BŘEZEN

FUN with Commodore

1994

časopis uživatelů Commodore 64/128

10. číslo

BASIC

Tipy a triky

Programujeme

DEMO

A/D převodník

Universal basic

Jiffy DOS

Postavte si scaner

64

OBSAH

CO BUDE V PŘÍŠTÍM ČÍSLE?

Basic
Assembler
Stane se C65 následníkem?
C64 a video
Když C64 stávkuje
Comotronic News

Adresa redakce: Dolnomlýnská 2, 787 01 Šumperk * Autoři čísla: Jaroslav Vančura a Jiří Kouřil
Podávání novinových zásilek povoleno Oblastní správou pošt v Ostravě pod č.j. 2882/92-P1 ze dne 14. 12. 1992
Podávanie novinových zásielok povolené SP š.p. ZsRP Bratislava č.j. 613-PD-1993 zo dňa 1. apríla 1993
FUN with Commodore – časopis Comotronic klubu pro uživatele počítačů Commodore 64/128.
2 / 94 – 10. číslo
Fotosazba REPROtisk J. Kotinský * Tisk: Vegaprint Šumperk

ÚVODNÍ SLOVO

Vážení čtenáři,

do dnešního čísla jsme připravili, věřím, řadu zajímavých článků. V rubrice Basic se pokusíme objasnit jak je to se Zero–page. Jedná se totiž o důležitou část paměti počítače obsluhovanou systémem. Zero–page se počítá od adresy 00 po adresu 256. Změna v datech na těchto adresách ovlivňuje zásadním způsobem chování počítače. Dnes uvádíme prvou část celého pojednání, dokončení si budete moci přečíst v příštím čísle.

V rubrice Assembler jsme se dostali k praktickým ukázkám programování ve strojovém kódu. Abyste i vy byli při testování jednotlivých vzorových programů úspěšní, budete potřebovat program typu monitor. Existuje jich celá řada (Basic 3.5–Tedmon, Hesmon, Smon). Poměrně komfortním monitorem jsou vybaveny i zásuvné moduly FCIII a FCII.

Řadu fyzikálních veličin lze převést na napětí a k měření potom využít C64. Pro sběr dat se hodí jednoduchý A/D převodník využívající principu převodu stejnosměrného napětí na frekvenci. Stavba převodníku je natolik jednoduchá, že ji zvládnou i méně zběhlí fandové C64.

Na dalších stránkách představujeme užitečné rozšíření basicu vašeho C64 o pár nových příkazů programem Universal–basic. Dále najdete hodnocení Jiffy DOS a stavební návod na jednoduchý scanner. Jeho stavbu však nedoporučuji začátečníkům.

Řada z vás má jistě obsáhlou sbírku programů na C64. Starý dobrý počítač však přestává stačit výkonem i pořídíte si PC. Co však s programy? Zbavit se jich nebo ne? Článek lecco napoví...

redaktor 10. čísla FUN
Jaroslav Vančura
Šumperk, březen 1994

POČÍTAČ PŘED STARTEM

Prvních 256 bajtů v paměti C64 je obestřeno záhadami. Pokud se do nich zapíší nějaké hodnoty, dělá počítač roztodivné věci. Znáte však příčiny?

Často citovaná nulová stránka paměti počítače (Zero–page) je zcela vyplněna důležitými daty. Teprve ona, spolu se strojovým programem interpreteru basicu a systémem v pamětech ROM probudí počítač k životu. Většina paměťových míst je rezervována pro basic a systém. Bez dat v Zero–page by počítač po zapnutí nevytiskl na obrazovku jediný znak. V dalším textu se budu odvolávat na tři důležité pojmy:

1. Pointer (ukazatel na začátek dat)
2. Vektor (ukazatel na strojové programy)
3. Flag (číselná hodnota reagující na výsledek matematické operace)

Pointry a vektory jsou v paměti sousedy. Protože paměťové místo má jen 256 hodnot, ale v celé paměti je však 65535 „domovních čísel", je potřeba pro dosažení vyšších adres použít dvou paměťových míst (Low/High byte). Oba články se pak spojují do jednoho čísla. Podívejme se např. na paměťová místa 43,44. Obě spolu představují pointer začátku basicu, tedy to místo v paměti kam se naláduje a odkud se odstartuje program v basicu. Nižší, paměťové místo 43 (lhostejno jaká hodnota je v něm uložena), bývá označováno jako low–byte. Jeho obsah se přebírá nezměněn, hodnotami se překrývá rozsah 0 až 255. Druhý bajt (44) bývá nazýván high–byte. Hodnoty, které toto paměťové místo obsahuje se násobí 256. To znamená, že pokud je na tomto místě „1", bere ji mikroprocesor jako 256, 2 dává 512 atd. až poslední možné číslo 255 dává 65280. Pokud se k těmto hodnotám přičtou čísla z adresy 43, dá se jimi zobrazit celá číselná řada mezi 0 a 65535, tím se pokryje celá celá paměť C64. Pro vyčíslení takových pointerů se dá s úspěchem využít následující rovnice:

adresa = low–byte + (high–byte * 256)

nebo v našem případě pro přečtení pointeru:

PRINT PEEK (43) + PEEK (44)*256

Stejný postup se používá při výpočtu pointerů a vektorů. Vraťme se však zpět k zero–page.

0 – směrový registr dat procesorového portu (hodnota 47)

Určuje směr toku dat na procesorovém portu (=paměťové místo 1).

Mikroprocesor 6510 v C64 má oproti svému předchůdci 6502 šest volně programovatelných vstupů/výstupů. Obvod může sám bez další podpory vyřídit řadu důležitých úloh. Proč však jen 6 vývodů a ne 8, když jeden bajt má 8 bitů? Na otázku existuje jednoduchá odpověď. Po odečtení nutných pinů pro adresy a data zbývá volných právě šest vývodů.
Adresa 0 představuje pro tento port odpovídající registr směru toku dat. To znamená, že každý ze šesti vývodů se dá zapsáním dat do této adresy přepnout jako „přijímač" nebo „vysílač". Jednotlivé bity odpovídají vlastnímu portu (paměťové místo 1). Využity jsou bity 0 až 5. Pro všech šest bitů platí:

bit na 0 = vstup bit na 1 = výstup

Při zapnutí počítače zapíše systém do tohoto registru číslo 47. Významné bity se počítají zleva doprava. Bit s nejnižší hodnotou je zcela vpravo a je označen jako bit 0.

dekadicky 47 = binárně 00101111

Vidíme, že bit 4 je programován jako vstup. Bity 6 a 7 se nevyužívají.

1 – Input/Output (55)

Představuje port mikroprocesoru. Jeho šest vstupně výstupních linek je shodně mechanicky provedeno. Všechny vývody se dají dotazovat nebo zapisovat podle toho, jak se stanoví ve směrovém registru 0. V C64 jsou jednotlivým vývodům přiřazeny následující funkce:

bit	funkce	
0	basic–ROM=1	RAM=0
1	Kernal ROM=1	RAM=0
2	I/O=1	znaková sada=0
3	data z datasetu	
4	klávesa datasetu stlačena=0	nestlačena=1
5	motor zap=1	motor vyp=0
6	nevyužit=0	
7	nevyužit=0	

Po zapnutí přístroje proběhne resetovací rutina a do registru se zapíše hodnota dekadicky 55 = binárně 00110111. To znamená, že se zapnou ROM basic (40960 až 49151) a Kernal (57344 až 65535). Pro pochopení této konfigurace se musí vědět, že v C64 se paměťové rozsahy překrývají a podle konfigurace dané hodnotou na adrese 1 se přepne buď ROM (paměť jen pro čtení) nebo RAM (paměť pro čtení i zápis). Paměť RAM ztratí při opakovaném vypnutí/zapnutí svá data. Bez dat ovšem mikroprocesor nemůže pracovat a bloudí.
V paměťovém rozsahu 53248 až 57343 (bit 2) je

pokryta oblast I/O ROM. Zbylá data se týkají kazetového provozu.
Pozor! Pokud dosadíte do paměťového místa 1
POKE 1,53
přepnete na RAM. Mikroprocesor se ztratí v labyrintu své paměti a přestane normálně reagovat (53=00110101). K životu jej přiměje až opakované vypnutí a zapnutí (Reset).

2 – nevyužito

3/4 – vektor pro převedení plovoucí des. čárky na pevnou

(170/177=45482)
Používá se pro vnitřní výpočetní operace. Ukazuje na rutinu v systému, na kterou se basic při přepočítávání odvolává. Proto se dá využívat jen ve strojových rutinách.

5/6 – vektor pro přepočítávání pevné des. čárky na plovoucí

(145/179=45969)
Používá se pro vnitřní přepočet. Je obrácená k funkci vektoru 3/4. Také tato funkce se dá využívat jen ve strojových programech.

7 –

Využívá se v basicovských rutinách jako bufer pro přezkoušení textových zadání. Normálně se sem odkládá ASCII hodnota zkoušeného znaku. Protože tato buňka je využívána příkazy READ, AND,INT, EXP není možno provést její vyhodnocení z basicu.

8 – flag

Podobně jako paměťové místo 7 slouží i toto paměťové místo jako bufer. Převážně se používá při převádění příkazů basicu do tzv. token (příkazové kódy).

9 – sloupec TAB

Je využíván příkazy TAB a SPC. Ukazuje pozici sloupce, ve kterém je kurzor před provedením příkazu.

10 – Load/Verify

0=Load, 1=Verify. Protože rutiny Load a Verify jsou identické, rozlišuje systém na základě údaje na této adrese mezi oběma módy.

11 – flag pro počet DIM a pro zadávání

Je používán pro přepočet při určování počtu polí a po uzavření zadání přes klávesnici jako bufer.

12 – flag pro DIM

Bufer pro rutiny basicu k rozlišení mezi dimenzovanou proměnnou nebo polem. Mimo to rozpozná, zda se jedná o dimenzované pole nebo nedimenzované pole.

13 – flag, jedná-li se o numerickou proměnnou nebo string

Ukáže interpreteru basicu druh zpracovávaných dat.

14 – typ integer nebo real

Pracuje dohromady s paměťovou buňkou 13. Pro celé číslo je 128, pro číslo bez des čárky 0.

15 – flag při List

Přes tuto paměťovou buňku se přečtením flagu zjistí, zda následuje textový řetězec. Dále se zjistí, zda musí být provedena „Garbage Collection" (odstranění neobsazených proměnných). Pokud potom není možné uložení nových proměnných, ukáže se hlášení OUT OF MEMORY.

16 – flag pro FN

Je při hledání proměnných pro interpreter basicu rozhodující flag pro rozhodnutí mezi normální proměnnou a samostatně definovanou funkcí.

17 – INPUT/GET/READ

Protože všechny rutiny jsou částečně identické, rozhodne systém, do které z programových větví má odskočit:

```
  0  = INPUT
 64  = GET
152  = READ
```

18 – znaménko TAN/SIN/COS

Protože znaménko u trigonometrických funkcí je v jednotlivých kvadrantech rozdílné, pozná se touto buňkou jeho střídání. Pro osvětlení ještě jeden příklad:

```
10 FOR I=0 TO 10 STEP 0.01
20 PRINT PEEK(18), INT(I*100)/;SIN(I):NEXT
```

po startu RUN zjistíte, že se znaménko skutečně mění.

19 – aktivní přístroj pro I/O

Říká systému, který periferní přístroj je aktivní. Do úvahu připadají klávesnice, dataset, RS 232, user-port, obrazovka, tiskárna nebo flopy.

20/21 – adresa integeru

Obsahuje číslo řádku při zpracování příkazů LIST,GOTO,GOSUB a ON. Protože číslování řádků smí jít až do 63999, potřebuje tato rutina dvoubajtové zobrazení. Po jejím zpracování se objevuje zásadně hodnota 20.

22 – pointr stringstack

Ukazuje na následující volné paměťové místo v předchozím buferu stringů.

23/24 – poimter posledni string

Ukazatel na adresu posledního znakového řetězce v TSDC.

25–33 Temporary String Descriptor Stack

Údaje v tomto stacku obsahují délku, začáteční a koncovou adresu předchozího stringu.

34–37 pointer pro různé účely

Je používán interpreterem basicu pro různé mezivýsledky a flagy.

38–42 akku 1

Používá se při násobení a dělení jako pracovní paměť.

43/44 – poimtr BASIC–start

Ukazuje na začátek paměti basicu. Na tuto pozici se natahuje program příkazem
LOAD „xxx",8

45/46 – pointr start proměnných
Ukazuje na počáteční adresu paměti pro proměnné. Proměnné začínají jen po oddělovači (dvě nuly) bezprostředně po basicovském programu.
47/48 – pointr ARRAY–start
Ukazuje na počáteční adresu paměti pro pole (arrays)
49/50 – pointr ARRAY–konec
Ukazuje na koncovou adresu (+1) paměti pro pole.
51/52 – pointr RAM–konec
Ukazuje na konec volné paměti proměnných.
53/54 – pomocný pointr pro string
V tomto ukazateli je adresa naposled čtených proměnných.
55/56 – pointr BASIC RAM–konec
Ukazatel sděluje interpreteru nejvyšší adresu v basicu použitelné paměti.
57/58 – BASIC – číslo řádku
Obsahuje čísla právě zpracovávaných řádků v basicu. V rozšíření ukazuje strojový program v příkazu TRACE i aktuální číslo na obrazovce.
59/60 – číslo řádku pro CONT
Obsahuje číslo naposled provedeného řádku po END,STOP nebo po přerušení klávesou <RUN/STOP>.
61/62 – pointer pro CONT
Ukazuje na paměťovou pozici naposled prováděného řádku po END,STOP nebo <RUN/STOP>.
63/64 – číslo řádku pro data
Obsahuje číslo právě čteného data řádku.
65/66 – poimtr na další data
Ukazuje paměťovou polohu dalšího data řádku.
67/68 –
Ukazuje adresu z níž příkazy INPUT, GET a READ berou znaky.
69/70 – název proměnné
Název právě zpracovávané proměnné.
71/72 – adresa proměnné
Ukazuje na adresu obsahu právě volané proměnné.
73/74 – pointr proměnné pro smyčku
Ve smyčce FOR/NEXT obsahuje adresu proměnné, před jejím zpracováním procesorem. Slouží jako bufer při podobných příkazech basicu.
75/76 – bufer pro programový ukazatel
Toto paměťové místo se používá jako bufer při matematických funkcích nebo při READ.
77 – maska pro logické operace
Při vyhodnocování logických operací AND a OR se sem odkládá maska, která dává výsledek. Z basicu se nedá testovat.
78/79 – pointr pro FN
Ukazuje na adresu, od které se odkládá volně definovaná funkce.
80–82 – string–descriptor
Ukazuje na provizorní paměťové místo právě zpracovávaného řetězce znaků.
83 – flag pro Garbage Collection
Při vyprazdňování paměti obsahuje hodnotu udávající jakou délku (3 nebo 7 bajtů) má proměnná.
84–86 – vektor pro funkce
V buňce 84 je obsaženo číslo 76, kterým nepřímý skok JMP (xx) 85 a 86 ukazuje na adresu v tabulce. Adresa ukazuje na zpracovávanou rutinu.
87–96 – akku 3 a 4
Bufer pro matematické operace.
97–102 – FAC
Akumulátor s pohyblivou čárkou č.1 hraje centrální roli při zpracování čísel.
97=exponent, 98–101=mantisa, 102=znaménko
103 – čítač pro vyhodnocení polynomu
Paměť při převádění ASCII do pohyblivé čárky a čítač při vyhodnocování polynomů.
104 – zaokrouhlení pro FAC
Přetečení z FAC. Používá se jen tehdy, když se po vnitřním přepočtu uvádí zobrazitelné číslo (menší než 1,70141183 E 38).
105–110 – ARG
Akumulátor s pohyblivou čárkou č.2. Obsahuje argument matematických operací.
111 – flag pro znaménko FAC/ARG
Pro stejná znaménka je zde >0<, pro rozdílná >255<.
112 – zaokrouhlovací bajt
Patří k FAC. Do tohoto místa se odloží a zpětně načtou přepočítávaná místa z mantisy, zvýší se tím přesnost.
113/114 – pointr pro různé účely
Využívá se mnoha rutinami pro interní operace.
115/138 CHRGET
Přečte následující znak textu v basicu. Tato rutina je jádrem interpreteru basicu. Při zapnutí počítače se okopíruje z ROM. Jako jediný strojový program interpreteru je schopen se sám měnit. Díky samomodifikování obsáhne celou paměť. Tím je umožněno zpracovávat příkazy z řádků psaných v basicu nebo i v přímém módu.
139–143 – poslední hodnota RND
Při zapnutí obsahuje hodnoty 128, 79, 19, 82 a 88. Po vyvolání funkce RND(x) se uloží nový výsledek jako výchozí hodnota pro další funkci.
144 – status ST
Obsahuje proměnnou status, která se dá vyvolat pomocí PRINT (ST). Pro flopy nebo tiskárnu je význam jednotlivých bitů následující:

0 – chyba při zápisu
1 – chyba při čtení
6 – konec dat
7 – chyba DEVICE NOT PRESENT

pokračování v příštím čísle

ASSEMBLER NA C64

(9. pokračování)

Vážení čtenáři! V této pravidelné rubrice jsme se dostali k praktickým aplikacím programování v assembleru/strojovém kódu. V tuto chvíli se poněkud nepříjemně projevuje skutečnost, že C64 po zapnutí nemá k dispozici monitor strojového kódu. Monitor se musí do počítače nahrát, iniciovat a pak je teprve možné zapisovat v rubrice Assembler na C64 uváděné programy. Protože práce v monitoru dále vyžaduje časté přepočty mezi dekadickou, hexadecimální a také binární soustavou, přechody mezi Basicem a Monitorem, je nutné pečlivě zvážit, jaký program Monitor budete používat, abyste si práci zbytečně nekomplikovali.
Řada uživatelů dnes používá na svém C64 BASIC 3.5, který obsahuje dobrý monitor strojového kódu TEDMON a má také příkazy pro přepočty mezi jednotlivými početními soustavami, které jsou při práci v assembleru velmi užitečné. Proto považujeme BASIC 3.5 za vhodný pro začínající programátory v assembleru. To byl také hlavní důvod, proč jsou v rubrice uváděné příklady strojových programů ukládány od adresy $3000, která nekoliduje s Basicem 3.5 po jeho softwarové instalaci. Jediná úprava nutná při práci na C64 s instalovaným Basicem 3.5 je změna adresy video–RAM. Dalším zájemcům je u nás Basic 3.5 kdykoliv k dispozici.

Nyní musíme ještě uložit do akumulátoru obrazovkový kód znaku, který chceme na obrazovku vypsat. Tento příkaz uložíme do první smyčky, kde bude později také změněn obsah akumulátoru přičtením čísla 40. Doplníme tedy program takto:

```
LDA #$41          ;obrazovkový kód pro znak "A"
```

Nyní můžeme začít s vnitřní smyčkou. Jako první přijde opět příkaz k uložení obrazovkového kódu na momentální adresu. Zde použijeme nepřímé indexované adresování. Na paměťových místech $D8/D9 je vždy uložena adresa začátku řádku a pomocí registru Y pak projdeme všech 40 sloupců řádku.

```
LABEL STA ($D8),Y
```

Nyní musíme samozřejmě zvýšit obsah registru Y o jednu a testovat, zda již bylo "A" vypsáno 40x. Pokud ne, skočíme pomocí příkazu BNE zpět na příkaz STA:

```
INY               ;zvýšit obsah registru Z o jednu
CPY #$28          ;srovnat s číslem 40
BNE LABEL         ;při nerovnosti skok na příkaz STA
```

Tím je vnitřní smyčka kompletní. Nyní musíme ještě zvýšit obsah paměti $D8/D9 o 40, stejně jako zvýšit obsah registru X. Pokud je obsah registru X roven 25, bude program ukončen a skočí pomocí příkazu BRK zpět do assembleru.
Dále je uveden kompletní program.

```
A 3000 LDA #$00     ; LOW-bajt počáteční adresy
A 3002 LDY #$04     ; HIGH-bajt počáteční adresy
A 3004 STA $D8      ; uložit nejdříve LOW
A 3006 STY $D9      ; a pak HIGH bajt
A 3008 LDX #$00     ; čítač řádku na 0
A 300A LDY #$00     ; čítač sloupce na 0
A 300C LDA #$41     ; do akumulátoru kód znaku
                      "A"
A 300E STA($D8),8   ; zápis znaku "A" do paměti
A 3010 INY          ; zvýšit čítač sloupce
A 3011 CPY #$28     ; je už 40 sloupců?
A 3013 BDE $300E    ; když ne, pak opět zápis
A 3015 CLC          ; připravit příznak pro ADC
A 3016 LDA $D8      ; uložit low-bajt do akumulá-
                      toru
A 3018 ADC #$28     ; přidat 40
A 301A STA $D8      ; a opět uložit
A 301C BCC $3020    ; žádné přetečení, proto skok
A 301E INC $D9      ; přetečení! zvýšit high-bajt
A 3020 INX          ; zvýšit čítač řádku
A 3021 CPX #$19     ; je už 25 řádků?
A 3023 BNE $300A    ; když ne, pak další smyčka
A 3025 BRK          ; když ano, pak do monitoru
```

Na tomto programu je zajímavé taky zvýšení 16ti bitového čísla o 40, které je provedeno pouhým zvýšením hodnoty vyššího bajtu o 1, když je nastaven příznak carry, čili když dojde k přetečení.

Až dosud jsme zapisovali znak přímo do obrazové RAM. Existuje však i jiná možnost, jak vypsat znak na obrazovku. V Basicu je možno příkazem PRINT zapsat znak na libovolné místo obrazovky. Uvnitř textů, které mají být vypsány na obrazovku, se mohou vyskytovat také řídicí znaky atd. Z toho vyplývá, že v operačním systému našeho počítače musí již být zabudován podprogram pro výpis znaku na obrazovku.

Tento podprogram, který nese jméno "BSOUT", je uložen od adresy $FFD2. Pokud se tato adresa volá pomocí příkazu JSR, vypíše se na momentální pozici kurzoru znak, jehož ASCII kód je uložen v akumulátoru. Uvnitř této rutiny se obsah registrů X a Z přesune do paměti a zpět, takže obsah těchto registrů není voláním rutiny ovlivněn.
Zkusíme si tedy vypsat znak na obrazovku pomocí podprogramu–rutiny BSOUT. Aktivujte monitor strojového kódu, pokud jej již nemáte spuštěný a zapište následující program:

```
A 3000 LDA #$0D   ; uložení kódu RETURN do
                    akumulátoru
A 3002 JSR $FFD2  ; výpis kódu RETURN
A 3005 LDA #$2A   ; akumulátor naplnit kódem
                    znaku "*"
A 3007 JSR $FFD2  ; výpis znaku "*"
A 300A BRK        ; do monitoru
```

Tento nový program odstartujte následujícím příkazem

G 3000 <RETURN>

Výstup tohoto programu se projeví asi takto:

```
.
BREAK

   PC   SR   AC   XR   YR   SP
;300C   70   2A   FF   00   F8
```

Jak vidět, podařilo se nám vypsat na obrazovku znak "*", příkazem G. V našem prográmku jsme nejdříve uložili do akumulátoru dekadickou hodnotu 13. Tato hodnota odpovídá ASCII–kódu pro kód RETURN. Potom je s touto hodnotou volána rutina BSOUT na adrese $FFD2. Když se provede kód RETURN, skočí kurzor na začátek dalšího řádku. V našem programu jsme pak uložili do akumulátoru dekadickou hodnotu 42, jinak kód znaku "*". Pak je opět volána rutina BSOUT, která vypíše znak "*" na aktuální pozici kurzoru. Protože v našem případě byla pozice kurzoru v prvním sloupci řádku pod zadaným příkazem "G", vypíše se znak "*" na tomto místě.

Jak vidíte, je rutina BSOUT mnohem komfortnější, než "poukování" znaků do video–RAM. A pokud chceme rutinou BSOUT vypsat na obrazovku 256 znaků, stačí zkonstruovat smyčku od 0 do 255, potom uložit do akumulátoru ASCII kód znaku, který chceme na obrazovku vypisovat a vyvolat rutinu BSOUT. Dále uvedený program slouží k právě k výpisu 256 znaků "A" na obrazovku.

```
A 3000 LDY #$00   ; startovní hodnota do registru Y
A 3002 LDA #$41   ; naplnit akumulátor kódem "A"
A 3004 JSR $FFD2  ; volání BSOUT, výpis "A"
A 3007 INY        ; zvýšení registru Y
A 3008 BNE $3004  ; když není větší než 255, skok
A 300A BRK        ; do monitoru
```

Před vstupem do smyčky v našem programu jsme pouze naplnili akumulátor ASCII hodnotou kódu znaku, který chceme na obrazovku vypisovat. Potom voláme 256krát rutinu BSOUT. Příkaz BNE vraci program zpět na adresu $3004, dokud nedojde k přetečení registru Y, to znamená k jeho vynulování. Toto dotazování příkazem BNE je možné proto, že příkaz INY, zařazený před podmíněný skok BNE ovlivňuje kromě jiného také nulový příznak.

Pokud potřebujeme pomocí rutiny BSOUT vypsat na obrazovku 1000 znaků, vyjde to i v tomto případě jednodušší, než s použitím přímého zápisu do video–RAM. Potřebujeme vlastně jen 1000krát po sobě volat rutinu BSOUT, tedy zkonstruovat vhodnou smyčku. Zde musíme přiznat, že rutina BSOUT má jeden háček. Pokud se zapíše znak do pravého dolního rohu obrazovky, skroluje se obsah obrazovky automaticky o 1 řádek nahoru. Abychom tento problém obešli, vypíšeme znaků jen 999. K tomu se nabízí třeba následující řešení potřebných smyček:

9 x 111 = 999

Vnější smyčka pak běží od 0 do 8 a vnitřní smyčka od 0 do 110 (na nulu nezapomeňte!). Tak bude rutina BSOUT volána přesně 999krát. Protože obsah akumulátoru se během programu nemá měnit, nemusíme zapisovat příkaz LDA dovnitř smyčky, ale můžeme začít přímo jím. Pro jednoduchost si vezmeme zase znak "A":

```
LDA #$41 ;kód znaku "A" do akumulátoru
```

Nyní musíme iniciovat oba indexové registry. Použijeme registr X pro vnější a registr Y pro vnitřní smyčku.

Nyní jsou dvě možnosti, jak konstruovat smyčku. První je, nechat vnější smyčku probíhat o 0 do 8 a testovat, zda obsah registru X není roven 9.

Druhá možnost je nechat smyčku probíhat o 9 do 1 a testovat, zda obsah registru X není roven 0.
Tyto možnosti se samozřejmě týkají jak vnější, tak vnitřní smyčky. Protože při využití druhé možnosti odpadnou dva srovnávací příkazy, přikloníme se samozřejmě k němu. Vždyť díky opakování ve smyčce tak ušetříme celkem 999 srovnávacích příkazů!

Naplníme tedy oba naše indexové registry následujícími hodnotami:

```
LDX #$09  ; registr X naplnit hodnotou 9
LDY #$6F  ; registr Y naplnit hodnotou 111
```

Nyní musíme uvnitř smyčky volat rutinu BSOUT:

```
LOOP2 JSR $FFD2    ;vypsat znak z akumulátoru
```

Nyní snížíme čítač vnitřní smyčky YR o jednu, pokud není roven 0. V tom případě skočíme opět na příkaz JSR:

```
DEY        ; snížit čítač vnitřní smyčky
BNE LOOP2  ; pokud je už 0, pak na příkaz JSR
```

Tu stejnou proceduru musíme nyní provést pro vnější smyčku a pak skočit zpět do monitoru. Dohromady vypadá program třeba takto:

```
A 3000 LDA #$93     ; ASCII kód pro SHIFT+CLR/
                      HOME
A 3002 JSR $FFD2    ; vypsat na obrazovku
A 3005 LDA #$41     ; ASCII kód znaku "A"
A 3007 LDX #$09     ; XR pro vnější smyčku
A 3009 LDY #$6F     ; YR pro vnitřní smyčku
A 300B JSR $FFD2    ; vypsat obsah akumulátoru
A 300E DEY          ; snížit vnitřní smyčku
A 300F BNE $300B    ; pokud už 0, pak JSR
A 3011 DEX          ; snížit vnější smyčku
A 3012 BNE $3009    ; pokud už 0, pak na vnitřní
                      smyčku
A 3014 BRK          ; konec, do monitoru
```

Jak vidíte, je nejdříve pomocí rutiny BSOUT vypsán ASCII znak 147, který odpovídá kombinaci kláves SHIFT+CLR/HOME. Tím se vymaže obsah obrazovky.
Program odstartujte z monitoru příkazem

G 3000 <RETURN>

Pokud nechcete věřit, že opravdu došlo k vyplnění celé obrazovky až na znak v pravém dolním rohu, upravte program změnou posledního řádku:

A 3014 JMP $3014 ;nekonečná smyčka

Procesor po této úpravě skáče stále na paměťové místo $3014. Tato nekonečná smyčka se dá přerušit pouze resetem počítače.

7.2. Výpis textu

V předchozí kapitole jsme si vysvětlili, jak se na obrazovku vypisují jednotlivé znaky, nebo jak jeden znak vypsat na obrazovku vícekrát. Ale praktická potřeba vypsání jednoho znaku nebo více stejných znaků je malá. Mnohem potřebnější je vypsání souvislého textu nebo jiné definované řady znaků. Z toho důvodu se dále budeme zabývat otázkou, jak se texty na obrazovku vypisují.
Nejdříve si připomeneme, jak se jednotlivý znak na obrazovku vypisuje. Jednak je tu možnost zapsat znak přímo do video–RAM, jednak možnost využít rutinu BSOUT.
Logicky je práce s rutinou BSOUT elegantnější. Protože si ale chceme objasnit i různé způsoby adresování, probereme si obě možnosti.

Text, který chceme na obrazovku vypsat, musí být nejdříve uložen někde v paměti počítače v nějaké tabulce. Abychom tento text mohli na obrazovku vypsat, musíme znát počáteční adresu tabulky. Co se týče adresy konce tabulky, je zda k dispozici více možností.

Za prvé musíme vědět, kolik znaků náš text obsahuje. Pokud je toto známo, není problém nastavit náš indexový čítač na přezkoušení konce tabulky. Tato metoda má nicméně jeden následek, který musíme zvážit.

Představte si, že váš program má reagovat na stisk klávesy, například po stisku kláves 0 – 9 vypsat různé texty. Abyste toho dosáhli, musíte založit následující tabulky:

a) tabulku s texty,
b) tabulku s počátečními adresami textů,
c) tabulku s délkou textů.

To je pochopitelně poněkud nepohodlné. Jednodušší bude, když ukončíme naše texty znakem označujícím konec, například bajtem s hodnotou 0. Pak budeme programově vypisovat znaky na obrazovku tak dlouho, až se narazí na znak s kódem 0. V tom okamžiku se výpis textu ukončí.

Před tím, než si naše dvě uvedené možnosti blíže rozebereme, ještě si objasníme jeden pomocný program.

Protože monitor strojového kódu neumožňuje přímý zápis znaků do paměti, musíme pracně v tabulce obrazovkových kódů hledat kódy jednotlivých znaků. Tyto kódy pak musíme ještě převést do hexadecimálního tvaru a pak je teprve můžeme uložit do paměti počítače. Tato procedura je při ukládání delších textu pochopitelně únavná. Proto je dále uveden krátký program, který tuto práci zajistí za vás. Po odstartování programu zapíšete jednoduše požadovaný text a počáteční adresu paměti, kam se má text ukládat. Tato adresa může být zadána jak dekadicky, tak hexadecimálně. Po ukončení zadání uloží program text v podobě obrazovkových kódů do paměti a oznámí nám další volné místo, následující za zadaným textem. Pokud pracujete se znakem pro konec textu, musíte na něj na konci textu rezervovat místo.
A takto vypadá náš pomocný program:

```
100 REM **** POKE - INSERT ****
110 :
120 INPUT "TEXT:";A$
130 :
140 INPUT "POCATECNI ADRESA:";AD$
150 IF LEFT$(AD$,1)<>"$" THEN AD=VAL(AD$):
    GOTO 180
160 AD$=RIGHT$(AD$,LEN(AD$)-1)
170 AD=DEC(AD$)
180 :
190 PRINT CHR$(147): PRINT A$
200 :
210 FOR I=0 TO LEN(A$)-1
220 ::POKE AD+I,PEEK(1024+I)
230 NEXT I
240 :
250 PRINT CHR$( )
260 PRINT"DALSI VOLNA ADRESA:"
270 PRINT"DEK.:";AD+I
280 PRINT"HEX.:";HEX$(AD+1)
290 :
300 END
```

Princip tohoto krátkého programu je jednoduchý. Zadaný text vypíše pomocí příkazu PRINT do levého horního rohu obrazovky. Potom se text vypíše ještě jednou ve formě obrazovkových kódů. Tak vlastně zapíšeme text pomocí BSOUT rutiny do video-RAM, abychom jej mohli později v našem strojovém programu z paměti přepsat do obrazové paměti.

Ale nyní pár slov k našim různým možnostem výpisu.

Jako ukazatel uvnitř našeho textu použijeme registr Y. Pomocí tohoto registru a indexovaného adresování uložíme vypisovanou hodnotu z naší tabulky do akumulátoru.

pozice	1	2	3	4	5	6	7
text:	Z	K	O	U	S	K	A

Příkaz LDA použijeme v této formě:

LDA $zač.tabulky,Y

Potom zapíšeme (opět v indexovaném adresování) hodnotu do obrazové paměti. K tomu použijeme příkaz STA, který je protipólem příkazu LDA. Adresa za příkazem STA je adresa ve video-RAM, na kterou se má vypsat první znak našeho textu.

STA $adr.zač.textu,Y

Potom zvýšíme indexregistr YR a testujeme, zda již jsou vypsány všechny znaky. K tomuto účelu použijeme příkaz CPY, který srovnává obsah registru Y s danou hodnotou. Jako příklad jsme si vybrali výpis textu ZKOUSKA. Tento text má celkem 7 znaků, to znamená, že při výpisu na obrazovku musíme zjišťovat, zda je již v indexovém registru uložena hodnota 7 (nezapomenout na 0!).

Až potud vypadá náš strojový program takto:

```
A 3000 LDY  #$00     ; vynulování indexregistru
A 3002 LDA  $300E,Y  ; vytáhnout kód z tabulky
A 3005 STA  $0400,Y  ; zapsat kód do video-
                       RAM
A 3008 INY           ; zvýšit indexregistr
A 3009 CPY  #$07     ; jsou již vypsány všechny
                       znaky?
A 300B BNE  $3002    ; ne, pak pokračovat ve vý-
                       pisu
A 300  BRK           ; a zpět do monitoru
```

K zápisu požadovaného textu do paměti je možno použít dvě cesty.

Jednak máme k dispozici výše uvedený pomocný program v Basicu, jednak si můžeme jejich kódy najít v tabulce a přímo je zapsat do paměti počítače. To se provede následovně. V monitoru zapíšete:

> 300E 1A 0B 0F 15 13 0B 01 <RETURN>

Pokud využijete pomocný program v BASICu, pak jej spusťte, na otázku TEXT: ? zapište ZKOUSKA a potom stiskněte <RETURN>. Dále se vás program zeptá na adresu, od které se má tabulka textu do paměti uložit. Zapište zde $300E a opět stiskněte <RETURN>. Pomocný program pak vygeneruje tabulku. Dále musíte opět inicializovat monitor strojového kódu a spustit zadaný program příkazem

G 3000 <RETURN>.

Jak se přesvědčíte sami, vypíše se text řádně na obrazovku. Nyní si program upravíme tak, aby reagoval na znak oznamující konec vypisovaného textu. To znamená, že text se má vypisovat tak dlouho, dokud program nenarazí na bajt, který má námi určenou hodnotu (dříve jsme si uvedli, že jako koncový znak použijeme bajt s hodnotou 0).
Určitou nevýhodou tohoto postupu je, že ve vlastním textu pak nemůžeme koncový znak (je to znak @ – tzv. závináč) používat.

(JK)

TIPY A TRIKY

JDE OPĚT O ČAS

Zkoušeli jste už někdy v basicu realizovat hodiny, které čítaly zpět? Potom jistě víte, jak těžko se hledá správné řešení. Jediný timer obsluhovatelný z basicu počítá vpřed. Následující malý program ukazuje použití TI$:

```
10  PRINT "(CLR)"
20  TI$ = "000000"
30  PRINT TI$ "(UP)"
40  IF TI$ <> "000010" GOTO 30
50  END
```

TI$ bývá označovaná jako "rezervovaná proměnná", která je zpracovávaná systémem. Začíná na nule, kdy je počítač vypnut. Jiná numerická reservovaná proměnná se označuje TI a od zapnutí čítá 1/60 sekundy, zatímco proměnná TI$ mění svou hodnotu jedenkrát za sekundu (vyzkoušejte v hořejším programu v řádku 30 zaměnit proměnnou TI za TI$).
Těžší úlohou ovšem je, má-li se od určité hodnoty čítat dolů. Obvykle se člověk nevyhne složitým matematickým operacím, zvláště pak tehdy má-li čas být komfortně zobrazen v hodinách minutách a sekundách jak je normálně běžné při použití TI$. Krátký strojový program "COUNTDOWN", aktivovaný v paměti počítače, nechá systémový čítač běžet vzad.
Program se láduje příkazem:
LOAD "COUNTDOWN",8
a startuje potom
RUN.
Po krátké době se na obrazovce objeví DEMO, které ukazuje působení rutiny.
Nyní můžete zadat NEW a program vyzkoušet ještě jednou. Hodiny poběží stejně jako předtím. Teprve po aktivaci příkazem
SYS 679
běží hodiny obráceně (vyzkoušet).
Countdown běžící minutu by neměl být žádným problémem. Přitom se dá pomocí řetězcových funkcí izolovat každá část TI$:

```
10  SYS 676
20  TI$="000100": REM 1 minuta
30  PRINT "PROSIM BEHEM 1 MINUTY STISKNOUT
    KLAVESU"
40  PRINT "CAS JESTE "
50  PRINT MID$(TI$,3,2)" "RIGHT$(TI$,2)"(UP)"
60  GET A$: IF A$ <>"" THEN END
70  IF TI$ > "000000" THEN 40
80  PRINT "PRILIS POZDE !": END
```

Ačkoli tento speciální program zastavuje na nule, timer to nedělá. Provede podtečení a dále čítá od 235959 (23 hodin, 59 minut, 59 sekund, tedy celý jeden den) zpět k nule. S touto skutečností musíte počítat chcete-li COUNTDOWN použít ve vlastních programech.

Vlastní listing programu:

```
  1 REM "COUNTDOWN"
 10 FOR A=679 TO 733: READ B: C=C+B: POKE A,B
    :NEXT: REM NACTENI MC
 20 IF C >< 7836 THEN PRINT "DOWNCHYBA !":
    END
 30 DATA
    120,169,180,141,20,3,169,141,21,3,88,96
 40 DATA 198,162,165,162,201,255,208,28:REM
    ADRESU 162 SNIZIT
 50 DATA 198,161,165,161,201,255,208,20:REM
    ADRESU 161 SNIZIT
 60 DATA 198,160,165,160,201,255,208,12:REM
    ADRESU 160 SNIZIT
 70 DATA
    169,79,133,160,169,25,133,161,169,255,133,
    162:REM HODINY NA 23:59:59
 80 DATA 32,188,246:REM KLAVESA STOP
 90 DATA 76,52,234:REM DALE STAROU IRQ RU-
    TINOU
100 PRINT "DOWNTIS CITA TED SSEK VPRED/SPA-
    CE(KLAVESU)
105 POKE 198,0:WAIT 198,1
110 SYS 65418:REM COUNTDOWN VYP
120 TI$="000000"
130 PRINT TI$: IF TI$<"000005" THEN 130
140 PRINT "DOWNTIS CITA TED 10 SEK VZADU/SPA-
    CE(KLAVESU)
145 POKE 198,0:WAIT 198,1: TI$="000005"
150 SYS 679:REM COUNTDOWN ZAP
160 PRINT TI$: IF TI$<>"235953" THEN 160
170 PRINT "DOWNTOHLE A VEC!
180 SYS65418:REM COUNTDOWN VYP
```

MSE JAKO KOPÍROVACÍ PROGRAM

Programu MSE 1.0 lze s úspěchem použít jako kopírovacího programu. Příkazem <CTRL L> se dá totiž natáhnout program z diskety a příkazem <CTRL S> je možno jej opět uložit na jinou disketu. Podmínkou je, aby strojový program neobsazoval zrovna tu část paměti, která je vyhrazena pro vlastní MSE.
MSE 2.0 a jeho lepší verze MSE 2.1 používají přirozeně pro loading a saving jiné příkazy. Nejprve zadejte v hlavním menu MSE název programu. Potom pomocí <F3> program naládujte a pomocí pomocí <F5> znovu uložte.

2B OR NOT 2B

Basic 2.0 žel neobsahuje logickou funkci XOR (exclusive–or). Tato funkce se zpravidla obchází jinak:
E = (NOT A AND B) OR (A AND NOT B)
Použije-li se při úpravě tohoto výrazu de Morganova pravidla (not(a and b) = (not a or not b)), dá se tento term zkrátit na tvar
E = NOT (A AND B) AND (A OR B).
Úprava přináší výhodu ve zkrácení času vlastního výpočtu. Pro matematické fanoušky uvádím způsob odvození:

```
E  = (not A and B) or (a and not b)
E  = not(not((not A and B)or(A and B)))
E  = not(not(not A and B) and not(A and not B)))
E  = not((A or not B)and(not A or B))
E  = not((A and notA)or(A and B)or(not A and not
     B)or(B and not B)
E  = not((A andB)or(not A and not B))
E  = not(A and B)and not(not A and not B)
E  = not(A and B)and(A or B)
```

v tomto zobrazení je převod poněkud nepřehledný. Napíšete-li však jednotlivé termy pomocí logických symbolů situace se rázem změní.

SPRÁVNÉ ZAOKROUHLOVÁNÍ

U C64 se speeddos jsou často výsledky matematického zaokrouhlování chybné. Sem tam se chyba v zaokrohlení vyskytne i u normálního systému C64. Problém spočívá jednak v relativně nepřesné matematické rutině C64, ale i v použité rovnici. Ve většině případů se k zaokrouhlování na S míst za desetinnou čárkou v programu používá rovnice
X = INT (X*10S+.5/10S)
Pokud se za pomoci této rovnice zkouší počítat, řekněme, odmocnina z 41 zaokrouhlená na 5 míst za des. čárkou, obdrží se namísto očekávaného 6,40312 výraz 6,40311999. Správné zaokrouhlení zařídí následující rovnice:
X = INT (X*INT10S+:5)+.5)/INT(10S+.5)
Tato rovnice dává správné výsledky jak s použitím speeddos tak i s normálním systémem C64.

(JV)

PROGRAMUJEME DEMO

(pokračování)

Vážení čtenáři, v poslední době se objevují z vašich řad připomínky, že námi uváděné programy, zvláště v této kapitole, nelze zapisovat v monitoru, protože obsahují texty, které monitor "nebere". Je to dáno tím, že programy jsou psány v editoru Turbo–Ass, který je součástí balíku editor–compiler–debugger–utility, vhodném pro vývoj delších programů a vážnou práci ve strojovém kódu. Obdobou tohoto balíku je Merlin–assembler, který je u nás dostupný jako PD–disk. Důvodem k tomuto řešení je snaha předkládat programy ve formě, která je užitečná uživatelům, kteří již o programování něco vědí a také přehlednost v editoru zapsaných programů, která se při zápisu v monitoru zajistit nedá. Schopný programátor, který není úplný začátečník, dokáže pak i v editoru zapsaný program převést do monitoru. Proto těm, pro které je náš zápis nesrozumitelný, nezbývá než doporučit hlubší studium a konzultace u zkušenějších kamarádů.

V minulém pokračování jsme si popsali jednotlivé rutiny, nutné pro pořádné demo, pracující v přerušení. Dnes se konečně dostaneme od suchého čtení a teorie k radostnější práci. Probereme si téma barevných rastrů a perfektní plynulé skrolování.

RASTROVÉ PÁSY

Rastrové pásy, zvané také Copper–pásy podle Amigy, kde jsou podobné pásy realizovatelné za pomoci Copper–čipu, najdeme dnes ve skoro každém demu a hře. Buď se skrolují, nebo podkládají nějaký text nebo něco podobného. Rastrové pásy jsou dnes prostě neodmyslitelné. Při jejich realizaci narážíme však na vážný problém – správné časování. Jak snad víte, potřebuje VIC v každém osmém rastrovém řádku kvůli čtení obrazové paměti více času, aby tento řádek vygeneroval, než pro ostatní. To znamená, že nemusíme jen nalézt správné časování, ale musíme z něj ještě udělat proměnnou. Pro každý rastrový řádek potřebujeme proto vlastní hodnotu časování, která musí tak dlouho zpětně čítat (dekrementovat), dokud se nevygeneruje celý rastrový rá-

dek s požadovanou hodnotou barvy. Nicméně: zapíšeme hodnotu barvy do registru $D021 (barva obrazovky) a pokud si to přejeme, také do registru $D020 (barva rámečku). Změnami hodnot barev pro každý řádek pak dojde ke konečném fantastickému efektu – nejméně tehdy, když si barevné rozdělení v tabulce barev správně navrhneme. Takže dost řečí, zde je listing:

```
CHARSCROLL
        LDX SCRHELP     ; vytáhnout obsah
                          registru
        DEX             ; a dvakrát snížit
        DEX             ; (rychlost skrolová-
                          ní)
        STX SCRHELP     ; zapsat do pomoc-
                          ného
        STX SCROLLREG   ; a obrazového re-
                          gistru
        CPX #$BF        ; došlo již k podte-
                          čení?
        BEQ HARDSCR     ; pak tvrdé skrolo-
                          vání
        RTS             ; konec

HARDSCR
        LDX  #$C7       ; nastavit registr
        STX SCROLLREG   ; zpět
        STX SCRHELP     ;
        LDX #$00        ; obrazovkový řádek
HARD1   LDA $0721,X     ; rotovat
        STA $0720,X     ; o jedno místo
        INX             ; vlevo
        CPX #$27        ;
        BNE HARD1       ;
CHANGE  LDA TEXT        ; natáhnout textový
                          bajt
        CMP #$00        ; text na přerušení
        BNE CONT        ; ne, potom CONT
        LDA #$TEXT      ; lowbajt
        STA CHANGE+1    ; zapsat
        LDA #$20        ; a akumulátor na
        STA $0747       ; prázdný znak
        RTS             ; konec
        CONTSTA $0747   ; zapsat znak
        INC  CHANGE+1   ; CHANGE v kódu
                        ; o jednu zvýšit
SCREND  RTS             ; konec

TEXT .TEXT ((zde musí být uveden text))
```

;———————————————————

```
RASTERSHOW
        LDX #$00        ; čítač na $00
COLOR1  LDA COLTAB,X    ; vytáhnout hodnotu
                          barvy
        LDY WAITTAB,X   ; vytáhnout hodnotu
                          čekání
WAIT1   DEY             ; zpětné čítání
        BNE WAIT1       ; a srovnání cyklů
        STA  $D020      ; zapsat
        STA  $D021      ; barvu
        INX             ; čítač+1
        CPX #$40        ; už 40 barev?
        BNE COLOR1      ; ne, potom CO-
                          LOR1
        LDA #$00        ; ano, pak nastavit
        STA $D020       ; obrazovku na čer-
                          nou
        STA $D021       ;
        RTS             ; konec

COLTAB
        .BYTE  $06, $04, $0E, $03, $07, $0F
        .BYTE  $01, $01, $0F, $07, $03, $0E
        .BYTE  $06, $04, $00, $00, $09, $02
        .BYTE  $0A, $07, $0F, $01, $01, $0F
        .BYTE  $07, $0A, $02, $09, $00, $00
        .BYTE  $09, $0B, $08, $0C, $0F, $07
        .BYTE  $01, $01, $0F, $0C, $0F, $07
        .BYTE  $09, $00, $00, $02, $0A, $07
        .BYTE  $0F, $01, $0B, $0C, $0F, $01
        .BYTE  $01, $0F, $0C, $0B, $01, $0F
        .BYTE  $07, $0A, $02, $00, $00
COLEND
WAITTAB
        .BYTE  $09, $08, $08, $01, $08, $08
        .BYTE  $08, $08, $08, $08, $08, $01
        .BYTE  $08, $08, $08, $08, $08, $08
        .BYTE  $08, $01, $08, $08, $08, $08
        .BYTE  $08, $08, $08, $01, $08, $08
        .BYTE  $08, $08, $08, $08, $08, $01
        .BYTE  $08, $08, $08, $08, $08, $08
        .BYTE  $08, $01, $08, $08, $08, $08
        .BYTE  $08, $08, $08, $01, $08, $08
        .BYTE  $08, $08, $08, $08, $08, $01
        .BYTE  $08, $08, $08, $08, $08
```

;———————————————————

Jak vidíte, obsahuje listing hned dvě tabulky. Jednou jsou definovány barvy, druhá je určena pro časování. Pozor:

Pokud u vás některý řádek přes uvedenou definici časování bliká, běží tabulka pravděpodobně přes $FF v low-bajtu adresy. (Například když tabulka začíná na adrese $34F0 a končí na $3580.) V tom případě potřebuje C64 ještě nějaký čas navíc k načtení kódu barvy a proto definované časování stoprocentně nesedí. Dbejte proto na to, aby vaše tabulka začínala vždy na $00 v low-bajtu adresy, například na adrese $3000, $3F00, $5A00 atd. To se

dá v editoru zařídit snadno například u Turbo-assembleru přes pseudo-optokód "*=ADRESA", v VIS-Ass přes "SA"

OBLOŽENÍ PAMĚTI PRO DEMO PROGRAM:

$1000-$2000 Hudba vč. Playeru
$2800-$3000 Znaková sada 1
$3000-$3800 Znaková sada 2
$4000-$4450 Demo-rutina vč. textu, tabulek atd.

JEMNÉ SKROLOVÁNÍ

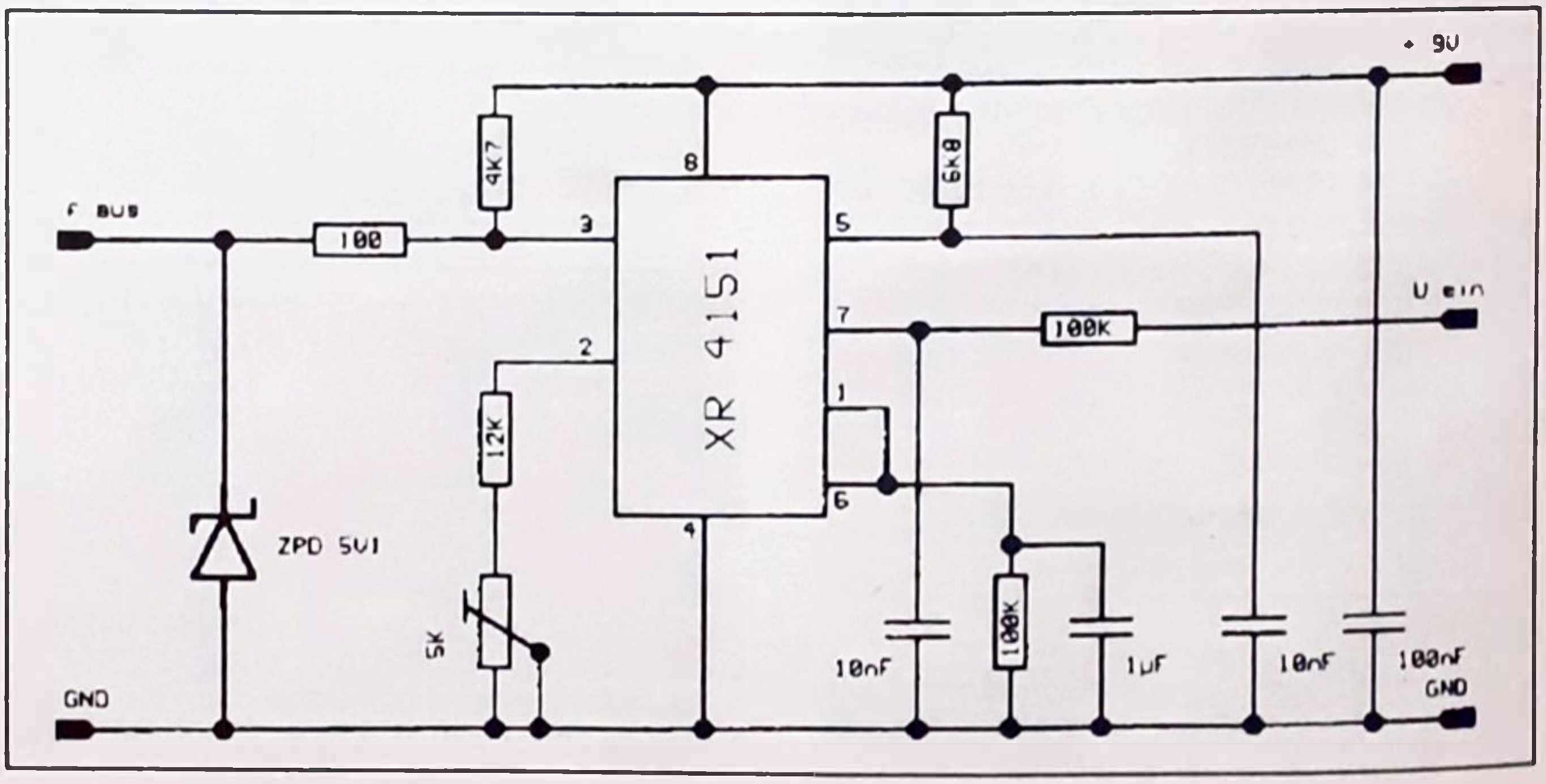
Jemné skrolování také není tak těžké naprogramovat. V našem předchozím pokračování jsme uvedli rutinu pro jemné skrolování CHARSCROLL, kterou jsme volali mezi dvěma rastrovými rozsahy pomocí JSR CHARSCROLL.
U této rutiny se na moc věcí dávat pozor nemusí. Uvedli jsme si paměť pro skrolování, do které je ukládána aktuální pozice registru $D016. To je nutné, protože zbytek obrazovky se nebude společně skrolovat. Tady v jednom rozsahu zapneme registr pro skrolování, v druhém jej nastavíme pro běh obrazovky na normální hodnotu. Pokud se v našem rozsahu bude posunovat 7 bitů, musíme také zde nastavit skrolovací registr zpět. Ke skrolování textu musíme taky ještě vytáhnout další textový bajt, zapsat na konec řádku (tedy do posledního sloupce) a hotovo. U návěští CHANGE se normálně vytáhne další bajt. Proto se při každém volání neobjeví ten stejný znak. Musíme nicméně použít jinou metodu než X- nebo Y- indexované adresování a to přímé adresování. Manipulujeme proto přímo s low-bajtem v návěští CHANGE. Pokud běží low-bajt přes $FF, začíná text odpředu. Pokud rutina narazí na bajt $00, automaticky se přesuší a začíná znovu překopírovávat text.
Kompletní listing uveřejníme v příštím čísle, neboť zabírá 4 strany.

(JK)

PŘEDSTAVUJEME

A/D PŘEVODNÍK

Commodore C64 se dá bez větší námahy převést na citlivý měřicí přístroj, který s vhodným senzorem poslouží pro měření řady veličin (proudu, napětí, el odporu, kapacity, koncentrace alkoholových par, radioaktivity, síly větru, délek atd.).
Protože téměř všechny veličiny indikované příslušnými čidly lze získat jako analogové signály, je první úlohou měření převod mezi analogovou a digitální formou. Jádrem jednoduchého převodníku je integrovaný obvod RCA 4152 (RCA 4151), který propor-



cionálně převádí stejnosměrné napětí na frekvenci. Výhodou je poměrně vysoká linearita převodu (odchylky dosahují okolo 1%) při lácí obvodu. Protože obvod RCA 4152 převádí napětí na řadu impulsů, lze jej připojit přímo na čítací vstup user-portu. K vyhodnocení převodu se dá potom s úspěchem použít programu napsaného v basicu, s částí strojového programu pro řízení reálného času C64.
Obvod RCA 4152 musí být napájen 9V. Proto je napětí ze zdroje 15 V nejprve stabilizováno obvodem 7809 a teprve potom přivedeno na vývod 8 RCA 4152.
S uvedeným obvodem jsme realizovali poměrně přesný milivoltmetr s následujícími parametry:

doba převodu	1 s
vstupní odpor	100kohmů
vstupní napětí	0 až 5 V_{ss}
výstupní napětí	TTL úroveň
výstupní frekvence	5kHz (5V)

Zájemci si mohou u firmy Comotronic objednat budicí program.

UNIVERSAL BASIC

Universal basic představuje malé, ale užitečné rozšíření basicu. Vedle různých příkazů DOS má i efektní příkazy pro basic. Opište pomocí MSE následující program, uložte jej a potom odstartujte pomocí RUN.

SCRATCH drv, "name"
Vymazává v drajvu <drv> datový soubor <name>. Jako <drv> může být uvedeno číslo 8 až 11.

VALIDATE drv
Potvrzuje disk v drajvu <drv>.

FORMAT drv, id, "name"
Formátuje disketu v drajvu <drv>. Opatří ji číslem <id> a názvem <name>. <id> je dvojmístné číslo mezi 00 a 99.

QFORMAT drv, "name"
Vymaže directory na disketě v drajvu <drv> a opatří ji novým názvem <name>.

BITSET adr,b
Nastaví bit <b> na adrese <adr>. <b> je hodnota mezi 0 až 7.

BITCLR adr,b
Vymaže bit <b> na adrese <adr>.

BITCHG adr,b
Změní bit <b> na adrese <adr>.

CLS
Vymaže obrazovku.

INVSCR
Invertuje obrazovku.

LOCATE x,y
Nastaví kurzor na souřadnice <x> (0 až 39) a <y> (0 až 24).

COLOR b,bg,fg
Obarví rámeček barvou <b>, pozadí barvou <bg> a písmo barvou <fg>.

BEEP
Zapne zvukový signál.

PAUSE t
Pauza odpovídá zvolenému času <t>. T je hodnota od 0 do 255 a vyjadřuje čas v 0,1 s. Tím se dosáhne časové zpoždění počítače až 25s.

BORDER on/off
Zapíná/vypíná border.

BLINE l
Nastaví border na řádek l (0 až 24).

BDATA z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,b,bg
Nastaví barvu borderu

<z1>	barva 5.řádku nad písmem
<z2>	barva 4.řádku nad písmem
<z3>	barva 3.řádku nad písmem
<z4>	barva 2.řádku nad písmem
<z5>	barva 1.řádku nad písmem
<z6>	barva 1.řádku pod písmem
<z7>	barva 2.řádku pod písmem
<z8>	barva 3.řádku pod písmem
<z9>	barva 4.řádku pod písmem
<z10>	barva 5.řádku pod písmem
<b>	barva rámečku písma
<bg>	barva pozadí písma

Upozornění! Program musíte ládovat v každém případě příkazem
LOAD "universal-basic",8
V žádném případě neprovádějte absolutní loading (",8,1"). Program by nešel příkazem RUN odstartovat.
Následující krátký listing ukáže jak s jednotlivými příkazy zacházet.

```
 5 A=PEEK(53280):B=PEEK(53281):C=PEEK(646)
10 CLS:COLOR 0,0,14,:BLINE 2
```

```
20 BDATA 1,3,14,11,6,9,2,4,10,7,6,6
30 LOCATE 8,2
40 PRINT "*** UNIVERSAL-BASIC ***"
50 BORDER ON: BEEP
60 LOCATE 6,5: COLOR 0,0,15
70 PRINT "(C) 1993 BY M. TSCHEREPANOW"
80 LOCATE 9,5
90 FOR T=9868 TO 1993 STEP -125: PRINT: LO-
   CATE 9,5: NEXT
100 COLOR 0,0,1: LOCATE 8,2: PRINT"*** UNI-
   VERSAL-BASIC ***"
110 FOR I=1 TO 5:
   BDATA 1,3,14,11,6,9,2,4,10,7,10,10:PAUSE 3
120 BDATA 1,3,14,11,6,9,2,4,10,7,14,14: PAUSE 3:
   NEXT
130 LOCATE 8,2: PRINT"23xSPACE": FOR T=2 TO
   5: BLINE T: NEXT
140 LOCATE 6,5: PRINT"28xSPACE"
```

```
PROGRAM  : UNIVERSAL-BASIC

 000 : 0E 08 C9 07 9E 20 32 30  7A
 008 : 36 34 3A A2 00 00 00 A2  81
C010 : 08 A0 01 84 FB 88 84 FD  4B
C018 : A9 08 85 FC A9 C0 85 FE  7B
C020 : B1 FB 91 FD C8 D0 F9 E6  BC
C028 : FC E6 FE CA D0 F2 A2 10  FF
C030 : 8E 20 D0 8E 21 D0 CA 8E  B5
C038 : 86 02 A2 80 8D 8A 02 A9  00
C040 : 93 20 16 E7 4C 47 C0 A9  BB
C048 : 52 8D 08 03 A9 C0 8D 09  AC
C050 : 03 60 A5 7A 8D 73 C7 A5  1B
C058 : 7B 8D 74 C7 A2 00 20 73  41
C060 : 00 DD 75 C7 D0 08 E8 E0  58
C068 : 03 D0 F3 4C 54 C2 AD 73  53
C070 : C7 85 7A AD 74 C7 85 7B  E1
C078 : A2 00 20 73 00 DD 78 C7  F1
C080 : D0 08 E8 E0 04 D0 F3 4C  DA
C088 : 5F C2 AD 73 C7 85 7A AD  10
C090 : 74 C7 85 7B A2 00 20 73  4A
C098 : 00 DD 7C C7 D0 08 E8 E0  52
C0A0 : 06 D0 F3 4C 7D C2 AD 73  20
C0A8 : C7 85 7A AD 74 C7 85 7B  19
C0B0 : A2 00 20 73 00 DD 82 C7  51
C0B8 : D0 08 E8 E0 06 D0 F3 4C  32
C0C0 : 9F C2 AD 73 C7 85 7A AD  88
C0C8 : 74 C7 85 7B A2 00 20 73  82
C0D0 : 00 DD 89 C7 D0 0B E8 E0  E5
C0D8 : 04 D0 F3 A9 00 48 4C 04  F2
C0E0 : C3 AD 73 C7 85 7A AD 74  1B
C0E8 : C7 85 7B A2 00 20 73 00  74
C0F0 : DD 88 C7 D0 0B E8 E0 05  A3
C0F8 : D0 F3 A9 FF 48 4C 04 C3  AB
C100 : AD 73 C7 85 7A AD 74 C7  80
C108 : 85 7B A2 00 20 73 00 DD  4D
C110 : 8D C7 D0 08 E8 E0 07 D0  09
C118 : F3 4C C0 C2 AD 73 C7 85  5A
C120 : 7A AD 74 C7 85 7B A2 00  46
```

```
C000 C7FF----------------------------

C128 : 20 73 00 DD 94 C7 D0 0B  9E
C130 : E8 E0 06 D0 F3 A9 FF 48  41
C138 : 4C 2E C3 AD 73 C7 85 7A  C2
C140 : AD 74 C7 85 7B A2 00 20  D7
C148 : 73 00 DD 9A C7 D0 0E E8  93
C150 : E0 04 D0 F3 A9 00 48 8D  BC
C158 : FE C7 4C 2E C3 AD 73 C7  1A
C160 : 85 7A AD 74 C7 85 7B A2  F8
C168 : 00 20 73 00 DD 9E C7 D0  E9
C170 : 08 E8 E0 06 D0 F3 4C 73  AA
C178 : C3 AD 73 C7 85 7A AD 74  B3
C180 : C7 85 7B A2 00 20 73 00  0C
C188 : DD A4 C7 D0 08 E8 E0 04  17
C190 : D0 F3 4C 9E C3 AD 73 C7  48
C198 : 85 7A AD 74 C7 85 7B A2  30
C1A0 : 00 20 73 00 DD A9 C7 D0  79
C1A8 : 08 E8 E0 05 D0 F3 4C 17  09
C1B0 : C4 AD 73 C7 85 7A AD 74  EC
C1B8 : C7 85 7B A2 00 20 73 00  44
C1C0 : DD AE C7 D0 08 E8 E0 02  50
C1C8 : D0 F3 4C 3F C4 AD 73 C7  A4
C1D0 : 85 7A AD 74 C7 85 7B A2  68
C1D8 : 00 20 73 00 DD B0 C7 D0  E9
C1E0 : 10 E8 E0 06 D0 F3 A9 00  B0
C1E8 : 48 A9 FF 8D FE C7 4C 2E  72
C1F0 : C3 AD 73 C7 85 7A AD 74  2B
C1F8 : C7 85 7B A2 00 20 73 00  84
C200 : DD B6 C7 D0 08 E8 E0 04  98
C208 : D0 F3 4C 8D C4 AD 73 C7  AE
C210 : 85 7A AD 74 C7 85 7B A2  A8
C218 : 00 20 73 00 DD BA C7 D0  7A
C220 : 08 E8 E0 05 D0 F3 4C 37  C1
C228 : C2 AD 73 C7 85 7A AD 74  62
C230 : C7 85 7B 4C E4 A7 00 20  EE
C238 : 73 00 20 9E B7 8E 36 C2  D5
C240 : AE 36 C2 E0 00 F0 0A CA  1B
C248 : 8E 36 C2 20 64 C7 4C 40  DC
C250 : C2 4C AE A7 A9 93 20 0C  A9
C258 : E1 20 73 00 4C AE A7 20  3F
C260 : 73 00 20 9E B7 8E 20 D0  C1
C268 : 8E D6 C7 20 F1 B7 8E 21  B1
C270 : D0 8E D5 C7 20 F1 B7 8E  83
C278 : 86 02 4C AE A7 20 73 00  31
C280 : 20 9E B7 E0 28 90 03 4C  A5
C288 : 48 B2 8A 48 20 F1 B7 E0  07
C290 : 19 90 03 4C 48 B2 68 A8  49
C298 : 18 20 0A E5 4C AE A7 20  19
C2A0 : 73 00 20 9E B7 8E D2 C7  BA
C2A8 : 20 C6 C5 AE D2 C7 20 D9  12
C2B0 : C5 AE D2 C7 20 2A C6 AE  46
C2B8 : D2 C7 20 05 C6 4C AE A7  EF
C2C0 : 20 73 00 20 9E B7 8E D2  25
C2C8 : C7 20 C6 C5 AE D2 C7 20  EA
C2D0 : D9 C5 20 5A C6 AC D9 C7  A8
C2D8 : C8 C8 8C D9 C7 20 EC C6  21
C2E0 : A9 0D 20 16 E7 A2 00 BD  EA
C2E8 : EB C7 C9 00 F0 07 20 16  1D
C2F0 : E7 E8 4C E7 C2 20 28 C7  B9
C2F8 : AE D2 C7 20 05 C6 20 73  F3
```

```
C300 : 00 4C AE A7 20 73 00 20  A5
C308 : 9E B7 8E D2 C7 20 C6 C5  A4
C310 : AE D2 C7 20 D9 C5 AE D2  49
C318 : C7 68 D0 06 20 8A C6 4C  12
C320 : 25 C3 20 B6 C6 AE D2 C7  C2
C328 : 20 01 C6 4C AE A7 20 73  93
C330 : 00 20 8A AD 20 F7 B7 20  79
C338 : FD AE 20 9E B7 E0 08 90  2C
C340 : 03 4C 48 B2 E8 A9 01 CA  47
C348 : E0 00 F0 04 0A 4C 47 C3  8D
C350 : A8 68 D0 0F AD FE C7 D0  D6
C358 : 12 98 41 14 21 14 81 14  6A
C360 : 4C AE A7 98 01 14 81 14  DF
C368 : 4C AE A7 98 41 14 81 14  EB
C370 : 4C AE A7 20 73 00 A2 FA  B9
C378 : BD FF 03 49 80 9D FF 03  1A
C380 : BD F9 04 49 80 9D F9 04  49
C388 : BD F3 05 49 80 9D F3 05  78
C390 : BD ED 06 49 80 9D ED 06  A7
C398 : CA D0 DD 4C AE A7 20 73  5B
C3A0 : 00 20 73 00 C9 91 D0 03  00
C3A8 : 4C C5 C3 C9 4F D0 11 20  01
C3B0 : 73 00 C9 46 D0 0A 20 73  23
C3B8 : 00 C9 46 D0 03 4C FB C3  53
C3C0 : A2 0B 4C 37 A4 20 73 00  FB
C3C8 : 78 A9 D0 8D 14 03 A9 C4  84
C3D0 : 8D 15 03 A9 01 8D 1A D0  64
C3D8 : A9 7F 8D 0D DC AD D3 C7  60
C3E0 : 8D 12 D0 AD 11 D0 29 7F  9B
C3E8 : 8D 11 D0 AD 20 D0 8D D6  54
C3F0 : C7 AD 21 D0 8D D5 C7 58  48
C3F8 : 4C AE A7 20 73 00 78 A9  F6
C400 : 31 8D 14 03 A9 EA 8D 15  B0
C408 : 03 A9 00 8D 1A D0 A9 81  63
C410 : 8D 0D DC 58 4C AE A7 20  7F
C418 : 73 00 20 9E B7 E0 01 90  0F
C420 : 1B E0 18 B0 17 CA 8A 0A  CD
C428 : 0A 0A 18 69 37 AA CA CA  F4
C430 : CA 8E D4 C7 CA CA 8E D3  54
C438 : C7 4C AE A7 4C 18 B2 20  D8
C440 : 73 00 20 9E B7 8E DA C4  74
C448 : 20 F1 B7 8E EE C4 20 F1  9A
C450 : B7 8E 02 C5 20 F1 B7 8E  15
C458 : 11 C5 20 F1 B7 8E 22 C5  96
C460 : 20 F1 B7 8E 4B C5 20 F1  80
C468 : B7 8E 5F C5 20 F1 B7 8E  84
C470 : 6E C5 20 F1 B7 8E 7F C5  81
C478 : 20 F1 B7 8E 8E C5 20 F1  CC
C480 : B7 8E FD C7 20 F1 B7 8E  84
C488 : FC C7 4C AE A7 20 73 00  9A
C490 : A2 00 8A 9D 00 D4 E8 E0  95
C498 : 17 D0 F8 A9 0F 8D 18 D4  F2
C4A0 : A9 CF 8D 00 D4 A9 22 8D  D2
C4A8 : 01 D4 A9 83 8D 06 D4 A9  9E
C4B0 : 33 8D 05 D4 A9 21 8D 04  68
C4B8 : D4 A0 80 A2 00 E8 D0 FD  D7
C4C0 : C8 D0 FA A9 00 8D 04 D4  0A
C4C8 : A9 00 8D 18 D4 4C AE A7  91
C4D0 : AE D3 C7 8E 12 D0 AE D4  38
C4D8 : C7 A9 01 EC 12 D0 D0 FB  35

C4E0 : A0 00 C8 C0 11 D0 FB 8D  6D
C4E8 : 20 D0 8D 21 D0 A9 0F A0  D0
C4F0 : 00 C8 C0 06 D0 FB EA EA  B4
C4F8 : EA EA EA 8D 20 D0 8D 21  C5
C500 : D0 A9 0C A0 00 C8 C0 07  13
C508 : D0 FB 8D 20 D0 8D 21 D0  DD
C510 : A9 0B A0 00 C8 C0 07 D0  B7
C518 : FB EA EA 8D 20 D0 8D 21  F6
C520 : D0 A9 00 A0 00 C8 C0 07  30
C528 : D0 FB 8D 20 D0 8D 21 D0  FD
C530 : AD FD C7 AE FC C7 A0 00  34
C538 : C8 C0 07 D0 FB 8D 20 D0  8A
C540 : 8E 21 D0 AD D4 C7 18 69  07
C548 : 0D AA A9 00 EC 12 D0 D0  59
C550 : FB A0 00 C8 C0 11 D0 FB  84
C558 : 8D 20 D0 8D 21 D0 A9 0B  30
C560 : A0 00 C8 C0 01 D0 FB 8D  EC
C568 : 20 D0 8D 21 D0 A9 0C A0  44
C570 : 00 C8 C0 07 D0 FB EA EA  54
C578 : 8D 20 D0 8D 21 D0 A9 0F  58
C580 : A0 00 C8 C0 07 D0 FB 8D  6C
C588 : 20 D0 8D 21 D0 A9 01 A0  37
C590 : 00 C8 C0 07 D0 FB EA EA  74
C598 : 8D 20 D0 8D 21 D0 AD D6  18
C5A0 : C7 AE D5 C7 A0 00 C8 C0  DB
C5A8 : 07 D0 FB 8D 20 D0 8E 21  CD
C5B0 : D0 EE 19 D0 4C 31 EA C9  E5
C5B8 : 2F B0 03 4C 08 AF C9 3A  23
C5C0 : 90 03 4C 08 AF 60 E0 08  78
C5C8 : B0 05 A2 09 4C 37 A4 E0  97
C5D0 : 0C 90 05 A2 09 4C 37 A4  D3
C5D8 : 60 A9 0F A0 0F 20 BA FF  C2
C5E0 : A9 01 A2 C0 A0 C7 20 BD  0F
C5E8 : FF 20 C0 FF C9 05 D0 0B  45
C5F0 : 48 A9 0F 20 C3 FF 68 AA  08
C5F8 : 4C 37 A4 A9 0F 20 C3 FF  3F
C600 : 60 A9 00 85 B7 A9 0F A8  3C
C608 : 20 BA FF 20 C0 FF A2 0F  3E
C610 : 20 C6 FF 20 CF FF B8 C9  0B
C618 : 0D F0 06 20 16 E7 4C 13  1B
C620 : C6 20 CC FF A9 0F 20 C3  44
C628 : FF 60 A9 0D 20 16 E7 A2  FB
C630 : 00 BD C1 C7 C9 00 F0 07  E6
C638 : 20 16 E7 E8 4C 31 C6 68  B4
C640 : AA 48 A9 0F A8 20 BA FF  D1
C648 : A9 0B A2 BF A0 C7 20 BD  5C
C650 : FF 20 C0 FF A9 0F 20 C3  AA
C658 : FF 60 20 FD AE 20 44 C7  DC
C660 : A9 22 20 FF AE A0 00 8C  2B
C668 : D9 C7 C9 2C F0 18 C9 22  58
C670 : F0 14 AC D9 C7 C0 10 F0  75
C678 : 0D 99 02 C8 C8 8C D9 C7  D3
C680 : 20 73 00 4C 67 C6 20 4F  AF
C688 : C7 60 20 CB C6 20 5A C6  65
C690 : 20 E1 C6 AC D9 C7 A9 2C  C3
C698 : 99 02 C8 C8 AD D7 C7 99  69
C6A0 : 02 C8 C8 AD D8 C7 99 02  24
C6A8 : C8 C8 C8 C8 8C D9 C7 20  16
C6B0 : F7 C6 20 73 00 60 20 5A  B9
C6B8 : C6 20 E1 C6 AC D9 C7 C8  2A
```

```
C6C0 : C8 8C D9 C7 20 F7 C6 20  5B
C6C8 : 73 00 60 20 FD AE 20 B7  9D
C6D0 : C5 8D D7 C7 20 73 00 20  28
C6D8 : B7 C5 8D D8 C7 20 73 00  3B
C6E0 : 60 A9 4E 8D 00 C8 A9 3A  BB
C6E8 : 8D 01 C8 60 A9 53 8D 00  9F
C6F0 : C8 A9 3A 8D 01 C8 60 A9  F8
C6F8 : 0D 20 16 E7 A2 00 BD DA  6F
C700 : C7 C9 00 F0 07 E8 20 16  2E
C708 : E7 4C FE C6 A9 0F A8 AE  C1
C710 : D2 C7 20 BA FF AD D9 C7  8A
C718 : A2 00 A0 C8 20 BD FF 20  2C
C720 : C0 FF A9 0F 20 C3 FF 60  0D
C728 : AE D2 C7 A9 0F A8 20 BA  93
C730 : FF AD D9 C7 A2 00 A0 C8  B3
C738 : 20 BD FF 20 C0 FF A9 0F  0B
C740 : 20 C3 FF 60 A9 EA 85 80  57
C748 : 85 81 85 82 85 83 60 AD  91
C750 : 6F C7 85 80 AD 70 C7 85  9D
C758 : 81 AD 71 C7 85 82 AD 72  0D
C760 : C7 85 83 60 A2 70 A0 47  95
C768 : CA D0 FD 88 D0 FA 60 C9  25
C770 : 20 F0 EF 00 00 43 4C 53  F6
C778 : 43 4F 4C B0 4C 4F 43 41  5B
C780 : 54 45 C5 49 44 41 54 45  3C
C788 : 51 81 4D 41 54 53 43 52  A7
C790 : 41 54 43 48 42 49 54 53  3C
C798 : 45 54 42 49 54 9C 49 4E  AD
C7A0 : 56 53 43 52 42 B0 44 45  00
C7A8 : 52 42 4C 49 4E 45 42 83  77
C7B0 : 42 49 54 43 48 47 42 45  67
C7B8 : 45 50 50 41 55 53 45 56  13
C7C0 : 20 56 41 4C 49 44 41 54  4A
C7C8 : 49 4E 47 20 44 49 53 4B  81
C7D0 : 0D 00 00 32 34 00 00 00  67
C7D8 : 00 00 46 4F 52 4D 41 54  91
C7E0 : 54 49 4E 47 20 44 49 53  45
C7E8 : 4B 0D 00 53 43 52 41 54  99
C7F0 : 43 48 49 4E 47 20 46 49  95
C7F8 : 4C 45 0D 00 00 00 00 FF  2A
```

JIFFY DOS

To, co nezvládne Commodore v základní verzi, musí vyřídit za něj ostatní. V plné míře toto tvrzení platí o flopy–speedrech. Speciálně pro své přístroje a nejen pro ně vyvinula firma CMD speeder Jiffy DOS. Zda splňuje i vaše představy o solidním speederu posuďte sami.

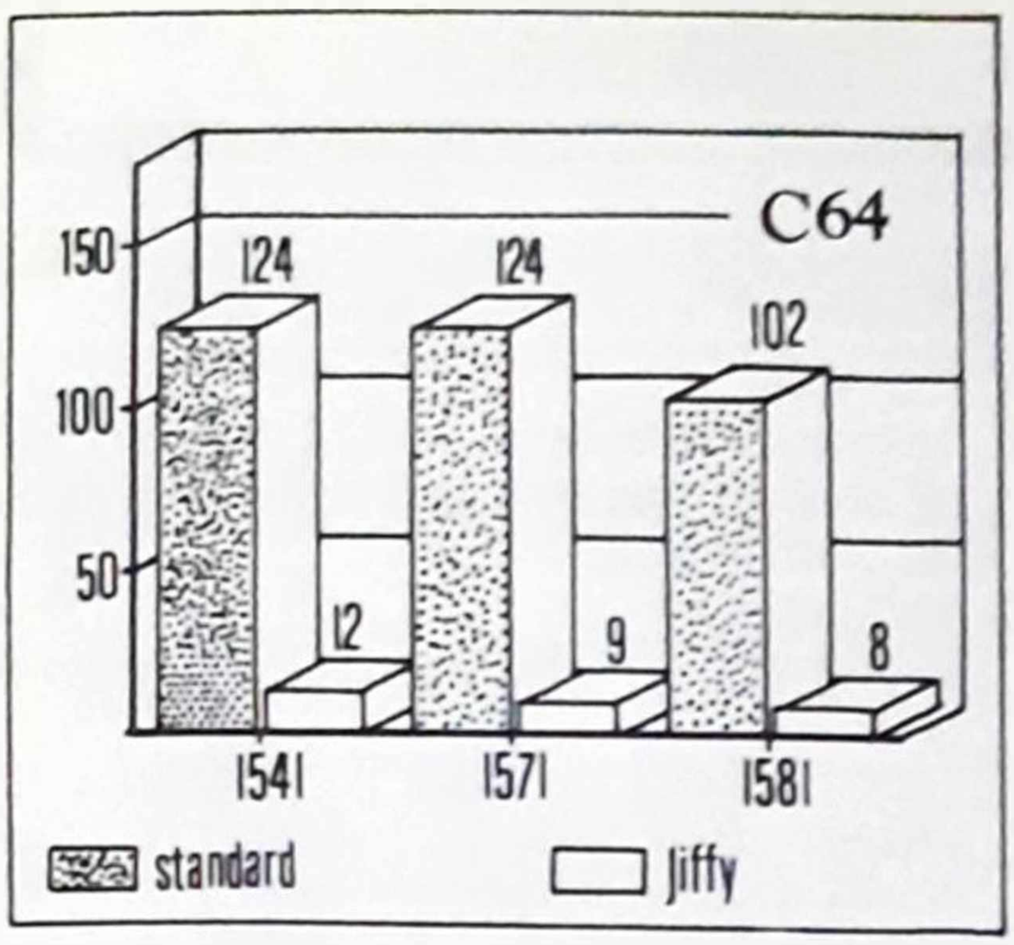
Uživatelé commodoráckých drajvů se musí při loadingu a zpracovávání delších programů obrnit notnou dávkou trpělivosti a počítat s tím, že bude nucená přestávka na kávu. Z uvedeného důvodu mají výrobci speedrů na trhu s C64 své pevné místo. Výjimkou není ani firma CMD, jež je známa spíše solidními hardwarovými doplňky. Její produkt Jiffy DOS zapadá do kategorie speedrů. Na rozdíl od mnoha jiných, pracuje Jiffy DOS pouze na sériovém busu, tedy bez paralelního kabelu mezi počítačem a drajvem. Proto je i jeho zabudování do počítače např. oproti Prologic DOS jednodušší. Ostatně, pokud jde o tuto operaci, redukuje se na "pouhou výměnu" systémových ROM v C64 a drajvu. Pokud jsou vaše přístroje u zmíněných obvodů vybaveny navíc paticí, můžete si spokojeně vydechnout. Předpokládám, že u větší části C64 tomu tak není. Potom nezbude než vyletování starých švábů nechat odborníkům.

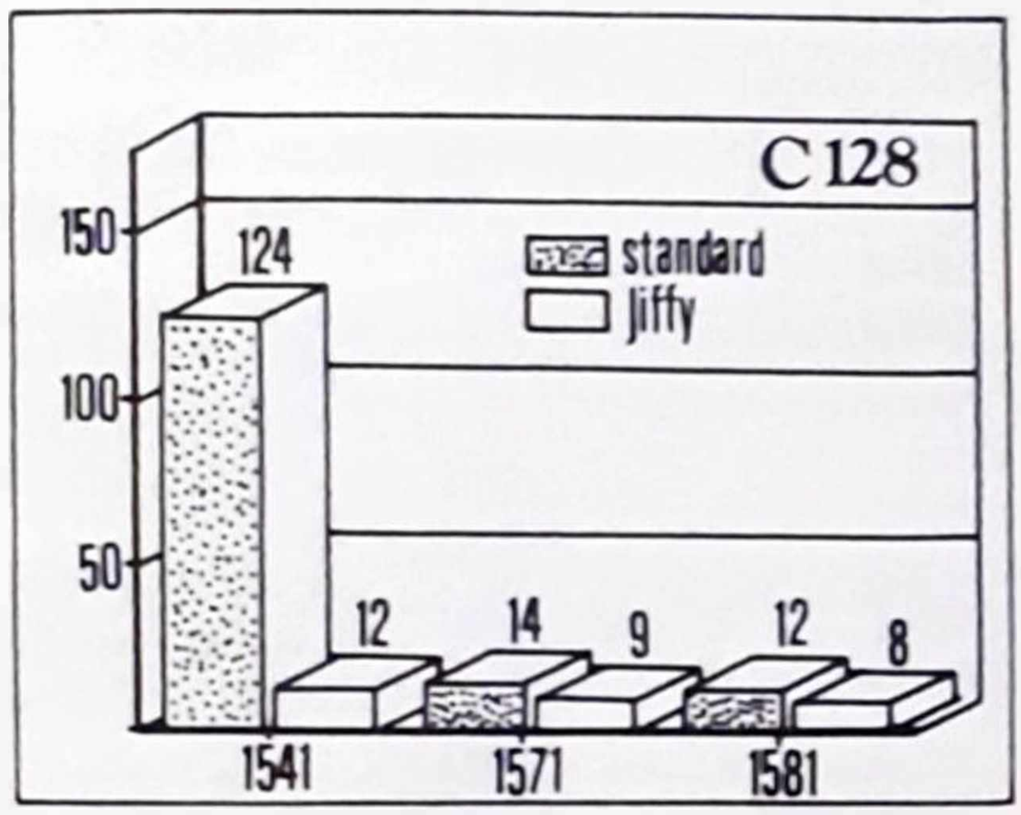
Po zabudování nových eprom se musí ještě do pouzder přístrojů vyvrtat díry pro přepínače, jimiž se přepíná mezi speedrem a normálním provozem. Celou práci usnadňuje podrobný návod (v němčině, angličtině) s obrázky a popisem jednotlivých verzí počítače s přesným umístěním obvodů, které mají být vyměněny.

Po zapnutí počítače lze z obměněného menu vidět, že Jiffy DOS je připraven. Na době loadingu to pocítíte také. Výkonu paralelního systému sice nedosahuje, ale urychlovací faktor 10–13 je pro sériový systém velmi slušný. Při ukládání programů činí zrychlovací faktor 3 až 4 násobek běžné hodnoty.

Největší výhodou Jiffy DOS je podpora všech typů drajvů, tedy 1541, 1571 a 1581. Pro všechny přístroje lze objednat novou ROM. Rozdíly v rychlosti jednotlivých drajvů byly sotva postřehnutelné. Rovněž tak nepřijde zkrátka žádný z typů počítačů C64 (C64, SX64, C128, C128–D). Speeder urychluje i HD a FD 4000 (2000).

Mimo urychlovače disketových mechanik obsahuje Jiffy DOS účinné příkazy pro práci s datovými soubory. Vedle kompletního DOS 5.1 jsou to kopírovací rutiny pro všechny dru-





hy datových souborů. Typ datových souborů je možno navíc měnit.
Ve shodě s ostatními speedery i Jiffy DOS obsazuje funkční klávesy nejdůležitějšími příkazy. Soubor završují neobvyklé příkazy umožňující listing programů přímo z diskety. Ve zvláštním odstavci na konci řádku je uveden krátký výpis příkazů. Co se týče množství, komfortu a účinnosti příkazů, lze Jiffy DOS hodnotit jako velmi dobrý. Příručky v anglickém a německém jazyce všechny příkazy na jednoduchých příkladech důkladně osvětlují. Tato skutečnost jistě potěší jak začátečníky, tak i profesionály.

Na závěr zůstalo jediné téma. Kompatibilita. U jiných speederů se stává poměrně často, že zrovna program o který stojíte, si s rozšířením nerozumí. Musíme přiznat, že si Jiffy DOS neporadil se zdánlivě triviálním kopírovacím programem Hypra copy. Program byl sice bez problému natažen, avšak po spuštění se zablokoval. Naopak hry, obsahující důmyslné ochrany proti kopírování, jinak velmi kritické, se dají bez problémů natahovat a startovat. Ostatně o této skutečnosti hovoří i příručka CMD.
Shrneme-li tedy poznatky, nezbývá než konstatovat, že Jiffy DOS představuje solidní speeder s dobrými urychlovacími účinky. Navíc obsahuje řadu užitečných příkazů pro práci s datovými soubory všech druhů na disketách. Zabudování součástek s Jiffy DOS do počítače a programování doprovází dobrá anglická dokumentace s mnoha příklady. Jiffy DOS se tedy vyplatí.

Tab.1 Porovnání rychlosti diskových operací různých typů drajvů připojených k C64, SX64, C128 v módu C64

	1541		1571		1581	
disková operace	orig.	Jiffy	orig.	Jiffy	orig.	Jiffy
loading programu202 bloků	124	12	124	9	102	8
saving programu100 bloků	75	24	75	20	40	15
čtení 125 bloků SEQ nebo USR	84	15	84	13	63	9
zápis 100 bloků SEQ	81	27	81	24	44	17

Přehled příkazů Jiffy DOS

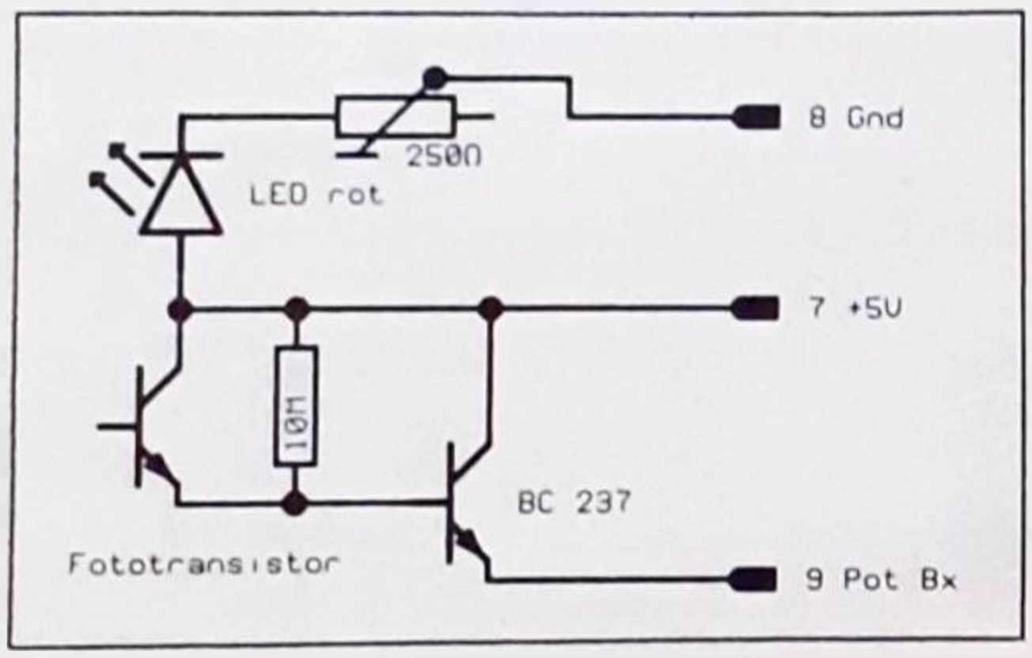
čtení chybového kanálu	@
zobrazení directory	@$
změna čísla přístroje	@#device
/file	loading programu
%file	loading strojového programu
*file type	kopírování souboru

pozn. Veškeré produkty CMD jsou i v nabídce firmy Comotronic.
(M)

POSTAVTE SI SCANER

Scaner nemusí být bezpodmínečně drahým přístrojem. S trochou technické šikovnosti se dá postavit i doma.

Ukažte mi skutečného počítačového fandu, ktery by nesnil o tom, aby mohl načíst obrázky do počítače a potom je podle vlastní představy modifikovat! Naznačenou operaci umožňují scanery. Pomocí nich se dají převést do počítače takové předlohy jako jsou texty nebo obrázky. Pomocí vhodného programu pro malování lze potom obrázek kdykoliv změnit popřípadě nechat vytisknout. Žel ty pravé scanery nejsou zrovna levnou záležitostí. Proto nabízím alternativní řešení vlastního bastlu podle 64'er. Scaner využívá pohonu tiskové hlavy. Nezbytnou elektroniku představují běžně dostupné součástky. Jako držák pro hlavu scaneru slouží použitá kazeta pro barvící pásku. Nastavení je nutno provést pouze jedenkrát. Při dalším scanování se kazeta pouze nasune a optimální výsledek je v každém případě zaručen. Dále popsané řešení se bez úprav týká pouze Star LC 10. U jiných tiskáren musí být eventuelně přizpůsoben software.



Software

Scaner se připojuje na joystickový port 1.
Aby se dal obrázek scanovat ve čtyřech stupních jasu, musí se naládovat program "SCANNY.BAS". Program se natáhne automaticky po zavedení a odstartování hlavního programu psaného v basicu. Po instalaci obou strojových rutin se objeví hlavní menu:

1. Test–scan
   Tento bod kontroluje jas světelné diody. Bílý papír podržený před scanovací hlavou musí dávat minimální hodnoty.
2. Nové zadání dat
   Nejprve se stanoví kontrasty pro odstupňování barvy.
   Počítač přiřazuje hodnotám různé stupně šedi. Všechna měření, která jdou hodnotou přes horní hranici, budou zobrazena jako černá. Od střední po horní hranici zobrazí počítač hodnoty jako tmavěšedé, mezi dolní a střední hodnotou jako světlešedé a všechny hodnoty pod touto hranicí zůstanou bílé.
   Potom následuje stanovení odstupu řádků a délky řádku. Při každém řádkovém posuvu otočí tiskárna válec o N/216 inch.

   Délka řádku – udává kolik kroků jede tisková hlava napravo (u malých obrázků se vhodným stanovením délky řádku ušetří spousta času).

   Rychlost načítání – Počet taktovacích cyklů mezi dvěma body, které se načítají. Čím větší je číslo, tím menší je obrázek.

   Levý okraj – Zpoždění začátku načítání je nutné v případě, že předloha nezačíná přesně na levém kraji. Protože při větších zpožděních často dojde k lehké deformaci, mělaby hodnota být pokud možno 0.

   Ukládání – Zadané hodnoty se dají ukládat na disketu. Při novém startu programu se automaticky dotáhnou.
3. Zobrazení grafiky
   Ukáží se obrázky sejmuté scanerem.
4. Scanning
   Pokud je tiskáma přepnuta na "on–line" začne hned počítač založený obrázek scanovat. Ve zřídkavých případech může dojít k havarii programu. Potom pomůže restart.
5. Ukládání
   obrázku ve formátu Koala.
6. Hranice barev – změna

Další hodnoty barev zvolíme v bodě 2 mohou být dodatečně změněny. Každý bod je složen z osmi jako 8 bitové číslo. Po zadání nových mezí se zakázko obrázek znovu přenáší. Kontrasty se mohou též dodatečně optimalizovat. Je-li vysoká světlá hrana, může se pomocí zdání vhodné hodnoty zjasnit. Pokud vystupuje špatné pouze jedna barva, změní se jen odpovídající hranice barev.

7 Změna barev

Namísto různých tónů šedi se mohou nasadit také jiné barvy.

Program odpovídá shora uvedenému pro černobílé zobrazení. Dvoubarevné obrázky mohou být scanovány v módu vysokého rozlišení, tedy s dvojnásobným rozlišením v X-směru. V menu zvolím chybí bod " Změna hranice barev ", neboť pro nová obrazová data by se muselo přepočítat celých 64 kbajtů.

Všechny DIP-přepínače IC 10 nastavit na ON. Pro ostatní tiskárny dbejte upozornění: signál Line Feed musí být vysílán počítačem.

Stavba

Nejdříve si připravte přípojku pro joystickový port, potřebujete 9 pólový konektor Canon 9. Na kolík 7 (+5V) a 9 (pot 5x) přiletujte oba stíněné vodiče. Stínění přijde na kolík 8 (GND). Druhý konec kabelu přijde na vlastní scanovací hlavu. Dejte pozor, aby kabel nebyl příliš krátký. Hlava scaneru se musí i s kabelem volně pohybovat nad celou šířkou papíru.

Detekční obvod scaneru se dá postavit na universální desce plošných spojů. Pouzdro zhotovíte ze staré kazety s barvicí páskou. Dále si opatřete dva cca 8cm dlouhé kousky světlovodu. Jeden připevněte koncem k červené diodě LED (černá lepicí páska). Fototranzistor překryjte kouskem pryžové hadičky. Dbejte na to, aby překrytí před nežádoucím světlem bylo dokonalé. Do hadičky udělejte špendlíkem díru

pro zastrčení konce druhého světlovodu. Oba volné konce světlovodů svažte dohromady do vlastní scanovací hlavy a dobře připevněte. Nezapomeňte také oba kabliky dokonale světelně oddělit. Obrázek leckdy mnohé napoví. V pravém rohu jsou svedeny oba konce světlovodů. Pro scanovací hlavu jsme použili vyřazenou čtečku čárového kódu.

Máte-li všechno hotovo, naládujte program a v menu zvolte bod 1 – test scan. Pro optimální nastavení světlovodu použijte bílý papír. Podržte jej před čtecí hlavou a posunujte světlovodičem tak dlouho, až dostanete minimální hodnotu.

Opatrným otáčením odporového trimru nastavte jas diody. Optimální nastavení předpokládá hodnotu pod 20.

Podržíte-li nyní před čtecí hlavičkou černý papír, měla by se zobrazit hodnota přes 100.

Po tomto nastavení zbývá elektroniku zabudovat do prázdné kazety. Nakonec experimentálně odzkoušejte vzdálenost čtecí hlavičky od papíru a na optimálním místě ji přilepte na kazetu.

Listing

```
100 IF PEEK (2047)=22 THEN 130                    <166>
110 Q=Q+1:IF Q=1 THEN LOAD
    "SCANNY ASS",8,1                              <107>
120 POKE 2047,22: IF Q=2 THEN LOAD
    "SCANNY CEN",8,1                              <008>
130 F1=1: F4=0: OPEN 1,8,0,
    "WERTE":INPUT#1,G1,G2,G3, Z,ZL,X,V:
    CLOSE 1                                       <121>
140 POKE 53281,0: POKE 53280,0: POKE
    56325,60: POKE 650,127                        <191>
150 PRINT"CLR,WHITE,2DOWN,6SPACE(1)
    PROBESCANN"                                   <026>
160 PRINT"2DOWN,6SPACE(2) DATEN
    NEUEINGABE"                                   <125>
170 PRINT"2DOWN,6SPACE(3) GRAFIK
    BETRACHTEN"                                   <071>
180 PRINT"2DOWN,6SPACE(4) EINSCANNEN"
                                                  <002>
190 PRINT"2DOWN,6SPACE(5) BILD
    ABSPEICHRN"                                   <007>
200 PRINT"2DOWN,6SPACE(6) FARBENGRENZEN
    AENDERN"                                      <128>
210 PRINT"2DOWN,6SPACE(7) FARBEN
    AENDERN"                                      <254>
230 GET A$: IF A$="1" THEN 670                    <137>
240 IF A$="2" THEN 310                            <042>
250 IF A$="3" THEN 600                            <184>
260 IF A$="4" THEN 420                            <077>
270 IF A$="5" THEN 540                            <235>
280 IF A$="6" THEN 630                            <114>
```

```
290 IF A$="7" THEN 730                                  <000>
300 GOTO 230                                            <030>
310 PRINT"CLR,3DOWN,GREY 2,3SPACE
    OBERGRENZE4SPACE (MAX. 255)
    3SPACE"G1,:INPUT"6LEFT";G1                          <067>
320 PRINT"DOWN,GREY 3,3
    SPACEMITTELGRENZE 3SPACE(MAX."G1"
    LEFT)4SPACE"G2;:INPUT"6LEFT";G2                     <237>
330 PRINT" DOWN,WHITE,3SPACE
    UNTERGRENZE3SPACE(MAX."G2"LEFT,)
    5SPACE"G3;: INPUT"6LEFT";G3                         <131>
340 PRINT"DOWN,3SPACEZEILENABSTAND
    5SPACE(UM 5)3SPACE"Z;:
    INPUT"4LEFT";Z                                      <225>
350 PRINT"DOWN,3SPACEZEILENLAENGE
    5SPACE(UM 55)
    3SPACE"ZL;:INPUT"5LEFT";ZL                          <035>
360 PRINT"DOWN,3SPACEEINLESEGESCHW.
    (UM 3000)3SPACE"X;:INPUT"7LEFT";X                   <188>
370 PRINT"DOWN,3SPACELINKER RAND
    4SPACE(MAX. 99)
    3SPACE"V;:INPUT"5LEFT";V                            <238>
380 PRINT"2DOWN,3SPACEWERTE SPEICHER
    (J/N)"                                              <022>
390 GET A$:IF A$="J" THEN 580                           <053>
400 IF A$="N" THEN 140                                  <234>
410 GOTO 390                                            <244>
420 POKE 53281,F1: POKE 53270,216                       <206>
430 POKE 49330,G1: POKE 49334,G2:
    POKE 49338,G3                                       <081>
440 POKE 646,F4: PRINT"CLR":Y=INT(X/256):
    POKE 49280,Y: POKE 49285,(X-Y*256)
                                                        <105>
445 SYS 49152                                           <249>
450 SYS 52224                                           <246>
452 OPEN 1,5                                            <053>
465 PRINT #1,CHR$(27)CHR$(51)CHR$(Z)
                                                        <170>
468 PRINT#1,CHR$(27)CHR$(196)CHR$(ZL)
    CHR$(0)                                             <153>
470 PRINT #1,CHR$(9)".":PRINT#1,"."                     <119>
480 PRINT #1,CHR$(9)"."                                 <099>
490 FOR Y=1 TO V:NEXT:POKE 709,0                        <081>
500 PRINT #1,"."                                        <131>
510 PRINT #1,CHR$(10)                                   <201>
520 IF PEEK(708)=0 THEN480                              <193>
530 POKE 53265,27:POKE 53272,21:
    CLOSE 1:GO TO 140                                   <169>
540 INPUT"CLR,6DOWN,4SPACEFILENAME";
    NA$                                                 <139>
550 A$="ORANGEPIC "+NA$:FOR Y=
    0 TO 14-LEN(A$):A$=A$+" ":NEXT Y                    <016>
560 POKE 49529,F4:SYS 49487:POKE
    26384,F1:SYS 57812A$,8                              <172>
570 POKE 193,0:POKE 194,64:POKE 174,17:
    POKE175,103:SYS 62957:GOTO 140                      <022>
580 OPEN1,8,1:"@:WERTE":PRINT#1,
    G1,",",G2,",",G3,",",Z                              <214>
590 PRINT#1,ZL,",",X,",",V:
    CLOSE1: GOTO 140                                    <253>
600 POKE 646,F4:PRINT"CLR":POKE 198,0:
    SYS 49546                                           <170>
610 POKE 53281,F1: WAIT 198,1:
    POKE 53270,200: GOTO 530                            <078>
630 PRINT"CLR,3DOWN,GREY 2,3
    SPACEOBERGRENZE
    4SPACE(MAX. 255)
    4SPACE"G1;:INPUT"6LEFT";G1                          <133>
640 PRINT"DOWN,GREY 3,3
    SPACEMITTELGRENZE2 SPACE
    (MAX. "G1"LEFT)5
    SPACE"G2;:INPUT"6LEFT";G2                           <047>
650 PRINT"DOWN,WHITE,
    3SPACEUNTERGRENZESPACE
    (MAX. "G2"LEFT)6SPACE"G3;:
    INPUT"6LEFT";G3                                     <197>
660 POKE 49631,G1: POKE 49635,G2:
    POKE49639,G3:
    SYS 49773:SYS 49581:GOTO 600                        <195>
670 PRINT"CLR,2DOWN,SPACEHELLES
    & DUNKLES PAPIER"                                   <235>
680 PRINT" VOR DEN SCANNER HALTEN.
    (TASTE)"                                            <024>
690 A=0:FOR K=1 TO 10:
    A=A+PEEK(54297):NEXT:A=A/10                         <015>
700 PRINT"HOME,5DOWN,SPACEWERT :
    4SPACE,4LEFT"INT(A)                                 <190>
710 GET A$: IF A$=" " THEN 690                          <045>
720 GOTO 140                                            <204>
730 INPUT"CLR,2DOWN,2SPACEFARBE
    13SPACE(SCHWARZ)";F4                                <150>
740 INPUT"2DOWN,2SPACEFARBE2
    (DKL. GRAU)";F2                                     <073>
750 INTUP"2DOWN,2SPACEFARBE
    32SPACE(HELLGRAU)";F3                               <123>
760 INPUT"2DOWN,2SPACEFARBE
    45SPACE(WEISS)";F1                                  <233>
770 POKE 49189,F3*16+F2:
    POKE 49547,F3*16+F2:
    POKE49510,F3*16+F2                                  <017>
780 GOTO 600                                            <242>
```

PROGRAM : SCANNY.ASS

```
C000 : A9 00 8D C4 02 85 F7 A9  25
C008 : 20 85 F8 8D C5 02 A9 40  6E
C010 : 85 A8 A0 00 84 A7 98 91  1C
C018 : F7 88 D0 FB E6 F8 A6 F8  CA
C020 : E0 40 90 F3 A9 FB A0 00  C0
C028 : 99 00 04 99 00 05 99 00  84
C030 : 06 99 E8 06 88 D0 F1 A9  28
C038 : 04 8D C0 02 A9 27 8D C1  01
C040 : 02 A9 07 8D C2 02 A9 18  9D
C048 : 8D C3 02 A9 00 85 F9 A9  D4
C050 : 20 85 FA A9 38 8D 11 D0  2D
C058 : A9 18 8D 18 D0 78 A9 6A  C0
C060 : 8D 14 03 A9 C0 8D 15 03  C0
C068 : 58 60 AD C4 02 0D C5 02  98
C070 : F0 0D A9 00 8D 05 DC A9  19
C078 : 80 8D 04 DC 4C 34 EA A9  C1
C080 : 0D 8D 05 DC A9 00 8D 04  0A
C088 : DC A9 00 85 F7 85 F8 A2  BE
C090 : 08 AD 19 D4 18 65 F7 85  E7
C098 : F7 90 02 E6 F8 CA D0 F1  42
C0A0 : 46 F8 66 F7 46 F8 66 F7  B1
C0A8 : 46 F8 66 F7 A5 F7 20 43  24
C0B0 : C1 C9 78 B0 0C C9 50 B0  3C
C0B8 : 0C C9 28 B0 0C A9 00 F0  B9
C0C0 : 0A A9 03 D0 06 A9 02 D0  D1
C0C8 : 02 A9 01 85 02 CE C0 02  2D
C0D0 : 30 19 A5 02 AC C0 02 88  20
C0D8 : 30 04 0A 0A 10 F9 85 02  B9
C0E0 : A0 00 B1 F9 05 02 91 F9  C6
C0E8 : 4C 31 EA A9 03 8D C0 02  60
C0F0 : CE C1 02 30 0E A5 F9 18  4C
C0F8 : 69 08 85 F9 90 02 E6 FA  B0
C100 : 4C D2 C0 A9 27 8D C1 02  05
C108 : A9 04 8D C0 02 8D C5 02  D6
C110 : CE C2 02 30 10 C6 FA A5  34
C118 : F9 38 E9 37 85 F9 B0 02  7D
C120 : C6 FA 4C 31 EA CE C3 02  D5
C128 : 30 13 A9 07 8D C2 02 A5  6F
C130 : F9 18 69 01 85 F9 90 02  1E
C138 : E6 FA 4C 31 EA EE C4 02  12
C140 : 4C 31 EA A0 00 91 A7 E6  EC
C148 : A7 F0 01 60 E6 A8 60 A9  3C
C150 : 20 85 60 A0 00 84 5F 0A  15
C158 : 85 5B 84 5A A9 60 85 59  5E
C160 : 84 58 20 BF A3 A9 FB A0  C9
C168 : 00 99 40 5F 99 40 60 99  81
C170 : 40 61 99 40 62 C8 D0 F1  63
C178 : A9 00 99 28 63 99 28 64  F9
C180 : 99 28 65 99 28 66 C8 D0  34
C188 : F1 60 A9 FB A0 00 99 00  04
C190 : 04 99 00 05 99 00 06 99  E6
C198 : E8 06 88 D0 F1 A9 3B 8D  34
C1A0 : 11 D0 A9 18 8D 18 D0 A9  B7
C1A8 : D8 8D 16 D0 60 A9 40 85  46
C1B0 : A8 A9 00 85 A7 85 F9 A9  BF
C1B8 : 20 85 FA A9 04 8D C0 02  42
C1C0 : A9 27 8D C1 02 A9 07 8D  3D
C1C8 : C2 02 A9 18 8D C3 02 A9  4B
C1D0 : 35 85 01 78 A0 00 B1 A7  37
```

C000 C284----------

```
C1D8 : E6 A7 D0 02 E6 A8 C9 78  D2
C1E0 : B0 0C C9 50 B0 0C C9 28  F6
C1E8 : B0 0C A9 00 F0 0A A9 03  15
C1F0 : D0 06 A9 02 D0 02 A9 01  34
C1F8 : 85 02 CE C0 02 30 19 A5  9B
C200 : 02 AC C0 02 88 30 04 0A  F7
C208 : 0A 10 F9 85 02 A0 00 B1  D2
C210 : F9 05 02 91 F9 4C D4 C1  17
C218 : A9 03 8D C0 02 CE C1 02  60
C220 : 30 0E A5 F9 18 69 08 85  F8
C228 : F9 90 02 E6 FA 4C FF C1  5C
C230 : A9 27 8D C1 02 A9 04 8D  A1
C238 : C0 02 CE C2 02 30 10 C6  75
C240 : FA A5 F9 38 E9 37 85 F9  F5
C248 : B0 02 C6 FA 4C D4 C1 CE  1A
C250 : C3 02 30 13 A9 07 8D C2  11
C258 : 02 A5 F9 18 69 01 85 F9  57
C260 : 90 02 E6 FA 4C D4 C1 A9  D0
C268 : 37 85 01 58 60 A0 00 A9  0C
C270 : 20 85 F8 84 F7 98 91 F7  9C
C278 : 88 D0 FB E6 F8 A6 F8 E0  AE
C280 : 40 90 F3 60 01 FF FF FF  21
```

CC00 CDC7----------

```
CC00 : A9 FF 8D 03 DD A9 1D 8D  27
CC08 : 26 03 A9 CC 8D 27 03 AD  2D
CC10 : 02 DD 09 04 8D 02 DD A9  77
CC18 : 00 8D 60 CE 60 86 FE 48  97
CC20 : EA A5 9A C9 04 F0 3E C9  11
CC28 : 06 F0 5D C9 05 F0 05 68  F4
CC30 : 58 4C CA F1 68 48 29 E0  CE
CC38 : 4A 4A 4A 4A 4A AA 68 29  71
CC40 : 1F 18 7D 5D CC 48 29 40  AB
CC48 : F0 1B 68 48 29 1F F0 15  62
CC50 : 18 69 05 29 20 D0 0E 68  15
CC58 : 49 20 48 D0 08 00 20 40  5F
CC60 : 60 80 A0 60 A0 68 8D 01  BA
CC68 : DD AD 0D DD AD 00 DD 29  BF
CC70 : FB 8D 00 DD A9 10 2C 0D  D3
CC78 : DD F0 FB AD 00 DD 09 04  9D
CC80 : 8D 00 DD 18 58 A6 FE 60  FF
CC88 : 68 48 C9 0D F0 64 C9 FF  82
CC90 : D0 04 68 A9 7E 48 20 B4  65
CC98 : E6 29 60 D0 2B A5 D4 D0  1A
CCA0 : 1B 68 C9 12 D0 02 85 C7  67
CCA8 : C9 92 D0 04 A9 00 85 C7  AF
CCB0 : C9 1D D0 05 A9 20 48 D0  3B
CCB8 : 0F 58 18 60 68 18 69 40  73
CCC0 : C9 80 10 15 69 40 D0 11  6E
CCC8 : 68 48 29 E0 4A 4A 4A 4A  6F
CCD0 : 4A AA 68 29 1F 18 7D BF  D7
CCD8 : CD A6 C7 F0 02 09 80 AE  D0
CCE0 : 60 CE 9D 00 CE E8 8E 60  3E
CCE8 : CE E0 50 F0 0A A6 FE 18  5A
CCF0 : 58 60 A9 00 85 C7 68 AD  76
CCF8 : 00 DD 49 03 0A 0A 0A 0A  C7
CD00 : 0A 0A 85 F9 AD 18 D0 29  E1
CD08 : 0E 0A 0A 05 F9 85 F9 AD  4E
CD10 : 60 CE F0 4B A9 FC 85 02  19
CD18 : A5 F9 29 70 C9 10 D0 08  83
CD20 : A9 D0 05 F9 85 F9 C6 02  F9
```

```
CD28 : A9 1B 20 66 CC A9 2A 20  37
CD30 : 66 CC A9 04 20 66 CC AD  AB
CD38 : 60 CE 48 0A 0A 0A 20 66  91
CD40 : CC 68 4A 4A 4A 4A 4A 20  7D
CD48 : 66 CC A9 00 85 FA A6 FA  3F
CD50 : BD 00 CE 20 6E CD E6 FA  AC
CD58 : A5 FA CD 60 CE D0 EF A9  80
CD60 : 0D 20 66 CC 58 18 A6 FE  8F
CD68 : A9 00 8D 60 CE 60 48 4A  26
CD70 : 4A 4A 4A 4A 18 65 F9 85  5B

CD78 : FC 68 0A 0A 0A 85 FB A9  7C
CD80 : 80 85 D7 98 48 EA A5 01  40
CD88 : 48 25 02 85 01 A9 00 85  FD
CD90 : FD A0 07 B1 FB 25 D7 18  4E
CD98 : 69 FF A9 00 6A 48 98 AA  0C
CDA0 : 68 E8 CA F0 04 4A 4C A2  56
CDA8 : CD 05 FD 85 FD 88 10 E3  54
CDB0 : 68 85 01 A5 FD 20 66 CC  E4
CDB8 : 46 D7 D0 C9 68 A8 60 E0  66
CDC0 : 20 00 40 C0 60 40 60 FF  92
```

DOMLUVÍ SE C64 S PC?

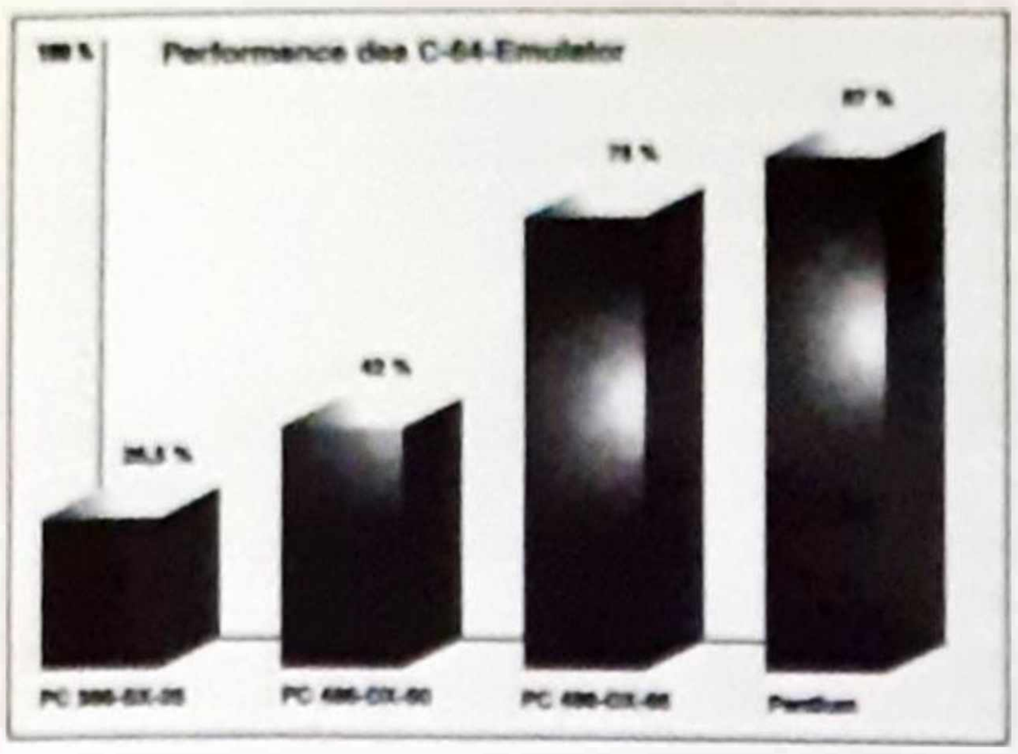
Mnoho let existují snahy emulovat C64 jinými počítači a tím zpřístupnit uživatelům starý software na jiných systémech. V redakci 64'er zkoušeli nový softwarový emulátor ve verzi Beta na počítači PC.
Aby software C64 mohl běhat na jiných počítačových systémech, musí dané počítače být schopny emulovat C64. Emulace se dá provést čistě hardwarově, tato varianta je však značně finančně náročná, neboť C64 se musí provést jako zásuvná deska a přizpůsobit danému počítači. Se softwarovým emulátorem to vypadá poněkud příznivěji.
V minulosti existovaly emulátory C64 na mnoha systémech. Co se kompatibility týče, zdaleka nedosáhly výkonu C64. Mnoho programů na nich neběželo. Světlou výjimku tvoří emulátor od firmy Micro Media.

Emulátor

Emulátor existuje momentálně ve verzi Beta. Je napsán MS-DOS-PC, jeho funkce předpokládá nasazení minimálně 386-SX s HD a VGA monitorem. Po instalaci a startu se dostanete do menu emulátoru, ze kterého jsou přístupné funkce. Je jich celá řada. Paleta zahrnuje možnosti od konfigurace disketové mechaniky nebo joystickového portu po časování timeru CIA. Lze rovněž volit mezi systémem a Kernalem. Využívání basicovských rozšíření se předpokládá také. Při startu programu se dá z HD nainstalovat jiný programovací jazyk. Všechna nastavení se dají uložit jako konfigurační file. Až na několik výjimek je obsazení klávesnice identické s C64. Přitom i zde je možno provést individuelní přizpůsobení. Po startu z menu se ohlásí C64 obvyklým způsobem a vše se může rozběhnout. Klávesou Esc se běh emulátoru přerušuje a program skáče zpět do hlavního menu. Po tomto kroku se dá například C64 resetovat, přičemž se rozlišuje mezi měkkým (všechna data v paměti zůstanou zachována) a tvrdým (paměť se vymaže) resetem. Program se pak vrátí k úvodnímu hlášení Commodore.

Kompatibilita

Aby bylo možno nechat běhat programy z C64 na emulátoru, musí se diskety překopírovat z formátu C64 na DOS. Pro tento účel jsou k dispozici různé tooly (Janus, BDOS nebo Transfile – uvedeny na stránkách 64'er Sonderheft). K emulátoru (v plné verzi) se dodává kabel, který přes paralelní port spojuje C64 a PC a zajišťuje tak přenos dat. Potřebné podpůrné programy obsahuje tool emulátoru už ve správném formátu. Tyto programy mají zvláštní označení a mohou být z emulátoru ládovány obvyklými rutinami. Předtím se ovšem musí v Setupu emulátoru nakonfigurovat flopyjednotka (objeví se na HD jako podadresář). Do ní se kopírují konvertované programy z C64. Pokud se potom naláduje directory, po zadání LIST se objeví jednotlivé názvy tak, jak je zvykem.
Schopnost programů bezproblémově běžet záleží především na na čistotě programování. Programy s autostartem emulátor bojkotuje. U programů v basicu nejsou žádné problémy. U programů ve strojovém kódu vypadá situace poněkud jinak. Hry a programy s kritickými časovacími rutinami vždy neběží. Dalším problémem jsou sbalené datové soubory. Mimo to se často hry dají na PC obtížně přenést pro rafinované ochrany proti kopírování.
Celkem tedy nepřekvapí konstatování, že GEOS nemůže právě pro ochrany proti kopírování a celkově komplikovaný vlastní DOS být konvertován do formátu MS-DOS.
Problémy nastražuje programu sem tam i používání systémových rutin. Test osahávání klávesnice psaný v assembleru SMONem s rutinou na adrese (JSR $ffe4) nedopadl dobře. Rovněž s řadou uživatelských programů byly problémy (sprite editor vývojového systému Katakis). Největší vadou na kráse ovšem bylo nenápadné uspání péčka po loadingu a savingu jednoho file testovaného SMON a Turbo Ass. Teprve totální reset postavil počítač zpět na nohy. Při ukládání pomocí Turba Ass se datový soubor



sice na HD zapsal dobře, ovšem potom následovalo zmíněné zřícení systému. U basicových programů s loadingem a savingem nebyly žádné problémy. Hraní s Copper-Bars a sprity je také možné. Musí se ovšem vzít do úvahy časovací poměry jednotlivých počítačů a jim musí být přizpůsobeny rutiny. PC joystick je programově ošetřen. Ve zvláštním menu se přizpůsobí hodnoty.

Pro přátele tiskáren nemáme dobré zprávy, neboť Beta-verze na ně nepamatuje.
Nakonec se chci zmínit ještě o jedné nevýhodě čistě technického charakteru. VGA monitory zobrazují grafiku mnohem ostřeji a detailněji, zatímco grafika emulátoru nepůsobí tak dobrým dojmem jako na monitoru pro C64.

Dobrý koncept

Beta-verze emulátoru C64 je výkonný software dobře emulující zmíněný počítač. Kompatibilita je však omezená a závisí na typu programu. Pokud se podaří odstranit poukázané nedostatky, mohl by emulátor být bezesporu zajímavý. Nezamlčujeme však, že provoz emulátoru žádá své i od PC, což se projevuje v jeho ceně.

Jaký počítač?

V podstatě běží C64 emulátor na PC 386 SX. Čím rychlejší PC, tím lépe. Nejlepších výsledků bylo dosaženo na PC 486-66, kdy se rychlost zpracování programů blížila originální rychlosti C64 (viz. obr).

COMOTRONIC NEWS

Jiffy DOS

Speeder pro všechny typy C64 a disketových mechanik. Zrychlovací faktor se blíží 14. User i expanzní port zůstává volný, není potřeba žádný paralelní kabel. Vlastní speeder sestává ze dvou pamětí eprom s redukcí patice a přepínačem. Tím se dá vždy přepnout na původní operační systém C64, takže nevzniknou s kompatibilitou žádné problémy.
Program obsazuje funkční klávesy příkazy pro listování v adresáři disket a pro natahování a spouštění programů. Při objednávání nutno uvést typ počítače a disketové jednotky.

Cena 25000,- Kč

DALŠÍ NABÍZENÉ VÝROBKY:

Programová disketa 5

-je zaměřena na programy využívající schopnosti C64, resp. obvodu SID generovat zvuky. Disketa obsahuje programy:

Digisound Ripper
Soundwriter V2.0
Funky-Drummer
Musicmaster
První program, Digisound-Ripper slouží k vyhledávání digitalních hudebních dat v ostatních programech pro C64. Soundwriter je pak špičkový hudební editor. pro komponování rozsáhlejších skladeb. Poslední dva programy simulují hudební nástroje, Funky-Drummer bicí a Musicmaster klávesové.

Cena 120,- Kč

Plotbasic II

Program firmy Billa je určen pro spolupráci se všemi dostupnými plotry, které lze k C64 připojit, ALFI, AMAGRAF 507 (Minigraf),XY 4150. Pogram dokáže při kreslení otáčet libovolné souřadné osy, pracuje i v polárních souřadnicích. Libovolná velikost a sklon písma jsou samozřejmostí. Program je vybaven různými fonty s českou diakritikou pro psaní textů. Psát je možno v režimu OPEN (obdoba psacího stroje, nebo v režimu TEXT (editor). Paměť C64 je využívá-

na jako buffer, do kterého se ukládá až 8000 kroků plotru.
Program je dodáván pro konkrétní typ plotru buď na kazetě nebo disketě.

Cena 280,- Kč

FONTEDITOR

Program využitelný ve spojení s Profibasicem III. Slouží k vytváření vlastních znakových sad nebo liter (viz FUN 8).

Cena 150,- Kč

HIRES MASTER

Grafické rutiny dělají v principu všechny totéž. Se sekundovou rychlostí kreslí kružnice, obdélníky či

desítky. Liší se však v provádění rychlosti, se kterou se obrazce objevují na obrazovce, asi jako Golf proti Maserati. Hires Master překonává v rychlosti všechny rekordy. Přes 40 nových příkazů převádí C64 na grafický stroj slušné úrovně. Na rozdíl od srovnatelných rozšíření Basicu byla rychlost grafických rutin optimalizována. Např. příkaz LINE se provádí rychlostí 13000 pixelů/sec! U jiných geometrických útvarů je rychlost o něco nižší. Mimo to zpracovává grafický systém 5 navzájem rozdílných nezávislých obrazovek.
Program je dodáván na disketě spolu s vysvětlením syntaxe a účelu jedn jednotlivých příkazů. Disketa je doplněna demem, které ozřejmuje způsob užívání příkazů.

Cena 110,- Kč

INZERCE

Prodám C64-II, VC 1541-II, dataset, tiskárnu, Final cartridge III, 2x joystick, diskety + box, kazety, množství literatury a programů. Možno i jednotlivě.

Roman JIRGALA
nám. Armády 4
669 02 Znojmo
tel. 0624 / 40 70

Prodám Commodore 64 II a disketovou jednotku 1541 II v bezvadném stavu. Ovládací systém GEOS 1 (disketa + příručka s českým překladem). Software Markt & Technik: SPECIAL BASIC a HYPRA BASIC, The Best of Grafik (2 diskety + 2 příručky). Myš, joystick. Cena dohodou.

Ing. Václav HEJL
Mohelnická 885
783 91 Uničov tel. 0643 505 67

Prodám program pro kopírování grafických obrazovek z C64 plotrem Minigraf 0507 (V disketové i kazetové verzi). Dále prodám upravený program DFM-českou databázi pro C64 a Minigraf (kazetová verze).

Ing. Bohumil MIKOTA
Polská 744
562 01 Ústí nad Orlicí

Predám program Anglicko-Slovenský slovník na C64. Obsahuje vyše 10 000 anglických slov a ich prekladov, dokáže POVEDAŤ ľubovoľné anglické slovo, možno v ňom listovať, zmestí sa na necelú disketu a stojí 70 Sk.

Stanislav KUCHAR
Vajanského 1129/29
02401 Kysucké Nove Mesto

COMOTRONIC

Firma COMOTRONIC s.r.o.
výroba a prodej

nabízí k okamžitému dodání

nejširší sortiment příslušenství k počítačovým systémům C64

Systém Commodore 64/128 (více než 250 položek)

HARDWARE

Počítače Commodore 64, magnetofony, disketové jednotky,
tiskárny, myši, joysticky, monitory
Turbo Cartridge - Final Cartridge - paměťové moduly
RAM/EPROM
Měřicí a řídicí moduly ❖ A/D - D/A převodníky
Programátory pamětí EPROM

FIRMWARE - BOOKWARE - SOFTWARE

Kazety a diskety ❖ Kabely všeho druhu
Příručky, manuály, návody
Časopis pro uživatele C64 - FUN with Commodore
Hry na kazetách a disketách ❖ PD programy na disketách
Programové kazety a diskety s návodem
Programovací jazyky ❖ Programy firmy SCANNTRONIK
Operační systém GEOS ❖ Originální hry

Zásilková služba dle katalogu a prodejna v sídle firmy

Sortiment vyráběný a dodávaný firmou COMOTRONIC s.r.o.
je také možno zakoupit v síti vybraných prodejen v celé
ČR a SR.

COMOTRONIC s.r.o, Dolnomlýnská 2, Šumperk 787 01

COMOTRONIC