GRAFIKA
V ZÁKLADNÍM MÓDU COMMODORE 64

Patrick Zandl

programování grafiky

OBSAH

Úvodní informace

Grafické možnosti počítače Commodore 64 se zakládají na video-interface čipu 6567 (VIC). Tento čip umožňuje různá grafická zobrazení, včetně textového zobrazení 40 na 25 znaků, vysokorozlišitelné grafiky 320 na 200 bodů (HIRES), ale i spritů - malých objektů, využívaných zejména ve hrách. Navíc je možné tyto módy míchat. Lze tedy vytvořit horní polovinu obrazovky ve vysokorozlišitelné grafice a spodní část v textovém módu. Sprity je možné zobrazit v každém módu. O nich však až v samostatné knize.

Nejdříve se seznámíme s ostatními možnostmi grafiky. S VIC čipem jsou možné tyto módy:

A) Znakový mód
   1. standardní znaky
      a) ROM - znaky
      b) RAM - programovatelné znaky
   2. vícebarevné znaky
      a) ROM - znaky
      b) RAM - programovatelné znaky
   3. rozšířená barva pozadí
      a) ROM - znaky
      b) RAM - programovatelné znaky

B) Mód bitové mapy
   1. standardní bitová mapa
   2. vícebarevná bitová mapa

C) Sprity
   1. standardní sprity
   2. vícebarevné sprity

Poloha grafických znaků v paměti

Nejdříve základní informace: obrazovka má 1000 pozic. Normálně začíná obrazová paměť na adrese 1024 ($0400 HEX) a končí na adrese 2023. Do každé této adresy může být zapsáno 8 bitů, které dohromady tvoří celé číslo mezi 0 a 255. Obrazová paměť tedy odpovídá skupině o 1000 adresách, které se nazývají barevnou pamětí RAM, neboli color-RAM (cRAM). Tato cRAM začíná od místa 55296 ($D800 HEX) a pokračuje až do adresy 56295. Každá tato adresa v cRAM dostane 4 bity, které mohou být libovolným celým číslem od 0 do 15. Proto má C64 16 barev a může s nimi velmi dobře pracovat.

V jednom okamžiku může být zobrazeno 256 různých znaků. Při normálním obrazovém módu dostane každá z 1000 adres číslo, které VIC čipu řekne, který znak má na tomto místě obrazovky ukázat. Pro obsluhu grafických módů má VIC čip 47 řídících registrů. Mnoho grafických funkcí lze tedy řídit zapsáním správné hodnoty do správné adresy. Tyto řídící registry se nalézají na adresách od 53248 ($D000 HEX) do 53294 ($D02E HEX).

Volba VIDEO banky

Čip VIC může najednou obsáhnout jeden paměťový úsek o 16 KB. Třebaže C64 má 64 KB paměti, může VIC přehlédnout celou paměť. Jsou totiž 4 různé banky o kapacitě po 16 KB. Vy musíte žádat, ve které z nich bude VIC pracovat. Pokud je budete plynule přepínat, přehlédne čip celou oblast paměti. Bit, který volí banku, se nalézá v complex-interface adaptér chip#2 (CIA#2) 6526. BASIC příkazy PEEK a POKE či jejich ekvivalenty ve strojovém kódu bude banka volena nultým a prvním bitem z PORTU A CIA#2 (adresa 56576 či $DD00 HEX). Ke změně paměťové oblasti musí být vyslány tyto dva bity jako výstupní. Toho lze dosáhnout následujícně:

POKE 56578, PEEK (56578)OR3:REM bity 0 a 1 jako výstupní
POKE 56576, (PEEK(56576)AND252)OR A:REM volba videobanky

kde "A" musí mít následující hodnotu:

Hodn. A	Bity	Banka	Počátek	Místo VIC-II čipu:
0	00	3	49152	$C000-$FFFF
1	01	2	32768	$8000-$BFFF
2	10	1	16384	$4000-$7FFF
3	11	0	0	$0000-$3FFF

UPOZORNĚNÍ: Znaková sada není pro C64 v bance 1 a 3 volně přístupná (nahlédni do dílu "Znaková paměť").

Tento výpis 16 KB bank hraje svojí roli při každém použití čipu VIC. Musíte vědět, se kterou oblastí paměti VIC právě pracuje, odkud dostává data, kde se nalézá aktuální obrazovka nebo odkud přicházejí informace o podobě spritu. Po zapnutí C64 je automaticky aktivována banka 0 ($0000-$3FFF), odkud přicházejí informace.

Obrazová paměť

Díky POKE v kontrolním registru na adrese 53272 ($D018) mohou být měněny adresy obrazové paměti. Tento registr je také používán k řízení změny každé znakové sady. Dejte proto obzvláště pozor na tuto část řídícího registru a svévolně ji neměňte. Horní 4 bity řídí místo obrazové paměti. Ke změně umístění obrazovky je optimální následující instrukce:

POKE 53272, (PEEK(53272)AND 15)OR A

kde "A" nabývá následujících hodnot:

A	BIT	DEC	HEX
0	0000xxxx	0	$0000
16	0001xxxx	1024	$0400
32	0010xxxx	2048	$0800
48	0011xxxx	3072	$0C00
64	0100xxxx	4096	$1000
80	0101xxxx	5120	$1400
96	0110xxxx	6144	$1800
112	0111xxxx	7168	$1C00
128	1000xxxx	8192	$2000
144	1001xxxx	9216	$2400
160	1010xxxx	10240	$2800
176	1011xxxx	11264	$2C00
192	1100xxxx	12288	$3000
208	1101xxxx	13312	$3400
224	1110xxxx	14336	$3800
240	1111xxxx	15360	$3C00

Mějte na paměti, že počáteční adresy každé banky čipu VIC budou muset být sečteny.

Barevná paměť (cRAM)

Barevná paměť nemůže být posunuta. Nachází se na místech 55296 ($D800) až 56295 ($DBE7). Obrazová i barevná paměť budou v různých módech různě použity. Obraz v jednom módu tedy bude v jiném módu vypadat rozdílně.

Znaková paměť

Pro programování grafiky je podstatné, odkud čip skutečně dostává informace o podobě znaku. Obrysy písmen dostává čip z "character generator ROM". V této paměti jsou nahrány vzory,

představující různá písmena, číslice, interpunkční znaménka a ostatní znaky klávesnice.

Jednou z předností C64 je používání znaků definovaných do RAM. Vzor v RAM můžete libovolně měnit, takže dostanete takřka neohraničený prostor symbolů pro hry, výpočty a další. Normální znaková sada obsahuje 256 znaků.

Každý znak se skládá z 8 bytů, tedy celá znaková sada zabere 256*8=2 KB paměti. Protože VIC obsáhne najednou 16 KB paměti, je zde 8 různých možností pro plnou znakovou sadu. Znaková sada musí začínat na jednom z osmi startovacích míst. Poloha znakové paměti je čipem VIC kontrolována třemi bity na adrese 53272 ($D018 HEX). Bity 3, 2, 1 ukazují místo, kde se nalézají znakové sady o 2 KB. Bit nula zde nehraje roli. Nezapomeňte, že tyto registry jsou rovnocenné, tomu má odpovídat poloha obrazové paměti. Ke změně polohy znakové paměti použijte následující příkaz BASICU:

POKE 53272, (PEEK (53272)AND240)OR A

přitom "A" nabývá následujících hodnot:

Hodn. A	Bity	DEC	HEX
0	xxxx000x	0	$0000-$07FF
2	xxxx001x	2048	$0800-$0FFF
4	xxxx010x	4098	$1000-$17FF ROM-IMAGE
6	xxxx011x	6144	$1800-$1FFF ROM IMAGE
8	xxxx100x	8192	$2000-$27FF
10	xxxx101x	10240	$2800-$2FFF
12	xxxx110x	12288	$3000-$37FF
14	xxxx111x	14336	$3800-$3FFF

Nezapomeňte, že na této adrese se sčítá počátek znakové sady a obrazové paměti.

ROM-IMAGE v tabulce se vztahuje k character generator ROM. Vyskytuje se v RAM na horních místech banky 0. Mimo to se vyskytuje v odpovídající RAM na místech 36864-40959 ($9000 - $9FFF) v bance 2. Jelikož VIC může současně obsáhnout jen 16 KB paměti, nalézá se ROM - znakový vzor v sadě, v níž je přímo obsazen. Proto je systém vyvinut tak, že VIC odtud vyjde, najde si ROM - znaky na adresách 4096-8191 ($1000-$1FFF), jsou-li vaše data v bance 0, a na adresách 36864-40959 ($9000-$9FFF), jsou-li v bance 2. ROM znaky se ve skutečnosti nalézají na adresách 53248-57343 ($D000-$DFFF).

Toto "zrcadlení" se odvolává jen na data znaku, která vidí VIC. RAM na těchto adresách může být použita pro programy jako jiná RAM paměť. Zdržuje-li vás toto ROM-zrcadlení vlastní grafiky, zvolte volebním bitem banku bez obsazení (banka 1 a 3). ROM vzor se tam neobjeví.

Pozorného čtenáře po nahlédnutí do tabulky zajisté napadne, jak je možné, aby znaky z ROM zaujímaly místo řídících registrů pro VIC. Je to možné proto, že toto místo není požadováno současně pro obojí, ale postupně.

Potřeboval-li čip obsah znakových dat, tak bude ROM zapnuta. V 16 KB paměťové banky, které VIC obsáhne, vzniká odpovídající zrcadlení. Mimo to bude tento obvod požadován vstupně -výstupními registry a znaková ROM může být dosazena pouze čipem VIC.

blok	DEC	HEX	zrcadlení	obsah
0	53248	D000-D1FF	1000-11FF	velká písmena
	53760	D200-D3FF	1200-13FF	grafické znaky
	54272	D400-D5FF	1400-15FF	velké inv. znaky
	54784	D600-D7FF	1600-17FF	graf. inv. znaky
1	55296	D800-D9FF	1800-19FF	malá písmena
	55808	DA00-DBFF	1A00-1BFF	velká p., graf. z.
	56320	DC00-DDFF	1C00-1DFF	malá inv. písmena
	56832	DE00-DFFF	1E00-1FFF	velká inv. písmena

Může se však stát, že potřebujete znakovou ROM. To se stane nejvíce tehdy, když chcete používat programovatelné znaky a kopie části znakové ROM potřebujete pro definici znaku. V tomto případě musíte vstupně/výstupní registry vypnout a zapnout znakovou ROM. Potom můžete začít kopírovat. Pak obnovte zapnutí vstupně/výstupních registrů. Během kopírování, když jsou registry vypnuty, nesmí dojít k žádnému přerušení. Pro přerušení jsou totiž I/O registry potřeba a vy jste je právě vypojil.

Pokud na to zapomenete a provedete přerušení, budou se dít věci neočekávané a nepředložené. Během kopírování nesmí být rovněž čteno z klávesnice. Pro odpojení klávesnice a dalších přerušení, která jsou na C64 možná, použijte následující POKE:

```
POKE 56334,PEEK (56334)AND254 :REM přerušení vypnuto
```

Máte-li obsah znakové ROM přenesen a připraven pro další programování, bude tímto POKE klávesnice opět zapojena

```
POKE 56334, PEEK (56334) OR1:REM přerušení zapnuto
```

Následující POKE vypne I/O registry a zapne znakovou ROM:

```
POKE 1,PEEK (1)AND251
```

Znaková ROM se nyní nalézá na adresách od 53248 do 57343 ($D000-$DFFF). Pro zpětné zapojení normální funkce I/O na adrese $D000 použijte tento POKE:

```
POKE 1,PEEK(1)OR4
```

Standardní znakový mód

Při zapnutí C64 se nalézáte ve standardním znakovém módu. Je to mód, ve kterém normálně programujete.

Znaky mohou být přeloženy z ROM do RAM. Normální písmo bude nadále uchováno v ROM. Potřebujete-li pro program speciální znaky, musíte si znakový vzor definovat do RAM a čipu VIC sdělit, že má informace o znacích brát z RAM a nikoliv z ROM (viz dále). Informace o znacích zobrazovaných na obrazovce barevně obsahuje VIC v obrazové paměti s určením znakových kódů pro toto místo

na obrazovce. Současně obsahuje v barevné RAM informace o barvě ukazovaných znaků. Znakové kódy budou čipem VIC přesunuty s vaším vzorem na startovací adresy 8 bytových vět definujících znaky. Tyto věty se nalézají ve znakové paměti.

Přenesení není příliš komplikované, k vypočtení náležité adresy budou kombinovány některé další body. Nejdříve bude použit kód znaku v obrazové paměti násobený osmi. Potom bude sečten začátek znakové paměti - nahlédněte do oddílu "Znaková paměť". Teď bude brán v potaz bit banky násobený základní adresou (nahlédněte do odstavce "Volba videobanky").

Celý vzorec vypadá takto:

Adresa znaku=obraz. kód*8 +(znak. skupina *2048)+(banka*16384)

Definice znaku

Každý znak je zobrazován v matici 8*8 bodů. V ní mohou jednotlivé body svítit nebo zůstat pohaslé, a tak dohromady vytvářet podobu znaku. U C64 jsou podoby znaků ve znakovém generátoru ROM. Každý znak je věta o 8 bytech. Každý byte znamená jeden řádek bodové matice o 8 bitech. Každý bit znamená bod, který buď svítí (1), nebo nesvítí (0). Znaková sada začíná v ROM od adresy 53248 (při vypnutém I/O registru). Prvních 8 bytů od adresy 53248 ($D000) do 53255 ($D007) obdrželo písmeno $, kterého znakový kód v obrazové paměti je 0. Dalších 8 bytů od adresy 53256 do 54263 má písmeno A.

obsazení	binárně	PEEK
oooxxooo	00011000	24
ooxxxxoo	00111100	60
oxxooxxo	01100110	102
oxxxxxxo	01111110	126
oxxooxxo	01100110	102
oxxooxxo	01100110	102
oooooooo	00000000	0

Každá kompletní znaková sada zabírá paměťovou kapacitu 2 KB (2048 bytů). Celkem je to 256 znaků, přičemž každý znak potřebuje 8 bytů. V ROM jsou celkem dvě znakové sady - první tvoří velká písmena a grafické symboly, druhá pak velká a malá písmena. Obě sady dohromady spotřebují 4 KB paměti ROM.

Programovatelné znaky

To, že jsou znaky nahrány v ROM, svádí k myšlence, že nelze znakovou sadu volně naprogramovat. Ovšem paměťové místo, které VIC oznamuje, odkud má brát znaky, je přece programovatelným registrem. Ten může být libovolně pozměněn tak, aby ukazoval na vaši znakovou sadu. Má-li se vaše znaková sada nalézat v RAM, pak musíte dodržovat dva velice důležité body:

1. Má-li čip VIC vykázáno místo, kde má brát informace o podobě znaku, pak není možné brát znaky z ROM. Proto je dobré znaky z ROM překopírovat do RAM, kde je pak můžete používat v programu. Můžete si zvolit libovolné znaky, nepotřebujete dbát na jejich pořadí v paměti.
2. Vaše znaková sada používá stejnou paměť jako programy BASICU. Proto je chod programu zabírajícího okolo 38 KB velmi problematický, má-li fungovat s definovanou znakovou sadou. POZOR: Dávejte tedy pozor, aby nebyla vaše znaková sada programem přemazána.

Dvě adresy C64 nesmíte označit jako začátek znakové sady:
adresu 0 a adresu 2048!!!

První adresa nesmí být použita proto, že systém na stránku 0 (Page 0) uschovává důležitá data.

Adresa 2048 je začátek vašeho programu v BASICU.

Pro vaše znakové sady zbývá ještě 6 volných pozic pro začátek sady. Nejlepší je zvolit jako počátek adresu 12288 ($3000

HEX). To provedete, POKE spodní 4 bity adresy 53272 číslem 12. Proto vyzkoušejte následující příkaz:

POKE 53272,(PEEK(53272)AND240)+12

Najednou se všechny znaky na obrazovce nesmyslně změnily. Vy totiž nemáte žádnou znakovou sadu na adrese od 12288 ... pouze neuspořádané bity. Zmáčkněte RUN/STOP+RESTORE, čímž obnovíte znakovou sadu v ROM.

Nyní chceme zvolit nějaké grafické symboly. Abyste ochránili svoji znakovou sadu před přepsáním nějakým programem, měli byste zredukovat paměť určenou pro BASIC. Pochopitelně paměť vašeho počítače zůstane nezměněna ... vy pouze poručíte BASICU, aby určitou část paměti nepoužíval.

Zkuste následující:

PRINT FRE(0)-(SGN(FRE(0))(0)*65535

Číslo ukazuje nevyužitou paměť počítače.

Zadejte nyní následující:

POKE 52,48:POKE 56,48:CLR

a nyní opět:

PRINT FRE(0)-(SGN(FRE(0))(0)*65535

Vidíte nyní na obrazovce jiné číslo? BASIC má nyní méně volné paměti. V tomto blokovaném úseku paměti můžete definovat svoji znakovou sadu. Pro začátek uložíme data od adresy 12288 ($3000). Pokusně tedy převedeme do RAM znakový vzor z ROM. Dohromady to bude 64 znaků, které tento program přenese:

```
 5  PRINT CHR$(142):REM velké znaky
10  POKE 52,48:POKE56,48:CLR:REM rezervace paměti
20  POKE 56334,PEEK(56334)AND254:REM blokuje přerušení
30  POKE 1,PEEK(1)OR4:REM přepnutí znaků
40  FOR I=0 TO 511:POKE I+12288,PEEK(I+53248):NEXT
50  POKE 1,PEEK(1)OR4:REM přepnutí I/O
60  POKE 56334,PEEK(56334)OR1:REM odblokování přerušení
70  END
```

Nyní zadejte pomocí POKE na adresu 53272 hodnotu

(PEEK(53272)AND 240)+12.

Nic se nestalo, že? Ale tak je to správné. C64 dostává znakové informace z RAM a nikoliv z ROM. A ty jsou přesně okopírovány, proto není vidět žádný rozdíl ... zatím ještě ne.

Znaky nyní můžete lehce měnit. Smažte obrazovku a stiskněte klávesu $. Nyní zadejte následující příkazy:

FOR I=12288 TO 12288+7:POKE I, 255-PEEK(I):NEXT

Nyní jste změnili znak $ na inverzní. Takto můžete měnit i další znaky. Tabulka obrazových kódů vám ukáže, kde jsou v RAM jednotlivé znaky. Mějte na mysli, že k uchování znaku je potřeba 8 bytů. Zde je několik příkladů:

Znak	Obrazový kód	Nynější start. adresa
$	0	12288
A	1	12296
!	32	12552
>	62	12784

Nezapomeňte ovšem, že jste přenesli pouze prvních 64 znaků. Budete-li si brát ještě jiné znaky, pak berte ohled na toto:

Co to udělá, když chcete znak 154 jako obrácené Z? Můžete tohoto dosáhnout, když Z obrátíte, nebo můžete sadu obrácených znaků zkopírovat z ROM nebo jednoduše znak z ROM překopírovat a nahradit jím jiný nedůležitý znak v RAM. Předpokládejme tedy, že znak > již nebudeme potřebovat. Tento znak má být tedy zaměněn za negativní Z. Zadejte tedy následující:

FOR I=1 TO 7:POKE 12784+I,255-PEEK(I+12496):NEXT

Zadejte nyní >. Zobrazí se jako obrácené Z. Dokud nestisknete RUN/STOP+RESTORE, bude tento znak takto zobrazován. Tato změna je ovšem pouze na obrazovce. V paměti počítače má tento znak nadále svoji původní funkci. Tak ho lze i v programu použít (např. v aritm. výrazech).

Zkuste nyní příkaz, při kterém použijete tento znak. Přemýšlejme spolu. Nyní můžeme znaky z ROM kopírovat do RAM. Můžete

přitom zvolit jednotlivé znaky. Ohledně programování znaku vám chybí již jen jeden bod, a to ten nejdůležitější - definice vlastního znaku.

Pamatujete si ještě, jak jsou znaky v ROM uloženy? Každý znak je uchován jako skupina 8 bytů. Bitová maska je zadávaná příkazy pomocí bytů. Každý znak je tedy uchován jako matice 8*8 bodů, kde bod svítí (1) nebo nesvítí (0). Svítí-li, pak je na tomto místě bod, nesvítí-li, pak je tu mezera. K definici vašeho znaku stačí na adresu znaku, který chcete předefinovat, zapsat hodnoty pro nový znak. Zadejte NEW a vyzkoušejte následující program:

```
10 FOR I =12488 TO 12455: READ A: POKE I,A: NEXT
20 DATA 60,66,165,129,165,153,66,60
```

Zadejte RUN. Program udělá z klávesy "T" jakýsi "obličej". Chcete-li si obličej lépe prohlédnout, stiskněte vícekrát T. Každé číslo v DATA na řádku 20 je jedním řádkem v tomto obličeji. Ten je kreslen podle následující matice:

```
7 6 5 4 3 2 1 0     binární   DEC
 : :x:x:x:x: :      111100     60: : : : : : :
 :x: : : : :x:      1000010    66--------------
x: :x: : :x: :x     10100101  165: : : : : : :
x: : : : : : :x     10000001  129--------------
x: :x: : :x: :x     10100101  165: : : : : : :
x: : :x:x: : :x     10011001  153--------------
 :x: : : : :x:      1000010    66: : : : : : :
 : :x:x:x:x: :      111100     60--------------
 --------------                   : : : : : : :
                                  --------------
                pracovní matice   : : : : : : :
                                  --------------
                                  : : : : : : :
                                 7=6=5=4=3=2=1=0
```

Pracovní matice vám může pomoci při programování vašich vlastních znaků. Je to plocha 8*8 bodů. V příloze najdete převodní tabulku mezi binárním a dekadickým kódem. Jinak lze postupovat tak, že sečtete dvojky umocněné číslem sloupce, ve kterém je vepsána jednička - bod svítí. Tedy:

217+215+211=162 (DEC)=10100010(BIN)

K definici jednoho znaku je potřeba 8 čísel od 0 do 255. To je vše, co k tomu potřebujete.

Upozornění: Vertikální linie ve vašich znacích by měly být široké alespoň dva body (bity). Tím bude při zobrazování na obrazovce zmenšená chyba v barvě. Linie s jedním bodem vycházejí bez videopřipojení monitoru rozmazané.

Nyní se podíváme na příklad programu s programovatelnými znaky:

```
10 REM * příklad 1 *
20 REM tvorba programovatelných znaků
31 POKE56334,PEEK(56334)AND251:REM klávesnice a I/O vyp.
35 FOR I=0TO63:REM počet kopírovaných znaků z ROM
36 FOR J=0TO7:REM 8 bytů pro znak
37 POKE12288+I*S+J,PEEK(53248+I*S+J):REM kopírování bytů
38 NEXTJ:NEXTI:REM další byte či znak
39 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1:
   REM zapnout klávesnici a I/O
40 POKE53272,(PEEK(53272)AND240)+12:
   REM VIC bere informace o znacích z adresy 12288
60 FOR CHAR=60TO63:REM programuje znaky 60-63
80 FOR BYTE =0TO7:REM 8 bytů pro znak
100 READ NUMBER:REM čte 1/8 dat pro znak
120 POKE 12288+(S*CHAR)+BYTE,NUMBER:REM data do paměti
140 NEXTBYTE:NEXTCHAR:REM teď bere další byte nebo znak
150 PRINTCHR$(147)TAB(255)CHR$(60);
155 PRINTCHR$(61)TAB(55)CHR$(62)CHR$(63)
160 REM řádek 150 dá nové definované znaky na obrazovku
170 GETA$:REM čeká na stisk klávesy
180 IF A$=""THENGOTO 170:REM návrat normálních znaků
200 DATA 4,6,7,5,7,7,3,3:REM data pro znak 60
210 DATA 32,96,224,160,224,224,192,192:REM data pro znak 61
220 DATA 7,7,7,31,31,95,143,127:REM data pro znak 62
230 DATA 224,224,224,248,248,248,240,224:REM data znaku 63
240 END
```

Nyní už stačí jen zadat RUN a pochopit, jak je program sestaven. K tomu vám pomohou doprovodné REM.

Vícebarevná grafika

Díky standardní vysokorozlišitelné grafice můžete zobrazit na obrazovku i samostatné body. Pro každý bod musí být do paměti, stejně jako pro znak, zadáno 1 - svítí, 0 - nesvítí. Má-li tedy bod hodnotu 1, pak bude na obrazovce zobrazen ve vámi zvolené barvě.

U vysokorozlišitelné grafiky (HIRES) může být v matici 8*8 bodů zobrazena barva pozadí a barva bodu. Tím je omezeno použití barev v této oblasti. Tak mohou vznikat potíže, například když se kříží dvě čáry různých barev. Tento problém odpadá u vícebarevné grafiky (MULTICOLOR). Zde může každý bod obdržet tyto barvy: barva obrazovky (registr barvy pozadí #0), barva pozadí registru #2, barva pozadí registru #3 a barva znaku. Jediné omezení je v horizontálním rozlišení. Tam je ve vícebarevném zobrazení každý bod zdvojen oproti vysokému rozlišení. O tom se však zmíníme až později.

Mód vícebarevné bitové grafiky

K zapnutí módu pro vícebarevné znaky bude bit 4 řídícího registru VIC nastaven na 1. K tomu použijte tento POKE na adrese 53270 ($D016):

```
POKE 53270,PEEK(53270)OR16
```

K vypnutí tohoto módu stačí bit 4 na téže adrese 53270 změnit pomocí následujícího POKE na 0:

```
POKE 53270,PEEK(53270)AND239
```

MULTI mód bude vypnut či zapnut pro každou pozici na obrazovce, takže můžeme kombinovat grafiku HIRES a MULTI. To zařídíme pomocí 3. bitu v barevné paměti. Ta začíná na adrese 55296 ($DB00 HEX). Je-li číslo v barevné paměti menší než 8 (0 - 7), tak platí pro odpovídající pozici na obrazovce vysokorozlišitelná

grafika se zvolenou barvou (0 - 7). Je-li číslo v paměti v intervalu 8 - 15, potom bude odpovídající pozice zobrazována ve vícebarevné grafice. Barva znaku může být změněna POKE do barevné paměti. POKE s číslem od 0 do 7 bude znak ukazovat v normální barevné rozlišitelnosti. S POKE od 8 do 15 platí pro tuto pozici obrazovky vícebarevný mód. Zapnutím 3. bitu v barevné paměti bude zvolen mód MULTI, vypnutím bude zvolen HIRES.

Takto vypadá v HIRES módu zobrazené písmeno "A", kde barva znaku je tam, kde je bit 1:

```
  bitová maska:
  xx   00011000
 xxxx  00111100
xx  xx 01100110
xxxxxx 01111110
xx  xx 01100110
xx  xx 01100110
xx  xx 01100110
```

Zatímco u MULTICOLOR se používá "páru":

```
     bitová maska:
   AA BB 00 01 10 00
   CC CC 00 11 11 00
AA BB AA BB 01 10 01 10
AA CC CC BB 01 11 11 10
AA BB AA BB 01 10 01 10
AA BB AA BB 01 10 01 10
AA BB AA BB 01 10 01 10
           00 00 00 00
```

V obou vyobrazeních je AA v barvě pozadí #1, BB v barvě pozadí #2, CC v barvě znaku. O tom hovoří následující tabulka s bitovými páry:

bit. pár	barvový registr	paměťové místo
00	barva pozadí #0 (obraz)	53281 ($D021)
01	barva pozadí #1	53282 ($D022)
10	barva pozadí #2	53283 ($D023)
11	spodní 3 bity cRAM: barevná paměť	

Zadejte NEW a následující program:

```
100 POKE53281,1:REM pozadí #0 jako bílé
110 POKE53282,3:REM pozadí #1 jako tyrkysové
120 POKE53283,8:REM pozadí #2 jako oranžové
130 POKE53270,PEEK(53270)OR16
140 C=13*4096+8*256:REM C jako čítač barevné paměti
150 PRINTCHR$(147)"AAAAAAAAAA"
160 FORL=0TO9
170 POKEC+L,8:REM použití MULTI černé
180 NEXT
```

Barva obrazovky je bílá, znaku černá, jeden registr má tyrkysovou, druhý oranžovou barvu.

Nezadává skutečné kódy barev pro barvu znaku, ale vlastně používáte odkaz na dočasný barvový registr. Tak je šetřena paměť, neboť k volbě jsou používány dva bity - mezi 16 popř. 8 barvami pozadí nebo znaku. Díky tomu jsou možné rafinované triky. Jednoduchou změnou nepřímého registru bude každý bod, který je zobrazován v určité, barvě změněn. Všechno, co je ukazováno v barvě obrazovky či pozadí, může být změněno na téže obrazovce. Nyní uvidíte příklad ke změně barvy pozadí #1. Názvy v programu napsané v hranatých závorkách značí klávesy, které máte současně stisknout. Tyto zprávy zásadně do programu nepište, pouze proveďte stisknutí kláves.

```
100 POKE53270,PEEK(53270)OR16:REM zapnutí MULTICOLOR
110 PRINTCHR$(147)CHR$(18);
120 PRINT"[commodore 1][commodore 1]";:
    REM com.1 - oranžové pozadí MULTICOLOR
130 FORL=1TO22:PRINTCHR$(65);:NEXT
135 FORL=1TO500:NEXT
140 PRINT"[CTRL 7][CTRL 7]";:REM CTRL + 7 pro modrou
145 FORT=1TO500:NEXT
```

```
150 PRINT"[CTRL 1] STISKNI KLAVESU"
160 GETA$:IFA$=""THEN160
170 X=INT(RND(1)*16)
180 POKE 53282,X
190 GOTO 160
```

Pomocí klávesy COMMODORE a kláves pro změnu barvy mohou dostat všechny znaky libovolnou barvu. Zadejte například následující příkaz:

```
POKE 5470,PEEK(53270)OR16:PRINT" CTRL 3 ";
```

Slovo READY a všechno další, co zadáte klávesnicí, bude zobrazeno ve vícebarevném módu. Pomocí jiného řízení barev můžete volit i normální mód.

Dále se podívejme na příklad programu s programovatelnými vícebarevnými znaky:

```
10  REM *PRIKLAD 2*
20  REM programovatelné MULTICOLOR znaky
31  POKE56334,PEEK(56334)AND254:POKE 1,PEEK(1)AND251
35  FORI=0TO63:REM počet znaků kopírovaný z ROM
36  FORJ=0TO7:REM kopíruje všech 8 bitů znaku
37  POKE 12288+I*8+J,PEEK(53248+I*8+J):REM kopíruje byty
38  NEXTJ,I:REM skok na další byte nebo znak
39  POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1:
    REM zapnutí I/O a klávesnice
40  POKE53272,PEEK (53272)AND240)+12:
    REM nastaví znakovou paměť na adresu 12288
50  POKE53270,PEEK(53270)OR16
51  POKE53281,0:REM nastaví barvu pozadí #0 na černou
52  POKE53282,2:REM nastaví barvu pozadí #1 na červenou
53  POKE53283,7:REM nastaví barvu pozadí #2 na žlutou
60  FORCHAR=60T063:REM programuje znaky 60-63
80  FORBYTE=0TO7:REM všech 8 bitů znaku
100 READNUMBER:REM čte 1 osminu dat znaku
120 POKE12288+(8*CHAR)+BYTE,NUMBER:
```

```
    REM uložení dat do paměti
140 NEXTBYTE,CHAR
150 PRINT" SHIFT + CLR/HOME "TAB(255) CHR$(60) CHR$(61)
  TAB(55) CHR$(62) CHR$(63)
160 REM řádek 150 posílá nové definované znaky na obrazovku
170 GETA$:REM čeká na stisk klávesy
180 IFA$="" THEN170:REM není-li klávesa stisknuta - znovu
190 POKE53272,21:POKE53270,PEEK(53270)AND239:
    REM normální znaky
200 DATA129,37,21,29,93,85,85,85:REM data znaku 60
210 DATA66,72,84,116,117,85,85,85: REM data znaku 61
220 DATA87,85,21,8,8,40,0:REM data znaku 62
230 DATA213,213,85,84,32,32,40,0:REM data znaku 63
240 END
```

Rozšířený mód barvy pozadí

V tomto režimu můžete řídit každý jednotlivý znak jak v barvě pozadí, tak v barvě popředí. Tak je např. možné zobrazit na bílé obrazovce modré znaky se žlutým pozadím. Pro rozšíření módu barvy pozadí jsou 4 registry. Pro každý registr můžeme zvolit jednu ze 16 barev. V tomto módu budou v barevné paměti pevně zadány barvy popředí. Použití je stejné jako u standardního znakového módu. U rozšířeného módu je počet různých oznamovaných znaků omezen. Je-li zapnut rozšířený barevný mód, můžeme používat pouze prvních 64 znaků ze znakové ROM (nebo prvních 64 ve vaší znakové sadě). Dva bity znakových kódů budou totiž použity pro volbu barvy pozadí. Znakový kód (který je POKEován na obrazovku) písmene A je 1. V rozšířeném barevném módu se po POKE s hodnotou 1 objeví A. Normální znakový mód musí zobrazit po POKE 65 znak se znakovým kódem CHR$(129), tedy "obrácené A". To se v rozšířeném barevném módu nestane. Zde se rozsvítí stejné A jako předtím, jen v jiné barvě pozadí. Kód si vyberte z následující tabulky:

znakový kód			registr barvy pozadí	
okruh	bit 7	bit 6	číslo	adresa
0-63	0	0	0	53281($D021)
64-127	0	1	1	53282($D022)
128-191	1	0	2	53283($D023)
192-255	1	1	3	53284($D024)

K zapnutí rozšířeného barevného módu bude mít 6 registrů VIC s adresou 53265 ($D011 HEX) nastaven na 1.

Toho lze dosáhnout následujícím POKE:

POKE 53265,PEEK(53265) OR64

K vypnutí je třeba tentýž bit registru nastavit na 0. To docílí následující příkaz:

POKE 53265,PEEK(53265)AND191

Grafika za pomoci bitové mapy

Při psaní her, tabulek a dalších různých náročnějších programů budete dříve nebo později potřebovat zobrazení ve vysokorozlišitelné grafice HIRES. Commodore 64 nám tedy nabízí programovat vysokorozlišitelnou grafiku pomocí bitové mapy. Bitová mapa je metoda, při níž je každý zobrazovaný bod na obrazovce řízen jedním bitem z určitého místa v paměti. Je-li tento bit 1, pak bod svítí, je-li tento bit 0, pak bod nesvítí. Práce s vysokorozlišitelnou grafikou má přece jen jisté nevýhody, a proto se nepoužívá stále. Nejdříve bude bitovou mapou celé obrazovky zaplněna podstatná kapacita paměti. Každý bod potřebuje jeden bit, tj. potřebuje 1 byte pro 8 bodů. Protože znak je 8*8 bodů a rozměr obrazovky je ve znacích 40*25 znaků, potřebuje HIRES grafika pro svých 320*200 bodů celkem 64000 bodů, tedy 8000 bytů.

Ve všeobecné přípravě pro nastavení grafiky HIRES se používá mnoha krátkých, avšak opakujících se rutin. Ty jsou však pro takový jazyk, jako BASIC, velmi pomalé. Strojový kód má v těchto rutinách výhodu rychlosti. Proto se nám spíše vyplatí programovat

grafiku ve strojovém kódu, a pak ji vyvolávat příkazem SYS. Ostatně, co si budeme namlouvat, dělat grafiku v BASICU je stejně obtížné, jako ji programovat ve strojáku. Pouze příkaz POKE je zde nahrazen assemblerovským LDA a STA.

Avšak v této kapitole se budeme zabývat příklady napsanými v BASICU. A nyní k technickým detailům:

BITOVÁ MAPA - s C64 jsou možné dvě různé bitové mapy

1) Standardní bitová mapa (HIRES) - 320x200 bodů
2) Vícebarevná bitová mapa (MULTI) - 160x200 bodů

Při standardním módu je sice vyšší rozlišovací schopnost, ale zato je omezený výběr barev. U vícebarevné grafiky bude menší horizontální rozlišení vyváženo možností většího výběru barev v matici 8x8 bodů.

Standardní bitová mapa s vysokým rozlišením

U standardní matice máte možnost využívat rozlišovací schopnost 320x200 bodů a můžete si zvolit v matici 8x8 bodů mezi dvěma barvami. K zapnutí bitové mapy musí být na 5. bitu adresy 53265 ($D011 HEX) nasazena 1. Proto použijte následující příkaz:

```
POKE 53265,PEEK(53265)OR32
```

K vypnutí poslouží nastavení 5. bitu téže adresy na hodnotu 0. K tomu použijte následující příkaz:

```
POKE53265,PEEK(53265)AND223
```

Před tím, než se budeme dále zabývat bitovou mapou, musíme vyřešit další problém - umístění bitové mapy.

Moudrá funkce

Pokud si ještě vzpomínáte na odstavec "Programovatelné znaky", pak zajisté víte, že bitové vzory znaků uložené v RAM je možno měnit. Je využívána stejná obrazovka, jako s programovatelnými znaky. Každé místo v obrazovkové paměti, které je používáno pro řízení návratu znaku, bude nyní užíváno pro informace o barvách. Tak nyní nebude po zadání POKE 1024,1 v levém horním rohu

zobrazeno A, ale bude určena barva bodu v levém horním rohu. Při bitových mapách nepřichází barvy 1000 obrazových pozic z barevné paměti, jako při normálním módu. Barvy budou brány z obrazové paměti. Vrchní čtyři bity obrazové paměti obsazují pevně barvy pro bity, které jsou v tomto paměťovém místě obrazovky (v matici 8x8 bodů) nasazeny na 1. Spodní čtyři bity obsahují barvu pro bity nasazené na 0. Příklad - zadejte následující:

```
5 BASE=2*4096:POKE53272,PEEK(53272)OR8:
    REM bitová mapa od 8192
10 POKE 53265,PEEK(53265)OR32:REM zapnutí módu bitové mapy
```

Zadejte nyní na ukázku RUN.

Na obrazovce se nyní neobjevilo nic potřebného, že? Stejně jako "normální" obrazovka, musí být nejdříve i HIRES obrazovka smazána. V tomto případě však nefunguje funkce CLR. Musíte vynulovat mnoho paměti, kterou používáte pro svoji grafiku. Stiskněte klávesy RUN/STOP+RESTORE a přidejte ke svému programu ještě tyto řádky na smazání HIRES obrazovky:

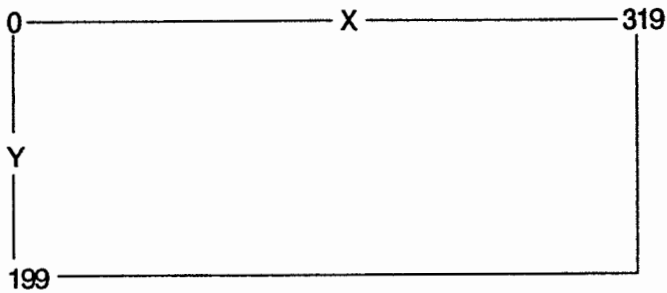
```
20  FOR I =BASE TO BASE+7999:POKEI,0:NEXT:
    REM smazaná bitová mapa
30  FORI =1024 TO 2023:POKEI,3:NEXT:
    REM nastaveny barvy tyrkysová a černá
```

Napište opět RUN a odešlete. Obrazovka bude smazaná a dostane po celé ploše tyrkysovou barvu. Nyní můžete na HIRES obrazovce zhášet a rozsvěcovat jednotlivé body. Chcete-li zobrazit nebo smazat nějaký bod, musíte vědět, jak najít ten správný bit ve znakové paměti, který má dostat hodnotu 1. Proto musíte vyhledat příslušný řádek znaku, jakož i odpovídající bit v tomto řádku.

Pro toto propočítání potřebujete několik rad:

používáme X a Y pro horizontální a vertikální pozici bodu. Bod, který má X=0 a Y=0, se nalézá v levém horním rohu obrazovky. Napravo ležící body mají vyšší hoddnotu X a všechny body ležící směrem dolů mají vyšší hodnotu Y.

Toto je schématický náčrt:



Každý bod má svoje souřadnice X a Y. V tomto souřadnicovém systému je každý bod na obrazovce snadno popsán. Programovatelné znaky bitové mapy jsou uspořádány do 25 řádků a 40 sloupců. To je sice dobrá metoda pro textové zobrazení, horší je to však u bitové mapy (pro tuto metodu je dobrý důvod, nahlédni do oddílu "Kombinované druhy provozu").

Díky následujícím řádkům se dá snadno určit místo, na kterém bude bod zobrazen. Začátek obrazové paměti bude označen jako BASIC. Číslo řádku (od 0 do 24) vašeho bodu je:

ROW=INT(Y/8) (je 320 bytů na řádek)

Pozice znaku na této řádce je (0-39):

CHAR=INT(X/8) (je 8 bytů na znak)

Řádek této znakové pozice je (0-7):

LINE=Y AND 7

Bit v tomto bytu je:

BIT=7-(X AND 7)

Nyní dáme všechno dohromady. Byte, ve kterém leží paměťové místo bodu X,Y, bude spočítán:

BYTE=BASE+ROW*320+CHAR*8+LINE

Chcete-li zobrazit libovolný bod se souřadnicemi X,Y, použijte následující řádek:

POKE BYTE,PEEK(BYTE)OR21BIT

Použijme tento propočet v programu. V následujícím příkladu zobrazí C64 sinusoidu:

```
50  FORX=0TO319STEP.5:REM vlnovka vyplní obrazovku
60  X=INT(90+80*SIN(X/10))
70  CH=INT(X/8)
80  RO=INT(Y/8)
90  BY=BASE+RO*320+8*CH+LN
100 BI=7-(XAND7)
110 POKEBY,PEEK(BY)OR(21BI)
120 NEXT X
125 POKE1024,16
130 GOTO 130
```

Jako další příklad bude uveden program kreslící sinusoidu změněnou tak, že vychází jako půlkruh. Tento program je připsán k předchozímu, proto nezadávejte NEW.

```
50  FORX=0TO160:REM polovina obrazovky
55  Y1=100+SQR(160*X-X*X)
56  Y2=100-SQR(160*X-X*X)
60  FORY=Y1TO Y2STEPY1-Y2
70  CH=INT(X/8)
80  RO=INT(Y/8)
85  LN=YAND7
90  BY=BASE+RO*320+8*CH+LN
100 BI=7-(XAND7)
110 POKEBY,PEEK(BY)OR(21BI)
114 NEXT
```

Tímto programem bude na HIRES obrazovce zobrazena půlkružnice.

POZOR: proměnné BASICU mohou přepsat vaší obrazovku HIRES.

Používáte-li více paměťového místa, musíte přeložit začátek BASICU nad oblast HIRES obrazovky nebo obrazovku přesunout. Tento problém u strojového kódu odpadá.

Vícebarevná bitová mapa - MULTICOLOR

Jako ve vícebarevném módu znaky, tak také při vícebarevné bitové mapě můžete zobrazit v matici 8*8 bodů čtyři různé barvy. Také zde je však zredukováno horizontální rozlišení z 320 na 160 bodů. Pro vícebarevnou bitovou mapu je použito 8 KB paměti.

Barvy vícebarevné bitové mapy zvolíte: první z registru barvy pozadí 0 (obrazovka - barva pozadí), druhou z videomatrice (horní čtyři bity udávají první možnou barvu, spodní 4 bity udávají druhou možnou barvu) a třetí z barevné paměti. K zapnutí tohoto režimu bude bit 5 na adrese 53265 ($D011) a bit 4 na adrese 53270 ($D016) nasazen na 1. To obstará následující POKE:

```
POKE 53265,PEEK(53265)OR32:POKE53270,PEEK(53270)OR16
```

Ke zrušení módu MULTICOLOR musí být tytéž bity nasazeny na 0. K tomu slouží následující POKE:

```
POKE 53265,PEEK(53265)AND223:POKEE 53270, PEEK(53270)
AND239
```

Jako při standardní bitové mapě HIRES přetrvává poměr 1:1 mezi pro zobrazení užívanými 8 KB paměti a zobrazením na obrazovce. Horizontální bod je však široký 2 bity. Každé dva bity v zobrazovací paměti tvoří jeden bod, který může mít jednu ze čtyř barev:

BITY	informace o barvě pozadí přicházejí z
00	barva pozadí #0 (barva obrazovky)
01	vyšší 4 bity obrazové paměti
10	nižší 4 bity obrazové paměti
11	barvový nibble (nibble=1/2 bytu=4 bity)

Kontinuální přesunování

S čipem VIC je možné jednoduché "přesouvání" (rolování obrazovky) jak v horizontálním, tak ve vertikálním směru. Přesunování je vlastně jednobodový pohyb celé obrazovky v jednom směru. Pohyb může být proveden dolů, nahoru, vlevo i vpravo. Takto budou

zobrazovány nové informace a současně zmizí znaky na protilehlé straně obrazovky. Když čip VIC přebírá mnoho vašich úloh, musí toto přepisování probíhat nejlépe ve strojovém kódu. Díky čipu VIC může být přenesena videoobrazovka v libovolných 8 horizontálních a 8 vertikálních pozicích. Pozice budou nastaveny registrem VIC (nazývá se SCROLL-registr) k rolování obrazovky.

Čip VIC má také mód 38 sloupců a 24 řádků. Menší velikost obrazovky vám dá prostor pro nová data, která budete potřebovat při přesunování.

Postupujte při posunování následovně:

1) Zmenšit obrazovku (okraj obrazovky bude silnější).
2) SCROLL-registr nastavit na maximální hodnotu (nebo min. podle směru rolování).
3) Zadat nová data do vhodné oblasti obrazovky.
4) SCROLL-registr zvětšit (nebo zmenšit), až dosáhne maximální (nebo minimální) hodnoty.
5) V jednom okamžiku odrolovat celý obraz ve směru přepisování o znak. Použijte k tomu svoji rutinu ve strojovém kódu.
6) Nyní se vraťte k bodu 2.

K přechodu na 38znakový mód bude 3. bit adresy 53270 ($D016) nastaven na 0. To provede následující POKE:

```
POKE 53270,PEEK(53270)AND247
```

Ke zpětnému návratu do 40znakového módu je nutné tento bit opět nastavit na 1. To provede následující POKE:

```
POKE 53270,PEEK(53270)OR8
```

Pro 24řádkový mód bude 3. bit na adrese 53265 ($D011) změněn na 0. K tomu použijte toto:

```
POKE 53265,PEEK(53265)AND247
```

Při návratu do módu 25řádkového musí být tentýž bit nastaven na 1. To provedete takto:

```
POKE 53265,PEEK(53265)OR8
```

U přesunování ve směru osy X musí být čip v módu 38 znaků. Tím bude vytvořeno místo pro nová data. Při posunování doleva

budou nová data zadávána pochopitelně vpravo. Posunováním vpravo vyjdou nová data na odpovídajícím místě vlevo. Všimněte si prosím, že obrazová paměť má stále ještě 40 sloupců, jen 38 jich však je viditelných.

Při přesouvání ve směru osy Y musí být čip VIC v módu 24 řádků. Při rolování nahoru se budou nová data objevovat v poslední řádce. Při rolování dolů se naproti tomu budou nová data objevovat v první řádce.

Při X-posunu jsou neviditelné oblasti po obou stranách obrazovky. Při Y-posunu je neviditelná však jen jedna oblast. Je-li Y -scroll registr nastaven na 0, pak je první řádka neviditelná a připravena na nová data. Je-li Y-scroll registr nastaven na 7, pak je neviditelná poslední řádka. K rolování ve směru osy X se nalézá SCROLL -registr na 2. až 0. bitu řídícího registru čipu VIC - na adrese 53270 ($D016 HEX). Také zde smí být měněny v tomto případě pouze tyto bity. To zajistí následující příkaz:

POKE 53270,(PEEK(53270)AND248)+X

kde X je obrazovková souřadnice od 0 do 7.

K rolování ve směru osy Y se nalézá SCROLL-registr na bitu 2 až 0 řídícího registru čipu VIC - na adrese 53265 ($D011 HEX). Také zde smíme změnit jen tyto dva bity. K tomu slouží následující POKE:

POKE 53265,(PEEK(53265)AND248)+Y

kde Y udává obrazovkovou polohu Y od 0 do 7.

Aby text roloval odspodu obrazovky, musí být spodní 3 bity adresy 53265 nastaveny mezi 0 - 7, zadána další data v odkrývané řádce dole na obrazovce a potom bude postup opakován. Mění-li se posouvaný bit s délkou kroku -1, tak bude text posunut v opačném směru.

Příklad: Rolování odspodu obrazovky nahoru

```
10 POKE 54365,PEEK(53265)AND247:REM mód 24 znaků
20 PRINTCHR$(147):REM maže obrazovku
30 FORX=1TO1024:PRINTCHR$(17):NEXT:
   REM přenese kurzor dolů
```

```
40 POKE53265,(PEEK(53265)AND248)+7:PRINT:
   REM pozice pro první scrollování
50 PRINT " AHOJ";
60 FORP=6TO0STEP-1
70 POKE53265,(PEEK(53265)AND248)+P
80 FORX=1TO50:NEXT:REM zpožďovací smyčka
90 NEXT:GOTO40
```

Dodatky

Seznam důležitých adres

53265 ($D011) - řídící registr VIC
bit 7 - podoba rastru: (Bit 8)
nahlédni na 53266
bit 6 - rozšířený mód barevného textu
1=zapnut
bit 5 - mód bitové mapy
1=zapnut
bit 4 - zhasnutí obrazovky
1=zhasnuta
bit 3 - volí zobrazení 24/25 řádek
1=25 řádek
bit 2-0 - rolování po pozici Y
0-7
53266 ($D012) - zápis/čtení rastr
hodnota podoby IRQ
53270 ($D016) - řídící registr VIC
bit 7-6 - nepoužit
bit 5 - TENTO BIT MUSÍ MÍT HODDNOTU 0!
bit 4 - vícebarevný mód
1=zapnut

bit 3 - volí zobrazení 38/40 znaků na řádek
1=40 znaků
bit 2-0 - rolování po pozici X
53272 ($D018) - řídící registr paměti VIC
bit 7-4 - základní adresa VIDEO matrice
bit 3-1 - základní adresa znakového generátoru
53273 ($D019) - příznak přerušení VIC (bit nastaven na 1 - IRQ zapnuto)
53280 ($D020) - barva okraje obrazovky
53281 ($D022) - barva pozadí 0
53282 ($D022) - barva pozadí 1
53283 ($D023) - barva pozadí 2
53284 ($D024) - barva pozadí 3

Mazání grafické obrazovky

Trvá v BASICU kolem minuty. My si ukážeme jednoduchý trik, který nám dobu smazání HIRES z obrazovky zkrátí na vteřinu:

```
POKE47,249:POKE48,31:POKE49,249:POKE50,31
DIM AR%(3999):CLR
```

Jak je vidět, program řeší smazání grafické obrazovky jednoduše - nadimenzuje do ní samé nuly. Jedinou nevýhodou programu je, že maže veškeré proměnné. Proto je ještě lepší používat strojového kódu. Smyčku podobnou FOR TO NEXT v BASICU utvoří tento program:

```
C300  A9 00 85 FC A9 20 85 FD
C308  A9 00 A0 00 91 FC E6 FC
C310  D0 02 E6 FD A6 FC E0 41
C318  F0 03 4C 0C C3 A6 FD E0
C320  3F F0 03 4C 0C C3 60 00
```

Program zadáte pomocí monitoru a spustíte příkazem SYS49920. Za půl vteřiny vám smaže celou obrazovku HIRES od adresy 8192. Pokud jste ve strojáku zběhlejší, nedá vám práci

program modifikovat i pro jiné oblasti paměti. Adresa, odkud se bude mazat, totiž leží hned na řádku C300 a C304 - jsou zadány systémem LOHI.

Jak vidíte, strojový kód bude asi přece jen rychlejší a efektivnější. S chutí do toho! Nebo že by cesta vedla přes vyšší jazyky?

COMMPAS
vydavatelství, služby a distribuce
Dr. Ivan Pavlíček, Pavel Škvrně
p. o. box 80
140 00 Praha 4