Pavel DOČEKAL

ABC o počítačích ATARI®

600XL/800XL

knihovnička klubu mikroelektroniky ssm

1. PRINCIP ZOBRAZOVÁNÍ NA TELEVIZNÍM PŘIJÍMAČI

Pochopení principu televizního zobrazování je nezbytným předpokladem při výkladu dalších kapitol.

Zobrazování na televizní obrazovce pracuje rastrovacím způsobem. Elektronový paprsek v obrazovce vhodně vychylovaný dopadá na stínítko a díky speciálním luminoforům nanesených zevnitř stínítka emituje světlo. Paprsek postupně rastruje z levého horního rohu a na celé obrazovce tak postupně vytvoří 312,5 řádků (televizory pracují v tzv. prokládaném řádkování, aby se co nejvíce potlačilo nežádoucí blikání obrazu, takže řádků je pak ve skutečnosti 625, ale pro jednoduchost výkladu budu prokládané řádkování ignorovat. Jakmile paprsek dorazí k pravému okraji stínítka, jeho intenzita na okamžik poklesne a vychylovací soustava přesune paprsek k levému okraji stínítka o řádek níže a zobrazování pokračuje. Okamžik přesunu paprsku se nazývá horizontální zatemnění. Po zobrazení všech 312 řádků je potřeba paprsek podobným způsobem přenést opět do levého horního rohu stínítka, a doba potřebná k přesunu se nazývá vertikální zatemnění. Vykreslení všech řádků trvá 20 ms a vertikální zatemnění z toho trvá asi 1,4 ms. Vykreslení jednoho řádku trvá asi 64 μs, z toho horizontální zatemnění zabere asi 14 μs.

Z různých důvodů využívá ATARI pouze 192 obrazových řádků, a tím i maximálního možného vertikálního rozlišení 192 bodů.

Jednotkou horizontálního rozlišení je tzv. barvový bod. Šířka obrazu je volitelná na základě definování šířky barvového bodu. K dispozici je celkem 228 barvových bodů, lze však využít nejvýše 176 barvových bodů. Běžná šířka je 160 barvových bodů. Každý barvový bod se dá rozdělit na dvě poloviny. Běžně je tedy horizontálně využitelných 320 bodů.

2. POPIS ZOBRAZOVÁNÍ

2.1 OBRAZ

Základním problémem každého mikropočítače je připojení běžného televizoru, neboť televizní zobrazování je dynamický proces. Z tohoto důvodu většina systémů používá speciální obvody k výrobě obrazu systémem:

mikroprocesor ——→ obrazová paměť
obrazové obvody ——→ TV obrazovka

Mikroprocesor zapisuje informace do obrazové paměti, obrazové obvody průběžně informace z obrazové paměti vybírají a kódují je do televizního signálu. Na obrazových obvodech pak do značné míry závisí kvalita obrazu.

2.2 DISPLAY-LIST

Display - list je možno volně přeložit na obrazový (pod)program a patří bezesporu mezi nejdůležitější prostředky počítače ATARI. Na rozdíl od počítačů PET nebo TRS - 8Ø, používající jeden grafický režim, či APPLE se třemi grafickými režimy, používá ATARI 14 grafických režimů. Kromě toho lze obrazovou paměť umístit kdekoliv do prostoru paměti počítače.

To vše umožňuje speciální obrazový mikroprocesor ANTIC. Konstruktéři počítače předali celý složitý systém obrazových obvodů jediné součástce - mikroprocesoru. A ANTIC opravdu mikroprocesorem je, neboť má svůj instrukční soubor, program a data. Program pro ANTIC má už známý název display - list (obrazový program). V dalším textu pouze zkráceně DL.

V DL nalezneme adresu obrazové paměti, obrazový režim a další spec. údaje.

Chceme-li úspěšně využívat kvalit DL, je důležité se zbavit starého pohledu na obrazovou paměť a vidět v ní horizontálně navršené televizní řádky přes celou šířku obrazovky. GRAPHICS 2 je např. vysoký 16 obrazových řádků, zatímco GRAPHICS 7 pouze 2 řádky. Přestože je obrazová paměť lehce dosažitelná v mnoha gr. režimech přímo BASICem, nemělo by to omezovat naši obrazotvornost a fantazii při tvorbě programů, neboť DL umí mnohem víc ...

2.3 INSTRUKČNÍ SOUBOR ANTICu

V podstatě se jedná o čtyři druhy instrukcí:

1. Map - mode	pracuje s barvou obrazových bodů
2. Blank - line	dokáže v určeném místě obrazovky vytvořit prázdný řádek s barvou rámečku
3. Charakter - mode	pracuje s obrazovými znaky
4. Jump	po vykonání této instrukce přejde program skokově na novou adresu.

Kromě těchto instrukcí jsou k dispozici ještě další čtyři:

1. DLI - display list interrupt - přerušení systému DL
2. LMS - load-memory scan - určení obrazové paměti
3. vertical scroll - svislé rolování
4. horizontal scroll - vodorovné rolování

Instrukce Map - mode vytváří barevné obrazové body. Barva je dána barvovým registrem a výběr barvového registru specifikují obrazová data. V gr. režimech využívající čtyři barvy (3, 5, 7) určuje dvojice bitů barvový registr.

Tab. I

HEX.	DEC.	Použitý barvový registr
ØØ	Ø	COLBAK
Ø1	1	COLPFØ
1Ø	2	COLPF1
11	3	COLPF2

Protože je k určení barvy jednoho obrazového bodu potřeba pouze dvou bitů, určí čtyři obrazové body jeden byte obrazových dat.

Příkl. $1B_H$ = ØØ Ø1 1Ø 11_B

První dvojice bitů určí reg. pozadí, druhá barvový reg. Ø. atd. V grafických režimech pracují se dvěmi barvami (4, 6, 8) - určuje každý bit obrazového bytu jeden ze dvou barvových registrů. Bit s hodnotou Ø určí registr pozadí a bit s hodnotou 1 určí barvový registr. Osm obrazových bodů je takto určeno jedním bytem obrazových dat.

Tab. II

ANTIC	BASIC	POČET BAREV	POČET TEL. Ř. NA ŘÁDEK	POČET ŘÁDKŮ NA OBRAZOVKU	POČET BYTE NA ŘÁDEK	POČET BYTE NA OBRAZOVKU
2	Ø	2	8	4Ø	4Ø	96Ø
3	/	2	1Ø	4Ø	4Ø	76Ø
4	12	4	8	4Ø	4Ø	96Ø
5	13	4	16	4Ø	4Ø	48Ø
6	1	5	8	2Ø	2Ø	48Ø
7	2	5	16	2Ø	2Ø	24Ø
8	3	4	8	4Ø	1Ø	24Ø
9	4	2	4	8Ø	1Ø	48Ø
A	5	4	4	8Ø	2Ø	96Ø
B	6	2	2	16Ø	2Ø	1 92Ø
C	14	2	1	16Ø	2Ø	3 84Ø
D	7	4	2	16Ø	4Ø	3 84Ø
E	15	4	1	16Ø	4Ø	7 68Ø
F	8,9,10, 11	2	1	32Ø	4Ø	7 68Ø

Všimněte si, že čísla ANTIC se nekryjí s BASIC.

Instrukce blank - line vytvoří na obrazovce "mrtvý" řádek, jehož barva bude shodná s barvou rámečku. Instrukcí této skupiny je osm.

Skoková instrukce JUMP provede skok programu ANTIC na novou adresu, určenou operandem za instrukcí. Skočit lze však nejvýše o 1 KB, neboť programový čítač ANTIC je pouze desetibitový.

Druhá skoková instrukce JVB (Jump vertical blank) se používá častěji. Instrukce nejprve změní hodnotu programového čítače, vyčká na vykonání snímkového zatemňovacího impulsu a pak skočí na novou adresu. Běžné je použití na konci DL, kde po vykonání snímkového zatemňovacího impulsu skočí opět na počátek DL.

Jump i JVB jsou 3 bytové instrukce, první byte je operační kód a druhý a třetí byte operand udávající nižší a vyšší část adresy skoku.

Operand instrukce LMS je ze dvou částí

- čtyři vyšší bity mají hodnotu $Ø1ØØ_B$
- čtyři nižší bity mají hodnotu grafického režimu ANTIC tabulky II.

<u>Příklad:</u> požadujeme GRAPHICS 6
ANTIC = B
LMS = 4B hexadecimálně.

Za LMS následuje 2 bytový operand udávající adresu počátku obrazové paměti. Obvykle je LMS použita jen jednou na počátku programu DL. Je - li však obrazová paměť větší než 4 KB (Čítač memory - scan je pouze dvanáctibitový) je potřeba LMS použít podruhé

a znovu definovat zbytek obrazové paměti. Tento případ nastává u GR. 24.

2.4 STRUKTURA OBRAZOVÉHO PROGRAMU DL

Všechny standartní DL začínají třemi instrukcemi blank - line. Následuje LMS a 2 bytový operand adresy paměti. Další byte jsou kódy požadovaného grafického režimu v ANTIC (viz. tab. II). Např. GR.Ø bude 23 bytů s hodnotou 2. DL musí končit instrukcí JVB. DL pro režim Ø (BASIC):

Tab. III

Hex. adr.	Byte	Komentář
7BEØ	7Ø	tři "mrtvé" řádky shora obrazu
7BE1	7Ø	
7BE2	7Ø	
7BE	42	LMS pro GR.Ø
7BE	2Ø	adresa obrazové paměti
7BE	7C	
.	Ø2	23x ANTIC kód Ø2
.	.	
.	.	
.	.	
.	Ø2	
7BFD	41	JVB
7BFE	BØ	adresa počátku DL
7BFF	7B	

Celý program je dlouhý 32 byte a nejdelší DL přesahují málo přes 2ØØ byte. Nyní DL zkuste sami modifikovat, popř. sestavit nový program DL, přičemž by program neměl přesáhnout délku 1KB. Pak bude nutné použít JMP a přepsat adresu v ukazovátku počátku programu DL (560 + 561 * 256).

2.5 UMÍSTĚNÍ OBRAZOVÝCH DAT

Obrazová paměť může být umístěna na kterémkoliv místě adresového prostoru počítače. DL definuje počátek obrazových dat pomocí instrukce LMS,

zobrazující každý řádek jako 4Ø bytový, tedy i ten, který je jinak 2Ø bytový.

Popsanou nepříjemnost lze obejít nepoužíváním příkazů PRINT a PLOT, nebo zobrazováním různých řadků, ale se stejným počtem bytů na řádek.

2.6 APLIKACE DL

Použitím instrukcí blank - line lze vyrobit na obrazovce "okno", do kterého pak lze umístit důležité sdělení či nápis.

Velké možnosti nabízí instrukce LMS. Např. změnou adresy za instrukcí LMS během zatemňovacího impulsu se dá okamžitě změnit obraz, přičemž je možné bez blikání obrazovky proložit až čtyři různá zobrazení. (Pozor na barvy!)

Pomocí změny DL lze rovněž vytvořit program pro editaci textů s možností snadného posuvu obrazových dat nahoru a dolů.

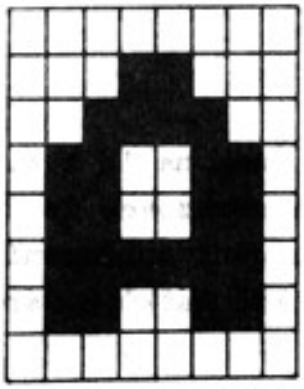
3. G R A F I K A

3.1 STAVBA OBRAZOVÝCH ZNAKŮ

ATARI používá dvou standartních souborů znaků uložených od adresy 256 * 224 (abeceda + grafické znaky) a od adresy 256 * 2Ø4 (abeceda + španělská, německá a další speciální písmena). Nic však nebrání vytvoření vlastního souboru znaků a jeho používání.

Každý znak je na obrazovce vytvořen soustavou 8 x 8 bodů a v paměti ROM je pak jeden znak určen osmicí bytů. Vypadá to takto:

Tab. IV

znak	binární vyjádření	hex. vyjádření
	ØØØØØØØØ	ØØ
	ØØØ11ØØØ	18
	ØØ1111ØØ	3C
	Ø11ØØ11Ø	66
	Ø11ØØ11Ø	66
	Ø111111Ø	7E
	Ø11ØØ11Ø	66
	ØØØØØØØØ	ØØ

Kompletní soubor znaků definuje 128 znaků a každý znak může být znázorněn inverzně. Soubor zaujímá 128 x 8 = 1Ø24 byte paměti a musí začínat adresou, která je celým násobkem 1KB.

Soubor znaků pro mod 1 a 2 používá pouze 64 vybraných znaků, v paměti zabírá 512 byte a počátek souboru musí být celým násobkem Ø,5 KB.

U standartního souboru znaků definuje prvních osm byte prázdný znak (space), dalších osm byte vykřičník atd. (viz tab. v základní příručce ATARI na straně 89 originálu).

3.2 TVORBA VLASTNÍCH ZNAKŮ

K zobrazení vlastních znaků je potřeba zapsat novou adresu počátku souboru znaků, dělenou 256, do ukazovátka souboru znaků. Ukazovátko (CHBAS) má adresu 756_D. Poté lze obdržet již nové znaky, např. azbuku či českou abecedu.

3.3 DODATEK

Mod 3 ANTIC zvětší obrazový znak na 10 x 8. Tím se sice zmenší počet řádků na obrazovce ze 24 (mod Ø) na 19, současně se však umožní zobrazení vysokých znaků (nebo v české abecedě háčků a čárek).

4. PLAYER-MISSILE GRAPHICS

4.1 ÚVOD

Prokládání obrazů na obrazovce je důležitou schopností osobních počítačů. V podstatě existují dva problémy:

- Jestliže se pracuje s grafikou s malým rozlišením, není pohyb obrázku plynulý. Je tedy nutné pracovat s grafikou jemnější.
- Řešení druhého problému je mnohem horší. Zatímco organizace paměti je jednorozměrná, k prokládání obrazů je potřeba jinak organizované paměti nebo zcela jiný způsob zobrazování. ATARI používá systému Player-missile graphics. Doslovný překlad je Hráč - - střela - grafika. V dalším textu pouze zkratka PMG. Problém organizace paměti vyřešil ATARI u systému PMG tak, že jeden obraz je v běžné obrazové paměti, jejíž počátek je dán DListem a jiné obrazy se pohybují ve zcela jiné části paměti, přičemž je počátek této paměťové plochy definován na počátku programu. Přestože jsou obrazy umístěny v různých místech paměti, PMG umožní jejich společné zobrazení. PMG umožňuje zobrazování a pohyb dvou typů obrázků: hráčů a střel (samozřejmě kromě dalších obrázků v běžné obrazové paměti).

4.2 HRÁČI

Jejich počet je maximálně čtyři. Soubor čísel představujících obrázek se uloží do oblasti paměti, vyhrazené pro PMG. Každý hráč má k dispozici určitou vymezenou plochu paměti.

Vertikální posuv hráče o velikosti 8 byte lze provést v BASIC např. takto:

```
FOR I = 7 TO Ø STEP - 1
POKE YPOS + I + 1, PEEK (YPOS + I)
NEXT I : POKE YPOS, Ø
YPOS = YPOS + 1                    ( ~ 2ØØ ms)
```

Ve strojovém kódu bude posuv hráčů podstatně rychlejší

```
        LDX #  Ø7
LOOP    LDA YPOS, X
        STA YPOS + 1, X
        DCX
        BNE LOOP
        LDA #  ØØ
        STA YPOS
        .
        .
        .                          ( ~ 85 µs)
```

Časový rozdíl je opravdu obrovský a přitom program ve strojovém kódu není dlouhý.

Pohyb hráče v horizontálním směru je podstatně jednodušší. Využívá se tzv. registru horizontální polohy. Ihned po provedení POKE RHP, X se obrázek přesune na novou pozici, určenou X. Horizontální a vertikální pohyb není na sobě přitom závislý. Horizontální poloha může nabýt až 256 různých hodnot s určitým omezením. Pozice Ø ÷ 44 je mimo levý okraj obrazovky a pozice 220 ÷ 255 je mimo pravý okraj obrazovky. V praxi se využívá ještě užší rozsah, asi 6Ø ÷ 2ØØ. Televizory jsou různé. Každý z hráčů může mít jinou barvu danou registry COLP(X), hráč je jednobarevný. SIZEP(X) umožňuje různou šířku hráče (jednotlivě):normální, dvojnásobnou, čtyřnásobnou.

S výškou obrázku, ať už hráčů nebo střel, je to horší. Lze volit body o výšce jednoho nebo dvou televizních řádků. Výška bodů je pak jednotná pro všechny hráče i střely. Volba je provedena bitem D4 registru DMACTL.

Podobně jako u horizontálního posuvu, rovněž u vertikálního posuvu je prvních 1Ø pozic mimo horní okraj obrazovky a posledních 2Ø pozic mimo dolní okraj obrazovky (platí pro rozlišení 2 televizních řádků na bod). Pro rozlišení jednoho televizního řádku na bod je to 20 pozic shora a 40 pozic zdola. V prvním případě může nabýt vertikální směr hodnot 1Ø ÷ 1Ø7 a ve druhém případě 2Ø ÷ 215.

4.3 STŘELY

Obrázky střel jsou široké pouze 2 bity (na rozdíl od šířky hráčů 8 bitů) a všechny čtyři střely se pohybují ve společném paměťovém prostoru (to je další rozdíl od hráčů).

Každá střela přináleží jednomu hráči a jejich barvy jsou shodné. Každá střela má od ostatních střel nezávislý horizontální pohyb díky registru horizontální pozice HPDSM(X). Šířka střel je nezávislá na hráčích a nastavitelná v registru SIZEM.

Nezávislý pohyb střel ve vertikálním směru je problémový díky dvoubitové šířce střely a společnému prostoru v tabulce PMG.

PMG umožňuje nejen nezávislý pohyb hráčů a střel na jiných obrázcích, ale dává i jiné možnosti, např. prioritní zobrazení libovolných hráčů nebo střel nad jinými hráči, střelami a ostatními obrázky. Díky této schopnosti PMG lze vytvořit plastické obrazy. Priorita obrázků se volí registrem PRIOR.

Střely je možno použít dohromady, jako jeden společný útvar, jejich barvu pak lze volit společně nastavením bitu D4 registru PRIOR.

4.4 KOLIZNÍ REGISTRY

Umožňují vyhodnocení setkání dvou obrázků na obrazovce. K dispozici jsou registry k vyhodnocení

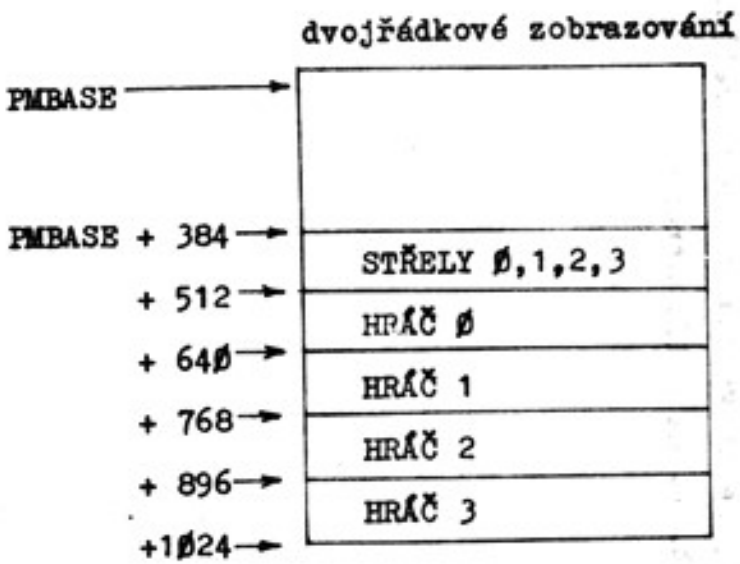
střetu: střela - grafické pozadí
střela - hráč
hráč - hráč
hráč - grafické pozadí,

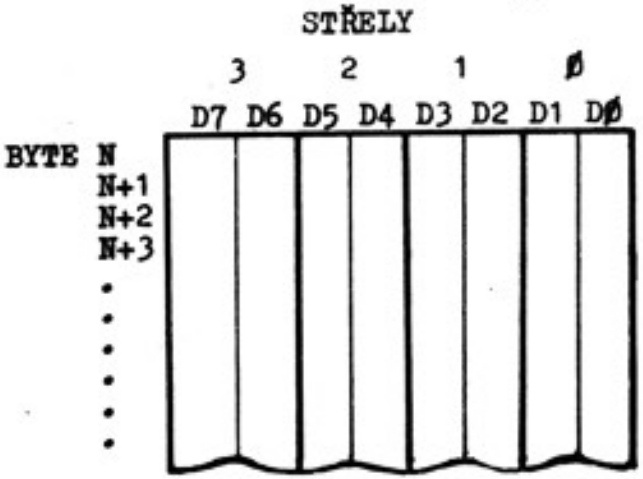
celkem 16 registrů. Při střetu se v registru nastaví 1. Nulování kolizních registrů nelze provést přímo, nýbrž skrze registr HITCLR. Vynulováním HITCLR dojde k vynulování všech kolizních registrů.

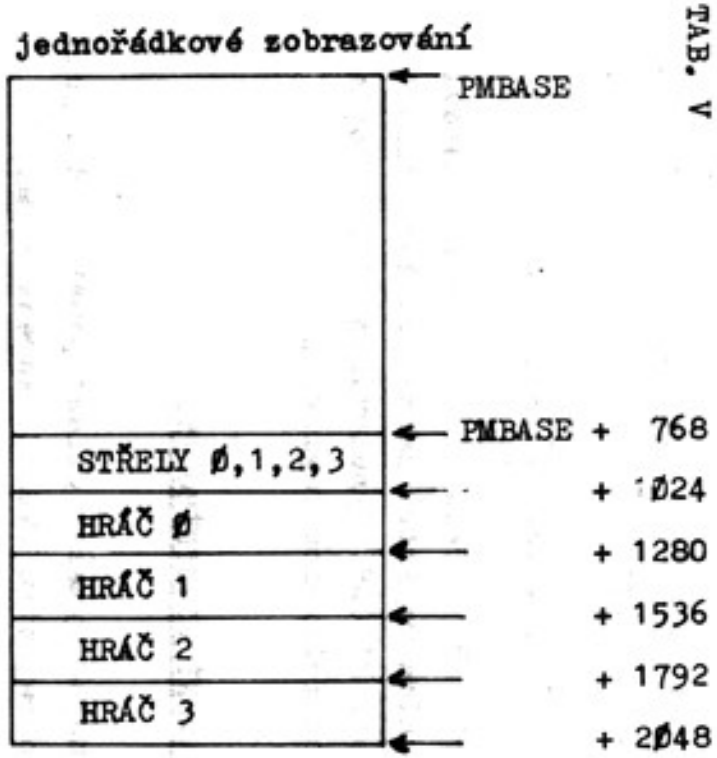
4.5 ZPŮSOB SESTAVENÍ VLASTNÍHO PMG

Nejprve je nutno vymezit oblast paměti RAM a její počátek zapsat do registru PMBASE. Adresa v PMBASE musí být celým násobkem 1KB nebo 2KB (pro jemnější PMG). Ve skutečnosti potřebuje PMG pouze 640 byte, nebo 1280 byte (pro jemnější PMG), takže zbytek paměti, která je vlastně chráněná, se dá využít k jiným účelům. Rozčlenění plochy je v tab. V. na další straně.

TAB. V







u dvojřádkového zobrazování je N ≤ 128
u jednořádkového zobrazování je N ≤ 256

Nyní se oblast "vyčistí", vynuluje a do vyčištěné oblasti se vsadí vlastní hráči nebo střely. Následuje nastavování registrů barev, horizontální pozice, šířky obrázku a je-li to nutné, registru prioritního zobrazení. Informace pro procesor ANTIC o volbě vertikálním zobrazování (jedno či dvojřádkovém) se zapíše do registru DMACTL bitu D4. Do DMACTL se rovněž zapíší potřebné údaje o PMG. (Opatrně s manipulací tohoto registru).

Jednoduchý program v BASIC umožňuje pomocí páky pohybovat s malou kresbou letadla.

PROGRAM Č. 1

```
  1  PMBASE=54279:REM        adresa ukazovátka PMG
  2  RAMTOP=1Ø6:REM          adresa ukazovátka RAMTOPu
  3  SDMCTL=559:REM          adresa sledujícího registru
                                  DMACTL v RAM
  4  GRACTL=53277:REM        grafický kontrolní registr
                                  obvodu CTIA
  5  HPOSPØ=53248:REM        adresa registru horizontální
                                  pozice PØ
  6  PCOLRØ=704:REM          adresa sledujícího barvového
                                  registru po V RAM
 1Ø  GRAPHICS Ø:SETCOLOR 2,Ø,Ø:REM  nastavení barvy pozadí
 2Ø  X=1ØØ:REM               počáteční horizontální pozice
                                  hráče
 3Ø  Y=48:REM                počáteční vertikální pozice
                                  hráče
 4Ø  A=PEEK(RAMTOP)-8:REM    změna RAMTOPu o 2KB
 5Ø  POKE PMBASE,A:REM       do PMBASE adresa počátku PMG
                                  plochy
 6Ø  MYPMBASE=256*A:REM      výpočet skutečné adresy PMG
                                  plochy
 7Ø  POKE SDMCTL,46:REM      nastavení dvouřádkového zobra-
                                  zování
 8Ø  POKE GRACTL,3:REM       nastavení zobrazování PMG
 9Ø  POKE HPOSPØ:REM         nastavení registru horizontální
                                  pozice PØ
1ØØ  FOR I=MYPMBASE + 512 TO
     MYPMBASE + 64Ø:REM      nulování plochy hráče
```

```
110 POKE I,0
120 NEXT I
130 FOR I=MYPMBASE + 512 + Y TO
    MYPMBASE + 518 + Y
140 READ A:REM                    sestavení hráče (letadýlka)
150 POKE I,A
160 NEXT I
170 DATA 8, 17, 35, 255, 32, 16, 8
180 POKE PCOLR0,88:REM            nastavení barvy hráče
190 A=STICK(0):REM                čtení polohy páky
200 IF A= 15 THEN GOTO 190:REM páka v klidové poloze
210 IF A= 11 THEN X=X-I:POKE HPOSP0,X
220 IF A=7 THEN X=X+I:POKE HPOSP0,X
230 IF A< >13 THEN GOTO 280
240 FOR I=8 TO 0 STEP - 1
250 POKE MYPMBASE+512+Y+I,PEEK
    (MYPMBASE+511+Y+I)
260 NEXT I
270 Y=Y+I
280 IF A< >14 THEN GOTO 190
290 FOR I=0 TO 8
300 POKE MYPMBASE+511+Y+I,PEEK
    (MYPMBASE+512+Y+I)
310 NEXT I
320 Y=Y-I
330 GOTO 190
```

Pokud se v probíhajícím programu přestane PMG používat, je vhodné zrušit zobrazování PMG, ušetří se asi 10 % času při dalším vykonávání BASIC programu.

Další program vhodně dokumentuje možnosti prioritního zobrazování.

PROGRAM č. 2

```
  1 RAMTOP=106:REM          adresa ukazovátka RAMTOPu
  2 PMBASE=54279:REM        adresa ukazovátka PMG
  3 SDMCTL=559:REM          adresa sledujícího registru
                            DMACTL v RAM
  4 GRACTL=53277:REM        grafický kontrolní registr
                            obvodu GTIA
  5 HPOSPØ=53248:REM        adresa registru horizontál-
                            ní pozice hráče PØ
  6 PCOLRØ=704:REM          adresa sledujícího barvové-
                            ho registru hráče PØ v RAM
  7 SIZEPØ=53256:REM        adresa registru šířky hráče
  8 GPRIOR=623:REM          adresa registru prioritní-
                            ho zobrazení
 1Ø GRAPHICS 7
 2Ø SETCOLOR 4,8,4
 3Ø SETCOLOR 2,Ø,Ø
 4Ø COLOR 3
 5Ø FOR Y=Ø TO 79:REM       vyplnění plochy obrazovky
 6Ø PLOT Ø,Y
 7Ø DRAWTO 159,Y
 8Ø NEXT Y
 9Ø A=PEEK(RAMTOP)-2Ø:REM   přesun RAMPOPu
1ØØ POKE PMBASE,A
11Ø MYPMBASE=256*A
12Ø POKE SDMCTL,46
13Ø POKE GRACTL,3
14Ø POKE HPOSPØ,1ØØ
```

```
15Ø FOR I=MYPMBASE+512 TO MYPMBASE
    +64Ø
16Ø POKE I,255:REM                    vytvoření hráče č.Ø
                                      (část plochy)
17Ø NEXT I
18Ø POKE PCOLRØ,88
19Ø POKE SIZEPØ,3:REM                 nastavení šířky hráče
                                      na čtyřnásobek
2ØØ POKE GPRIOR,4:REM                 nastavení registru prio
                                      rity
21Ø COLOR 4
22Ø FOR Y=3Ø TO 4Ø:REM                vytvoření hráče č.1
                                      (část plochy)
23Ø PLOT Y+22,Y
24Ø DRAWTO Y+43,Y
25Ø NEXT Y
```

Tento program vytvoří na obrazovce znak integrálu

PROGRAM č. 3

```
  1 RAMTOP=1Ø6:REM                    adresa ukazovátka RAMTOPu
  2 PMBASE=54279:REM                  adresa ukazovátka PMG
  3 SDMCTL=559:REM                    adresa sledujícího regist-
                                      ru DMACTL v RAM
  4 GRACTL=53277:REM                  grafický kontrolní registr
                                      obvodu GTIA
  5 HPOSPØ=53248:REM                  adresa registru horizon-
                                      tální pozice hráče PØ
  6 PCOLRØ=7Ø4:REM
 1Ø GRAPHICS Ø:A=PEEK(RAMTOP)-
    16:REM                            přesun RAMTOPu
 2Ø POKE PMBASE,A
```

```
30 MYPMBASE=256*A
40 POKE SDMCTL,62
50 POKE GRACTL,3
60 POKE HPOSP0,102
70 FOR I=MYPMBASE+1024 TO
   MYPMBASE+1280
80 POKE I,0
90 NEXT I
100 POKE PCOLR0,140
110 FOR I=0 TO 15
120 READ X
130 POKE MYPMBASE+1100+I,X
140 NEXT I
150 DATA 14,29,24,24,24,24,24,24
160 DATA 24,24,24,24,24,24,184,112
170 ?" ↰ ":REM                          vyčištění obrazovky
180 POSITION  5,6
190 ?"x cx"
```

TAB VI.

Hardware registry				Sledující registry		
Symbolický název	Funkce	Adresa Hexadec.	Dekad.	Symbolický název	Adresa Hexadec.	Dekad.
COLBK	barva pozadí	D01A	53274	COLOR4	2C8	712
COLPF0	barva obrazu 0	D016	53270	COLOR0	2C4	708
COLPF1	barva obrazu 1	D017	53271	COLOR1	2C5	709
COLPF2	barva obrazu 2	D018	53272	COLOR2	2C6	710
COLPF3	barva obrazu 3	D019	53273	COLOR3	2C7	711
COLPM0	barva P-M 0	D012	53266	PCOLR0	2C0	704
COLPM1	barva P-M 1	D013	53267	PCOLR1	2C1	705
COLPM2	barva P-M 2	D014	53268	PCOLR2	2C2	706
COLPM3	barva P-M 3	D015	53269	PCOLR3	2C3	707
DMACTL	řízení přímého přístupu k paměti	D400	54272	SDMCTL	22F	559
GRACTL	řízení graficky CTIA	D01D	53277			
GRAFM	grafický registr střel	D011	53265			
GRAFP0	grafický registr pro hráče0	D00D	53261			
GRAFP1	grafický registr pro hráče1	D00E	53262			
GRAFP2	grafický registr pro hráče2	D00F	53263			
GRAFP3	grafický registr pro hráče3	D010	53264			
HITCLR	nulování kolizních registrů	D01E	53278			
HPOSM0	horizont. pozice střely 0	D004	53252			
HPOSM1	horizont. pozice střely 1	D005	53253			
HPOSM2	horizont. pozice střely 2	D006	53254			
HPOSM3	horizont. pozice střely 3	D007	53255			
HPOSP0	horizont. pozice hráče 0	D000	53248			
HPOSP1	horizont. pozice hráče 1	D001	53249			
HPOSP2	horizont. pozice hráče 2	D002	53250			
HPOSP3	horizont. pozice hráče 3	D003	53251			
M0PF	kol. reg. střela 0 - obraz	D000	53248			
M0PL	kol. reg. střela 0 - hráč	D008	53256			
M1PF	kol. reg. střela1- obraz	D001	53249			
M1PL	kol. reg. střela1- hráč	D009	53257			
M2PF	kol. reg. střela 2 - obraz	D002	53250			
M2PL	kol. reg. střela 2 - hráč	D00A	53258			
M3PF	kol. reg. střela 3 - obraz	D003	53251			
M3PL	kol. reg. střela 3 - hráč	D00B	53259			
P0PF	kol. reg. hráč 0 - obraz	D004	53252			
P0PL	kol. reg. hráč 0 - hráč	D00C	53260			
P1PF	kol. reg. hráč1- obraz	D005	53253			
P1PL	kol. reg. hráč1- hráč	D00D	53261			
P2PF	kol. reg. hráč 2-obraz	D006	53254			
P2PL	kol. reg. hráč 2 - hráč	D00E	53262			
P3PF	kol. reg. hráč 3 - obraz	D007	53255			
P3PL	kol. reg. hráč 3 - hráč	D00F	53263			
PMBASE	ukazovátko počátku PMGplochy	D407	54279			
PRIOR	výběr prioritního zobraz.	D01B	53275	GPRIOR	26F	623
SIZEM	šířka střel	D00C	53260			
SIZEP 0	šířka hráče 0	D008	53256			
SIZEP 1	šířka hráče 1	D009	53257			
SIZEP 2	šířka hráče 2	D00A	53258			
SIZEP 3	šířka hráče 3	D00B	53259			

4.6 PODROBNĚJŠÍ INFORMACE O REGISTRECH POUŽÍVANÝCH PMG SYSTÉMEM

COLBK (barva pozadí)

BARVA				JAS			
D7	D6	D5	D4	D3	D2	D1	nepoužit
x	x	x	x	Ø	Ø	Ø	nulový jas
(viz níže)				Ø	Ø	Ø	
				atd.			
				1	1	1	maximální jas

Ø	Ø	Ø	Ø	šedá
Ø	Ø	Ø	1	zlatá
Ø	Ø	1	Ø	oranžová
Ø	Ø	1	1	červeno - oranžová
Ø	1	Ø	Ø	růžová
Ø	1	Ø	1	purpurová
Ø	1	1	Ø	purpurově - modrá
Ø	1	1	1	modrá
1	Ø	Ø	Ø	modrá
1	Ø	Ø	1	světle modrá
1	Ø	1	Ø	tyrkysová
1	Ø	1	1	zeleno - modrá
1	1	Ø	Ø	zelená
1	1	Ø	1	žluto - zelená
1	1	1	Ø	oranžově - zelená
1	1	1	1	světle oranžová

Sledující registr: COLOR 4

COLPFØ - COLPF3 (barva obrazu)

D7	D6	D5	D4	D3	D2	D1	DØ

(význam bitů stejný jako u COLBK)

Sledující registr: COLORØ÷3

COLPMØ - COLPM3 (barva P - M)

D7	D6	D5	D4	D3	D2	D1	DØ

(význam bitů stejný jako u COLBK)

Sledující registr: PCOLRØ÷3

DMACTL (řízení přímého přístupu k paměti)

nepoužit	D5	D4	D3	D2	D1	DØ

D5 = 1 umožnění ANTIC operací
D4 = 1 jednořádkové zobrazování PMG
D4 = Ø dvojřádkové zobrazování PMG
D3 = 1 povolení DMA pro hráče
D2 = 1 povolení DMA pro střely
D1,DØ = Ø|Ø povolení DMA pro obraz
= Ø|1 úzký obraz
= 1|Ø normální obraz ?!
= 1|1 široký obraz

viz GRACTL. Sledující registr:SDMCTL

GRACTL (řízení graficky)

nepoužit	D1	DØ

D1 = povolení DMA pro hráče skrze GRAFP(X)
DØ = povolení DMA pro střely skrze GRAFM

DMA je povoleno nastavením bitů v DMACTL a GRACTL. Pouhým nastavením DMACTL se spotřebovává strojový čas, ale zobrazování není.

GRAFM (grafický registr střel)

D7	D6	D5	D4	D3	D2	D1	DØ
L	R	L	R	L	R	L	R
M3		M2		M1		MØ	číslo střely

GRAFPØ ÷ GRAFP3 (grafický registr hráčů)

D7	D6	D5	D4	D3	D2	D1	DØ
vlevo						vpravo	
hráč na obrazovce							

HITCLR (nulování kolizních registrů)

nepoužito

HPOSMØ ÷ HPOSM3 (horizontální pozice střely)

D7	D6	D5	D4	D3	D2	D1	DØ

HPOSPØ ÷ HPOSP3 (horizontální pozice hráče)

D7	D6	D5	D4	D3	D2	D1	DØ

MØPF,M1PF,M2PF,M3PF (kolizní registr střela - obraz)

nepoužito, vždy Ø	D3	D2	D1	DØ	
	3	2	1	Ø	číslo obrazu

MØPL,M1PL,M2PL,M3PL (kolizní registr střela - hráč)

nepoužito, vždy Ø	D3	D2	D1	DØ	
	3	2	1	Ø	číslo hráče

P0PF,P1PF,P2PF,P3PF (kolizní registr hráč - obraz)

nepoužito, vždy 0	D3	D2	D1	D0	
	3	2	1	0	číslo obrazu

P0PL,P1PL,P2PL,P3PL (kolizní registr hráč - hráč)

nepoužito, vždy 0	D3	D2	D1	D0	
	3	2	1	0	číslo hráče

Hráč n proti hráči n je vždy 0.

PMBASE (ukazovátko počátku PMG plochy)

Jednořádkové zobrazování

D7	D6	D5	D4	D3	nepoužito	
						PMBASE

Dvouřádkové zobrazování

D7	D6	D5	D4	D3	D2	nepoužito	
							PMBASE

PRIOR (registr priority)

D7	D6	D5	D4	D3	D2	D1	D0

D7-D6 nastaveny na 0

D5 je-li 1, pak se uplatňuje logická funkce OR mezi hráči č. Ø a 1 a mezi hráči č. 2 a 3. Umožňuje se překrývání pozic dvou hráčů

D4 povolen pátý hráč (P5). Nastavení bitu na 1 získají všechny střely barvu zapsanou v COLPF3. Střely se pak zviditelňují na odlišném barevném podkladě

D3-DØ výběr priority. Pouze jeden ze čtyř bitů může být nastaven na 1. D3-DØ vybírají jeden ze čtyř typů priority. Objekty s vyšší prioritou budou ukazovány jakoby "před" objekty s nižší prioritou.

	D3=1	D2=1	D1=1	DØ=1
	PFØ	PFØ	PØ	PØ
	PF1	PF1	P1	P1
	PØ	PF2	PFØ	P2
	P1	PF3+P5	PF	P3
	P2	PØ	PF2	PFØ
	P3	P1	PF3+P5	PF
	PF2	P2	P2	PF2
postup priority	PF3+P5	P3	P3	PF3+5
	BAK	BAK	BAK	BAK

P_n = hráč n
PF_n = obraz n
BAK = pozadí
\+ = funkce OR
(PF3 a P5 jsou vždy stejné barvy)

Sledující registr: GPRIOR

SIZEM (šířka střel)

D7	D6	D5	D4	D3	D2	D1	DØ
M3		M2		M		MØ	

registr horizontální šířky střel

Ø	Ø	normální šířka
Ø	1	dvojnásobná šířka
1	Ø	normální šířka
1	1	čtyřnásobná šířka

SIZEPØ - SIZEP3 (šířka hráčů)

nepoužito	D1	DØ

registr horizontální šířky hráčů

Ø	Ø	normální šířka
Ø	1	dvojnásobná šířka
1	Ø	normální šířka
1	1	čtyřnásobná šířka

5. DISPLAY-LIST INTERRUPT

5.1 ÚVOD

Display-list interrupt (obrazový podprogram, využívající přerušení), zkráceně DLI, vyžaduje od uživatele dobré znalosti systému, strojového jazyka atd. Poskytuje však neocenitelné služby ve spojení s PMG, barvovými registry, obrazovými znaky a dalšími specialitami počítače ATARI.

Je-li například potřeba rychle měnit některé parametry, na kterých závisí zobrazování, musí být změny synchronní se zobrazovacím cyklem. Tuto synchronizaci je schopen zajistit obrazový mikroprocesor ANTIC.

Přerušení je vyvoláno v okamžiku, kdy ANTIC narazí v DL na instrukci obsahující v bitu D7 log. 1. ANTIC ještě počká na provedení posledního televizního řádku a poté se zeptá registru NMIEN (nonmaskable interrupt enable) bitu D7, zda je nastaven na log. 1. Pokud ne, ANTIC žádost o přerušení ignoruje a pokračuje v další práci. V opačném případě se provede přerušení normální práce mikroprocesoru 65Ø2 a ANTIC bude pokračovat v obrazové rutině. 65Ø2 začne vykonávat nový program, jehož počátek je zapsán v paměti jako 16-ti bitová adresa. Ukazovátko DLI programu je na adrese 512, 513_D.
Po skončení DLI programu, (který je samozřejmě ve strojovém jazyku , se mikroprocesor vrátí ke své původně vykonávané činnosti.

5.2 VYTVOŘENÍ VLASTNÍHO DLI

Vlastní program musí začít úschovou registrů, které budou použity (příznakový registr není potřeba uschovávat, provedeno automaticky). Pro uložení podprogramu je nejvhodnější prostor na šesté stránce paměti. (adresa 1536 - 1791_D). Program končí návratem registrů procesoru 65Ø2 ze zásobníku na původní místa a závěrečnou instrukcí RTI (return from interrupt). Ukazovátko DLI programu na adrese 512 a 513_D se změní v našem případě na ØØ → 512 a Ø6 → 513 . V obrazovém programu DL je nutné změnit potřebnou instrukci tak, aby byla v bitu D7 log. 1. Nakonec se povolí přerušení nastavením bitu D7 na log. 1 v registru NMIEN (adresa 54286_D).

Práce se samotným programem není tak jednoduchá, jak by se na první pohled mohlo zdát. Při změnách hodnot barvových registrů by došlo ke změně barvy právě vykreslovaného nového televizního řádku a hranice barvy by byla nejistá, roztřepená. Problém snadno řeší registr WSYNC (wait for horizontal sync) na adrese D4ØA hexadecimálně. Je-li jakoukoliv cestou registr adresován, mikroprocesor 65Ø2 poté počká, až se dokončí právě se vykreslující televizní řádek a ihned potom pokračuje další instrukcí, která změní hodnotu kteréhokoliv barevného registru. WSYNC tak zaručí jednoznačné přechody barev televizních řádků.

Celý proces změny barvových registrů pomocí DLI ještě jednou:
Strojový program začíná úschovou registru mikroprocesoru

65Ø2 do zásobníku. Následuje naplnění registrů 65Ø2 čísly, které se po WSYNC přesunou do barevných registrů. Nyní se provede důležitá instrukce, nastavující WSYNC a hned po ní instrukce nastavující barvové registry. Na konci programu se ze zásobníku přesunou uschované vnitřní registry 65Ø2 a poslední instrukcí je RTI. Viz následující program:

```
PHA                    úschova akumulátoru
TXA
PHA                    úschova X registru
LDA # 5Ø               barvy      tmavá pro znaky
LDX # 58                          růžová
STA WSYNC              čekání
STA COLPF1             zápisy do systémových barvových
STX COLPF2                       registrů
PLA
TAX                    návrat registrů 65Ø2
PLA
RTI                    návrat z podprogramu
```

Strojový program použitý v ukázce lze přepsat do BASIC příkazu DATA a využít v následujícím programu.

Program č. 4

```
1Ø DLIST = PEEK(56Ø) + PEEK(561)*256     -adresa DL
2Ø POKE DLIST + 15, 13Ø                  -zápis instrukce
                                          interrupt
3Ø FOR I = Ø TO 19                       zápis DLI programu
4Ø READ A: POKE 1536 + I,A: NEXTI        na 6. str. paměti
5Ø DATA 72, 138, 72, 169, 8Ø, 162, 88
```

```
6Ø DATA 141, 1Ø, 212, 141 , 23, 2Ø8
7Ø DATA 141, 24, 2Ø8, 1Ø4, 17Ø, 1Ø4, 64
8Ø POKE 512, Ø: POKE 513, 6          -nast. ukazovátka
                                      DLI
9Ø POKE 54286, 192                   -povolení DLI
```

Po RUN dolní polovina obrazovky změní barvu z modré na růžovou a znaky změní barvu na barvu pozadí, ale s jiným jasem. Horní polovina obrazovky zůstává nezměněna. Proč? Během obrazového zatemňovacího impulsu provede operační systém podprogram, který přesune hodnoty z barvových registrů na adresách 7Ø8 ÷ 712_D do barvových registrů na adresách 5327Ø ÷ 53274_D což jsou tzv. systémové barvové registry (každý systémový barvový registr má svého "dvojníka" v operační paměti). To znamená, že se každý nový televizní obrázek začne s barvami zapsanými v barvových registrech operační paměti.

Není možné použít DLI okamžitě s prvním televizním řádkem.

5.3 Časování DLI

Program DLI je možné rozdělit do tří fází. První fáze trvá od počátku programu po instrukci STAWSYNC. Druhá fáze končí začátkem nového televizního řádku. Druhá fáze koresponduje s řádkovými zatemňovacími impulsy. Všechny grafické změny se dějí právě během této fáze. Třetí fáze začíná se začátkem nového tel. řádku

a končí s posledním strojovým příkazem DLI. Trvání třetí fáze není kritické.

Vykreslení jednoho televizního řádku trvá 114 strojových cyklů mikroprocesoru 65Ø2. Bylo by tedy žádoucí, aby 1. fáze netrvala déle, přičemž se provede přerušení (7 cyklů), úschovy registrů (11 cyklů), STAWSYNC (11 cyklů), občerstvování dynamických pamětí (9 cyklů) a dalších 2Ø cyklů k blíže neurčeným účelům. Zbývající čas lze s výhodou užít k vlastní potřebě.

Fáze 2 je časově značně náročná, neboť k dispozici je pouze 24 strojových cyklů. Z toho část zabere - pokud je použit - PMG 5 cyklů, obrazová instrukce 1 cykl (je-li použita LMS, spotřebuje 2 cykly), 1 až 2 cykly zabere občerstvování dynamických pamětí. V nejhorším případě zbývá 14 strojových cyklů, což postačuje k instrukcím STA a STX. Při změnách většího počtu barvových registrů je možné s určitým omezením využít i fáze 1 a fáze 3, ale pouze za předpokladu, že změny zrovna nebudou zobrazovány.

Pokud se programátor pustí do rozsáhlejších grafických změn, musí počítat se zvýšeným úsilím při stavbě a odpočítávání času DLI.

Jedním z řešení časové tísně druhé fáze je použití dvou jednodušších DLI programů, přičemž druhý DLI se vyvolá bezprostředně za prvním instrukcí single - scan - - line blank v programu DL. Tato instrukce spotřebuje část obrazového prostoru.

Jiný způsob se nabízí využitím času během snímkového zatemnění.

5.4 VÍCENÁSOBNÝ DLI

Pokud vyvstane potřeba použití více programů DLI, nastává problém, neboť ukazovátko DLI je jen jedno. Jsou-li programy DLI odlišné kupříkladu pouze v adresací registrů, dá se využít indexovaného adresování.

Viz příklad:

```
        PHA
        TXA
        PHA
        INC COUNTR
        LDX COUNTR
        LDA COLTAB,X          použití stránky FØ pro tabulku barev
        STA WSYNC             čekání
        STA COLBAK
        CPX #4F               poslední řádek tabulky?
        BNE ENDDLI            ne, konec podprogramu
        LDA #ØØ               ano, vynulování čítače
        STA COUNTR
ENDDLI  PLA
        TAX
        PLA                   návrat ukazovátka
        RTI
```

Strojový program v ukázce je použit v BASIC programu.

Program č. 5

```
 1Ø GRAPHICS 7
 2Ø DLIST = PEEK(56Ø) + PEEK(561)*256        adresa DL
```

```
30 FOR J = 6 TO 84
40 POKE DLIST + J, 141
50 NEXT J
```

ke každému obrazovému řádku přidána DLI

```
60 FOR J = 0 TO 30
70 READ A: POKE 1536 + J,A:NEXT J
80 DATA 72, 138, 72, 238, 32, 6, 175, 32, 6
90 DATA 189, 0, 240, 141, 10, 212, 141, 26, 208
100 DATA 224, 79, 208, 5, 169, 0
110 DATA 141, 32, 6, 104, 170, 104, 64
```

DLI program na stránku 6 paměti

```
120 POKE 512,0: POKE 513,6
```

změna ukazovátka DLI

```
130 POKE 54286, 192
```

povolení DLI

Jiné řešení umožňuje použití čítače sloužícího jako podklad pro tabulku větvení DLI. Větvení se musí vykonat během první fáze DLI a to může vyvolat opět časovou tíseň.

Třetí způsob se používá nejčastěji a spočívá ve změně adresy ukazovátka DLI výhodně ve třetí fázi předcházejícího DLI.

6. ROLOVÁNÍ OBRAZU

6.1 ÚVOD

Každý, kdo aktivně pracuje s počítačem ATARI ví, že při prohlížení programů, uložených v paměti se data na obrazovce posunují zdola nahoru - rolují. Tento způsob rolování mají samozřejmě i další mikropočítače, ATARI však poskytuje navíc ještě další dva způsoby jednoduché práce s obrazovou pamětí a tím i obrazu. Je to Load Memory Scan (LMS), vlastně instrukce mikroprocesoru ANTIC umožňující obyčejné rolování a jemné rolování.

Běžné počítače využívají obyčejného rolování, tzn. že se celý řádek posunuje o celou svou výšku. V mnoha případech takový způsob rolování nevyhovuje, neboť celý obraz nepříjemně "poskakuje". Snahou konstruktérů počítačů je - alespoň pro některé speciální případy- - plynulý posuv obrazu nejrůznějšími programovými prostředky. Nejprve však k obyčejnému rolování.

6.2 OBYČEJNÉ ROLOVÁNÍ

Využívá instrukce LMS obrazového mikroprocesoru, přesněji její dva operandy, které tvoří adresu počátku obrazové paměti. Změnou adresy počátku obrazové paměti se vlastně obraz posunuje po ploše paměti. Jednoduchý program ukazuje, co provede změna adresy u instrukce LMS v DL:

Program č. 6

```
 1Ø  DLIST = PEEK(56Ø) + PEEK(561)*256   zjištění adresy DL
 2Ø  LMSLOW = DLIST + 4                  nižší byte adresy
                                         v LMS
 3Ø  LMSHIGH = DLIST + 5                 vyšší byte adresy
                                         v LMS
 4Ø  FOR I = Ø TO 255
 5Ø  POKE LMSHIGH, I                     změna vyššího byte
                                         LMS
 6Ø  FOR J = Ø TO 255
 7Ø  POKE LMSLOW, J                      změna nižšího byte
                                         LMS
 8Ø  FOR Y = 1 TO 5Ø: NEXT Y             časová prodleva
 9Ø  NEXT J
1ØØ  NEXT I
```

Obrazovka postupně zobrazí celý adresový prostor ovšem pochybovým způsobem, protože vertikální rolování se bude míchat s rolováním horizontálním. Čistě vertikální rolování umožňuje další krátký program:

Program č. 7

```
 1Ø  GRAPHICS Ø
 2Ø  DLIST = PEEK(56Ø) + PEEK(561) * 256
 3Ø  LMSLOW = DLIST + 4
 4Ø  LMSHIGH = DLIST + 5
 5Ø  SCREENLOW = Ø
 6Ø  SCREENHIGH = Ø
 7Ø  SCREENLOW = SCREENLOW + 4Ø
 8Ø  IF SCREENLOW < 256 THEN GOTO 12Ø
 9Ø  SCREENLOW = SCREENLOW - 256
1ØØ  SCREENHIGH = SCREENHIGH + 1
11Ø  IF SCREENHIGH = 256 THEN END
12Ø  POKE LMSLOW, SCREENLOW
13Ø  POKE LMSHIGH, SCREENHIGH
14Ø  GOTO 7Ø
```

Rovněž tento program postupně zobrazí celý paměťový prostor, data po obrazovce budou "putovat" zdola nahoru.

Horizontální rolování bez nežádoucího vertikálního rolování vyžaduje kompletní změnu programu DL. Výrobu nového DL pro horizontální rolování s jeho spuštěním v modu 2 BASIC ukazuje tento program:

Program č. 8

```
 1Ø  REM                                  výroba nového DL
 2Ø  POKE 1536, 112: POKE 1537,112: POKE 1538,112
 3Ø  FOR I = 1 TO 12
 4Ø  POKE 1536 + 3*I, 71                  LMS mod 2 BASIC
 5Ø  POKE 1536 + 3*I + 1,Ø                nižší část adresy
 6Ø  POKE 1536 + 3*I + 2,I                vyšší část adresy
 7Ø  NEXT I
11Ø  POKE 1575, 65                        instrukce JVB
12Ø  POKE 1576,Ø: POKE 1577,6             adresa za JVB
13Ø  REM                                  změna adresy v ukazovátku
                                          DL
14Ø  POKE 56Ø,Ø: POKE 561,6
16Ø  REM                                  následuje horizontální
                                          rolování
17Ø  FOR I = Ø TO 235
18Ø  FOR J = 1 TO 12
19Ø  POKE 1536 + 3*J + 1,I
2ØØ  NEXT J: NEXT I: GOTO 17Ø
```

Řádek 2Ø udělá shora tři mrtvé řádky. Řádky 3Ø ÷ 7Ø přiřadí dvanácti instrukcím v novém DL adresy zobrazovaného prostoru. Jeden řádek obrazu bude mít na starosti zobrazovat jednu stránku paměti, tj. 256 byte. Samozřejmě nikoli najednou, nýbrž jen 2Ø-ti bytový výsek.

Řádky 11Ø a 12Ø vytvoří v DL instrukci s adresou skoku 1536_D, tj. počátek nového DL.

Řádek 14Ø změní adresu počátku programu DL na 1536 a od tohoto okamžiku ANTIC provádí zobrazování podle nově vytvořeného DL.

Řádky 17Ø ÷ 2ØØ zajišťují horizontální zobrazování. Data rolující zprava doleva, jsou vlastně data prvních dvanácti stránek paměti RAM. Rolování by vypadalo lépe, kdyby se řádky 17Ø ÷ 2ØØ nahradily programem v assembleru. Posun všech řádků o jednu pozici by pak proběhl najednou.

Horizontální i vertikální rolování pomocí změny adresy za instrukcí LMS v programu DL je poměrně rychlou a jednoduchou záležitostí.

6.3 JEMNÉ ROLOVÁNÍ

Zatímco obyčejné rolování posunuje obraz vždy nejméně o jeden obrazový znak, jemné rolování provádí posuv obrazu nejméně o jeden obrazový řádek ve vertikálním směru.

Při rolování celou obrazovkou je nutné kombinovat jemné rolování s obyčejným.

K uskutečnění jemného rolování je potřeba udělat dva kroky. První spočívá v nastavení bitů v DL instrukcích definujících mod řádků (těch, se kterými chceme rolovat). Pro jemné rolování celé obrazovky v modu Ø BASIC je tedy nutné nastavit všech 23 instrukcí.

Bitem D5 v instrukci se nastavuje vertikální rolování, bitem D4 horizontální rolování. V druhém kroku se do registru rolování zapisuje hodnota velikosti rolování.

Pro potřeby jemného rolování jsou k dispozici: registr horizontálního rolování (HSCROL) s adresou $D4Ø4_H$ tj. 54276_D a registr vertikálního rolování (VSCROL) s adresou $D4Ø5_H$, tj. 54277_D.

Při aplikaci jemného rolování vyvstává problém nesprávného formátování obrazu. Náš počítač může pracovat se třemi různými šířkami barevného bodu: normálním 16Ø bodů (1 barevný bod je široký 2 obrazové body), úzkým 128 bodů a širokým 192 bodů. Tyto šířky lze nastavit v registru SDMCTL s adresou $22F_H$, tj. 559_D (nebo ve DMACTL s adresou $D4ØØ_H$). Jestliže je nyní SDMCTL nastaven na normální šířku 16Ø barevných bodů, v BASIC moduØ se zobrazuje 4Ø znaků na řádek. Avšak ANTIC zobrazí každý řádek nastavený k jemnému rolování jako 48 bytový. Takovéto nesprávné formátování obrazu lze obejít např. vytvořením vlastního DL způsobem podobným předchozímu programu.

Jemné rolování lze provádět pouze o 16 obrazových řádků vertikálně nebo o 16 barvových bodů v horizontálním směru. Při pokusu o překročení této hranice ANTIC jednoduše vyšší bity v registrech VSCROL a HSCROL ignoruje . Z toho vyplývá, že při jemném rolování celé obrazovky o více než 16 obrazových řádků je nutné použít rovněž obyčejné rolování (v okamžiku, kdy je VSCROL naplněn, musí být vynulován a poté provedeno obyčejné rolování).

Níže uvedený program ukazuje jemné rolování jak v horizontálním, tak ve vertikálním směru se všemi dříve popsanými problémy.

Program č. 9

```
  1  HSCROL = 54276 : VSCROL = 54277
 1Ø  GRAPHICS Ø : LIST
 2Ø  DLIST = PEEK(56Ø) + PEEK(561) * 256
 3Ø  POKE DLIST + 1Ø, 5Ø                  nastavení bitů D4
 4Ø  POKE DLIST + 11, 5Ø                  a D5 a ANTIC 2
 5Ø  FOR Y = Ø TO 7
 6Ø  POKE VSCROL, Y                       vertikální rolování
 7Ø  FOR J = 1 TO 1ØØ : NEXT J            časová prodleva
 8Ø  NEXT Y
 9Ø  FOR X = Ø TO 3                       horizontální rolo-
1ØØ  POKE HSCROL, X                       časová prodleva vání
11Ø  FOR J = 1 TO 1ØØ : NEXT J
12Ø  NEXT X
13Ø  GOTO 5Ø
```

Nejvhodnější je použít jemné rolování ve spojitosti se snímkovými zatemňovacími impulsy a problém tímto svěřit strojovému programu.

6.4 APLIKACE

Autor (Chris Crawford) popisuje svůj program v BASIC modu 2. V paměti má uloženu mapu SSSR o velikosti deseti obrazovek. S pomocí páky se lze po zabraném paměťovém prostoru volně pohybovat. Program není přitom nikterak náročný na paměť počítače. Data, nový DL, nové obrazové znaky - to vše zabírá pouhé 4 KBRAM. Touto cestou lze rovněž realizovat velká elektronická schémata, apod.

7. A T A R I B A S I C

7.1 ÚVOD

ATARI BASIC pracuje stejně jako u většiny ostatních osobních mikropočítačů interpretačním způsobem. BASIC interpreter je uložen v paměti ROM od adresy $A000_H$ do adresy $BFFF_H$, tzn. že zabírá 8 KB paměti.

7.2 VÝHODY A NEVÝHODY ATARI BASIC

Aby se co nejlépe BASIC využil, je dobré znát jeho charakteristické rysy.

Mezi silné stránky ATARI BASIC patří:

* grafika pomocí jednoduchých BASIC příkazů
* hardware (technické vybavení) zajišťuje SOUND, STICK, PADDLE a jednoduché připojení periférií
* snadné volání strojových programů pomocí USR funkce

Mezi slabší stránky ATARI BASIC patří:

* není možná práce s tzv. celočíselnou aritmetikou. Všechna čísla jsou uchovávána jako 6-ti bytová čísla s pohyblivou řádovou čárkou
* protože všechna čísla jsou 6-ti bytová, probíhají všechny matematické operace pomalu
* řetězcové proměnné mohou být pouze jednorozměrné

7.3 PRÁCE INTERPRETERU

Dá se shrnout do tří základních bodů:

1. BASIC přijme zadání a přetransformuje ho do formy vhodné pro počítač (překlad).

2. Přeložené zadání se ukládá do určeného místa paměti
3. Překlad je k dispozici pro případné vykonání.

Detailní popis práce interpreteru bude proveden ve čtyřech následujících kapitolách:

* proces překládání
* struktura tabulek překladu
* způsob vykonávání programu
* komunikace s periferiemi

7.4 PROCES PŘEKLÁDÁNÍ

* BASIC přijme zadání
* kontrola správnosti zadání - kontrola syntaxu
* během kontroly je zadání překládáno
* překlad je ukládán na určené místo paměti
* je-li zadání v přímém modu, ihned se vykoná

K lepšímu pochopení překladového procesu je nutné definovat některé pojmy:

VÝROK -	kompletní příkaz nebo funkce např. INPUT A$ nebo STRIG (Ø). Nachází-li se v jednom BASIC řádku více výroků, jsou odděleny dvojtečkami
BASIC ŘÁDEK -	jeden nebo více výroků seřazených za sebou a oddělených dvojtečkami. Na počátku BASIC řádku je uvedeno číslo řádku v rozsahu Ø ÷ 32 767. Při zadání v přímém modu číslo řádku chybí
EOL (end of line) -	v překladu znak pro konec BASIC řádku

PŘÍKAZ
OPERÁTOR
FUNKCE

význam těchto výroků nejlépe pochopíte z tabulky č. VII

Překlad začíná přijetím zadání. Zadání je běžně prováděno skrze obrazový displej, ale stejně tak je možné zadávat pomocí příkazu ENTER z magnetofonu. Data se pak nezapisují rovnou do prostoru paměti, kde je překlad uložen, ale jsou uloženy nejprve do vstupního bufferu o adrese $14Ø8_D \div 1535_D$. Při bezchybném příjmu se pak operační systém postará o přesun informací z bufferu do určeného místa paměti a o další příjem do vstupního bufferu.

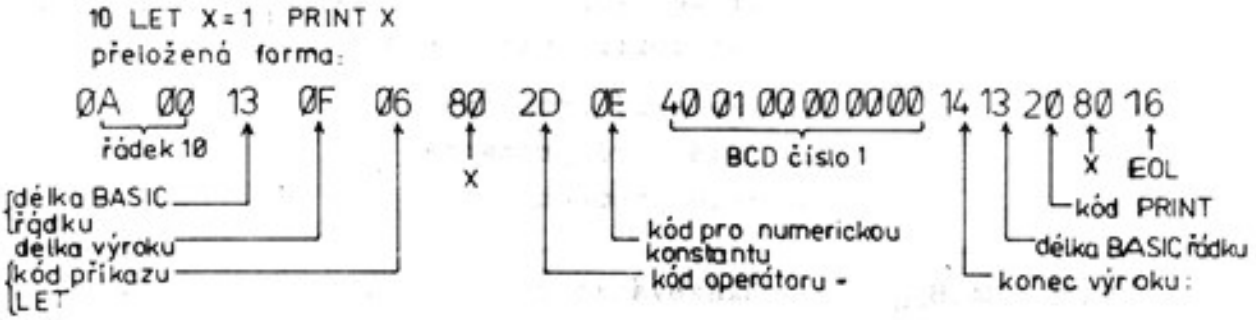
* Po přijetí zadání započne syntaxní kontrola a překladový proces. (Viz obr. 1) Jako první se přeloží číslo řádku do celočíselné 2-bytové podoby. Není-li číslo řádku zadáno, počítač považuje zadání v přímém modu za řádek $32\ 768_D$ a ihned jej vykoná. Takový řádek se vůbec do prostoru paměti překladu nedostane, ale rovnou se provede v tzv. výstupním bufferu překladu, vyhrazeném operačním systémem. Buffer má velikost 256 byte.
* Do překladu se vloží kompletní bytová délka celého BASIC řádku.
* Vloží se bytová délka výroku BASIC.
* Nyní BASIC vyhledá v ROM tvar vloženého příkazu. Pokud příkaz nalezne, jeho kód zapíše na další pozici překladu. V opačném případě se překlad zastaví, chyba se v BASIC řádku označí negováním některého ze znaků, který chybu vytvaří a celý BASIC řádek se i s nápisem ERROR vytiskne.

Příkazy			Operátory			Funkce		
HEX.	DEC.	NÁZEV	HEX.	DEC.	NÁZEV	HEX.	DEC.	NÁZEV
ØØ	Ø	REM	ØE	14	číselná konstanta	3D	61	STR$
Ø1	1	DATA	ØF	15	řetězová konstanta	3E	62	CHR$
Ø2	2	INPUT	1Ø	16	nevyužito	3F	63	USR
Ø3	3	COLOR	11	17	nevyužito	4Ø	64	ASC
Ø4	4	LIST	12	18	:	41	65	VAL
Ø5	5	ENTER	13	19	$	42	66	LEN
Ø6	6	LET	14	2Ø	konec výroku	43	67	ADR
Ø7	7	IF	15	21	:	44	68	ATN
Ø8	8	FOR	16	22	konec řádku	45	69	COS
Ø9	9	NEXT	17	23	GOTO	46	7Ø	PEEK
ØA	1Ø	GOTO	18	24	GOSUB	47	71	SIN
ØB	11	GOTO	19	25	TO	48	72	RND
ØC	12	GOSUB	1A	26	STEP	49	73	FRE
ØD	13	TRAP	1B	27	THEN	4A	74	EXP
ØE	14	BYE	1C	28	#	4B	75	LOG
ØF	15	CONT	1D	29	<=	4C	76	CLOG
1Ø	16	COM	1E	3Ø	<> — číselné	4D	77	SQR
11	17	CLOSE	1F	31	>= porovnávání	4E	78	SGN
12	18	CLR	20	32	<	4F	79	ABS
13	19	DEG	21	33	>	5Ø	8Ø	INT
14	2Ø	DIM	22	34	=	51	81	PADDLE
15	21	END	23	35	^	52	82	STICK
16	22	NEW	24	36	*	53	83	PTRIG
17	23	OPEN	25	37	+	54	84	STRIG
18	24	LOAD	26	38	-			
19	25	SAVE	27	39	/			
1A	26	STATUS	28	4Ø	NOT			
1B	27	NOTE	29	41	OR			
1C	28	POINT	2A	42	AND			
1D	29	XIO	2B	43	(			
1E	3Ø	ON	2C	44	)			
1F	31	POKE	2D	45	= /přiřazení čísla/			
2Ø	32	PRINT	2E	46	= /přiřazení řetězce/			
21	33	RAD	2F	47	<=			
22	34	READ	3Ø	48	<>			
23	35	RESTORE	31	49	>= — řetězové			
24	36	RETURN	32	5Ø	< porovnávání			
25	37	RUN	33	51	>			
26	38	STOP	34	52	=			
27	39	POP	35	53	+ — zvláštní			
28	4Ø	?	36	54	- operátory			
29	41	GET	37	55	(levá závorka řetězce			
2A	42	PUT	38	56	(levá závorka pole			
2B	43	GRAPHICS	39	57	(levá závorka příkazu DIM pole			
2C	44	PLOT	3A	58	(funkce levé závorky			
2D	45	POSITION	3B	59	(levá závorka příkazu DIM řetězce			
2E	46	DOS	3C	6Ø	,čárka v DIM poli			
2F	47	DRAWTO						
3Ø	48	SETCOLOR						
31	49	LOCATE						
32	5Ø	SOUND						
33	51	LPRINT						
34	52	CSAVE						
35	53	CLOAD						
36	54	/IMPLIED LET/						
37	55	ERROR-/SYNTAX/						

TAB. VII Tabulka překladových hodnot BASIC

* Po správném příkazu může následovat numerická hodnota, proměnná, operátor, funkce, úvozovky, další výrok, nebo EOL.
* Nenásleduje-li numerická hodnota, porovnává BASIC daný výraz s tabulkou jmen proměnných. Není-li výraz nalezen, zjišťuje dále BASIC, zda nejde o funkci či operátor. Pokud ne, pochopí BASIC, že jde o jméno nové proměnné. Každá proměnná je umístěna v tabulce a je pro ni rezervováno 8 byte. Tabulka je maximálně pro 128 proměnných.
* Následuji-li úvozovky, BASIC pochopí, že další byte budou tvořit znakový řetězec zakončený opět uvozovkami.
* Bude-li se jednat o numerickou hodnotu, BASIC ji přetransformuje do 6-bytové BCD konstanty. Do překladu se nejprve vloží byte, znamenající, že následuje numerická konstanta a za ním přetransformované BCD číslo.
* Jestliže je překlad jednoho BASIC řádku delší než 256 byte, vyhlásí se ERROR 14 a BASIC čeká na přijetí nového BASIC řádku.

Příklad jak vypadá jeden BASIC řádek ukazuje obr. 1



7.5 STRUKTURA TABULEK PŘEKLADU

Na nulové stránce paměti na adresách 8Ø ÷ 91_H, tj. 128 ÷ 145_D je devět 2 bytových ukazovátek (LOMEN, VNTP atd.), ukazujících počátky adres bufferů, tabulek atd.

TAB č. VIII

BASICem běžně používána ukazovátka v nulové stránce paměti	
LOMEN (8Ø,81_H)	- Výstupní buffer BASIC překladu - Užívá se pro překlad jednoho BASIC řádku. Délka je 256 byte a začíná na konci RAM vyhražené pro OS.
VNTP (82,83_H)	- Tabulka názvů proměnných - Jedná se o seznam názvů všech proměnných vyskytujících se v programu a jsou uloženy ve formě ATASCII kódu. Každý název je uložen v pořadí v jakém do programu vstoupil. Jedná se o: 1. Skalární proměnné 2. Řetězcové proměnné 3. Pole proměnných
VNTD (84,85_H)	- Ukazovátko konce tabulky VNTP - Pokud se v tabulce vyskytuje méně než 128 proměnných, ukazovátko je naplněno náhradní (dummy) adresou. Při 128

proměnných ukazuje na poslední byte posledního názvu proměnné.

VVTP ($86,87_H$) - Tabulka hodnot proměnných - Obsahuje všechny základní informace o každé proměnné pro níž je v tabulce rezervováno 8 byte.

BYTE	1	2	3 4	5 6	7 8
SKALAR	ØØ	VAR#	6-ti bytová BCD hodnota		
POLE					
(dimenzované)	41	VAR#	adresa od STARPu(8C, 8D)	první DIM+1	druhý DIM+1
(nedimenzované)	4Ø				
ŘETĚZEC					
(dimenzovaný)	81	VAR#	adresa od STARPu(8C, 8D)	délka	DIM
(nedimenzovaný)	8Ø				

Skalární proměnná obsahuje numerickou hodnotu. Např. X = 1. Skalárem je X a jeho hodnota = 1 a je uložena jako 6-ti bytové BCD číslo.
Řetězec je složen ze znaků a jeden znak zabírá jeden byte. Pole je složeno z číselných hodnot nebo z řetězcových proměnných.
První byte v tabulce udává druh proměnné, druhý byte číslo proměnné. V případě skaláru je 3 ÷ 8 byte BCD číslo. Pro pole a řetězce obsahuje 3. a 4. byte počátek (adresu) daného pole nebo řetězce.

	Pro pole obsahuje 5. a 6. byte jeho první dimenzovaný rozměr a 7. a 8. byte druhý dimenzovaný rozměr. Pro řetězce udává 5. a 6. byte délku řetězce a 7. a 8. byte dimenzovaný rozměr.
STMTAB(88,89_H)	- Tabulka výroků - Tento blok dat obsahuje všechny přiložené BASIC řádky. Dočasně zahrnuje rovněž BASIC řádek v přímém modu.
STMCUR(8A,$8B_H$)	- Běžný výrok - Toto ukazovátko má adresu právě vykonávaného výroku v tabulce výroků STMTAB.
STARP(8C,$8D_H$)	- Řetězcová plocha (paměť řetězcových proměnných) - blok obsahuje vlastní řetězce a pole. Řetězcové znaky jsou 1-bytové, pole 6-ti bytové pro jedno číslo. Plocha je následně zvětšována každým výrokem DIM.
RUNSTK(8E,$8F_H$)	- Zásobníková paměť pro FOR-NEXT a GOSUB .Pro FOR-NEXT je zde rezervováno 16 byte. První šestice bytů je BCD číslo počátku čítání, druhá šestice bytů je rovněž BCD číslo a udává velikost kroku.Třináctý byte je kód pro

		proměnnou čítače. Následující 2byte představují adresu příkazu FOR a poslední byte udává pozici FOR od počátku přiloženého řádku. Příkaz GOSUB vyžaduje rezervaci čtyř bytů. První je Ø a začíná jím každý GOSUB. Následující dva byte udávají celočíselnou hodnotu volaného řádku. Poslední byte udává pozici GOSUB od počátku přeloženého BASIC řádku.
MEMTOP(9Ø,91_H)	-	Vrchol použité paměti RAM - Program se dále může ukládát od této adresy až po počátek programu DL. Funkce FRE využívá k výpočtu volné paměti právě MEMTOP a HIMEM(2E5, $2E6_H$).

7.6 ZPŮSOB VYKONÁVÁNÍ PROGRAMU

BASIC přijímá dříve jím uložený překlad, rozpozná, ve kterém místě překladu se právě nachází a na základě skokové tabulky podprogramů vykonává dané příkazy. Na právě vykonávaný výrok ukazuje STMCUR. První řádek programu je vždy RUN nebo GOTO v přímém modu.

Přeložený program se nachází v prostoru, jehož počátek ukazuje STMTAB a jednotlivé přeložené BASIC řádky jsou zde uloženy vzestupně podle čísla řádku.

Stiskem tlačítka BREAK se vykonávaný program přeruší, neboť po provedení každého výroku operační systém tlačítko

testuje. Na obrazovce se po přerušení programu vytiskne nápis STOPPED AT LINE XXXX. Příkazem CONT se program může rozběhnout dalším následujícím řádkem.

7.7 KOMUNIKACE S PERIFERIEMI

BASIC komunikuje s operačním systémem přes tzv. IOCB (input/output control block). Následující tabulka ukazuje příkazy, které používají IOCB.

TAB č. IX

BASIC příkaz	Parametry OS - IOCB
OPEN 1,12,Ø,"E:"	IOCB=1; příkaz = 3(OPEN) tax 1=12(vstup/výstup); tax 2=Ø; ADR("E:")
GET 1,X	IOCB = 1; příkaz = 7(CET) znak uložen do X
PUT 1,X	IOCB = 1; příkaz = 11(PUT) znak z X vyslán
INPUT 1,A	IOCB = 1; příkaz = 5(příjem znaků) délka A $ nesmí být větší než 256 byte
PRINT 1,A	IOCB = 1; vyslání série znaků uložené v A $
XIO 18, 6,12,Ø, "S:"	IOCB = 6; příkaz = 18(FILL) Anx 1 = 12 Anx 2 = Ø

IOCB je vlastně tabulkou hodnot, či informací využita k řízení toku informací mezi počítačem a periferním zařízením. Při nahrávání BASIC programu příkazy SAVE

nebo CSAVE do periférního zařízení jsou zapisovány dva bloky informací. První blok sestává ze sedmi ukazovátek uvedených v tab. č. VIII. Jejich obsah však není zapsán jako absolutní adresa, nýbrž tak, že v LOMEN je změněna adresa na $ØØØØ_H$ a další adresy v ukazovátcích jsou vyšší o svůj rozdíl. Druhý blok sestává pak z tabulky názvů proměnných, tabulky hodnot proměnných, tabulky výroků (přeložený BASIC program) a řádku v přímém modu.

Při nahrávání programu do paměti počítače příkazy LOAD nebo CLOAD přečte nejprve operační systém adresu v MEMLO (počátek volné paměti RAM), k této adrese pak přičítá relativní adresy ukazovátek, které z periferního zařízení přicházejí jako data a ukládá je na svá místa ve stránce nula paměti RAM.

7.8 RŮZNÁ ZLEPŠENÍ PROGRAMU

Někdy se stane, že doba vykonávání programu je příliš dlouhá, nevyhovuje. Program je potřeba upravit. Nejprve je si však nutné uvědomit, zda čas strávený nad úpravou programu bude stát zato:

* vylepšování algoritmu vykonávání programu
* často volané podprogramy a smyčky FOR/NEXT je vhodné umístit k počátku programu, neboť BASIC hledá řádek vždy od počátku přeloženého programu
* místo často volaných krátkých podprogramů je vhodnější vykonat rutinu přímo. BASIC se tak vyhne přesunům informací přes RUNSTK.
* vykonávání programu lze zkrátit řazením výroků za sebou do jednoho řádku

* významné zkrácení exekuce programu asi o 30 % umožní vyloučení zobrazování příkazem POKE 559,Ø.
* podobně se dá ušetřit zkrácením DL na nezbytné minimum.
* použití strojového jazyka často zkracuje vykonávání i tisíckrát.

Další doporučení se týkají úspor programové paměti:

* zkuste nahradit čísla požita jen jednou nebo dvakrát aritmetickou kombinací dříve definovaných proměnných. Např. Z1 = 1, Z2 = 2. Potřebujete 3, pak napíšete Z1 + Z2.
* při častých voláních stejného řádku příkazem GOTO (nebo GOSUB) se s úsporou asi 5 byte nahradí řádek definovaný za GOTO (GOSUB) proměnnou. Např. místo 1ØØ se napíše Z1ØØ a na počátku programu se provede Z1ØØ = 1ØØ.
* čím méně proměnných, tím lépe. Každá proměnná vyžaduje 8 byte plus další byte pro název proměnné.
* jména proměnných definujte co nejúsporněji. Každý znak v názvu je jeden byte navíc.
* třikrát, nebo více použitou stejnou konstantu je vhodné nahradit proměnnou.
* větší množství jednodušších proměnných uložte do DATA. Zde je každý znak 1 byte, zatímco proměnná zabírá vždy 7 byte.
* větší množství textů uložte do řetězových proměnných
* pokud to nevyžaduje přehlednost programu, je vhodné seřadit co nejvíce výroků do jednoho BASIC řádku. Každý vypuštěný řádek ušetří 3 byte.

* jednoduché celé numerické hodnoty (Ø ÷ 255) lze místo do pole vložit do řetězcové proměnné, použitím funkce CHR $. Numerickou hodnotu lze zpětně obdržet přes funkci ASC. Na jedno číslo vznikne úspora 6 bytů
* při znalosti operačního systému můžete snadno nahradit příkaz SETCOLOR příkazem POKE. Úspora je 8 bytů.
* Místo příkazu POSITION je v některých případech vhodnější použití pohybu kurzoru programově v PRINT "↑→→→↓". Příkaz POSITION vyžaduje 15 byte, jeden kurzorový znak jeden byte.
* krátké strojové rutiny (do 11Ø bytů) lze napsat: X = USR (ADR(XXXXX), 16) apod.

7.9 ZAJÍMAVÉ APLIKACE

Dobrá znalost BASIC systému umožní jednoduchým způsobem provádět velmi výkonné aplikace.

- Inicializace řetězce:

```
1Ø  DIM A $ (1ØØØ)
2Ø A $ (1) = "A": A$ (1ØØØ) = "A"
3Ø A $ (2) = A$
```

Tento kratičký program naplní celý řetězec A$ znakem "A".

- Programové vypuštění řádků

PROGRAM č. 10

```
1Ø REM PŘÍKLAD PRO PROGRAMOVÉ VYPUŠTĚNÍ ŘÁDKŮ
2Ø GRAPHICS Ø: POSITION 2,4
```

```
3Ø PRINT  7Ø : PRINT  8Ø : PRINT 90:PRINT "CONT"
4Ø POSITION 2,Ø
5Ø POKE 842,13 : STOP
6Ø POKE 842,12
7Ø REM TYTO ŘÁDKY
8Ø REM BUDOU
9Ø REM VYPUŠTĚNY
```

Program ukazuje na možnosti operačního systému programově vypouštět nebo vkládat řádky. POKE v řádku 5Ø uvede obrazový editor do stavu "nuceného čtení" informací z obrazovky. POKE v řádku 6Ø vrací řízení počítače do původního stavu (více informací poskytnou publikace PERSONAL COMPUTER SYSTEM, OPERATING SYSTEM atd.)

- PMG s řetězcem A

PROGRAM č. 11

```
 1ØØ DIM A $ (512),B $ (2Ø)
 11Ø X=X+1:READ A:IF A<>-1 THEN B $ (X,X)=CHR$ (A):GOTO 12Ø
 12Ø DATA Ø,255,129,129,129,129,129,129,129,129,255,Ø,-1
2ØØØ POKE 559,62:POKE 7Ø4,88:I=PEEK(106)-16:POKE 54279,I
2Ø1Ø POKE 53277,3:POKE 71Ø,224
2Ø2Ø VTAB=PEEK(134)+PEEK(135)* 256:REM adr. v ukazovátku VVTP
2Ø3Ø ATAB=PEEK(14Ø)+PEEK(141)* 256:REM adr. v ukazovátku STARP
2Ø4Ø OFFS=I * 256 +1Ø24-ATAB
2Ø5Ø HI=INT(OFFS/256):LO=OFFS-HI * 256
2Ø6Ø POKE VTAB +2,LO:POKE VTAB +3,HI:REM adr. A $ do PMG plochy
3ØØØ Y=6Ø: Z=1ØØ:V=1:H=1
4ØØØ A$ (Y,Y+11)=B $ :POKE 53248,Z:REM  vert. a hor. pozice
                                                změněna
```

```
4Ø1Ø Y=Y+V:Z=Z+H
4Ø2Ø IF Y > 213 OR Y < 33  THEN V=-V
4Ø3Ø IF Z > 2Ø6 OR Z < 49  THEN H=-H
4Ø4Ø GOTO 4ØØØ
```

Malé okénko, jehož tvar je dán daty v řádku 12Ø se pohybuje po obrazovce. V řádku 4ØØØ se obrazec přemístí z B$ do definovaného místa A $, což je vlastně PMG plocha. Řádky 4Ø2Ø a 4Ø3Ø hlídají maximální nebo minimální velikost vertikální a horizontální polohy.

TAB č. X
znázorňuje některá ukazovátka používaná BASICem nebo operačním systémem

ukazovátka operačního systému			ukazov. BASIC	
jméno	adr.(hex)	6.stránka	adr.(hex)	jméno
MEMLO	2E7,2E8		8Ø,81	LOMEN
			82,83	VNTP
			84,85	VNTP
		překlad	86,87	VVTP
		BASIC	88,89	STMTAB
		programu	8A,8B	STMCUR
			8C,8D	STARP
			8E,8F	RUNSTK
APPMHI	ØE,ØF		9Ø,91	MEMTOP
		volná	ØE,ØF	APHM
		paměť		
MEMTOP	2E5,2E6	RAM	2E5,2E6	HIMEM
SDLST	23Ø,231	DL		
SAVMSC	88,89			
		obrazová		
TXTMSC	294,295	paměť		
		textové		
RAMTOP	6A	okénko		
RAMSIZ	2E4			

8. B A R V Y

8.1 ÚVOD

Domácí počítač ATARI umožňuje uživateli používat množství barev a barevných odstínů díky výkonnému systému barvových registrů (o barvy se stará obvod GTIA montovaný do všech počítačů ATARI od počátku roku 1982).

8.2 PRINCIP BAREVNÉHO PŘENOSU

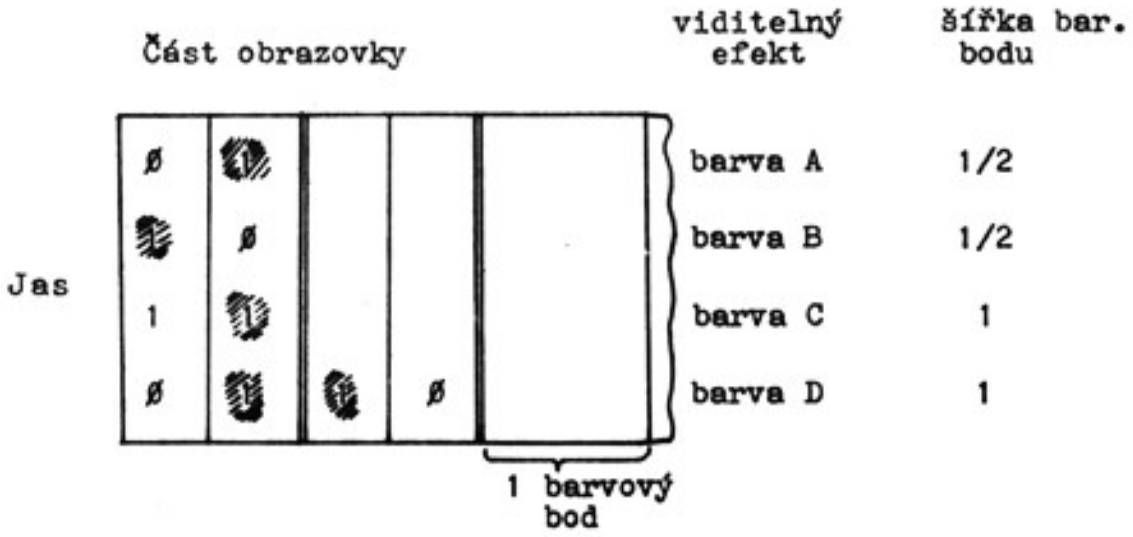
Na začátek krátký program:

```
GRAPHICS 8
COLOR 1
POKE 71Ø,Ø
PLOT 6Ø,6Ø
PLOT 63,6Ø
```

Zajímavá věc. Každý z bodů vytvořených stejným příkazem PLOT má jinou barvu. Proč?

Modulace barevného televizního signálu obsahuje dvě základní složky: informaci o jasu (barevném odstínu) a barvě. Jasová informace je primární a nese rovněž časové a synchronizační impulsy. Informace o barvě jsou také vneseny do jasové informace. Jas bodu na obrazovce je přímo závislý na amplitudě jasového signálu; vyšší amplituda, vyšší jas. Informace o barvě je zakódována pomocí fázové modulace s frekvencí 3.759 MHz (autoři originálu popisují normu NTSC). Amplitudově modulovaný signál jasu má frekvenci dvojnásobně vyšší tj. 7.16 MHz.

Zdálo by se, že jasová a barvová složka jsou signály na sobě nezávislé, není to však pravda. Jakákoliv změna jasu vyvolává drastické efekty ve fázově modulovaném barvovém signálu. Pro jeden nebo více barvových bodů s konstantním jasem problém není, horší je to při zobrazení osamocených bodů. Mohou nastat v podstatě čtyři možnosti, jak dokumentuje obr. č. 2 .



Barvy A ÷ D jsou rozdílné pro každý televizní přijímač. Předchozí obrázek dokládá program č. 12.

Program č. 12

```
1ø GRAPHICS 8: POKE 87,7:POKE 71ø,ø:POKE 7ø9,14
2ø COLOR 1:PLOT 1ø,5:DRAWTO 1ø,7ø
3ø PLOT 4ø,5: DRAWTO 4ø,7ø
```

```
 4Ø COLOR 2:PLOT 2Ø,5:DRAWTO 2Ø,7Ø
 5Ø PLOT 41,5: DRAWTO 41,7Ø
 6Ø COLOR 3:PLOT 3Ø,5: DRAWTO 3Ø,7Ø
 7Ø FOR X = 1 TO 3:COLOR X: POKE 765,X
 8Ø PLOT X*25+6Ø,5: DRAWTO X*25+6Ø,7Ø
 9Ø DRAWTO X*25+4Ø,7Ø: POSITION X*25+4Ø,5
1ØØ XIO 18,#6,12,Ø,"S:"
11Ø NEXT X
```

Program kreslí čáry v barvě A ÷ D předchozího obrázku. Příkaz POKE 87,7 vnutí operačnímu systému mod 7, přestože paměť bude jako v modu 8. Řádky 7Ø ÷ 11Ø vytvoří pomocí příkazu XIO (využívá podprogram FILL operačního systému - - bude napsán dále) tři obdélníky v barvě A ÷ C.

8.3 MOD 9, 1Ø, 11

Všechny tři mody pracují v rastru 8Ø x 192 bodů, tzn. šířka k výšce bodu je 4 : 1. Každý řádek vyžaduje 32Ø bitů (tj. 4Ø byte) a celá obrazová paměť zabírá asi 8KB. Mody jsou vybrány podle nastavení registru PRIOR na adrese $DØ18_H$, nebo sledujícího registru na adrese $26F_H$ (tj. 623_D). (zde stojí za zmínku, že obsahy sledujících registrů jsou přepisovány každou 1/5Ø sekundy do jim přiřazených registrů v hardwarové struktuře počítače). Mod určují bity D6 a D7.

D7	D6	VOLBA
Ø	Ø	---
Ø	1	mod 9
1	Ø	mod 1Ø
1	1	mod 11

Možnost přímého nastavení registru PRIOR přijde vhod při práci se strojovým programem. V čem se jednotlivé grafické mody 9, 1Ø a 11 liší bude popsáno dále.

Grafický mod 9: Poskytuje 16 různých barevných odstínů (jasů) jediné barvy. Registr barvy pozadí se nastavuje příkazem

SETCOLOR 4, BARVA, Ø

přičemž BARVA je v rozsahu Ø ÷ 15. Příkaz COLOR umožňuje kreslení bodů v různých jasech. Program č. 13 ukazuje možnosti modu 9.

Program č. 13

```
1Ø GRAPHICS 9: SETCOLOR 4,12,Ø: REM <------zelená
2Ø FOR I = Ø TO 15
3Ø COLOR I
4Ø PLOT 2*I,1Ø:DRAWTO 2*I,8Ø
5Ø NEXT I
6Ø GOTO 6Ø
```

Ve strojovém programu se barva zapisuje do vyšších čtyř bitů barvového registru na adrese $2C8_H$ (tj. 712_D) a jas bodu se určí přes registr ATACHR na adrese $2FB_H$ (tj. 763_D) v rozmezí Ø ÷ 15.

Grafický mod 11: Poskytuje 16 barev v jediném odstínu (jasu). Příkaz SETCOLOR 4,Ø,JAS definuje jas bodů, přičemž JAS může být číslo Ø ÷ 15, počítačem však budou brány na zřetel sudá čísla (nejnižší bit je ignorován), tzn. že odstínů bude pouze 8. Krátký program dokumentuje mod 11.

Program č. 14

```
1Ø GRAPHICS 11: SETCOLOR 4,Ø,12: REM      jas bodů
2Ø FOR I = Ø TO 15
3Ø COLOR I:REM                            výběr barvy
4Ø PLOT 2 * I,1Ø: DRAWTO 2 * I,8Ø
5Ø NEXT I
6Ø GOTO 6Ø.
```

Ve strojovém programu se jas zapisuje do nižších čtyř bitů barvového registru na adrese $2C8_H$ (tj. 712_D) a barva bodu se určí přes registr ATACHR na adrese $2FB_H$ (tj. 763_D) v rozmezí Ø ÷ 15.

<u>Grafický mod 1Ø:</u> využívá všech devíti barvových registrů, přičemž každý z nich nese informaci jak o barvě, tak o jasu. Příkazem SETCOLOR mohou být nastaveny pouze barvové registry na adresách 7Ø8 ÷ 712_D. Registry na adresách 7Ø4 ÷ $7Ø7_D$ je nutné nastavit pomocí POKE. (Příklad nastavení bude uveden v další kapitole).
Příkazu COLOR lze využít k výběru jednotlivých barvových registrů. Například COLORØ vybírá registr s adresou $7Ø4_D$, COLOR 3 vybírá registr s adresou $7Ø7_D$ apod.
K lepšímu pochopení modu 1Ø krátký program č. 15:

```
1Ø GRAPHICS 1Ø
2Ø FOR I = Ø TO 8
3Ø POKE 7Ø4 + I, 255 * RND(1)
4Ø COLOR I
5Ø PLOT 2 * I,1Ø
6Ø DRAWTO 2 * I,8Ø
7Ø NEXT I
8Ø GOTO 8Ø
```

9. POHYBLIVÉ OBRAZY POMOCÍ BARVOVÝCH REGISTRŮ

9.1 ÚVOD

Pohyb obrázků nakreslených počítačem je vždy záležitost nelehká a což teprve barevných obrázků. Odměnou za zvládnutí této techniky bývají velmi efektní a zajímavé programy. ATARI má k dispozici celkem 9 barvových registrů a změnou jejich obsahů lze docílit změny barev velkých ploch obrazu a tím i zdánlivého pohybu obrazu.

Princip změny obsahů barvových registrů k vytvoření pohybu obrazu bude demonstrován v dalších odstavcích programy průchod příkopem (The Trench) a pohyb vody vodopádem a řečištěm (Fall Waterfall).

9.2 ROTACE BAREV

Princip je zcela jednoduchý a dal by se přirovnat k devíti dětem, přehazujících si v kruhu neustále dokola devět barevných míčů najednou. Děti představují barvové registry, barevné míče, barvy zapsané v registrech.

9.3 BARVOVÉ REGISTRY

Tabulka č. 11 ukazuje většinu grafických modů a jim přiřazené barvové registry. První sloupec udává mod pro BASIC nebo ANTIC. Ve druhém sloupci je stav barvových registrů v základním stavu operačního systému (po RESET nebo po zapnutí počítače). SETCOLOR ve třetím sloupci slouží k snadnému nastavení barvových registrů. POKE šetří paměť, ale ztrácí se přehlednost a kromě toho je třeba znát adresy barvových registrů. Příkaz COLOR slouží

v programech k snadnému výběru barvových registrů. Poslední sloupec popisuje přiřazení znaků, pozadí, okraje a bodu k příslušným barvovým registrům.

9.4 PRVNÍ VOLBA BAREV V PROGRAMU

Program č. 16

```
10 GRAPHICS 3 + 16
20 COLOR 2: PLOT 10,8
30 COLOR 1: DRAWTO 29,8
40 COLOR 3: PLOT 30,8
50 COLOR 0: PLOT 20,8
140 GOTO 140
```

Řádek 10 volí grafický mod s rastrem 40 x 24, což pro ukázku zcela vyhovuje. Řádek 20 nakreslí na obrazovce bod světle zelené barvy (nahlédni do tabulky č.XI). Řádek 30 nakreslí od zeleného bodu úsečku oranžové barvy. Řádek 40 nakreslí modrý bod a řádek 50 využívá ke kreslení barvy pozadí (černé).

9.5 SETCOLOR V PROGRAMU

ATARI má k dispozici 16 barev. Ke změně barvy v barvových registrech slouží příkaz SETCOLOR registr,barva,jas.

TAB. č. XII. přiřazuje jednotlivým barvám čísla pro SETCOLOR

TAB č. XI

MOD	první volba	SETCOLOR (n)	POKE adresa	COLOR (n)	vlastnosti
GRAPHICS 0 (textový mod s textové okno pro všechny mody)	světle modrá	1	709	normálně nepouž to	znak
	tmavě modrá	2	0		pozadí
	černá	4	12		okraj
ANTIC 4 a 5 GRAPHICS 12,13 (5 barev speciální textové mody)	oranžová	0	708	ormálně epoužito	znak
	světle zelená	1	709		znak
	modrá	2	710		znak
	červená	3	711		znak
	černá	4	712		pozadí, okraj
GRAPHICS 1 a 2 (velké textové mody 5 barev)	oranžová	0	708	normálně nepoužito	znak
	světle zelená	1	709		znak
	modrá	2	710		znak
	červená	3	711		znak
	černá	4	712		pozadí okraj
GRAPHICS 3,5a7 (čtyři barvy)	oranžová	0	708	1	bod
	světle ze ená	1	709	2	bod
	modrá	2	710	3	bod
	černá	4	712	0	bod (pozadí,okraj)
GRAPHICS 4 a 6 (2 barvy)	oranžová	0	708	1	bod
	černá	4	712	0	bod(pozadí, okraj)
GRAPHICS 8 (jedna barva, dva jasy)	světle modrá	1	709	1	jas bodu
	tmavě modrá	2	710	0	jas (pozadí)
	černá	4	712		okraj
GRAPHICS 9 (mode OTIA, jedna barva 16 jasů. Změna barvy s SETCOLOR 4,BARVA,0 nebo POKE 712, BARVA)	černá	4	712	0	bod(pozadí, okraj)
		-	-	1	bod
		-	-	2	bod
		-	-	3	bod
	tmavě šedá	-	-	4	bod
		-	-	5	bod
		-	-	6	bod
		-	-	7	bod
	šedá	-	-	8	bod
		-	-	9	bod
		-	-	10	bod
		-	-	11	bod
	světle šedá	-	-	12	bod
		-	-	13	bod
		-	-	14	bod
	bílá	-	-	15	bod

GRAPHICS 10 (mod GTIA, devět barev)	černá	-	704	0	bod(pozadí, okraj)
	černá	-	705	1	bod
	černá	-	706	2	bod
	černá	-	707	3	bod
	oranžová	0	708	4	bod
	světle zelená	1	709	5	bod
	modrá	2	710	6	bod
	červená	3	711	7	bod
	černá	4	712	8	bod
GRAPHICS 11 (mod GTIA, jeden jas; 16 barev. Změna jasu s SETCOLOR 4,0,JAS nebo POKE 712,JAS)	černá	4	712	0	bod(pozadí,okraj)
	světle oranžová	-	-	1	bod
	zlato oranžová	-	-	2	bod
	červeno oranžová	-	-	3	bod
	růžová	-	-	4	bod
	purpurová	-	-	5	bod
	purpurově modrá	-	-	6	bod
	azurově modrá	-	-	7	bod
	blankytně modrá	-	-	8	bod
	světle modrá	-	-	9	bod
	tyrkysová	-	-	10	bod
	zeleno modrá	-	-	11	bod
	světle zelená	-	-	12	bod
	žluto zelená	-	-	13	bod
	oranžově zelená	-	-	14	bod
	světle oranžová	-	-	15	bod

Ještě jeden program v PMG dokumentující práci registru GPRIOR

```
 1Ø REM VORBEREITUNGEN (INICIALIZACE)
 2Ø GRAPHICS 3 + 16
 3Ø A=PEEK(1Ø6) - 16: POKE 1Ø6,A
 4Ø POKE 559,62
 5Ø V = 53 248:POKE V + 29,3:POKE V + 1Ø31,A
 6Ø REM AUFBAU (KRESLENÍ)
 7Ø BEG = (A + 4)*256:Y = Ø
 8Ø FOR I = Ø TO 2
 9Ø FOR J = Ø TO 255
1ØØ POKE BEG + I*256+Y+J, 199 + ABS(I-1)*56
11Ø NEXT J
12Ø NEXT I
13Ø REM FARBEN (BARVY)
14Ø POKE 7Ø4,55: POKE 7Ø5,111: POKE 7Ø6,137
15Ø REM BEWEGUNG (POHYB)
16Ø FOR I = Ø TO 228
17Ø POKE V,228-I: POKE V+1,15Ø: POKE V+2,I
18Ø FOR J=1 TO 3Ø: NEXT J
19Ø NEXT I
2ØØ GOTO 16Ø
```

Podle časopisu HOME COMPUTER NR.9 SEPT.1984

TAB č. XII.

BARVA	ČÍSLO PRO SETCOLOR
šedá	Ø
světle oranžová (zlatá)	1
oranžová	2
červeně oranžová	3
růžová	4
nachová	5
nachově modrá	6
azurově modrá	7
blankytně modrá	8
světle modrá	9
tyrkisová	1Ø
zeleně modrá	11
zelená	12
žlutozelená	13
oranžově zelená	14
světle oranžová	15

Příklady použití ukazuje tabulka č. XIII.

TAB č. XIII.

PŘÍKAZ	BARVA	JAS	VÝSLEDNÁ BARVA
SETCOLOR 4,Ø,14	šedá	14	bílá
SETCOLOR 4,Ø,Ø	šedá	Ø	černá
SETCOLOR 4,1,4	světleoranž.	4	hnědá
SETCOLOR 4,1,12	světle oranž.	12	světležluté
SETCOLOR 4,3,4	červenšoranž.	4	tmavěčervená
SETCOLOR 4,3,12	červenšoranž.	12	červená

Následující řádky programu připište k programu č. 17

```
 5Ø COLOR 1:PLOT 2Ø,5: PLOT 2Ø,6
 6Ø PLOT 19,7: DRAWTO 21,7
 7Ø PLOT 19,9: DRAWTO 21,9
 8Ø SETCOLOR 1,3,6: SETCOLOR 2,12,6
 9Ø FOR I = 1 TO 5Ø: NEXT I
1ØØ SETCOLOR 1,Ø,Ø: SETCOLOR 2,Ø,Ø
11Ø FOR I = 1 TO 4ØØ
12Ø IF RND (Ø) * 2Ø < 1  THEN SETCOLOR 4,Ø,14
    SETCOLOR 4,Ø,Ø
13Ø NEXT I
14Ø GOTO 8Ø
```

Kresbička na obrazovce představuje letadlo z čelního pohledu s majáčky na koncích křídel, prolétavající bouří. Řádky 8Ø - 11Ø májáčky rozsvěcují a zhášejí. SETCOLOR 1 mění barvu bodu nakresleného příkazem COLOR 2 a SETCOLOR 2 mění barvu bodu nakresleného příkazem COLOR 3. Jestliže je v řádku 12Ø náhodné číslo větší než Ø.Ø5, změní se na okamžik barva pozadí příkazem SETCOLOR 4.

Celý efekt by bylo možné udělat překreslováním obrázku s novými barvami, ale tím by se ztratila rychlost vykonávání programu, i když by to v tomto případě nevadilo.

9.6 POUŽITÍ POKE KE ZMĚNÁM BAREV

Každý barvový registr má v paměti svou adresu. Hodnota v registru může být změněna příkazem SETCOLOR, nebo také příkazem POKE. V grafickém modu 1Ø je to dokonce

jediná cesta, jak změnit hodnoty prvních čtyř barvových registrů (704 ÷ 707). Příkaz POKE má tvar:

POKE adresa, barva * 16 + jas

Například v modu 7 je SETCOLOR 0,4,8 roven POKE 708,72 nebo SETCOLOR 4,3,12 je roven POKE 712,60.

9.7 POHYB POMOCÍ BARVOVÝCH REGISTRŮ - PROGRAM "PŘÍKOP"

Program č. 18

```
 10 REM                                  pohyb příkopem
 20 GOTO 200
100 REM                                  rotace barev
110 SOUND 3,255,0,8: REM                 zvukové pozadí
120 IF PEEK (753) = 0 THEN TEMP = PEEK (710):
    POKE 710,PEEK (709): POKE 709, PEEK (708):POKE 708,
    TEMP: GOTO 140
130 TEMP = PEEK (708): POKE 708, PEEK (709): POKE 709,
    PEEK (710): POKE 710,TEMP
140 PDL = 25: REM                  PDL je rychlost průchodu
150 SOUND 0,PDL,0,8: SOUND 1,PDL + 90,0,8:      příkopem
    SOUND 2,PDL + 160,0,8
160 FOR PAUSE = 1 TO PDL: NEXT PAUSE
170 GOTO 120
200 REM                                  inicializace
210 COL = 1: Y1 = 45: Y2 = 49
220 GRAPHICS 7 + 16
230 SETCOLOR 0,3,8: SETCOLOR 1,3,8: SETCOLOR 2,3,4
320 REM                                  kreslení příkopu
```

```
33Ø FOR X = 2 TO 79
34Ø COLOR INT (COL + 0.5)
35Ø PLOT X + 8Ø, Y1: DRAWTO 8Ø+X,Y2: DRAWTO 79-X,Y2: DRAWTO 79-X,
                                                      Y1
36Ø Y1 = Y1 - Ø.6: Y2 = Y2 + 0.6
37Ø IF Y1<Ø THEN Y1 = Ø
38Ø IF Y2>95 THEN Y2 = 95
39Ø COL = COL + (79 - X)/16Ø
4ØØ IF COL + Ø.5 >=4 THEN COL = COL - 3
41Ø NEXT X
42Ø GOTO 1ØØ
```

Program vytváří iluzi pohybu v nekonečném, rovném příkopu. Lze se pohybovat vpřed nebo vzad (podle stisknuté klávesy; řádek 12Ø) různou rychlostí volitelnou v řádku 14Ø.

Inicializace (2ØØ ÷ 23Ø)

Nejprve se nastaví registry použité při dalším kreslení příkopu. Poté se inicializuje grafický mod 7. V řádku 23Ø si povšimněte, že se sice nastavují tři barevné registry, ale pouze v jedné barvě a dvou odstínech. Světlejší plochy v obrazu pak budou dvakrát širší, než tmavší plochy.

Kreslení příkopu (32Ø ÷ 42Ø)

Kreslení začíná na horizontu (vlastně uprostřed obrazovky) a pokračuje směrem k pozorovateli. Vykreslují se dvě vertikální stěny příkopu a dno. COLOR postupně využívá všech tří nastavených barvových registrů.

Rotace barev (1ØØ ÷ 17Ø)

Rotaci vykonává buď řádek 12Ø, nebo 13Ø, podle toho, je-li klávesa stisknuta, nebo ne. Pokud máte k dispozici knipl,

bude vypadat řádek 14Ø takto:

```
14Ø PDL = PADDLE (Ø)/5.
```

Jinak lze použít páku nebo hardwarových tlačítek START, OPTION atd. s příslušným ošetřením.

Obrazový efekt je zvýrazněn ještě zapnutím všech čtyř zvukových kanálů. Část programu Rotace barev je umístěna na počátku programu, aby se prováděcí doba co nejvíce zkrátila (i když i v tomto případě je času k exekuci programu dost).

9.8 VODOPÁD S ŘEČIŠTĚM

V programu se bude využívat grafického modu 1Ø, tzn., že k dispozici bude všech devět barvových registrů. Na obrazovce bude podzimní krajina se stromy a jejich stíny v časném ranním slunci. Voda poteče vodopádem dolů, a pak přes zelené údolí.

Mod 1Ø pracuje v rastru 80 x 192 (poněkud nezvyklý poměr). Program je poměrně dlouhý, nejvíce paměti zabírá vykreslení krajiny. Samotný "tok" vody zabírá pouze tři řádky programu. Rozdělení barvových registrů: k rotaci barev (vyvolávání efektu "toku" vody) je použito čtyř registrů (7Ø5 ÷ 7Ø8). Registr pozadí (7Ø4) je samozřejmě modrý (obloha) a další čtyři registry mají barvu trávy, kmene stromu, stínu vrženého stromem a vodní pěny. Sráz vodopádu má stejnou barvu jako kmeny stromů. Více barvových registrů už není k dispozici.

K vykreslování velkých barevných ploch je použit

podprogram FILL, který má ATARI k dispozici ve svém operačním systému. Naneštěstí to není ten známý FILL nebo FLOOD, používaný v profesionálních grafických terminálech. FILL v ATARI klade na uživatele při jeho použití mnohá úskalí a problémy. V podstatě není k dispozici žádný přímý příkaz k provedení FILLu. Existuje pouze příkaz XIO, aktivující FILL, ale je možné jej použít až po některých dodatečných nastaveních. Je to nepohodlné, ale aspoň něco. (Více informací o FILL je možné získat v Computer Animation Primer, nebo v ATARI BASIC manual).

Program je sestaven ze tří podstatných částí: inicializace, kreslení krajiny a pohybu vody (rotace barev).

Program č. 19

```
 1Ø REM    *** VODOPÁD ***
 2Ø GOTO 2ØØ
1ØØ ROTACE BAREV
11Ø TEMP = PEEK (7Ø5) : POKE (7Ø5), PEEK (7Ø6): POKE 7Ø6,
   PEEK(7Ø7): POKE 7Ø7,PEEK(7Ø8):POKE 7Ø8, TEMP
12Ø FOR WT = 1 TO 5: NEXT WT
13Ø GOTO 11Ø
2ØØ REM                                  inicializace
21Ø FILL = 13ØØ
22Ø GRAPHICS 1Ø
23Ø POKE 7Ø4,9 * 16+1Ø: REM    obloha - COLORØ
24Ø POKE 7Ø5,8 * 16+1Ø: REM    voda - COLOR1
25Ø POKE 7Ø6,8 * 16+ 8: REM    voda - COLOR2
26Ø POKE 7Ø7,8 * 16+ 6: REM    voda - COLOR3
27Ø SETCOLOR Ø,8,4: REM        voda - COLOR4
28Ø SETCOLOR 1,12,4: REM       stíny stromů - COLOR5
```

```
290 SETCOLOR 2,2,4: REM          útes a kmeny stromů - COLOR6
300 SETCOLOR 3,12,6: REM         tráva - COLOR 7
310 SETCOLOR 4,3,6: REM          koruna stromů - COLOR 8
400 REM                          kreslení trávy a útesu
410 COLOR 7: POKE 765,7: REM     tráva
420 PLOT 79,10: DRAWTO 79,45 : X1 = 78:
    Y1 = 10: X2 = 66: Y2 = 15: GOSUB FILL
430 X1 = 65: Y1 = 15: X2 = 61: Y2 = 18: GOSUB FILL:
    X1 = 60: Y1 = 18: X2 = 56: Y2 = 25: GOSUB FILL
440 X1 = 56: Y1 = 25: X2 = 65: Y2 = 35: GOSUB FILL:
    X1 = 66: Y1 = 35: X2 = 78: Y2 = 45: GOSUB FILL
450 COLOR 6: POKE 765,6: REM     útes
460 PLOT 79,46:DRAWTO 79,145:X1 = 56: Y1 = 26: X2 =56:
    Y2 = 117: GOSUB FILL
470 Y1 = 117: X2 = 68: Y2 = 132: GOSUB FILL: X1 = 68:
    Y1 = 132: X2 = 78: Y2 = 145: GOSUB FILL
480 COLOR 7: POKE 765,7:REM      více trávy
490 PLOT 0,191: DRAWTO 79,191: DRAWTO 79,146: X1 = 0:
    Y1 = 191: X2 = 0: Y2 = 91: GOSUB FILL
500 REM                          kreslení svahu a řeky
510 FALL = 58: CFLAG = 0:REM     kreslení řeky na vrcholu útesu
520 FOR Y = 25 TO 34
530 GOSUB 1500
540 FOR X = 79 TO FALL STEP - 1
550 COLOR COL
560 PLOT X,Y
570 COL = COL - 1: IF COL = 0 THEN COL = 4
580 NEXT X
590 FALL = FALL + 1
600 NEXT Y
610 FALL = 0: CFLAG =-1:REM      kreslení vody ve vodopádu
```

```
62Ø FOR X = 58 TO 66
63Ø FALL = FALL + 1
64Ø GOSUB 15ØØ
65Ø PLOT X,25 + FALL
66Ø FOR Y = 3Ø TO 12Ø STEP 4
67Ø COLOR COL
68Ø DRAWTO X,Y + FALL
69Ø COL = COL - 1: IF COL = Ø THEN COL = 4
7ØØ NEXT Y: NEXT X
71Ø COLOR 6: PLOT 58,28: DRAWTO 58,25: DRAWTO 59,25:
    PLOT 66,38: DRAWTO 66,129: REM      úklid
72Ø COLOR 7: PLOT 73,33: DRAWTO 79,33: PLOT 68,34:
    DRAWTO 79,34
73Ø FALL = 57: CFLAG = 1: REM            kreslení řeky v údolí
74Ø FOR Y = 121 TO 128: GOSUB 15ØØ: FOR X = FALL TO
    Ø STEP - 1: COLOR COL: PLOT X,Y: COL = COL - 1:
    IF COL = Ø THEN COL = 4
75Ø NEXT X: FALL = FALL + 1: NEXT Y
9ØØ REM                                  kreslení stromu
91Ø FOR T = 1 TO 11: READ X,Y: COLOR 8: REM  koruna stromu
93Ø FOR I = Ø TO 2: PLOT X - I, Y - 4Ø + 2 * I:
    DRAWTO X - I, Y - 2Ø - 2 * I: NEXT I
95Ø FOR I = -2 TO -1: PLOT X - I,Y - 4Ø - 2 * I:
    DRAWTO X - I, Y - 2Ø + 2 * I: NEXT I
96Ø COLOR 6: REM                         kmen stromu
97Ø PLOT X, Y: DRAWTO X, Y - 21
98Ø COLOR 5: REM                         stín stromu
99Ø PLOT X, Y + 1: DRAWTO X + 7, Y + 4: PLOT X + 8, Y + 3:
    DRAWTO X + 8, Y + 5: DRAWTO X + 9, Y + 6
1ØØØ DRAWTO X + 9, Y + 3: DRAWTO X + 1Ø, Y + 3:
    DRAWTO X + 1Ø, Y + 7
```

```
1Ø1Ø PLOT X + 11, Y + 7: DRAWTO X + 11: Y + 4: DRAWTO
     X + 12, Y + 5: DRAWTO X + 12, Y + 7
1Ø2Ø COLOR 8: REM                          okolí kmene stromu
1Ø3Ø FOR I=1 TO 15: RX = X + INT(RND(Ø)*7) - 3:
     IF RX = X THEN 1Ø3Ø
1Ø4Ø RY - Y + INT (RND(Ø)*8) - 3: PLOT RX, RY:
     NEXT I: NEXT T
11ØØ REM                                   kreslení pěny
111Ø COLOR Ø: REM                          stejná barva jako obloha
112Ø PLOT 57,114: DRAWTO 65,122: PLOT 57,116: DRAWTO 65,124
115Ø PLOT 56,116: DRAWTO 65,125: PLOT 56,117: DRAWTO 65,126
117Ø PLOT 56,118: DRAWTO 65,127: PLOT 56,119: DRAWTO 65,128
119Ø PLOT 55,119: DRAWTO 64,128: PLOT 55,12Ø: DRAWTO 63,128
125Ø REM                                   nastavení zvukových
                                                 kanálů
126Ø FOR I = Ø TO 3: SOUND I,I*5Ø,Ø,8: NEXT I
127Ø GOTO 1ØØ
13ØØ REM                                   podprogram vykreslování
131Ø PLOT X1,Y1:POSITION X2,Y2: XIO 18,#6,Ø,Ø,"S:":RETURN
15ØØ COL=INT(RND(Ø)*4)+1: IF COL=STARTCOL THEN 151Ø
152Ø STARTCOL = COL + CFLAG
153Ø IF STARTCOL = Ø THEN STARTCOL = 4
154Ø IF STARTCOL= 5 THEN STARTCOL = 1
155Ø RETURN
2ØØØ REM                                   data pro rozmístění
                                                 stromů
2Ø1Ø DATA  7,1Ø6,13,96,3Ø,1ØØ,4Ø,112,47,145,
     7,179,15,155,27,164,35,173,6Ø,181,66,174
```

Inicializace (2ØØ ÷ 31Ø): v této části se provádí nastavení všech barvových registrů.

Kreslení trávy a útesu (4ØØ ÷ 49Ø):
Stejně jako při olejomalbě je potřeba i při kreslení na obrazovku udělat nejprve pozadí a potom detaily. K rychlejšímu vykreslení je použit již dříve zmiňovaný podprogram FILL.

Kreslení svahu a řeky (5ØØ ÷ 75Ø):
Program využívá k zdánlivému pohybu řeky (to je vlastně jediný pohyb na celé scéně) čtyř barvových registrů. Řeka protéká nejprve na vrcholu útesu (51Ø ÷ 6ØØ), pak padá vodopádem (61Ø ÷ 72Ø) a nakonec protéká údolím (73Ø ÷ 75Ø). Voda vlastně sestává z proužků, jejichž barva je zvolena generátorem náhodného čísla tak, aby dva proužky vedle sebe neměly stejný odstín (151Ø).

Kreslení stromů (9ØØ ÷ 1Ø4Ø):
Tato část programu kreslí 11 stejných stromů, jejichž umístění je dáno souřadnicemi, zapsanými v řádku 2Ø1Ø jako DATA. Řádky 98Ø ÷ 1Ø1Ø vykreslují stíny stromů tmavě zelenou barvou.

Kreslení pěny (11ØØ ÷ 119Ø):
K ještě něžnějšímu vykreslení celého obrazu je na úpatí vodopádu vykreslena pěna v barvě oblohy.

Nastavení zvukových kanálů (125Ø ÷ 127Ø):
K vytvoření zvukového doprovodu je využito všech čtyř zvukových kanálů. Nastavení není během programu již změněno.

Rotace barev (1ØØ ÷ 13Ø):
Rotace čtyř barvových registrů vytváří iluzi pohybu vody.

9.9 MODIFIKACE PROGRAMU

* Iluzi západu slunce by bylo možno snadno vytvořit postupným zmenšováním jasu ve všech barvových registrech.
* Změnou některých částí programu a barvových registrů by se dala střídat různá roční období.

10. Z V U K (H A R D W A R E)

10.1 ÚVOD

Domácí počítač ATARI dovede dobře generovat zvuk. Má čtyři nezávislé zvukové kanály, každý kanál má vlastní registr výšky tónu a registr určující hlasitost a zabarvení tónu. Zvláštní nastavení dále dovoluje vyrobit filtr typu horní propust, zvolit vstupní taktovací frekvenci čítače, modifikovat poly čítače atd.

10.2 ZVUKOVÉ HARDWAROVÉ REGISTRY

Zvuk je generován mikroprocesorem POKEY a tento obvod rovněž obsluhuje seriovou sběrnici a klávesnici. Inicializaci mikroprocesoru provádí každé připojitelné zařízení (disk, kazetový magnetofon).

Inicializace POKEY v BASICu pro zvuk je SOUND Ø,Ø,Ø,Ø. Přímo přístupný pro strojový jazyk je registr AUDCTL na adrese $D2Ø8_H$ (tj. 53768_H). AUDCTL bude popsán dále. Registry AUDF1 ÷ AUDF4 jsou frekvenční registry jednotlivých kanálů umístěné na adresách: D2ØØ, D2Ø2, D2Ø4 a D2Ø6 hexadecimálně (decimálně 5376Ø, 53762, 53764, 53766). Registry AUDC1 ÷ AUDC4 jsou řídící registry zvuku (určují hlasitost a zabarvení), umístěné na adresách: D2Ø1, D2Ø3, D2Ø5, D2Ø7 hexadecimálně (decimálně 53761, 53763, 53765, 53767).

10.3 FREKVENČNÍ REGISTRY

Každý frekvenční registr obsahuje číslo, které je vlastně dělícím poměrem v čítači frekvence daného kanálu.

10.4 ŘÍDÍCÍ REGISTR ZVUKU

Každý kanál má svůj řídící registr, ve kterém je zapsáno zabarvení zvuku a hlasitost podle následující tabulky.

TAB č. XIV

AUDCn

Bit	7	6	5	4	3	2	1	Ø
Použití	zabarvení			pulsní režim	hlasitost			

10.5 HLASITOST

Nižší čtyři bity registru AUDCn přímo určují hlasitost zvukového kanálu n v rozsahu Ø (bez zvuku) až 15 (nejsilnější hlasitost). Nezapomeňte, že při nastavení všech čtyř zvukových kanálů na nejvyšší hlasitost není výstupní výkon čtyřnásobný, ale pouze dvojnásobný, neboť

$$P(dB) = 1Ø * CLOG\left(\frac{4}{1}\right) \doteq 6\ dB$$

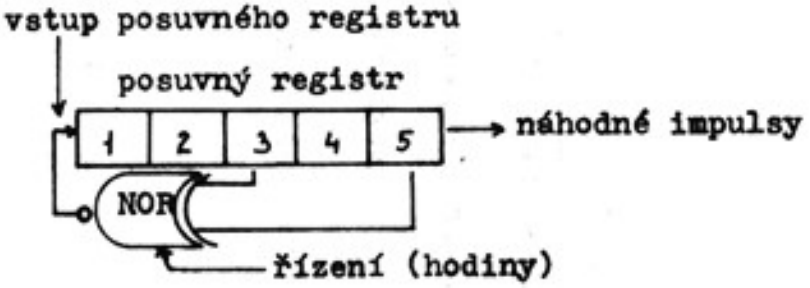
a 6 dB je dvojnásobný přírůstek výkonu.

10.6 ZABARVENÍ

Tři nejvyšší bity registru AUDCn slouží k nastavení zabarvení zvuku zvukového kanálu n. Zabarvení se hodí k nejrůznějším zvukovým efektům, kde by čistý tón nebyl nejvhodnější. Je jednoduše vytvářeno vynecháváním některých částí z čistého obdélníkového průběhu signálu.

Pro lepší pochopení je nutné vysvětlit pojem poly čítače. Je to v podstatě zdroj náhodných impulsů, které modulují čistý obdélníkový průběh. Poly čítač sestává z posuvného registru a řídící logiky. Na obrázku č. 2 představuje řídící logiku řízené hradlo NOR.

Obrázek č. 2



Z funkce obvodu vyplývá, že impulsy budou spíše pseudonáhodného charakteru, sekvence impulsů se budou po čase opakovat. Čím delší posuvný registr bude, tím bude opakování totožných sérií impulsů řidší. Impulsy z poly čítače jsou poté komparovány s čistým signálem a výstup z komparátoru je požadovaný výstupní signál.

Protože impulsy z poly čítače se v pravidelných sériích opakují, bude rovněž výstupní signál cyklický. Takto se dá jednoduše napodobit zvuk motoru, letadla apod.

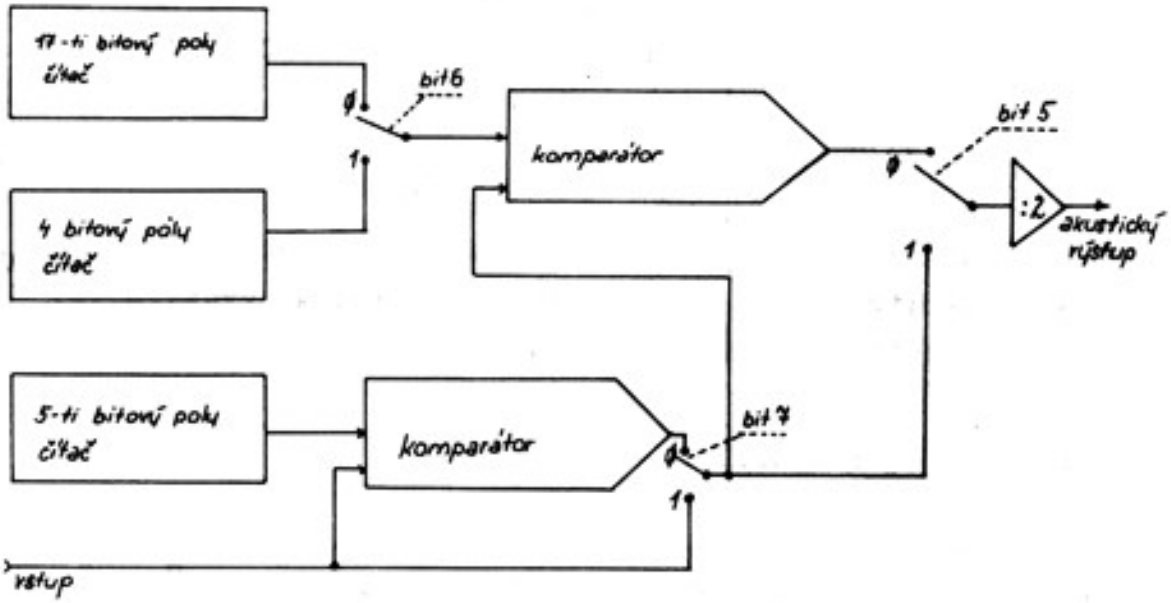
ATARI má k dispozici tři poly čítače s délkou posuvného registru 4, 5 a 17 bitů. 17-ti bitový poly čítač má takovou délku, že výstupní impulsy je možno považovat za náhodné, což lze využít např. ke generování zvuku exploze nebo tzv. bílého šumu.

Každý ze čtyř zvukových kanálů nabízí šest různých kombinací tří poly čítačů viz tabulka č. XV.

TAB č.XV

bit v AUDCn			Zabarvení zvuku na výstupu		
7	6	5	nízká frekvence	střední frekvence	vysoká frekvence
ø	ø	ø	Geigerův počítač	neklidný oheň	rychlé letadlo pára
ø	X	1	kulomet auto na volnoběh	el. motor	výkonný transformátor
ø	1	ø	klidný oheň	auto	auto
1	ø	ø	stavební ruch	rozhlasové rušení	vodopád
1	X	1	čistý tón		
1	1	ø	letadlo žací stroj	holící strojek	

Obrázek č. 3 znázorňuje zapojení poly čítačů a komparátorů v závislosti na bitech 7, 6 a 5 registru AUDCn.



Obr. č. 3

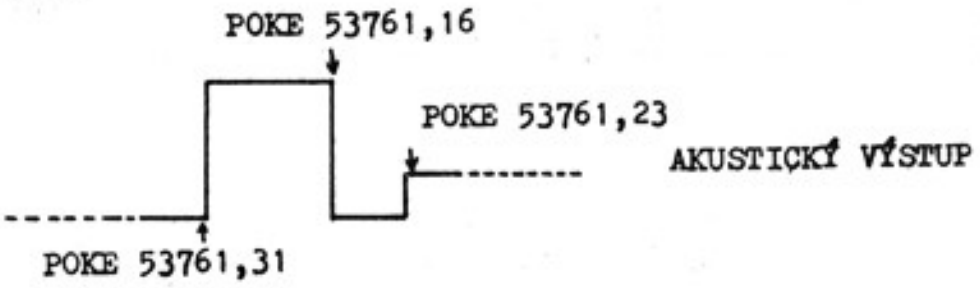
10.7 PULSNÍ REŽIM

Čtvrtý bit v registru AUDCn specifikuje tzv. pulsní režim. Je-li nastavena hodnota hlasitosti určena bity 0 ÷ 3 je posláno přímo na výstup zvukového kanálu. Ale jen jeden impuls! Zkuste udělat v přímém modu:

POKE 53761, 31: POKE 53761, 16: POKE 53761, 23

V reproduktoru televizního přijímače se objeví krátký puls, neboť první příkaz vyslal impuls s maximální amplitudou a druhý příkaz vyslal impuls s nulovou amplitudou, třetí příkaz nakonec vyslal impuls s poloviční amplitudou. Jev ukazuje obrázek č. 4.

Obr. 4



Pulsní režim není dobře využitelný v BASICu. V jedenácté kapitole bude popsán strojový program využívající 4. bit registru AUDCn.

10.8 REGISTR AUDCTL

Každému bitu registru AUDCTL na adrese $D2\emptyset8_H$, (tj. 53768_D) je přidělena speciální funkce, jejichž popis naleznete v tabulce č. XVI.

TAB č. XVI

bit AUDCTL	Vykonávání funkce, je-li bit nastaven
ø	přepínač časové základny 64 KHz, 15 KHz (pro bit ø = 1)
1	horní propust v kanálu 2. Jsou propouštěny signály s vyšší frekvencí, než jaká je nastavena v kanálu 4.
2	horní propust v kanálu 1. Jsou propouštěny signály s vyšší frekvencí, než jaká je nastavena v kanálu 3
3	kanály 3 a 4 jsou spojeny v jeden 16-ti bitový čítač
4	kanály 1 a 2 jsou spojeny v jeden 16-ti bitový čítač
5	na vstup kanálu 3 je připojeno 1,79 MHz
6	na vstup kanálu 1 je připojeno 1,79 MHz
7	změna 17-ti bitového poly čítače v 9-ti bitový

10.9 ČASOVÁ ZÁKLADNA

Aktivita mikroprocesoru má cyklický charakter. Procesor vybírá instrukci z paměti, vykonává ji, vybírá další instrukci. Taková posloupnost úkonů vyžaduje přesné řízení v čase - tzv. časování s pomocí nezávislého oscilátoru - časové základny, kterou výhodně využívají i zvukové kanály. Frekvence časové základny v ATARI je 1,79 MHz. Z této základní frekvence je odvozeno ještě 64 KHz a 15 KHz. Výběr odvozených kmitočtů se provádí bitem Ø registru AUDCTL. Je-li bit Ø = Ø, je na vstupy všech kanálů připojena frekvence 64 KHz, tzn. že výstupní frekvence se pohybuje v rozsahu 25Ø ÷ 32 ØØØ Hz a je-li bit Ø = 1, výstupní frekvence se pohybuje v rozsahu 58 ÷ 7 5ØØ Hz. Nastavením bitu 5 nebo 6 je možné získat na výstupech kanálů 3 nebo 1 frekvenční rozsah 7Ø19 ÷ max.slyšitelná frekvence ve velmi jemném odstupňování.

10.10 16-TI BITOVÝ ČÍTAČ

Umožňuje nastavit frekvenci na výstupu kanálu 2 nebo 4 ve velmi širokém rozsahu s jemným odstupňováním nastavením bitu 4 nebo 3 registru AUDCTL.

10.11 HORNÍ PROPUST

Pomocí horní propustě nastavitelné bitem 1 nebo 2 pro kanály 2,4 nebo 1,3 lze vytvářet různé zajímavé zvukové efekty. Zkuste krátký program č.20.

Program č. 20

```
1Ø SOUND Ø,Ø,Ø,Ø
2Ø POKE 53768,4
3Ø POKE 53761,168:POKE 53765,168
4Ø POKE 5376Ø,254:POKE 53764,127
```

Elektronik nechť se při experimentování s horní propustí ATARI nediví, že výsledný efekt není takový, na jaký je zvyklý při práci se skutečnou elektrickou horní propustí.

10.12 9-TI BITOVÝ POLY ČÍTAČ

Nastavením 7. bitu registru AUDCTL se 17-ti bitový poly čítač změní v 9-ti bitový. Impulsy, které bylo možné na výstupu 17-ti bitového poly čítače považovat za náhodné se změní v pravidelný sled impulsů. Vložte nejprve

```
SOUND Ø,8Ø,8,8
```

a pak

```
POKE 53768,128
```

První výraz aktivuje 17-ti bitový poly čítač, druhý výraz nastavuje 9-ti bitový poly čítač.

11. ZVUK (SOFTWARE)

11.1 ÚVOD

Generování zvuku může být provedeno staticky nebo dynamicky. Druhý způsob je sice těžší, zato však skýtá nepřeberné možnosti. Počítač při něm mění během programu nastavení zvukových dat.

Např: SOUND Ø,128,8,8

generuje zvuk staticky, zatímco:

```
FOR X = Ø TO 255
SOUND Ø,X,8,8
NEXT X
```

generuje dynamický zvuk.

11.2 ZVUK STATICKY

Statický zvuk je svým působením obvykle omezen na krátká pípnutí, bzučení apod. Existují však i vyjímky. Jeden příklad byl už uveden v předchozí kapitole, druhý, využívající interference dvou blízkých kmitočtů vypadá takto:

```
SOUND Ø,255,1Ø,8
SOUND 1,254,1Ø,8
```

Čím je frekvence obou kanálů blíže, tím je "pohupování" tónu pomalejší a naopak.

11.3 ZVUK DYNAMICKY

Větší zvukové celky (doprovodná hudba) vyžadují dynamickou (častou i rychlou) změnu zvukových dat.

Existují tři možnosti obsluhy zvukových registrů:

* v BASICu
* pomocí přerušovacího režimu 50 Hz s využitím strojového jazyka
* využitím strojového jazyka

11.4 ZVUK BASICem

V BASICu je programátor omezen pouze na příkazy SOUND a POKE (lze jím přímo nastavovat registry AUDFn, AUDCn a AUDCTL). Pozor, příkaz SOUND ruší nastavené bity v registru AUDCTL.

Použití BASICu ke generování rychle se měnících zvuků je limitováno rychlostí, s jakou může BASIC pracovat. Při práci s více kanály vyvstanou další problémy při nutnosti změny zvukových dat najednou. Pomůže jen strojový jazyk, jak to ukazuje následující program č. 21.

Program č. 21

```
10 SOUND 0,0,0,0:DIM SIMUL $(16)
20 RESTORE 9999:X=1  :REM  načtení strojového programu
25 READ Q:IF Q<>-1 THEN STIMUL$(X)=CHR$ (Q):X=X+1:GOTO 25
27 RESTORE 100:REM  čtení zvukových dat
30 READ F1,C1,F2,C2,F3,C3,F4,C4
40 IF F1=-1 THEN 27
50 X=USR (ADR(SIMUL $ ),F1,C1,F2,C2,F3,C3,F4,C4)
55 FOR X=0 TO 200: NEXT X
60 GOTO 30
90 REM  zvuková data
```

```
1ØØ DATA 182,168,Ø,Ø,Ø,Ø,Ø,Ø
11Ø DATA 162,168,182,166,Ø,Ø,Ø,Ø
12Ø DATA 144,168,162,166,35,166,Ø,Ø
13Ø DATA 128,168,144,166,40,166,35,166
14Ø DATA 121,168,128,166,45,166,4Ø,166
15Ø DATA 128,168,121,166,47,166,45,166
16Ø DATA 96,168,1Ø8,166,53,166,47,166
17Ø DATA 91,168,96,166,6Ø,166,53,166
999 DATA -1,Ø,Ø,Ø,Ø,Ø,Ø,Ø
999Ø REM   data pro strojový program
9999 DATA 1Ø4,133,2Ø3,162,Ø,1Ø4,1Ø4,157,Ø,21Ø,232,228
     2Ø3,2Ø8,246,96,-1
```

Strojový program v řádku 5Ø umožňuje rychle vkládat zvuková data do registrů, čímž nevznikají žádné vedlejší rušivé zvuky.

11.5 ZVUK ZA POMOCÍ PŘERUŠENÍ 50Hz

Tato programovací technika je universální a využitelná ve velkém rozsahu aplikací.

Každých 2Ø ms je práce počítače hardwarovým vybavením přerušena (interrupt), počítač přechodně přejde z hlavního programu do strojového programu, jehož počátek je určen programově. Po skončení strojového programu se počítač vrací přes krátkou obslužnou rutinu zpět do hlavního programu, který může být zapsán jak v BASICu, tak i ve strojovém jazyku. Pro tvorbu větších zvukových ploch vzniká díky této programové technice ideální situace, neboť jednotlivé zvukové registry lze modifikovat v přesném časovém sledu až 5Øx za sekundu - což je pro většinu aplikací dostačující - a přitom není přerušení práce

hlavního programu postřehnutelné. V následujících bodech bude popsáno, co je potřeba provést, aby počítač mohl zahrát třeba písničku "Až ráno ..." v celé kráse.

1. Strojový program umístit do paměti RAM (např. do 6. stránky paměti, která je pro takové kreace přímo vyhrazena)
2. Poslední instrukce strojového programu musí být JMP $E462. Na této adrese je krátký podprogram se symbolickým názvem XITVBL (eXIT Vertical Blank interrupt), přes který se počítač vrací zpět do přerušeného programu
3. Do registru X zapsat vyšší byte adresy strojového programu (v našem případě Ø6)
4. Do registru Y zapsat vyšší byte adresy strojového programu (v našem případě ØØ)
5. Do akumulátoru zapsat Ø7
6. Provést JSR $E45C. Na této adrese je krátký podprogram se symbolickým názvem SETVBV (SET Vertical Blanc Vector), který provede zápis adresy strojového programu do ukazovátka strojového programu. Ukazovátko je na adrese 224 a 225_H (tj. 548 a 549_D). Jinak je zde běžně adresa XITVBL.

Program pak bude pracovat následovně:

1. Žádost o přerušení od hardware
2. Počítač skáče do strojového programu, jehož počáteční adresa je zapsána v ukazovátku na adrese 224 a 225_H
3. Vykonání strojového programu
4. Na konci strojového programu je skok na XITVBL
5. XITVBL vrací řízení zpět programu, vykonávaného před přerušením

Pokud si se sestavením vlastního programu nebudete vědět rady, nahlédněte do příručky ATARI Program Exchange.

Upozornění: Mikroprocesor 65Ø2 vykonává všechny aritmetické operace pro BASIC dekadicky. Tzn. že přijde-li přerušení v době vykonávání aritmetických operací, budou i všechny aritmetické operace během přerušení provedeny dekadicky, nikoliv binárně!

11.6 GENEROVÁNÍ ZVUKU POMOCÍ STROJOVÉHO JAZYKA

Přímé řízení zvukových registrů strojovým programem otevírá nové možnosti generování zvuků.

Program se podobně jako v předchozím případě zapíše do paměti RAM. Spuštěný program se však nyní bude zabývat po většinu času jen zvukovými registry. Tak je možné věrně generovat zvuky nejrůznějších nástrojů.

11.7 GENEROVÁNÍ ZVUKU PŘÍMÝM OVLÁDÁNÍM VÝSTUPNÍHO KANÁLU.

Jak již bylo v 10. kapitole řečeno, výstupní kanál zvuku lze také ovládat přímým nastavením bitu D4 a úrovně (bity DØ ÷ D3) v registrech AUDCn. BASIC je pro tuto práci příliš pomalý, takže nezbývá, než využít opět strojový jazyk.

Následující strojový program ukazuje využití přímého řízení výstupu zvukových kanálů.

```
1Ø ;VONLY     volume only AUDC1÷4 bit test routine
2Ø AUDCTL = $ D2Ø8
3Ø AUDF1 = $ D2ØØ
4Ø AUDC1 = $ D2Ø1
5Ø SKCTL = $ D2ØF
6Ø                        *= $ BØ
7Ø TEMPO                     .BYTE1
8Ø MSC                       .BYTEØ
9Ø
1ØØ                       *= $ 4ØØØ
11Ø                       LDA# Ø
12Ø                       STA AUDCTL
13Ø                       LDA#3
14Ø                       STA SKCTL
15Ø                       LDX# Ø
16Ø                       LDA #Ø
17Ø                       STA$ D4ØE; zakázáno přerušení systémem
                                                          50Hz
18Ø                       STA$ D2ØE; zakázáno nemaskované přeruš.
19Ø                       STA$ D4ØØ; zakázáno zobrazování
2ØØ LOO                   LDA DTAB,X
21Ø                       STA MSC
22Ø LØ                    LDA VTAB,X
23Ø                       LDY TEMPO
24Ø                       STA AUDC1
25Ø L1                    DEY
26Ø                       BNE L1
27Ø ;                     dekrementace významnějšího bytu čítače
28Ø                       DEC MSC
29Ø                       BNE LØ
3ØØ ;                     nový tón
31Ø                       INX
```

```
32Ø                     CPX NC
33Ø                     BNE LOO
34Ø                     LDX #Ø
35Ø                     BEQ LOO
36Ø NC                  BYTE 28; čítač not
37Ø ;                   tabulka velikosti amplitudy
38Ø VTAB
39Ø                     .BYTE 24,25,26,27,28,3Ø,31
4ØØ                     .BYTE 3Ø,29,28,27,26,25,24
41Ø                     .BYTE 23,22,21,2Ø,19,18,17
42Ø                     .BYTE 18,19,2Ø,21,22,23
43Ø ;                   tabulka trvání amplitudy
44Ø DTAB
45Ø                     .BYTE 1,1,1,2,2,2,3,6
46Ø                     .BYTE 3,2,2,2,1,1,1
47Ø                     .BYTE 1,1,2,2,2,3,6
48Ø                     .BYTE 3,2,2,2,1,1
```

Program generuje tón o kmitočtu asi 1Ø KHz. Řádky 17Ø ÷ 19Ø zakazují jakékoliv přerušení. Tón bude jinak nečistý (brum od 5ØHz přerušení, zpomalení obrazovým procesorem ANTIC, který si půjčuje sběrnice 65Ø2).

L I T E R A T U R A

Kapitoly 1 ÷ 7 Chris Crawford
8 Kathelen Pitta + Lane Winner
9 David Fox + Mitchell Waite
10, 11 Bob Fraser

Vyšlo v časopisech BYTE 1981 - 1982.

O B S A H

ABC o počítačích ATARI 600XL,800XL

Autor:	Pavel Dočekal
Tech.redakce:	Oldřich Burger
Vydal:	Klub mikroelektroniky ZO SSM SOUŽ Jimramov
Ve spolupráci:	Klub mikroelektroniky ZO Svazarmu TLMAČE
Tisk:	Tiskárna SSM, Ostrava 1
Náklad:	500 ks
Rok:	1986

Příručka obsahuje základní popis počítačů ATARI 600XL,800XL.Formou názorného vysvětlení na příkladech pomahá proniknout do systému počítače a některých zvláštností při tvorbě programů.Je určena zejména těm začínajícím uživatelům HC ATARI, kteří mají alespoň základní vědomosti o vnitřní struktuře a funkci počítačů jiných typů. Obsažené informace nemají za cíl suplovat základní návod k použití počítače (Uživatelskou příručku), ani popis programovacího jazyku ATARI BASIC.

Předmluva

Kniha,"ABC o počítačích ATARI 600XL,800XL"(130XE),je volným přepisem zahraničního titulu ATARI TUTORIAL. Protože autor brožury, P. DOČEKAL, ovládá angličtinu pouze pasivně, vznikl shrnutím pochopených pasáží knihy Chrise Crawforda knižní originál.Při úvahách o názvu brožury, která vznikla samostatným tvůrčím zpracováním podkladů z několika zahraničních pramenů uvedených v odkazu, byla zvažována opodstatněnost zachování původního názvu knihy. S ohledem na existující konvence producentů technické literatury jsme nakonec v nejlepším přesvědčení nazvali tuto brožuru "ABC o počítačích ATARI 600XL,800XL" (130XE).

Ačkoli nelze vyloučit sporadický výskyt drobných chyb, jejichž úplné eliminování z originálu předlohy pro reprografické práce je téměř nemožné, lze zaručit solidnost obsahu alespoň v tom směru, že všechna lehce ověřitelná fakta a důsledně všechny v příkladu využité cvičné programy, byly autorem ověřeny v praxi.Tato záruka je velmi významná zejména z pohledu tápajícího začátečníka, jenž namnoze, spoléhajíc na solidnost autorů kompilačních titulů, pracně ověřuje zdroje a příčiny chyb, které často leží mimo něj. Z tohoto úhlu pohledu je předložená publikace seriosním informačním zdrojem, který nalezne odpovídající ocenění zejména mezi novými uživateli počítačů ATARI.

Veříme, že i v těch případech, kdy by se našel důvod pro věcnou a opodstatněnou kritiku, naleznete dostatek tolerance k naší práci. Bez všech nadsázek můžeme s dobrým pocitem konstatovat, že publikace, která se Vám dostává do rukou, vznikla z iniciativy a nadšení úzkého tvůrčího kolektivu, který se do zpracování rukopisu a zajištění jeho rozmnožení pustil s mlhavými představami o společenském, případně ekonomickém zhodnocení předpokládané práce.

Přáli bychom si, aby Vám tato brožura, stejně tak jako další připravované tituly knihovničky klubu mikroelektroniky SSM, usnadnily rychlé načerpání potřebných informací o počítačích ATARI.