COMMPAS
vydavatelství a služby
uživatelům mikropočítačů

Základy programování

2. díl

GRAFIKA

* Přehled grafických možností

* Uložení grafických dat

* Standardní znakový mód

* Programovatelné znaky

* Mód multicolor

* Mód s rozšířením barev pozadí

* Grafika v bitové mapě

* Mód multicolor v bitové mapě

* Jemné skrolování

* Skřítci (Sprites)

* Další vlastnosti grafiky

* Programování skřítků - jiný pohled

Přehled grafických možností

Za všechny grafické vymoženosti vděčí C64 obvodu 6567 (známý také jako VIC-II, Video Interface Chip). Tento čip umožňuje množství grafických modů, jako zobrazení textu ve 40 sloupcích a 25 řádcích, 320 x 200 bodovou jemnou grafiku, vytváření malých programovatelných objektů, schopných pohybu po obrazovce - skřítků - které usnadňují vytváření her. A pokud toho nebylo dost, mnoho grafických modů lze směšovat na téže obrazovce. Tak je např. možné zadefinovat horní polovinu obrazovky pro práci v jemné grafice, zatímco spodní polovina pracuje v textovém módu. A skřítkové mohou být používáni v kombinaci s jakýmkoli módem.
VIC-II má následující grafické módy:

A) Znakový mód:

1) Znakový mód standardní
   a) znaky z ROM generátoru
   b) znaky programovatelné v RAM
2) Znakový mód multicolor
   a) znaky z ROM generátoru
   b) znaky programovatelné v RAM
3) Znakový mód s rozšířením barev pozadí
   a) znaky z ROM generátoru
   b) znaky programovatelné v RAM

B) Jemná grafika:

1) Standardní bit-mapový mód
2) Multicolor bit mapový mód

C) Skřítková grafika:

1) Standardní skřítci (sprites)
2) Multicolor skřítci

Uložení grafických dat

Nejprve hlavní informace. Na obrazovce TVP je 1000 možných znakových lokací. Normálně začíná obrazovka na lokaci 1024 ($0400) a končí na lokaci 2023. Každé této lokaci přísluší 8 bitů paměti, tzn., že každá lokace může uchovat jakékoliv přirozené číslo od 0 do 255. K obrazovkové paměti dále patří skupina 1000 lokací zvaná COLOR RAM (paměť barvy). Ta začíná na lokaci 55296 ($D800) a končí na 56295. Každá buňka této paměti má pouze čtyři bity, což znamená, že může uchovat číslo od 0 do 15. A protože C64 pracuje se 16 barvami, vychází to dobře.

Kromě toho může být současně zobrazeno 256 různých znaků. Ve standarním znakovém módu každá z 1000 buněk obrazovkové paměti ibsahuje kódové číslo, které říká VIC-II čipu, který znak se má na příslušném místě obrazovky zobrazit. Různé grafické módy se ovládají a volí pomocí 47 ovládacích registrů čipu VIC-II. Mnoho grafických funkcí může být ovládáno POKEování správných hodnot do těchto registrů. Registry čipu VIC-II jsou umístěny na lokacích 53248 ($D000) až 53294 ($D02E).

Volba VIDEOBANKY

VIC-II čip může mít najednou přístup k datům z bloku 16 k B paměti. Protože C64 má 64 kB paměti, budete asi chtít, aby měl VIC II přístup k celé paměti. Proto existují čtyři VIDEOBANKY po 16 kB paměti.
Bity provádějící výběr videobanky jsou umístěny v obvodu 6526 (Complex Interface Adapter chip - CIA 2). Pomocí POKE a PEEK z BASICu (nebo jejich verzemi ve strojním jazyce) ovládáme bity 0 a 1 portu A obvodu CIA 2 na lokaci 56576 ($DD00(k volbě videobanky. Oba bity však musí být dříve nastaveny jako výstupní. To provedeme nastavením bitů 0 a 1 na lokaci 56578 ($DD02) na 1. Následující příklad Vám to ozřejmí:

```
POKE 56578, PEEK(56578)OR3: REM ZAJISTI, ABY BITY 0 A 1
   BYLY NASTAVENY JAKO VYSTUPNI
POKE 56576, PEEK(56576)AND252)OR A: REM ZVOL VIDEOBANKU
```

"A" by mělo nabývat jedné z následujících hodnot:

hodnota A	bity	čís.banky	poč.adresa	VIC-II rozsah
0	00	3	49152	($C000 - $FFFF) *
1	01	2	32768	($8000 - $BFFF)
2	10	1	16384	($4000 - $7FFF) *
3	11	0	0	($0000 - $3FFF) obvykle

* pozn.: ve videobankách 1 a 3 nemá VIC-II přístup k souboru znaků v generátoru ROM.

Vždy byste si měli být vědomi, kterou videobanku VIC-II užívá, protože to bude mít vliv na to, odkud přicházejí data, kde je obrazovková paměť, kde jsou data skřítků atd. Když Váš C64 zapnete, bity 0 a 1 na lokaci 56576 jsou automaticky nastaveny na banku 0 ($000 - $3FFF).

Obrazovková paměť

Umístění obrazovkové paměti v dané videobance lze měnit jednoduše instrukcí POKE v ovládacím registru VIC#24 na lokaci 53272 ($D018). Pozor však na to, že je tento registr užíván též k ovládání, který soubor znaků je užit. Pouze horní čtyři bity ovládají umístění obrazovkové paměti ve videobance. Pro zadání její polohy v paměti by se mohl použít následující řádek:

```
POKE 53272,(PEEK(53272)AND15)OR A
```

kde "A" má jednu z následujících hodnot:

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

* pozn.: nezapomeňte, že musíte přičíst navíc poč.adresu videobanky

Dále musíte také sdělit obrazovkovému editoru, kam jste obrazovkovou paměť umístili. Můžete to učinit např. pomocí POKE648, STRANKA, kde STRANKA = poč.adresa/256 (např. 1024/256 = 4, tedy POKE648,4).

COLOR RAM (paměť barvy)

Paměť barvy se nedá přesunovat. Je vždy umístěna v paměti od 55296 ($D800) do 56295 ($DBE7). Obrazovková paměť a paměť barvy se užívají různě v různých grafických módoech. Obrázek vytvořený v jednom módu bude často vypadat úplně jinak v módu jiném.

Kde přesně VIC-II získává informace pro vytváření znaků je velice důležité pro vytváření vlastních programovatelných znaků. Normálně čip získává informace o podobě znaku, který chcete zobrazit, ze znakového generátoru v paměti ROM. V tomto čipu jsou uloženy vzory, podle nichž se vytvářejí různá písmena, číslice, interpunkční znaménka a vůbec všechny znaky, které vidíte na klávesnici C64. Jedním z rysů C64 je však i schopnost užívat vzorů uložených v paměti RAM. Tyto vzory do paměti RAM můžete sami navrhovat a tak máte neomezený počet symbolů pro hry, obchodní aplikace atd.

Normální soubor znaků obsahuje 256 znaků, z nichž každý znak je určen daty ve skupinách osmi bajtů. Protože každý znak obsahuje 8 buněk, pak pro plný soubor znaků potřebujeme 256*8 = 2048 t.j. 2kB paměti. Jelikož VIC II nahlíží do 16kB paměti, je 8 možných umístění kompletního znakového souboru ve videobance.

Poloha znakové paměti je zadána 3 bity registru VIC+24 na adrese 53272 ($D018). Bity 3,2 a 1 udávají, kde bude umístěn 2kB-blok souboru znaků.

Bit 0 není využit. Vzpomeňte si, že tento registr udává též, kde je umístěna obrazovková paměť a dejte pozor, abyste nezměnili 4. a 7. bit tohoto registru. Ke změně polohy znakové paměti může být použit následující BASICový příkaz:

```
POKE 53272, (PEEK(53272) AND 240) OR A
```

kde "A" nabývá jedné z následujících hodnot

ROM obraz v uvedené následující tabulce se vztahuje ke znakovému generátoru ROM. Objeví se v místě paměti RAM na lokacích uvedených v tabulce ve videobance 0. Objeví se také v odpovídající oblasti RAM v bance 2, lokacích 36864 - 40959 ($9000-$9FFF). Protože čip VIC II může obsáhnout pouze 16 kB paměti současně, budou se jevit ROM vzory znaků v 16 kB bloku z hlediska čipu VIC II vždy takovýmto způsobem. Proto byl systém navržen tak, aby si VIC-II myslel, že ROM znaky jsou v bance 0 na pozicích 4096-8191 ($1000-$1FFF), v bance 2 na pozicích 36864-40959 ($9000-$9FFF), ačkoliv ROM znakový generátor je ve skutečnosti

fyzicky v paměti na lokacích 53248-57343 ($D000-$DFFF). Toto zobrazení je jenom zdánlivé a poskytuje data pro VIC-II čip. Toto zobrazení je jen zdánlivé a tato paměť může být použita pro programy, jiná data, atp. zrovna tak, jako jakákoliv jiná paměť RAM.

OKTANT	HOD. A	BITY	UMÍSTĚNÍ ZNAKOVÉ PAMĚTI	
			DECIMÁLNĚ	
HEXADECIMÁLNĚ				
0	0	xxxx 000x	0	($0000-$07FF)
1	2	xxxx 001x	2048	($0800-$0FFF)
2	4	xxxx 010x	4096	($1000-$17FF) ROM obraz v bankách 0 a 2 (normálně)
3	6	xxxx 011x	6144	($1800-$1FFF) ROM obraz v bankách 0 a 2
4	8	xxxx 100x	8192	($2000-$27FF)
5	10	xxxx 101x	10240	($2800-$2FFF)
6	12	xxxx 110x	12288	($3000-$37FF)
7	14	xxxx 111x	14336	($3800-$3FFF)

* pozn.: nezapomeňte přičíst počáteční adresu videobanky.

Pozn.: Pokud Vám tyto ROM obrazy vadí při Vaší grafice, pak nastavte bity pro výběr videobanky tak, abyste pracovali ve videobance bez těchto obrazů (banky 1 nebo 3). ROM vzory zde potom nebudou.
Umístění a obsah ROM generátoru znaků je následující:

BLOK	ADRESA	VIC-II obraz	OBSAH
	DEKADICKY HEXADECIMÁLNĚ		
0	53248 $D000-$D1FF	$1000-$11FF	velká abeceda
	53760 $D200-$D3FF	$1200-$13FR	grafické znaky
	54272 $D400-$D5FF	$1400-$15FF	inverz. vel.abeceda
	54784 $D600-$D7FF	$1600-$17FF	" - graf.znaky
	55296 $D800-$D9FF	$1800-$19FF	malá abeceda
	55808 $DA00-$DBFF	$1A00-$1BFF	vel.abc. a graf.zn.
	56320 $DC00-$DDFF	$1C00-$1DFF	inv.malá abeceda
	56832 $DE00-$DFFF	$1E00-$1FFF	inv.velká abeceda a inv. graf. znaky

Pozorní čtenáři si právě něčeho všimli. Místo v paměti, užité pro ROM gnerátor znaků je totéž, které používají ovládací registry čipu VIC II. Toto je možné, protože nepoužívají nikdy tuto lokaci současně. Když VIC II potřebuje získat data z ROM generátoru znaků, zapne jej. Tím se dostane jeho obrz do 16 kB videobanky, do které VIC-II nahlíží. Jinak jsou v této paměťové lokaci umístěny I/O ovládací registry a ROM gnerátor znaků je přístupný jedině obvodu VIC-II.

Možná však budete potřebovat ROM generátor znaků zkopírpovat, když budete chtít užívat programovatelné znaky a budete chtít pozměnit pouze některé znaky z ROM generátoru. V tomto případě musíte vypnout I/O registry VIC II a připojit ROM generátor samí a zkopírovat jej do vhodné oblasti paměti. Až skončíte, musíte opět připojit I.O registry pro VIC-II. Během kopírování (kdy jsou I/O registry vypnuty), nesmí přijít žádný interrupt. To proto, že I/O registry jsou nezbytné při obsluze interuptů. Pokud byste na to zapoměli a přišel by unterupt, děly by se opravdu divné věci. Během kopírování by se tedy neměla číst klávesnice. K vypnutí klávesnice a ostatních běžných interuptů, které by mohly do počítače přijít, můžeme použít následující příkaz:
POKE 56334, PEEK(56334) AND 254 (zabránění interruptů)

Poté, co skončíte s prací s ROM generátorem znaků a můžete pokračovat ve Vašem programu, musíte znovu zapnout skanování klávesnice pomocí následujícího POKE:
POKE 56334, PEEK(56334) OR 1 (povolení interruptů)
Následující POKE vypne I/O registry a připojí ROM generátor znaků.
POKE 1, PEEK(1) AND 251 (ovládáme 2. bit I/O reg.procesoru) ROM gnerátor znaků je nyní v paměti od 53248 do 57343 ($D000-$DFFF).

K odpojení ROM generátoru znaků a opětovnému připojení I/O registrů: POKE 1, PEEK(1) OR 4.

Standardní znakový mód

Do standardního znakového módu je C64 přepnut vždy po zapnutí. Je to mód, ve kterém budete převážně programovat. Vzory pro znaky mohou být převzaty z ROM nebo RAM, ale normálně jsou brány z ROM paměti. Když chcete pro Váš program specielní grafické znaky, musíte definovat vzory nových znaků do RAM a sdělit VIC-II čipu, odkud má brát informace o znacích namísto z původního ROM generátoru znaků. To je detailně probráno v následující kapitole.

Shrňme si nyní, že chceme-li zobrazovat znaky na obrazovce v barvách, VIC-II čip musí jednak nahlížet do obrazovkové paměti, která určuje lokaci jednotlivých znaků na obrazovce a současně nahlíží do paměti barvy, která určuje, jakou barvou chcete jednotlivé znaky zobrazit. Kód znaků z obrazovkové paměti je uvnitř VIC-II přeložen jako počáteční adresa osmibajtového bloku ve znakové paměti, která obsahuje vzor znaků.

Převod není tak složitý, ale při vytváření adresy vzoru znaku je zkombinováno množství položek. První - kód znaku, který chcete užít k pokeování do obrazovkové paměti, musíte násobit osmi. Dále přičíst počáteční adresu znakové paměti. Pak jsou brány v úvahu bity určující výběr videobanky tak, že je přičtena bázová adresa videobanky. Následující vzorec ilustruje generaci požadované adresy:

ADR. ZNAKU = (POZICE ZNAKU V ZNAK. PAMETI NA OBRAZOVCE x 8) + (OKTANT ZNAK. PAMETI x 2048) + (CISLO BANKY x 16384)

DEFINICE ZNAKU

Každý znak je tvořen mozaikou 8 x 8 bodů, kde každá tečka může být buď zapnutá nebo vypnutá. U C64 jsou vzory těchto znaků uloženy v čipu ROM generátoru znaků. Znaky jsou uloženy jako soubor 8 bajtů pro každý znak a každý bajt reprezentuje vzor jednoho řádku znaku, každý bit reprezentuje jeden bod. Pokud je bit 0, tečka je vypnutá, pro 1 je zapnutá.

Znakový generátor ROM začíná na adrese 53248 (pokud jsou I/O registry vypnuty). Prvních 8 bajtů 53248 ($D000) - 53255 ($D007) obsahuje např. vzor pro znak a , který má v obrazovkové paměti kód znaku = 0. Dalších 8 bajtů 53256 ($D008) - 53263 ($D00F) obsahují informaci pro vytvoření písmene A:

Vzor	binárně	PEEK
...ww...	00011000	24
..wwww..	00111100	60
.ww..ww.	01100110	102
.wwwwww.	01111110	126
.ww..ww.	01100110	102
.ww..ww.	01100110	102
.ww..ww.	01100110	102
....	00000000	0

Každý úplný soubor znaků zabírá 2kB paměti, 256 znaků po 8 bytech. Protože máme dva znakové soubory, jeden pro malou abecedu, jeden pro velkou, zabírá ROM generátor znaků plných 4kB lokací.

Programovatelné znaky

Protože jsou vzory znaků uloženy v ROM, zdálo by se, že je není možné měnit. Čip VIC II má však programově přístupný registr, který mu říká, kde má hledat vzory znaků. Změníme li ukazatel paměti znaků tak, aby ukazoval do RAM, soubor znaků může být pozměněn pro jakékoliv potřeby.

Pokud budete chtít užívat soubor znaků definovaný v RAM, musíte vzít na vědomí několik následujících zásad:

1) Jde o proces vše nebo nic. Obecně, pokud užijete soubor Vámi definovaných znaků, tím, že zadáte VIC-II čipu, kde získáváte informace o podobě znaků, které jste připravili do RAM, je Vám 64 standardních znaků nedostupných. Abyste je zachovali, musíte nejprve do RAM zkopírovat písmena, číslice nebo další základní grafické znaky C64, které chcete dále používat. Můžete si vybrat jen některé znaky, které změníte nebo je dokonce můžete nechat beze změny.

2) Váš znaková soubor musí být uložen mimo dosah BASICových programů. Samozřejmě, že při 38 kB RAM pro BASICové programy Vám to při většině aplikací nebude činit potíže. Musíte být opatrní, abyste si nepřepsali Váš znakový soubor BASICovým programem.

Pokud pracujete v BASICu, jsou dvě lokace paměti, kde můžete začínat Váš znakový soubor. Jsou to lokce 0 a 2048. První nemůže být použita, protože zde si operační systém ukládá důležitá data - stránka 0 paměti. Druhá nemůže být použita, protože zde začíná Váš BASICový program. Nejlepším místem pro Váš soubor znaků při práci v BASICu je lokace od 12288 ($3000).

Začněme s navrhováním grafických znaků. Abyste uchránili Váš soubor znaků před přepisem BASICovým programem, měli byste omezit množství paměti, které má BASIC k dispozici. Zadejte:

```
PRINT FREE(0) - (SGN(FRE(0)) < 0) * 65535
```

Číslo, které se nyní zobrazí, udává množství paměťového prostoru, který je dosud nevyužit. Nyní zadejte následující příkaz:

```
POKE52,48:POKE56,CLR      y a opět
PRINT FRE(0) - (SGN(FRE(0)) < 0) * 65535
```

Vidíte změnu? BASIC si teď myslí, že má k dispozici méně paměti. Paměť, kterou může BASIC používat, sahá právě do místa, kde začíná Váš soubor znaků a tak je chráněn před zásahem BASICu. Dalším krokem je přenesení vzorů znaků do RAM.

Následující program přemístí prvních 64 prvků z ROM do Vašeho souboru znaků u RAM:

```
5  PRINT CHR$(142): REM PREPNI NA VELKOU ABECEDU
10  POKE  52,48:  POKE56,48:  CLR: REM  REZERVUJ  PAMET  PRO
   ZNAKOVY SOUBOR
20  POKE  56334,   PEEK(56334)AND254:REM  VYPNI  PRERUSOVACI
   TIMER PRO SKANOVANI KLAVESNICE
30 POKE 1, PEEK (1)AND251: REM AKTIVUJ ROM GENERATOR ZNAKU
40  FOR I = 0 TO 511: POKE I + 12288, PEEK(I+53248) : NEXT:
   REM PRENES ZNAKY
50 POKE 1, PEEK(1)OR4 : REM AKTIVUJ I?O REGISTRY
60  POKE  56334,  PEEK(56334)OR1.  REM  RESTARTUJ  SKANOVACI
   PRERUSOVACI TIMER
70 END
```

Nyní přepněte znakovou paměť na lokaci a počáteční adresou 12288 pomocí POKE 53272, (PEEK(53272)AND 240)OR 12 – nic se nestalo, že? Téměř nic, kromě toho, že COMMODORE 64 nyní získává vzory pro znaky z Vaší RAM místo z ROM generátoru. Ale protože jste zkopírovali znaky z ROM přesně, žádná změna není vidět ... ještě ne.

Nyní můžete jednoduše změnit znak. Smažte obrazovku a zadejte znak a. Pak sjeďte kurzorem o několik řádků níže a zadejte:

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

Právě jste vytvořili reverzovaný znak a !
Nyní vraťte kurzor do předchozího programu a pomocí RETURN znova reverzujte znak (zpět do normální podoby). Z tabulky bodů znaků můžete snadno nalézt, kde v RAM je každý jednotlivý znak umístěn. Jen nezapomeňte, že každý znak zaujímá 8 buněk paměti. Zde je několik příkladů, abyste věděli, jak začít.

Znak	Kód znaku	Počáteční adresa v RAM
a	0	12 288
A	1	12 296
!	33	12 552
)	62	12 784

Nezapomeňte, že jsme zkopírovali pouze prvních 64 znaků. Něco jiného nastane, budeme-li chtít jeden z dalších znaků. Co se stane, budete-li chtít znak s kódem 154, reverzované Z? No, mohli byste to udělat buď reverzováním znaku Z, nebo byste mohli zkopírovat z ROM generátoru soubor reverzovaných znaků, nebo použít z ROM generátoru pouze jeden znak a přemístit jej na místo znaku, který máte v RAM a nepotřebujete jej.

Předpokládejme, že jste se rozhodli, že nepotřebujete znak). Teď umístěme na místo tohoto znaku data reverzovaného Z.
Zadejte:

```
FOR I = 0 TO 7 : POKE 12784 + I, 255 – PEEK(I+12496) : NEXT
```

Nyní stiskněte klávesu znaku). Zobrazí se jako reverzované Z. Nehledě na to, kolikrát stisknete tuto klávesu, vždy se bude zobrazovat jako reverzované Z. Ačkoliv znak nyní vypadá jako reverzované Z, v programu vždy vystupuje jako). Vyzkoušejte nějaký příkaz, ve kterém potřebujete znak). Zadání bude pracovat správně, jenom bude divně vypadat.

Rychlé zopakování

Nyní umíte zkopírovat znaky z ROM do RAM. Můžete dokonce najít a pozměnit libovolný znak. Teď už zbývá pouze jeden krok v programování znaků (ten nejlepší)... vytváření vlastních znaků. Vzpomeňte si, jak jsou znaky uloženy v ROM. Každý znak je uložen jako skupina 8 bajtů. Jednotlivé bity pak přímo ovládají tečky, z nichž je znak složen. Pokud je bit 1, je nahrazena tečka na dané lokaci, pokud je 0, nezobrazuje se. Zadejte NEW a následující program:

```
10 FOR I = 12 448 TO 12 455: READ A: POKE I,A: NEXT
20 DATA 60,66,165,129,165,153,66,60
```

Nyní program pomocí RUN stiskněte. Program zamění znak T za "usmívající se tvář". Stiskněte několikrát T, abyste ji uviděli. Každé číslo v příkazu DATA na řádku 20 reprezentuje jednu řadu "usmívající se tváře". Matice "tváře" vypadá takto:

		7 6 5 4 3 2 1 0	BINARNE	DEKADICKY
řada	0	x x x x	00111100	60
	1	x x	01000010	66
	2	x x x x	10100101	165
	3	x x	10000001	129
	4	x x x x	10100101	165
	5	x x x x	10011001	153
	6	x x	01000010	66
	7	x x x x	00111100	60

Nahraďte čísla v příkaze DATA na řádku 20 vlastními čísly, která jste vypočítali a spusťte program pomocí RUN. Pak stiskněte T. Při každém stisku uvidíte svůj znak.

Upozornění: Pro dosažení nejlepších výsledků vždy navrhujte svislé linie ve Vašem znaku přinejmenším 2 tečky široké. To napomůže předcházet chromatickému šumu (barevnému rušení) ve Vašem znaku, když je zobrazován na TV obrazovce.

Na str. 114 se nalézá program užívající programovatelných znaků.

```
10 REM EXAMPLE 1
20 REM CREATING PROGRAMABLE CHARACTERS
31  POKE  56  334, PEEK(56344)AND 254: POKE 1,  PEEK  (1)AND
   251: REM TURN OFF KB AND I/O
35  FOR  I = 0 TO 63: REM CHARACTER RANGE TO BE COPIED  FROM
   ROM
36 FOR J = 0 TO 7: REM COPY ALL 8 BYTES PER CHARACTER
37 POKE 12 288 + J, PEEK (53248+I  8 + J): REM COPY A BYTE
38 NEXT J: NEXT I: REM GO TO NEXT BYTE OR CHARACTER
39  POKE 1, PEEK(1)OR 4: POKE 56 334, PEEK (56 334)OR 1: REM
   TURN ON I/O AND KB
40  POKE  53  272,(PEEK(53272)AND  240)+12:  REM  SET  CHAR
   POINTER TO MEM. 12 288
```



```
60 FOR CHAR = 60 TO 63: REM PROGRAM CHARACTERS 60 THRU 63
80 FOR BYTE = 0 TO 7: REM DO ALL 8 BYTES OF A CHARACTER
100 READ NUMBER: REM READ IN 1/8TH OF CHARACTER DATA
120  POKE  12  288 +(8  CHAR)+ BYTE, NUMBER: REM  STORE  THE
   DATA IN MEMORY
140 NEXT BYTE: NEXT CHAR: REM ALSO COULD BE NEXT BYTE, CHAR
150 PRINT CHR$ (147) TAB(255) CHR$(60);
155 PRINT CHR$(61) TAB(55) CHR$(62) CHR$(63)
160  REM  RADEK  150  ZOBRAZUJE  NOVE  DEFINOVANE  ZNAKY  NA
   OBRAZOVKU
170 GET A$: REM CEKEJ AZ UZIVATEL STISKNE KLAVESU
180  IF  A$  = ""THEN GO TO 170:REM POKUD  NEBYLA  STISKNUTA
   ZADNA KLAVESA, ZKUS TO OPET
200 DATA 4,6,7,5,7,7,3,3: REM DATA PRO ZNAK 60
190 POKE 53272,21: REM RETURN TO NORMAL CHARACTERS
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 PRO ZNAK
   63
240 END
```

Znakový kód MULTICOLOR

Standartní jemná grafika Vám dává možnost ovládat velmi malé plošky na obrazovce. Každá tečka v paměti znaků může mít 2 možné hodnoty - 1 a 0. Pokud je tečka vypnutá, je pro místo, které je pro tečku vyhrazeno, použita barva obrazovky - pozadí. Pokud je tečka zapnutá, je zbarvena barvou, která je dána v paměti barvy pro znak na dané pozici. Když užíváte standardní jemnou grafiku, mohou všechny tečky pro 8x8 znak nabývat buď barvy pozadí nebo barvy popředí. To určitým způsobem omezuje barevné vyjádření v tomto prostoru. Např. se může vyskytnout problém, když se kříží dvě různé různobarevné linie.

Mód multicolor Vám umožňuje řešit tento problém. Každá tečka v režimu multicolor může nabývat čtyř barev: barvu obrazovky (pozadí - registr =0), barvu pozadí v registru = 1, barvu pozadí v registru = 2, nebo barvu znaku.

Obětujeme tím však rozlišovací schopnost ve vodorovném směru, protože v modu multicolor je každá tečka 2x širší než ve standardním režimu jemné grafiky. Tato ztráta rozlišovací schopnosti je však více než kompenzována zvláštními možnostmi modu multicolor.

BIT MODU MULTICOLOR

Pro zpanutí znakového modu multicolor nastavte bit 4 řídícího registru čipu VIC II na adrese 53 270 ($D016) na

1: POKE 53 270, PEEK (53 270) OR 16

Vypnutí se provádí nastavením 4. bitu na lokaci 53 270 na

0: POKE 53 270, PEEK (53 270) AND 239

Mod multicolor se nastavuje pro každou položku (matici) 8x8 zvlášť a tak může být multicolor grafika směšována s normální jemnou grafikou. To zajišťuje 3. bit paměti barvy. Paměť barvy začíná na adrese 55 296 ($D800). Pokud je číslo v paměti barvy menší než 8 (0 7), odpovídající matice bude ve standardní jemné grafice. Pokud číslo v paměti barvy je větší nebo rovné 8 (od 8 15), pak bude matice zobrazována v multicolor modu.

POKEováním čísel do paměti barvy můžeme měnit barvu znaku v daném místě obrazovky. POKEujeme-li číslo od 0 do 7, volíme barvu normálního znaku. POKEujeme-li číslo od 8 do 15, volíme barvu matice v modu multicolor. Jinými slovy, nastavujeme MULTICOLOR mod, nastavíme-li 3. bit na nulu, nastavujeme normální jemnou grafiku.

V matici, která byla nastavena pro mod MULTICOLOR, bity určují, kterou barvou budou tečky zobrazeny. Např. zde je obrázek písmene A a jeho bitové vyjádření:

Obrázek	Bitové vyjádření
. .	00011000
. . . .	00111100
. . . .	01100110
. . . . .	01111110
. . . .	01100110
. . . .	01100110
. . . .	01100110
	00000000

V normálním znakovém modu nebo jemné grafice bude zobrazovaná barva obrazovky všude tam, kde jsou ve znaku bity nulové a barvou znaku budou zobrazeny tečky, jejichž bity jsou jednotkové. Multicolor mod užívá bity vždy v párech:

Obrázek	Bitové vyjádření
AABB	00 01 10 00
CCCC	00 11 11 00
AABBAABB	01 10 01 10
AACCCCBB	01 11 11 10
AABBAABB	01 10 01 10
AABBAABB	01 10 01 10
AABBAABB	01 10 01 10
	00 00 00 00

Bitový pár	barvový registr	lokace	
00	background = 0 (barva obraz.)	53 281	$D021
01	background = 1	53 282	$D022
10	background = 2	53 283	$D023
11	barva specifikovaná spodními 3 bity	COLOR RAM	

Pozn.: BARVA popředí SKRITKA je 10
BARVA popředí ZNAKU je 11

Zadejte NEW a následující ukázkový program:

```
100  POKE 53 281,1: REM NASTAV BARVU POZADI # 0  (OBRAZOVKY)
   NA BILOU
110 POKE 53 282,3: REM NASTAV BARVU POZADI # 1 NA CYAN
120 POKE 53 283,8: REM NASTAV BARVU POZADI # 2 NA ORANZOVOU
130  POKE  53  270, PEEK(53 270) OR 16: REM PREPNI  DO  MODU
   MULTI COLOR
140  C=13*  4096  + 8 * 256: REM NASTAV C ABY  UKAZOVALO  DO
   PAMETI BARVY
150 PRINT CHR$(147) "AAAAAAAAAA"
160 FOR L=0 TO 9
170 POKE C+L,8: REM POUZIJ MULTI-CERNOU
180 NEXT
```

Barva obrazovky je bílá, barva znaku je černá, jeden barevný registr je zelenomodrý, druhý oranžový. Do matice tedy ukládáte kod barvy, ale ve skutečnosti užíváte odkazu na registry nesoucí informaci o těchto barvách. To šetří paměť, neboť 2 bity zachycují 16 barev pro jednotlivá pozadí a 8 barev pro znak. To také umožňuje pěkné triky. Jednoduchá změna jednoho z nepřímých regostrů změní všechny tečky kreslené touto barvou. Zde je příklad, jak měnit barvu pozadí #1:

```
100  POKE  53  270, PEEK (53270) OR 16: REM PREPNI  DO  MODU
   MULTICOLOR
110 PRINT CHR$ (147) CHR$ (18);
120 PRINT "(2 srdce)";:REM ZADEJ C= 1 PRO ORANZOVOU NEBO
   C = 1 MULTI CERNOU
130 FOR L=1 TO 22: PRINT CHR$(65);: NEXT
135 FOR T=1 TO 500: NEXT
140  PRINT  "2šipky";:REM ZADEJ CTRL 7 PRO ZMENU  NA  MODROU
   BARVU CTRL 7
145 FOR T=1 TO 500: NEXT
150 PRINT "čtverečekHIT a KEY" CTRL 1
160 GET A$: IF A$ = ""THEN 160
170 X=INT (RND(1)*16)
180 POKE 53 282, X
190 GOTO 160
```

Užitím C= klávesy a klávesy pro barvu může být měněna barva znaku na jakoukoliv jinou vč. modu multicolor. Např. zadejte:
POKE 53270,PEEK(53270) OR 16: PRINT "znak";:REM SVETLE CERVENA/MULTICERVENA CTRL 3

Slovo READY a cokoliv napíšete bude zobrazeno v modu multicolor. Jakékoliv nastavení barvy nás může nastavit zpět do normálního textového modu.

Na str. 119 je příklad užívající programovatelných znaků v režimu MULTICOLOR:

```
10 REM*EXAMPLE 2 *
20 REM CREATING MULTI COLOR PROGRAMMABLE CHARACTERS
31 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)AND251
35 FORI=0TO63:REM CHARACTER RANGE TO BE COPIED FROM ROM
```



```
36 FORJ=0TO7:REM COPY ALL 8 BYTES PER CHARACTER
37 POKE12288+I*8+J,PEEK(53248+I*8+J),:REM COPY A BYTE
38 NEXTJ,I:REM GOTO NEXT BYTE OR CHARACTER
39  POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1:REM   TURN  ON
   I/O AND KB
40  POKE53272,(PEEK(53272)AND240)+12:REM SET CHAR POINTER TO
   MEM. 12288
50 POKE53270,PEEK(53270)OR16
51 POKE53281,0:REM SET BACKGROUND COLOR #0 TO BLACK
52 POKE53282,2:REM SET BACKGROUND COLOR #1 RO RED
53 POKE53283,7:REM SET BACKGROUND COLOR #2 TO YELLOW
60 FORCHAR=60TO63:REM PROGRAM CHARACTERS 60 THRU 63
80 FORBYTE=0TO7:REM DO ALL 8 BYTES OF A CHARACTER
100 READNUMBER:REM READ 1/8TH OF CHARACTER DATA
120  POKE12288+(8*CHAR(+BYTE,NUMBER:REM  STORE THE  DATA  IN
   MEMORY
140 NEXT BYTE,CHAR
150 PRINT"srdce" TAB(255)CHR$(61)TAB(55)CHR$(62)CHR$(63)
160  REM  LINE 150 PUTS THE NEWLY DEFINED CHARACTERS ON  THE
   SCREEN
170 GETA$:REM WAIT FOR USER TO PRESS A KEY
180 IFA$=""THEN170 REM IF NO KEYS WERE PRESSED, TRY AGAIN
190  POKE532272,21:POKE53270,PEEK(53270)AND239:REM   RETURNE
   TO NORMAL CHARACTERS
200 DATA129,37,21,29,93,85,85:REM DAYTA FOR CHARACTER 60
210 DATA66,72,34,116,117,85,95,85:REM DATA FOR CHARACTER 61
220 DATA97,87,85,21,8,8,40,0:REM DATA FOR CHARACTER 663
230 DATA212,213,85,84,32,22,40,0:REM DATA FOR 240 END
```

Znakový mód s rozšířením barev pozadí

Mód s rozšířením barev pozadí nám dává možnost ovládat barvu pozadí každého jednotlivého znaku, zrovna tak jako barvu popředí. V tomto modu můžete např. zobrazit modrý znak na žlutém pozadí a při bílé barvě obrazovky. Pro mod s rozšířením barev pozadí máme k dispozici 4 registry, z nichž každý může být nastaven na jednu z 16 barev.

Paměť barvy nese informaci o popředí znaku tak, jak to je ve standardním znakovém modu.

Tento mód však omezuje množství různých znaků, které můžete zobrazit. Pokud je zvolen tento mod, může být užito pouze prvních 64 znaků z generátoru znaků ROM (nebo prvních 64 znaků, které si naprogramujete). To proto, že dva bity z kódu znaku jsou užity pro volbu barvy pozadí. To můžeme ilustrovat na následujícím příkladu.

Kód znaku (číslo, které byste měli POKEovat, aby se objevilo na obrazovce) písmene "A" je 1. Pokud je aktivován mod s rozšířením barev pozadí a POKEujete 1, na obrazovce by se mělo objevit "A". Pokud POKEujete 65, normálně byste na obrazovce očekávali znak s tímto kódem. To se však nestane u modu s rozšířením barev pozadí. Místo toho se objeví to samé A jako předtím, ale s odlišnou barvou pozadí.

Následující tabulka zachycuje kódy pro pozadí:

Kod znaku			Registr barvy pozadí	
Rozmezí	Bit 7	Bit 6	Cislo	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)

Mod s rozšířením barev pozadí zapneme nastavením bitu 6 registru VIC-II na 1 na adrese 53265($D011) např. pomocí následujícího příkazu:

```
POKE 53265,PEEK(53265)OR64
```

Tento mod vypneme nastavením bitu 6 na nulu u registru VIC-II na adrese 53265($D011) tak, jak to dělá následující příkaz:

```
POKE 53265? PEEK(53265) AND 191
```

Jemná grafika

Při psaní her, kreslení grafů a tabulek pro obchodní aplikace a další typy programů, dříve či později zjistíte, že potřebujete zobrazení s vyšší rozlišovací schopností.

COMMODORE 64 byl právě pro tento účel navržen. Jemná grafika je možná díky tomu, že obrazovka může snímat informaci z bitové mapy. Bitová mapa je metoda, ve které má každý možný bod (pixela) na obrazovce přiřazen svůj vlastní bit v paměti. Pokud je bit v paměti jedničkový bod, je zobrazován, pokud je bit v paměti nulový, zobrazován není.

C64 má u jemné grafiky i nedostatky. Předně, každá pixela musí mít v paměti svůj vlastní bit a tak se spotřebovává spousta paměti. Protože každý znak má 8x8 pixel a na obrazovce můžeme mít 25 řádků se 40 znaky na každém řádku, ropozlišitelnost je 320 pixel (bodů)krát 200 pixel pro celou obrazovku. To dává 64.000 různých pixel, z nichž každá potřebuje 1 bit paměti. Jinými slovy, na zmapování celé obrazovky je třeba 8.000 bitů.

Obecně jsou operace s jemnou grafikou převážně krátké, jednoduché a hlavně často se opakující rutiny. Na neštěstí tento typ rutin je obvykle poněkud pomalý, pokud je zapíšeme v BASICu. Avšak krátké, jednoduché a opakující se rutiny je přesně to, k čemu se nejlépe hodí strojní jazyk. Rešení je tedy buď napsat program přímo ve strojovém jazyce, nebo vyvolat rutiny pro práci s jemnou grafikou z BASICového programu pomocí příkazu SYS. Touto cestou dosáhneme obojího jednoduchý zápis v BASICu a rychlosti strojového jazyka pro grafiku. Lze také zakoupit VSP kartridž s BASICovými příkazy pro jemnou grafiku.

Všechny příklady uvedené v této sekci, budou v BASICu, aby byly jasné. Nyní technické detaily.

BITOVÁ MAPA je jednou z nejpopulárnějších grafických technik ve světě počítačů. Je užívána k vytváření vysoce detailních obrázků. V podstatě, když C64 přechází do modu BITOVÉ MAPY, přímo zobrazuje 8K paměti na TV-obrazovce a můžete přímo ovládat, zda jednotlivé body na obrazovce budou, či nikoliv.

Existují dva typy bitové mapy u C64:

1) Standardní (jemná grafika) bitová mapy (320x200 bodů)
2) MULTICOLOR bitová mapa (160x200 bodů). Oba typy jsou velmi podobné módům znakovým s podobnými názvy - stand. typ má vyšší rozlišovací schopnost a možnosti, ale menší výběr barev. Na druhé straně MULTICOLOR bitová mapa na účet snížení rozlišitelnosti v horizontálním směru na polovinu, umožňuje použití většího množství barev v matici 8 x 8 bodů.

Každá adresa v paměti obrazovky, kterou jsme užívali k uložení informace o tom, které znaky budou zobrazeny nese nyní informace o barvě. Např. místo toho, aby se při POKEování 1 na adresu 1024 zobrazilo "A" v levém horním rohu obrazovky, obsah buňky 1024 nyní ovládá barvy bitů v této matici 8x8 bodů v levém horním rohu.

Barva matice v modu bitové mapy nepochází nyní z paměti barvy, jak to bylo v znakových modech. Místo toho nese barvovou informaci pamat obrazovky. Horní 4 bity paměti obrazovky nesou barvu kteréhokoliv bitu nastaveného na jedničku v mozaice 8x8 odpovídající polohou adrese v paměti obrazovky. Nižší 4 bity ovládají barvu jakékoliv bitu nastaveného na 0.
Příklad: Zadejte následující

```
5 BASE = 2*4096:POKE 53272,PEEK(53272)OR8:REM UMISTI
  BITOVOU MAPU NA 8192
10 POKE 53265,PEEK(53265)OR32:REM PREPNI DO MODU BITOVE
  MAPY
```

Nyní pomocí RUN spustte program.
Na obrazovce se objevil zmatek, že? Stejně tak jako v normálním znakovém modu, musíte vymazat obrazovku jemné grafiky předtím, než ji použijete. Naneštěstí, pouhé zadání příkazu CLR v tomto případě nepomůže. Místo toho musíte vymazat oblast paměti, kterou užíváte pro Vaše programovatelné znaky. Stiskněte RUN/STOP a RESTOR klávesy současně a pak připojte k programu následující řádky, které způsobí vymazání obrazovky pro jemnou grafiku.

```
20 FOR I=BASE TO BASE+7999:POKEI,0:NEXT:REM VYMAZ
  BITOVOU MAPU
30 FOR I=1024 TO 2023:POKEI,3:NEXT:REM NASTAV BARVU NA
  CYAN A CERNOU
```

Nyní opět spustte program. Měli byste vidět, jak se obrazovka maže a pak zbarvuje zelenomodrou (cyan) barvou. Co budete chtít dělat nyní, je zapínat a vypínat tečky na obrazovce pro jemnou grafiku. Pro zapnutí a vypnutí tečky musíte vědět, jak najít správný bit v bitové mapě, který chcete zapnout. Na to potřebujete vzorec.

Užijeme proměnných X,Y k určení vodorovné a svislé polohy tečky. Tečka, mající X=0 i Y=0 se nalézá v levém horním rohu obrazovky. Tečky více vpravo od ní mají vyšší x-ovou hodnotu, tečky blíže ke spodku obrazovky mají vyšší y-ovou hodnotu. Nejlepší způsob, jak užívat bitovou mapu, je upravit si bitovou mapu podle následujícího obrázku

```
0.....X.....319
.                .
.                .
.                .
Y                .
.                .
.                .
.                .
199...............
```

Každá tečka bude mít X-ovou a Y-ovou souřadnici. V tomto případě je jednoduché ovládat jakoukoliv tečku na obrazovce. Avšak ve skutečnosti jsou dat uspořádána takto:

```
------ Byte 0   Byte 8   Byte 16  Byte 24 ..... Byte 312
Vrchní Byte 1   Byte 9   .       .           Byte 313
       Byte 2   Byte 10  .       .           Byte 314
řada   Byte 3   Byte 11  .       .           Byte 315
       Byte 4   Byte 12  .       .           Byte 316
(ř.0)  Byte 5   Byte 13  .       .           Byte 317
       Byte 6   Byte 14  .       .           Byte 318
------ Byte 7   Byte 15  .       .           Byte 319
-------------------------------------------------------
------ Byte 320Byte328 Byte 336 Byte 344 ..... Byte 632
Druhá  Byte 321Byte329  .       .           Byte 633
       Byte 322Byte330  .       .           Byte 634
řada   Byte 323Byte331  .       .           Byte 635
       Byte 324Byte332  .       .           Byte 636
(ř.1)  Byte 325Byte333  .       .           Byte 637
       Byte 326Byte334  .       .           Byte 638
------ Byte 327Byte335  .       .           Byte 639
```

Programovatelné znaky, které vytvářejí bitovou mapu, jsou uspořádány ve 25 řádcích a 40 sloupcích. Zatímco je to dobrá metoda pro uspořádání textu, v bitové mapě je to velmi obtížné.

(Pro toto řešení je však velmi dobrá důvod - viz oddíl "Smíšené mody").

Následující vzorec zjednoduší ovládání teček v bitové mapě.

Číslo řady (od 0 do 24), ve které leží naše tečka, je:
ROE = INT(4/8) (je 320 bytů v jedné řadě)
Pozice znaku v dané řadě (od 0 do 39) je:
CHAR = INT(X/8) (pro jeden znak je 8 bytů)
Linka ve znaku, na které je požadovaná tečka (od 0 do 7)
LINE = Y AND 7
Bit v daném bytu nalezneme podle:
BIT = 7 - (X AND 7)

Nyní složíme tyto vzorce dohromady. Byt, na kterém je tečka v daných souřadnicích X, Y vypočteme:

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

Pro zpanutí Jakéhokoliv bitu v síti X, Y užijeme tento řádek:

POKE BYTE, PEEK(BYTE) OR 2+BIT

Nyní použijeme tyto výpočty v programu. Následující příklad vykreslí na obrazovku graf sinusové křivky:

```
50 FOR X=0 TO 319 STEP.5:REM VLNA VYPLNI OBRAZOVKU
60 Y=INT(90+80*sin(X/10))
70 CH=INT(X/8)
80 RO=INT(Y/8)
85 LN=Y AND 7
90 BY=BASE+RO*320+CH*8+LN
100 BI=7-(XAND7)
110 POKE BY,PEEK(BY)OR(2BI)
120 NEXT X
125 POKE 1024,16
130 GO TO 130
```

Výpočet na řádce 60 změní hodnoty sinové funkce z rozmezí (+1,-1) do rozmezí (10,170). Rádky 70 až 100 vypočítávají znak, řadu, byt a bit, které mají být měněny, užívajíce výúpočtů, které jsme si ukázali dříve. Rádek 125 signalizuje, že program skončil tím, že změní barvu matice v levém horním rohu obrazovky. Rádek 130 zmrazí program tím, že jej zařadí do cyklu nekonečné smyčky. Když už jste se dost vynadívali na obrázek, podržte jen RUN/STOP a stiskněte RESTORE.

V dalším příkladu můžete modifikovat program pro zobrazení sinusovky tak, aby zobrazoval polokružnici. Zde jsou řádky, které provedou potřebné změny v programu:

```
50  FOR X=0 TO 160:REM PRACUJ V POLOVINE OBRAZOVKY
55  Y1=100+SQR(160*X-X*X)
56  Y2=100-SQR(160*X-X*X)
60  FOR Y=Y1 TO Y2 STEP Y1-Y2
70  CH=INT(X/8)
80  RO=INT(Y/8)
85  LN=Y AND 7
90  BY=BASE+RO*320+CH*8+LN
100 BI=7-(X AND 7)
110 POKE BY,PEEK(BY)OR(2 /šipka nahoru/ BI)
114 NEXT
```

Program vytvoří polokružnice na obrazovce jemné grafiky.

Upozornění: BASICové proměnné se mohou dostat až do paměti obrazovky jemné grafiky. Pokud potřebujete více pámětového prostoru, musíte přemístit začátek BASICu nad paměť bitové mapy. Nebo musíte přemístit paměť bitové mapy. Tento problém se nevyskytne při používání strojového jazyka. Může se přihodit pouze při psaní programů v BASICu.

Bitová mapa MULTICOLOR

Zrovna tak jako multicolor mod, umožní Vám mod multicolor bitové mapy zobrazení čtyřmi různými barvami v každé matici 8x8 bodů bitové mapy. A stejně jako ve znakovém modu multicolor opětujeme vodorovné rozlišení z 320 na 160 bodů.

Bitová mapa multicolor užívá 8k paměti. V tomto režimu si můžete barvu vybírat:

1) z registru pozadí#0 (barva pozadí obrazovky)
2) z videomatice - 4 horní bity určují jednu barvu
   4 spodní bity určují druhou barvu
3) z paměti barvy

Bitová mapa multicolor se aktivuje nastavením bitu 5 na adrese 53265 ($D011) a bitu 4 na adrese 53270 ($D016) do jedničky. Např. pomocí následujícího POKE:

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

Vypnutí se provádí nastavením bitu 5 na adrese 53265 ($D011) a bitu 4 na adrese 53270 ($D016) na 0.

POKE 53265,PEEK (53265)AND 223: POKE 53270,PEEK (53270)AND 239

Tak jako ve standardní bitové mapě, je i zde vztah mezi 8k paměti užívanou k zobrazení informací na obrazovku. Nyní jsou však tečky ve vodorovném směru 2x širší. Každá dvojice bitů v paměti 8k pak představuje tečku, která může nabývat jednu ze 4 barev.

Bity	Barvová informace pochází z:
00	background color#0 (barva obrazovky)
01	horní 4 bity paměti obrazovky
10	spodní 4 bity paměti obrazovky
11	barvový nybl (nybl-1/2 bytu=4 bity) z paměti barvy

Jemné skrolování

Čip VIC II nabízí jemné skrolování v obou směrech - horizontálním i vertikálním. Jemné skrolování je pohyb celé obrazovky o šířku jedné pixely jedním směrem. Pohyb se může dít buď nahoru, dolů, doleva nebo doprava. Užívá se k jemnému zavedení nových informací na obrazovku, zatímco znaky na druhé straně obrazovky mizí.

Ačkoliv čip VIC II za nás dělá spoustu práce, skutečné skrolování musí být ovládáno programem ve strojním jazyce. Rysem čipu VIC II je schopnost umístit zobrazované informace v 8 vodorovných pozicích a 8 svislých pozicích. Nastavení pozice je ovládáno VIC II skrolovácím registrem. VIC II čip má také možnost 38 sloupcového a 24 řádkového modu. Menší velikost obrazového pole nám dává možnost ukládat nová data, která mají být skrolována.

Pro jemné skrolování se musíte řídit následujícími kroky:

1) Zmenšit zobrazované pole (ohraničení se rozšíří)
2) Nastavit skrolovací registr na maximum (minimum) podle toho, v jakém směru chceme skrolovat.

3) Umístit nová data do oblasti bitové mapy, která není zobrazována
4) Inkrementovat (dekrementovat) skrolovací registr až dosáhneme max. (min.) hodnoty
5) V tomto místě použijte Vaši rutiny ve strojním jazyce k posunutí celé obrazovky ve směru skrolování
6) Vratte se na krok 2

Abychom mohli přejít do 38 sloupcového režimu, bit 3 na adrese 53270 ($D016) musí být nastaven na 0. To dělá následující příkaz:

POKE 53270?PEEK(53270)AND 247

K navrácení do 40sloupcového režimu musíme bit 3 na adrese 53270 ($D016) nastavit do 1 jako následující příkaz:

POKE 53270,PEEK(53270)OR 8

Abychom přešli do 24řádkového modu, bit 3 na adrese 53265 ($D011) musí být nastaven na nulu. To provádí následující příkaz:

POKE 53265,PEEK(53265)AND 247

K navrácení do 25řádkového modu musíme nastavit bit 3 na adrese 53265 ($D011) na jedničku tak, jak to provádí následující příkaz:

POKE 23265, PEEK (53265)OR 8

Skrolujeme-li ve směru X, je nutné uvést VIC II do 38 sloupcového režimu. Tak se vytváří místo pro nová data, která mají být skrolována.

Při skrolování doleva by měla být nová data umístěna vždy nalevo. Při skrolování doprava by měla být nová data umístěna vždy vpravo.Všimněte si, že stále zůstává 40 sloupců paměti, ale pouze 38 je viditelných.

Při skrolování v Y-novém směru je nezbytné uvést VIC II do 24řádkového režimu. Při skrolování nahoru umístěte nová data do posledního řádku. Při skrolování dalů umístěte nová data do prvního řádku. Narozdíl od skrolování v X-ovém směru, kdy je překryta plocha po obou stranách obrazovky, při skrolování v Y-ovém směru je překryta jen plocha po jedné straně obrazovky. Pokud je skrolovací Y-ový registr nastaven na 0, je překryta první řádka a připravena pro nová data. Je-li skrolovací Y-ový registr nastaven na 7, je překryta poslední řádka.

Při skrolování v X-ovém směru je skrolovací registr umístěn v bitech 0 až 2 v řídícím registru VIC II na adrese 53270 ($D016). Jako vždy je důležité měnit pouze tyto bity. Následující POKE to tak činí:

```
POKE 53270,(PEEK(53270)AND 248)+X
```

kde X je X-ová pozice obrazovky od 0 do 7
Pro skrolování v Y-ovém směru je skrolovací registr umístěn v bitech 0 až 2 v řídícím registru VIC II na adrese 53265 ($D011). Obdobným způsobem můžeme měnit jeho bity:

```
POKE 53265,(PEEK(53265)AND 248)+Y
```

kde Y je Y-ová pozice obrazovky od 0 do 7.

Ke skrolování textu na obrazovce odspodu byste měli krokovat 3 nejnižší bity na lokaci 53265 od 0 do 7, vložit data do zakrytého řádku na spodu obrazovky a pak proce opakovat.

Ke skrolování znaků na obrazovce zleva doprava byste měli krokovat 3 nejnižší bity na lokaci 53270 od 0 do 7, natisknout či naPOKEovat další sloupec nových dat do sloupce 0 obrazovky a pak proces zopakovat.

Pokud krok skrolovacích bitů bude -1, bude se text pohybovat opačným směrem.

Příklad:

Text skrolující se od spodu obrazovky:

```
10 POKE 53265,PEEK (53265)AND 247: REM PREJDI DO 24-RADKOVEHO REZIMU
20 PRINT CHR$ (147):REM SMAZ OBRAZOVKU
30 FOR X=1 TO 24: PRINT CHR$ (17);:NEXT: REM PREMISTI KURZOR NASPOD OBRAZOVKY
40 POKE 23265,(PEEK(53265)AND 248)+7:PRINT:REM POZICE PRO 1.SKROLOVANI
50 PRINT "HELLO"
60 FOR P=6 TO 0 STEP-1
70 POKE 53265,(PEEK(53265)AND 248)+P
80 FOR X=1 TO 50:NEXT:REM CASOVE ZPOZDENI
90 NEXT: GOTO 40
```


S P R I T Y

Sprite je s pecielní typ uživatelsky definovaného znaku, který může být zobrazen kdekoliv na obrazovce. Sprity jsou přímo podporovány čipem VIC II. Uživatel může o každém spritu definovat: "jak bude vypadat", "v jaké barvě bude", "kde se zobrazí". Sprite se může zobrazit v některé ye 16 barev. Sprity se mohou použít v jakémkoliv modu - bitovém, znakovém, multi-color atd. a vždy mají svůj tvar. Sprity si zachovávají svoji vlastní barvu, mod (HIRES nebo MULTI-COLOR) a svůj tvar.

Čip VIC II automaticky současně podporuje 8 spritů. Zobrazení více než 8 spritů je umožněno RASTER INTERPUT technikou.
Vlastnosti spritu:

1. velikost je 24 horizontálních bodů x 21 vertikálních bodů
2. individuální barva každého spritu
3. MULTICOLOR sprity
4. zvětšení (2x) v horizontálním, vertikálním nebo obou směrech
5 výběr priority spritu k pozadí
6. pevná priorita spritu k druhému spritu
7. detekce kolize spritu
8. detekce kolize spritu a pozadí

Tyto speciální vlastnosti spritu umožňují jednoduché programování mnoha různých her. Protože sprity jsou podporovány hardwarem, je možné napsat kvalitní hry i v BASICu.

Protože 8 spritů je přímo řízeno čipem VIC II, jsou očíslovány 0-7. Každý sprite má svoje definované umístění, registry pozice a barvy a svoje bity pro detekci kolize.

Ukazatelé spritů

Každý sprite je definován 63 byty a jedním bytem umístěným na konci spritu. Sprite se pak skládá ze 64 bytů. To usnědňuje výpočet adresy paměti pro umístění definice spritu.

Každý z 8 spritů má svůj byt, který se jmenuje SPRITE POINTER (směrník, ukazatel). Sprite pointer uakzuje, kde je v paměti umístěna definice spritu. Těchto 8 bytů je vždy umístěno v posledních 8 bytech 1KB části paměti obrazovky. Obvykle v C64 je tato paměť umístěna na adrese 2040 ($07F8)hex). Jestliže přesouváte paměť obrazovky, pak umístění sprite pointeru se musí přesunout také.

Každý sprite pointer může obsahovat číslo mezi 0 a 255. Toto číslo ukazuje na definici daného spritu. Protože každá definice spritu má 64 bitů, to znamená, že pointer může "vidět" kdekoliv v 16k bloku paměti, ke které má VIC II přístup (protože 256x64=16k).

Např. jestliže sprite pointer 0 na adrese 2040 obsahuje číslo 14 to znamená, že sprite 0 bude zobrazen použitím 64 bitů začínajících na adrese 64x14=896, což je buffer pro magnetofon (datasette).
Následující vzorec vše objasňuje:

adresa=(BANK*16384)+(obsah sprite pointeru*64)

kde BANK je 16k segment paměti, který obhospodařuje VIC II a může nabývat hodnot od 0 do 3.

Vzorec udává počátek 64 bytového definičního bloku spritu. Když VIC II obhospodařuje BANK 0 nebo BANK 2, nastavení znaku ROM IMAGE je přítomno v jisté oblasti, jak bylo uvedeno dříve. Pak zde nemůže být umístěna definice spritu. Jestliže z nějakých důvodů potřebujete více než 128 definicí spritu, pak použijte bank bez IMAGE ROM (1 nebo 3).

Zapnutí spritu

Řídící registr na adrese 53265 ($D015 hex) je znám jako SPRITE ENABLE registr. Každému spritu odpovídá jeden bit v tomto registru, který rozhoduje, zda sprite je ON nebo OFF. Registr vypadá následovně:

$D015 7 6 5 4 3 2 1 0

Např. zapnutí spritu 1 znamená nastavit bit 1. Lze to provést následujícím příkazem POKE:

```
POKE 53269,PEEK(53269) OR 2
```


B a r v y

Sprity mohou mít některou ze 16 barev, které jsou generované čipem VIC II. Každý sprite má vlastní registr pro barvy. Adresy ukazuje následující tabulka:

ADRESY		POPIS
53287	($D027)	registr barvy spritu 0
53288	($D028)	registr barvy spritu 1
53289	($D029)	registr barvy spritu 2
53290	($D02A)	registr barvy spritu 3
53291	($D02B)	registr barvy spritu 4
53292	($D02C)	registr barvy spritu 5
53293	($D02D)	registr barvy spritu 6
53294	($D02E)	registr barvy spritu 7

Všechny body spritu jsou zobrazeny v barvě, jenž je definována v odpovídajícím registru pro barvu.

Multicolor mód

Multi-color mód umožňuje použít 4 různé barvy pro každý sprite. Avšak počet bodů v horizontálním směru je poloviční. Jinými slovy, pracujete-li v multi-color modu, pak se místo 24 bodů ve spritu používá 12 párů bodů. Každý pár bodů se nazývá bitový pár (BIT PAIR). Následující tabulka udává hodnoty bitových párů, které jsou potřebné pro zapnutí každé ze 4 zvolených barev.

BITOVÝ PÁR	POPIS	
00	"PRUHLEDNY", barva obrazovky	
01	SPRITE MULTI-COLOR REGISTR #0 (53285)	($D025)
10	SPRITE COLOR REGISTR	
11	SPRITE MULTI-COLOR REGISTR #1 (53286)	($D026)

Poznámka: Barva popředí spritu je 10. Pozadí znaku je 10.

Nastavení spritu do Multi-color modu

K přepnutí spritu do MULTI-COLOR modu se musí zapnout řídící rregistr na adrese 53276 ($D01C). Provádí se následující POKE:

POKE 53276,PEEK (53276) OR (2šipka nahoruSN)

kde SN je číslo spritu (0 až 7).

Vypnutí spritu z MULTI-COLOR modu se provede vynulováním řídícího POKE 53276,PEEK (53276) AND (255-2 šipka nahoru SN), kde SN je číslo spritu (0 až 7).

Rozšíření spritu

Čip VIC II má možnost rozšířit sprite ve vertikálním horizontálním nebo obou směrech. Při rozšíření spritu je každý jeho bod dvakrát širší nebo dvakrát vyšší. Při rozšíření spritu v horizontálním směru musí být zpanut odpovídající bit řídícího registru na adrese 53277 ($D01D), t.j. musí být nastaven na 1. Následující POKE rozšiřuje sprite v x-ovém směru:

POKE 53277,PEEK (53277) OR (2 šipka nahoru SN)

kde SN je číslo spritu (0 až 7).

Pro zmenšení spritu je nutné odpovídající bit v řídícím registru na adrese 53277 ($D01D) vypnout, t.j. nastavit na 0. Následující POKE zmenšuje sprite v x-ovém směru:

POKE 53277,PEEK (53277) AND (255-2 šipka nahoru SN)

kde SN je číslo spritu (0 až 7).

Zmenšení spritu ve vertikálním směru se provádí vypnutím odpovídajícího bitu v řídícím registru na adrese 53271 ($D017), t.j. nastaví se na 0. Následující POKE zmenšuje sprite v y-ovém směru: POKE 53271,PEEK (53271) AND (255- šipka nahoru SN)

kde SN je číslo spritu (0 až 7).

Pozice spritu

Pro pohyb spritu na obrazovce slouží třípoziční registry:

1. registr pozice x
2. registr pozice y
3. rozšiřující registr pozice C

Každý sprite má svoje registry pozice x,y a rozšiřující registr pozice x. Sprite se může umístit v pozicích 512 (x-pozice) x 255 (y-pozice). Registry pozice X,Y pracují společně, ve dvojici. Umístění X a Y registru v mapě paměti je následující.

Nejprve je X registr pro sprite 0, pak Y registr pro sprite 0. Následující X registr pro sprite 1, Y registr pro sprite 1 atd. Po všech 16 X a Y registrech následuje registr rozšiřující pozice X (X MSB). Tabulka ukazuje seznam všech registrů pozic pro každý sprite. S těmito registry se pracuje pomocí příkazu POKE ...

STANDARDNÍ BITOVÁ MAPA

Standard. bitová mapa nám dává 320 rozlišitelných bodů ve směru horizontálním a 200 ve směru vertikálním a výběr 2 barev v každé matici 8 x 8 bodů. Bitová mapa je aktivována nastavením bitu 5 řídícího registru VIC-II na adrese 53265 ($D011) na jedničku. Tak to dělá následující POKE:

POKE 53265,PEEK(53265)OR 32

Bitová mapa je odstavena nastavením bitu 5 řídícího registru VIC-II na adrese 53265($D011) na nulu.

POKE 53265,PEEK(53265) AND 223

Předtím, než se detailně pustíme do modu BITOVE MAPY, je tu ještě jeden problém, kterého bychom si měli všimnout - kde je umístěna paměť BITOVE MAPY.

Jak pracuje?

Pokud si vzpomenete na kapitolu programovatelných znaků, mohli jste nastavit bitové příznaky znaku uloženého v RAM tak, jak jste chtěli. Pokud jste současně mohli pozměňovaný znak vidět, mohli jste měnit jednotlivé tečky ve znaku a vidět, co se děje. To je základ bitové mapy. Celá obrazovka je plněna programovatelnými znaky a vy provádíte změny přímo v paměti, ve které jsou uloženy příznaky programovatelných znaků.

ADRESY		POPIS
DEC	HEX	
53248	$D000	SPRITE 0 REGISTR POZICE X
53249	$D001	SPRITE 0 REGISTR POZICE Y
53250	$D002	SPRITE 1 REGISTR POZICE X
53251	$D003	SPRITE 1 REGISTR POZICE Y
53252	$D004	SPRITE 2 REGISTR POZICE X
53253	$D005	SPRITE 2 REGISTR POZICE Y
53254	$D006	SPRITE 3 REGISTR POZICE X
53255	$D007	SPRITE 3 REGISTR POZICE Y
53256	$D008	SPRITE 4 REGISTR POZICE X
53257	$D009	SPRITE 4 REGISTR POZICE Y
53258	$D00A	SPRITE 5 REGISTR POZICE X
53259	$D00B	SPRITE 5 REGISTR POZICE Y
53260	$D00C	SPRITE 6 REGISTR POZICE X
53261	$D00D	SPRITE 6 REGISTR POZICE Y
53262	$D00E	SPRITE 7 REGISTR POZICE X
53263	$D00F	SPRITE 7 REGISTR POZICE Y
53264	$D010	SPRITE X MSB REGISTR

Pozice spritu je vypočítána na horní levý roh 24x21 bodové oblasti. Nezávisí na počtu navržených bodů spritu. I když užijete jen jeden bod jako sprite a chcete jej umístit do středu obrazovky, musíte spočítat přesnou pozici horního levého rohu.

Vertikální pozicování

Na TV obrazovce se může programovat 200 různých bodů ve vertikálním (Y) směru. Registr pozice Y však může nabývat hodnoty až 255. To znamená, že můžete více než dostatečně posouvat se spritem nahoru a dolů. Můžete sprite posunout na obrazovku i pryč z obrazovky. K tomu je však zapotřebí více než 200 hodnot.

První hodnota, kdy je sprite vidět na vrcholu obrazovky je v Y-směru 30. Pro rozšířený sprite je to 9 (protože každý bod je dvojnásobný).

První Y hodnota, kdy je celý sprite vidět (všech 21 možných bodů) je 50 (neplatí pro rozšířený).

První hodnota, kdy je již celý sprite z obrazovky, je 250.

Příklad:

```
10 PRINT "(SHIFT)(CLR/HOME)"      REM:SMAZANI OBRAZOVKY
20 POKE 2040,13                   REM:NAST. SPRITU 0 DO 13
30 FOR I=0 TO 62: POKE 832+I,129:REM:DATA DO BL.13 POKE SPR
35 NEXT I
40  V=53248                              REM:NASTAVENI  POCATKU
   VIDEO C 50 POKE V+21,1         REM:ZAPNUTI SPRITU 0
60 POKE V+39,1                    REM:NAST. BARVY SPRITU 0
70 POKE V+1,100                   REM:NAST. Y POZICE SPRITU
80 POKE V+16,0: POKE V,100        REM:NAST. X POZICE SPRITU
```

Horizontální pozicování

Pozicování v X-ovém směru je komplikovanější, protože je zde více než 256 pozic. To znamená použití zvláštního bitu nebo devátého bitu. Přidáním zvláštního bitu se získá pro sprite 512 možných pozic v x-ovém směru, t.j. zleva doprava. Každý sprite může být v pozici 0 až 511. Avšak pouze hodnoty mezi 24 a 343 zajišťují viditelnost spritu na obrazovce. Jestliže pozice X (spritu) je větší než 255 (na pravé straně obrazovky), pak se musí nastavit odpovídající bit v X MSB registru. Jestliže X pozice je menší než 256 (na levé straně obrazovky), pak bit v X MSB registru musí být vynulován. Bity 0 až 7 v X MSB registru odpovídají spritům 0 až 7. Následující program pohybuje spritem přes obrazovku:

Příklad:

```
10 PRINT "(SHIFT)(CLR/HOME)"
20 POKE 2040,12
30 FOR I=0 TO 62:POKE 832+I,129:NEXT I
40 V=53248
50 POKE V+21,1
60 POKE V+39,1
70 POKE V+1,100
80 FOR J=0 TO 347
90 HX=INT(J/256):LX=J-256*HX
100 POKE V,LX:POKE V+16,HX:NEXT J
```

Když přesouváte rozšířený sprite na levou stranu obrazovky v X-ovém směru, musíte na začátku mít sprite mimo obrazovku na pravé straně. To je proto, že rozšířený sprite je větší než činí prostor na levé straně obrazovky.

Příklad:

```
10 PRINT "(SHIFT)(CLR/HOME)
20 POKE 2040,10
30 FOR I=0 TO 62: POKE 832+I,129: NEXT I
40 V=53248
50 POKE V+21,1
60 POKE V+39,1: POKE V+23,1: POKE V+29,1
70 POKE V+1,100
80 J=486
90 HX=INT(J/256): LX=J-256*HX
100 POKE V,LX: POKE V+16,HX
110 J=J+1: IF J)511 THEN J=0
120 IF J)498 OR J(349 GOTO 90
```

Obrázek 3-3 vysvětluje pozicování spritu. Užitím těchto hpodnot můžete pozicovat každý sprite.

Pozicování spritů všeobecně

Nerozšířené sprity jsou alespoň částečně viditelné v 40ti sloupcovém a 25ti řádkovém modu s následujícími parametry:

1 (= X (= 343
30 (= Y (= 249

V 38 sloupcovém modu se parametr X mění následovně:

8 (= X (= 334

V 24 řádkovém modu se parametr Y mění následovně:

34 (= Y (= 245

Rozšířené sprity jsou alespoň částečně viditelné v 40ti sloupcovém a 25ti řádkovém modu s následujícími parametry:

489) = X (= 343
9) = Y (= 249

V 38 sloupcovém modu se parametr X mění následovně:

496) = X (= 334

V 24 řádkovém modu se parametr Y mění následovně:

13 (= Y (= 245

Priorita zobrazování spritu

Sprity mají schopnost křížit svoje cesty nebo křížit se před, nebo za jinými objekty na obrazovce. To umožňuje třírozměrné efekty.

Vzájemná priorita spritu je pevná. To znamená, že sprite 0 má vyšší prioritu než sprite 1. Sprite 7 má pak nejnižší prioritu. Jinými slovy, jestliže sprite 1 a sprite 6 jsou posouvány tak, že se překříží, pak sprite 1 bude před spritem 6.

Když plánujete, který sprite se objeví v popředí obrazu, pak mu musíte navrhnout nižší číslo než spritu, který chcete mít v pozadí scény. T.j. tento sprite bude mít vyšší číslo.

Poznámka: Efekt okno: jestliže sprite s vyšší prioritou má "díry" v oblasti, kde nejsou body nastaveny na 1 a je zapnut, pak sprite s nižší prioritou bude vidět skrz. Také lze udělat se spritem a daty v pozadí.

Priorita spritu a pozadí je řízena registrem (SPRITE/BACKGROUND PRIORITY) priority spritu a pozadí na adrese 53275 ($D01B). Každému spritu odpovídá bit v tomto registru. Jestliže bit je 0, pak sprite má vyšší prioritu než pozadí na obrazovce. Jinými slovy, sprite se zobrazí před daty v pozadí. Jestliže bit je 1, pak sprite má nižší prioritu než pozadí. Pak se sprite zobrazí za daty v pozadí.

Detekce kolize

Jeden z nejzajímavějších aspektů čipu VIC II je detekce kolize. Kolize může být objevena mezi sprity nebo sprity a pozadím.

Kolize nastane, když nenulová část spritu se překryje s nenulovou části jiného spritu nebo znaku na obrazovce.

Kolize spritu

Kolize spritu jsou registrovány řídícím registrem čipu VIC II na adrese 53278 ($D01E) (registr kolize spritu). Každý sprite má svůj bit v tomto registru. Jestliže bit má hodnotu 1, pak je sprite zapojen do kolize. Bity tohoto registru mohou být čteny příkazem PEEK. Po čtení je registr automaticky vynulován, proto je vhodné hodnotu registru uložit do proměnné.
Poznámka: Kolize je zjištěna i tehdy, jsou-li sprity mimo obrazovku.

Detekce spritu a dat

Kolize spritu a dat je zjišťována registrem kolize spritu a dat na adrese 53279 ($D01F).

Každý sprite má svůj odpovídající bit v tomto registru. Jestliže má hodnotu 1, pak nastala kolize. Registr lze číst příkazem PEEK. Po čtení je registr automaticky vynulován, proto je vhodné jeho hodnotu uložit do proměnné.
Poznámka: MULTI-COLOR data 01 jsou brána do úvahy při kolizi, když jsou zobrazena na obrazovce. Pokud jsou v barvě pozadí obrazovky, je vhodné nevyvolávat kolizi 01 v multi-color modu.

Další grafické vlastnosti

Čištění (blanking) obrazovky

Bit 4 řídícího registru VIC II na adrese 53265 ($D011) řídí funkci čištění obrazovky. Když je zapnut (nastaven na 1), je obrazovka normální. Když je bit 4 nulový (je vypnut), pak je celá obrazovka v barvě okolí (BORDER).

Následující POKE čistí obrazovku. Data nejsou ztracena, pokud nebyly zobrazeny.

POKE 53265,PEEK (53265) AND 239

zpětnou funkci provádí příkaz

POKE 53265,PEEK (53265) OR 16

Raster registr

Raster registr je na adrese 53266 ($D012). Raster registr má dvojí smysl. Když čtete tento registr, dostáváte nižších 8 bitů běžné rester pozice. Raster pozice nejplatnějšího bitu (the most significant bit* je v registru na adrese 53265 ($D011). Užitím raster registru nastavíte časování výměn Vašeho zobrazení tak, že můžete obrazovku zbavit blikání. Výměny Vaší obrazovky se budou provádět, když raster není přítomen v zobrazované ploše, což je tehdy, když pozice Vašich bodů jsou mezi 51 a 251.

Když je zapsán raster registr (vč. MSB), je zapsané číslo uloženo pro použití v raster srovnávací funkci. Pokud je aktuální hodnota rasteru stejná jako číslo zapsané v raste registru, je bit v interrupt registru na adrese 53273 ($D019) zapnut nastavením na 1.
Poznámka: Jestliže správný interrupt bit to umožňuje, nastane přerušení (IRQ).

INTERRUPT STATUS REGISTR (ISR) = stavový registr přerušení

ISR ukazuje běžný stav zdroje přerušení. Běžný stav bitu 2 přerušovacího registru bude 1, když se dva sprity srazí. To stejné platí pro bity 0-3, viz tabulka. Bit 7 je nastaven také na 1, pokud nastane přerušení.

ISR je na adrese 53273 ($D019) a vypadá následovně:

ZKRATKA	BIT	POPIS
IRST	0	Nastaví se, pokud se ukládá číslo rastru
IMDC	1	Nastaví se v případě kolize spritu a dat
IMMC	2	Nastaví se v případě kolize sprítů
ILP	3	Nastaví se při zápor. přechodu světel. pera
IRQ	7	

Jakmile má být nastaven bit přerušení, musí být zapsána 1 do tohoto bitu v registru přerušení, když jste připraven s ním manipulovat. To dovoluje oddělenou manipulaci bez ukládání dalších bitů přerušení.

INTERRUPT ENABLE REGISTR je na adrese 53274 ($D01A). Má stejný formát jako stavový registr přerušení (interrupt status register). Jestliže odpovídající bit v IER je nastaven na 1, nenastane žádné přerušení z tohoto zdroje. ISR může být ještě použit pro informace, ale nebude generováno žádné přerušení. Aby bylo umožněno přerušení, musí být odpovídající bit nastaven na 1 (viz tabulka shora).

Tato mocná struktura umožňuje využívat rozčlenění obrazovkových modů. Nyní můžete např. mít polovinu obrazovkových bitů mapových, polovinu textu, více než 8 spritů současně atd. Tajemstvím je užívání přerušení správně. Např. chcete-li mít horní polovinu obrazovky v mapovém modu a dolní polovinu v textovém, nastavte srovnávací registr (jak bylo vysvětleno v předchozím) pro spodní část obrzovky. Dojde-li k přerušení, přesuňte (pomocí VIC II) znaky z ROM, pak nastavte raster srovnávací registr na přerušení v horní polovině obrazovky. Jakmile nastane přerušení v horní polovině obrazovky, pak pomocí VIC II nastavte znaky z RAM (bit map mod).

Můžete také zobrazit více jak 8 spritů stejným způsobem. Bohužel BASIC není pro tento účel dostatečně rychlý. Chcete-li tedy začít využívat zobrazovacích přerušení, pracujte ve strojovém jazyce.

Navržení obrazovky a kombinování barev znaků

Barevné TV přijímače jsou omezeny ve svých schopnostech umístit některé barvy vedle sebe do stejného řádku. Některé kombinace barev znaků a obrazovky tvoří rozmazané obrazy. Následující tabulka ukazuje, kterým barevným kombinacím se vyhnout a které jsou naopak vhodné.

BARVA ZNAKU

barva obrazovky	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	x	.	x	.	.	.	x	.	.	x	.	.	.	.	.	.
1	.	x	.	x	.	.	.	x	.	.	.	.	.	x	.	.
2	x	.	x	x	.	x	x	.	.	x	.	x	x	x	x	.
3	.	x	x	x	x	.	.	x	x	x	x	.	x	x	.	x
4	.	.	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5	.	.	x	.	x	x	x	x	x	x	x	x	.	x	.	x
6	.	.	x	.	x	x	x	x	x	x	x	x	x	.	.	.
7	.	x	.	x	x	x	.	x	.	.	.	.	.	x	x	x
8	.	.	.	x	x	x	x	.	x	.	x	x	x	x	x	.
9	x	.	x	x	x	x	x	.	.	x	.	x	x	x	x	.
10	.	.	.	x	x	x	x	.	x	.	x	x	x	x	x	.
11	.	.	x	.	x	x	x	.	x	x	x	x	.	.	.	.
12	.	.	.	x	x	x	.	x	x	.	x	.	x	x	x	.
13	.	x	x	x	x	.	.	x	x	x	x	.	x	x	x	x
14	.	.	x	.	x	x	.	x	x	x	x	.	x	x	x	.
15	.	.	.	x	.	.	.	x	x	.	.	.	.	x	.	x

Programování spritů - další pohledy

Tato část je navržena jako základní seznámení s programováním spritů.

Vytváření spritů v BASICu - krátký program

V této části jsou 3 různé programovací techniky v BASICu, s kterými vytvoříte grafické obrazy a animační triky. Můžete použít počítačové grafické symboly. Můžete programovat vlastní znaky, nebo můžete použít "sprite grafiku". Pro ilustraci následuje snadný program v BASICu, jenž vytváří sprite:

```
10 PRINT "(SHIFT)CLR/HOME)"
20 POKE 2040,13
30 FOR S=832 TO 832+62:POKES,255:NEXT
40 V=53248
50 POKEV+21,1
```

```
60 POKEV+39,1
70 POKEV,24
80 POKEV+1,100
```

Tento program zahrnuje klíčové "ingridience", které potřebujete k utvoření spritu. Tento program definuje první sprite ... sprite 0 ... jako celistvý bílý čtverec na obrazovce. Následuje vysvětlení programu, řádek po řádku.

řádek 10 čistí obrazovku

řádek 20 nastavuje "ukazatel spritu", kde se budou ukládat data spritu. Sprite 0 je nastaven na 2040, sprite 1 na 2041, sprite 2 na 2042 a tak dále, až sprite 7 na 2047 Můžete nastavit všech 8 ukazatelů spritu na 13 užitím těchto příkazů"

```
20 FOR SP=2040 TO 2047: POKE SP,13: NEXT SP
```

řádek 30 zapisuje první sprite (sprite 0) do 63 bytové oblasti paměti RAM s počátkem na adrese 832 (každý sprite potřebuje 63 bytů paměti). První sprite (sprite 0) je adresován v oblasti paměti od 832 do 894.

řádek 40 nastavuje do proměnné "V" hodnotu 53248, což je počáteční adresa VIDEO čipu. Tento vstup se používá ve tvaru V+číslo pro nastavování spritu. My užijeme vzorce V+číslo, protože tento formát chrání paměť a pracuje se s méně čísly. Např. POKE 53248+21 nebo POKE 53269 je to stejné, ale V+21 zabírá méně místa a lépe se pamatuje.

řádek 50 umožňuje "zapnout" sprite 0. Máme 8 spritů, od 0 do 7. Zapnutí jednoho spritu, nebo kombinace spritů může provést POKE V+21 následován číslem od 0 (vypnutí všech spritů) do 255 (zapnutí všech 8 spritů). Můžete zapnout jeden nebo více spritů pomocí následujících čísel:

vše	Sprite 0	Sp. 1	Sp. 2	Sp. 3	Sp. 4	Sp. 5	Sp. 6	Sp. 7
255	1		2	4	8	16	32	64	128

Také můžete zapínat kombinace spritů, např. POKE V+21, 129 zapíná sprity 0 a 7, sečte se 1 a 128.

řádek 60 nastavuje barvu spritu 0. Existuje 16 barev pro sprity; 0 (černá) až 15 (zelená). Každý sprite potřebuje vlastním příkazem POKE nastavit svoji barvu od V+39 do V+46. Např. POKE V+46,15 sprite 7 barva zelená.

Když vytvoříte sprite, můžete měnit barvu, pozici a tvar v přímém nebo bezprostředním modu. Spusťte program příkazem RUN a napište následující příkaz bez čísla řádku a stalčte RETURN:

```
POKE V+39,8
```

Sprite na obrazovce bude nyní oranžový. Zkuste příkazem POKE další číslo mezi 0 a 15. Avšak toto děláte v přímém modu, jestliže dáte opět RUN, sprite bude znovu v bílé barvě.

řádek 70 určuje horizontální ("X") pozici spritu na obrazovce. Číslo reprezentuje umístění levého horního rohu spritu. Nejvzdálenější levá pozice, kdy můžete sprite vidět je 24, ačkoliv můžete přesunout sprite z obrazovky do, pozice 0.

řádek 80 určuje vertikální pozici ("Y") spritu. V tomto programu máme sprite umístěn v X pozici 24 a Y pozici 100. Zkuste změnit umístění příkazem POKE v přímém modu:

```
POKE V,24:POKE V+1,229
```

Každé číslo od 832 do 895 ve Vašem spritu 0 reprezentuje jeden blok 8 bodů, se třemi osmibodovými bloky v každé horizontální řadě spritu. Cyklus na řádku 30 provádí následující: POKE 832,255 vytváří prvních 8 pevných bodů ... pak POKE 833,255 vytváří dalších 8 bodů a tak dále až do adresy 894, kde je poslední skupina 8 bodů ve spodním pravém rohu spritu. Lépe to uvidíte, když zkusíte v přímém modu natypovat následující, druhá skupina 8 bodů pak zmizí:

```
POKE 833,0 (nazpět typujte POKE 833,255 nebo RUN)
```

Následující řádek, který můžete přidat k Vašemu programu, ruší bloky uprostřed spritu:

```
90 FOR A=836 TO 891 STEP 3: POKE A,0: NEXT A
```

Pamatujte si, že body tvořící sprity jsou vytvořeny skupinami bloku po osmi. Tento řádek ruší pátou skupinu 8 bodů (blok 836) a každý třetí blok až do bloku 890. Zkuste POKEem jiná čísla bloku mezi 832 a 894 vytvořit nebo zrušit.

Tip pro vytváření spritu:

```
10 PRINT CHR$(147):V=53428:POKE V+21,1:POKE 2040,13:POKE
   V+39,1
20 FOR S=832 TO 894:POKE S,255:NEXT S:POKE V,24:POKE
   V+1,100
```

Tento program je oproti ukázkovému hutnější.

Pozicování SPRITU na obrazovce

Obrazovka je rozdělena do X a Y souřadnic, jako graf X souřadnice je horizontální pozice přes obrazovku a Y souřadnice je vertikální pozice shora dolů (viz obr. 3-4).

Pozici spritu na obrazovce musíte provést dvěma POKE ... X a Y pozici ... tím říkáte počítači, kde má zobrazit horní levý roh spritu. Pamatujte si, že sprity obsahují 504 individuýlních bodů, 24x21 ... takž POKEm umístíte sprite do levého horního rohu obrazovky, sprite se zobrazí jako grafický obraz z 24x21 bodů, počínaje na X-Y pozici, kterou určíte. Sprite se vždy zobrazí podle horního levého rohu i když definujete jen malou část 24x21 bodové plochy.

Pro porozumění jak pracovat s X-Y pozicemi si prohlédněte obr. 3-5, který ukazuje X a Y v relaci k obrazovce. Tmavá plocha ukazuje viditelnou televizní plochu a bílá plocha reprezentuje oblasti, které nejsou vidět.

Chcete-li umístit sprite v daném místě, musíte nastavit X a Y pro každý sprite. Pamatujte si, že každý sprite má svůj vlastní X POKE a Y POKE. Na následující tabulce uvidíte nastavení X a Y pozic pro všech 8 spritů.

	SP 0	SP 1	SP 2	SP 3	SP 4	SP 5	SP 6	SP 7
X	V,X	V+2,X	V+4,X	V+6,X	V+8,X	V+10,X	V+12,X	V+14,X
Y	V+1,Y	V+3,Y	V+5,Y	V+7,Y	V+9,Y	V+11,Y	V+13,Y	V+15,Y
PR X	V+16,1	V+16,2	V+16,4	V+16,8	V+16,16	V+16,32	V+16,64	V+16,128

POKE X POZICE: Možné hodnoty pro X jsou mezi 0 až 255; počítá se zprava doleva. Hodnoty 0 až 23 dávají celý nebo část spritu mimo obrazovku na levé straně. Hodnoty 24 až 255 zobrazují sprite na obrazovce. Umístění spritu v těchto pozicích je právě tento příkaz POKE. Např. umístit sprite 1 do levého horního rohu: POKE V+2,24.

Hodnoty X za 255 POZICI: Nastavit se za 255 pozici je potřeba provést druhý POKE, užít čísla "PR X" z předchozí tabulky. Horizontální číslování (X) normálně pokračuje od 255.pozice do 256,257 atd.,ale protože registr má jen 8 bitů, musíme užít "druhého registru", který rozšiřuje pravou stranu obrazovky a začíná se číslovat opět od 0. Jestliže chcete číslovat za 255, musíte uvést POKE V+16 a číslo příslušného spritu. To dává dalších 65 pozic v X směru (přečíslováno od 0 do 659) na pravé straně obrazovky.

POKE V POZICE: Možné hodnoty jsou od 0 do 255 shora dolů. Hodnoty 0 až 49 umísťují celý sprite nebo jen část mimo obrazovku. Hodnoty 50 až 229 umísťují sprite do viditelné části obrazovky. Hodnoty 230 až 255 dávají celý sprite nebo část spritu mimo spodní část obrazovky.

Podívejte se, jak se pozicuje sprite 1:

```
10    PRINT   "(SHIFT)(CLR/HOME)":V=53248:POKE   V+21,2:POKE
   2041,13:FOR S=832 TO 895:POKE S,255:NEXT
20 POKE V+40,7
30 POKE V+2,24
40 POKE V+3,50
```

Tento jednoduchý program vytváří sprite 1 jako pevný blok a umísťuje ho do levého horního rohu obrazovky. Nyní změňte řádek 40:

```
40 POKE V+3,229
```

Ten přesune sprite do spodního levého rohu obrazovky. Nyní otestujeme PR X spritu. Změňte řádek 30:

```
30 POKE V+2,255
```

Ten přesune sprite doprava, ale limitu PR X, který je 255. V tomto bodě se musí nastavit rozšiřující bit v registru 16. Jinými slovy, musíte typovat POKE V+16 a číslo. Typujte následující:

```
30 POKE V+16,PEEK:V+16) OR 2:POKE V+12,0
```

POKE V+16,2 nastavuje rozšiřující bit X pozice pro sprite 1 a začíná od pozice 256. POKE V+2,0 zobrazuje sprite v nové pozici nula, která je resetována na bod 256.

Návrat zpět na levou stranu obrazovky se provede vynulováním rozšiřujícího bitu příslušného spritu.

Jak celkově pozicovat ... POKE X pozici od 0 do 255 pro sprite. Rozšířit pozici za bod 255 lze použitím POKE (V+16). Tímto příkazem se X počítá znovu od 0. Zpět na levou stranu X pozice se dostanete příkazem POKE (V+16), PEEK (V+16) AND 254.

Pozicování více spritů na obrazovce

Program, který následuje, definuje 3 různé sprity (0,1,2) v různých barvách a pozicích na obrazovce.

```
10 PRINT "(SHIFT)(CLR/HOME)": V=53248: FOR S=832 TO 895:
   POKE S,255:NEXT
20 FOR M=2040 TO 2042: POKE M,13: NEXT
30 POKE V+21,7
40 POKE V+39,1: POKE V+40,7: POKE V+41,9
50 POKE V,24: POKE V+1,50
60 POKE V+2,12: POKE V+3,229
70 POKE V+4,255: POKE V+5,50
```

Pro jednoduchost jsou všechny tři sprity definovány jako plné čtverce a data jsou umístěna na stejném místě. Bílý sprite 0 je v levé horní části obrazovky. Zlutý sprite 1 v elvé spodní části obrazovky, ale polovina spritu není vidět. Konečně oranžový srite 2 je v pravé části (pozice 255) ... ale co udělat, jestliže chcete zobrazit sprite za 255. pozicí do prava?

Zobrazení spritu za 255. pozicí

Je umožněno speciálním příkazem POKE, který nastaví rozšiřující bit X pozice a tím se může pozicovat přes 256 pozicí obrazovky.

Nejprve proveďte POKE V+16 s číslem spritu. Potom přiřaďte X pozici, změňte řádek 50:

```
50 POKE V+16,1:POKE V,24:POKE V+1,75
```

POKE V+16 s číslem umožní "otevřít" pravou část obrazovky ... nová X pozice 24 pro sprite 0 nyní bude 24 bodů za 255. pozicí doprava. Změňte řádek 60:

```
60 POKE V+16,1:POKE V,65:POKE V+1,75
```

Stejným způsobem se nastaví sprity, které potřebujete přesouvat zleva do prava.

Priorita SPRITU

Můžete vytvářet různé sprity a přesouvat je pod nebo nad sebou na obrazovce. Tato třírozměrná iluze je dána prioritou spritů, jenž určuje, který sprite má prioritu před jiným, je-li jich na obrazovce vyšší počet než dva.

Pravidlo "První přišel, první obsloužen" znamená, že sprity s nižšími čísly mají automaticky prioritu před sprity s vyššími čísly. Např. zobrazíte-li sprite 0 a sprite 1, pak sprite 0 se objeví před spritem 1. Sprite 0 bude vždy před všemi ostatními sprity, protože má nejnižší číslo. Pro srovnání sprite 1 má vyšší prioritu než 2-7 sprite atd. Sprite 7 (poslední) má nejnižší prioritu a bude zobrazen vždy za ostatními sprity, které překrývají jeho pozici.

Pro ilustraci jak pracuje priorita, změňte řádky 50,60,70 v následujícím programu:

```
10 PRINT"(SHIFT)(CLR/HOME)":FOR S=832 TO 895:POKE
   S,255:NEXT
20 FOR M=2040 TO 2042:POKE M,13:NEXT
30 POKE V+21,7
40 POKE V+39,1:POKEV+40,7:POKE V+41,8
50 POKE V,24:POKE V+1,50:POKE V+16,0
60 POKE V+2,34:POKE V+3,60
70 POKE V+4,44:POKE V+5,70
```

Vidíte, že bílý sprite je nad žlutým spritem a ten nad oranžovým spritem. Nyní když vidíte, jak se pracuje s prioritou spritu, můžete přesouvat sprity a využít výhod priority pro animaci.

Kreslení SPRITU

Kreslení spritu je obdobné jako vymalovávání prázdných ploch v omalovánkách. Každý sprite obsahuje body. Nakreslit sprite znamená obarvit některé body. Podívejte se na síť pro tvorbu spritu na obr. 3-6. Je to prázdný sprite:
obr. 3-6

Každý malý čtvereček reprezentuje jeden bod spritu. Sprite má 24x21 bodů, t.j. 504 bodů celkem. Vytvořit sprite znamená obarvit tyto body. Ale jak to provést? Napsat 504 čísel? Ne, stačí 63 čísel pro každý sprite. Jak to pracuje ...

Vytvoření spritu krok za krokem

Budeme spolu vytvářet jednoduchý příklad spritu krok za krokem.

KROK 1:

Zapsat tento program vytvářející sprite na bílý papír. Budete potřebovat speciální data sekci, která bude obsahovat 63 čísel pro vytvoření spritu.

```
10 PRINT "(SHIFT)(CLR/HOME)":POKE 53280,5:POKE 53281,6
20 V=53248"POKE V+34,3
30 POKE 53269,4"POKE 2042,13
40 FOR N=0 TO 62: READ Q:POKE 832+N,Q:NEXT

100 DATA 255,255,255
101 DATA 128,0,1
102 DATA 128,0,1
*
*
*
120 DATA 255,255,255
200 X=200: Y=100: POKE 53252,X: POKE 53253,Y
```

KROK 2:

Namalujte si síť - viz obr. 3-6 (nebo použijte grafický papír). Tužkou namalujte tvar spritu na síti. Pro náš příklad poslouží obdélník.

KROK 3:

Podívejte se na prvních osm bodů. Každá skupina bodů má číslo (128,64,32,16,8,4,2,1). Použije se binární aritmetika, která se používá u většiny počítačů. Následuje celkový pohled na první skupinu bodů v levém horním rohu sprite:

128 64 32 16 8 4 2 1

KROK 4:

Sečteme horní čísla nad vyplněnými body. První skupina má všech osm bodů bodů vyplněných, celkový součet je 255.

KROK 5:

Vstupuje číslo jako první DATA příkaz na řádku 100. Vstup 255 pro druhou a třetí skupinu.

KROK 6:

Podívejte se na prvních osm bodů v druhém řádku spritu. Sečtěte hodnoty plných bodů. Protože jen jeden z těchto osmi bodů je plný, celkový počet je 128.

KROK 7:

Sečtěte hodnoty další skupiny somi bodů (která je 0, protože je prázdná) a vstoupí řádek 101. Nyní přejděte na další skupinu a opakujte proces pro každých 8 bodů (t.j. 3 skupiny na řádku a 21 řádků). To je celkem 63 čísel. Každé číslo představuje jednu skupinu 8 bodů a 63 skupin po 8 se rovná 504 bodům. Podívejte se na program následovně. Každý řádek reprezentuje jeden řádek spritu. Každá tří čísla v řádku reprezentují 3 skupiny osmi bodů. A každé číslo říká, které body budou zobrazeny a které zůstanou nezobrazeny.

KROK 8:

Zmenšete program do nejmenšího možného paměťového prostoru. Vynechte všechny mezery a maximálně využijte řádku.

Přesouvání spritu na obrazovku

Nyní, když jste sprite vytvořili, přesuňte jej na obrazovku. Přidejte tyto dva řádky do Vašeho programu:

```
50 POKE V+5,100:FOR X=24 TO 255:POKE V+4,X:NEXT:POKE
V+16,4
55 FOR X=0 TO 65:POKE V+4:NEXT X:POKE V+16,0:GOTO 50
```

Řádek 50 dělá POKE Y pozice (zkuste 50 nebo 229). Pak nastavte pomocí FOR ... NEXT cyklu X pozici od 0 do 255. Až je dosažena pozice 255, pak POKEm se nastaví rozšiřující bit X pozice (POKE V+16,4), který zajistí přesun na pravou stranu obrazovky.

Řádek 55 sprite pokračuje do poslední 64. pozice. Příkaz GOTO 50 zajistí rolování spritu v X-ovém směru. Následující řádky přesouvají sprite zpět a vpřed.

```
50 POKE V+5,100:FOR X=24 TO 255:POKE V+4,X:NEXT:POKE
V+16,4: FOR X=0 TO 64:POKE V+4,X:NEXT X
```

```
55 FOR X=65 TO 0 STEP -1:POKE V+4,4,x(NEXT:POKE
   V+16,0:FOR X=255 TO 24 STEP -1:POKEV+4,X:NEXT
60 GOTO 50
```

Tento program provádí to stejné co předchozí, ale mimoto při dosažení konce pravé strany obrazovky vrací sprite zpět v opačném směru.

Vertikální rolování

Tento typ přesunu spritu se nazývá rolování. Váš sprite roluje shora nebo zdola v Yovém směru. Zrušte řádky 50 a 55 a nahraďte je:

```
50 POKE V+4,24:FOR Y=0 TO 255:POKE V+5,Y:NEXT
```

Tančící myš - příklad programu se spritem

Některé techniky popsané v tomto manuálu jsou tak málo srozumitelné, že je spolu probereme v programu "Michaelova tančící myš". Tento program užívá tři různé sprity v roztomilé animaci se zvukovými efekty - pomůže porozumět, jak pracují a rozebereme využití každého příkzu, takže uvidíte, jak je program tvořen.

PROGRAM "TANCICI MYS"

Řádek 5:

S=54272 Nastavujte proměnnou S na hodnotu 54272, na níž začíná paměť SOUND CHIPU. Od nynějška se bude používat POKE S plus hodnota, mimo přímého adresování

POKE S+24,15
Stejně jako POKE 54296, nastaví se nejvyšší hlasitost.

POKE S,220 Stejně jako POKE 54272, který nastavuje nízkou frekvenci hlasu 1 pro tón, který aproximuje vysoké C oktávu 6.

POKE S+1,60 Nastavuje vysokou frekvenci hlasu 1 pro tón, který přibližuje vysoké C oktávu 6.

POKE S+5,15 Nastavuje ATTACK/DECAY pro hlas 1 a v tomto případě se skládá z maximální úrovně DECAY bez attacku, který produkuje "echo" efekt.

POKE S+6,215
Nastavuje SUSTAINT/RELEASE pro hlas 1 (215 představuje kombinaci hodnot sustain a release).

Řádek 10:

POKES+7,120 Nastavuje nízkou frekvenci hlasu 2

POKES+8,100 Nastavuje vysokou frekvenci hlasu 2

POKES+12,15 Nastavuje ATTACK/DECAY hlasu 2 na stejnou úroveň jako hals 1

POKES+13,215
Nastavuje SUSTAIN/RELEASE hlasu 2 na stejnou úroveňjako hlas 1

Řádek 15:

PRINT "SHIFT
Čistí obrazovku
CLR/HOME"

V=53248 Definuje proměnnou V jako počáteční adresu VIC čipu, který řídí sprity. Od této chvíle se bude paměť pro sprity definovat V plus.

POKE V+21,1 Zapíná sprite 1 (je použitelný)

Řádek 20:

FOR S1=12288
Používáme jeden sprite (sprite 0) pro animaci,ale

TO 12350 použijeme tři definice dat pro tři různé případy. Pro naši animaci budeme přepínat pointer na sprite 0 na tři místa v paměti, kde máme uložena data, která definují naše tři různé podoby. Stejný sprite bude předefinován vždy přes sebe ve třech různých případech pro vznik animace tančící myši. Můžeme definovat tucet spritu v DATA příkazech a rotovat tyto podoby v jedňom nebo více spritech.

Vidíte, že nejsme limitování jednou podobu spritu, nebo verzí. Jeden sprite může mít mnoho různých podob jednoduchým měněním ukazatele spritu na různá místa paměti, kde jsou uložena data pro různé podoby. Tento řádek znamená, že ukládáme data pro podobu spritu 1 na adresy 12288 až 12350.

READ Q1 — Čtení posloupnosti 63 čísel z příkazu DATA, který začíná na řádku 100. Q1 je libovolná proměnná. Stejně může být A, Z1 nebo jiná proměnná.

POKE S1,Q1 — Zapsání prvního čísla z příkazu DATA (první "Q1" je 30) do prvního paměťového místa na adrese 12288.

NEXT — Říká počítači:"prováděj vše mezi FOR a NEXT (READ Q1 a POKE S1,Q1) tak dlouho, dokud S1 nedosáhne hodnoty 12350. Poslední je POKE 12350,0.

Řádek 25:

FOR S2=12352 TO 12414 — Druhá podoba spritu nula je definována mezi adresami12352 až 12414. Adresa 12352 je užita pro definování první skupiny spritů, ale neobsahuje žádná data.

READ Q2 — Čte 63 čísel, která následují čísla, jenž jsme užili pro první podobu spritu.

POKE S2,Q2 — Zapisuje data (Q2) do paměti (S2) pro druhou podobu spritu, která začíná na adrese 12352.

NEXT — Stejně jako na řádku 20.

Řádek 35:

POKE V+39,15 — Nastavuje světle zelenou barvu spritu 0.

POKE V+1,68 — Nastavuje horní pravou stranu rohu čtverce spritu do vertikální (Y) pozice 68. Pro porovnání, pozice 50 je vrch levé části rohu Y pozice ve viditelné části obrazovky.

Řádek 40:

PRINT TAB(160) — Slouží jako tabulátor 160 mezer z levé části prostoru pro znaky na obrazovce, což je stejné jako 4 řádky čisticího příkazu zde začíná PRINT zprávy na 6. řádku obrazovky.

"(CTRL)(WHT)" — Současně stiskněte klávesy CTRL a WHT. Jestliže toto provedete mezi závorkami, zobrazí se "reverzní" E. Nastaví se bílá barva pro další zobrazované znaky příkazem PRINT.

I AM THE DANCING MOUSE — Nastaví se opět světle modrá barva na konci příkazu PRINT. Současným stisknutím kláves C= a 7

C=7" — mezi uvozovkami se zobrazí "reverzní symbol kosočtverce".

Řádek 45:

P=192 — Nastavuje proměnnou P na hodnotu 192. Číslo 192 je ukazatel, který musíte použít, v tomto případě na "bod" spritu 0 v paměti, která začíná na adrese 12288. Změnou tohoto ukazatele na umístění dalších dvou spritů je tajné užitíéí jednoho spritu pro vytvoření animace.

Řádek 50:

FOR X=0 TO 347 — Přesun Vašeho spritu 3 po X pozici (rychlý přesun)

STEP 3 — z pozice 0 do pozice 347.

Řádek 55:

RX=INT(X/256) — RX je celá část X/256, což znamená, že RX je nula když X má větší pozici než 256 a RX je 1, má-li X pozici 256. Použijeme RX v případě POKE V+16 s 0, nebo 1 k zapnutí pravé strany obrazovky.

LX=X-RX*256 Když je sprite v X pozici na 0, vztah vypadá jako: LX=0-(0 krát 256) nebo 0. Když je sprite v X pozici 1, vztah vypadá takto: LX=1-(0 krát 256), nebo 1. Když je sprite v X pozici 256, vztah vypadá takto: LX=256-(1 krát 256) nebo 0, což je návrat X zpět do 0 a musí vždy být uděláno, než přejedete na pravou stranu příkazem POKE V+16,1.

Řádek 60:

POKE V, LX POKE V nastavuje hodnotu horizontální X pozice spritu 0 na obrazovce. Jak bylo vidět dříve, hodnota LX, což je horizontální poloha spritu, se mění od 0 do 255 a automaticky se nuluje při dosazení 256.

POKE V+16,RX

POKE V+16 vždy zapíná "pravou stranu" obrazovky za pozici 256 a nuluje horizontální souřadnice. RX je buď 0 nebo 1 a závisí na pozici spritu, jak určuje RX vztah na řádku 55.

Řádek 70:

IF P=2 THEN Jestliže je ukazatel spritu nastaven na 192 (první
GOSUB 200 podoba spritu), pak je řízení WAVEFORM pro první zvukový efekt nastaveno na 129 a 128 na řádku 200.

Řádek 75:

IF P=193 THEN

Jestliže je ukazatel spritu nastaven na 193 (druhá
GOSUB 300 podoba spritu) řízení WAVEFORM pro druhý zvukový efekt je nastaveno na 129 a 128 na řádku 300.

Řádek 80:

POKE 2040,P Nastaví ukazatel spritu na adresu 192 (pamatujete si P=192 na řádku 45? Kde užíváme P?)

FOR T=1 TO 60
NEXT

Časová smyčka, která nastavuje rychlost tance myši (zkuste rychleji nebo pomaleji zvýšením/snížením čísla 60).

Řádek 85:

P=P+1 Nyní zvýšíme hodnotu ukazatele přičtením 1
IF P=194 THEN

k původní hodnotě P. Ceká se na bod spritu ve
P=192 třetí paměťové alokaci.,. 192 ukazuje na adresy 12288 až 12350, 193 na 12352 až 12414 a 194 na 12416 až 12478. Tento řádek říká počátači, nastav zpět do P 192 jakmile P bude 195. P je 192,193,194 a pak se vrací na 192 a ukazatel ukazuje na tři různé podoby spritu na tři 64-bytové skupiny.

Řádek 90:

NEXT X Sprite má jednu ze 3 podob, jen tak je připuštěn k přesunu přes obrazovku. Pak skočí o další 3 X pozice. NEXT X odpovídá příkazu FOR na řádku 50.

Řádek 95:

END Konec programu, jenž nastane, když myš vyjde z obrazovky.

Řádek 100-109:

DATA Podoby spritu se čtou z těchto čísel. Prvních 63 čísel dává podobu 1 atd. Tyto data jsou umístěna do tří paměťových míst.

Řádek 200:

POKE S+4,129

Nastavení Waveform na 129, zapnutí zvukového efektu.

POKE S+4,128
Nastavení Waveform na 129, vypnutí zvukového efketu.

RETURN Vrací řízení programu na konec řádku 70 po nastavení waveform.

Řádek 300:

POKE S+11,129
Nastavení waveform na 129, zapnutí zvukového efektu.

POKE S+11,128
Nastavení waveform na 128, vypnutí zvukového efektu.

RETURN Vrácení řízení programu na konec řádku 75.

Poznámky k vytváření SPRITE

Alternativní paměť pro SPRITE a bufer kazety.

nastavení ukazatele	SPRITE 0 2040,130	SPRITE 1 2041,140	SPRITE 2 2042,15	Jestliže užíváte 1 až 3 SPRITE, můžete použít tuto paměť pro buffer kazety (832-1023), ale pro více než 3 sprity používejte umístění od 12288 do 12798 (viz graf).
Body spritu	832 až	894 až	960 až	
Umístění pro bloky 13-15	894	958	1022	

Zapnutí spritu:

Sprity můžete zapínat individuálně užitím POKE V+21 a číslo z grafu ALE zapnutím jednoho spritu vypnete ostatní. Chcete-li zapnout dva nebo více spritů, sečtete spolu čísla spritů, které chcete zapnout (např. POKE V+21,6 zapíná sprity 1 a 2). Toto je metoda, kterou můžete použít k vypnutí jednoho spritu a jenž nezpůsobí nic jiného (užití pro animaci).

Příklad:

Vypnutí spritu 0: POKE V+21,PEEK(V+21) AND (255-1). Změňte číslo 1 v (255-1) na 1,2,4,8,16,32,64, nebo 128 (pro sprity 0-7). Další použití spritu bez ovlivnění dalších spritů normálně zapněte: POKE V+21,PEEK(V+21)OR 1 a zaměňte OR 1 za OR 2 (sprite 2), OR 4 (sprite 3), atd.

Hodnota X-pozice za 255:
X pozice je od 0 do 255. K zaslání spritu za 255 X-pozici musíte nejprve nastavit POKE Vé21, pak POKE nové hodnoty od 0 do 63, která umístí sprite na další pozici v pravé části obrazovky. Návrat zpět na pozice 0-255, se provede příkazem POKE V+16,0 a POKE do X s hodnotou od 0 do 255.

Hodnota Y-pozice:
Y-pozice je od 0 do 255 vč. 0 až 49 v neviditelné části plochy obrazovky, 50 až 229 ve viditelné části a 230 až 255 v neviditelné části spodní plochy.