vetsimi cisly a vyssi presnosti. Nulty byte BCD konstanty obsahuje v bitu 7 znamenko (signum , +=nenastaveno, -=nastaveno) a v bitech 6 obsahuje 0 exponent k zakladu 100. Je-li decimalni hodnota bitu 6 az 0 vetsi, nebo rovena 64, preda nulty byte pozitivni exponent tak velky, o kolik hodnota presahuje 64. Tedy 64 znamena, ze E=0 (100↑0=1, cislo lezi mezi 0-99), 65 znamena, ze E=1 (cislo lezi mezi 0-9999). Hodnoty mensi nez 64 davaji zaporne exponenty, tedy mista po desetinne carce.
Dalsich 5 bytu obsahuje cislice, ze kterych se cislo sklada. Prvni je vzdy binarne kodovana cislice obsazena v jednom

```
Napr.: hodnota promenne.........-12
         "      nulteho bitu.....192(128+64)
         "      prvniho bitu.....12
horni nibble obsahuje v nejnizsim bitu.....1(dec.1=0001)
dolni nibble     "     "       "      " .....2(dec.2=0010)
Ma-li byt obsazeno cislo 78, pak je nulty byte 64, sedmicka se
nastavi v hornim nibblu jako 0111, osmicka se nastavi v dolnim
nibblu jako 1000.
Priklad:

dec. cislice                        7              8

bity                      0  1  1  1      1  0  0  0

cislo bitu                7  6  5  4      3  2  1  0

hodnota                 128 64 32 16      8  4  2  1
```

V nasledujicim programu je hodnota prvku pole urcena primo v tabulce. Cislo dostane sled cislic a desetinna tecka je nastavena pomoci exponentoveho bitu mezi 6. a 7. misto. Nula na 4. miste za des. teckou se neobjevi.

```
1       REM  ADR141
2       REM  ************************************
3       REM  * ZMENA DIMENZOVANYCH PROMENNYCH *
4       REM  ************************************
10      DIM Z(9)
20      FOR J=0 TO 9: Z(J)=0: NEXT J
30      SP=PEEK(140) + PEEK(141) *256
40      POKE SP,66: REM 10↑2
50      POKE SP+1,18: REM 1/2
60      POKE SP+2,52: REM 3/4
70      POKE SP+3,86: REM 5/6
80      POKE SP+4,120:REM 7/8
90      POKE SP+5,144:REM 9/0
100     ? Z(0): ?: ?
110     FOR J=0 TO 5
120     ? PEEK (SP+J); "     ";
130     NEXT J


10-     velikost DIM je libovolna
20-     vsechny prvky pole jsou nulove
30-     vypocet adresy na kterou ukaze STARP
40-     nulty byte obsahuje znamenko a exponent
50-     prvni byte si pamatuje cislice 1 a 2 binarne kodovane
              (0001/0010), cimz ziska hodnotu 18
60-     druhy byte obsahuje cislice 3 a 4(0011/0100=52)
70-     treti   "     "        "   5 a 6(0101/0110=86)
80-     ctvrty   "     "        "   7 a 8(0111/1000=120)
90-     paty     "     "        "   8 a 0(1001/0000=144)
100-    vytiskne dec.  Z
110-130-sest datovych bytu odpovidajici BCD konstanty
```

142,143 $8E,$8F RUNSTK

Adresa tzv. "runtime stacku" (pomocneho zasobniku pri behu programu). Obsahuje udaje pro vsechny prave probihajici GOSUB (vzdy 4 byty) a FOR-NEXT (vzdy 16 bytu).
Zadani GOSUB obsahuje v bytu 0 nulu. Jako indikator pro "GOSUB", v bytu 1 a 2 cislo radku, ktery je v GOSUB vyvolan (LO,HI) a v tretim bytu offset v radku, aby se orientoval RETURN a mohl provest pristi prikaz. Zadani FOR-NEXT obsahuje v bytech 0-5 konecnou hodnotu pro ciselne promenne, tj. hodnotu udanou za "TO" ve forme BCD konstanty. V bytech 6-11 udaj STEP, tez v BCD forme. V bytech 12-13 cislo radku, ktere je narizeno FOR. V 15 bytu radkovy offset prikazu FOR. Pomoci RUNSTK je zaroven znacen konec tabulky poli promennych.

144,145 $90,$91 MEMTOP

Ukazuje na konec pametoveho rozsahu, ktery je obsazen BASICovym programem. Kolik pameti zbyva mezi MEMTOP, zacatkem DL a obrazovkovou pameti, zjisti FRE(0) odectenim odpovidajiciho ukazatele SDLSTL a MEMTOP. Ohraniceni pametoveho prostoru musi byt zaneseno v RAMTOP (106=$6A)

186,187 $BA,$BB STOPLN

Pamatuje si cislo radku, ve kterem doslo k preruseni programu pri ERROR, STOP, TRAP, BREAK. S timto registrem se spoji kazde preruseni zpusobene programovou chybou, napr. v ramci ochrany software, nebo jde-li o nejjednodussi formu k premosteni programoveho preruseni pomoci vhodnych cislicovych hodnot. V nasledujicim programu je dimenzovana dvojrozmerna indexovana promenna, jejiz prvky jsou zvyseny pomoci nahodnych hodnot. Protoze tyto hodnoty mohou byt i vyssi nez bylo dimenzovano, pouzije se TRAP trik:

```
0       REM  ADR186
1       REM  ********************
2       REM  * ERROR - OCHRANA *
3       REM  ********************
9       TRAP 9: F=F+1: GOTO PEEK(186)+PEEK(187)*256+10
10      REM TRAP=CIL SKOKU PRI CHYBE
100     DIMM(6,6): FOR I=0 TO 6: FOR J=0 TO 6:
        NEXT J: NEXT I
110     X=INT(RND(0)*6)
120     Y=INT(RND(0)*6)
130     AL=4-X: AR=4+X
140     BL=4-Y: BR=4+Y
150     M(AL,BL)=M(AL,BL)+1
160     M(AR,BL)=M(AR,BL)+1
170     M(AR,BR)=M(AR,BR)+1
180     M(AL,BR)=M(AL,BR)+1
190     Z=Z+1: IF Z<50 THEN 110
200     FOR J=0 TO 6
210     FOR I=0 TO 6: ? M(J,I); "  ";: NEXT I: ?
220     NEXT J
230     ?:?:? F
```

9- je-li TRAP v cinnosti, dosahne program teto radky, kde se TRAP obnovi. Nasledujici GOTO cte ze STOPLN, ve kterem radku TRAP vystoupil a vede na radek o 10 vyssi (nasledujici).Cely program ma skoky po 10 a v poslednim radku nemuze vystoupit chyba(protoze sam prikaz GOTO by vedl k hlaseni chyby a tim by program tento radek nikdy neopustil). V ramci ochrany software dava takovy radek jistotu, ze uzivatel nemuze program znicit spatnym zadanim. Mozne je take: 100- TRAP 20000; 20000-NEW: LOAD"D: filename.ext". Promenna F pocita kolik chybovych hlaseni bylo v tomto programu premosteno TRAPem.

10- v prvnim probehu programu stoji ukazatel STOPLN na 0. V radku 9 tedy vyjde GOTO 10. Proto musi program obsahovat radek 10. Muze take skocit na REM.

100- array M, dimenzovana na 7*7

110-120-X a Y prijmou prislusne hodnoty 0-5.

130-140-pomoci X a Y dostanou promenne AL, AR, BL, BR hodnoty z nichz nektere nemohou byt pouzity pro array.

150-180-kdyby k tomu presto doslo, dojde k hlaseni: chybny rozsah cisel. Ptat se na nevhodne hodnoty X a Y pomoci IF je nepripustne, zvlast pro pripady, kdy jen jedna z promennych presahuje rozsah a kdy pak prvky pole, ktere budou nevhodnou hodnotou obsazeny, budou jeste zvyseny. Dojde-li k pokusu vyvolat nevhodny prvek pole, dojde k hlaseni chyby a TRAP pokracuje na dalsim radku.

190- program je 50* zpracovan

200-220-pak se pole promennych vytiskne a v radku

230- dava F zpravu, kolikrat doslo k chybe

195 $C3 ERRSAVE

Obsahuje cislo chyby, ktera vedla ke STOP nebo TRAP.

201 $C9 PTABW

Urcuje pocet mezer mezi, ktere byly narizeny zapsanim desetinne carky v prikazu PRINT; tedy odstup mezi poslednim znakem predchoziho PRINT a prvnim znakem nasledujiciho. Funkce je nezavisla na klavese TAB. Default hodnota je 10. Nejmensi mozna hodnota je 3. K hodnote ulozene do PTABW pomoci POKE se pricita 2, tedy POKE 201,1 zpusobi skok tabulatoru 3 mezery. Nejvyssi mozna hodnota je 255, tedy odstup 257 sloupcu. Kdyz se do PTABW ulozi 0, pak se system pri prvni desetinne carce zastavi na PRINT a musi nasledovat RESET.

212-241 $D4,$F1 ruzne

Registr pro operace v pohyblive desetinne carce

242 $F2 CIX

Znakovy index. Rozdil mezi tim, co se uklada do textoveho bufferu a obsahem INBUFF

243,244 $F3,$F4 INBUFF

Ukazuje na vstupni textovy buffer, tedy na buffer pro uzivatelsky programovy radek.

245-250 $F5-$FA ZTEMP1-3

Pomocny registr

251 $FB RADFLG

Indikator(flag)pro RAD(obloukova mira) nebo DEG(stupne). Default hodnota 0 necha probehnout vsechny trigonometricke funkce v obloukove mire. Sestka nebo BASICovy prikaz DEG prepne na stupne.

256-511 $100-$1FF Page 1

Toto je rozsah zasobnikove pameti pro OS, DOS, BASIC. Strojove prikazy USR, PHA a preruseni zpusobi zapis dat na stranku 1. Pri powerup nebo RESET se nasadi ukazatel zasobniku na adresu 511. Stack je pak opsan az dolu na 256. V pripade overflow (preplneni) skoci ukazatel stacku zpet na 511.

512,513 $200$201 VDSLST

Vektor pro display list interrupt(DLI). Tady se uklada adresa, ke ktere se ma skocit v DLI.
DLI se pouziva, aby se v mikrosekundach horizontalniho zatemneni vyridily dalsi programove ulohy (napr. melodie). V DLI lze menit GR. mody hodnoty v barvovych registrech...Tak lze v kazdem radku nastavit jinou barevnou hodnotu(256 barev najednou) OS nepouziva zadny DLI. Musi byt urcen, napsan a vektorovan uzivatelem. VDSLST je inicializovan tak, ze ukazuje na 59315($E7B3). K umozneni DLI se musi nejprve nasadit 54286($D40E) na 192, aby ANTIC poznal pozadavek(request). Pak se musi ulozit (POKE) na 512(LO) a 513(HI) adresa na ktere zacina strojova rutina, ktera ma byt provedena behem DLI. V instrukci display listu, kde ma DLI nastat, se musi nastavit bit 7. Dle grafickeho modu je pro DLI k dispozici 14-61 strojovych cyklu. Nejdrive se musi vsechny registry 6502 posunout do zasobnikove pameti(stack) a DLI musi koncit instrukci RTI.
Protoze DLI musi probihat ve strojovem kodu, mohou se zmeny zadat (POKE) primo do hardware-registru. Zmeny ve stinovych registrech, kterych se ma dosahnout, mohou byt dotazany jen po VBLANK.Proto jsou zmeny napr. barvovych hodnot v barvovych registrech sice opticky velmi rychle k dosazeni, ale vzdy pusobi na celkovy obsah obrazovky. Pomoci DLI je ale napr. mozne nastavit barvovy registr 710(pozadi u GR.0) na 140(svetle modra obloha a po nekolika zpusobovych radcich nastavit pro zbytek grafickeho okenka na 182(majova zelen)

```
1     REM  ADR512
2     REM  *******
```

```
3     REM  * DLI *
4     REM  *******
10     POKE 710,140: POKE 712,0:
      POKE 709,2
20    X=PEEK(560)+PEEK(561)*256
30    P=1536:FOR DLI=P TO P+10:
      READ B:POKE DLI,B:NEXT DLI
40    DATA 72,169,182,141,10,212,
      141,24,208,104,64
50    POKE 512,0:POKE 513,6
60    POKE 54286,192
70    POKE X+6+J,130
80    FOR I=0 TO 300:NEXT I
90    POKE X+6+J,2: J=J+1:
      IF J<21 THEN 70
100   GOTO 100
```

10- zde se ukladaji (POKE) barvove hodnoty. Registr 712 urcuje pri GR.0 kraj(0=cerny);709 jas pisma

20- prepocita ukazatel SDLSTL na display list

30- P je pocatecni adresa strojove rutiny. P je zacatek stranky 6(6*256=1536). Zde lezi male chranene misto RAM pod LOMEM. FOR-NEXT smycka cte datove byty ulozene v pametovych bunkach lezicich za P. Tyto predstavuji strojovy program.

40- strojova rutina. Vedle teto metody (umisteni stroj. rutiny do databytu) lze tez stroj. rutinu ulozit do stringu. Datove byty v tomto radku je mozne zmenit v retezec pomoci FOR J=0 TO 10:READ A:B$(J,J)=CHR$(A):NEXT J. Strojova rutina se pak radi do tabulky pole promennych a pri internich posunech v pameti pomoci OS je spolehlive posouvana, coz uzivateli bere starost o prekroceni chraneneho mista. Pocatecni adresa stringu v tabulce stringu a poli muze byt zadana BASICovym prikazem ADR(B$) a USR(ADR(B$)) pripravi pocitac k cinnosti

50- vektor DLI je nastaven na stranku 6(LO=0, HI=6)

60- v ANTICovem NMIEM(54286=$D40E) se musi nastavit bity 7 a 6 (dec.192), aby se umoznil DLI

70- zde se zmeni byte ANTIC povelu v display list ze 2(pro GR.0-radek) na 130 (2+128 pro GR.0-radek s DLI)

80- okamzik pauza

90- zde se ANTIC povel opet zmeni na 2. Pak se J zvetsi o 1, takze pri nejblizsim prubehu je uveden DLI na nejblizsi nizsi radek. Tim se dosahne efektu, ze horni dil grafickeho okenka (obloha) se radek po radku roztahuje, takze se zda, ze letadlo stoupa vzhuru.

Pokud se vam nastavene barvy nelibi, zmente pro: oblohu POKE 710,n; kraj POKE 712,n; jas pisma POKE 709,m. Pro m,n volit hodnotu 0-255, hodnoty 0-14 pro m ovlivnuji pouze jas.

Chcete-li menit barvy po DLI, tedy v dolnim dilu graf. okenka, pak musite take zmenit odpovidajici databyte ve stroj. programu. To je treti hodnota v radku 40. Take zde jsou dovoleny jen prime hodnoty.

Protoze mame jen tuto jednu adresu vektoru DLI, musi se zmenit vlastni vektor rutiny DLI ma-li se provest vice DLI. Krome toho by se mel potlacit impulz z klavesnice,

protoze s DLI dochazi k interferencim a podobnym porucham.

514,515 $202,$203 VPRCED

Vektor k IRQ rutine pro seriovou sbernici

516,517 $204,$205 VINTER

Vektor k IRQ rutine seriove sbernice

518,519 $206,207 VBREAK

Vektor pro instrukci 6502 BRK(opcode=00). Nema nic spolecneho s klavesou break.

520,521 $208,$209 VKEYBD

POKEY vektor klavesnicoveho interruptu. Pouziva se pro vymazani interruptu pomoci libovolne klavesy mimo BREAK. Inicializuje na 65470 rutinu (keyboard-IRQ-OS)

524,525 $20C,$20D VSEROR

POKEY vektor na rutinu k vysilani seriovych dat.

526,527 $20E,$20F VSEROC

POKEY vektor pro ukonceni serioveho prenosu dat

528-533 $210-$215 VTIMR 1,2,4

Interrupt vektory pro POKEY timer 1,2,4(AUDF 1,2,4). Spocita odpovidajici POKEY timer az na 0 a skoci na adresu na kterou ukazuje prislusny vektor.

534,535 $216,$217 VIMIRQ

Hlavni vektor IRQ interruptu

536,537 $218,$219 CDTMV1

System timer 1. Pocita po 1/50s zpet k nule. Kdyz ji dosahne, skoci na adresu ulozenou v prislusnem vektoru CDTMA1 (550,551=$226$227). Timer 1 by nemel byt pouzivan, protoze OS ho pouziva pro I/O rutinu.

538-545 $12A-$221 CDTMV2-5

viz. CDTMV1

546,547 $222,$223 VVBLKI

Vektor pro okamzity VBLANK interrupt. Tento ukazatel lze zmenit behem strojove rutiny VBLANKu, muze mit delku asi 3500 strojovych taktu.

548.549 $224,$225 VVBLKD

Ukazatel pro odlozeny VBLANK interrupt, ktery muze mit az 2000 strojovych cyklu.

550,551 $226,$227 CDTMA1

Skokova adresa pro TIMER 1(CDTMV1)

552,553 $228,$229 CDTMA2

Skokova adresa pro TIMER 2(CDTMV2)

554 $22A CDTMF3

Skokova adresa pro TIMER 3(CDTMV3)

555 $22B SRTIMR

Timer, ktery se pouziva k osaleni klavesnice. Zpusobuje zpozdeni do te doby, nez je nastavena opakovaci funkce klavesnice. Vzdy, kdyz je klavesa stisknuta zacina timer na 48. Jak dosahne STIMR 0 a je-li stale stisknuta prejde hodnota klavesy s opakovacim casem 1/10s do CHR(764=$2FC)

556 $22C CDTMF4

Flag pro timer 4 (CDTMV4)

557 $22D INTEMP

Pomocny registr pouzivany SETVBL rutinou

558 $22E CDTMF5

Flag pro timer 5(CDTMV5)

559 $22F SDMCTL

Stinovy registr od 54272. DMA (direct memory access) rizeni pro ANTIC(viz. dodatek PMG)

bit 5 (32) umoznuje ANTICu nacist instrukci DL
bit 4 (16) jednoradkove rozliseni hrace pri PM grafice
bit 3 (8) umoznuje player DMA
bit 2 (4) umoznuje missile DMA
bit 1 (2) rozsah displeje
bit 0 (1) rozsah displeje

bit 1 a 0 stanovi v jakem rozsahu (sirce) bude TV obrazovku ANTIC zpracovavat.
Bit 1 nastaven, bit 0 nastaven: siroky displej (v literature se oznacuje jako "playfield"- hraci pole; zde ve spojeni s PM grafikou). Pri sirokem displeji se v GR.0 vejde na kazdy radek 48 znaku. Tim se meni pouze znazorneni samo. Editor pracuje dale se 40 znaky pro radek. Kdyz tedy zmenite nastavenim obou dolnich bitu sirku displeje a pak napr. program vylistujete, objevi se radky o dotycnou hodnotu presazeny. Tedy pri 48 znacich zacne editor na 41. miste s 1. znakem pristiho radku..
Bit 1 nastaven, bit 0 nenastaven: vyda normalni sirku na 40 znaku
Bit 1 nenastaven, bit 0 nastaven: uzke hraci pole s 32 znaky. Zde vyda editor na na prvnich osmi sloupcich nasledujiciho radku zbyle znaky ze sveho radku o 40-ti znacich a zacne u 9. sloupce se svym druhym radkem.
Bit 1 nenastaven, bit 0 nenastaven: zadny displej. Vystup na obrazovku lze pomoci ANTICu zcela odpojit. K tomu je treba nastavit bity 5,1,0 na nulu. Protoze se tim odlehci cely system, probihaji nektere operace rychleji. Prilezitostne zapnuti nebo vypnuti obrazu je vyhodne zvlaste pri beznych vypoctech. Nejjednodussi je nejdrive aktualni hodnotu adresy 559 prevest na promennou(PEEK..). Pak POKE 559,0. Kdyz ma byt obrazovka opet ozivena, ulozte do STMCTL hodnotu promenne: POKE 559,promenna. Tim je zajisteno, ze po vypnuti bude v tomto registru opet stara hodnota. Vypnuti obrazovky muze mit vyznam take v pripade, kdy je obraz urcen vysokym grafickym modem, coz muze trvat i nekolik minut. Jestlize displej predem vypnete, ulozi se grafika do obrazovkove pameti a pak v 559 opet zapnete, objevi se cela grafika najednou. Jeste lepsi metodu predstavuje program PAGING(88,89=$58,$59)

560,561 $230,$231 SDLSTL

Stinovy registr od 54274 a 54275. Vektor na display list(DL). DL lezi primo pred obrazovkovou pameti. Je to strojovy program, ktery ridi ANTIC. Obsahuje udaje o poloze a interpretaci obrazovych dat. Existuji prikazy pro kazdy graficky mod, DLI, horizontalni a vertikalni rolovani a 2 skokove prikazy (viz. dodatek DL). Chcete-li se podivat na DL, vyzkousejte tento program:

```
1     REM  ADR560
2     REM  ****************
3     REM  * VYTISTENI DL *
4     REM  ****************
10     DIM D(201)
20     ? CHR$(125)
30     ?:?"ZADEJ GR. MOD"
40     ?:?"OD 0 DO 11(15)"
```

```
50      ?:?"GR.-MOD BEZ TEXT. OKNA +16"
60      ?:?"A RETURN"
70      ?"_____ "
100     INPUT G
110     GR. G
120     DL=PEEK(560)+PEEK(561)*256
130     FOR J=0 TO 201
140     D(J)=PEEK(DL+J)
150     Z=Z+1
160     IF D(J)=65 THEN POP: GOTO 180
170     NEXT J
180     J=Z:D(J)=PEEK(DL+J)
190     D(J+1)=PEEK(DL+J+1)
200     GR. 0
210     POKE 201,1:POKE 83,37
220     FOR J=0 TO Z+1
230     ? D(J),
240     NEXT J
250     GOTO 250
```

10- pole promennych D prevezme datove byty DL. Nejdelsi DL(GR.8+16) je dlouhy 202 byty.
20-70- vytisteni instrukci v text. okenku
100- zadani pro G od 0 do 31 vede k dumyslnym vysledkum. Program neni chranen proti chybnym udajum.
110- je zapojen zvoleny graficky mod
120- DL prevezme pocatecni adresu display listu
130- citac smycek je nastaven na max. hodnotu
140- je vybran datovy byte z DL a ulozen do D(n)
150- Z pocita ctene byty
160- 65(JVB-jump at vertical blank) je posledni prikaz DL, kteremu nasleduje skokovy cil(2 byty). Je-li pri cteni DL nalezeno 65, opusti program FOR-NEXT smycku s POP a pracuje na radku dal
180- kde je precten predposledni a v radku
190- posledni byte DL
200- pro tisk DL dat je zapojen GR.0
210- delka PRINT-tabulky (PTABW) je definovano jednickou, tedy 3 sloupce a pravy okraj(RMARGN) je urcen na 37.
220-240-vytiskne databyty DL na obrazovku
250- zamezuje ukonceni programu pomoci READY. Data z GR.24 pri tomto formatovani zaplni celou obrazovku. Pri ukonceni programu pomoci READY jsou 2 radky nahore z obrazovky rolovany. Musite tedy program ukoncit sami klavesou BREAK. Co znamenaji ruzne ciselne hodnoty DL, je vysvetleno v dodatku display listu.

562 $232 SSKCTL

Stinovy registr 53775. Ridi seriovou sbernici.

563 $233 LCOUNT

Citac bytu pro nahravaci rutiny

564 $234 LPENH

Stinovy registr od 54284. Horizontalni poloha svetelneho pera.

565 $235 LPENV

Stinovy registr od 54284. Vertikalni poloha svetelneho pera. Polohove hodnoty svetelneho pera odpovidaji hodnotam pri vystupu v PM grafice(neodpovidaji obycejnym hodnotam sloupcu a radku pri okamzitem vystupu v grafickem modu)

566,567-vektor pro interrupt pomoci klavesy

BREAK. Muze byt pouzit, aby se preslo na uzivatelskou rutinu ke zpracovani BREAK-key interruptu, napr. aby se program v ramci ochrany software pri aktivovani BREAK vymazal, nebo se preslo na dalsi nahravaci pochod.

568,569- volne

570-575-$23A-$23F .....

Interni pomocne promenne pro zpracovani seriovych dat.

576-579-$240-$243 .....

Interni pomocne promenne pro diskovy boot.

580- $244 COLDST

Indikator studeneho startu. Kazda hodnota mimo 0 znamena, ze probiha powerup inicializacni rutina. Je-li zde 0, vede RESET k teplemu startu.

582- $246 DSKTIM

Velikost time-out pro disketu (R/W =7 , formatovani=160).

623- $26F GPRIOR

Stinovy registr od 53275. Urcuje pri PM grafice prioritu zobrazeni obrazovych prvku, pokud se prekryvaji. Vytvari opticky "predozadni efekt. Prekryvajici se hraci mohou prijmout treti barvu, ctyri strely mohou byt pouzity jako paty hrac.

zkratky:P=player, PF=playerfield, BAK=pozadi a kraje
bit 7(128) */GTIA-mod
bit 6(64) */GTIA-mod
bit 5(32) prekryvajici se hraci dostanou treti barvu
bit 4(16) paty hrac misto 4 missiles
bit 3(8) sled priority:PF0-1,P0-3,PF2-3,BAK

```
bit 2(4)   sled priority:  PF0-3,P0-3,BAK
bit 1(2)   sled priority:  P0-1,PF0-3,P2-3,BAK
bit 0(1)   sled priority:  P0-3,PF0-3,BAK
```

bity 7 a 6 umoznuji GTIA mody 9,10,11. Tyto graficke mody pouzivaji stejny DL jako GR. 8, pouze jsou obrazova data jinak interpretovana. Misto 1 bitu pro pixel (tedy znazorneni 320 pixelu v radku ve 2 barvach) jsou v tomto tretim GTIA modu cteny pro pixel 4 bity. Tim lze odlisit max. 16 barevnych tonu. Aby to vyslo na stejne mnozstvi dat, je kazdy pixel znazornen 4* vetsi, takze jeden modovy radek naplni 80 pixelu.

bit 7 nenastaven, bit 6 nastaven: GR.9 (pixel je ve stejnem barevnem tonu, ale v 16-ti ruznych stupnich jasu)
bit 7 nastaven, bit 6 nenastaven: GR.10 (pixel je v 9-ti moznych volne definovanych barevnych tonech, k dispozici je 9 barvovych registru)
bit 7 nastaven, bit 6 nastaven : GR.11 (pixel je v 16-ti moznych standartnich barvach, jas pro vsechny stejny.

Neni-li nastaven bit 5, bude obrazovkova plocha cerna a obsazena dvema hraci.
Pri nastaveni bitu 4 se jedna s PM pametovym rozsahem pro 4 missiles jako s 5. hracem.
V bitech 0-3 lze definovat "odporujici" priority. Jestlize v dusledku toho vznikne kolizni situace, je dotycny rozsah obrazovky cerny.

624 $270 PADDLE0

Obsahuje hodnotu PADDLE0. Paddles se tez nazyvaji pots(potenciometry)

625-631 $271-$277 PADDLE1-7

U XL modelu se pouzivaji jen paddles 0-3.

632 $278 STICK0

Obsahuje hodnotu joysticku 0. Muze mit 9 ruznych ciselnych hodnot, ktere budou odpovidat jen dolnim 4 bitum.

(nastaven=+; nenastaven=-)

```
bity0-3.(+).(00001111=15).J v klid poloze
bit 3...(-).(00000111=7)..J vpravo
bit 2...(-).(00001011=11).J vlevo
bit 1...(-).(00001101=13).J dole
bit 0...(-).(00001110=14).J nahore
bity3 a 1.(-).(0101=5)....J vpravo dole
bity3 a 0.(-).(0110=6)..J vpravo nahore
bity2 a 1.(-).(1001=9)..J vlevo dole
bity2 a 0.(-).(1010=10).J vlevo nahore
```

633-635 $279$27B STICK1-3

Jako STICK0. U XL modelu jen STICK0-1.

636 $27C PTRIG0

Ukazuje, zda je stisknut na otocnem regulatoru 0 spoust. Obsahuje 1 kdyz ne, 0 kdyz ano.

656 $290 TXTROW

Pri modu deleneho okenka(GR.+text) je graficke okenko ovladano ridicim programem displeje a textove editorem("E:"). Pouzivaji se oddelene IOCB a 2 kurzory na sobe nezavisle. Graficka data pro textove okenko jsou krome toho v pevne stanovenem pametovem rozsahu, ktery je oddelen od ostatni obrazove pameti(viz. dodatek-obrazova pamet).
TXTROW obsahuje radek kurzoru textoveho okenka. Protoze textove okenko je vysoke jen 4 radky, mohou zde byt hodnoty 0-3. Kdyz se zmenou DL zvetsi okenko na 5 nebo vice radku, nastanou s pozicemi kurzoru stejne potize jako GR.7 1/2 pro registr DINDEX(87=$52)

657,658 $291,$292 TXTCOL

Obsahuje sloupec kurzoru textoveho okenka; tj. hodnoty 0-39. Pri vsech standartnich DL je HI byte (658) tedy 0.
Prikazy BASICu POSITION, PLOT nebo LOCATE se vztahuji jen na kurzor v grafickem okenku. Jsou tedy v textovem okenku neucinne a mohou se nahradit jen odpovidajicim POKE(viz. obrazova pamet)

659 $293 TINDEX

Obsahuje momentalni graficky mod textoveho okenka. Text. okenko je ekvivalent DINDEX(87=$57) a je stale 0 pokud je SWPFLG(123=$7B) 0.
TINDEX se inicializuje na 0. Graficky mod textoveho okenka ale muze byt pri odpovidajicich zmenach DL libovolne programovan. Ma-li se to stat, je nutne prizpusobit hodnotu v tomto registru.

660,661 $294,$295 TXTMSC

Pocatecni adresa obrazove pameti pro textove okenko, ktera obsahuje vlastni, na obrazove pameti graf. okna nezavisly obsah. Proto je take mozne textove okno zakryt, aniz by se ztratila data obrazoveho rozsahu.
TXTMSC ukazuje na pametovou bunku, ktera obsahuje obrazova data pro levy horni roh textoveho okenka. Je to ekvivalent SAVMSC(88,89=$58$59) a manipuluje se s nim stejne jak tam bylo popsano.

662-667 $296-$29B TXTOLD

Tento registr obsahuje ekvivalent OLDROW(90=$5A), OLDCOL(91,92=$5B,$5C), OLDCHR(93=$5D), OLDADR(94,95=$5E,$5F) pro data kurzoru v textovem okenku.

668-671 $29C-$29F /////

Docasne registry.

672 $2A0 DMASK

Maska pro pevne spojeni bitu v grafickem databytu, ktery nalezi k 1 pixelu. Podle grafickeho modu lze sdruzit do 1 bytu 8,4,2 nebo 1 pixelu dohromady. Maska obsahuje 0 pro vsechny bity, ktere nemaji byt pro zobrazeni jednoho prave zpracovavaneho pixelu nastaveny a 1 pro bity se kterymi pixel aktualne koresponduje. Dle graf. modu se nastavuje tato maska:

```
10000
00001100
00000011
GR.3,5,7
12,13,15...2bity/pixel=4 pixely/byte

10000000
01000000
do
00000010
00000001
GR.4,6,8,
14.........1bit/pixel=4 pixely/byte
```

673 $2A1 TMPLBT

Docasny registr pro bitovou masku.

674 $2A2 ESCFLG

Flag pro escape. Normalne na 0. Je-li stisknuta klavesa ESC, pak na 128. Po vystupu nasledujicich znaku se opet nastavi na 0. Kdyz maji byt ATASCII kontrolni znaky znazorneny bez ESCAPE, muze se DSPFLG(766=$2FE) nastavit na nenulovou hodnotu.

675-689 $2A3-$2B1 TABMAP

Maska tabelatoroveho skoku. Vztahuje se na logicky radek, ktery zahrnuje 3 fyzicke radky po 40 sloupcich. To vyda 120 sloupcovych pozic. TAB masku prevezme 15 bytu(15*8bitu=120). Kazdy nastaveny bit spusti TAB-stop. TAB stops jsou pro vsechny logicke radky stejne, pokud nejsou nove zmenene. Mohou se ale ve trech nasledujicich fyzickych radcich ruzne nastavit. Klavesnici je TAB maska zajistena klavesami TAB-SET a TAB-CLR. Levy okraj obrazovky LMARGN(82=$52) pusobi zaroven jako TAB-stop.
Default hodnota pro vsechny byty TABMAP je 1, coz ovlivnuje

TAB-stops ve sloupcich 7,15,23...Default hodnotu (standartni) obnovuji RESET, kazdy GR.-prikaz a OPEN"S:" nebo "E:". TAB maska pusobi take v textovem okenku.
Kdyz se ma TAB maska zmenit pomoci POKE, je nutne nastavit TAB-stops jako bity do ruznych bytu, jejich hodnoty secist a soucty ulozit (POKE).

```
byte0    byte1    byte2    byte3
00001000 01000000 00010000 10000000
=8       =66      =16      =132    atd.
```

Pro docileni TAB-stop v kazdem 5. sloupci je nutne ulozit do registru 675 decimalni hodnoty 8,66,16,132...atd. Tabelatorovy skok se vyvola stisknutim klavesy TAB, muze byt naprogramovan PRINT"ESC"TAB

690-693 $2B2-$2B5 LOGMAP

Maska pro zacatek logickych radku. Sklada se ze 4 bytu, pricemz posledni se ignoruje. Kazdy z (3*8) 24 bitu je pro radek v modu GR.0. Kdyz zacina v odpovidajicim fyzickem radku logicky radek, tak je bit nastaven.
Fyzicke radky ve vztahu k bitum bytu 690-692:

```
byte bit 7 6 5 4 3 2 1 0

690       0  1  2  3  4  5  6  7
691       8  9 10 11 12 13 14 15
692      16 17 18 19 20 21 22 23
```

694 $2B6 INVFLG

Indikator pro invertovane znaky. Inicializovan na 0. Pri stisknuti klavesy VIDEO-INV. zde bude nastaven bit 7(=128).
Ridici program displeje spojuje hodnoty znaku zadanych z klavesnice s hodnotami v tomto registru pomoci logickeho OR. Pokud INVFLG sejme hodnotu 0 nebo 128, neni klavesa stisknuta, protoze cela sada znaku obsahuje jen 128 znaku. Souctem 128 s hodnotou znaku se vyvola stejny, ale inverzni znak. Do tohoto registru je mozne ulozit uplne jine hodnoty(=bit vzory) a tim invertovat bity 0-6 vyvolanych znaku. Zmeni se tim decimalni hodnota a objevi se jiny znak. Tedy ne ten, ktery byl zadan klavesou, ale ten se zmenenou hodnotou. Pritom je treba dat pozor, aby se invertovani vztahovalo na kod klavesnice a ne na ATASCII-hodnotu. Bit-vzor klavesnicoveho kodu volaneho znaku je zmenen v INVFLG pomoci logicke funkce "OR". Takto zmenena hodnota se pak prevede na odpovidajici ATASCII znak a zobrazi se na obrazovce.
INVFLG pracuje jen pro prime zadani. Pritom musi byt registr pred zadanim popsan.
Program pro tajne pismo:

```
1     REM  ADR694
2     REM  ***************
3     REM  * TAJNE PISMO *
4     REM  ***************
```

```
10    DIM T$(1):?CHR$(125)
20    TRAP 20
30    ?:?"DEC. HODNOTA PRO INVFLG:"
40    ?:?"(OD 1 DO 127)":?
50    INPUT J
60    IF J<1 THEN 20
70    IF J>127 THEN 20
80    POKE 694,J
90    TRAP 90
100   ?:?"NAPIS LIBOVOLNY ZNAK":?:?
110   INPUT T$
```

10- DIM T$ je libovolne, pri behu programu nepusobi
30-70-J prevezme dec. hodnotu, ta se bude ukladat do INVFLG. 128 vyvola jen negativni zobrazeni stejneho znaku. Hodnoty >128 pusobi stejne, pouze bit 7 je nebo neni nastaven. Nastavovani bitu 7 nasleduje po kazdem stisknuti klavesy INVERSE.
80- J se ulozi do INVFLG
110- zde lze libovolne zadavat

695 $2B7 FILFLG

FILL-indikator pro prikaz DRAWTO. Probiha-li DRAWTO, ma registr 0; je-li to FILL(prikaz XIO18), je zde nenulova hodnota.

696-698 $2B8-$2BA TMPROW/COL

Docasny registr pro ROWCRS a COLCRS

699 $2BB SCRFLG

Indikator pro radkove rolovani. Obsahuje(pocita)pocet radku -1, ktery byl zhasnut na hornim okraji obrazovky. Protoze jeden logicky radek obsahuje az 3 fyzicke radky, muze SCRFLG mit hodnotu 0-2.
Kdyz se roluje textove okenko, je to stejne, jakoby se cela obrazovka GR.0 posunula o 800 bytu nahoru. Muze to vest k prepsani dat napr. PM-gr.-dat, rady znaku nebo jinych, ulozenych nahoru od RAMTOPu. V nejistem pripade je lepe v miste, kde by se mohlo rolovani vyskytnout, drzet 800 volnych bytu.

700,701 $2BC$2BD ////

Docasny registr pri behu FILL.

702 $2BE SHFLOK

Indikator pro klavesy SHIFT a CONTROL. Bit 7 nastaven=stisknut CTRL, kontrolni kody a pseudograficke znaky dodany. Je-li bit 7 nenastaven, lze zobrazit pismena(velka i mala). Bit 6 nastaven=stisknut SHIFT. Protoze SHFLOK je inicializovano na 64, je klavesa SHIFT v normalnim stavu stale stisknuta. Bit 6

se nastavi na 0 klavesou CAPS; v tomto rezimu je nutne k napsani velkeho pismene stisknout SHIFT.

703 $2BF BOTSCR

Ukazuje pocet moznych textovych radku pro PRINT. 24 je normalni hodnota pro GR.0; 4 je hodnota pro textove okenko; 0 je hodnota pri vsech graf. modech bez textoveho okenka. Barevne mody znaku(GR.1,2,12,13) jsou zde vylouceny (uzavreny). Manipulacni program displaye prezkousi, zda je v tomto registru vyvolana delena obrazovka (GR.+text). Tim je take mozne textove okenko vyvolat v GR.0 (POKE 703,4). Hornich 20 radku je pak popsano PRINT prikazy jako graficke okenko (PRINT#6). Tento horni dil take zustava, zatimco textove okenko roluje. Tato technika se pouziva pri DOS menu.

704 $2C0 PCOLR0

Barvovy registr pro hrace 0 a strelu 0. Barvove registry PCOLR0-3 budou cteny v PM grafice a GTIA modu GR.10. BASIC prikaz SETCOLOR nedosahne registru 704-705. Barvove hodnoty sem mohou byt ulozeny jen pomoci POKE. ATARI muze zobrazit 16 barevnych tonu. Dodatecne lze nastavit jeste 0=tmava az 15=bila ruznych stupnu jasu. Z toho dostaneme celkem 256 ruznych barevnych(jasovych) hodnot z nichz je mozne podle GR. modu zobrazit 1-16 zaroven, aniz by se pouzilo DLI. Barvova hodnota se vypocita podle vzorce:
barvova hodnota=barevny ton*16 + stupen jasu
To znamena, ze barevny ton(od 0=0000 do 15=1111) obsadi horni nibble a stupen jasu(tez od 0=0000 do 15=1111) obsadi dolni nibble v barvovem registru. Protoze bit 0 v barvovem registru neni cten, je ve skutecnosti k dispozici jenom 8 stupnu jasu.(jen prime hodnoty). Pouze v GTIA modu GR.9 se ukaze vsech 16 stupnu jasu jednoho barevneho tonu. Kdyz je registr GPRIOR(623=$2CF)nastaven bit 5 (treti barva pri prekryti), vyda se barevna hodnota pro tento prekryty barevny ve spojeni s logickym OR:
prvni bar. hodnota 0100 = staroruzova = dec. 52 = bin. 0011

druha bar. hodnota 0010 = zelena = dec. 226 = bin. 1110

OR funguje = 0110 = dec.246 = bin. 1111

Propojeni logickym OR vede k tomu, ze barevny ton v prekryte oblasti ma vzdy vyssi barvovou hodnotu nez obe jednotlive barvy, tzn. barevny ton s vyssim cislem a vetsi stupen jasu.

705-707 $2C1-$2C3 PCOLR1-3

Nasledujicich 5 barvovych registru se cte v ruznych graf. modech. Popisuji se prikazem BASICu SETCOLOR, jehoz format je SETCOLOR r,f,h. r- prebira hodnoty 0-4 a vztahuje se k barvovym registrum COLOR0(708=$2C4) az COLOR4(712=$2C8).
f- prebira hodnoty 0-15 a vztahuje se k 16 ti standartnim

[illegible] ost

Prikaz SETCOLOR je nejzbytecnejsi prikaz ATARI-BASICu, protoze se pise hure nez prikaz POKE, ktery pusobi stejne. Protoze se hodnoty f a h musi vyhledavat experimentalne, muze se take hned napsat do registru barevna hodnota(f*16+h). Kdyz se ma definovat vice barvovych hodnot, muzete pomoci POKE v BASIC programu usetrit pametove misto.

709-712 $2C5-$2C8 COLOR1-4

Barvovy registr 1-4. V ruznych grafickych modech prejimaji registry stridave ulohy a vyvolavaji se prikazy COLOR. Prehlednou tabulku najdete v priloze k obrazove pameti. Pri zapnuti se implicitne nastavi hodnoty zadane vyrobcem.

registr barev.	hodnota	bar.ton	svetlost
708			
COLOR0	40	2	9
709			
COLOR1	202	12	10
710			
COLOR2	148	9	4
711			
COLOR3	70	4	6
712			
COLOR4	0	0	0

Nasledujici program dava priklad moznosti:

```
1     REM COLPOK
2     REM ******************
3     REM * BAREVNE ZMENY *
4     REM ******************
10    GR.10
20    FOR J=1 TO 8
30    POKE 704+J, J*4
40    NEXT J
50    FOR C=0 TO 7
60    COLOR C
70    FOR I=0 TO 7
80    PLOT C*8+I,0:DRAWTO C*8+I, 191
90    NEXT I
100   NEXT C
200   A=PEEK(705)
210   FOR J=0 TO 6
220   POKE 705+J,PEEK(706+J)
230   REM  FOR W=0 TO 200:NEXT W
240   NEXT J
250   POKE 712,A
260   GOTO 200
```

Jeste sirsi vyber pouziva GR.10, ktery ma vedle barvove hodnoty pro pozadi jeste 8 dalsich volne definovatelnych tonu. V tomto programu rotuji barvove hodnoty v registrech 705-712. 704 znamena barvu pozadi a je nastaven na 0, coz znamena cerne.

```
20-40-  FOR-NEXT zapisuje do registru 705(704+1) az 712(704+8)
        hodnoty od 4(1*4) do 32(8*4). Registr 704 nemusi byt
        popsan, protoze jeho default hodnota 0 odpovida zadane
        barevne hodnote.  Kazdemu registru lze priradit
        libovolnou hodnotu.  Pro nas efekt je ale lepe
        odstupnovat barvy jasem.
50-100- plni obrazovku sirokymi svislymi barevnymi pruhy v
        poradi barevnych registru
200-    A prijme bar. hodnotu registru 705
210-    pak se vymeni bar. hodnoty v sedmi registrech(J=0 TO 6)
220-    zatimco je do predchoziho registru zapsana hodnota
        nasledujiciho
250-    na konci dostane posledni registr(712) hodnotu
        prvniho(705), ktery je obsazen v A.
```

Jestlize nechate program takto behat, vznikne dojem rotujiciho bubnu. Rychlost otaceni lze menit odstranenim REM v radku 230. Cim vyse je definovan citac smycek W, tim pomaleji se bude buben otacet.
Pusobivy efekt zpusobi, jestlize se barevna hodnota v registru rychle meni.

```
260     X=X-2:IFX<0 THEN X=254
270     POKE 704,X
280     GOTO 200
```

Pripojte tyto tri radky k listingu a pozadi(704) zacne barevne blikat. Chcete-li videt, jak rychle lze menit bar. hodnoty pomoci POKE, prepiste skokovy cil v radku 280 z 200 na 260.

729 $2D9 KEYDEL

Urcuje dobu do nastaveni opakovaci funkce klavesnice(repeat). Default hodnota je 40. Se vzrustem hodnot 1-255 se zpozdeni zvetsuje. Pri 1 je tak kratke, ze stisknutim klavesy nedokazeme napsat jeden znak (jen u XL).

730 $2DA KEYREP

Kdyz se po pruchodu KEYDEL nastavi repeat, urci KEYREP opakovaci frekvenci.Default hodnota je 5. Cim mensi je hodnota, tim rychlejsi je opakovani. Je-li zde 0, je mozne jen jedno opakovani (jen u XL).

731 $2DB NOCLIC

Ukazatel signalu z klavesnice. Nula signal potlaci, jina hodnota umozni.

732 $2DC HLPFLG

Indikator klavesy HELP. Default hodnota je 0. Stisknutim klavesy HELP se nastavi bity 0 a 4(=17). Stisknutim HELP a SHIFT se zaroven nastavi bit 6, registr pak ma hodnotu 81.

HELP a CONTROL nastavi dodatecne bit 7(=145). SHIFT a CONTROL zpusobi neucinne HELP. Nastavene bity se pri uvolneni klavesy nikdy nevraci zpet na 0.

740 $2E4 RAMSIZ

Velikost RAM, ktera je k dispozici. Obsahuje jen HI byte, dava tedy velikost RAM ve strankach. Obsahuje stejnou hodnotu jako RAMTOP (106=$6A).

741,742 $2E5,$2E6 MEMTOP

Ukazatel na horni hranici volne RAM. Tato hodnota se obnovi pri zapnuti nebo RESET, pri kazdem GR. prikazu nebo pri kazdem OPEN vztahujicim se na displej.

743,744 $2E7,$2E8 MEMLO

Ukazatel na dolni hranici volne RAM. Je zmenen napr. pomoci DOS, ktery je ulozen v dolnim pametovem rozsahu. MEMLO ukazuje na prvni volnou pametovou bunku, kterou lze obsadit uzivatelskym programem.

750,751 $2EE,$2EF CBAUD

Obsahuje udaj (rychlost v baudech, tj. v bitech za sekundu) pro kazetovy magnetofon o rychlosti nahravani. Je nastaven na 1484, coz odpovida rychlosti 600 bitu/s. Rychlost je upravovana SIO. Kazdy kazetovy zaznam zacina sekvenci (vzorem) bit zapnut, bit vypnut, na ktery pak ridici program kazety koriguje baudovou rychlost.

752 $2F0 CRSINH

Ukazatel na potlaceni kurzoru. Je nastaven na 0 po zapnuti, RESET, BREAK a OPEN"S:" nebo "E:" coz ovlivnuje viditelnost kurzoru (=inverzni prazdny znak). Kazda nenulova hodnota cini pri dalsim pohybu kurzor neviditelnym.

753 $2F1 KEYDEL

Citac pro odstraneni zakmitavani tlacitka. Ma hodnotu 0, kdyz neni stisknuta zadna klavesa. Kdyz se stiskne libovolna klavesa, ulozi se sem hodnota 3, ktera klesa o 1 pri kazdem VBLANK. Nejblizsi zadani z klavesnice je mozne az pote, co se zde dosahne 0. KEYDEL tedy urcuje casovy odstup mezi dvema zadanimi.

754 $2F2 CH1

Obsahuje hodnotu drive stisknute klavesy, kterou sem zadal CH(764=$2FC)

755 $2F3 CHACT

Registr pro znazorneni znaku. Je-li 0, budou vsechny inverzni znaky zobrazeny jako normalni. Kurzor bude neviditelny. Jednicka zpusobi, ze vsechny inverzni budou vydany jako prazdne znaky. Kurzor neviditelny, ale znak pod nim zmizi. Default hodnota 2. Trojka zobrazi vsechny inverze jako inverzni prazdne znaky. Po nastaveni bitu 2(=4) se zobrazi vsechny znaky vzhuru nohama.

```
1     REM  ADR755
2     REM  ******************
3     REM  * OTOCENE ZNAKY *
4     REM  ******************
10    GR.0: POKE 703,4
20    ?"321 321 CBA CBA"
30    FOR J=0 TO 255
40    POKE 755,J
50    ?  6;J;"  ";
60    OPEN #1,4,0,"K:"
80    CLOSE #1
70    GET#1,D
90    NEXT J
```

Listing obsahuje inverzni znaky, ktere jsou pro pochopeni nutne.

```
10-     vGR.0 se pomoci POKE zadava textove okenko
20-     v grafickem okenku se vytisknou normalni i inverzni
        znaky
30-     FOR-NEXT prochazi vsechny mozne decimalni hodnoty,
        ktere mohou byt v tomto bitu nastaveny
40-     do 755 se pomoci POKE ulozi hodnota
50-     ulozena hodnota se ukaze v textovem okenku
60-80-  az uzivatel stiskne nejakou klavesu, bude program
        pokracovat tim, ze ulozi do 755 dalsi hodnotu
```

Jak vidite, pusobi v tomto registru pouze bity 0-2. Pri osmi moznostech lze take programovat ruzne blikani prepinanim dvou hodnot v CHACT. Nasledujici program predvadi vsech 64 kombinaci:

```
1     REM  ADR755B
2     REM  ***********
3     REM  * BLIKANI *
4     REM  ***********
10    GR.0:POKE 703,4
20    POSITION 5,12
30    ?#6;"BLIK BLIK"
40    FOR I=0 TO 7
50    FOR J=0 TO7
60    ?I; "/"; J:?
70    POKE 755,I
80    FOR W=0 TO 100: NEXT W
90    POKE 755,J
100   FOR W=0 TO 100:NEXT W
110   IF PEEK(764)=255 THEN 70
120   OPEN #1,4,0,"K:"
130   GET #1,D
```

```
140   CLOSE #1
150   NEXT J
160   NEXT I
```

10-30- do stredu GR.0 bude tisten text
40-50- smycky probehnou vsech 8*8 kombinaci v textovem okenku mezi nimiz lze v CHACT prepinat
60- vyda ke kontrole v textovem okenku okamzitou kombinaci
80-100- zde lze menit frekvenci blikani
110-130-stisknutim libovolne klavesy zacne blikat nasledujici kombinace.

756 $2F4 CHBAS

Ukazatel na pocatecni adresu standardni sady znaku ATARI. CHBAS je pouze HI-bytovy ukazatel. Abychom dostali skutecnou adresu, musime zde nalezenou hodnotu nasobit 256. Default hodnota je 224. Sada znaku obsahuje 128 znaku. Negativni znaky (ATASCII 128-255) nevyzaduji specialni data. Vznikaji invertovanim normalniho znaku. Ukazatelem pro inverzni zobrazeni je bit 7(=128). ATASCII hodnota inverzniho znaku je o 128 vyssi nez pro normalni znak.
Ve znakovem modu GR.1 a 2 je k dispozici jen polovina sady znaku, tzn. vetne znaky, cislice a velka pismena. Presazenim CHBAS na zacatek zadni casti sady znaku lze zobrazit i mala pismena. Nelze tedy zaroven zobrazit velka i mala pismena, aniz by se sada znaku nepremistila. Ukazatel se premisti prikazem POKE 756,226.

Prirozene se u standartni sady znaku jedna o americkou sadu znaku. U XL lze nastavit evropskou sadu znaku (nemeckou) pomoci POKE 756,204.

760- $2F8 ROWINC

Znamenko(+1 nebo -1) k DELTAR(118=$76). (:121=$79)

761- $2F9 COLINC

Znamenko k DELTAC (119,120=$77,$78). (:122=$80)

762- $2FA CHAR

Obsahuje interni kod naposledy psaneho nebo cteneho znaku. Protoze BASIC je zpravidla na cteni v tomto registru prilis pomaly, prinasi PEEK(762) vetsinou jen hodnotu 128 (viditelny kurzor = negativni prazdny znak) nebo 0 (neviditelny kurzor = prazdny znak).80

763- $2FB ATACHR

Obsahuje ATASCII hodnotu naposledy psaneho nabo cteneho znaku a pouziva se pri prevodu z ATASCII do interniho kodu. Pri grafickem provozu se zde uklada COLOR hodnota grafickeho bodu a

pri prikazech XIO 17 **(DRAW)** a XIO 18 (FILL) barvova hodnota tazene linie.

764- $2FC CH

Obsahuje klavesnicovy kod naposledy stisknute klavesy. Zde lze cist pomoci PEEK(764) udaje z klavesnice, aniz by se muselo zadavat RETURN. Registr obsahuje 255 , pokud neni stisknuta zadna klavesa. Klavesnicovy kod ostatnich klaves obsahuje nasledujici tabulka. Abychom ziskali zadany kod znaku, musime secist decimalni hodnoty na levem a hornim kraji:

	0	1	2	3	4	5	6	7
00	L	J	;			K	+	*
08	O		P	U	RET	I	-	=
16	V		C			B	X	Z
24	4		3	6	ESC	5	2	1
32	,	SP	.	N		M	/	INV
40	R		E	Y	TAB	T	W	Q
48	9		O	7	BACK	8	(	)

Tyto znaky obsazuji bity 0 - 5 . Nezavisle na tom nastavi soucasne stisknuti klavesy s SHIFT bit 6 a s CONTROL bit 7 . Soucasne stisknuti SHIFT a CONTROL je bezvyznamne a OS ignorovano. Klavesnicovy kod znaku je zmena znaku v tabulce klavesnicovych definici, ktera se pouziva k prevodu klavesnicoveho kodu do ATASCII. Uzivatel si muze sestavit vlastni tabulku definici a kazde klavese libovolne priradit nove ATASCII hodnoty. Tabulka vsak musi byt vzdy rozdelena na tri dily.Prvni tretina do 64 bytu se dosahne pri malych pismenech, druha tretina 64 bytu pri tlacitku + SHIFT a treti 64 bytu pri tlacitku + CONTROL. Po ulozeni tabulky do pameti je nutne premistit ukazatel KEYDEF (121,122=$79,$7A) na jeji zacatek.

Tlacitka RESET, BREAK, SHIFT, CONTROL, funkcni tlacika OPTION, SELECT, START, HELP a kombinace CONTROL + "1" tato tabulka nezpracovava a proto zde vubec nemusi byt obsazeny. Pouziti CH najdete v programu ADR. 7550 - radek 110.

765- $2FD FILDAT

COLOR hodnota pro vyplneni rozsahu pri XIO 18 (FILL) . Pouziti FILL prikazu ukazuje nasledujici program:

```
1     REM  ****************
2     REM  * XIO18 - FILL *
3     REM  ****************
10    GR.31
```

```
20    POKE 708,4:POKE709,50:POKE710,144
30    POKE 765,1
40    COLOR 2
45    REM PLOT 80,80
50    PLOT 139,95
60    DRAWTO 109,0
70    DRAWTO 49,0
80    POSITION 19,95
90    XIO 18,#6,0,0,"S:"
95    GOTO 95
100   PLOT 109,191
110   DRAWTO 139,96:REM COLOR 1
120   DRAWTO 19,96
130   POSITION 49,191
140   XIO 18, 6,0,0,"S:"
150   GOTO 150
160   COLOR 3
170   PLOT 159,191
180   DRAWTO 159,0
190   DRAWTO 0,0
200   POSITION 0,191
210   POKE 765,2
220   XIO 18,#6,0,0,"S:"
230   GOTO 230
240   PLOT 159,191:REM COLOR 2
250   DRAWTO 159,0
260   DRAWTO 110,0
270   COLOR 2:POSITION 140,95
280   XIO 18,#6,0,0,"S:"
290   PLOT 159,96
300   DRAWTO 140,96
310   POSITION 110,191
320   XIO 18,#6,0,0,"S:"
330   GOTO 330
```

10- je vyvolan GR.15 bez textoveho okenka(+16)

20- barvove hodnoty pro COLOR 1,2,3. COLOR 0, pozadi, registr 712 obsahuje svou default hodnotu 0(=cerna).

30- ve FILLDAT je pro XIO 18 urcena COLOR hodnota 1

40- pro nasledujici PLOTovani je vyvolana barva 2

50-80- FILL funkce potrebuje nasledujici udaje: PLOTovanou linii jako prave ohraniceni, PLOTovanou linii jako horni hranu; tyto 2 linie 3 rohove body FILLovane plochy: vpravo dole vpravo nahore a vlevo nahore: jako posledni orientace se pouzije bod vlevo dole; zadany prikaz POSITION. Prikazem FILL lze zasadne vyplnit pouze ctyrrohove plochy, ovsem v libovolne forme. Jedina dodatecna podminka je, aby levy horni roh lezel vyse nez pravy horni roh, protoze

90- prikaz FILL udela nasledujici: od leveho horniho rohu se k bodu urcenemu POSITION PLOTuje linie v prave vyvolane COLOR hodnote. Od kazdeho jednotliveho prave plotovaneho bodu se tahne linie(s COLOR hodnotou ulozenou ve FILDAT) doprava, az prave PLOTovany bod dosahne praveho ohraniceni. Potom se PLOTuje dalsi bod zase leve strany ctyruhelnika, opet se dotahne FILL linie doprava atd.

K demonstraci zavedte do listingu radek 45 a zadejte

RUN. Vidite, ze kdyz rutina FILL dosahne radku 80, vyplni se pouze do sloupce 79, a za 80 zustava radek ve svem starem COLOR stavu.

95- Odstrante tento radek a radek 45 take. Pak se pokracuje.

100 - 140- Sestiuhelnik se musi rozdelit na dve trojuhelniky. Zde nasleduje dolni polovina.

110- Kdyz nechate program bezet, uvidite, jak se nejdrive PLOTuje levy a horni kraj ctyruhelnika. Behem FILL vznikne take leva hrana. Smerem dolu je ctyruhelnik ohranicen hranou FILL plochy. Pokud si prejete jeste barevny ramecek, je nutne jeste dodatecne PLOTovat dolni hranu v zadane barve. Na druhe strane vidite, ze horni hrana dolni poloviny sestiuhelnika je vydana pro PLOTovani v platne barve, a za jeho delici linie je vedena vodorovne a probiha stredem sestiuhelnika. Pokud ji chcete odstranit, zruste v radku 110 REM, takze bude pusobit prikaz COLOR. Je ostatne mozne urcit tutez COLOR hodnotu pro PLOT i FILL.

150- Po vyzkouseni techto manipulaci tento radek odstrante, abyste zpristupnili ostatni programove casti.

160 - 220- Zde se PLOTuji s COLOR 3 vnejsi hrany grafickeho okenka jako ohraniceni pro prikaz FILL. Kdyz ale vyjde prikaz XIO 18 s COLOR 2 , bude brat prave urcenou levou stranu ohraniceneho pravouhelnika jako prave ohraniceni. Nevyplni se tedy cely zbytek obrazovky, ale pouze plocha od leveho kraje k sestiuhleniku.

230- Po prohlednuti odstrante tento zachytny radek

240 - 280- Aby bylo mozne FILLovat jeste zbylou plochu vpravo od sestiuhelnika, je nutne nasledujici:neni totiz mozne vyplnit tuto plochu nejakym prikazem. Je-li vyPLOTovane leve a horni ohraniceni plochy(kraj obrazovky) a je-li pak kurzor pomoci POSITION nastaven na vnejsi misto vlevo dole, lze pak tahnout pouze levou hranu. Tato hrana lezi uvnitr sestiuhelnika. Prikaz FILL by mel zacinat uvnitr sestiuhelnika a brat prave hrany sestiuhelnika jako prave ohraniceni. Ale jiz jednou FILLovana plocha nemuze byt znovu pomoci FILL zmenena. Prava zbyvajici plocha se tedy musi vyplnit ve 2 dilech. Radky 240-280 vyplni horni polovinu. Rusici levou hranu lze barevne prizpusobit odstranenim REM v radku 240. Tazeni praveho ohraniceni (kraje obrazovky) je zbytecne, protoze v radku 100 a 110 jsme podnikli neuspesny pokus o vyplneni (FILL) cele plochy najednou. V radcich 290-300 to lze s uspechem zaridit.

766- $2FE DSPFLG

Displejovy indikator pro ovladani ridicich znaku. Ridici znaky jako pohyb kurzoru, zhaseni obrazovky, INSERT, DELETE, nebo TAB se v BASICu mohou uvest dvema zpusoby. Jednim je PRINT CHR$(n), napr. pro n=125 se vycisti obrazovka. Tato forma ma tu vyhodu, ze ji kazda pripojena tiskarna bez problemu vytiskne.
Druha moznost spociva v naprogramovani zadane formy pomoci prislusnych klaves; tj. formou PRINT"(ESCAPE) (CLEAR)" , pricemz udaje v zavorkach znamenaji , ze pri zadani musite stisknout odpovidajici klavesu. Pri zadani v teto forme se na obrazovce objevi graficky symbol , v tomto pripade doleva

zakrivena sipka. Pro neskoleneho programatora je to jednodussi, protoze si nemusi pamatovat ATASCII hodnoty prislusnych funkci. Stisknuti ESCAPE pusobi jen pro nasledujici klavesu. Chcete-li zadat vice ridicich znaku najednou, musite pred kazdym stisknout ESCAPE. Kdyz pak svuj program vytisknete, asi tuto metodu opustite, protoze tiskarna interpretuje znaky , ktere nejsou urceny ATASCII normou, po svem. Take kody vetsi nez 128 (napr.TAB) vam kazda tiskarna vytiskne jinak.

Totez plati pro pseudograficke znaky hodnot 0-31. To jsou totiz pro vsechny tiskarny nejdulezitejsi znaky, jimiz se zde urcuji skoky TAB, posun radku a formularu, navrat voziku a zvonek. Firma ATARI dosud nema pripravenu tiskarnu, ktera by vytiskla celou sadu ATASCII znaku. Podobne zarizeni dosud neni na trhu, proto se doporucuje nahradit v programech, ktere maji byt vytisteny, vsechny tyto znaky odpovidajicimi CHR$ prikazy. Pokud je v DSPFLG nenulova hodnota, zobrazi se na obrazovce odpovidajici ridici znak pomoci graficke ho ekvivalentu, tedy tak, jako by bylo stisknuto ESC. Normalni stav tohoto registru je nula.

767- $2FF SSFLAG

Start/stop ukazatel pro vystup na obrazovku. Je-li zde nula, pokracuje vystup nerusene dal. S hodnotou 255 (inverzne 00000000) je vydej prerusen a system ceka, az zde bude opet nastavena nula. SSFLAG se prepina pomoci soucasneho stisknuti CONTROL a "1".

768- $300 DDEVIC

Identifikacni kod periferie.

Zarizeni: 49 $31 disketa
64 $40 tiskarna
80 $50 linka RS 232
96 $60 magnetofon

769- $301 DUNIT

Cislo periferie (1-8 u disket).

770- $302 DCOMND

Prikazovy byte . Obsahuje kod operace, ktera ma byt provedena.

!	33	$21	formatovani sd
"	34	$22	formatovani dd
N	78	$4E	cist konfiguracni blok
O	79	$4F	zapsat konfiguracni blok
P	80	$50	zapsat bez verifikace
R	82	$52	cist
S	83	$53	cist status
W	87	$57	zapsat s verifikaci

771- $303 DSTATS

Pred SIO ! 128 $80 zapis , S 64 $40 cteni.Po SIO stejne jako registr y vraci stav periferie.1 = operace v poradku , >127 kod chyby.

772,773- $304,$305 DBUFLO/HI

Adresa datoveho bufferu (u magnetofonu 1021 $3FD , u tiskarny 960 $3C0 , u diskety 6748 $1AEC).

774- $306 DTIMLO

Casova prodleva (timeout) ridiciho programu periferie.

775- $307 DUNUSE

Volny byte.

776,777- $308,$309 DBYTLO/HI

Pocet bytu v bufferu, na ktery ukazuje DBUFLO/HI.

778,779- $30A,$30B DAUX1/2

Pomocne informace (pri disketovych operacich cislo sektoru,u magnetofonu 0,0 = normalni mezera ; 0,128 = kratka mezera).

780,781- $30C,$30D TIMER 1

Pocatecni hodnota citace baudove rychlosti. Slouzi pro nastaveni CBAUD

783- $30F CASFLG

Pri nahravani na kazetu je nenulovy.

784,785- $310,$311 TIMER 2

Koncova hodnota citace 2. TIMER 1 a TIMER 2 se pouzivaji k zajisteni baudove rychlosti kazetovych operaci. Porovnavaji standardni hodnotu s bitovou kombinaci na zacatku kazetoveho souboru. Rozdil hodnot citacu se vyuziva k vyhledani korekce, pomoci ktere se pak zapise baudova rychlost na CBAUDLO/HI (750,751=$2FE,$2FF).

791- $317 TIMFLG

SIO navesti casove prodlevy.

792- $318 STACKP

SIO ukazatel zasobniku.

793- $319 TSTAT

Docasny registr pro SIO stav informace.

794-831- $31A-$33F HATABS

Tabulka adres ridicich programu periferii.
Lze vytvorit az 38 tribytovych zapisu.
Prvni byte obsahuje jmeno periferie (C,D,E,K,P,S,R) v ATASCII kodu.
Byty 2 a 3 obsahuji LO a HI pocatecni adresu ridiciho programu.
Neobsazene byty jsou na nule.

832-847- $340-$34F IOCB0

I/O kontrolni blok 0. Pouziva se pro obrazovy editor ("E:") normalnim zpusobem. V GR.modech je otevren kanal 0 k textovemu okenku. Pokud neni po ruce zadne textove okenko a kanal 0 je otevren, prejde cela obrazovka na vystupni mod (GR.0). To se stane napr. tehdy, kdyz je zpracovavan graficky program bez textoveho okenka , aby se vydalo READY.
BASIC prikazy NEW a RUN uzaviraji vsechny kanaly mimo 0. Otevrenim kanalu pro "S:" nebo "E:" se cela obrazova pamet vymaze. Ma-li program obsahovat uzivatelem zadany vystup dat na obrazovku nebo tiskarnu , je mozne zamezit castemu zadavani vystupnich prikazu (PRINT,LPRINT) . K tomu staci zmenit v adresach 838 (=$340) LO a 839 (=$34F) HI . Normalne obsahuje tento ukazatel hodnoty 163 a 246. Kdyz se ulozi do LO 166 a do HI 238, vyda se na tiskarnu totez, co prave jde na editor.

842- $34A ICAX1

Tato adresa ma zvlastni vyznam, protoze umoznuje cteni z obrazovky. Normalne je zde hodnota 12. Kdyz na adresu 842 ulozi 13, zapne se RETURN klavesovy mod, ktery umoznuje cteni z obrazovky.
A to pracuje takto:

```
1     REM  ADR842
2     REM  **********************
3     REM  * CTENI Z OBRAZOVKY *
4     REM  **********************
10    GR.0:POSITION 2,4
20    ? 1000
30    FOR W=0 TO 500: NEXT W
40    ? 2000
50    FOR W=0 TO 500: NEXT W
60    ? 3000
70    FOR W=0 TO 500: NEXT W
80    ?"4000 REM * TENTO RADEK SE PRI CTENI Z OBRAZOVKY
```

```
      VESTAVI DO  PROGRAMU JAKO BASIC RADEK *"
90    FOR W=0 TO 500: NEXT W
100   ? "CONT"
110   FOR W=0 TO 500: NEXT W
120   POSITION 2,0
130   FOR W=0 TO 500: NEXT W
140   POKE 842,13: STOP
150   FOR W=0 TO 500: NEXT W
160   POKE 842,12
170   FOR W=0 TO 500: NEXT W
1000  REM TENTO RADEK BUDE VYPUSTEN
2000  REM   "      "    "       "
3000  REM   "      "    "       "
4000  REM TENTO RADEK BUDE PREPSAN
```

10- GR. prikaz vymaze obrazovku, protoze z ni ma byt cteno pouze to co tam program napise. Je dulezite zacit s popisovanim obrazovky co nejniz, aby se systemovym hlasenim STOPPED neprepsalo napsane.

20- Na obrazovce je napsano cislo 1000. Chvili po precteni vezme BASIC toto cislo jako cislo radku a protoze tam neni zadny prikaz, bude radek 1000 vymazan z pameti.

30- cekaci smycka pro lepsi opticke sledovani behu programu. Pri skutecnem pouziti se vynecha.

80- je mozne vytisknout nektery programovy radek, ktery se ctenim z obrazovky zaradi do programu.

100- zaverem se musi napsat CONT, aby po precteni bezel program dal.

120- pote, co se na obrazovce vytiskne vse, co ma byt cteno, je nutne pod to umistit na obrazovku neviditelny kurzor.

140- zde se uklada 13 pro cteni a program se zastavi.

160- po precteni CONT z obrazovky bezi program dal a ICAX1 se zpetne nastavi na vydej na obrazovku.

Kdyz ted program spustite, uvidite, jak budou napsana cisla 1000, 2000, 3000, 4000. Pak kurzor skoci na sloupec 2 v radku 0 a vyda "STOPPED IN LINE 140". Cteci proces probiha a skonci, kdyz dole programator oznami READY. A nyni zadejte LIST !

848-863 $350-$35F IOCB1

I/O kontrolni blok 1.

864-879 $360-$36F IOCB2

I/O kontrolni blok 2.

880-895 $370-$37F IOCB3

I/O kontrolni blok 3.

896-911 $380-$38F IOCB4

I/O kontrolni blok 4.

912-927 $390-$39F IOCB5

I/O kontrolni blok 5.

928-943 $3A9-$3AF IOCB6

I/O kontrolni blok 6.

GR.prikaz otevre kanal 6 k obrazovce ("S:") . Proto nelze pouzit kanal 6, kdyz byl GR.0 opusten, popr. se nejdriv musi kanal 6 zavrit (CLOSE). Pak uz nelze vyvolat prikazy PLOT, DRAWTO, nebo LOCATE. Prikazy PRINT na graficke okenko v modech 1,2,12,13 a 0 musi byt proto take rizeny na kanal 6.

944-959 $3B0-$3BF IOCB7

Prikaz LPRINT pouziva kanal 7. Pokud se tento kanal otevre pro jinou periferii a zada se LPRINT, vyvola se chybove hlaseni. Take prikaz LIST pouziva kanal 7 a po pouziti kanal uzavre.

960-999 $3C0-$3E7 PRNBUF

Buffer tiskarny.Zde se ukladaji data pred vystupem na tiskarnu po EOL.

1001 $3E9 STRTFLG

Tento ukazatel obsahuje nenulovou hodnotu, jestlize pri zapnuti pocitace tiskneme START.

1021-1151 $3FD-$4FD CASBUF

Kazetovy buffer.

1152-1791 $480-$6FF /////

Tento rozsah je k dispozici jako uzivatelska RAM pro programy ve strojovem kodu do 640 bytu. Celkem pouziva floating-point rutina adresy 1406-1535 (=$57E-$5FF). Skutecne jiste pametove misto v tomto dolnim rozsahu skyta jen stranka 6 (1536-1791=$600-$6FF).

1792-5377 $700-$1501 FMS

FMS (file management system) je soucasti DOS. Stanovi spojeni mezi BASIC nebo DUP a disketovou jednotkou.

5440- $1540- DUP

DUP (disc utilities package) ovlada ruzne DOS funkce.
Pouzitelny rozsah pameti se meni. Dosahuje az k adrese urcene MEMLO, max. do 13062 (=$3306). Kdyz jsou z DOS menu vyvolany DUP utilities, muze dojit k prepsani dolnich uzivatelskych RAM rozsahu.

Je-li na disketu narizeno MEM.SAV, zaznamenaji se data tohoto rozsahu na disketu a po opusteni DOS se tam opet ulozi. To je sice velmi pohodlne, ovsem MEM.SAV obsadi pomerne podstatnou cast disketovych sektoru a spotrebuje nejaky cas. Proto se doporucuje MEM.SAV nepouzivat a misto pres DOS provadet disketove operace s XIO prikazem.
Nasledujici DOS operace mohou byt nahrazeny:

volba D; soubor vymazat:
```
XIO 33, 1,0,0,"D:filename.ext"
```

volba E; soubor prejmenovat:
```
XIO 32, 1,0,0,"D:filename.old,filename.new"
```

volba F; soubor zajistit:
```
XIO 35, 1,0,0,"D:filename.ext"
```

volba G; soubor odjistit:
```
XIO 36, 1,0,0,"D:filename.ext"
```

volba I; disk formatovat:
```
XIO 254, 1,0,0,"D:"
```

Volba A; vystup adresare (directory) lze relativne lehce nahradit.
Potrebujete k tomu jen dva radky, ktere mohou byt na konci kazdeho programu v BASICu:

```
1     REM DIRCPRNT
2     REM ****************************
3     REM * VYTISTENI DISC-DIRECTORY *
4     REM ****************************
32766 END
32767 CLR: DIM FLN$(18): CLOSE #1:
      OPEN #1,6,0,"D:*.*": FOR FLS=0 TO
      64: INPUT #1,FLN$: ?FLN$: NEXT FLS
```

32766- oddeluje pred timto radkem stojici program v BASICu od directory programu.
32767- CLR maze DIMenzovani promenne. To je nutne, protoze se sem bude casto skakat. FLN$ se dimenzuje na 18 znaku a prebira jmeno souboru a pocet obsazenych sektoru, tzn. 18 znaku. Prikaz CLOSE je ochrana pro pripad, ze kanal cislo 1 je prave otevren. Potom se otevre kanal cislo 1. Sestka urcuje provozni zpusob kanalu- cteni obsahu z disku. "*" se oznacuje jako wild cards. Nahrazuji kazdou dalsi radu znaku. "*.*" nahrazuji kazde mozne jmeno souboru (filename) tzn. ze budou cteny vsechny soubory. Smycka FOR-NEXT projde vsechny soubory. Na jedne diskete lze zaznamenat max.64 souboru. I kdyz zustanou nektere sektory volne, nelze dalsi soubor

zaznamenat, protoze adresar je jiz plny. Jako posledni je cten pocet volnych sektoru.

Pokud se misto PRINT FLN$ zada LPRINT FLN$, lze prowest vystup na tiskarnu. Pokud disketa jeste neni naplnena 58 soubory, konci vystup hlasenim chyby (ERROR 136) .

6.0

PLAYER-MISSILE-GRAPHICS

53248-53505 $D000-$D0FF GTIA

53248- $D000 HPOSP0

(W) Zde je ulozena horizontalni poloha hrace 0. Player(hrac) a missile(strela) jsou graficke objekty. Obycejne se tyto prvky nazyvaji sprites (skritek). ATARIho hrac se na obrazovce tahne jako uzky vertikalni pruh v cele vysi. Horizontalni pozici tohoto pasu obsahuje HPOSPn.
Polohy PM objektu se vztahuji k fyzickemu rozdeleni obrazovky. Proto jsou mozne horizontalni hodnoty 0-227, ale hrac vstupuje do obrazu nejprve vlevo, kdyz ma horizontalni pozici asi 45 a vystupuje vpravo pri hodnote asi 210. Presne hodnoty kolisaji podle jednotlivych pristroju. Vertikalni pozici hrace urcuje poloha jeho dat. Kazdy PM objekt je rizen zvlastnim pametovym rozsahem (jeho organizace - viz PMBASE 54279=$D407). Hrac je vzdy siroky 1 byte, kazdy bit muze zobrazit (dle definice v prislusnem registru SIZE0 53256=$D008) dva, ctyri nebo osm obrazovych bodu, coz vede k ruznym sirkam hracu hracu na displeji. Protoze hrac obsazuje celou vysku obrazovky, zabira 128 bytu (pri dvoubunkovych bytech) nebo 256 bytu (pri jednobunkovych bytech). Nema-li hrac zabrat celou vysku obrazovky, nastavuji se odpovidajici byty na nulu. Jak jsou obrazove body v bytu organizovany, najdete v kapitole sada znaku.
Tyto a mnohe dalsi registry maji pro cteni (R) a psani (W) ruzne funkce. Po ulozeni horizontalni pozice hrace 0 nelze nasledne tuto pozici cist pomoci PEEK(53248), protoze R ukazuje na kolizi (stret) mezi strelou 0 a hracim polem. S kterym hracim polem doslo ke kolizi, ukazuje dolni nibble. Ruznymi hracimi poli se mini pixely ruznych COLOR hodnot.

bit	7	6	5	4	3	2	1	0
dec.hodnota					8	4	2	1

hraci pole nepouzito 3 2 1 0

Po stretu strely 0 s hracim polem (-barva) 2, najde PEEK(53248) ctyrku.

53249-53251 $D001-$D003 HPOSP1-3

(W) Horizontalni pozice hrace 1-3. (R) Strela 1-3/hraci pole - kolize

53252 $D004 HPOSM0

(W) Horizontalni pozice strely 0. Strely se pohybuji horizontalne stejne, jak bylo vysvetleno pro hrace. (R) Kolize hrace s hracim polem.

bit	7	6	5	4	3	2	1	0
dec.hodnota					8	4	2	1
hraci pole	nepouzito				3	2	1	0

Po stretu hrace 0 s hracim polem (-barva) 3, najde PEEK(53252) osmicku.

53253-53255 $D005-$D007 HPOSM1-3

(W) Horizontalni pozice strely 1-3 (R) Hrac 1-3/hraci pole - kolize

53256 $D008 SIZEP0

(W) Urcuje sirku hrace 0. Hrac je vzdy 1 byte (8 bitu) siroky. Kazdy jednotlivy nastaveny bit vytvori jeden pixel. SIZEP0 urcuje, kolik obrazovych bodu vytvori sirku pixelu. Nula nebo dvojka sem ulozena zpusobi, ze pixel bude siroky dva obrazove body. Jednicka vyda na displej pixel siroky ctyri obrazove body , a hrac je pak siroky 32 obrazovych bodu. Trojka definuje sirku pixelu osm, 1 byte 64 obrazovych bodu.
Vyska pixelu je nastavena v SDMCTL (559=$22F) v bitu 4. Normalne, kdyz bit 4 neni nastaven, je kazdy pixel hrace vysoky dva obrazovkove radky. Nastavenim bitu 4 na 556 lze zadat vysku jako 1 TV radek.

Poradi cisel v tabulce:
bitova kombinace
normalni sirka
dvojita sirka

10000001- (=129)

11110000000000000000000000001111
1100000000000011

01000010- (=66)

```
00001111100000000000000000111110000
00110000000001100


00100100-   (=36)

00000000011111000000001111100000000
00001100001100000


00011000-   (=24)

00000000000001111111100000000000000
00000001111000000


000010000-   (=8)

00000000000000000111100000000000000
00000000110000000


00111000-   (=56)

00000000011111111111110000000000000
00001111110000000


11101111-   (=239)

11111111111110000111111111111111111
11111100111111111
```

(Zobrazeni hrace v jednoradkovem reseni; kazda 1 odpovida jednomu obrazovemu bodu. Pri dvouradkovem reseni se musi bodovy vzor, ktery vytvari 1 byte, zobrazit na 2 obrazovkove radky. Pixel tak bude dvojnasobne vysoky.)

(R) Cten, ukazuje tento registr kolize strely 0 s hracem.

bit	7	6	5	4	3	2	1	0
dec.hodnota					8	4	2	1
hrac	nepouzito				3	2	1	0

Jestlize strela 0 najde hrace 0 , pak najde PEEK(53256) nulu.

53257-53259 $D009-$D00B SIZEP1-3

(W) Sirka hrace 1-3. (R) Strela 1-3/hrac - kolize.

53260 $D00C SIZEM

(W) Kazde 2 bity tohoto registru urcuji sirku strely:

bit	7	6	5	4	3	2	1	0

```
hodnota 128  64  32  16  8   4   2   1
strela  ---3---  --2--  --1--  --0--
```

A nasledujici pary bitu urcuji sirku strely:
00 normalni sirka, 2 obrazove body pro pixel
01 dvojita sirka, 4 obrazove body pro pixel
10 normalni sirka, 2 obrazove body pro pixel
11 ctyrnasobna sirka, 8 obrazovych bodu pro pixel

Kdyz tedy chcete znazornit strelu 3 ve ctyrnasobne sirce, musite nastavit bit 7 a 6.
Nastaveni bitu 4 da strele 2 dvojitou sirku. Jestlize maji mit strely 1 a 0 normalni sirku, bity 3-0 se nenastavuji. Tak dostaneme bitovy vzor 11010000 s hodnotou 128 + 64 + 0 + 16 + 0 + 0 + 0 + 0 = 208. Ten musite ulozit do SIZEM, abyste zmenili vypocitane sirky ruznych strel.
(R) PEEK zde muze zjistit, zda nedoslo ke stretu hrace 0 s nejakym jinym hracem.

```
bit          7  6  5  4  3  2  1  0
dec.hodnota                8  4  2  1
hrac         nepouzito     3  2  1  -
```

53261- $D00D GRAFP0

(W) Zde se muze pro hrace 0 urcit graficka forma. Je ale ucinna pouze tehdy, je-li pro hrace vypnut DMA v GRACTL (53277=$D01D), protoze se tento graficky registr naplni daty (ulozenymi v PM rozsahu pameti) pro hrace pri DMA (direct memory access = primy pristup do pameti).
Je-li DMA vypnut, muze se zde v BASICu ulozit pouze 1 byte, jehoz bit-vzor urci formu hrace v celkove vysce, tzn. ze zde vznikaji svisle linie. Kdyz se do GRAFP0 napise napr.255 , tedy vsechny bity nastaveny, pak se objevi hrac jako 8 bitu siroky pas, tzn. pri normalni sirce 16 obrazovych bodu. Pro toto ohraniceni je pouziti GRAFPn registru dost omezene.
(R) Hrac 1/hrac - kolize.

53262-53263- $D00E-$D00F GRAFP1-2

(W) Forma pro hrace 1 a 2. (R) Hrac 2/hrac - kolize.

53264 $D010 GRAFP3

(W) Forma pro hrace 3. (R) Ukazatel pro spoust joysticku 1. Bude zapsan v BASICu ve stinovem registru STRIG0 (644=$284) . Spoust joysticku 0 je na pinu 6 konektoru 1. Kdyz neni spoust stisknuta, je v bitu 0 jednicka, ktera je strelbou nastavena na nulu. Kdyz je nastaven v GRACTL (53277=$D01D) bit 2, ctou vsechny TRIGn registry nulu, dokud neni GRACTL opet nastaven. Tim lze nastavit delku strelby vsech spousti.

53265 $D011 GRAFM

(W) Forma pro vsechny strely. Pracuje jako GRAFPn . Kazdy

bitovy par se vztahuje na 1 strelu.

```
bit       7   6   5   4   3   2   1   0
strela    --3--   --2--   --1--   --0--
```

Kazdy nastaveny bit vytvori 1 pixel sirokou vertikalni linii pres celou vysku obrazovky. Lze definovat tvar pro kazdou strelu zvlast, pricemz jsou nastaveny odpovidajici bity a decimalni hodnota teto bitove kombinace je ulozena v GRAFM.
(R) Spoust joysticku 1. Stinovy registr STRIG1 (645=$285).

53266 $D012 COLPM0

(W) Barvova hodnota pro hrace a missile 0. Missiles maji vzdy stejnou barvovou hodnotu jako prislusny hrac. Barvova hodnota pro player/missile 0 je ulozena v BASICu ve stinovem registru PCOLR (704=$2C0).
Jsou-li ctyri missiles (nastavenim bitu 4 v GPRIOR 623=$26F) sdruzeny do pateho hrace, pak tento hrac dostane barvu z registru COLOR3 (711=$2C7), barveveho registru pro hraci pole 3 (v mnoha pripadech se to nepouziva).
(R) Spoust joysticku 2. Stinovy registr STRIG2 (646=$286).

53267 $D013 COLPM1

(W) Barvova hodnota pro hrac/missile 1. Stinovy registr PCOLR1 (705=$2C10.
(R) Spoust joysticku 3. Stinovy registr STRIG3 (647=$287).

53268 $D014 COLPM2

(W) Barvova hodnota pro hrac/missile 2. Stinovy registr PCOLR2 (706=$2C2).
(R) Pouziva se ke zjisteni, zda pristroj pouziva PAL - bity 1-3 nastaveny na 0, nebo NTSC - bity 1-3 nastaveny na 1 = dec.14.
NTSC je platna TV norma v USA; TV norma PAL se pouziva v Evrope; TV norma SECAM je zavedena v NDR, Francii a SSSR.
PAL je taktovan na 50 Hz, tedy VBLANK probehne kazdou 1/50 sec. Je tedy o 12 procent pomalejsi nez NTSC, kde probiha kazdou 1/60 sec. ATARI verze 50 Hz pouziva 2,217 MHz krystal a bezi o 25 procent rychleji nez US 60tiherzova verze.

Norma PAL mimoto pracuje misto s 262 se 312 obrazovymi radky (scan lines).Tento rozdil se vyrovnava tim, ze DL verze PAL zacina s 24 prazdnymi radky. To znamena ze lze vyuzit vetsi "vyrez" z obrazovky. Jak to probiha, si prectete v dodatku DL.

53269 $D015 COLPM3

Barvova hodnota pro hrac/strelu 3. Stinovy registr PCOLR3 (707=$2C3).

53270-53273 $D016-$D019 COLPF0-3

Barvova hodnota pro hraci pole 0-3. Stinovy registr COLOR0-3

(708-711=$2C4-$2C7).

53274 $D01A COLBK

Barvova hodnota pozadi (BAK). Stinovy registr COLOR4 (712=$2C8).

53275 $D01B PRIOR

(W) Zde se stanovi, ktere graficke prvky (player, missile, playerfield) prekryji ostatni pri vzajemnem stretu. Muze byt v BASICu popsano pouze pomoci stinoveho registru GPRIOR (623=$26F) , protoze BASICovy POKE by byl po PRIOR v nasledujicim strojovem taktu prepsan, zatimco hodnota (ulozena ve stinovem registru) by se v PRIOR obnovovala. Nasledujici priority se urcuji nastavenim odpovidajiciho bitu (PL=player, PF=plyerfield, BAK=pozadi):

bit 3	bit 2	bit 1	bit 0
PF 0	PF 0	P 0	P 0
PF 1	PF 1	P 1	P 1
P 0	PF 2	PF 0	P 2
P 1	PF 3/P 5	PF 1	P 3
P 2	P 0	PF 2	PF 0
P 3	P 1	PF 3/P 5	PF 1
PF 2	P 2	P 2	PF 2
PF 3/P 5	P 3	P 3	PF 3/P 5
BAK	BAK	BAK	BAK

Kdyz se v bitech 0-3 nastavi priority, ktere si odporuji, budou graficke prvky v koliznim rozsahu (cerne).
Bit 4 nastaven - sjednoti 4 strely do pateho hrace
Bit 5 nastaven - ORuje prekryvajici se barvove hodnoty
Bity 6 a 7 urcuji GTIA mod 9-11. Blizsi viz GPRIOR (623=$26F).

53276 $D01C VDELAY

(W) Umoznuje pohub hrace a strely v jednom radku. Je-li vyvolano dvouradkove reseni. Kdyz je nastaven odpovidajici bit, pohybuje se dotycny prvek o jeden radek dolu. Kdyz se aktivuje DMA zalozenim bit-vzoru do PM pametoveho rozsahu, ovlivnuje pohyb o vice nez jeden radek (viz program PMGRPLAY).

bit 7 (=128)	player 3
bit 6 (=64)	player 2
bit 5 (=32)	player 1
bir 4 (=16)	player 0
bit 3 (=8)	missile 3
bit 2 (=4)	missile 2
bit 1 (=2)	missile 1
bit 0 (=1)	missile 0

53277 $D01D GRACTL

(W) Bity 0-2 lze popsat. Je-li bit 2 nastaven, prepnou vsechny spouste vsech joysticku a paddlu na trvalou palbu pri jedinem stisknuti.Neni ale mozne nastavit na trvalou palbu jedinou spoust. Trvala palba skonci nastavenim bitu 2 zpet na nulu. S bitem 1 se aktivuji hraci a bitem 0 missiles v PM grafice.

53278 $D01E HITCLR

(W) Zapisem libovolne hodnoty do tohoto registru se vymazou vsechny registry PM kolizi. Jelikoz vsechny registry kolizi mohou registrovat dalsi kolizi az po svem vymazani, mel by byt HITCLR pravidelne popsan.

53279 $D01F CONSOL

Detekce funkci OPTION, SELECT a START. Kdyz neni stisknuto zadne tlacitko, jsou nastaveny bity 0-2 a CONSOL tedy cte sedmicku. Stisknuti jednoho z tlacitek nastavi odpovidajici bit zpet na nulu.

tlacitko	bit	okamzity stav bitu
OPTION	2	0 0 0 0 1 1 1 1
SELECT	1	0 0 1 1 0 0 1 1
START	0	0 1 0 1 0 1 0 1
dec.hodnota		0 1 2 3 4 5 6 7

Jak se PM grafika skutecne programuje, by vam mel priblizit nasledujici program:

```
1     REM  PMGRPLAY
2     REM  ******************
3     REM  *  VYCHOD SLUNCE *
4     REM  ******************
10    GR. 23
20    POKE 708,226:POKE 709,208:
      POKE 710,6: POKE 712,128
30    COLOR 1:FOR X=0 TO 5:PLOT X,0:
      DRAWTO X,95: NEXT X
40    FOR X=152 TO 159:PLOT X,0:
      DRAWTO R,Y: NEXT X
50    FOR Y=0 TO 69:READ R:PLOT 6,Y:
      DRAWTO R,Y: NEXT Y
60    FOR Y=0 TO 53:READ L:PLOT L,Y:
      DRAWTO 151,Y: NEXT Y
70    DATA 37,38,39,40,40,41,41,41,40,
      40,38,36,35,34,33,33,33,32,33,35,
      36,38,37,37,36,35,34,28,25,24,23,
      22,23,24,24
80    DATA 26,27,26,24,22,20,17,18,19,
      20,20,18,16,14,11,8,7,8,7,9,8,10,
      13,15,15,14,14,10,9,11,10,10,9,7,
      8,7
90    DATA 148,146,144,141,138,139,143,
      147,148,143,141,147,145,143,140,
      137,134,131,128,129,130,134,138,
```

```
      143,141
100   DATA 139,137,138,140,136,132,128,
      123,124,125,128,131,135,139,143,
      149,147,145,142,139,135,131,126
      121,119
110   DATA 120,122,125,131
120   COLOR 3:Z=Z+1
130   X=INT(RND(0)*80)+40:Y=INT(RND(0)*
      70): PLOT X,Y
140   IF Z<30 THEN 120
150   COLOR 2:FOR Y=0 TO 7:PLOT 6,81+Y:
      DRAWTO 52,84+Y:DRAWTO 80,80+Y:
      DRAWTO 135,86+Y:DRAWTO 150,82+Y:
      NEXT Y
160   FOR Y=86 TO 95:PLOT 6,Y:
      DRAWTO 150,Y: NEXT Y
200   POKE 704,34
210   POKE 623,8
220   POKE 559,42
230   P=PEEK(106)-20
240   POKE 54279,P
250   PMBASE=P*256
260   POKE 53256,3
270   POKE 53277,2
280   X=30:Y=91
300   POKE 53248,X
310   FOR J=PMBASE+512 TO PMBASE+639:
      POKE J,0: NEXT J
320   FOR J=PMBASE+512+Y TO PMBASE+512+
      31+Y:READ D:POKE J,D: NEXT J
330   DATA 24,60,60,60,126,126,126,126,
      126,255,255,255,255,255,255,255
340   DATA 255,255,255,255,255,255,255,
      126,126,126,126,126,60,60,60,24
400   S=STICK(0)
410   IF S=15 THEN 400
420   IF S=7 THEN X=X+1
430   IF S=11 THEN X=X-1
440   POKE 53248,X
450   IF S=13 THEN GOSUB 500
460   IF S=14 THEN GOSUB 600
470   GOTO 400
500   IF Y>92 THEN RETURN
510   FOR J=32 TO 0 STEP-1
520   POKE PMBASE+512+Y+J,PEEK(PMBASE+
      511+Y+J)
530   NEXT J
540   C=C-1: IF C<0 THEN C=0
550   CO=INT(C/15)*2
560   POKE 704,34+CO:POKE 712,128+CO/2
      :POKE 709,208+CO
570   Y=Y+1
580   RETURN
600   IF Y<16 THEN RETURN
610   FOR J=0 TO 32
620   POKE PMBASE+511+Y+J,PEEK(PMBASE+
      512+Y+J)
630   NEXT J
640   C=C+1:IF C>75 THEN C=75
```

```
650   CO=INT(C/15)*2
660   POKE 704,34+CO:POKE 712,128+CO/2:
      POKE 709,208+CO
670   Y=Y-1
680   RETURN
```

10- prikaz GRAPHICS se vztahuje pouze na hraci pole. PM prvky na nem vubec nezavisi.

20- v odpovidajicich barvovych registrech se ulozi barvove hodnoty grafickych bodu (hraci pole) a pozadi.

30-110- kresli s COLOR 3 vlevo siluetu listnateho stromu a vpravo jedli.

120-140- PLOTuji nahodne na nocni oblohu 30 hvezdicek.

150-160- vytvori malebny horsky hrbet.

200- ted najizdi PM grafika. Pro hrace 0 se ulozi barvova hodnota v odpovidajicim barvovem registru.

210- priorita je nastavena tak, ze stromy a hory lezi pred hracem 0. Tmava nocni obloha lezi za hracem 0 a je dana pozadim.

220- vyvola se normalni sirka hraciho pole (bit 1=2), hrac (bit 3=8) a DMA (bit 5=32).

230- pomoci RAMTOP se zjisti, kde lezi hranice pameti. GR.23 obsadi se svym DL presne 16 stranek. Protoze PM grafika ma probihat ve dvouradkovem reseni, postaci vam 4 stranky. Tedy: zakladni adresa PM grafiky musi lezet 20 stranek pod RAMTOP. Tuto hodnotu obsahuje P.

240- registr 54279 je PMBASE, startadresa PM pametoveho rozsahu. Protoze se PMBASE stejne jako RAMTOP vydava jen HI bytem, obsahuje skutecna

250- pocatecni adresa hodnoty znasobenim 256.

260- hrac 0 dostane ctyrnasobnou sirku. Kazdych 8 bitu znazornuje sirku 8 obrazovych bodu a hrac celkem obsazuje na stinitku 64 obrazovych bodu.

270- zapne hrace.

280- pocatecni souradnice hrace.

300- v HPOSP0 se ulozi x=30 jako horizontalni poloha hrace 0. Je to pozice na nejzazsim viditelnem okraji TV stinitka. Je-li hrac nastaven jeste vic vlevo, zmizi z obrazovky. Preruseni ERROR ale nastane az v pripade, ze hodnota x je mensi nez nula, nebo na prave strane vetsi nez 255.

310- pametova mista 512-639 za PMBASE jsou pri dvouradkovem reseni vyhrazena pro hrace 0. Zde se tento pametovy rozsah vymaze, coz neni bezpodminecne nutne. Preruseni programu napr.BREAK totiz PM pamet nemaze, takze pri obnovenem startu programu se na obrazovce objevi 2 hraci. Jeden, nepohyblivy, je ve vertikalni pozici, kterou zaujimal pri preruseni programu a druhy je tam, kam byl nastaven druhym prubehem programu, tedy v pocatecni pozici. Je tedy lepsi nejdrive vymazat PM pametovy rozsah a pak teprve ulozit hrace do registru 53277.

320- zde se datove byty predstavujuci hrace nactou do pametoveho rozsahu pro hrace 0. Hrac je vzdy jeden byte siroky a pri dvouradkovem reseni 128 bytu vysoky. Pro techto 128 bytu jsou k dispozici pametove bunky PMBASE + 512 - PMBASE + 639. Prvni byte je cten o hodnotu Y=91

nize nez PMBASE + 512 a nasleduje dalsich 31 bytu.

330,340-obsahuji DATA znazornujici hrace. Kazda decimalni hodnota reprezentuje jednu bitovou kombinaci. Kazdy nastaveny bit vyrabi jeden pixel, jehoz sirka je urcena v odpovidajicim registru (zde ctyrnasobek= 8 obrazovych bodu) a jehoz vyska pri dvouradkovem reseni je dva obrazove radky, pri jednoradkovem reseni jeden TVradek, coz je ulozeno v registru 559 nastavenim nebo nenastavenim bitu 4.

400- Joystick 0. Je dobre priradit hodnotu nalezenou ve STICK0 (632=$278) nejake promenne, aby se dlouhy vyraz STICK(0) nemusel vickrat opakovat. BASIC prikaz STICK(0) nezpusobi o nic mene nez statement PEEK(632), ale STICK(0) zabere mene pameti.

410- kdyz je STICK(0) v klidu (=15), je hned zase tazan.

420- je-li joystick tlacen doprava, mel by se tam pohybovat i hrac, tzn. jeho horizontalni poloha by mela prijimat stale vyssi hodnotu.

430- pohyb vlevo stejnym zpusobem.

440- horizontalni poloha hrace je jednoduse zapsana v prislusnem registru.

450- jinak to vypada se svislym pohybem. Hrac se pohybuje ve vertikalnim smeru podle ulozeni jeho dat v pametovem rozsahu. Ma-li se hrac pohybovat dolu, musi byt vsechny jeho databyty ukladany stale o jednu pametovou bunku nize. Cim vic dat hrac obsahuje, tim dele tento pochod trva. Rutina k tomu je ulozena v podprogramu na radku 500.

460- stejne probiha pohyb vzhuru.

470- po zjisteni smeru pohybu se program vrati zpet k dalsimu dotazu na joystick.

500- zde zacina rutina pro pohyb dolu. Pri dosazeni Y pozice 92, by nemel byt dovolen dalsi pohyb dolu. To je nutne, protoze data putuji pomoci prekladacich rutin v pametovych oblastech kam nepatri, napr. v pametovem sektoru druheho hrace.

510- protoze hrac je zde vysoky 32 bytu, musi byt ulozeno 33 (0-32) bytu. Pri pohybu dolu se zacne s nejnizsim bytem, ktery se prelozi o jednu adresu nize. Pak totez s dalsimi. Po prelozeni vsech 32 bytu o jeden radek dolu se mus prelozit dolu jeste nejblizsi vyssi byte. Tento byte obsahuje nulu, kdyz tam hrac konci. Tak je posledni byte po prelozeni dolu ve sve stare bunce smazan.

540,560-kdyz slunce jako hrac 0 pomalu vychazi za horami v COLOR 2 a jeho rude svetlo pada na stromy v COLOR 1, melo by byt prirozene svetlejsi. Slunce se meni prez oranzovou do bile, obloha se stava zarive modrou a prezaruje hvezdy a louky se meni do zelene. V techto radcich se meni barevne hodnoty, kdyz se slunce pohybuje dolu, tedy pri zapadu slunce. Barvy tmavnou a svetlost se snizuje.

570- ted se musi aktualizovat y-hodnota a pak

580- RETURN k hlavnimu programu

600-680-analogicky pohyb vzhuru

Jak se pracuje s registry kolizi, ukazuje dalsi program:

```
1     REM PMGRCOLL
2     REM ************************
3     REM * PM SVETELNA PODIVANA *
4     REM ************************
10    GR. 19
200   POKE 704,50:POKE 705,152
210   POKE 623,8+32
220   POKE 559,42
230   P=PEEK(106)-8
240   POKE 54279,P
250   PMBASE=P*256
260   POKE 53256,3:POKE 53257,3
270   POKE 53277,2
280   X0=30:Y0=92:X1=190:Y1=16
300   POKE 53248,X0:POKE 53249,X1
310   FOR J=PMBASE+512 TO PMBASE+767:
      POKE J,0:NEXT J
320   FOR J=PMBASE+512+Y0 TO PMBASE+512
      +31+Y0:READ D:POKE J,D:NEXT J:
      RESTORE
330   FOR J=PMBASE+640+Y1 TO PMBASE+640
      +31+Y1:READ D:POKE J,D:NEXT J
340   DATA 24,60,60,60,126,126,126,126,
      126,255,255,255,255,255,255,255
350   DATA 255,255,255,255,255,255,255,
      126,126,126,126,126,60,60,60,24
400   S=STICK(0)
410   IF S=15 THEN 700
420   IF S=7 THEN X0=X0+1
430   IF S=11 THEN X0=X0-1
440   POKE 53248,X0
450   IF S=13 THEN GOSUB 500
460   IF S=14 THEN GOSUB 600
470   IF PEEK(53260)=2 THEN GOSUB 1000
480   GOTO 700
500   IF Y0>92 THEN RETURN
510   FOR J=32 TO 0 STEP-1
520   POKE PMBASE+512+Y0+J,PEEK(PMBASE
      +511+Y0+J)
530   NEXT J
540   Y0=Y0+1:RETURN
600   IF Y0<16 THEN RETURN
610   FOR J=0 TO 32
620   POKE PMBASE+511+Y0+J,PEEK(PMBASE
      +512+Y0+J)
630   NEXT J
640   Y0=Y0-1:RETURN
700   S=STICK(1)
710   IF S=15 THEN 400
720   IF S=7 THEN X1=X1+1
730   IF S=11 THEN X1=X1-1
740   POKE 53249,X1
750   IF S=13 THEN GOSUB 800
760   IF S=14 THEN GOSUB 900
770   IF PEEK(53261)=1 THEN GOSUB 1000
780   GOTO 400
800   IF Y1>92 THEN RETURN
810   FOR J=32 TO 0 STEP-1
820   POKE PMBASE+640+Y1+J,PEEK(PMBASE
```

```
      +639+Y1+J)
830   NEXT J
840   Y1=Y1+1:RETURN
900   IF Y1<16 THEN RETURN
910   FOR J=0 TO 32
920   POKE PMBASE+639+Y1+J,PEEK(PMBASE
      +640+Y1+J)
930   NEXT J
940   Y1=Y1-1:RETURN
1000  SOUND 0,3,6,8
1010  FOR Q=0 TO 6:POKE 712,Q*2:NEXT Q
1020  POKE 53278,255
1030  POKE 712,0
1040  SOUND 0,0,0,0
1050 RETURN
```

10- zapne se GR.3 (mala spotreba pameti). Tento graficky mod je oblibeny, protoze pouziva pouze barvu pozadi (0= cerna).
200- barvove hodnoty pro hrace 0 a hrace 1.
210- dodatecne se zde k priorite hrace a hraciho pole aktivuje (+32) treti, ORovana barva pri prekryti. Prirozene lze zaroven take ulozit (POKE) 40.
220- DMA
230- pro nepatrnou spotrebu pameti GR.3 (popr.19), vyzaduje zde PM baze ulozit pod RAMTOP pouze 8 stranek.
260- oba hraci ziskaji ctyrnasobnou sirku.
280- hrac 0 zacina vlevo dole, hrac 1 vpravo nahore.
310- tentokrat je vymazan PM rozsah pro hrace 0 a 1, a pak se v radku
320- budou cist data pro hrace 0 a v radku
330- data pro hrace 1. Protoze oba maji mit stejnou postavu, cte hrac 1 po RESTORE v radku 330 stejna
340,350-DATA jako hrac 0.
400-460-zeptaji se na STICK(0) a vydaji v radku 700, kdy prijde na radu STICK(1).
470- kdyz PEEK(53260)=2, pak se pta hrac 0 na hrace 1, od nej k subroutine na radku 1000.
500-640-svisly pohyb hrace
700-760-dotaz na STICK(1)
770- kdyz je registr 53261 nastaven na 1, pak se stretli oba hraci
800-940-vertikalni pohyb hrace 1
1000- kdyz na sebe pustite oba hrace, vznikne zde dira
1020- vymazte hned kolizni registry(jiskreni). Hrac 0 sviti cervene, hrac 1 modre. Prunikova plocha se zbarvi staroruzove.

Jeste jednou vypocet barvy pruniku:

cervena(bar. ton 3, svetlost 2) 0011 0010 (=3*16+2=50)
modra (" " 9, " 8) 1001 1000 (=9*16+8=152)
zelen (" " 11, " 10) 1011 1010 (=11*16+10=186)

Jestlize prerusite PM program pomoci BREAK, je normalni, ze na miste hracu zustanou blikajici pasy. Odstranime pomoci RESET.

7.0 ZVUK

53760-54015 D200-D2FF POKEY

53760 $D200 AUDF1

ATARI ma 4 tonove generatory rizene POKEY(potentiometer and keyboard chip).Ke kazdemu generatoru patri jeden registr pro frekvenci a jeden kontrolni registr pro rizeni hlasitosti. Zeslabeni se provadi "polycounterem" a registrem zeslabeni pomoci POKEY.
(W) AUDF1 prebira frekvenci pro tonovy kanal 1. Mohou byt zapsany hodnoty 0-255. Pokey pricita 1, takze jsou ucinne hodnoty 1-256. Tato hodnota urcuje pocet vystupnich impulzu,(definovanych hodinami obvodu POKEY) ktere musi probehnout, aby byl vydan jeden vystupni impulz. Cim vyssi je hodnota, tim mene impulzu vychazi a vysledna frekvence je nizsi. Vzorec vypoctu ciselne hodnoty pro urcitou frekvenci: INT(31960(f+2)), kde pro f je treba dosadit frekvenci tonu v Hz.

Polycounter (polynomialni citac) se uziva jako zdroj nahodnych impulzu pro generaci sumu. V obvodu POKEY jsou tri takove citace: ctyr-, peti-, 17-ti bitovy. Uziti kratkych polycounteru vytvari opakovane zvukove sekvence, zatimco v dlouhem se neobjevuje zadne opakovani.

(R)Hodnota paddlu 0. Stinovy registr 624.

53761 $D201 AUDC1

(W) Horni 3 bity se pouzivaji ke stanoveni zeslabeni (k sumu)

bit 7 6 5	hodn.zeslab. (BASIC)	modulac.postup (POKEY)
0 0 0	0	
17-bit-polycounter		
0 0 1	2	jen 5bitovy polycounter
0 1 0	4	
4-bit-polycounter		
1 0 0	6	jen 17bit-polycounter
0 1 1	8	jen 5bit-polycounter
1 0 1	10	zadny sum (zeslabeni)
1 1 0	12	jen 4bit-polycounter
1 1 1	14	zadny sum (zeslabeni)

Bity 0-3 urcuji decimalni hodnotu hlasitosti. Jsou mozne hodnoty 0 (0000) az 15 (1111).
Prikaz BASICu SOUND k,f,r,v je pomoci k spojen s tonovym kanalem 0-3. Poznavaci cislo f urcuje frekvenci v AUDFn registru,r je zeslabeni (prime hodnoty 0-14) a v je hlasitost (0-15) v AUDCn registru.

(R) Otocny regulator 1.

53762 $D202 AUDF 2

(W)Frekvence tonoveho kanalu 2 (R)Paddle 2.

53763 $D203 AUDC 2

(W)Kontrolni registr k tonovemu kanalu 2. (R)Paddle 3.

53764 $D204 AUDF 3

(W)Frekvence tonoveho kanalu 3. (R)Paddle 4.

53765 $D205 AUDC 3

(W)Kontrolni registr k tonovemu kanalu 3. (R)Paddle 5

53766 $D206 AUDF 4

(W)Frekvence tonoveho kanalu 4. (R)Paddle 6

53767 $D207 AUDC 4

(W)Kontrolni registr k tonovemu kanalu 4. (R)Paddle 7

53768 $D208 AUDCTL

(W)Regulace(rizeni) zvuku. AUDCTL je registr volby, ktery ovlivnuje vsechny tonove kanaly. Nastavenim ruznych bitu se dosahne nasledujiciho:

bit	funkce
7	ze 17-ti bitoveho polycounteru dela 9-ti bitovy
6	taktuje kanal 1 na 2,217 MHz
5	taktuje kanal 3 na 2,217 MHz
4	spojuje kanaly 1 a 2 (16bit)
3	spojuje kanaly 3 a 4 (16bit)
2	zavadi filtr pro vysoke tony do kanalu 1 , taktovan kanalem 2
1	zavadi filtr pro vysoke tony do kanalu 2 , taktovan kanalem 4
0	zapina hlavni takt ze 64kHz na 15kHz.

(R)Cte 8 paddlu najednou, kazdy v 1 bitu.

54016-54271 $D300-D3FF PIA (6520)

54016 $D300 PORT A

Cteni/zapis z I/O portu (joysticky)

54017 $D301 PORT B

Rizeni pameti.

54018 $D302 PACTL

Ridici registr portu A.

54019 $D303 PBCTL

Ridici registr portu B.

8.0 RUZNE

53770- $D20A RANDOM

(R) Z tohoto registru se ctou nahodna cisla. Obsahuje hornich 8 bitu 17ti bitoveho polycounteru. Tak se zde prubezne nachazeji nahodna cisla mezi 0-255. BASIC prikaz RND(0) prevede nalezena nahodna cisla na decimalni cisla mezi 0 a 1, pricemz hodnotu z RANDOM deli 256. Prikazy PRINT RND(0) a PRINT PEEK(53770)/256 davaji stejny vysledek.
K dosazeni celych nahodnych cisel 5-14 se v BASICu programuje INT(RND(0)*10)+5. RND(0)*10 vyda hodnoty 0-9,99..., ktere se pomoci INT zaokrouhli na cela cisla (0-9). Pricteni 5 pak vede k zadanym hodnotam 5-14. INT(PEEK(53770)/256)+5 ma stejny ucinek.

53774- $D20E IRQEN

(W) Aktivace umozneni pozadavku na preruseni. Nula v prislusnem bitu blokuje pozadavek na preruseni, jednicka ho umoznuje. Stinovy registr POKMSK(16=$10).

BIT	
0	timer 1
1	timer 2

2 timer 4
3 ukonceni serioveho vystupu
4 seriova vystupni data ocekavana
5 seriova vstupni data pripravena
6 jakakoliv klavesa (mimo BREAK)
7 klavesa BREAK

54272-54783 D400-D5FF ANTIC

54272- $D400 DMACTL

kontrolni registr DMA. V BASICu je popsan pomoci stinoveho registru SDMCTL (559=$22F).

54273 $D401 CHACTL

Kontrolni registr charakteroveho modu. Je popsan stinovym registrem CHACT (755=$2F3).

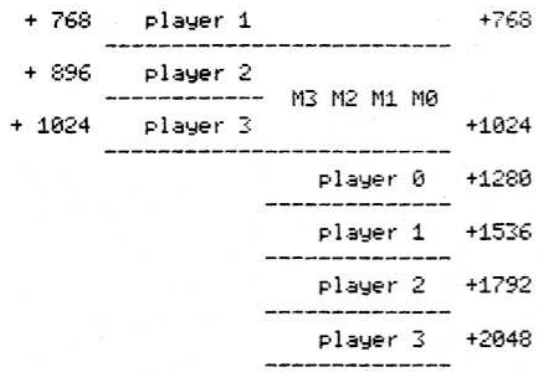
54274,54275 $D402,$D403 DLISTL/H

LO a HI byte ukazatele na display list (viz dodatek DL).

54279 $D407 PMBASE

(W) HI byte PM startadresy.
Pro data hrace a strely existuje pametovy plan, ktery vykazuje kazdemu z prvku pametove misto. Hrac je vzdy 1 byte siroky a ve vertikalnim smeru pokryva cele stinitko i pres graficke okenko zpracovane grafickym modem. Pri dvouradkovem reseni je hrac 128 bytu vysoky, pri jednoradkovem vyzaduje 256 bytu pameti.
Ctyri strely, ktere se mohou sdruzit do pateho hrace, obsazuji celkem tolik pametoveho mista, jako jeden hrac. Dohromady s nepouzivanym pametovym rozsahem na zacatku PM pameti vystupuje potreba 1 kbytu pri dvouradkovem a 2 kbytu pri jednoradkovem reseni. Jak se obycejne rozdeluje pametovy rozsah, vam ukazuje obrazek:

Plan PM pameti:



Je bezpodmineene nutne dodrzet toto usporadani, protoze se funkce (jako kolizni registr atd.) PMBASE a tato organizace pameti primo dotykaji. Je ucelne ukladat PM tabulku na konec RAM, tedy primo pred DL a obrazovkovou pamet. Cislo stranky (HI byte adresy), na ktere zacina PM pamet, musi byt ulozeno v PMBASE.
K nalezeni bezpecne startadresy pro PM grafiku lze zaokrouhlit potrebu pameti aktivovaneho grafickeho modu a k nemu prislusneho DL na stranky plus delku PM tabulky, tzn.odecist od RAMTOP 4 nebo 8 stranek.
Ke stejnemu vysledku se dojde, kdyz se po zapnuti zadaneho grrafickeho modu nahledne do ukazatelu na display list DSDLSTL (560,561=$230,$231). Tato adresa, zaokrouhlena na cele stranky a odectena od potreby PM pameti, da prave PB zakladni adresu, od ktere musi byt HI byte v PMBASE ulozen.
Na zaklade PMBASE lze vypocitat hodnotu pro APPMHI, pricemz se k PMBASE pricte potreba PM pameti o 4 strankach pri dvouradkovem a o 8 strankach pri jednoradkovem reseni. APPMHI oznacuje dolni hranici pro dispaly list a obrazovkovou pamet.

54281 $D409 CHBASE

(W) Pocatecni adresa standardni sady znaku ATARI, ktera je prirozene ulozena v ROM rozsahu. Sadu znaku tvori 128 znaku (characters), kazdy po 8 bytech. Obsazuje tedy presne 4 stranky (page) nebo 1 kbyte. Ukazatel na sadu znaku lze v BASICu zmenit ve stinovem registru CHBAS (756=$2F4). Dalsi udaje - viz dodatek sada znaku.

9.0 DISPLAY LIST

DL je strojovy program pro mikroprocesor ANTIC. Tento program urcuje, z ktereho pametoveho rozsahu se graficka data vyvolaji a v jakem zpusobu se na displeji zobrazi. Kdyz se v BASICu zada prikaz GRAPHICS, automaticky se rezervuje pro obrazovku zadany pametovy rozsah (SM=screen memory) a pred nim se ulozi odpovidajici DL. Pokud znate stavbu DL a sadu prikazu ANTIC, muzete DL zmenit, nebo si stanovit uplne vlastni DL, cimz lze smichat vsechny graficke moznosti znazorneni ANTICem. Jak je DL sestaven, vam na monitoru ukaze program ADR560 (viz SDLSTL 560,561=$230,$231).

ANTIC zpracovava nasledujici 8-bitove instrukce:
(1,2 atd jsou primo jejich operacni kody)

1. JMP, prikaz nepodmineneho skoku. Bezprostredne za timto prikazem musi nasledovat adresa skoku v poradi LO,HI byte. Timto prikazem lze napr. skocit do podprogramu (subroutine), treba pro druhy, maly DL, ktery pres nebo pod normalni graficke okenko vydava jeden nebo vice radku pracujicich nezavisle na velkem, hlavnim DL.
2. Znakovy mod, ktery se v BASICu vyvola pomoci GR. Zpusobovy radek je delen na 40 zapisovych mist a je vysoky 8 TV radku (scan lines). Lze zaroven znazornit 2 barvy.
3. Znakovy mod. 40 znakovych mist, 10 scan lines, 2 barvove hodnoty.
4. Znakovy mod (GR.12).
40 znakovych mist, 8 scan lines, 5 barvovych hodnot.
5. Znakovy mod (GR.13). 40 znakovych mist, 16 scan lines, 5 barvovych hodnot.
6. Znakovy mod (GR.1) 20 znakovych mist, 8 scan lines, 5 barvovych hodnot.
7. Znakovy mod (GR.2) 20 znakovych mist, 16 scan lines, 5 barvovych hodnot.
8. Bit-map mod (GR.3) 40 pixelu na radek, kazdy zpusobovy radek vysoky jako 8 TV radku, 4 COLOR
9. Bit-map mod (GR.4) 80 pixelu, 4 scan lines, 2 COLOR.
10. Bit-map mod (GR.5) 80 pixelu, 4 scan lines, 4 COLOR.
11. Bit-map mod (GR.6) 160 pixelu, 2 scan lines, 2 COLOR.
12. Bit-map mod (GR.14) 160 pixelu, 1 scan line, 2 COLOR.
13. Bit-map mod (GR.7) 160 pixelu, 2 scan lines, 4 COLOR.
14. Bit-map modu (GR.15) 160 pixelu, 1 scan line, 4 COLOR.
15. Bit-map mod (GR.8) 320 pixelu, 1 scan line, 2 COLOR.

Udaje o pixelech pro radek se vztahuji k normalni sirce TV obrazovky:

0	1	prazdny TV radek
16	2	" "
32	3	" "
48	4	" "
64	5	" "
80	6	" "
96	7	" "
112	8	" "

64. Pricita se k operacnimu kodu instrukce pro zpusobovy radek (tedy bit 6 nastaven). Aktivuje LMS (load memory scan), tzn.ze graficka data budou ctena z obrazovkove pameti. Kazdemu zadani LMS musi nasledovat startadresa obrazovych dat, jako LO a HI byte.
65. JVB. VBLANKem podmineny skok. Touto instrukci na konci DL dosahneme synchronizace mezi pocitacem a obrazovkou. Na konci DL ceka pocitac na VBLANK-signal, aby skocil na zacatek DL a tak synchronne s TV obrazovkou zacne dalsi krok. Jako kazdemu skokovemu prikazu, musi i JVB nasledovat prima adresa skoku (LO,HI).
128. DLI (display list interrupt).Prictena k operacnimu kodu instrukce zpusoboveho radku umoznuje hodnota 128 (bit 7) preruseni DL, aby se v HBLANK zpracovala kratka strojova rutina. Jako priklad programu viz ADR512.
Standardni DL zpracovavaji 192 scan lines. U PAL verze se pouziva k vyrovnani systemu 3*8 prazdnych radku, takze se zpracovava 192+24=216 TV radku. Zmenou DL lze 24 prazdnych radku aktivne vyuzit. Modifikovany DL muze zpracovat take mene nez 216 popr.192 scan lines. Pak zustane na dolnim okraji obrazovky siroky prazdny cerny pas. Pri prikazu JVB ceka pocitac na katodovy paprsek obrazovky, takze i libovolne kratsi DL vzdy bezi s monitorem synchronne.
Protoze PAL system deli obrazovku na 312 (misto u NTSC 262) radku, urcuje 24 prazdnych radku na zacatku DL to, ze graficke okenko stoji na PAL obrazovce priblizne uprostred vertikalniho smeru. To ale znamena, ze prostrednictvim standardniho DL pri PAL lze dodatecne vyuzit jeste zbylych radku (asi 20). Je-li DL delsi, nez muze TV radkove zpracovat, neni tedy jeste zpracovan, kdyz jeste na obrazovce probiha VBLANK, pak je synchronizace nemozna, obraz se rozpadne a zacne behat. K poskozeni obrazovky to ale nevede.
Nasledujici program dava priklad, jak lze v BASICu programovat vlastni DL:

```
  0     REM DLTWUP
  1     REM*****************************
  2     REM*                           *
  3     REM* GR.8 TEXT.OKENKO NAHORE   *
  4     REM*                           *
  5     REM*****************************
 10     GRAPHICS 24:POKE 703,4
 20     DL=PEEK(560)+PEEK(561)*256
 30     FOR J=0 TO 9:READ B:POKE DL+J,B
        :NEXT J
 40     DATA 112,66,96,159,2,2,2,79,80,
        129
100     COLOR 1
110     PLOT 0,0:DRAWTO 319,191
120     PLOT 319,0:DRAWTO 0,191
130     PLOT 0,0:DRAWTO 100,89
140     PLOT 100,93:DRAWTO 207,191
200     ? "GRAPHICS 8,TEXT.OKENKO NAHORE"
```

10- at uz standardni DL modifikujete pouze z casti, nebo ho cely prestavite, nemuzete se v BASICu obejit bez

vyvolani nejakeho grafickeho modu. Ten totiz vyridi nektere ukoly, ktere bez dalsiho nelze vyridit ani s PEEK nebo POKE.
GR. totiz rezervuje pametove misto, ktere vyvolany graficky mod potrebuje pro obrazova data. Kdyz sestavite vlastni DL, musite spotrebu SM vypocitat a zapojit nejaky standardni graficky mod, ktery ma vetsi nebo nejmene stejnou spotrebu pameti. Pametovou spotrebu ruznych zpusobovych radku najdete v tabulce v kapitole obrazovkova pamet.
Vyvolani grafickeho modu prikazem GR. ulozi do pameti take odpovidajici DL od mista daneho vektorem SAVMSC (88,89=$58,$59).
DL nesmi prekrocit hranici 1 kbytu bez obnoveneho prikazu skoku (JMP...). Nejdelsi DL GR.8 je rada 202 bytu dlouha. Je tedy mozne umistit DL mezi 1 kbytovou hranici, ale na to se musi dat pozor.
Presto muze DL dosahovat pozoruhodnych delek. LMS pro ANTIC je zalozen na nastaveni bitu 6 (=64) a bytu s instrukci pro zpusobovy radek, tedy decimalne vzato se k poznavacimu cislu zpusoboveho radku pricte 64. Teoreticky je mozne opatrit kazdy zpusobovy radek jednim LMS a tim kazdemu grafickemu radku priradit vlastni pametovy rozsah. Po kazdem LMS ale musi nasledovat vektor na obrazovkovou pamet. A to jsou dva dodatecne byty. Dalsi omezeni spociva v tom, ze pamet grafickych dat (SM) nesmi prekrocit 4kbytovou hranici. Kdyz nahlednete do pameti, kam se zavadeji dlouhe DLI GR.8 - 11, 14 a 15 (SM spotreba 7680 bytu), najdete kousek pred stredem novou instrukci LMS s odpovidajicim ukazatelem na adresu (ulozenou kousek pred stredem) obrazovkove pameti, od ktere pak pokracuji graficka data pro dolni cast obrazovky.
Vlozeny graficky mod take urcuje, jak budou zpracovana data z SM, tedy ktera bitova maska (DMASK 672=$2A0) se pouzije, tzn. kolik bude pro pixel cteno bytu a kolik barvovych registru bude pouzito. Tuto funkci lze ovlivnit ulozenim (do DINDEX 87=$57) hodnoty simulujici nejaky jiny graficky mod. GR. nebo jina hodnota ulozena v DINDEX ovlivnuje zas jen prikazy pro obrazovku, jako PLOT, LOCATE atd. Zde v programovem radku 10 se jeste stanovi vyska textoveho okenka na ctyri (GR.0 radky), protoze je programovan GR.24, tedy 8 bez textoveho okenka. Jednoduse proto, ze s tim spojeny DL nam pri potizich v programovani lepe vyhovuje.
Textove okenko by melo byt prelozeno na horni okraj obrazovky. Proto je zvolen DL bez textoveho okenka, protoze na jeho dolnim konci nejsou nutne zadne dalsi zmeny.

20- vypocita startadresu DL a jde na radek
30- kde zacne s prestavbou. Smycka FOR-NEXT precte 10 hodnot a ulozi je (POKE) do DL.
40- zde jsou hodnoty. 112 zpusobi 8 prazdnych TV radku. 66 je LMS (64) + jeden radek GR.0 (2). Novy DL zacina tedy o 16 prazdnych TV radku vyse. 96 a 159 jsou LO a HI vektory na pametovy rozsah textoveho okenka. Nasleduji dalsi 3 radky GR.1. 79 obsahuje opet LMS (64) + prikazovou hodnotu 15 pro jeden radek GR.8, kteremu nasleduje LO (80) a HI (129) SM ukazatele SAVMSC.

Zbytek DL muze zustat beze zmeny.
Pokud jste si zvolili GR.8, opet se nevyhnete problemum. Pro tento instrukcni program byl GR.8 zvolen umyslne, protoze pri vysokem modu nastavaji problemy s vyse zminenymi "kilovymi" hranicemi.Novy rozsah GR.8 zacina primo pod textovym okenkem a ve svem hornim dilu probiha docela klidne. Pak ale prijde hranice, kde je v DL novy LMS poukaz s vektorem na zbylou SM. Ted uz nebude nove graficke okenko vysoke 192 zpusobovych radku (GR.24!), jak to ocekava OS a jak to pro vsechny prikazy PLOT predpoklada, ale jen 176 radku. A rozdil 16 zpusobovych radku musi nekde zustat.

100-120-PLOT a DRAWTO kresli pres cele grraficke okenko diagonalni kriz, a vidite, ze uprostred chybi prouzek. Tento nedostatek nelze odstranit ani zmenou ukazatele po druhem LMS, protoze OS pocita vsechny PLOTs, ktere vyzaduje prikaz DRAWTO, na zaklade zapnuteho GR.8 (popr.24).

130,140-V takovem pripade nezbyva nez programovat prikazy na obrazovku pro kazdou cast zvlast.

200- jen textove okenko funguje bez problemu.

Zde predstavena problematika DL je prikladem toho, ze zprvu zvolena lehci cesta nemusi byt vzdy zrovna nejlepsi. Kdybyste totiz v radku 10 nevyvolali GR.24 ale GR.8 a DL zmenili tak, ze byste ho na dolnim konci text. okenka zkratili a nahore zabezpecili, aby se prazdne radky (TV) nezaktivovaly, pak by pametove misto souhlasilo, a nenastaly by zadne problemy. Zkuste to!

10.0 OBRAZOVKOVA PAMET

Jako sreen memory (SM, obrazovkova pamet) se oznacuje ta oblast, do ktere se ukladaji data urcujici graficky vzhled obrazovky. LMS v DL ocekava, ze najde graficka data. Zpusob prevedeni techto dat na graficke informace urcuje prikaz pro zpusobovy radek. Existuji dva zcela ruzne provozni zpusoby, a to znakovy mod a bit-map mod.
Ve znakovem modu se premeni kazdy datovy byte, ktery se ma precist v SM, na nejaky znak. Znak je urcite usporadani nejakych obrazovych bodu. Vztah mezi znakem, jeho hodnotou a vzorem bodu, ktery je na stinitku zobrazen, je dan sadou znaku (viz dodatek sada znaku). Znaky sady znaku jsou urceny hodnotami 1-127. Tyto hodnoty jsou vsak odlisne od ATASCII

kodu. Pak je rec o internim kodu.
Ve znakovem modu se interpretuji hodnoty z obrazovkove pameti jako interni kody znaku. Znak se precte ze sady znaku a na obrazovku je vydana bitova kombinace, ktera ho reprezentuje.
Prikaz LOCATE x, y, n vyhleda v obrazovkove pameti bunku odpovidajici obrazovkove pozici x,y precte zde ulozenou hodnotu (interni kod) a prevede do ATASCII. LOCATE tedy urci ATASCII hodnotu znaku, ktery stoji na obrazovce na urcenem miste. Prikaz LOCATE pracuje pouze pri behu prikazu GR., tzn. ze se musi GR.0 zvlast programovat, coz jindy neni nutne.
Take GR.1 a 2 jsou znakove mody. Rozdil je pouze v tom, ze se ctou pouze bity 0-5, aby se znak v sade znaku vyhledal. Proto lze znazornit jen polovinu sady znaku, interni kod 0-63. Bity 7 a 6 maji vyznam barvove informace. Dvema bity lze znazornit 4 ruzne hodnoty (0-3). Proto lze znaky obou techto provoznich zpusobu znazornit ve ctyrech ruznych barvach. Pata barva je urcena pro pozadi.
Take znakove mody 12 a 13 mohou zobrazit znaky vicebarevne, a presto lze vyvolat celou sadu znaku, protoze barvova informace se zde ziskava jinym zpusobem. Pri standardnim znaku reprezentuje kazdy bit jeden obrazovy bod, ktery se muze objevit ve dvou barvach, totiz v barve znaku a pozadi. V obou techto grafickych modech se dva bity zobrazi jako jeden obrazovy bod, kteremu proto mohou odpovidat 4 ruzne barvove hodnoty (registry). Jestlize se v tomto modu zobrazi znak ze standardni sady znaku, pak bude temer necitelny, protoze graficka forma je presazena pouze castecne, zatimco prichazeji nesmyslne barvove informace (blizsi viz kapitola sada znaku).

V bit-map modu se na TV obrazovce znazorni pixel, jehoz velikost zavisi na zvolenem grafickem modu. Kdyz kazdy bit znazorni jeden graficky bod, pak ten muze prijmout pouze dve barvy, totiz 0 nebo 1. Ktera barva (COLOR) dostane 0 nebo 1 je urceno v prislusnem barvovem registru COLORu (viz tabulku) pomoci POKE nebo SETCOLOR. Osm takovych grafickych bitu se umisti v 1 bytu SM. Podle polohy bitu uvnitr bytu , popr. pixelu na obrazovce, dostane bit hodnotu dle znameho zpusobu:

```
        DVE  BARVY:
     pixel na obrazovce

     ----------------
     !*! ! !*! ! !*!*!
     ----------------

      bitova kombinace

     ----------------
     !1!0!0!1!0!0!1!1! =147
     ----------------
hodn.=128!64!32!16!8!4!2!1!
```

Hodnota 147 v jedne pametove bunce SM vyvola na obrazovce shora uvedeny sled pixelu.
Graficke mody 0,4,6,8 a 14 pracuji ve dvou barvach.Maji-li se rozeznavat 4 barvy, je pro kazdy pixel nutna dvoubitova informace:

```
              CTYRI BARVY:
```

pixel	0.	1.	2.	3.
=147	!1!0!0!1!0!0!1!1!			
color	2	1	0	3

Datovy byte 147 pak vytvari radu o 4 pixelech v barvach 2, 1, 0, 3. V obrazovkove pameti je tedy ulozena COLOR hodnota pixelu v binarni forme. Pro COLOR hodnoty se najdou odpovidajici barvove registry. Protoze prikaz PEEK najde vzdy pouze decimalni hodnotu bytu, vytvari bity nekolika pixelu vzdy jednu spolecnou decimalni hodnotu. Graficke mody 3, 5, 7 a 15 pracuji se dvema bity pro jeden pixel. V GTIA modech 9, 10 a 11 se pouzivaji pro pixel 4 bity. Se ctyrmi bity lze odlisit 16 barvovych hodnot. Protoze ATARI ma ale jen 9 barvovych registru, neda se 16 barev definovat. V GR.9 znamena COLOR 0-15 ruzne svetlosti jedne urcite barvy, GR.11 pouziva 16 standardnich barev, ale ve stejne svetlosti. V GR.10 lze sice volne definovat barvove hodnoty, ale protoze je k dispozici pouze 9 barvovych registru, muze se vydat jen 9 COLOR prikazu.

1

16 BAREV

pixel	0.	1.
= 147	!1!0!0!1!0!0!1!1!	
color	9	3

V GTIA modu ovlada 147 dva pixely s COLOR hodnotami 9 a 3. Obrazovkova pamet sama je usporadana linearne. Prvni SM byte urcuje obsah leveho horniho rohu grafickeho okenka. Pak nasleduji data pro nejvyssi radek az k pravemu okraji, pak data pro druhy a dalsi radky az k posledni hodnote, ktera urcuje podobu praveho dolniho rohu.
Pocet bytu, ktere se vejdou do 1 radku, zavisi na velikosti pixelu a na poctu moznych barev, tedy na bitu pro pixel.
V GR.3 lezi v jednom radku 40 pixelu. Protoze jsou mozne 4 barvy, obsazuje kazdy pixel dva bity. Jeden radek v GR.3 obsazuje 80 bitu=10 bytu. Aby se na obrazovce dosahlo urciteho bodu, je nutne sdelit, v kterem bytu jsou ulozena data pro tento bod. Ma-li bod souradnice x,y pak jeho data lezi v bytu =SAVMSC + y * rpz * INT(x/ppb), kde rpz = RAM pro radek nebo byte pro radek , ppb = pixel pro byte , a SAVMSC = vektor na startadresu obrazovkove pameti. Hodnota v tomto ukazateli smeruje na prvni byte SM.
Po nalezeni spravneho bytu se musi stanovit, na kterem miste v bytu lezi data zmineneho gr.bodu. V GR.3 by mel lezet pixel v COLOR 2 na pozici 5,2. Pozice 5,2 udava offset od SAVMSC o 21 bytech. Ve 21.bytu pod SAVMSC lezi pixely s nasledujicimi obrazovkovymi souradnicemi:

bit	7	6	5	4	3	2	1	0
dec.h.	128	64	32	16	8	4	2	1
POSITION	-4,2-		-5,2-		-6,2-		-7,2-	
COLOR	0	0	1	0	0	0	0	0

Na pozici 5,2 je ulozena COLOR hodnota 2 (bin.10). Kdyz zustanou zbyvajici 3 body cerne (pozadi, COLOR0), pak dostane byte 21 hodnotu 32. Prikaz BASICu COLOR 2 :PLOT 5,2 lze nahradit prikazem
POKE SM + 21,32 , pricemz prirozene musime SM jako startadresu obrazovkove pameti vypocitat z vektoru SAVMSC: SM= PEEK(88) + PEEK(89) * 256.
Maji-li se na obrazovku privest jednotlive graficke body, pak to prikaz PLOT velmi rychle provede. Ale take POKE do SM ma sve vyhody.

POKE nazavisi na pozici kurzoru, ktery muze byt po zmene DL (od OS) kdekoli a nemusi byt zrovna spolehlive rozeznan. S POKE lze tedy popsat urcite obrazovkove rozsahy, ktere jiz s PLOT a DRAWTO nejsou vubec pristupne.

Jine zajimave pouziti spociva v pametech obrazovkovych rozsahu, kdy se byte po bytu z obrazovkove pameti vybiraji (PEEK) a ukladaji (PUT) pres datovy kanal na disketu. Obracene se pak data z disketoveho souboru ctou byte po bytu (GET) a ukladaji (POKE) do SM.
pouziti pro prime adresovani obrazovky ukazuje nasledujici program:

```
  0     REM SMPOKE
  1     REM ********************
  2     REM *                  *
  3     REM * POKLUSEM KLUS ! *
  4     REM *                  *
  5     REM ********************
 10     S=40040:? CHR$(125)
 20     FOR P=0 TO 39:READ B:POKE S+P,
        B:NEXT P
 30     DATA 0,10,0,10,0,10,0,10,0,10,0,10,0,0,10,0,10,0,10,
        0,10,0,10,0,10,0,10,0,10
 40     DATA 0,52,101,115,116,0,10,0,10,0,10
 50     J=PEEK(S):K=PEEK(S+1):L=PEEK(S+
        2):M=PEEK(S+3)
 60     FOR I=4 TO 39
 70     POKE S+I-4,PEEK(S+I)
 80     NEXT I
 90     POKE S+36,J:POKE S+37,K:POKE
        S+38,L:POKE S+39,M
100     GOTO50
```

10- S je adresa SM
20- smycka FOR-NEXT cte pro kazdy obrazovkovy radek (0-39) datove byty v SM pametovych bunkach, ktere vydaji zadany text.
30,40- hodnoty znaku musi odpovidat internimu kodu.
50- obsah prvnich 4 adres se ulozi do promennych,
60-80- pak jsou obsahy zbylych bunek obrazoveho radku prelozeny o 4 adresy nahoru, tzn. na obrazovce 4 sloupce doleva a
90- konecne se ulozi (POKE) 4 zachovane hodnoty znaku do 4 zadnich mist radku.

Timto pochodem se pismena obrazovkoveho radku posunou pri kazdem prubehu programu o 4 sloupce doleva. Jeste dulezitejsi je prime adresovani obrazovky pro textove okenko, protoze tam je prikaz

POSITION neucinny.
S SM-POKE lze znaky umistit libovolne a spolehlive a tak lze bezne vydavat ruzne stridave hodnoty z jedineho mista (napr.citac):

```
  0     REM TWPOKE
  1     REM ******************
  2     REM *                *
  3     REM * POKE-TEXT.OK. *
  4     REM *                *
  5     REM ******************
 10     DIM Z$(5):P=40897:? CHR$(125):
        N=25
 20     FOR J=0 TO 4:POKE P+J,16:NEXT J
 30     Z=Z+1
 40     Z$=STR$(Z)
 50     L=LEN(Z$)
 60     FOR J=1 TO L
 70     POKE P+4-L+J,VAL(Z$(J,J))+16
 80     NEXT J
 90     IF PEEK(P)=N THEN IF PEEK (P+1)
        =N THEN IF PEEK (P+2)=N THEN
         IF PEEK(P+3)=N THEN IF PEEK (
        P+4)=N THEN GOSUB 200
100     REM IF Z=99999 THEN GOSUB 200
110     GOTO 30
200     RESTORE 300
210     FOR J=0 TO 8
220     READ D
230     POKE P-10+J,D
240     POKE P+6+J,D
250     NEXT J
260     Z=0:FOR J=0 TO 4
270     POKE P+J,16
280     NEXT J
290     RETURN
300     DATA 47,54,37,50,38,44,47,55,1
```

10- Z$ prejima sled cislic o stavu citace. Levy horni roh textoveho okenka lezi pri vsech standardnich DL na adrese 40800.

20- na pet mist citace se zapisi nuly (interni kod=16).

40- aby se mohl stav citace ukladat (POKE) na obrazovku, musi se Z rozlozit na jednotlive cislice. Prvni krok spociva v premene numericke promenne Z na string Z$, protoze z retezce lze jednotlive prvky lehce vycist.

50- k umisteni jednotlivych cislic na spravnem miste ve vydanem cisle je nutne nejdriv stanovit, kolik mist bude mit cislo, kolik prvku bude mit Z$.

60-80- smycka FOR-NEXT cte z retezce cisel kazdou jednotlivou cislici (Z$(J,J)), prevede ji na numerickou hodnotu (VAL) a zmeni cislici prictenim 16 do jejiho interniho znakoveho kodu. Interni kod 1=17, 2=18 atd. Takto sdelena hodnota znaku se ulozi na misto P+4-L+J. Tak prijde kazda cislice na spravne misto.

90- take takovy prikaz ATARI zvladne! Stanovi, zda jsou vsechny cislice rovne 9 (int.kod=25).

100- prirozene toho lze dosahnout jednoduseji.
110- uzavira smycku zpetnym skokem na 30.
200-300-kdyz citac napocita 99999, pak by zde... ne, vyzkousejte si to sami. Nez citac uplne probehne, bude to chvili trvat.

Pri preskoku z 99999 na 00000 zpozorujete, ze citac bezi pri malych cislech mnohem rychleji. To proto, za retezec cislic je kratsi a tak je potreba mnohem mene programovych kroku.

U textoveho okenka je jeste jedna zvlastnost. Ma totiz vlastni nezavislou obrazovkovou pamet. Vektor TXTMSC (660,661=$294,$295) ukazuje na startadresu, ktera je pri vsech standardnich DL 40000.
Nezalezi na tom, zda zvolite graficky mod s nebo bez textoveho okenka. Pro graficke okenko je vzdy rezervovano misto v pameti, ktere muze prijmout data pro cely obsah obrazovky. Za nim lezi 160 bytu velka pamet pro textove okenko. Tato nezavislost ma take za nasledek neucinnost PLOT a POSITION. Na druhe strane toto deleni uzavira moznost popsat pamet textoveho okenka, ackoliv toto treba vubec neni zapojeno, a take moznost zakryt nebo odkryt textove okenko v grafickem okenku, aniz by se pritom jedno nebo druhe neztratilo.
Tohle je ovsem mozne jen tehdy, kdyz je pri usporadani prikazu GR. pamatovano na to, aby se nemazal obsah obrazovkove pameti tak, jak je to obvykle. To se stane nastavenim bitu 5.

bit 7 nepouzito
bit 6 nepouzito
bit 5 (=64) obrazova pamet nemazana
bit 4 (=16) bez text. okenka
bit 3 az 0
(=15-0) gr. mod 0-15 s text. oknem

Programove kroky pro zakryti nebo odkryti textoveho okenka vam ukaze nasledujici priklad:

```
1   REM  TWZAP
2   REM ******************************
3   REM * ZAPINANI/VYPINANI TEXT.OKNA *
4   REM ******************************
10  GOSUB 300
20  Z=Z+1 Z$=STR$(Z):L=LEN(Z$)
30  FOR J=1 TO L:
    POKE P+6-L+J,VAL(Z$(J,J))+16:NEXT J
40  IF PEEK(764)=255 THEN 20
100 OPEN #1,4,0,"K:":GET #1,T:CLOSE #1
110 YA=Y:Y=Y-((T=28) AND (Y>0))
120 Y=Y+((T=29) AND (T<23))
130 XA=X:X=X-((T=30) AND (X>0))
140 X=X+((T=31) AND (X<39))
150 IF T=69 THEN GR.3+32: POKE 708,36:POKE709,38:
    POKE 710,208:POKE 712,208
160 IF T=65 THEN GR.3+16+32:POKE 708,50
    :POKE 709,52:POKE 710,208:
    POKE 712,208
200 COLOR 0:PLOT 0,YA:DRAWTO 39,YA:
    PLOT XA,0:DRAWTO X,23:GOTO 20
```

```
210 COLOR 1:PLOT 0,Y:DRAWTO 39,Y:
    COLOR 2:PLOT X,0:DRAVTO X,23:
    GOTO 20
300 DIM Z$(7):P=40896
310 GR.19:POKE 708,50:POKE709,52:
    POKE 710,208:POKE 712,208
320 FOR J=0 TO 6:POKE P+J,16:NEXT J
330 COLOR 2:PLOT 0,0:DRAWTO 0,23
340 COLOR 1:PLOT 0,0:DRAWTO 30,0
350 RETURN
```

10- není vždy potřeba usporadat program chronologicky

20-30- obsahují stejné příkazy jako výše diskutovaný čítač, který byl volně prodloužen na 7 míst a přebudován na méně logických řádku. Program běží tím rychleji, čím méně má logických řádku a to v přímé závislosti skokových příkazu, protože BASIC začne při každém skoku od nejnižší řádky, prohlíží je, až najde ten pravý. Proto by měly být všechny často odkazované řádky co nejblíže začátku programu.

40- CH(764=$2FC) obsahuje klávesnicový kód naposledy stisknuté klávesy. Když není stisknuta žádná, je zde 255. Tento řádek se stará o neruseny průběh programu, protože

100- příkaz GET přerušuje program a čeká na zadání uživatele. Přitom stojí také čítač.

110-140- v grafickém okénku se nachází nitkový kříž. Čtyřmi klávesami kurzoru jím lze pohybovat do všech čtyř směru. XA a YA zachovávají původní hodnoty, aby se pozdeji původní pozice kříže mohla smazat. Čtyři booleovské rovnice stanoví, zda je dotýcna klávesa (T) stisknuta a hraniční hodnota X popř. Y jeste není dosažena a pak aktualizují odpovídající hodnoty X popř. Y.

150- stisknutím klávesy "E" se aktivuje GR.3 (s textovým okénkem) +32 zároveň zaridí, aby se obrazovka nesmazala (textové okénko je odcloněno). Lze též místo 3+32 napsat GR.35 .

Každý příkaz GR. přitom obnovuje dafault hodnoty v barvových registrech. Proto zde musí být definované barvy obnoveny. Pomocí stejných barvových hodnot pro pozadí (712) a COLOR (710) se nakonec dosáhne toho, že grafické a textové okénko tvorí opticky stejný jas, což působí jako zacloneni. Jeste účelnejší by bylo zužení textového okénka. Protože by se k tomu musel změnit DL, mohl by tento zacloněný řádek být na stínítku kdekoliv.

160- stisknutím klávesy "A" bude s poznávacím číslem (GR.) 3+16+32 (=51) textové okénko opet odcloněno. Při zacloněni nebo odcloněni dostane kurzor lehce odlišnou barvovou hodnotu, aniž by musel být barvový registr nové popsán a je-li čítač zaclonen. Důležité je, že při zaclonenem textovém okénku se může střed kříže pohybovat na dolním okraji obrazovky, tedy tam, kde zrovna rozšiřuje textové okénko, aniž by došlo k hlášeni chyby o náhodné pozici kurzoru. To ale pracuje jen tehdy, když byl na začátku vyvolán grafický mód bez textového okénka. Když pak pozdeji bude text. okénko zacloněno, mohou přesto PLOT atd.popsat grafické okénko v oblastech obsazených textovým okénkem, protože stejné není nic vídet.

200- maze puvodni kriz s barvou pozadi a radek
210- PLOTuje aktualni kriz.
300-350-prikazy pouzite pouze jednou na zacatku programu je nejlepe na konci programu opet nastavit. Zde je definovana grafika a barvove hodnoty, sedm mist citace se zaplni nulami a jsou PLOTovany obe osy krize.

Nasledujici tabulka udava priklad ruznych gr.modu:

	GR.	ANTIC POVEL	SLOUPCE PRO PIXEL/ RADEK	MOD RADKY	BYTY PRO RADKY	TV PRO RADKY MOD RADKY	mBITY PRO PIXELY
CHAR.	0	2	40	20/24	40	8	8
	-	3	40	16	40	10	8
	12	4	40	20/24	40	8	8
MOD	13	5	40	10/12	40	16	8
	1	6	20	20/24	20	8	8
	2	7	20	10/12	20	16	8
BITA	3	8	40	20/24	10	8	2
	4	8	80	40/48	10	4	1
	5	10	80	40/48	20	4	2
MAPY	6	11	160	80/96	20	2	1
	14	12	160	160/192	20	1	1
	7	13	160	80/96	40	1	2
	15	14	160	160/192	40	1	2
MODY	8	15	320	160/192	40	1	1
GTIA	9		80	192	40	1	4
	10		80	192	40	1	4
	11		80	192	40	1	4
MODY							

* ve vsech GR. s textovym okenkem urcuje cislo 710 barvu pisma a pozadi a 709 svetlost pisma text. okenka. C=COLOR, R=RAND (OKRAJ), H=HINTERGRUND (POZADI).
** barvova hodnota registru 712 je ve vsech GR.

vyvolana s COLOR 0.

11.0 SADA ZNAKU

Aby mohl pocitac na obrazovku napsat informaci, musi vedet, jak se pisi znaky. On sam se omezuje pouze na cisla.
Uplatnuji se hned tri ruzne systemy prirazeni ciselnych hodnot k sade znaku na ruznych urovnich systemu.

Jednim z nich je klavesnicovy kod, ktery prirazuje kazde klavese na klavesnici numerickou hodnotu 0-63. Druha polovina sady znaku je dosazitelna tlacitkem SHIFT, jenze u ATARI se zakladnim stavem klavesnice se vydavaji jen velka pismena. Pomoci CAPS lze prepnout na mala pismena (pro velka je pak nutno stisknout SHIFT). Pro umozneni trojnasobneho obsazeni klavesy se pak u pocitace zavedla dalsi funkcni klavesa CONTROL (CTRL). Treti klavesa pusobici na vzhled znaku je INVERS. Je-li stisknuta, znaky se invertuji, tj. zobrazi se negativne.

Se zavedenim dalnopisu nastoupila jednotna norma pro prenos znaku , ktera se jmenuje ASCII. Prirazuje kazdemu znaku a kazde funkci dalnopisu ciselnou hodnotu 0-127. Tento kod tvori dnes zaklad pro pocitace a tiskarny. Teto americke norme byly pozdeji prideleny jeste mezinarodni znaky (a, a, o) , ktera ale v prvnich 128 znacich, tedy ve vlastni sade znaku nemaji misto. Prvnich 32 znaku ASCII kodu slouzi k funkcnimu ovladani tiskrny. Tyto ridici znaky pocitac nepotrebuje, protoze pracuje pres monitor. Toto volne misto je u ATARI vyuzito k dosazeni dodatecnych znaku (pseudo-, semi-, block-). K odliseni od americkeho se tento kod nazyva ATASCII. Z velke casti je identicky s ASCII normou.

Tretim systemem je interni kod, ktery urcuje poradi , v jakem jsou data 128 znaku zakotvena v ROM pameti. Od ATASCII kodu se lisi tim, ze cely soubor znaku je rozdelen do tri velkych bloku. Pomer mezi ATASCII a internim kodem si muzete predstavit takto:

prevod z ATASCII hodnot do intern.kodu

```
 0- 31 a 128-159  ATASCII + 64
32- 95 a 160-223  ATASCII - 32
96-127 a 224-255  identicke
```

prevod z interniho kodu na ATASCII

```
 0- 63 a 128-191  interni kod + 32
```

64- 95 a 192-223 interni kod - 64
96-127 a 224-255 identicke

Nulty znak interniho kodu je prazdny znak (ATASCII 32). S daty pro prazdny znak tedy zacina sada znaku. Kazdy znak je znazornen v matrici o 8*8 bodech. Kazdy jednotlivy bod sviti nebo nesviti, coz se zapamatuje nastavenim (nenastavenim) jednoho bitu. Osma za sebou lezicich bitu se sdruzuje do jednoho datoveho bytu. Kazdy znak je pak vysoky 8 bytu:

```
 128  64  32  16  8   4   2   1
---------------------------------
!   !   !   !   !   !   !   !   ! =  0
---------------------------------
!   !   ! * ! * !   ! * ! * !   ! = 54
---------------------------------
!   ! * ! * ! * ! * ! * ! * ! * ! =127
---------------------------------
!   ! * ! * ! * ! * ! * ! * ! * ! =127
---------------------------------
!   !   ! * ! * ! * ! * ! * !   ! = 62
---------------------------------
!   !   !   ! * ! * ! * !   !   ! = 28
---------------------------------
!   !   !   !   ! * !   !   !   ! =  8
---------------------------------
!   !   !   !   !   !   !   !   ! =  0
---------------------------------
```

Srdicko si tedy pocitac pamatuje jako radu 8 bytu s decimalnimi hodnotami 0,54,127,127,62,28,8,0. Ma ATASCII hodnotu 0 a interni kod 64. Tzn. ze lezi v pameti sady znaku na 65. miste. Protoze kazdy z 64 bytu pred tim obsazuje take 8 bytu, lezi prvni byte srdicka v (64*8)-te bunce za zakladni adresou sady znaku. Interni kod tedy urcuje posunuti dat jednoho znaku v pameti znaku. HI-byte startadresy sady znaku lezi v CHBAS(756=$2F4). K nalezeni dat urciteho znaku pocitame:
CHBAS*256+interni kod znaku*8
Data hledaneho znaku lezi v teto a v sedmi nasledujicich adresach.
K sestaveni vlastni sady znaku je nutne pro kazdy ze 128 znaku prepocitat vzor o 8*8 bodech na datove byty a ulozit techto 8*128=1024 bitu do pameti. Jestlize pak nastavime ukazatel CHBAS na start adresu teto alternativni sady znaku, pak se na obrazovce objevi pri PRINT nebo listingu nove znaky. Jak to vypada, ukazuje nasledujici program, ktery obsazuje ve forme DATA radku sadu znaku jako kurziva, tzn. ze jejich svisla osa je sklonena doprava, takze se podobaji rukopisu:

```
0 REM ITALIC.DAT
1 REM ***************
2 REM *             *
3 REM * KURZIVA-DATA *
```

```
    4 REM *               *
    5 REM *****************
   10 POKE 106,PEEK(106)-8
   20 GRAPHICS 0
   30 A=PEEK(106)+8:C=A*256
   40 FOR J=0 TO 1023:READ B:POKE C+J,
      B:NEXT J
   50 ? CHR$(125):POKE 752,1
  100 POKE 756,A
  110 POSITION 2,7
  120 ? "TAK VYPADA NOVE ZOBRAZENI"
  130 FOR W=0 TO 400:NEXT W
  140 POSITION 2,7
  150 ? "TAK VYPADA STARE ZOBRAZENI"
  160 POKE 756,224
  170 FOR W=0 TO 400:NEXT W
  180 GOTO 100
30000 DATA 0,0,0,0,0,0,0,0,0,12,12,24,
      24,0,48,0,0,51,51,102,0,0,0,0
30001 DATA 0,54,127,54,108,254,108,0,
      12,62,96,56,12,248,48,0,0,108,104
      ,24,16,54,38,0
30002 DATA 24,52,24,40,74,68,54,0,0,24,
      24,48,0,0,0,0,0,0,14,28,24,24,28,14,0
30003 DATA 0,112,56,24,24,56,112,0,0,
      102,60,255,60,102,0,0,0,8,24,126,
      24,16,0,0
30004 DATA 0,0,0,0,0,0,24,24,48,0,0,0,126
      ,0,0,0,0,0,0,0,0,0,24,24,0
30005 DATA 0,6,12,24,48,96,64,0,0,28,50
      ,110,118,76,56,0,0,6,28,12,24,24,
      48,48
30006 DATA 0,30,50,12,24,48,124,0,0,31,
      6,8,4,4,40,48,0,6,12,28,40,126,24
      ,48
30007 DATA 0,62,48,56,4,4,76,48,12,16,
      32,120,100,100,56,0,0,60,4,12,24,
      48,48,48
30008 DATA 12,18,50,56,76,76,56,0,0,28,
      54,34,22,12,24,48,0,0,24,24,0,48,
      48,0
30009 DATA 0,0,24,24,0,48,48,96,6,12,24
      ,48,24,12,6,0,0,0,126,0,0,126,0,0
30010 DATA 96,48,24,12,24,48,96,0,0,28,
      34,12,24,16,0,32,0,60,54,110,108,
      64,124,0
30011 DATA 0,15,27,50,102,252,204,0,0,
      62,50,124,102,198,252,0,0,30,50,
      32,96,100,56,0
30012 DATA 0,60,54,102,102,204,248,0,0,
      62,48,120,96,192,248,0,0,62,48,
      120,96,192,192,0
30013 DATA 0,30,48,96,110,204,120,0,0,
      51,51,126,102,204,204,0,0,30,12,
      24,24,48,248,0
30014 DATA 0,30,38,12,12,24,152,112,0,
      51,54,104,120,216,204,0,0,24,24,
      48,48,96,124,0
30015 DATA 0,33,51,127,107,198,198,0,
```

```
      0,51,59,106,110,204,196,0,0,30,51
      ,102,102,204,120,0
30016 DATA 0,30,19,54,60,96,96,0,0,30,
      51,102,102,216,108,6,0,62,51,102,
      124,216,204,0
30017 DATA 0,30,48,60,6,140,120,0,0,63,
      76,24,24,48,48,0,0,51,99,102,198,
      204,120,0
30018 DATA 0,51,35,102,100,104,48,0,0,
      33,99,67,203,222,228,0,0,34,22,28
      ,24,52,100,0
30019 DATA 0,17,35,22,28,24,48,96,0,62,
      6,24,48,192,248,0,0,30,24,48,48,
      96,120,0
30020 DATA 0,96,96,48,24,12,12,0,0,30,6
      ,12,12,24,120,0,0,8,28,54,99,0,0,
      0
30021 DATA 0,0,0,0,0,0,255,0,0,54,127,
      127,62,28,8,0,24,24,24,31,31,24,
      24,24
30022 DATA 3,3,3,3,3,3,3,3,24,24,24,248
      ,248,0,0,0,24,24,24,248,248,24,24
      ,24
30023 DATA 0,0,0,248,248,24,24,24,3,7,
      14,28,56,112,224,192,192,224,112,
      56,28,14,7,3
30024 DATA 1,3,7,15,31,63,127,255,0,0,0
      ,0,15,15,15,15,128,192,224,240,
      248,2,52,254,255
30025 DATA 15,15,15,15,0,0,0,0,240,240,
      240,240,0,0,0,0,255,255,0,0,0,0,0
      ,0
30026 DATA 0,0,0,0,0,0,255,255,0,0,0,0,
      240,240,240,240,0,28,28,119,119,
      8,28,0
30027 DATA 0,0,0,31,31,24,24,24,0,0,0,
      255,255,0,0,0,24,24,24,255,255,24
      ,24,24
30028 DATA 0,0,60,126,126,126,60,0,0,0,
      0,0,255,255,255,255,192,192,192,
      192,192,192,192,192
30029 DATA 0,0,0,255,255,24,24,24,24,24
      ,24,255,255,0,0,0,240,240,240,240
      ,240,240,240,240
30030 DATA 24,24,24,31,31,0,0,0,120,96,
      120,96,126,24,30,0,0,24,60,126,24
      ,24,24,0
30031 DATA 0,24,24,24,126,60,24,0,0,24,
      48,126,48,24,0,0,0,24,12,126,12,
      24,0,0
30032 DATA 0,24,60,126,126,60,24,0,0,0,
      28,6,62,204,122,0,0,48,48,124,102
      ,204,248,0
30033 DATA 0,0,60,96,96,192,120,0,0,3,3
      ,62,102,204,124,0,0,0,28,102,124,
      192,120,0
30034 DATA 0,14,24,60,24,48,48,96,0,0,
      31,54,102,60,12,120,0,48,48,124,
      102,198,204,24
```

```
30035 DATA 0,12,0,56,24,48,120,0,0,6,0
      12,12,24,24,240,0,48,32,108,120,
      208,204,0
30036 DATA 0,28,12,24,24,48,120,0,0,0,
      51,127,122,214,132,0,0,0,62,99,
      103,198,204,0
30037 DATA 0,0,30,102,102,204,120,0,0,0
      ,62,102,102,248,192,192,0,0,31,
      102,102,60,12,12
30038 DATA 0,0,26,48,48,96,96,0,0,0,31,
      96,60,12,248,0,0,12,63,24,24,48,
      48,0
30039 DATA 0,0,51,102,102,204,124,0,0,0
      ,51,102,102,120,48,0,0,0,35,107,
      87,252,216,0
30040 DATA 0,0,51,60,24,124,196,6,0,0,
      51,98,102,60,24,240,0,0,63,12,24,
      96,252,0
30041 DATA 0,24,60,126,126,24,60,0,24,
      24,24,24,24,24,24,24,0,126,120,
      124,110,102,6,0
30042 DATA 8,24,56,120,56,24,8,0,255,
      255,255,255,255,255,255,255
```

10- Je dulezite ulozit tuto sadu znaku do chraneneho pametoveho rozsahu.
Vektor RAMTOP (106=$6A) bude prelozen o 8 stranek dolu. Vlastne budou pouzity jen 4 stranky (= 1 kbyte) , ale je jistejsi rezervovat k pameti textoveho okenka nejaky buffer, aby se data sady znaku pri rolovani naprepsala.(jen#).

20- GR.prikaz zpusobi, ze RAMTOP priradi obrazovkove pameti a DL novou hodnotu. Nova sada znaku ostatne funhuje v GR.1 nebo v GR.2 nebo ve vsech grafickych modech s textovym okenkem. Musite ale startadresu alternativni sady znaku (jen HI byte) znovu ulozit (POKE) do adresy 765, protoze pri kazdem prikazu GR. se v CHBAS obnovuje default hodnota.

30- zde se obe pouzite hodnoty prepocitaji (stranka, popr. HI byte a uplna startadresa nove sady znaku.

40- smycka FOR-NEXT precte vsech 1024 bytu z DATA radku v rezervovanem pametovem rozsahu.

50- obrazovka se vymaze a kurzor zmizi.

100- stanovi se ukazatel na novou sadu znaku a

120- pise se na monitor:

150- pak my piseme jiny text a

160- opet se nastavi ukazatel na ROM- chranenou sadu znaku.

30000-30042-tak to je ona sada znaku, byte po bytu, a tak vypadaji nove a standardni ATARI znaky na obrazovce:

Znakove mody 12 a 13 , ktere u XL mohou byt vyvolany take pres GR., funguji v principu uplne stejne. Rozdil je v tom, ze vzdy 2 bity vytvori 1 pixel znaku. Tak lze rozeznat celkem 4 COLOR hodnoty, ale 1 byte obsahuje informace pro 4 pixely. Abychom se dostali na stejnou displayovou sirku, jsou pixely siroke 2 obrazove body.
COLOR hodnoty se vztahuji k barvovym registrum 708 (COLOR1=01), 709 (2=10), 710 (3=11) a pozadi 712 (0=00) pri normalnich

znacich. Inverze ziskavaji svou barvovou hodnotu pro COLOR3 z barvoveho registru 711.

Maly priklad definovani novych znaku:

```
  0  REM COLCHR.DAT
  1  REM ****************
  2  REM *              *
  3  REM * BAREVNE ZNAKY *
  4  REM *              *
  5  REM ****************
 10  GOSUB 100
 20  X=INT (RND(0)*39)
 30  Y=INT (RND(0)*21)
 40  Z=INT (RND(0)*11)
 50  J=Y*40+X+1:Z=Z+1
 60  P$(J,J)=C$(Z,Z)
 70  POSITION 0,0:?#6;P$:GOTO 20
100  DIM P$(880),C$(11):C$="#$%_'()
     *+,-"
110  P$="#":P$(880)=P$:P$(2)=P$
120  POKE 106,PEEK(106)-4
130  GRAPHICS 28
140  A=PEEK(106)+4:C=A*256
150  FOR J=0 TO 111:READ B:POKE C+J,
     B:NEXT J
200  DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,0,0,0
210  DATA 66,66,24,24,36,36,129,129,78
     ,78,116,116,39,39,177,177,112,112
     ,27,27,228,228,141,141
220  DATA 74,74,16,16,4,4,161,161,66,
     66,16,16,132,132,129,129,66,66,28
     ,28,180,180,129,129
230  DATA 74,74,28,28,52,52,161,161,74
     ,74,16,16,132,132,129,129,66,66,
     18,18,4,4,161,161
240  DATA 66,66,30,30,52,52,161,161,74
     ,74,28,28,180,180,129,129
250  POKE 756,A
260  RETURN
```

10- zacatek programu lezi opet ve spodnich radcich, aby pracujici cast bezela rychleji.

100- P$ prevezme obrazovkovou grafiku jako string. Pri 40 znacich pro zpusobovy radek se tim pokryje 22 radku. C$ prevezme sadu znaku, ktere se mohou pouzit take jako prvky obrazu.

110- naplni P$ (dle programu ADR135A) s 880 ciselnymi znaky. Zde je mozne take nastavit kazdy dalsi znak c$ nebo rady techto znaku.

120-150- rezervuji pametove misto pro novou sadu znaku, ktera nebude prilis velika, protoze bylo stanoveno jenom 12 novych znaku.

200- DATA novych znaku. V poradi interniho kodu lezi prazdny znak na nultem miste a je nasledovan vykricnikem a uvozovkami. Protoze uvozovky nemohou byt v ATARI BASICu

prijaty do retezce, prvni tri znaky se zde pro tento ucel nepouzivaji. Teprve nasledujicich 11 znaku se pouziva jako obrazove prvky. Proto cte smycka FOR-NEXT v radku 150 ctrnact (14 * 8 = 112) dat.

250- stanovi ukazatel CHBAS a

260- pak jde dal.

20,30- kdyz se do stringu zanese jako znak take obsah obrazovkove grafiky, pracuje se jako s obrazovymi pozicemi X a Y. Protoze retezec se vytiskne na obrazovku vzdy se zacatkem ve stejnem miste, lezi jednotlive obrazove prvky take na stale stejnem miste obrazovky. P$ zde neni nic jineho nez obrazovkova pamet obsahujici poznavaci cisla, s jejichz pomoci se provadeji komplexni barevne graficke sestavy (nami sestavene znaky), ktere se pak vytisknou na obrazovku.
Stejne si lze predstavit funkce obrazovkove pameti ve znakovem modu. V pametovych bunkach jsou ulozeny hodnotu znaku v internim kodu, s jehoz pomoci se znazorni na obrazovce bitova kombinace ctena ze sady znaku ROM.
Velka vyhoda stringove metody je v tom, ze se nemusime starat o rezervovano pametoveho mista.
To, co prinesou na obrazovku definovane obrazove prvky, by bylo mozne take v GR.15 (XL), ale jednotlive PLOTovani je namahave a spotrebuje mnoho casu. Zde se jednim prvkem (znakem) naplni plocha 8*8 obrazovych bodu (4*8 pixelu). Omezeni spociva v tom, ze obrazovkova grafika muze byt sestavena pouze jako patchwork , tzn. z predem stanovenych vzoru.
Stringova metoda tedy zjednodusuje sestavovaci praci (patchwork) barevne grafickych obrazovkovych obrazu pri nepatrne spotrebe pameti a relativne vysoke pracovni rychlosti.
Protoze lze definovat celkem 128 obrazovych bodu, je mozne definovat obecnou stavebnici grafickych dilu, ze kterych lze rychle sestavit cokoliv chceme (krajina, kaleidoskop...).
V obou techto programovych radcich jsou tedy sdeleny nahodne pozice na obrazovce, X a Y.

40- a zde se vyvola nahodne cislo 0-10, ktere lze pozdeji zvolit jako sestavovaci prvek.

50- J vypocita X a Y pozici v retezci tak, jak byl z X a Y sdelen radek v obrazovkove pameti, totiz hodnota radku (Y) * pocet znaku pro radek (40) + hodnota sloupce (X). Protoze pracujeme s retezcem a prvni znak retezce ted nema hodnotu 0 ale 1, musi se k nalezene hodnote pricist 1. To plati tez pro nahodne cislo Z. String nema nulty znak.

60- znak, ktery je v P$ na vypocitanem miste J (P$(J,J)), se nahradi nahodnym cislem Z, ktere stoji v C$ na miste Z.

70- tiskne P$ na obrazovku od leveho horniho rohu a zacne novy beh.
Pokud ma program slouzit pouze pro vytvoreni pestreho promenliveho obrazu, pak neni nutne zarazovat do stringu nahodne obrazove prvky a tisknout pri kazdem prubehu kompletni P$. Stejneho ucinku se dosahne oznacenim nahodneho obrazovkoveho mista pomoci POSITION a vytistenim obrazoveho prvku C$(Z,Z) od tohoto mista.

Zde uvedena metoda si pamatuje aktualni obsah obrazu, ktery je tedy nezavisly na svem zobrazeni na monitoru. Zmeny take mohou probihat, zatimco je na obrazovce neco jineho. Program muze take prepinat mezi P$ a ostatnimi retezci s obrazovymi obsahy a tim rychle stridat sceny.

12.0 SEZNAM DULEZITYCH ADFRES

APPMHI	14	GRACTL	53277	TMPCHR	80
ARGOPS	128	GRAFM	53265	TMPCOL	679
ATACHR	763	GRAFP0-3	53261	TMPLBT	673
ATRACT	77	HATABS	794	TEPROW	696
AUDC1	53761	HITCLR	53278	TRAMSZ	6
AUDC2	53763	HLPFLG	732	TSTAT	793
AUDC3	53765	HOLDCH	124	TSTDAT	7
AUDC4	53767	HOLD1	81	TXTCOL	657
AUDCTL	53768	HPOSM0-3	53252	TXTMSC	660
AUDF1	53760	HPOSP0-3	53248	TXTOLD	662
AUDF2	53762	ICAX1Z-6Z	42	TXTROW	656
AUDF3	53764	ICBALZ/HZ	36	VBREAK	518
AUDF4	53766	ICBLLZ/HZ	40	VDELAY	53276
BFENLO/HI	52	ICCOMT	23	VDSLST	512
BITMSK	110	ICCOMZ	34	VIMIRQ	534
BOOT?	9	ICDNOZ	33	VINTER	516
BOTSCR	703	ICHIDZ	32	VKYBD	520
BPTR	61	ICPTLZ/HZ	38	VNDT	132
BRKKEY	17	ICSTAZ	35	VNTP	130
BRKKY	566	INBUFF	243	VPRCED	514
BUFADR	21	INITAD	738	VSERIN	522
BUFCNT	107	INSDAT	125	VSEROC	526
BUFRFL	56	INTEMP	557	VSEROR	524
BUFRLO/HI	50	INTRVEC	552	VTIMR1-4	528
BUFSTR	108	INVFLG	694	VVBLKD	548
CASFLG	783	IOCB0-7	832	VVBLKI	546
CASINI	2	IRQEN	53774		
		KEYDEF	121	VVTP	134
CASSBT	75	KEYDEL	729	WARMST	8
CBAUD	750	KEYDEL	753	XMTDON	58
CDTMA1-2	550	KEYREP	730		
CDTMF3-5	554	LCOUNT	563		
CDTMV1-5	536	LINBUF	583		
CFB	570	LINZBS	0		
CH	764	LMARGIN	82		
CH1	754	LOGCOL	99		
CHACT	755	LOGMAP	690		
CHCTL	54273	LOMEM	128		
CHAR	762	LPENH/V	564		
CHBAS	756	MEMLO	743		
CHBASE	54281	MEMTOP	144		
CHKSNT	59	MEMTOP	741		

CHKSUM	49	NEWCOL	97
CIX	242	NEWROW	96
CKEY	74	NOCKSM	60
COLAC	114	NOCLIK	731
COLBK	53274	OLDADR	94
COLCRS	85	OLDCHR	93
COLDST	580	OLDCOL	91
COLOR0-4	708	OLDROW	90
COLPF0-3	53270	OUTBUFF	128
COLPM0-3	53266	PADDL0-7	624
COLRSH	79	PBPNT	29
CONSOL	53279	PBUFSZ	30
COUNTR	126	PCOLR0-3	704
CRETRY	54	PMBASE	54279
CRITIC	66	PRIOR	53275
CRSINH	752	PRNBUF	960
DAUX1-2	778	PTABW	201
DBSECT	577	PTEMP	31
DBUFLO/HI	772	PTIMOT	28
DBYTLO/HI	776	PTRIG0-7	636
DCB	768	RADFLG	251
DCOMND	770	RAMLO	4
DDEVIC	768	RAMSIZ	740
DEGFLG	251	RAMTOP	106
DELTAC	119	RANDOM	53770
DELTAR	118	RECVDN	57
DINDEX	87	RMARGN	83
DLISTL/H	54274	ROWAC	112
DMACTL	54272	ROWCRS	84
DMASK	672	ROWINC	760
DOSINI	12	RTCLOK	18
DOSVEC	10	RUNAD	736
DRETRY	55	RUNSTK	142
DRKMSK	78	SCRFLG	699
DSKFMS	24	SDLSTL	560
DSKTIM	582	SDMCTL	559
DSKUTL	26	SHFAMT	111
DSPFLG	766	SHFLOK	702
DSTAT	76	SOUNDR	65
DSTATS	771	SRTIMR	555
DTIMLO	774	SSFLAG	767
DUNIT	769	SSKTCL	562
DUNUSE	775	STACKP	792
DVSTAT	746	STARP	140
ENDPT	116	STATUS	48
ENDSTAR	142	STICK0-3	632
ERRSAVE	195	STMCUR	138
ESCFLG	674	STMTAB	136
ESIGN	239	STOPLN	186
FEOF	63	STRIG0-3	644
FILDAT	765	STRTFLG	1001
FILFLG	695	SWPFLG	123
FMZSPG	67	TABMAP	675
FREQ	64	TIMER1	780
FTYPE	62	TIMER2	784
GLBABS	736	TIMFLG	791
GPRIOR	623	TINDEX	659

13.0 ABC O POCITACI

ATASCII

Kod, ktery kazdemu znaku prirazuje dec.hodnotu 0-127. ATASCII kod je modifikace ASCII normy (Americen Standard Code for Information Interchange).

BCD

Pouzivaji se dve odlisne metody zaznamu numerickych hodnot. Pri integer metode se prevadeji decimalni hodnoty na binarni zpusob zapisu. Tato data se zpracovavaji velmi rychle.Druhou cestou je binarni kodovani (Binary Coded Decimals). Pri 6bytovych BCD konstantach, ktere pouziva ATARI, se pro kazdou ciselnou hodnotu stanovi rada 6 bytu. Prvni obsahuje znamenko a exponent , dalsich pet obsahuje cislice desitkoveho cisla v BCD forme. Protoze se ke znazoreni cislic 0-9 pouziva 4 bitu, muze kazdy BCD byte obsahovat dve cislice, takze 6bytova BCD konstanta muze obsahovat desetimistne cislo. Tim se obsadi relativne velke pametove misto a pocitaci procesy probihaji zretelne pomaleji (viz STARP 140,141=$8C,$8D).

BIT

Binarni cislice (BInary digiT) je nejmensi jednotka, se kterou pocitac pracuje. Binarni soustava ma zaklad 2, tzn., ze obsahuje pouze dve cislice , 0 a 1 , a kazde misto binarniho cisla predstavuje mocninu dvou. Pocitac pozna cislice 0 a 1 jako stavy elektrickeho napeti vypnuto - zapnuto. Bit je nejmensi jednotka informace. Obsahuje rozhodnuti ano - ne.

BOOT

Zavedeni programu ve strojovem kodu do pameti z kazety ci diskety a jeho spusteni. Casteji bootstrapping - uziti existujici jednoduche verze programu k zavedeni dokonalejsi verze.

BUFFER

Vyrovnavaci pamet.

BYTE

Sdruzeni osmi bitu, ktere muze predstavovat decimalni hodnotu 0-255. Osmibitovy procesor, t.c. standard pro domaci pocitace, zpracovava byte najednou, tzn. 8 bitu paralelne.

CIO

Central Input/Output, centralni rutina pro vstup a vystup, ktera hlida veskere tyto procesy.

DEFAULT

Implicitni (standardni) hodnota, kterou obsahuji dane registry (pametove bunky) po inicializaci operacniho systemu.

DISPLAY LIST

Display List, program ve strojovem kodu pro mikroprocesor ANTIC, ktery je provozovan s vlastni sadou instrukci. DL urcuje, v jakym zpusobem se vydaji data z pocitace na TV.

DLI

Display List Interrupt.Zatimco katodovy paprsek obrazovky skace z praveho okraje na zacatek dalsiho radku (HBLANK=horizontalni blank), muze se zpracovani DL prerusit, aby se zpracovaly dalsi programove casti.

DOS

Disk Operating System,software, ktery je nahran z diskety a prebira koordinaci mezi disketovou jednotkou a pocitacem.

DUP

Disk Utility Package,software urcitych DOS prikazu, napr.COPY apod.

DRIVER

Manipulacni program (handler), koordinuje pocitac s periferiemi, jako "E:" ditor, "K:"eyboard, "P:"rinter, "D:"iskette.

EOF

End of File, oznacuje konec jednoho souboru.

EOL

End of Line, oznacuje konec logickeho radku.

FMS

File Manager System, cast DOSu ridici I/O operace "D:".

FP

Floating Point, oznaceni cisel v pohyblive radove carce.

HI-BYTE

Aby se mohly znazornit vetsi ciselne hodnoty, napr.adresy, rozlozi se hodnota na horni a dolni dil. HI-byte udava nasobky 256, LO-byte zbytek cisla: HI*256+LO. Timto zpusobem lze znazornit hodnoty 0- 65535. Oba dily se ukladaji vzdy v poradi LO,HI.

I/O

Input/Output.Vstup/vystup.

IOCB

Input/Output Control Block, vstupni/vystupni kontrolni blok pouzivany CIO k dotazum na periferie.

LMS

Load Memory Scan, dvoubytovy ukazatel na cast pameti, ktera obsahuje data ke zpracovani.

LO-BYTE

viz HI-byte.

MODE LINE (ZPUSOBOVY RADEK)

Pracovni jednotka pro graficke zpracovani na obrazovce. Podle grafickeho modu muze obsadit 1,2,4,8,10 nebo 16 TV radku. S kolika scan lines se ma pracovat jako s jednim zpusobovym radkem, urci prislusny ANTIC prikaz.

NIBBLE

Polovina bytu, horni nebo dolni. Pri programovani ve strojovem kodu se sdruzi 8 bitu k hexadec.zapisu. Hex. cisla maji zaklad 16. Skladaji se ze 16 cislic (0-9, A-F). Kazde misto v hex. cisle predstavuje nasobek sestnacti. Ke znazorneni hex. cislice se pouzivaji 4 bity, tedy polovina bytu, neboli nibble. Byte saha binarne od 0000 0000 do 1111 1111, decimalne od 0 do 255 a hexadecimalne od 00 do FF.

OS

Operating System. Operacni system, ktery ridi funkci pocitace.

PAGE (stranka)

Jednotka pameti, ktera obsahuje 256 bytu. Jeji zacatek oznacuji HI byty vsech pametovych adres.

PIXEL

Picture Element, obrazovy prvek, z ktereho se sestavuje graficky obsah TV obrazu vytvoreneho pocitecem. Meritkove hodnoty obrazovky jsou radky (scan lines) a obrazove body (color clocks, jeden color clock obsahuje 2 obrazove body). Pixel muze byt siroky i vysoky podle ruzneho mnozstvi obrazovych bodu, coz zavisi na zvolenem grafickem modu.

PM-GRAPHICS

Player-Missile-Graphics, hrac a strela, zvlastni varianta sprites (skritci, duchove) , grafickych objektu, ktere pri pohybu nezavisi na normalnim obsahu obrazovky (GRAPHICS).

POWERUP

Zapnuti (powerup) ovlivnuje prubeh ruznych rutin, ktere pripravi pocitac k cinnosti. Napr. do registru se ulozi default hodnoty a prezkouseji se ruzne stavy, napr. je-li pripojena aktivni disketova jednotka.

RAM

Random Access Memory; volne pristupna pamet, kterou muze uzivatel popsat svym BASIC programem. RAM pamet uchova informace pouze do vypnuti pocitace, kdy se vymaze.

ROM

Read Only Memory; pouze ctena pamet, obsahuje potrebne programove vybaveni- operacni system a interpret BASIC. Zustava zachovana i po vypnuti.

STINOVY REGISTR

K umozneni vlivu BASICu na urcite hardwarove registry. Jsou v nich ulozeny potrebne hodnoty. Popsani samotneho hardwaroveho registru nema cenu, protoze muze byt prepsan kazdou 1/50 sec. BASIC je prilis pomaly, aby zde mohl zasahnout. Hodnoty ze stinovych registru se ctou v rychlem "tepu srdce" pocitace. Tak lze touto oklikou ovlivnit trvale zmeny.

SIO

Serial Input/Output, rutiny pro seriovy vstup a vystup dat.

VBLANK

Vertical Blank, casova prodleva v momentu, kdy katodovy paprsek dosahl pravy dolni roh obrazovky a skace zpet k levemu hornimu rohu. V tomto kratkem okamziku se paprsek vypina. VBLANK probiha v PAL norme kazdou 1/50 sec.

VEKTOR

Registr, ktery obsahuje urcitou adresu, napr. startadresu pametoveho rozsahu nebo programove rutiny. Kdyz ma vektor ukazat na libovolnou adresu, pouziva 2 byty. Jednobytovy vektor obsahuje pouze HI byte, ukazuje tedy jen na zacatek stranky.

HODNOTA

Cislice dostane podle postaveni uvnitr cisla urcitou hodnotu. V decimalnim systemu znamena kazde misto mocninu deseti, tedy 1, 10, 100, atd. Stejne tak dostane bit uvnitr bytu podle sveho umisteni urcitou hodnotu, ktera je mocninou dvou. Bit 7 ma hodnotu 2 na 7 = dec. 128.

UKAZATEL

viz vektor.

Pouzita literatura:

1/ Koch.:PEEK a POKES ZU ATARI 600XL/800XL, DATA BECKER Gmbh. 1985
2/ RESCHKE,J.,WIETHOFF,A.: DAS ATARI PROFIBUCH. 1985

OBSAH:

Igi/2019