MIKROBÁZE

04 _ nabídka programů 1987
_ emulátor PMD-85
_ kanály ZX-Spectra
_ modemy (1)

SPOLEČNÁ SLUŽBA AMATÉRSKÉHO RADIA A 602. ZO SVAZARMU PRO UŽIVATELE MIKROPOČÍTAČŮ

AR-602

OBSAH

Vydala 602. ZO Svazarmu pro potřeby vlastního aktivu. Zodpovědný redaktor ing. Jan Klabal, sestavení rukopisu Ladislav Zajíček. Adresa redakce: 602. ZO Svazarmu, Wintrova 8, 160 41 Praha 6, telefon 32 85 63. Neprodejné. Povoleno ONV Praha 6. Náklad 4000 výtisků. Praha, prosinec 1986

Program pro vnoučata

Když se má žena vrátit ze školení, na němž se seznámila s posledními novinkami výpočetní techniky (abych byl přesný - dozvěděla se, jak na štítek správně napsat sedmičku, aby ji čtečka nevyhodila jako jedničku), pravila: "Víš, ona ta barevná televize má něco do sebe; třeba ty Večerníčky vypadají kouzelně..." Souhlasil jsem. Jednak proto, že ženě zpravidla neodporuji, ale nakonec třeba i z toho důvodu, že ten náš šedivý fotbal se mi v barevném provedení zdá být o něco stravitelnější.

Souhlasil jsem, netuše, co mne čeká... Vždyť zhruba před rokem mne resortní pracovníci z (přižhavené) černobílé obrazovky mého letitého starouška v televizní besedě ujistili, že do roka a do dne (ne, Kozina se besedy neúčastnil) bude náš trh doslova zaplaven barevnými televizory. A občan, v roli spotřebitele, nebude vědět, pro který sáhnout dřív.

Nebudu popisovat své zklamání. Když jsem coby rodinný pověřenec obcházel síť prodejen za účelem nákupu barevného televizoru, při prvním pohledu na vystavený sortiment mi tam skoro bez nadsázky chyběl jen Bellův telegraf... Ale nechme toho; určitě by mi někdo povolaný zasvěceně vysvětlil, proč právě teď není mnou žádané. A určitě by se to vysvětlení jen hemžilo oblíbenými objektivními příčinami.

Nepochybuji ani o tom, že existuje analýza daného stavu. Jsme totiž národem analytiků. Analyzujeme kdeco. Třeba nedostatky, které vedly k debaklu fotbalistů ve Španělsku. Nevím, zda zásluhou té analýzy, ale do Mexika jsme se pak již ani nekvalifikovali. Analyzujeme příčiny naprostého výpadku hokejové reprezentace v Moskvě - a mne trochu mrazí v zádech při pomyšlení, zda za pár let nebudeme bojovat o holé udržení ve skupině B.

Přitom neustále ve všech pádech skloňujeme slova jako moderní metody řízení či výpočetní technika...ale stále nám někde něco pokulhává. Nejen proto, že mikroprocesor má pramalé pochopení pro objektivní potíže, ale kupříkladu i proto, že se ukázalo být jednodušším nedostatky zdůvodňovat a přivykat jim, než důsledně odstraňovat jejich příčiny.

Jsme národ analytiků. Vůbec nemám na mysli ty "pivní", kterým je všechno vždy jasné. Mluvím teď o celé, i když zdaleka ještě nezformované řadě odborně fundovaných i velmi talentovaných systémových analytiků a programátorů, schopných pracovat na nejvyšší srovnatelné úrovni. Rozvoj výpočetní techniky a elektroniky obecně se žádnému zdůvodňovateli příčin zastavit nepodaří. To, že tento rozmach postihl právě naši generaci, je pro nás neodvolatelným závazkem. Pokrok celé sféry i jejího efe-

ktivního nasazení bude vždy přímo záviset i na jejím programovém vybavení. Musí to samozřejmě být programové zázemí zahrnující naše specifické problémy, možnosti a podmínky, přístupy odpovídající našemu národnímu naturelu, způsobu myšlení a společenskému cítění. A není třeba nijak zvlášť diskutovat ani o tom, že ve své mateřšině umí jak technik, tak i běžný český či slovenský uživatel výpočetní techniky myslet nejoptimálněji.

Jistě, že lze se Sinclairem na stole prožít dlouhou noc do bílého dne při honbě potápěče za perlami, či ho nechat spolknout nějakou mořskou potvorou; řítit se po sjezdových svazích, či vést stříbřitý vůz tratí formule 1. Takový program může i pobavit, ale je nutno si uvědomit, že vznikl ve zcela odlišných podmínkách. Za čas omrzí. Pokud ovšem nejsem opravdu jen pouhým pasivním uživatelem probděných nocí a dvou otlačených prstů... Nikdy mi nešlo do hlavy spojení počítač-pasivní uživatel. Počítač svou podstatou musí burcovat touhu po poznání, zúročeném tvůrčí prací. Jde jen o to, otevřít poznání cestu. Proto nelze než vřele uvítat rozhodnutí Mikrobáze ustoupit od původně zamýšleného, vší původnosti zbaveného šíření zahraničních programů. Jedná se o nesporný kvalitativní krok kupředu, logicky konverzující s naléhavými potřebami naší výpočetní techniky i s vlastním posláním Mikrobáze. Odmítnutím šíření nepůvodních programů Mikrobáze navíc zasadí citelnou ránu infekčnímu snobismu pseudouživatelů mikropočítačů, z nichž mnozí se nesmyslně a stále hlouběji propadají do imaginárního světla herního opojení s často pochybeným obsahem i účelem. I když nic proti skutečně dobrému hernímu softwaru pro skutečně adekvátní odpočinek v přijatelném časovém rozsahu.

Má-li se výpočetní technika stát prvkem výrazného uvolnění lidského potenciálu tvůrčí a řídící práce, je nutno vychovat dostatek kvalitních a samostatných programátorů schopných pracovat koncepčně, efektivně. Jedna ze souběžných cest k tomuto cíli vede právě přes malé osobní počítače se širokou amatérskou základnou. Už i proto, že amatéři dávají do své činnosti devizu nejvyšší - opravdový zájem.

7. zasedání ÚV Svazarmu konstatovalo stále rostoucí zájem o elektroniku, i fakt, že úspěšná činnost řady ZO Svazarmu přitahuje stále nové zájemce o tvůrčí práci v tomto novodobém oboru. I z tohoto hlediska je kladem zvyšování náročnosti, kotvící v podpoře tvorby původního tuzemského softwaru. Je to zcela logické pokračování nastoupené cesty.

Opravdu není důvodu závistivě pokukovat po zahraniční produkci, víme-li, že je v našich silách vytvářet svou vlastní, která bude moci té zahraniční úspěšně konkurovat. A to bez jakéhokoli vynaložení investic! Ale i bez vytváření objektivních příčin. Chce-li někdo za investici považovat osobní počítač, prosím. Ale pak si musí uvědomit, že taková investice se nám všem při vytvoření nezbytných legislativních podmínek i vazeb poptávky a odbytu (tedy ryze administrativními nástroji) nejméně statisícinásobně vrátí.

Je na čase, aby v pohostinném srdci Evropy přestaly být hostem všechny objektivní potíže. Aby se naše vnoučata mohla těšit na barevný Večerníček, a až povyrostou, mohla pokračovat v tradicích, jejichž spoluzakladatelem v oboru výpočetní techniky se Mikrobáze svým novým programem stává. Programem, který se musí stát věcí nás všech.

-kš-

Herbář nápadů a zkušeností

Softwarové drobnosti pro ZX Spectrum

Chceme-li zastavit program v INPUTu (někdy ani jinou možnost nemáme), zadáme:
-STOP, ENTER v případě vstupu číselného parametru (INPUT a$)
-EDIT, STOP, ENTER při vstupu řetězce (INPUT a$)
-"kurzor dolů" - jako ad 2, ale bez uvozovek (INPUT LINE a$)
Pokud chcete vymazat vše, co jste do INPUTu právě napsali, místo DELETE můžete použít EDIT, který provede výmaz najednou.

V INPUTu můžeme zobrazovat i proměnné. Dáme je buď do závorky, nebo použijeme funkcí INT, STR$, VAL STR$, ABS... Můžeme použít i cyklu, např.:

```
10  DIM a(5)
20  FOR i=1 TO 5
30  INPUT "Položka"; INT i; "=";  a (i)
40  NEXT i
```

Zapisujeme-li více po sobě následujících bloků dat funkcí SAVE, můžeme odstranit nevítané hlášení "Start tape and then press any key" změnou obsahu adresy 23736 na 181. Uvedený POKE však radši neumísťujte před prvním blokem, počítač by začal s přenosem dat ihned, aniž byste třeba stačili spustit magnetofon. Příklad:

```
9000 SAVE "jméno" LINE z: POKE 23736,181: SAVE "obrázek" SCREEN$: POKE 23736,181:
SAVE "mc"  CODE xxxxx,yyy
```

Při záznamu pouhých dvou bloků nemusíme měnit obsah adresy 23736, ale po první projekci hlášení "Start tape..." TRUE nebo INVERSE VIDEO. Nápis se sice po záznamu prvního bloku objeví, ale přenos dat se nezastaví.

K některým účelům využití logických a aritmetických funkcí při tvorbě basicových programů. Tak místo:

```
50  IF x=5 THEN LET y = Ø
60  IF x=3 THEN LET y=12
```

lze psát:

```
50  LET y = (Ø AND x = 5)+(12 AND x = 3)
```

Kromě úspory paměti vede takový postup i ke zrychlení programu, hlavně v případě řízení běhu programu ovládacími tlačítky. Místo komplikovaného:

```
10  IF INKEY$="8" THEN LET p=p+1
20  IF INKEY$="5" THEN LET p=p-1
30  IF p>31  THEN LET p=31
40  IF p<0   THEN LET p=0
```

je rozhodně lepší:

```
10 LET p=p + (INKEY$="8") - (INKEY$="5") + (p<0) - (p>31),
```

Do našich spektrovských programů můžeme zavést funkci DRAW TO, běžnou u jiných počítačů. Využijeme přitom systémové proměnné COORDS:

```
10  PLOT 20, 30
20  LET x=100: LET y=80: GO SUB 9000
30  DRAW x,y
40  STOP
9000  LET  x=x-PEEK 23677: LET y=y-PEEK 23678: RETURN
```

nebo:

```
10  DEF FN x(h)=h-PEEK 23677
20  DEF FN y(v)=v-PEEK 23678
30  PLOT 20,30
40  DRAW FN x(100), FN y(80)
```

Hezkou hříčkou s užitím příkazu DRAW je tento program:

```
10  INPUT "i=";i:  CLS: PLOT 90,60: DRAW 80,50,i: GO TO 10
```

Za i zkuste dosazovat: 0, 1, 2, 3, 3.14, PI, 4, 197, 198, 199, 400, 450, 499, 600, 693, 777, 1981, 1994, 2109, 2110, 9000, 15000, 16000, 17000, 23000, 567890, 900000, 9876543, 9876543210, 777777777, vyzkoušet si můžete i hodnoty jiné.

Transformací PLOT na PRINT AT a naopak lze bez problémů směšovat obrazový text s grafikou:

```
DEF FN  x(s)=8 *s
DEF FN  y(r)=(21-r) *8
DEF FN  s(x)=INT  (x/8)
DEF FN  r(y)=21-INT  (y/8)
```

s - sloupec
r - řádek
x - souřadnice x bodu
y - souřadnice y bodu

Příklad užití:

```
10  DEF FN x(s)=8*s: DEF FN y(r)=(21-r)*8
20  FOR s=2 TO 30 STEP 2
30  LET r=s/2+3
40  PRINT AT r,s; FLASH 1;"+"
50  PLOT FN  x(s)-1, FN y(r): DRAW 0,9:  DRAW 0,-9: DRAW -9,0: NEXT s
```

Pro dosažení změny atributu nelze bohužel zadat LET ATTR (2,2)=56. Téhož efektu však dosáhneme pomocí definice funkce:

```
10  DEF FN a(y,x)=22528+32*y+x
20  FOR f=1 TO 704:  PRINT "$": NEXT f
30  INPUT "atribut ";hodnota;" řádek ";r;" sloupec ";s
40  POKE FN a(r,s) hodnota
50  GO TO 30
```

U změn permanentních atributů pro horní část obrazovky můžeme postupovat dvěma způsoby. Buď:

```
POKE 23693,78: CLS                          (78=BIN 01001110),   nebo:
10  PAPER 1:  INK 6:  FLASH 0: BRIGHT 1: CLS (obojí je shodné)
```

U permanentních atributů dolní části obrazovky: POKE 23624,n. Barva BORDERu je brána jako PAPER nezávisle na jeho hodnotách BRIGHT a FLASH. Zadáme-li: INPUT INK 2; PAPER 6; "text"; a$, zbarví se jen text v uvozovkách. Vkládaný řetězec už bude černý na bílém pozadí. Aby se zbarvil i on, zadáme napřed POKE 23624,50 (což je binárně 00110010) a pak INPUT. V okamžiku kdy začneme vkládat řetězec, zbarví se BORDER jako PAPER. Proto se musí BORDER zadat předem:

```
10  BORDER 0
20  POKE  23624,132
30  INPUT "text"
```

Při zápisu programu lze s atributy pracovat takto (kdekoli na programové řádce, ale až za jejím číslem):

```
V E módu: 0-7                          PAPER (CHR$ 17  a CHR$  0-7)
          0-7 (přes CAPS SHIFT)        INK   (CHR$ 16  a CHR$  0-7)
          8                            BRIGHT 0  (CHR$ 19 a CHR$ 0)
          9                            BRIGHT 1  (CHR$ 19 a CHR$ 1)
          8   (přes CAPS SHIFT)        FLASH  0  (CHR$ 18 a CHR$ 0)
          9   (přes CAPS SHIFT)        FLASH  1  (CHR$ 18 a CHR$ 1)
V L módu: 3   (přes CAPS SHIFT)        INVERSE 0 (CHR$ 20 a CHR$ 0)
          4   (přes CAPS SHIFT)        INVERSE 1 (CHR$ 20 a CHR$ 1)
```

Protože při tvorbě atributů vkládáte 2 znaky, pro výmaz každého z nich musíte i DELETE stisknout dvakrát. Poprvé se objeví otazník, který po druhém stisku zmizí. Z toho důvodu se i kurzor při projíždění obarvených řádek programu v místech uložení atributů trochu zdrží.

Přehled dalších řídících znaků:

```
CHR$ 6 - posouvá tisk na pozici TAB 16, př.; PRINT 1; CHR$ 6;2
CHR$ 8 - posouvá tisk o 1 znak zpět (angl. backspace)
CHR$ 21- funkce OVER, př.: PRINT "O"; CHR$ 21;  CHR$ 1;  CHR$ 8; "#"
CHR$ 13- posun tisku na nový řádek (angl. carriage return, CR)
Např. CHR$ 22 + CHR$ 1+CHR$ 4 je totéž, co AT 1,4
     CHR$ 23+  CHR$ a+CHR$ 6 je totéž co  TAB a+256*b
```

Vkládáním číselných parametrů na adresy 23617 a 23658 můžeme měnit módy L, C, E a G.

```
POKE 23658,8 - přepnutí do módu C  (s okamžitým účinkem jen
POKE 23658,0 - přepnutí do módu L   když na adr. 23617 je 0)

POKE 23617,2 - přepnutí do módu G
POKE 23617,1 - přepnutí do módu E
POKE 23617,0 - přepnutí do módu C a L  (podle obsahu adr. 23658)
```

Změny módu malých a velkých písmen jsou užitečné, zvláště když jimi volíme některou z nabízených programových eventualit. Nemusíme tak v programovém testu stisknutého tlačítka brát v úvahu oba módy, ale jen jediný, předem nastavený.

Můžete si rovněž vyhrát při vkládání jiných než uvedených hodnot na adresu 23617 a sledovat, jak vypadá kurzor před zadáním vstupního řetězce a po něm (při užití INPUTu). V programu to však raději nezkoušejte. Pokud se v INPUTu "zaseknete", stiskněte dvakrát GRAPHICS nebo EXTENDED.

ing. Petr Veselý

Funkce "pro všechno"

Všem majitelům manuálu ZX Spectra by měla být dobře známa možnost definovat své vlastní "uživatelské" funkce příkazem

DEF FN identifikátor (formální parametr) = výraz
a jejich volání příkazem
FN identifikátor (skutečné parametry)

Podstatnou výhodou těchto funkcí oproti basicovým podprogramům je:
- snadný přenos parametrů
- možnost jejich užití jako standardních funkcí přímo ve složitějších výrazech
- zvýšení přehlednosti programu

Ne všichni programátoři však využívají možností, které tyto funkce nabízejí. Zřejmě i proto, že prostě nevědí, co s nimi. Alespoň částečné zaplnění této mezery je jedním z cílů dále uvedených příkladů.

Důležitou dispozicí ZX Spectra, kterou budu v následujícím hojně používat, je standardní funkce VAL s řetězcovým parametrem. VAL interpretuje tento řetězec jako zcela obecný výraz a určuje jeho hodnotu, která se stává hodnotou funkce VAL.

```
Např.:      10 LET  a=3
            20 PRINT VAL  "2*a"          (vytiskne se číslo 6)
```

Rekurze

Jakkoli se vám to může zdát divné, v Basicu ZX Spectra můžeme napsat (i když dost komplikovaně) rekurzívní funkci. Většina z vás jistě zná rekurzívní funkci faktoriálu:

1. n! = n * (n-1) !
2. 0! = 1

Určitě vás napadne, že první řádek bychom mohli přepsat takto:

```
DEF FN  f (n) = n*FN f (n-1)
```

Ale přijdete i bez počítače na to, co pak udělá PRINT FN f (16)?
Takže tudy cesta nevede. Ale můžeme na to jít jinudy:

```
 5 REM - Příklad 1 - Rekurzívní faktoriál
10 DEF FN  f (n)=VAL (("1" AND n=0)+("n*FN f (n-1)" AND n>0))
20 PRINT FN  f (0)'FN f (1)'FN f (7)'FN f (33)
```

Projděte si uvedený příklad bedlivě - pak už vám nebude činit potíže pochopit následující:

```
 5 REM - Příklad 2 - Binomický koeficient rekurzívně
10 DEF FN  b (n,k) = VAL (("n" AND  k=1)+
                          ("n/k*FN b(n-1, k-1)"  AND k>1))
```

Další dva příklady jsou už o něco složitější. Zkuste si také zdůvodnit, proč bylo ve 3. příkladu nutné použít funkci STR$.

Fibonacciho posloupnost je definována takto:

a(0)=0
a(1)=1
a(n+2)=a(n)+a(n+1) pro n=0, 1, 2, 3,...

```
 5 REM - Příklad 3 - Fibonacciho posloupnost
10 DEF FN a(n)=VAL (("0" AND n=0)+("1" AND n=1)+
   (("FN a ("+STR$ (n-2)+")+(FN a("+STR$(n-1)+")") AND n>1))
20 PRINT FN a (0) FN a (1) FN a (2) FN a (7)
```

Z tohoto příkladu je vidět, jak pomalá je tzv. "exponenciální rekurze".

Euklidův algoritmus pro určení největšího společného dělitele (NSD) dvou přirozených čísel je založen na rekurzívním kroku:

Z (n+2) = Z(n) MOD Z (n+1)

Funkci pro výpočet NSD můžeme napsat třeba takto:

```
 5 REM - Příklad 4 - Výpočet NSD
10 DEF FN u(a,b)=VAL (("a" AND b=0)+(("FN u(b,"+
                  STR$(a-b*INT(a/(b+(b=0))))+")") AND b<>0))
20 DEF FN e(p,q)=VAL(("FN u(p,q)" AND p > q)+
                  ("FN u(q,p)" AND p <= q))
30 PRINT FN e (30, 1000), FN e (111,11)
```

Chcete-li se přesvědčit o tom, že jste problematiku zvládli, zkuste napsat funkci, která přirozenému číslu přiřadí:

hodnotu 1, jde - li o prvočíslo,
hodnotu 0 v ostatních případech

Závěrem lze konstatovat, že rekurzívní uživatelské funkce jsou pomalé a kladou značné nároky na paměť (viz Fabonacci). Na druhé straně však korespondují s pohodlím a úhledností zápisu algoritmů ve speciálních (zejména matematických programech.

Jiné šikovné funkce

ZX Spectrum nedisponuje funkcí MAX (x,y), jejíž hodnotou by bylo větší z čísel x,y. Můžeme ji ale nadefinovat:

```
 5 REM - Příklad 5 - Maximum, minimum
10 DEF FN m(x,y)=x+(y-x)*(y> x): REM maximum
20 DEF FN n(x,y)=x+(y-x)*(y< x): REM minimum
30 DEF FN t(x,y,z)=FN m(x,FNm(y,z)):  REM pro tři parametry atd., atd...
```

Zobecněním 5. příkladu dojdeme k této úloze: Naprogramujte uživatelskou funkci podle předpisu:

```
   sin (x-1)    pro x < 1
f(x)=  nebo
   ln(x)        pro x > =1
```

Protože pro x < =0 není funkce ln(x) definována, musíme si pomoci funkcí VAL:

```
 5 REM - Příklad 6  - Speciální funkce
10 DEF FN f(x)=VAL (("SIN(x-1)" AND x<1)+("LN x" AND x> =1))
```

A ještě jeden triviální, ale dle mých zkušeností málo známý způsob využití u řetězcových funkcí:

```
 5 REM -  Příklad 7 - Hodnocení
10 DEF FN h$=("slabý" AND v=0)+("průměrný" And v=1)+
            ("dobrý" AND v=2)+("výborný " AND v=3)
```

Naznačeným způsobem lze v Basicu naprogramovat jako rekurzívní funkci skutečně ledacos. Smysl takového počínání je však jen v jednodušších případech, jakým je třeba právě výpočet faktoriátu. Byl bych rád, kdyby můj příspěvek přiměl někoho z vás k rozpravě o tom, jak uvedené provést v elegantnějším Pascalu.

Ivan Libicher

MODEMY

1.část

V souvislosti s rozvojem elektroniky se stále častěji objevují termíny jako elektronická pošta, spolupracující databázové systémy apod. Člověk méně elektronizovaný si pod takovými názvy může leccos představit, ale podstata zůstává záhadná. Nejedná se však o nic jiného, než o vzájemnou komunikaci mezi počítači. V sérii dvou článků na ni zaměříme svou pozornost. V prvním z nich ji budeme věnovat základnímu komunikačnímu zařízení zvanému

- M O D E M -

KOMUNIKACE TELEFONEM

Obsah základní myšlenky počítačové komunikace je nadmíru jednoduchý - předat soubor bitů z vysílajícího počítače do přijímajícího tak, aby tyto bity byly přijaty naprosto věrně, tedy aby příjemce obdržel věrnou kopii vyslaného informačního vzorku.

Skoro každá jednoduchá myšlenka z oblasti výpočetní techniky při svém převodu do praxe naráží na řadu problémů. Jedním z nejzářnějších potvrzení tohoto pravidla je právě dálkový přenos dat. Byly zkoušeny speciální přenosové sítě, které však především po stránce požadované kvantity přenosu nevyhovovaly. Proto se průkopníci přenosu informací začali zajímat o využití stávajících sítí, které má každý "po ruce" - tou je především síť telefonní. Díky tomu dnes informační exploze neslevila nic ze svého lvího tempa.

Problém využití telefonních sítí spočívá především v omezené šířce frekvenčního pásma, jakou jsou schopny přenášet. Bývá to maximálně kolem 3 kHz. Srozumitelnosti řeči při jejím přenosu to nevadí. Poslouchat hudbu telefonem by ani průměrný posluchač moc dlouho nevydržel. Počítač se tomuto handikapu musí podřídit, a tak jsou přenosové rychlosti dat po telefonní lince skutečně šnečí.

Další problém - k dispozici máme jen jeden vodič. Na paralelní přenos tak musíme zapomenout a přenést jej na sériový. A tak jsme z běžně možných frekvencí paralelního přenosu (kolem 1 MHz) rázem dole na "sériové" frekvenci 3 kHz! Situaci komplikuje i to, že ústředny používají některé kmitočty pro své účely (tónové signalizace). A samozřejmě nelze zapomenout na někdy značné rušení i výpadky v síti.

Abychom se z toho všeho nějak dostali, je nutno:

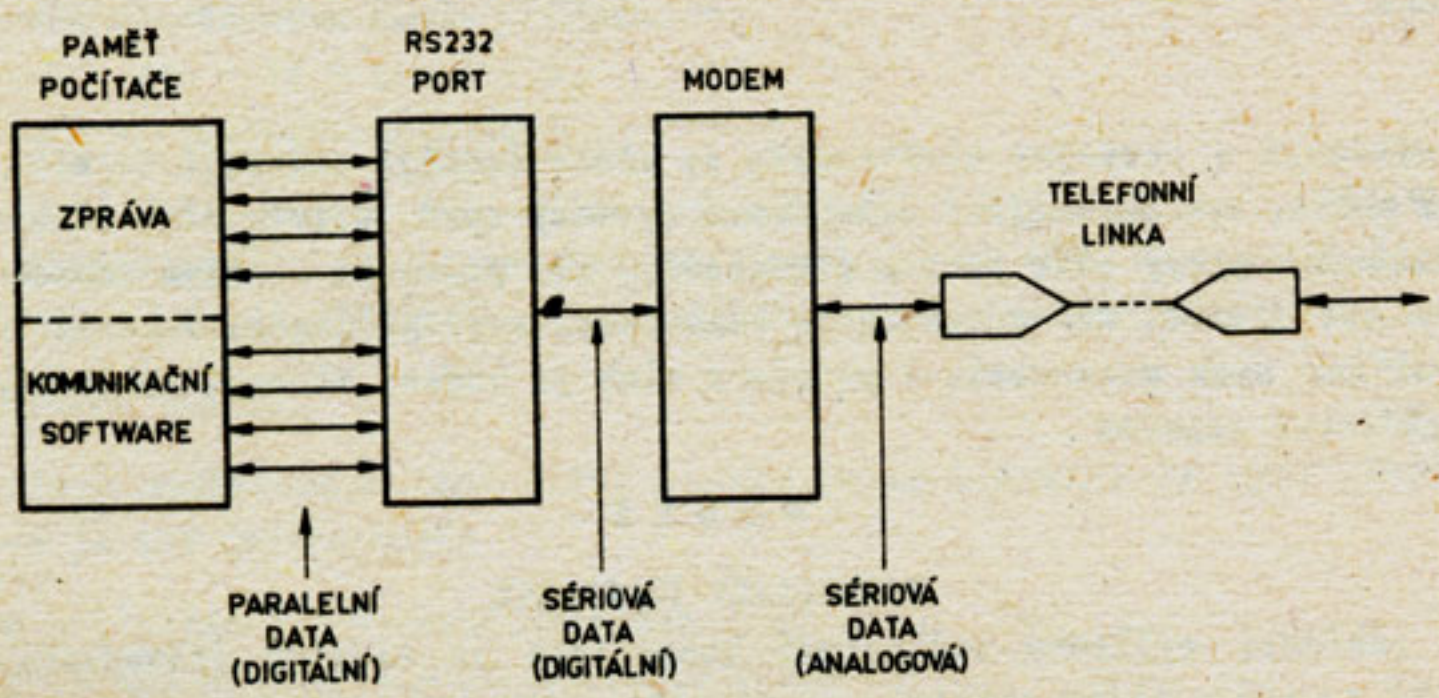
1) převést data z vysílacího počítače do vhodné formy pro pomalý sériový přenos,
2) digitální data musejí vyhovovat podmínkám přenosu po telefonní lince,
3) u přijímacího počítače musíme zajistit opačný proces uvedený pod bodem 1).

Teprve po zajištění těchto požadavků přijde čas na to, proč jsme to všechno dělali - na vlastní přenášený software. Nejdříve se tedy budeme věnovat způsobu převodu paralelních dat na sériové.

SÉRIOVÝ INTERFACE

Nejznámnějšími sériovými interfacy pro mikropočítače jsou: RS232C a RS423. Jsou mezi nimi určité rozdílnosti, funkčně jsou však shodné. Pro naše účely budeme dále zvažovat port RS232. Co vlastně tento port pro nás může vykonat?

Stručná odpověď zní - přesně to, co hledáme. Odebere paralelní data ze sběrnice počítače a převede je na sériový proud bitů (po sobě jdoucích). Zařízením, která



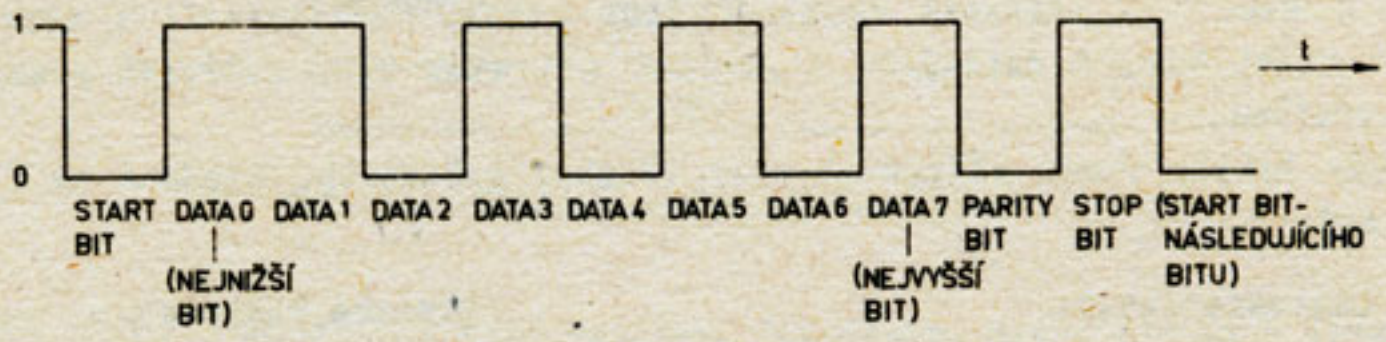
Obr. 1. Modemová komunikace

tuto konverzi provádějí, se obecně říká ACIA (Asynchronous Communications Interface Adaptor). Jsou napojena přímo na počítačovou sběrnici. Procesu převodu je třeba porozumět, abychom pochopili vše ostatní a uměli si v případě potřeby úspěšně poradit.

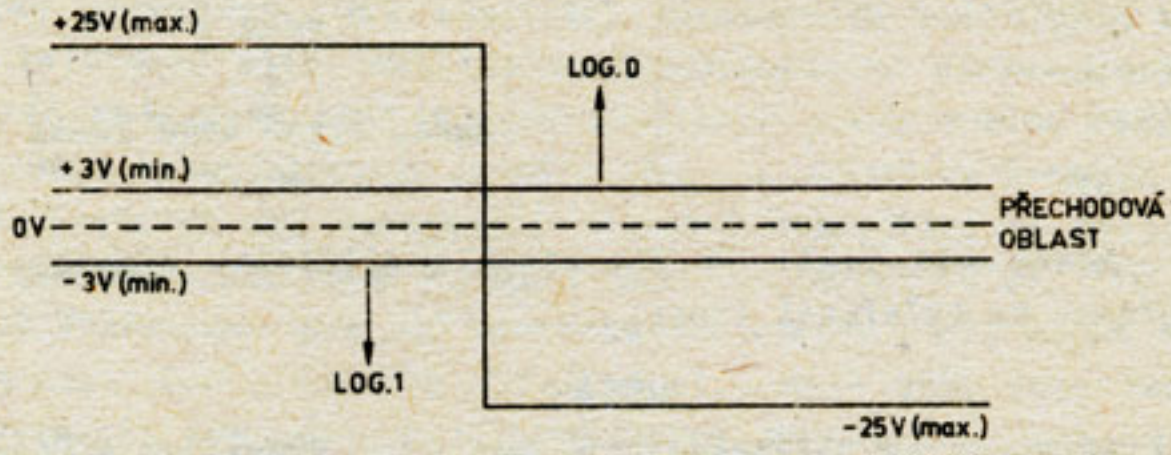
Standardní RS232 má konektor typu D s 25ti vývody, ačkoli je z nich využito jen několik (viz tab. 1). Bohužel různé firmy používají ve svých sériových portech zcela odlišné konektory, někdy i jen pocínované vývody desky plošných spojů. Důvod je zřejmý - šetření materiálem. Proto si vždy pečlivě zkontrolujte, zda máte vodiče konektoru správně propojeny.

č. vývodu RS232	Signální linka
1	Zem
2	Vysílaná data (TXD)
3	Přijímaná data (RXD)
4	Požadavek na vyslání (RTS)
5	Konec vysílání (CTS)

Sériový interface v podstatě pracuje s linkami "data in" a "data out" a zemnícím vodičem. To je minimum nezbytné pro obousměrný sériový přenos. Někdy jsou přidány ještě tzv. "handshaking lines" (česky přibližně linky pro vzájemné potřásání rukou), ač nejsou nezbytně nutné. Pokud jsou použity, data mohou být vyslána z počítače pouze tehdy, je-li CTS v urovni 0; a přijímána, když RTS je na úrovni 0. Použijeme-li místo tohoto hardwarového třesení rukou softwarové, propojíme vzájemně CTS s RTS a hned tu máme pouze tříllinkové vedení.



Obr. 2. Průběh signálu sériového přenosu jednoho bajtu



Obr. 3. Napěťové úrovně portu RS 232

Samozřejmě, že formát dat strany vysílající i přijímající musí být shodný. Obecně užívaný formát je na obr. 2. Obr. 3 zobrazuje jeho napěťové úrovně. V praxi se

pro obě polarity používá obvykle +12 V (log. 0) a -12 V (log. 1). Jak je z obrázku 2 patrno, je každá přenášená skupina dat umístěna mezi start bit, po němž následuje bit 0 dat, a dva závěrečné bity (bit parity a stop bit), které jsou vyslány po bitu 7 skupiny dat. Bit parity slouží ke kontrole správnosti obsahu přenesené skupiny dat. Některá zařízení jej nepoužívají.

Skupiny datových bitů sestávají ze 7 nebo 8 bitů. Oněch 7 bitů se používá především pro přenos textů, u jehož znaků není nejvyšší bit důležitý (může být trvale na úrovni log. 0). Typické bitové kombinace ukazuje tabulka 2:

Start bit	datové bity	parita	stop bity
1	7	sudá	2
1	7	lichá	2
1	7	sudá	1
1	7	licbá	1
1	8	žádná	2
1	8	žádná	1
1	8	sudá	1
1	8	lichá	1

Formáty jsou obvykle určeny použitým softwarem, který může obsahovat i více typů formátu pro umožnění potřebné volby. Podstatné je vždy to, že obě strany přenosové linky musejí být "naladěny" na tentýž formát, ale - ve vztahu k času - i shodnou rychlost přenosu dat (angl. data rate, resp. baud rate), která se udává v baudech. Např. přenese-li zařízení 1 digitální kmit (přechod ze stavu log. 0 na log. 1 a zpět) za 1 msec. je přenosná rychlost 1000 baudů. Použijete-li v této rychlosti formátu 1 start bit, 8 datových, 1 paritní a 1 stop bit, zařízení přenese cca 91 bajtů za vteřinu.

Na závěr tohoto úvodu můžeme shrnout všechny požadavky, kladené na formát přenášených dat:

Charakteristika	Typická hodnota
Rychlost přenosu	75, 300, 600, 1200 baudů
Datové bity	7 nebo 8
Start bit	1
Parita	sudá, lichá nebo žádná
Stop bity	1 nebo 2

Dalším krokem, který nás čeká, je převedení digitálního signálu na analogový pro jeho zdárný přenos po telefonní lince.

MODEMY

Slovo MODEM je odvozeno od počátečních písmen slov MOdulátor a DEModulátor. Tím zároveň jasně naznačuje vlastní funkci zařízení. Digitální signál je modulován na nosný kmitočet. Na straně přijímající pak probíhá jeho demodulace. Nosná vlna je pochopitelně v rozsahu slyšitelného kmitočtu, který jsou telefonní linky schopny přenést. Modulace je řízena digitálním signálem. Použitou technikou je klíčování-

- binární 0 vyvolá modulaci jedné frekvence, 1 pak frekvence odlišné.

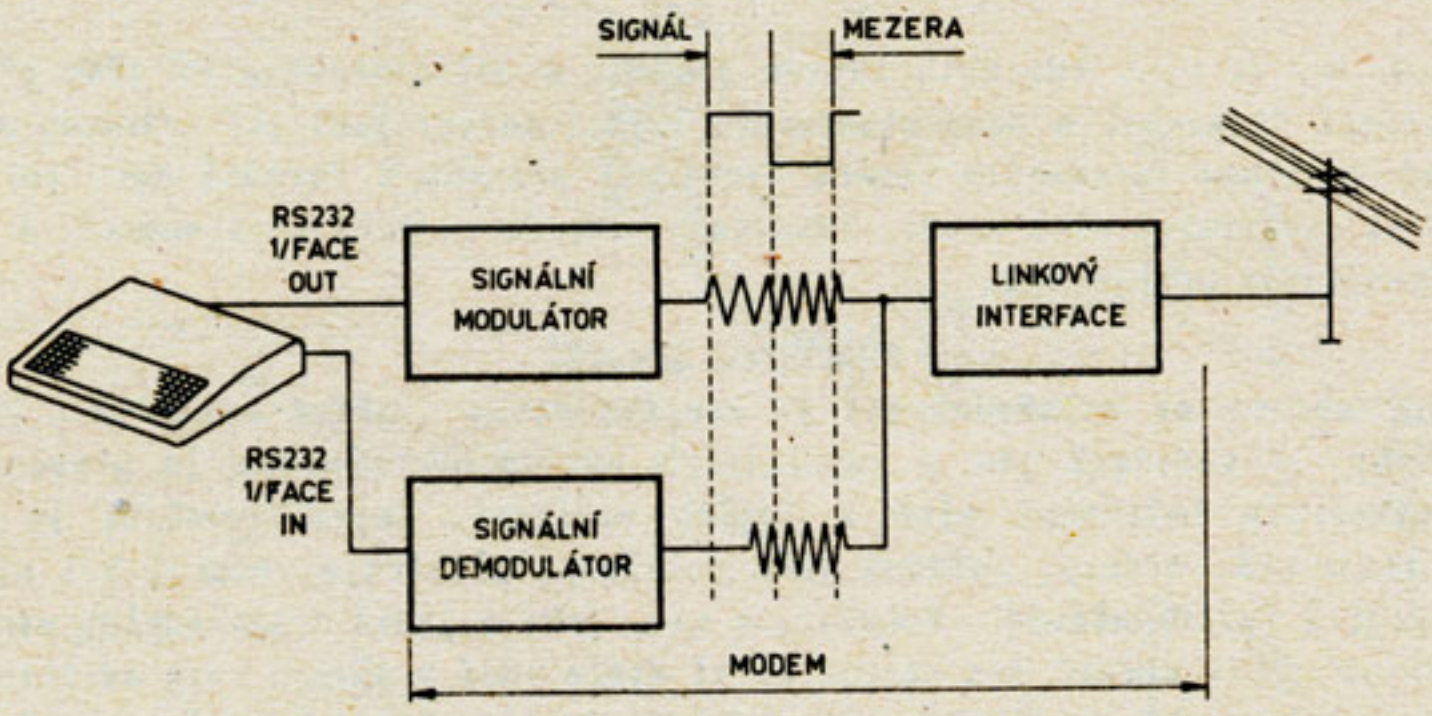
Pásmový rozsah telefonní linky neomezuje jen výšku modulovaných signálů, ale i rychlost jejich střídání. Lze obecně říci, že maximum, které můžeme "protáhnout" telefonní linkou, leží kolem hodnoty 1300 baudů.

Z hlediska směru přenosu rozlišujeme linky na:

1) simplex (přenos jen jedním směrem)
2) half-duplex (střídavě v obou směrech)
3) full-duplex (současně oběma směry)

Linky typu half-duplex mohou přenášet signál v nejvyšších rychlostech v každém směru. Proto se jim dává přednost - šetří poplatky za užívání linek. V běžné praxi se jen málokdy vyskytuje opodstatněná potřeba současného přenosu v obou směrech. Podobně je tomu i u linek simplex.

Základní aranžmá napojení modemu na telefonní linku najdete na obr. 4. Pokud



Obr. 4. Základní modemová konfigurace

by oba konce linky používaly stejné vysílací frekvence, docházelo by k interferenci (rušení). Proto je každému z obou komunikujících modemů přidělen jiný pár modulační frekvence. Každý modem je dále vybaven pásmovou propustí, která zároveň zabraňuje zpětnovazebnímu pronikání jeho vlastních modulačních kmitočtů i signálů a velké části poruch telefonní sítě. Mezi oběma komunikujícími stranami proto musí být dohoda o tom, kdo bude hrát roli v módu ORIGINATE a kdo v módu ANSWER. Obvykle ten, kdo volá, přechází automaticky do prvního módu, a volaný do druhého.

Po prvních zmatcích při nasazování modemové komunikace se přistoupilo ke standardizaci. V USA nese standard označení Bell, v Evropě CCITT. Tím jsou automaticky dány modulační frekvence pro obě komunikující strany i při volbě různých kombinací rychlosti přenosu. Např. 300/300 (CCITT V.21 a Bell 103) jako jedna z nejčastěji užívaných kombinací. Pro spojení s databankami se používá např. kombinace 1200/75. Je to proto, že přenos z databanky by měl a může proběhnout rychle, zatímco odpověď volajícího je obvykle dost pomalá a jednoduchá (jedná se převážně o tlačítkovou volbu z řady menu). Další tabulka ukazuje některé užívané modulační frekvence (v Hz) modemového styku:

Mód	Baudy	Vysíl. frekv.		Příj. frekv.		Duplex
		log.0	log.1	log.0	log.1	
CCITT V.21 Orig	300	1180	980	1850	1650	full
CCITT V.21 Answer	300	1850	1650	1180	980	full
CCITT V.23 Mode 1	600	1700	1300	1700	1300	half
CCITT V.23 Mode 2	1200	2100	1300	2100	1300	half
CCITT V.23 Back	75	450	390	450	390	--
Bell 103 Orig	300	1070	1270	2025	2225	full
Bell 103 Answer	300	2025	2225	1070	1270	full
Bell 202	1200	2200	1200	2200	1200	half

Zajímavé je, že ten, kdo volá (vyjma spojení s databankami), má vždy přiřazenu nižší vysílací frekvenci u obou standardů. Dnešní modemy jsou již vybaveny možností volit mezi většinou zavedených poměrů rychlosti přenosu i formátů dat, takže není problémem se spojovat i přes oceán. Takovýmto modemům se říká multimódové. A stojíme před předposledním problémem:

REALIZACE SPOJENÍ

Napojení modemu na telefonní síť je dvojího druhu - přímé a nepřímé. Přímé se užívá zřídka (v podstatě jen u speciálních aplikací). Důvodem je prevence před možným poškozením telefonní sítě a jejich zařízení. Nejrozšířenější je spojení nepřímé, akustické. Modemy, podobně jako telefonní přístroje, musejí být schváleny správou spojů k jejich užívání. Bohužel, u nás, jako v jedné z posledních evropských zemí, možnost užití modemů pro osobní užití stále není z důvodů ryze administrativních dovolena. To s sebou přináší obrovské časové, materiální a kvalitativně-kvantitativní informační ztráty se všemi jejich důsledky.

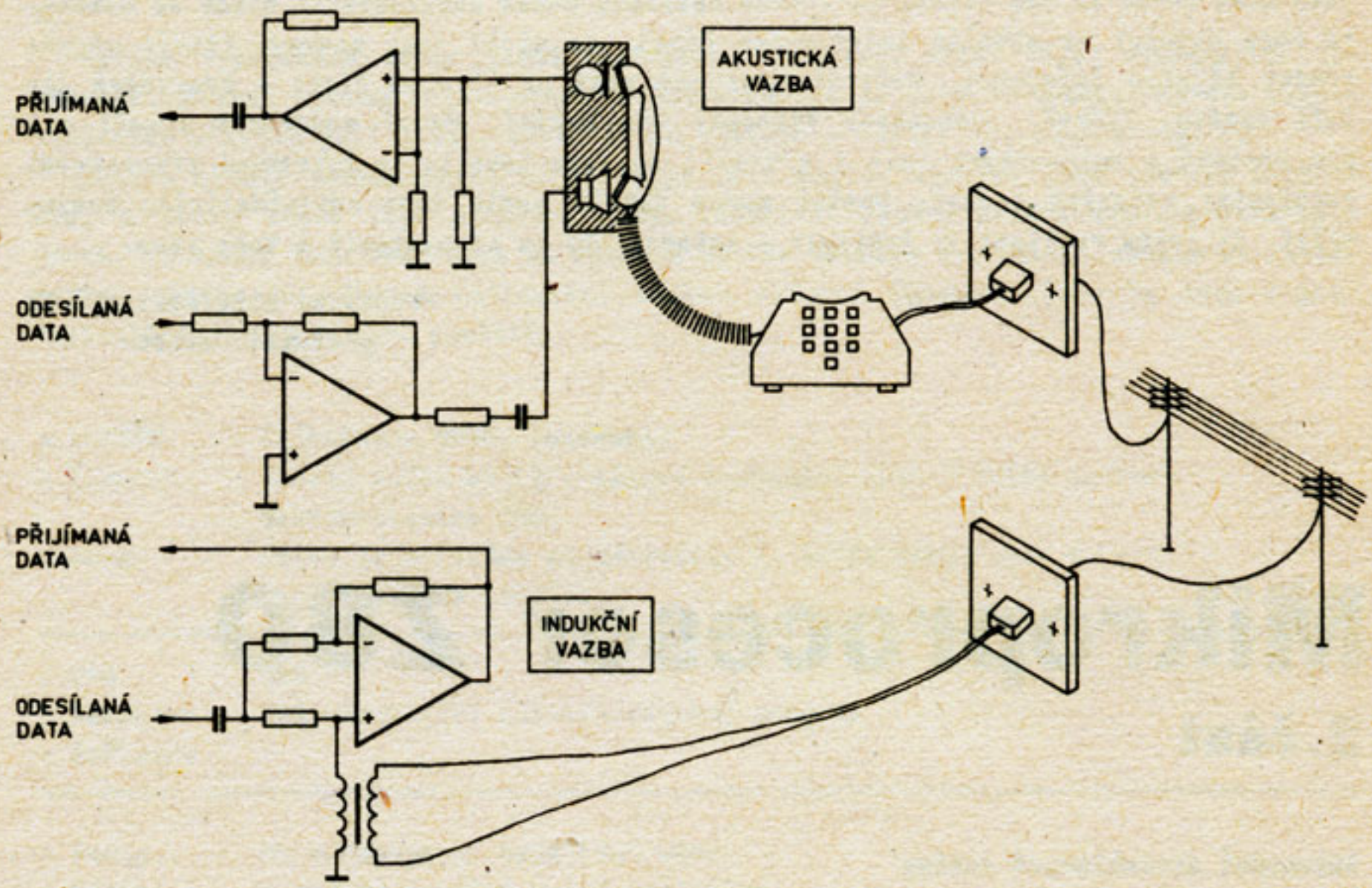
Akustický modem obsahuje malý mikrofon a reproduktor, podobně jako telefonní sluchátko, které se na modem pokládá tak, že jeho sluchátko "slyší" modulovaný signál vysílajícího modemu (a předává jej dále do sítě), zatímco do reproduktoru sluchátka přichází modulovaný signál ze sítě a je akustickou vazbou předán mikrofonu modemu pro jeho demodulaci. Dobrá akustická vazba je zajištěna gumovými mušlemi, do nichž se sluchátko telefonu pokládá. Přednost akustického modemu spočívá v jeho elektrické izolaci od telefonní sítě. Nevýhodou je možnost rušení okolními zvuky a vibracemi. K záporům se pojí i to, že sluchátka nemají standardní tvar, takže mnohá z nich "nepasují" do gumových mušlí.

Stále častěji se objevují modemy, o nichž se říká, že jsou napojena přímo do sítě, ale není tomu tak. Místo akustické vazby se u nich používá vazby prostřednictvím vestavěného transformátorku, tedy indukční. U tohoto provedení odpadají všechny nevýhody akustických modemů, výhoda elektricky nepřímého napojení zůstává. A problém poslední:

SOFTWARE

Právě na softwaru záleží, jak univerzální využití bude modem mít, jak jednoduchá

a příjemná bude jeho obsluha. Základní řečí computerové komunikace je kód ASCII (American Standard Code for Information Interchange) - mezinárodně přijatý standard sedmibitové reprezentace písmen anglické abecedy, některých nejúžívanějších gramatických a matemat. znaků a zákl. řídicích kódů. Se standardně kódovanými znaky datového souboru určeného pro přenos tedy nenastává žádný problém. Ten leží jinde. Zkušenější z vás vědí, že pomalu každý počítač má své zvláštnosti, jimiž se liší od ostatních. Proto i řídící software pro odběr a ukládání dat do počítače je vždy v něčem odlišný. A aby celá záležitost nebyla jen tak málo komplikována, i modemy mají rozdílné operační systémy, takže některé z nich musejí mít svůj speciální software. Dále k tomu přistupuje nutnost tvorby různých formátů dat v různých přenosových rychlostech.



Obr. Připojení modemů k telefonní síti

Uvedené problémy lze řešit třemi cestami - softwarem, hardwarem nebo kombinací obojího. Nejpohodlnější je pochopitelně řešení hardwarové. Pokud by však mělo obsáhnout celé spektrum problematiky, bylo by dost drahé. Proto se jím řeší převážně jen základní funkce (volba rychlostí přenosu a jejich kombinací). Ostatní obstará řídící software. Kromě toho, že takový program musí umět komunikovat na lince v obou směrech, musí rovněž být schopen bez chyby odebrat data z patřičného místa paměti a přijatá data umístit tam, kam je určeno, aniž by došlo k jakýmkoli kolapsovým následkům či jiným komplikacím.

Naprosto základním uživatelským požadavkem je, aby program byl tzv. menu-driven, tedy ovladatelný z menu promítaných na monitoru (nelze ztrácet čas listováním v manuálu). Všechna pro přenos připravená data by vždy měla být připravena tak, aby byla odebratelná jako jeden nepřerušený soubor dat. Přenos probíhá pomalu, proto je nutno brát v úvahu hledisko jeho efektivity, která se promítá do ušetřeného času i pěněz. Pro ten účel je vhodné mít k modemu připojenu speciální paměť, do níž uložíme soubor určený k odeslání, resp. i soubory přijímané. Zbytečně tak neblokujeme zařízení, která jinak stále používáme (především samotný počítač). Pochopitelně, že pro práci s touto pamětí musíme mít opět zvláštní řídící software.

V boji o trh vybavují výrobci své modemy některými zajímavými funkcemi a možnostmi. Jsou modemy, které po připojení k počítači mohou být plně řízeny v něm uloženým softwarem. Velmi příjemná je možnost automatického přijetí souboru dat, i když nejste doma (nebo se vám nechce či zrovna nemůžete modem obsluhovat). Někdy se nemůžete hned dovolat na několik telefonních čísel, na nichž jsou modemy, kterým chcete předat nějakou informaci. Počítač postupně volá na všechna čísla a odevzdává jim vaši zprávu. Zdárné dokončení činnosti vám oznámí. První modemisté museli mít pro kontakt s databankami (ale i přáteli) po ruce tabulky se zapsanými přenosovými rychlostmi volaných modemů. Dnešní modem dokáže podle tónu odpovídajícího modemu určit na jakou rychlost je nastaven a automaticky na ni nastavit i sebe sama.

Practical Electronics 6/86
přeložil -elzet-

Mikroprocesor Z80

2.část

Autonomní assemblerové rutiny

Pod tímto titulem uvádíme kratší programy ve strojovém kódu, které mají obecné uplatnění jako podprogramy velkého množství programů. Autonomní jim říkáme proto, že nezáleží na typu počítače (obsahujícího CPU Z80), do jehož paměti je umístíme - není v nich žádné volání nebo odskok do rutin ROMky, jejich funkce není závislá ani na používaném operačním systému. Jedním z cílů seriálu je pomoc programujícím členům Mikrobáze, kteří se s asemblerem a strojovým kódem teprve začínají seznamovat; své v něm najdou určitě i ti pokročilejší. Byli bychom rádi, kdybyste i vy přispěli svými vlastními autonomními rutinami, které velmi rádi otiskneme. Forma a výzva hesel popisu rutin byly uvedeny v minulém zpravodaji.

V několika dalších částech se zaměříme na pomocné rutiny, které sice obsahují některé monitory paměti (disassemblery), ale jichž můžeme výhodně využít i v kom-

binaci s jinými systémovými programy, resp. je použít při samostatných aplikacích. Postupně se seznámíte s rutinami: LENGTH (kalkulátor délky instrukcí; resp. počtu bajtů, které obsahují), FORMAT (výpis rutiny v její hexadecimální podobě; 1 instrukce na jedné řádce), NEWPC (rutina se dvěma volitelnými funkcemi - krokováním a tracingem), RELOC (relokace programu ve strojovém kódu s přepočtem všech absolutních adres).

Délka instrukcí Z80 leží v intervalu 1-4 bajtů. Proto není tak snadné určit, kde jedna instrukce končí a kde začíná další. Instrukce s indexovaným adresováním mají navíc tu zvláštnost, že jedny jsou jen o 1 bajt delší než jejich funkční analogie, jež pracují s párovým registrem HL, zatímco druhé (obsahující displacement) jsou delší o 2 bajty. Tento problém řeší rutina LENGTH. Obsahuje 4 porovnávací tabulky, s jejichž pomocí určíme délku instrukce, jejíž 1. bajt je adresován registrem HL.

Rutina vypočítává délku jedné instrukce. Chceme-li určit délku následující instrukce, musíme rutinu volat znova. Ovšem před tím je nutno k výchozí adrese minulé instrukce přičíst její délku, která je na výstupu z rutiny obsažena v reg. E.

Rutina LENGTH pochopitelně nerozlišuje mezi bajty, kterými jsou tvořeny instrukce, a těmi, které jsou prostými daty. Chtěli-li bychom předem vymezit intervaly adres, na nichž jsou instrukce (a vyhnout se tak datovým oblastem), musela by volací rutina obsahovat možnost definování těchto oblastí (podobně jako je tomu u profesionálnějších monitorů paměti).

* LENGTH	**Kalkulátor délky instrukce**
: Činnost	Určení počtu bajtů v platné instrukci instrukčního souboru mikroprocesoru Z80.
: Akce	Délka je určována porovnáváním 1. bajtu inkstrukce s DEFB porovnávacích tabulek.
: CPU	Z80
: Hardware	RAM obsahující cílový (objekt) program
: Software	-
: Vstup	HL adresuje 1. bajt instrukce
: Výstup	E obsahuje počet bajtů instrukce obsah AF, BC, E, HL je změněn
: Chyby	Data jsou zvažována jako bajty instrukcí
: Registry	AF, BC, E, HL
: Zásobník	2
: RAM	-
: Délka	139 (Rutina 66, tabulka 73)
: Cykly	Neuvedeny

: Třída 2	-diskrétní	* přerušitelná	* promovatelná
-xxx--	*opakovatelná	- relokovatelná	- robustní

```
:
TLEN1       EQU     14          ; Počet bajtů v tabulce 1
TLEN2       EQU     08          ; Počet bajtů v tabulce 2
TLEN3       EQU     25          ; Počet bajtů v tabulce 3
TLEN4       EQU     26          ; Počet bajtů v tabulce 4
:
LENGTH      LD      E, 01       ; Délka je min. 1 bajt      1E 01
:
LEN1        LD      A, (HL)     ; 1. bajt instrukce do A    7E
            INC     HL          ; Zvýšení adresy HL         23
            INC     E           ; Nastavení E na 2          1C
            CP      DDH         ; Jde o index. instr.,      FE DD
            JR      Z, LEN2     ;Začínající bajty DD        28 04
            CP      FDH         ; Nebo FD?                  FE FD
            JR      NZ, LEN3    ; Když NE, skok na LEN3     20 10
:
LEN2        PUSH    HL          ; Zpracování index. instr.  E5
            LD      A, (HL)     ; 2. bajt instr. do A       7E
            LD      HL, TAB1    ; Adr. TAB1 do HL           21 lo hi
            LD      BC, TLEN1   ; Do BC délka TAB1          01 0E 00
            CPIR                ; Hledání 2. bajtu v TAB1   ED 81
            POP     HL          ; Obnovení adr. 2. bajtu    E1
            RET     Z           ; Když bajt nalezen, je     C8
            INC     E           ; instr. o 1 bajt delší než 1C
            INC     HL          ; anal. HL instr.; jinak o 2 23
            RET                                             09
:
LEN3        CP      EDH         ; Jde o ED instrukci? Když  FE ED
            JR      NZ, LEN4    ; NE, bude to 1, 2, nebo 3 b. 20 0E
            LD      A, (HL)     ; 2. bajt ED instr. do A    7E
            LD      HL, TAB2    ; Zjištění, zda se jedná o  21 lo hi
            LD      BC, TLEN2   ; 4-bajt. instr. hledáním   01 08 00
            CPIR                ; v TAB2                    ED B1
            LD      E, 04       ; Když je bajt v TAB2       1E 04
            RET     Z           ; nalezen, je instr. 4-bajt. C8
            JR      LEN5        ; Jinak je 2-bajtová        18 10
:
LEN4        LD      HL,TAB3     ; Zjištění, zde jde o       21 10 hi
            LD      BC,TLEN3    ; 2-bajt. instr. v TAB3     01 19 00
            CPIR                                            ED B1
            RET     Z           ; Při nálezu návrat         C8
            INC     E           ; Jinak je instr. 3-bajtová 1C
            LD      BC, TLEN4   ;                           01 1A 00
            CPIR                                            ED B1
            RET     Z                                       C8
:
```

```
LEN5            DEC     E                                       1D
                DEC     E                                       1D
                RET                                             C9
:
:... Kódy instr. HL, u nichž je index analogie o 1 bajt delší
:
TAB1    DEFB    09H, 19H    ADD HL, BC      ADD HL, DE      09 19
        DEFB    21H, 22H    LD HL,  NN      LD (NN),HL      21 22
        DEFB    23h, 29h    INC HL          ADD HL, HL      23 29
        DEFB    2AH, 2BH    LD HL,(NN)      DEC HL          2A 2B
        DEFB    39H, E1H    ADD HL, SP      POP HL          39 E1
        DEFB    E3H, E5H    EX (SP),HL      PUSH HL         E3 E5
        DEFB    E9H, F9H    JO (HL)         LD SP, HL       E9 F9
:
:...4-bajtová skupina ED instrukcí
:
TAB2    DEFB    43H, 4BH    LD (NN), BC     LD BC,(NN)      43 4B
        DEFB    53H, 5BH    LD (NN), DE     LD DE,(NN)      53 5B
        DEFB    63H, 6BH    LD (NN), HL     LD HL,(NN)      63 6B
        DEFB    73H, 7BH    LD (NN), SP     LD SP,(NN)      73 7B
:
:...2-bajtové instrukce (vyjma index. a ED instr.)
:
        DEFB    06H, 0EH    LD B, N         LD C, N         06 0E
        DEFB    10H, 16H    DJNZ  N         LD D, N         10 16
        DEFB    18H, 1EH    JR N            LD E, N         18 1E
        DEFB    20H, 26H    JR NZ N         LD H, N         20 26
        DEFB    28H, 2EH    JR Z, N         LD L, N         28 2E
        DEFB    30H, 36H    JR NC, N        LD (HL), N      30 36
        DEFB    38H, 3EH    JR C,  N        LD A, N         38 3E
        DEFB    C6H, CBH    ADD A, N        CB instrukce    C6 CB
        DEFB    CEH, D3H    ADC A, N        OUT (N), A      CE D3
        DEFB    D6H, DBH    SUB N           IN A, VN/       D6 DB
        DEFB    DEH, E6H    SBC A, N        AND N           DE E6
        DEFB    EEH, F6H    XOR N           OR  N           EE F6
        DEFB    FEH         CP  N                           FE
:
:...3-bajtové instrukce (vyjma index. a ED instr.)
:
TAB4    DEFB    01H, 11H    LD BC, NN       LD DE, NN       01 11
        DEFB    21H, 22H    LD  HL, NN      LD (NN), HL     21 22
        DEFb    2AH, 31H    LD HL, (NN)     LD SP, NN       2A 31
        DEFB    32H, 3AH    LD (NN), A      LD A, (NN)      32 3A
        DEFB    C2H, C3H    JP NZ, NN       JP NN           C2 C3
        DEFB    C4H, CAH    CALL NZ, NN     JP Z, NN        C4 CA
        DEFB    CCH, CDH    CALL Z, NN      CALL NN         CC CD
```

DEFB	D2H, D4H	LP NC, NN	CALL NC, NN	D2 D4
DEFB	DAH, DCH	JP C, NN	CALL C, NN	DA DC
DEFB	E2H, E4H	JP PO, NN	CALL PO, NN	E2 E4
DEFB	EAH, ECH	JP PE, NN	CALL PE, NN	EA EC
DEFB	F2H, F4H	JP P, NN	CALL P, NN	F2 F4
DEFB	FAH, FCH	JP M, NN	CALL M, NN	FA FC

Pozornému čtenáři jistě neuniklo, že po rozšíření tabulek o syntaxi "chybějících" instrukcí bychom mohli tento program využití pro tisk assemblerového výpisu. Pochopitelně by bylo nutno program doplnit ještě o tisk adres, na nichž jsou instrukce uloženy, a o zápis číselných parametrů instrukcí, resp. i výpočet adres u instrukcí relativních skoků a displacementů.

Příště si probereme rutinu FORMAT, pomocí níž lze pořídit výpis instrukcí ve strojovém kódu. Její součástí bude i uvedená rutina LENGTH.

Assembler Routines for Z80
D.Barrow, 1985
přeložil a upravil
-elzet-

Mikrobázi hovory - horory

Někdy se člověk musí svěřit, aby mu dostavivší se úleva umožnila dále pokračovat v tom, čemu se trochu divně říká "práce s lidmi". Proto neberte následující věty nijak zle, nejsou tak myšleny. Fejeton se točí kolem telefonu - tlačítkového, s pamětí na posledně volené číslo... Promiňte, zrovna se rozezvučel...

"Dobrý den, tady X z města Y. Prosím vás, můj manžel by se vás rád zeptal, jak má zastavit hru."
"Dobře, tak mi ho dejte."
"To můžete říct mně, já mu to vyřídím."
"Vy umíte pracovat s počítačem?"
"Ne, to manžel si ho koupil."
"Tak mu vyřiďte, ať mi zavolá."
"Já bych ale ráda, kdybyste mi to řekl, já si to napíšu."
"Vážená paní, to takhle opravdu nepůjde. Problematika je to široká, navíc po-

třebuji vědět, do jaké míry ji váš manžel má zažitou. To byste teď mohla zavolat nějakému fyzikovi, že by váš manžel rád věděl, jak zastavit štěpnou reakci v reaktoru atomové elektrárny, že mu to pak vyřídíte."
"Tak to aspoň zkuste."
Spouštím ohňostroj programových fines na tajení her.
"Počkejte, já asi přece jen zavolám manželovi..."
"Na shledanou."
S povzdechem pokládám telefon. Ihned bzučí.
"Já jsem M z města N. Prosím vás, tady koukám, že v programu je číslo 30000. Jakto, když jsem se dočetl, že do bajtu se má přece vejít jen 255?"
"Do jednoho ano, ale když použijete dva současně, oba pak reprezentují interval 0 až 65535."
"Ale v každém je pak jen 255."
"To jistě. Ale oba dohromady dávají nikoli 8, ale 16 binárních číslic. Zkuste si sečíst 16 dvojek postupně mocněných exponenty od nuly do 15ti. Výsledek je 65535. Při třech bajtech to bude ještě o moc víc, atd."
"Ale jakto, když jeden bajt je 255?"
"Víte, co je dvojková soustava?"
"No, to moc ne."
"Tak se zahleďte do matematické učebnice, nebo se někoho zeptejte. Je to podobné jako desítková, jen je třeba si nad tím chvilku posedět."
"Ale je bajt jen 255?"
"Vždy jen v rozsahu nula až 255."
"No tak vidíte, skoro jste mě popletl."
"Velice nerad. Na shledanou."

Jojo... A zase bzučí!

"Já jsem ta X z Y. Mám tu manžela, jde o to zastavení hry."
"Tak mi dejte laskavě manžela."
"Já ho mám na druhé lince. Říkejte mi to a já mu to budu hned říkat do druhého sluchátka."
"On snad neumí mluvit?"
"Ale ano..."
"Prosím vás, tak ať mi zavolá sám, já s ním potřebuji být v přímém akustickém kontaktu!"
"Feďo, slyšíš? Pán říká, že mu máš zavolat, já ti to říkala! Ale ty neumíš ani zavolat do Prahy! Že jsi nekoupil to máslo, jak jsem ti ráno říkala..."

Tohle snad poslouchat nemusím. Abych nerušil, pokládám bez pozdravu. Bzukot.

"Volám z A. Koupili jsme před pár dny tenhle ... nó ... počítač.
Zapnem ho, i tu ...nó... televizi, a nic. Čím to je?"
"Vážená paní, příčin může být celá řada. Jaký počítač to je?"
"Ten z Tuzexu, to ... jak se ... Spectrum."
"Máte televizor vyladěný na patřičnou frekvenci, tedy jako ... no ... na kanál?"

"Jó, televize je Grundig, ta chodí, jéje!"
"A ladili jste ji na příjem obrazu z počítače?"
"Ale těma knoflíčkama sme točili porád sem a tam, a nic!"
"Když na počítač položíte ruku, hřeje?"
"Právě že jó! Že to bude tím?"
"Ne, to je v pořádku. Jediné, co vám mohu poradit, je, abyste se obrátila na servis, protože závad, které přicházejí pro tento případ do úvahy, může být celá řada. Chce to prokouknout, proměřit. Tak na shledanou."

Nestačím ani stisknout ENTER a je tu další telefon.

"Já jsem Z, volám z W. Prosím vás, chci dostat do Basicu pouk na hru. Jak to mám udělat?"
"Nahrajte Basic příkazem MERGE a pak ..."
"Moment, hned to bude!"
"To si uděláte pak, nezlobte se, nemám čas ..."
"Už to pouštím, chvilinku ...no, a je to. A co teď?"
"Přečtěte mi, kam pouk patří."
"Na LINE 2, na konec."
"Tak si vyeditujte druhý řádek."
"...nejde."
"Máte něco na obrazovce?"
"Jo, řádku nula, na ní nic není, pak je tam řádka 1 a 2."
"Tak si tu druhou vyeditujte."
"Nejde."
"A co to dělá? Bzučí to třeba?"
"Nic to nedělá."
"A co kurzor? Dá se s ním vůbec hýbat?"
"Je na řádce nula."
"On nejde posouvat?"
"A jak?"
"No přece kurzorovými tlačítky nebo příkazem LIST řádka!"
"Ajó, tak takhle..."
"Vážený pane, manuál se dává k počítači proto, abyste se jej - chcete-li - naučil ovládat, a tak šetřil čas druhým lidem. Na shledanou!"

Začínám rozumět pocitům rohožky... Bzučák.

"Já sem vám před chvílí volala. Že nám jako nejde to ...no... vobraz z počítače. Vy vůbec ničemu nerozumíte! Přišel soused, dal dozadu do televize takovej černej drát z počítače a už tam ten vobraz je! Já mám napsaný vaše méno a budu si na vás stěžovat, jak tam volbujete lidi! Že prej abych to nesla do servízu! Cha! Dyť vy ani nevíte, že ten počítač musí bejt tím drátem spojenej s televizí! Jaksidovolujetevůbecraditlidem! Tostetampěknáseb"

Dost! Telefon letí do hromady výpisů z tiskárny. A hned bzučí.
No počkej, já ti...

"Dobrý den, tady C z XXX. ZO Svazarmu z města D. Dozvěděli jsme se, že sháníte nějaké materiály pro zpravodaj Mikrobáze. Rádi bychom vám nabídli některá hardwarová zapojení našich členů."

Koktavě se probírám z mrákot a najednou mám chuť toho člověka obejmout. Svižně se s ním domlouvám a svět je náhle o poznání hezčí.

– elzet –

Zápis a čtení dat u mikropočítače IQ 151

Ukládání dat na magnetofonovou kazetu u mikropočítače IQ 151

Při použití mikropočítače IQ 151 pro různé druhy výpočtů, zpracování výsledků apod. vyvstává nutnost uchování vypočtených nebo shromážděných dat pro pozdější zpracování. Jedinou současnou možností je uchování dat na magnetofonovou kazetu.

Procedura, umožňující nahrávat data v průběhu programu na magnetofonovou kazetu, má tvar:

```
POKE HEX (EA), 2
CALL HEX (SB1C)
PRINT číslo řádku "DATA" proměnná
POKE HEX (EA), 0
```

Uspořádání nahraných dat vytvoříme příkazem procedury PRINT. To znamená, že jsou-li data např. delší než jeden řádek, napíšeme si cyklus, který nám očísluje řádky a seřadí proměnné jako data do těchto řádek. Např.:

```
POKE HEX (EA), 2
CALL HEX (CB1C)
FOR I:1 TO 5
PRINT I "DATA" A (I,1)", "A(I,2)", "...
NEXT I
POKE HEX (EA), 0
```

Tento program předpokládá, že v předchozím programu jsme si uložili data do proměnné A (I,J).

Po opětovném nahrání do paměti počítače budou mít v našem případě data tento tvar:

```
1 DATA A(1,1), A(1,2).... A(1,5)
.
.
5 DATA A (5,1), ..... A(5,5)
```

Je-li podprogram pro nahrávání dat na mgf. kazetu součástí delšího programu, je vhodné běh programu před spuštěním procedury zastavit a po zapnutí nahrávání mgf. vhodnou klávesou běh procedury spustit. Po nahrání dat je postup obdobný – běh programu se zastaví, následuje příkaz pro vypnutí mgf. a vhodnou klávesou se běh programu opět

spustí. Příklad krátkého programu s nahráváním dat na mgf. kazetu:

```
1 CLS -
2 FORI - 1TO3 -
3 FORJ = 1TO7 -
4 K = INT (RND(Ø) * 1Ø) -
5 A(I,J) = K*5 -
6 NEXTJ, I -
7 CLS:PRINT @ 10,1 "ZAPNI NAHRAVANI MGF."
:PRINT @ 12,1 "PO SPUSTENI ZMACKNI COKO
LIV"
8 Z = USR (HEX(FBO3)) -
9 IFZ = 13THEN1O -
10 POKEHEX (EA), 2 -
11 CALLHEX (CB1C) -
12 FORI = 1TO3 -
14  PRINT  "DATA"  A(I,1)",  "A(I,2)",
   A(I,3)",  A(I,4)",  A(I,5)",  A(I,6)"-
15 NEXTI -
16 POKEHEXEA /, O -
17 CLS:PRINT @ 10,1 "VYPNI MGF." -
```

Nahraná data mají tvar:

```
1 DATA 25, 0, 15, 25, 35, 10, 25-
2 DATA 35, 20, 20, 10, 35, 25, 20-
3 DATA 15, 40, 45, 40, 45, 25, 25-
```

Zvýšení spolehlivosti nahrávek programů v jazyce BASIC na mikropočítači IQ 151 prodloužením délky meziblokové mezery

Využíváme-li plně kapacitu programové řádky, může dojít k tomu, že při větším množství příkazů na jednom programovém řádku není modul BASIC 6 schopen včas přeložit řádku při přehrávání z magnetofonu a dochází buď ke zkomolení následujícího programového řádku, po kterém podle způsobu zkomolení dojde k přerušení a k výpisu chybového hlášení či vytvoření nového nesmyslného řádku, nebo nastane rovnou chybové hlášení a přeruší přehrávání. Proto je vhodné u těchto programů prodloužit při nahrávání na magnetofon délku meziblokové mezery.

Prodloužení provedeme buď z režimu BASIC přímým příkazem POKE 28,39 nebo z režimu MONITOR sekvencí

```
S1C  (SP)
ØØ1C:21 - 27  (CR)
R

READY
```

a nahráváme v režimu BASIC příkazem MSAVE. Při zpětném přehrávání z magnatofonu používáme příkaz MLOAD bez jakýchkoli úprav.

Rychlé nahrávání programů v jazyce BASIC na mikropočítači IQ 151 pomocí zkrácení meziblokových mezer

Máme-li delší program v jazyce BASIC, určitě uvítáme, když si ho budeme moci nahrát a přehrát přibližně dvojnásobnou rychlostí, než při běžném způsobu pomocí příkazů MLOAD a MSAVE.

Program v BASICu je uložen v paměti mikropočítače v hexadecimálním tvaru, a proto je možné nahrát a přehrát ho jako strojový kód. Navíc nahrávání strojového kódu je možné zrychlit zkrácením délky meziblokové mezery na minimum, protože neprobíhá při nahrávání překlad jednotlivých programových řádků.

Nahrávání na magnetofon:

Na adresu 1C (= 28 dekadicky), což je délka meziblokové mezery pro nahrávání, uložíme místo původního obsahu (21) délku 02 (= 2 dekadicky). Uložení provedeme buď z režimu BASIC přímým příkazem POKE 28, 2 nebo z režimu MONITOR sekvencí:

```
S1C  (SP)
001C:  21 - 02  (CR)
```

Po nahrávání je dále potřeba zjistit, kde končí v paměti uložený basicový program. Na adresách D0 a D1 (= 208 a 209 dekadicky) je uložena první volná adresa za programem. Nahráváme pak v režimu

MONITOR sekvencí:

WDO, [D1] [DO], O (CR)

Přehrávání z magnetofonu :

Při zpětném nahrávání programu z magnetofonu používáme v režimu MONITOR sekvencí:

L (CR)

bez úprav. Návrat do režimu BASIC provádíme obvyklým způsobem (R). Příklad:

klávesnice	obrazovka
(BR)	
S1C (SP)	001C 21-
02 (CR)	
SDO (SP)	00D0 30-
(SP)	00D1 11-
(CR)	
WDO, 1130, O (CR)	

Ing. Jana Suberová

Programátor EPROM pro PMD 85

Programátor byl zhotoven k mikropočítači PMD 85 a je určen pro EPROM typu 2708/8708/K573RF1. Tyto paměti v provedení PROM (MHB 8608) jsou v PMD použity pro monitor a BASIC-C a lze jimi modul rozšířit o uživatelské programy pouhým nasunutím do připravených objímek (6 kusů).

Programátor PRG-85 inzerovaný Teslou v uživatelské příručce stále ještě není vyráběn.

Programátor

Zapojení ze Sdělovací techniky č. 7/1986 bylo přizpůsobeno k PMD 85 a zjednodušeno pro paměti 1K. Programátor je připojen na kanál č. 4 GPIO. Brána A a bity PCØ, PC1 ovládají adresy, brána B v obousměrném režimu slouží pro zápis a čtení dat, PC2, PC3, PC6 a PC7 ovládají funkce programátoru. Konstrukčně je programátor umístěn v univerzální krabičce a propojen plochými kabely s PMD a síťovým zdrojem.

Napájecí zdroj

Paměti vyžadují napájení +5 V, -5 V, +12 V a programovací impulsy +26 V. Všechna napětí lze získat ze stávajícího síťového zdroje EA 1605 doplněním o zdvojovač a zkratuvzdorný stabilizátor 26 V. Ten je připojen na usměrňovací diody V9 až V12 sekce 12 V. Plošný spoj je připevněn k levému hornímu distančnímu sloupku, napětí jsou vyvedena na přídavný konektor (pětikolíkový nf), umístěný nad stávajícím konektorem.

Programové vybavení

Činnost programátoru je řízena programem, který umožňuje zápis a modifikaci adres

a dat, programování a kontrolu správnosti programování. Program nepovažujeme za zvlášť komfortní, ale svůj účel plní.

Komentář k programátoru EPROM

Pro zkrácení doby nahrávání programu je nahráván pouze vlastní program od adresy 3400H do 368BH. Po novém zavedení do paměti je proto třeba uložit od adresy 6000H instrukce C3, 52, 35.

První odstartování programu se provede od adresy 34FBH. Tím se vyplní oblast pro uložení vzorového programu hodnotami FF (oblast 4000H - 43FFH). Další spouštění se provádí od adresy 3500H a program uložený v oblasti zůstává zachován.

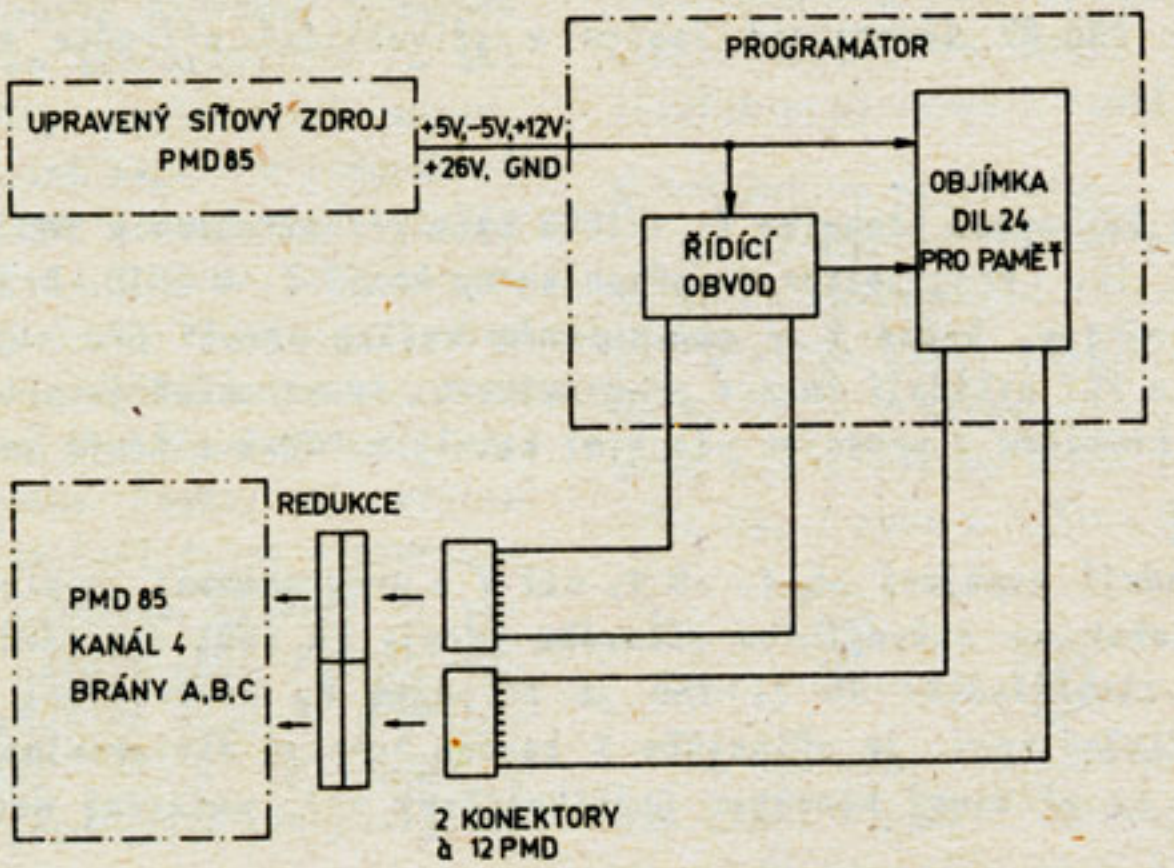
Přihlášení operátora se provede stiskem klávesy podle zprávy v dislogovém řádku. Neobsahuje-li zpráva volbu klávesy, provede se přihlášení stiskem libovolné klávesy.

Pro ukládání nebo čtení informací z příslušných lokací se používá příkazů monitoru, který se přihlásí zprávou OS READY.

Vzorový program je možno do paměti ukládat s využitím všech příkazů monitoru: používání KEY, magnetofonu, příkazů SUB, MEM, PTL atd. Kromě toho je možno pro přesun programu uloženého na jiném místě (pro ROM modul) použít programový segment od adresy 3656. Tento programový modul přenese data z lokace zadané adresami uloženými na: začátek 365DH = LB 365EH = HB
konec 3660H = LB 3661H = HB,
přičemž pokud provádíme další spouštění až od adresy 365C, jsou datové bloky seřazeny postupně za sebou.
(Ukazatel počátku ukládání v oblasti vzor. programu je na adresách 367BH = LB a 367CH = HB).

Při volbě režimu čtení se provede načtení obsahu paměti ROM do lokace 3000H - 33FFH, příkazy monitoru je možno oblast prohlížet (DUMP 3000).

Při volbě režimu programování se zkontroluje, zda je paměť dokonale smazána.



Blokové schéma připojení programátoru EPROM 8708

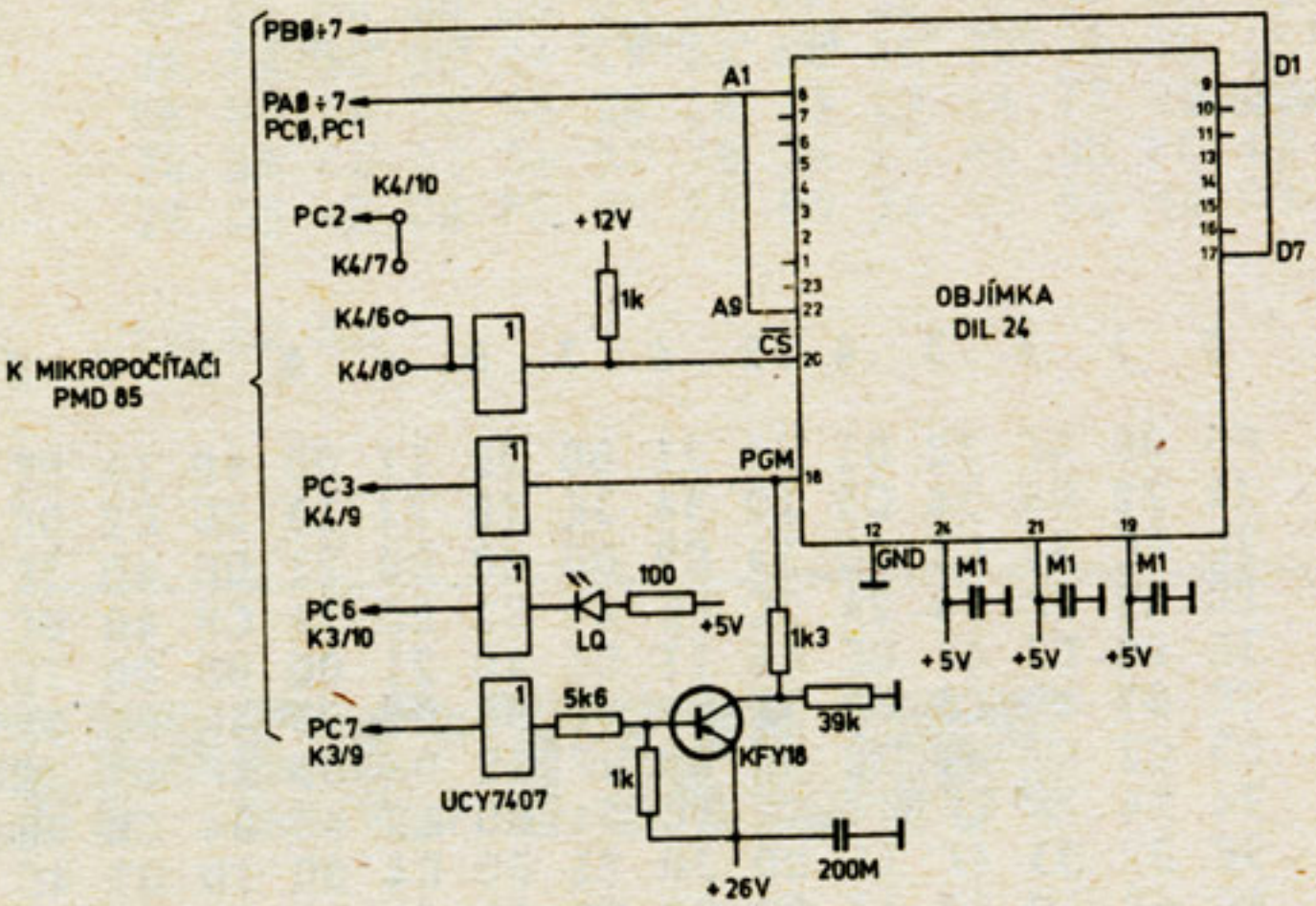


Schéma zapojení programátoru

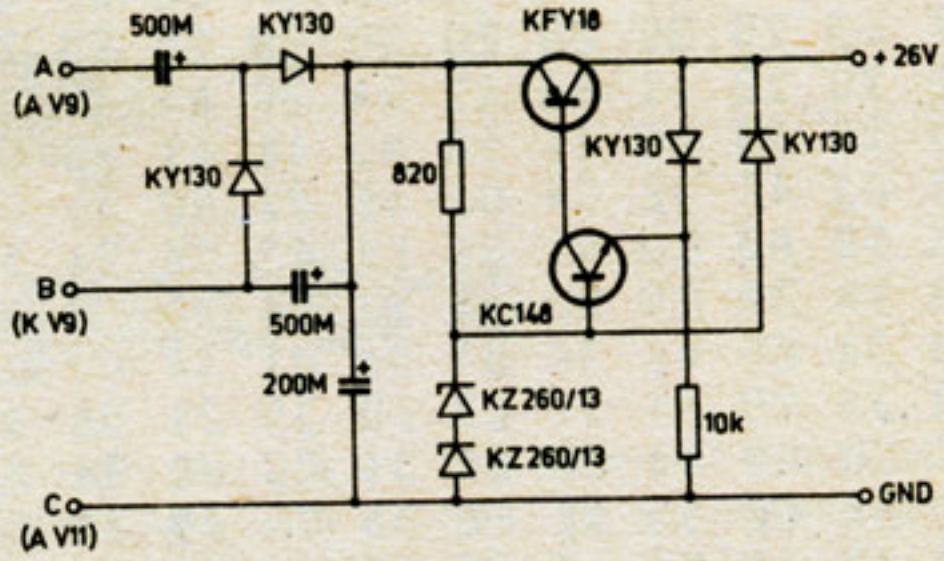


Schéma zapojení stabilizátoru 26 V

Pokud není smazána, je nutno volit režim čtení, a tím se operátor dostane do monitoru. V monitoru se odesláním příkazu W␣P provede naprogramování EPROM bez ohledu na předchozí stav. Naprogramování se provede podle vzorového programu, uloženého v lokaci 4000H - 43FFH (toto je možno využít při doplňování paměti EPROM v místech, která jsou ve stavu FF).

Po naprogramování se provede kontrola správnosti naprogramování porovnáním se vzorovým programem. Výsledný protokol je uložen bezprostředně po ukončeném programování v lokaci 3000H - 33FFH, přičemž místa, kde není chyba, mají hodnotu 28. Chyby jsou znázorněny jinou hodnotou (tou, která má v příslušném místě správně být). Porovnávací segment je možno vyvolat příkazem JUMP 3555 kdykoliv při práci v monitoru.

Pro racionalizaci práce se doporučuje naplnit KEY hodnotami:

KØ = SUB 4 K1 = DUMP 4 K2 = DUMP 3 K3 = MEM 4

K11 = JUMP 3500

...

ADR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3400	F5	00	AF	32	01	34	21	00	30	11	00	40	1A	BE	CA	17
3410	34	22	00	34	C3	19	34	3E	28	77	13	2C	C2	0C	34	24
3420	7C	FE	34	C2	0C	34	C9	00	00	00	21	00	40	3E	FF	77
3430	2C	C2	2F	34	24	7C	FE	44	C2	2D	34	C9	40	8C	00	00
3440	3E	80	D3	4F	0E	C6	11	00	40	21	00	80	00	7D	D3	4C
3450	7C	D3	4E	C3	80	36	06	42	3E	07	D3	4F	3E	0E	D3	4F
3460	05	C2	60	34	3E	0F	D3	4F	3E	06	D3	4F	13	2C	C2	4D
3470	34	24	7A	FE	44	C2	4C	34	0D	C2	46	34	C9	00	00	00
3480	3E	82	D3	4F	11	00	30	21	00	C4	00	7D	D3	4C	7C	D3
3490	4E	DB	4D	12	13	2C	C2	8B	34	24	7A	FE	34	C2	8B	34
34A0	C9	00	00	00	00	22	74	C0	CD	8C	8A	C9	00	00	00	00
34B0	20	20	20	43	48	43	45	53	20	50	52	4F	47	52	41	4D
34C0	4F	56	41	54	20	4E	45	42	4F	20	43	49	53	54	3F	20
34D0	50	2F	43	20	20	20	20	20	20	20	20	20	20	20	20	20
34E0	20	20	20	50	41	4D	45	54	20	4A	45	20	53	50	41	54
34F0	4E	45	20	53	4D	41	5A	41	4E	41	0D	CD	2A	34	00	00
3500	3E	82	D3	4F	3E	C4	D3	4E	21	80	35	CD	A5	34	21	B0
3510	34	CD	58	88	CD	A1	84	FE	43	CA	24	35	FE	50	CA	30
3520	35	C3	14	35	CD	80	34	21	94	35	CD	A5	34	C3	40	8C
3530	CD	80	34	CD	46	36	B7	3A	01	34	07	07	07	D2	49	35
3540	21	E0	34	CD	A5	34	C3	00	35	21	D6	35	CD	A5	34	C3
3550	40	8C	CD	40	34	CD	80	34	CD	02	34	B7	3A	01	34	07
3560	07	07	DA	6E	35	21	FE	35	CD	A5	34	C3	40	8C	21	21
3570	36	CD	A5	34	C3	40	8C	00	00	00	00	00	00	00	00	00
3580	20	20	20	5A	41	53	55	4E	20	50	41	4D	45	54	20	32
3590	37	30	38	0D	50	41	4D	45	54	20	4A	45	20	4E	41	43
35A0	54	45	4E	41	20	44	4F	20	4C	4F	4B	41	43	45	20	33
35B0	30	30	30	48	2D	33	33	46	46	48	0D	20	20	20	50	41
35C0	4D	45	54	20	4A	45	20	53	50	41	54	4E	45	20	53	4D
35D0	41	5A	41	4E	41	0D	5A	41	50	49	53	20	50	52	4F	47
35E0	52	41	4D	20	4F	44	20	34	30	30	30	48	20	50	4F	54
35F0	4F	4D	20	50	52	49	4B	41	5A	20	23	20	50	0D	20	20
3600	20	50	41	4D	45	54	20	4A	45	20	53	50	52	41	56	4E
3610	45	20	4E	41	50	52	4F	47	52	41	4D	4F	56	41	4E	41
3620	0D	20	20	20	50	41	4D	45	54	20	4A	45	20	53	50	41
3630	54	4E	45	20	4E	41	50	52	4F	47	52	41	4D	4F	56	41

```
3640    4E 41 0D 00 00 00 06 FF 3E 78 32 0C 34 CD 02 34
3650    3E 1A 32 0C 34 C9 21 00 40 22 7B 36 11 00 00 01
3660    FF 03 2A 7B 36 2B 1B 23 13 1A 77 79 BB C2 67 36
3670    78 BA C2 67 36 22 7B 36 C3 40 8C FF 43 00 00 00
3680    1A D3 4D 06 FF 05 C2 85 36 C3 56 34 03 FF FF FF

6000    C3 52 35
```

SPECTRUM

Kanály ZX Spectra

Jak připojit tiskárnu k ZX Spectru? Jak pro ni napsat obslužný program ve strojovém kódu? Jak informovat tento program, co má vlastně tisknout? Na otázky tohoto druhu dává odpověď následující příspěvek.

ZX Spectrum je počítač, u kterého je již počítáno s tím, že jeho uživatelé k němu budou chtít připojovat různá periferní zařízení (nejčastěji tiskárnu). Systém, který umožňuje připojení těchto periferií, je naprosto univerzální. Ke Spectru lze totiž připojit tiskárnu s libovolným druhem komunikace. Abychom toto připojení mohli provést, musíme znát, jak pracují kanály a linky našeho počítače. Počítač totiž se svými periferiemi komunikuje právě pomocí těchto kanálů a linek.

Linka je cesta, kterou se posílají jednotlivé zpracovávané znaky z počítače do periférie, nebo jsou periférií vysílány a počítač je přijímá, přičemž typ periférie je označován jako kanál. Potom kanál nemá nic společného s hardwarovou konstrukcí té které periférie, jedná se o čistě softwarové zajištění komunikace mezi hlavním programem a programy pro obsluhu periférií. Komunikace zde probíhá sériově, po jednom znaku. Konkrétně to vypadá tak, že chce-li program vyslat nějaký text na periferní zařízení, bere postupně jednotlivé znaky (bajty) tohoto textu a předává je určitým

způsobem podprogramu, který zpracování znaků zajistí (nejčastěji vyslání na port apod.). Podprogram je volán vždy znovu a znovu při vyslání dalšího znaku. Obrovskou výhodou tohoto uspořádání je fakt, že znaky, které mají být vyslány na periferní zařízení, vůbec nemusejí být uloženy v paměti počítače a tudíž lze pracovat s datovými soubory daleko většími, než operační paměť. Při vstupu z periférie do počítače probíhá komunikace obdobně. Program zavolá vstupní podprogram, který převezme znak z periferního zařízení a předá jej hlavnímu programu.

Nyní se podívejme, jak taková komunikace vypadá u ZX Spectra. Všechny vstupní i výstupní podprogramy jsou samozřejmě napsány ve strojovém kódu (jinak to ani nejde, vezmeme-li v úvahu rychlost ZX Basicu). Znak, který se přijímá nebo vysílá, je uložen v registru A (akumulátoru) mikroprocesoru Z-80. Volání výstupního podprogramu, chceme-li např. vytisknout znak "A", může vypadat následovně:

```
                .
                .
                .
PRT1:           LD      A, "A"          ; znak do akumulátoru.
                CALL    OUTP            ; volání výstupní rutiny.
                .
                .
                .
-------------------------------------------------
OUTP:           .                       ; vyslání znaku
                .                       ; na periferní zařízení.
                RET                     ; návrat do hl. programu.
```

Toto je ovšem zjednodušený příklad, neboť řídící program předpokládá, že výstupní podprogram začíná na určité konkrétní adrese, což zdaleka nemusí být splněno (například toto místo již může být obsazeno jiným programem). Proto se tento problém řeší tabulkou, ve které jsou uvedeny počáteční adresy vstupních a výstupních podprogramů, a hlavní program při volání potřebného podprogramu vždy bere patřičnou adresu z této tabulky:

```
                .
                .
                .
PRT2:           LD      A,"A"           ; znak do akumulátoru.
                LD      HL,(TAB)        ; adresa rutiny do HL.
                LD      DE,CONT         ; adresa pokračování
                PUSH    DE              ; programu do stacku.
                JP      (HL)            ; zavolání rutiny.
CONT:           .                       ; zde pokračuje hlavní
                .                       ; program.
                .
-------------------------------------------------
TAB:            DEFW    OUTP            ; adresy všech výstupních
                DEFW    OUTP 2          ; rutin.
                DEFW    OUTP 3
                .
```

```
                .
                .
OUTP:           .                       ; vyslání znaku
                .                       ; na periferní zařízení.
                RET                     ; návrat.
```

Nyní jsme vyřešili případ, že výstupní podprogram může být uložen na libovolném místě v paměti a jeho napojení na systém se provede zapsáním jeho počáteční adresy do patřičného místa tabulky. Potíže nastanou tehdy, jestliže tato tabulka nebude souvislá, ale rozdělená na více částí, mezi kterými budou např. umístěny vstupní a výstupní obslužné podprogramy. Pak musíme přidat ještě další tabulku, ve které budou odkazy na patřičná místa původní tabulky s adresami obslužných podprogramů (dvojnásobná nepřímá adresace). Celá tato konstrukce vypadá na první pohled dosti komplikovaně, proto její činnost napřed vysvětlím na funkci ZX Basicu.

ZX Spectrum má šestnáct linek číslovaných od nuly do patnácti. Každá z těchto linek může být zároveň vstupní i výstupní. Máme-li samotný počítač (tj. bez připojeného ZX Interface 1), tento počítač sám o sobě již používá dva kanály, a sice klávesnici a pole obrazovky (ty jsou, sice součástí Spectra, ale komunikace s nimi probíhá stejně jako s "vnějšími" periférierni). Třetím kanálem může být ZX Printer, je-li připojen. Majitelé ZX Interface 1 navíc mohou používat sériový výstup a vstup RS 232, network a microdrive. Jak se s těmito kanály pracuje? Pro připojení některého kanálu k lince slouží příkaz OPEN. Uvedu příklad:

```
OPEN # 5, "s"
```

připojí pátou linku na obrazovku. Kanály se u Spectra označují jednotlivými písmeny podle následující tabulky:

klávesnice	"k"		(keyboard)
obrazovka	"s"		(screen)
ZX Printer	"p"		
RS 232	"t"	nebo "b"	(text, binary)
network	"n"		
microdrive	"m"		

Kanály "t", "b", "n" a "m" se týkají jen příkazů vyžadujících ZX Interface 1. Příkaz PRINT vždy vysílá znaky do linky dvě, zatímco příkaz INPUT bere znaky standardně z linky jedna. Lze však tyto příkazy směrovat i na jiné linky. Např.:

```
PRINT # 5; "NAZDAR"
```

vytiskne text na obrazovce (jestliže jsme předtím samozřejmě provedli OPEN 5,"s"). Jestliže ovšem napíšeme

```
OPEN # 5, "p": PRINT # 5; "NAZDAR"
```

vytiskne se text na tiskárně. Totéž nastane při

```
PRINT # 3; "NAZDAR"
```

z čehož vidíme, že příkaz LPRINT je v podstatě totéž jako PRINT 3. Podobně PRINT je totéž jako LPRINT 2. Příkaz INPUT lze také psát ve formě

```
INPUT # 1; a
```

ale to u samotného Spectra (bez Interface 1) nemá význam, protože z tiskárny nebo obrazovky stejně nelze číst. Dalším příkazem pro práci s linkami je CLOSE.

Slouží k uzavření dané linky (odpojení vstupní nebo výstupní rutiny). Jestliže provedeme

```
CLOSE # 5: PRINT # 5; "NAZDAR"
```

způsobí to chybové hlášení (Invalid I/O device).

Programy ve strojovém kódu komunikují s periferiemi také pomocí linek, používají dokonce stejné linky jako ZX Basic. Jde jen o to, jak si mají potřebnou linku otevřít a používat ji. V systémových proměnných Spectra je proměnná STRMS (start 23568, délka 38 bytů). V této proměnné jsou uloženy informace o všech linkách. Proměnná je organizovaná jako devatenáct dvoubajtových hodnot příslušejících jednotlivým linkám. Proč je jich devatenáct místo šestnácti? První tři hodnoty patří totiž linkám mínus tři až mínus jedna, které používá Basic (editor, tisk chybových hlášek a reportů "Start the tape..." atd.), které nejsou ani obvyklými příkazy přístupné a do kterých není dobré zasahovat pomocí příkazů POKE ani jinak. Čtvrtá až devatenáctá hodnota udávají relativní posunutí (rozumí se od začátku) v tabulce obsahující adresy vstupních a výstupních (dále jen I/O) rutin představujících obsluhu jednotlivých kanálů. Tato důležitá tabulka začíná na adrese, která je obsahem systémové proměnné CHANS (adresa 23631). Je-li některá hodnota v proměnné STRMS nulová, je daná linka uzavřená (nevede na žádný kanál). Je-li linka otevřená, patřičná položka v STRMS je rovna alespoň jedné. Pro lepší pochopení následuje výpis těchto proměnných tak, jak jsou inicializovány po zapnutí počítače:

```
STRMS:      DEFW    1, 6, 11            ; linky - 3 až - 1.
            DEFW    1                   ; linka 0.
            DEFW    1                   ; linka 1.
            DEFW    6                   ; linka 2.
            DEFW    16                  ; linka 3.
            DEFW    0, 0, 0, ....,0     ; ostatní (uzavřené).
```

```
CHANS:      DEFW    CHAN_AD             ; adresa tabulky kanálů.
```

```
CHAN AD :   EQU     ¤                   ; tabulka kanálů.
            DEFW    # 09F4              ; adresa výstupní a
            DEFW    # 10AB              ; vstupní rutiny pro
            DEFB    "K"                 ; klávesnici,
            DEFW    # 09F4, #15C4       ; adresy rutin pro
            DEFB    "S"                 ; obrazovku,
            DEFW    #0F81, #15C4        ; adresy rutin pro
            DEFB    "R"                 ; Basic editor.
            DEFW    # 09F4, #15C4       ; adresy rutin pro
            DEFB    "P"                 ; ZX Printer.
```

Tabulka kanálů na adrese (CHANS) je organizována jako čtyři až šestnáct pětibajtových položek (podle toho, kolik kanálů je vytvořeno). Každých pět bajtů obsahuje adresu výstupní rutiny první dva bajty), dále adresu vstupní rutiny (druhé dva bajty) a písmeno (pátý bajt). Písmeno udává typ kanálu, tj. "k", "s", "p" atd. Sedmý

bit pátého bajtu může být nastaven nebo zrušen podle toho, zda daný kanál je vytvořen trvale nebo přechodně. Toto nás nemusí zajímat, protože pro nás jsou důležité jen trvalé kanály (s resetovaným sedmým bitem). Přechodné kanály používá ZX Interface 1 pro příkazy MOVE, SAVE*, LOAD*, MERGE* a VERIFY*. Ukazatel do tabulky kanálů dostaneme jako součet obsahu proměnné CHANS a obsahu patřičné položky proměnné STRMS, přičemž takto vzniklé číslo je adresou druhého bajtu v pětici (protože nulová hodnota položky v STRMS není pro otevřenou linku možná, pointer ukazuje na druhý bajt). Adresu I/O rutiny musíme bohužel získávat takovýmto složitým způsobem. Druhá tabulka (obsahující seznam vytvořených kanálů) není totiž souvislá, poněvadž mezi jednotlivými pěticemi bajtů mohou být ještě vyrovnávací paměti (buffery) pro network a microdrive.

Naštěstí můžeme volat také rutiny z ROM, které za nás hodně zařídí (abychom nemuseli počítat a hledat adresu I/O rutiny při vysílání každého znaku). K dispozici jsou celkem tři, které budeme potřebovat. Především je to rutina CHAN_OPEN (adresa 1601), která umožní směrovat vstup a výstup následujících znaků na určitou linku (která již ovšem musí být otevřena). Číslo aktualizované linky uložíme při volání rutiny do akumulátoru. Rutina v podstatě provede to, že uloží adresu v tabulce kanálů CHANS do systémové proměnné CURCHL (adresa 23633), aby se vždy nemusela vypočítávat. Máme-li takto aktualizovaný kanál, můžeme na něj vysílat nebo z něj přijímat znaky. Vyslání znaku provede rutina PRINT_CHARACTER (na adrese 0010), která se dá volat instrukcí RST 10. Kód znaku je při volání rutiny v akumulátoru. Chceme-li naopak znak číst, zavoláme rutinu INPUT_AD (na adrese 15E6). Rutina uloží přečtený znak do akumulátoru a je-li tento znak platný, nastaví příznak přenosu C. Rutiny PRINT_CHARACTER a INPUT_AD používají na začátku a konci instrukci EXX, takže nemusíme při jejich volání uschovávat aktuální sadu registrů do stacku. Pro názornost nyní uvedu příklad, jak vypadat již známý text na obrazovce, tentokrát však pomocí strojového kódu:

```
END:        EQU     #FF             ; zarážka.

PRT_TXT:    LD      A,#02           ; číslo kanálu do A .
            CALL    CHAN_OPEN       ; aktualizace kanálu.
            LD      HL,TEXT         ; adresa textu do HL.
LOOP:       LD      A, (HL)         ; znak do A.
            CP      END             ; test na zarážku.
            RET     Z               ; návrat při zarážce.
            RST     PRINT_CHAR      ; vyslání znaku na "s".
            INC     HL              ; pointer na další znak.
            JR      LOOP            ; opak. pro další znak.

TEXT:       DEFM "NAZDAR"           ; zobrazovaný text.
            DEFB    END             ; konec textu.
```

Teď už jde jen o to, jak napojit na stávající kanálový systém naši I/O rutinu. Je potřeba uložit její adresu na patřičné místo v tabulce kanálů (CHANS). Princip je jasný, uvedu zde proto program pro napojení výstupní rutiny pro tiskárnu na třetí linku (ta je tiskárně standardně vyhrazena):

```
OPEN_P:   LD    HL,(2*6+STRMS)      ; posunutí od CHANS do HL.
          LD    DE,(CHANS)          ; (CHANS) do DE.
          ADD   HL,DE               ; adresa v tabulce CHANS
                                    ; v HL.
          PUSH  HL                  ; HL do stacku.
          LD    HL,OUTP-OPEN_P      ; rozdíl adres do HL.
          ADD   HL,BC               ; adresa OUTP v HL.
          POP   DE                  ; adresa v CHANS v DE.
          EX    DE,HL               ; záměna DE a HL.
          LD    (HL),D              ; uložení adresy OUTP
          DEC   HL                  ; do tabulky v CHANS.
          LD    (HL),E
          RET                       ; návrat.
```

```
OUTP:     .                         ; vyslání znaku
          .                         ; na periferní zařízení.
          RET                       ; návrat.
```

Adresa rutiny OUTP je určena jako součet relativního posunutí od adresy OPEN_P a obsahu registru BC. Při volání funkce USR z Basicu je totiž adresa této rutiny uložena do BC.

Ještě několik poznámek k umístění I/O rutin do paměti Spectra. Rutiny lze samozřejmě dát do paměti kamkoli. Nejlepší je ovšem umístění do printer-bufferu od adresy 23296, protože nepoužívá -li se ZX Printer, tato oblast je volná. Jestliže se sem rutina nevejde, dáme ji někam nad RAMTOP (alespoň adresa posledního CLEARu plus jedna). Zásadně nedoporučuji umísťovat rutiny do řádek Basicu nebo do nějaké proměnné vytvořené příkazem DIM. Je to sice pohodlné, nicméně to způsobuje "záhadné" hroucení systému při posunutí basicového programu na jiné adresy, než si myslíme (program v Basicu totiž nemusí začínat na adrese 23755, jak se mnozí majitelé Spectra mylně domnívají). Nechci se v této souvislosti nikterak dotýkat autorů programů pro Centronix uveřejněných v AR A 8/86 a v Perspektivách, ale tyto programy jsou pro nás nevhodné. Doporučuji používat rutinu z AR, její inicializaci je však třeba provést program OPEN_P. Autor článku totiž předpokládá, že adresa výstupní rutiny je pevně na adrese 23749, což nemusí být vždy splněno.

Majitelé ZX Interface 1 mají používání tiskáren velmi usnadněno.Všechny standardní programy pro Spectrum ve strojovém kódu (GENS, MONS, PASCAL a C kompilátory, tasword a mnoho dalších) mají příkazy pro tiskárnu a volají ji přes třetí linku. Stačí proto tuto linku nasměrovat na patřičnou periférii. Je-li tiskárna připojena na RS 232, provedeme pouze

```
FORMAT "t"; baud: OPEN #3,"t"
```

a můžeme tisknout. Jinak použijeme výše uvedené programy. Je také možno provést.

```
OPEN #3; "m"; 1; "text"
```

a vypsat text z programu na microdrive a pak text z microdrive libovolným způsobem vytisknout. Jedna z možností je

```
MOVE "m"; 1; "text"  TO #3
```

Používání linek a kanálů na Spectru má velké výhody. Nejde zdaleka jen o připojo-

vání tiskáren, ale i čteček a děrovaček děrné pásky, hlasového výstupu, externích klávesnic a dalších periférií. Používáním kanálů se vyhneme různým těžkostem při využívání datových souborů vytvořených na jiném počítači (nutnost ukládat soubor do paměti jako CODE apod.). Já sám pracuji na Spectru s RAM-diskem, který rovněž volám jako kanál. Významná je také možnost takového volání kazetového magnetofonu, na kterém pak můžeme mít nahraná data pomocí příkazu PRINT, která lze příkazem INPUT pohodlně číst.

Jiří LAMAČ

Obrazová paměť ZX Spectra

1. část

Nejsympatičtější je zjištění, že na rozdíl od ZX 81 je paměť ZX SPECTRA na pevném místě a neplave. To nám bezesporu ušetří spoustu práce. První bajt obrazové paměti je na adrese 16384. Samotná paměť sestává ze dvou částí, na sobě v podstatě nezávislých:

1 - paměť kresby
2 - paměť atributů

Paměť kresby

obsahuje každý jednotlivý zobrazitelný bod obrazu a určuje, zda je "prázdný" (log. 0) nebo "plný" (log. 1). Paměť kresby tedy určuje vzhled "černobílého" podkladu kresby, kterou můžeme vybarvit pomocí atributů.

Paměť atributů

určuje barvu pozadí (PAPER) a kresby (INK), nižší nebo vyšší intenzitu barvy (BRIGHT 0, resp. BRIGHT 1) a blikání jednotlivých atributů (FLASH 1, resp. FLASH 0).

Když se začneme zajímat o rozsahy jednotlivých pamětí, dojdeme k zajímavým číslům:

$$\text{Paměť kresby} = \frac{\text{Počet TV řádků x rozliš. schopnost řádku}}{B} =$$

$$= \frac{192 \times 256}{B} = 6144$$

Paměť atributů = Počet znakových řádků x počet znak. míst = 24 x 32 = 768

Celkem = 6144 + 768 = 6914

To je také rozsah paměti, kterou obsáhneme třeba při povelu:

```
100 SAVE "xxx" SCREENS
```

V obou případech na první pohled překvapí, že je paměť větší, než je běžně známý rozsah obrazovky. To je způsobeno tím, že paměť obrazovky zahrnuje i dva editační řádky, které jsou běžným Basicovým programem nedostupné - nebo dostupné jen obtížně. Obrazovka si je však musí pamatovat. Dostupné jsou snadno programem ve strojovém kódu.

Obě tyto části jsou v posloupnosti paměti těsně za sebou. Paměť kresby je od adresy 16384 až do 22527, paměť atributů od adresy 22528 do 23296.

Z toho vyplývá, že obrazovka se dá nahrát dvěma způsoby, a to buď výše uvedeným, nebo povelem:

```
SAVE "xxx" CODE 16384, 6912
```

Oba tyto povely jsou naprosto rovnocenné, ovšem druhá alternativa nám dává možnost náhrát jen určitou část obrazovky.
Tak na příklad povelem:

```
SAVE "yyy" CODE 16384, 2048
```

nahrajeme pouze první třetinu obrazovky v černobílé kresbě.

Organizace a názvosloví

Nejdříve si musíme sjednotit názvosloví, abychom si rozuměli.

Rozlišovací schopnost obrazovky je sice víc než dvakrát menší, než u normálního televizního obrazu, ale přesto je ještě velmi vysoká a tvoří podle mého soudu onu rozumnou rovnováhu mezi slušnou schopností rozlišení a enormně velkým kusem "sežrané" paměti pro potřeby obrazovky.

Když uvažujeme celou pracovní plochu obrazovky ohraničenou okrajem (BORDER) i s oněmi zmíněnými editačními řádky, zobrazí se ve svislém směru celkem 192 televizních řádků - na každém řádku můžeme ve vodorovném směru zobrazit celkem 256 jednotlivých bodů. V originálu manuálu ke SPECTRU se bodu říká PIXEL a řádku PIXEL LINE. Slovník nám řekne, že PIXEL je "nejmenší obrazový element". To je pro náš účel poněkud dlouhé, takže budeme pro termín PIXEL používat české slovo bod a místo PIXEL LINE budeme mluvit o bodovém řádku. Protože však každý obrazový bod je prezentován jedním bitem paměti, můžete stejně dobře obrazovým bodům říkat obrazové bity a řádkům bitové řádky.

O bodech mluvíme, kdykoli vydáváme povel PLOT nebo DRAW a určujeme jejich souřadnice.

Na všech bodových řádcích lze tedy zobrazit
192 x 256 = 49152 bodů.
To je dost úctyhodné číslo. Kdyby každý tento bod měl být zapamatován samostatně, spotřebovala by paměť obrazovky celou paměť SPECTRA. Proto to museli konstruktéři

"romky" vyřešit trochu jinak.

Stačí si uvědomit, že k zapamatování jednoho bodu obrazovky vlastně nepotřebujeme obsazovat celé jedno paměťové místo (jeden bajt adresy). Každé paměťové mrsto (bajt) sestává z osmi prvků (bitů). Jeden bit pak zcela stačí k zapamatování stavu jednoho bodu na obrazovce. Znamená to tedy, že bajt je schopen si zapamatovat stavy osmi jednotlivých bodů obrazovky. Tak se nám osmkrát zmenší spotřeba paměti, takže teď vychází: 49152/8 = 6144 bajtů

Tak jsme si potvrdili předchozí výpočet velikosti paměti kresby. Díky tomuto rozčlenění na osmice je kromě bodů a bodových řádků obrazovka rozdělena na čtverce 8 x 8 bodů. Protože tyto čtverce přesně pokrývají plochu celé obrazovky, byly současně využity pro zobrazení písmen, přesněji vzato grafických znaků (neboť nejde jen o písmena).

Každý znak je orientován ve čtverci 8 x 8 bodů. Okrajové sloupce a řádky tohoto čtverce bývají až na výjimky prázdné a tvoří tak mezery mezi písmeny a řádky. Také tyto čtverce (znakové pozice) důvěrně známe, protože je používáme při zadávání souřadnic v příkazech AT nebo TAB.

Protože tyto čtverce udávají pozici znaku na obrazovce, budeme jim (i pro nedostatek vhodnějšího názvu) říkat znakové čtverce.

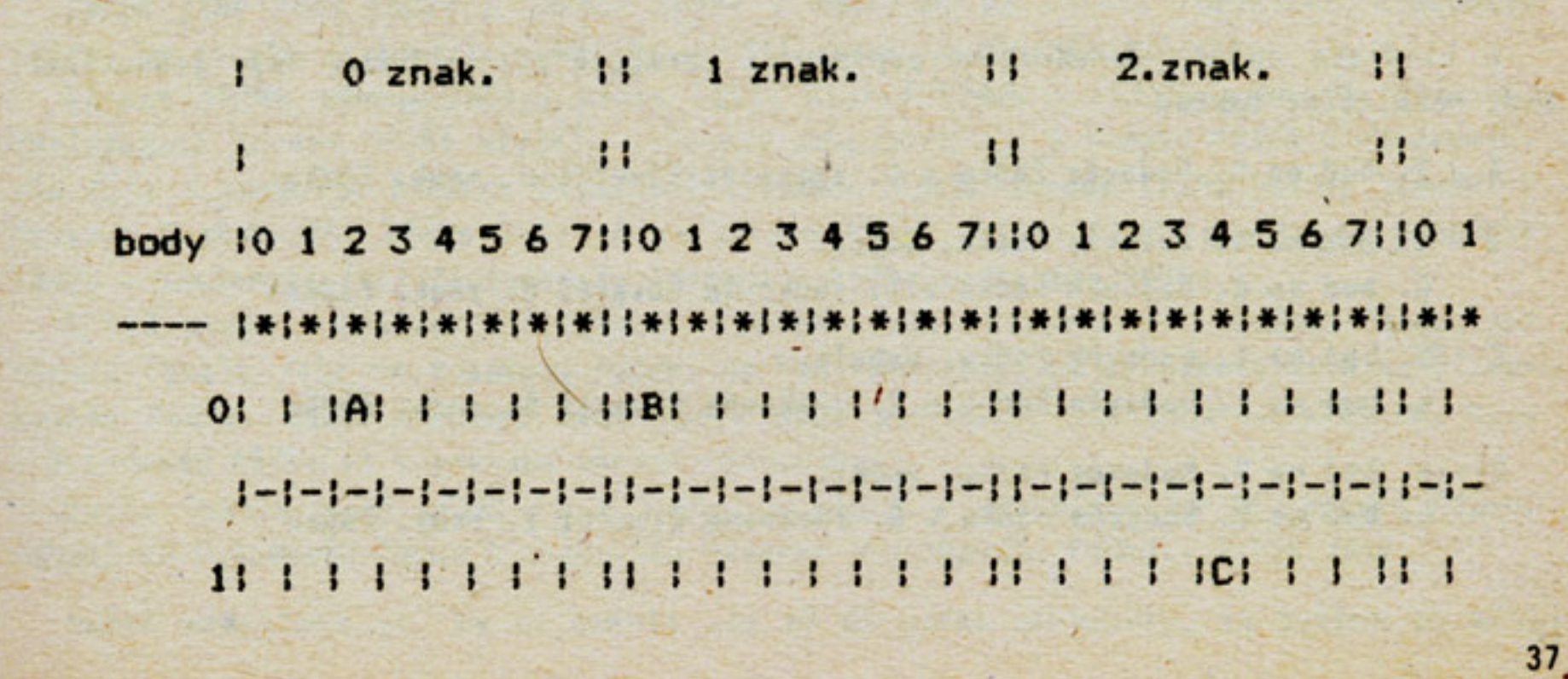
Obrazovka se tedy (kromě bodů) dělí ještě na znakové čtverce. Celá obrazová plocha obsahuje 24 znakových řádků a 32 znakových sloupců. Počet znakových pozic je tedy: 32 x 24 = 768,
přičemž poslední dva znakové řádky dole jsou již řádky editační zóny.

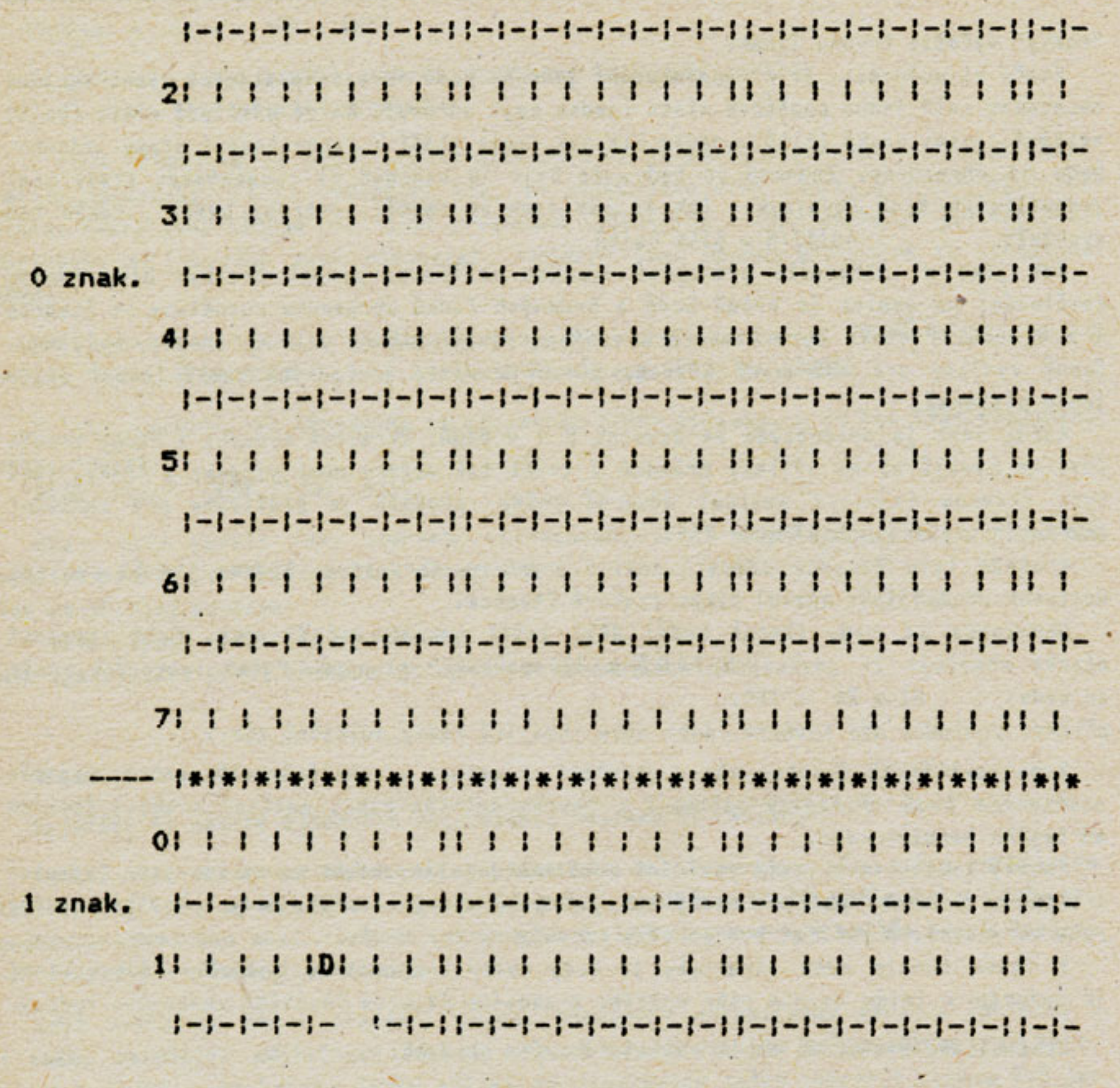
Konstruktéři ZX Spectra nám v obrazové orientaci vyrobili trochu zmatek, protože souřadnice bodů obrazovky počítají z levého dolního rohu, zatímco znakové čtverce od levého horního rohu.

Takový zmatek si v tomto povídání nemůžeme dovolit, takže my si zavedeme jednotné počítání. A budeme počítat hezky tak, jak jsme zvyklí i psát, tedy z levého horního rohu obrazovky. A počítat budeme vždy od nuly.

V levém horním rohu obrazovky je tedy nultý bod nultého bodového řádku, který je umístěn v levém horním rohu nultého znakového čtverce (nultého znakového řádku).

Schématicky bychom si mohli udělat takovýto obrázek:





Z hlediska našeho dohodnutého popisování obrazové plochy budou tedy jednotlivé body rozmístěny takto:

A - 2. bod na 0. bodovém řádku v 0. znakovém čtverci 0. znak. řádku

B - 8. bod na 0. bodovém řádku, neboli
0. bod na 0. bodovém řádku v 1. znakovém čtverci 0. znak. řádku

C -20. bod na 1. bodovém řádku, neboli
4. bod na 1. bodovém řádku v 2. znakovém čtverci 0. znak. řádku

D - 4. bod na 9. bodovém řádku, neboli
4. bod na 1. bodovém řádku v 0. znakovém čtverci 1. znak. řádku

Vidíme, že bod můžeme určit buď podle pouhých bodových souřadnic, nebo určíme bodové souřadnice ve znakovém čtverci, ale současně musíme ještě určit souřadnice tohoto čtverce.

Způsob zápisu kresby

Jak je ukládána kresba do paměti obrazovky? Není to složité. Prvních osm bodů na nultém bodovém řádku (body 0-7) jsou zapsány v první paměťové buňce, neboli lokaci, tedy na adrese 16384. Zápis odpovídá vždy binární formě dekadického čísla, které je v paměťové buňce zapsáno. Když zadáme povel: POKE 16384,1
zapíšeme tečku na sedmý bod na nultém bodovém řádku. Samozřejmě, že se v tuto chvíli zeptáte, proč na sedmý a proč ne na nultý, jak by se slušelo. Inu proto, že na osmici zaznamenaných bodů na této lokaci se musíme dívat jako na binární číslo.
Dekadická 1 vypadá v binární formě takto: 00000001
Nezapomeňme, že na jednu lokaci můžeme zapsat čísla 0-255, což je binárně právě těch potřebných osm bitů.

Zapíšeme-li tedy: POKE 16384,10
dostaneme na obrazovce černé tečky na 4. a 6. místě, protože 10 je binárně 00001010. Z toho vyplývá, že chceme-li zapsat tečku na 0. bodu 0. bodového řádku, musíme zadat povel: POKE 16384,128
protože 128 je binárně 10000000; atd., až konečně povel: POKE 16384,255
udělá v nultém znakovém čtverci nahoře souvislou čárku, čili zapíše všech osm bodů, protože 255 je binárně 11111111.

Těm, kterým dělá přepočet potíže, je vhodné připomenout, že SPECTRUM dovoluje zakreslovat přímo binárně a že tedy na příklad tečku na třetím místě (bod 2) lze zapsat těmito dvěma způsoby, které jsou rovnocenné.
POKE 16384,32 nebo POKE 16384,BIN 00100000
Ostatně - pokud jste obeznámeni s tvorbou vlastních USR znaků, vidíte, že organizace zápisů je naprosto shodná.

Místo zápisu kresby

Zatím jsme tedy pouze u lokace 16384 a umíme do ní zapsat libovolnou kombinaci teček. Do ostatních lokací se kresba ukládá naprosto stejným způsobem.
Adresa 16384 obhospodařuje pouze nultý bodový řádek v nultém znakovém čtverci. Následující adresa 16385 stejným způsobem obhospodařuje 0.bodový řádek v následujícím čili 1.znakovém čtverci 0.znakového řádku. Další adresa 16386 jde na 2.znakový čtverec 0.znakového řádku atd. až po adresu 16415, která tvoří paměť pro 0.bodový řádek v posledním čili 31.znakovém čtverci nultého znakového řádku.
A teď pozor - následující lokace - 16416 - nás nezavede na 1.bodový řádek, jak bychom čekali, ale na 8.bodový řádek, na body 0-7, čili na 0.znakový čtverec 1.znakového řádku. Další lokace, tedy 16417 atd. pokračují již zase ve stejném, tedy v 8.bodovém řádku až k pravému okraji obrazovky.
Další lokace zase přeskočí 7 bodových řádků a pokračuje na 16.bodovém řádku, neboli na 0.bodovém řádku 0.znakového čtverce 2.znakového řádku...a tak dále.

Ovšem nemyslete si, že to takhle jde až do konce obrazovky, to bychom to měli

příliš jednoduché. Obrazovka (nezapomeňte, že mluvíme pouze o činné ploše, bez okraje BORDER) je dále rozdělena na třetiny. Každá třetina obsahuje 8 znakových, tedy 64 bodových řádků.

Budeme-li postupovat po jednotlivých lokacích od adresy 16384 směrem nahoru, popíšeme nejprve 0.bodový řádek 0.znakového, pak 0.bodový řádek 1.znakového, tak se postupně dosataneme až na 0.bodový řádek 7.znakového a po něm doputujeme až na pravou stranu obrazovky. Protože na každém řádku je 32 znakových čtverců a řádků, prošli jsme jich tak 8 a dostali se až na adresu:

16384 + 32 x 8 = 16639

Tak jsme na konci nejhořejší, neboli nulté třetiny. Následující adresa bude patřit 1.bodovému řádku v 0.znakovém čtverci v 0.třetině a bude pokračovat zase dál po celé třetině přes všechny 1.bodové řádky. Potom se vrátí zase do 0.znakového řádku a bude kraslit do 2.bodového řádku a tak dál až postupně zaplní celou třetinu.

Bodových řádků v jednom znakovém je 8, znakových řádků také, a znakových čtverců v jednom řádku je 32, takže lehce spočítáme, kam až jsme se dostali:

16384 + 8 x 8 x 32 = 16384 + 2048 = 18431

Další třetiny obrazovky jsou organizovány naprosto stejně jako ta předchozí. Na adrese 18432 je 0.bodový řádek 8.znakového řádku, neboli 64.bodový řádek od začátku.

Takovýto popis vypadá složitě, ale my tu posloupnost vlastně velmi dobře známe z toho, jak se kreslí obrázek při nahrávání obrazovky. Na to jsme se dívali již mnohokrát, stačí si pořadí kresby jenom uvědomit.

Na konci druhé (poslední) třetiny se dostaneme do oblasti editační zóny a vidíme, že i zde můžeme kreslit. Nesmíme ovšem zapomenout na to, že editační řádky ihned zmizí, jakmile SPECTRUM potřebuje psát v editační zóně - třeba při INPUTU, nebo stavových hlášeních. Musíme s tím v programu počítat.

Poslední bajt paměti kresby má tedy adresu:

```
16384 + 3 x 8 x 32 x 8 = 16384 + 6144 = 22527
        ¦   ¦    ¦   ¦
        ¦   ¦    ¦   ¦- počet bodových řádků v
        ¦   ¦    ¦      jednom znakovém řádku
        ¦   ¦    ¦
        ¦   ¦    ¦- počet znakových čtverců v jednom
        ¦   ¦       znakovém řádku
        ¦   ¦
        ¦   ¦- počet znakových řádků v jedné třetině
```

```
!
!- počet třetin
```

Bajt následující adresu 22528 je první bajt atributů, ale o tom si povíme později.

Jiří Pobřísło

ON ERROR GO TO

V prvním čísle Zpravodaje jsme se setkali s jednou z možností, jak po tisku tlačítka BREAK převést řízení programu na předem určené místo. Posloužil k tomu i mód přerušení IM 2 mikroprocesoru Z80. V tomto článku si ukážeme, jak můžeme "vykolejit" Spectrum ze skoku do podprogramu jeho chybových hlášení - a to i bez použití módu přerušení. Jinými slovy - jak dosáhnout toho, aby všechny chyby (ale i BREAK), nebo jen některé - podle naší volby, nekončily vypsáním hlášení na obrazovce a zastavením chodu programu, ale aby program pokračoval nerušeně dál. Jistě si vzpomenete na nejeden program, kde vám takováto finesa znemožnila si jej prohlédnout nebo pozměnit.

Systémová proměnná ERR SP na adresách 23613 a 23614 obsahuje vektorovou adresu, na niž je převedeno programové řízení v případě potřeby vypsání chybového hlášení na obrazovce. Její hodnota je 4867 a je samozřejmě v ROMce. Její funkci si můžete vyzkoušet tak, že do systémové proměnné ERR NR (adr.23610) uložíte číslo požadovaného hlášení o 1 zvýšené (viz manuál Spectra) a zavoláte rutinu na adresa 4867 (např. povelem RANDOMIZE USR 4867). Hlášení se vám vypíše na obrazovce. Z uvedeného je zřejmé, že změnou vektoru ERR SP můžeme převést programový běh na jinou adresu, na níž můžeme mít vlastní rutinu s libovolnou funkcí. Mohli bychom ji využít i pro hlášení, která Spectrum vůbec nezná, ale to by nám nejspíš k ničemu nebylo (výjimky ovšem potvrzují pravidlo). Velmi se nám však může hodit při ladění delšího Basicového programu (počítače jako Amstrad aj. mají tuto funkci začleněnu do svého operačního systému).

Uvedená rutina ve strojovém kódu plní patřičnou funkci beze zbytku. Je relokovatelná, můžete ji umístit kamkoli do volné paměti. Relokovatelnost je zajištěna několika vstupními bajty programu. Voláme-li jakoukoli rutinu z Basicu, volaná adresa se ihned objeví v reg. BC mikroprocesoru. Přičteme-li k němu číslo (offset), které říká, o kolik bajtů dál začíná naše nová "chybová" rutina (TRAP), můžeme pak výsledek jednoduše uložit do proměnné ERR SP

a vše už bude dál pracovat samo. Teoreticky byste takových rutin mohli v programu mít několik. Vždy by stačilo jen změnit číslo offsetu a program by pokračoval od adresy, která je výsledkem součtu adresy volané a daného offsetu. Jiný účel, než je provedení nějakého žertéřského kousku, však stavba takových rutin zrovna nemá. Spíš se vyskytne potřeba, kdy budeme muset do ERR SP vložit, co tam původně bylo, a naši rutinu používat jen v případech, kdy pro to máme nějaké rozumné opodstatnění.

Abychom sami nepadli do jámy, kterou se snažíme nastražit, je z rutiny možnost úniku při hlášeních OK, Stop a Nonsense in Basic. Pokud však jejich kódy z rutiny vypustíte, nebude možno Basicový program zastavit jinak, než vypnutím přívodu proudu.

V rutině je využita možnost skoku do Basicového programu pomocí vložení čísla jeho řádky do systémové proměnné NEWPPC (zde je to řádka 9900) a pořadového čísla příkazu na dané řádce NSPPC (zde 1). To si ovšem můžete měnit dle libosti, stejně jako kódy hlášení, které byste chtěli mít jako výchozí pro vysmeknutí se z programu. "Fajnšmekři" mohou upravit program třeba i tak, že podle kódu hlášení bude uskutečněn odskok vždy na jinou řádku Basicu, či nějakou další rutinu strojového kódu.

Protože může být velmi užitečné vědět, kde došlo k aktivaci chybového hlášení, použil jsem pár adres "printer bufferu" k uložení této informace. Rutina bude samozřejmě fungovat i bez tohoto doplňku (adresy si můžete rovněž zvolit zcela jiné).

Před skokem do interpreteru BASICu musíme proměnné ERR NR vrátit číslo 255 (znamená žádná chyba), jinak by se program dostal do věčné smyčky. Do interpreteru vstupuji na adrese 7030, na niž je normálně převedeno řízení programu při jakémkoli chybovém hlášení. Volání rutiny můžete umístit kamkoli do programu. A nezapomeňte, že řádka, na kterou se z rutiny vracíte, musí v Basicu existovat.

```
;                    Výpis rutiny ON ERROR GO TO
;
211400     START   LD HL,OFFSET   ;Offset do HL
09                 ADD HL,BC      ;Přičti ofset k adr.STATR
ED583D5C           LD DE,(ERRSP)  ;Ulož do DE adr.chyb.rutiny ROM
EB                 EX DE,HL       ;a převeď ji do HL
73                 LD (HL),E      ;Vlož adresu rutiny TRAP do
23                 INC HL         ;syst.proměnné ERRSP, takže teď
72                 LD (HL),D      ;pájde chyb.řízení vždy na TRAP
212700             LD HL,TRAPA    ;Opět offset, tentokrát pro in-
09                 ADD HL,BC      ;strukci LD DE,nnnn na adr.PUSH
73                 LD (HL),E      ;Obsah reg.DE bude nnnn pro
23                 INC HL         ;uložení adr.TRAP do zásobníku
72                 LD (HL),D      ;instrukcí PUSH DE
C9                 RET            ;Offsety OK, zpět do Basicu
;
;rutina TRAP, na niž přechází řízení při aktivaci chyb.hlášení
;
210313     TRAP    LD HL,BTRAP    ;Původní ROM adresa pro ERR SP
3A3A5C             LD A,(ERRNR)   ;Kód chyb.hláš. do reg.A
3C                 INC A          ;Kód je vždy o 1 menší..zvýšit!
2808               JR Z,JUMP      ;Konec programu- kód 0 -návrat
FE09               CP 9           ;Je to kód 9 (Stop)?
2804               JR Z,JUMP      ;ano, tedy návrat
FE0C               CP 12          ;Je to Nonsence in Basic?
```

```
2001            JR NZ,PUSH      ;Není to žádný z nich, skok
E9      JUMP    JP (HL)         ;Kódy 0,9,12 - na adr.ERR SP
11EE5C  PUSH    LD DE,TRAP      ;Modifikováno 2.offsetem výše
D5              PUSH DE         ;Adr.TRAP do zásobníku
21425C          LD HL,NWPPC     ;Adr.syst.pr.čísla řádky
36AC            LD (HL),GTOLL   ;do NEWPPC naše číslo řádky...
23              INC HL          ;napřed nižší a pak
3626            LD (HL),GTOHL   ;vyšší bajt č.řádky 9900
23              INC HL          ;Do NSPPC č.příkazu na řádce
3601            LD (HL),GTOST   ;(příkaz 1.)
23              INC HL          ;Adr.syst.pr.PPC
11005B          LD DE,PRTBF     ;Adr.printer bufferu
010300          LD BC,3         ;Čítač instr.LDIR nastav na 3
EDB0            LDIR            ;Přenos PPC a SUBPPC do pr.buf.
3A3A5C          LD A,(ERRNR)    ;C.chyb.hlášení do reg.A,
3C              INC A           ;korekce čísla (o 1 zvýšit!),
12              LD (DE),A       ;číslo do printer bufferu
3EFF            LD A,255        ;Pro ERR NR "žádná chyba",
323A5C          LD (ERRNR),A    ;inicializace ERR NR
C3761B          JP MAINL        ;Skok do interpreteru Basicu
;
;definované bajty
;
0014    OFFSET  EQU  TRAP START ;ofset pro relokovatelnost
0027    TRAPA   EQU  PUSH START ;  "          "
1303    BTRAP   EQU  4867       ;původní adr.chyb.rutiny v ROM
5C42    NEWPPC  EQU  23618      ;syst.prom.pro č.řádky (GO TO)
5C3A    ERRNR   EQU  23610      ;  "    "   č.kódu hlášení (-1)
5B00    PRTBF   EQU  23296      ;1.adr.printer bufferu
0026    GTOHL   EQU  38         ;vyšší bajt č.řádky 9900
00AC    GTOLL   EQU  172        ;nižší   "     "    "
0001    GTOST   EQU  1          ;pořad.č.příkazu na řádce
1B76    MAINL   EQU  7030       ;adr.vstupu do interpreteru
5C3D    ERRSP   EQU  23613      ;syst.prom.adresy chyb.rutiny
;
;délka rutiny je 75 bajtů
```

ZX Computing 2-3/84
přeložil -elzet-

Konverze znaků textového souboru

Před pár lety pronikl mezi uživatele ZX Spectra slovní procesor TASWORD, který brzy získal značnou oblibu, tím i rozšíření. Netrvalo dlouho, a objevil se další program tohoto typu - SPECTRAL WRITER s několika praktickými přednostmi. Existují i další textové editory a slovní procesory zahraniční produkce. Žádný z nich však pochopitelně nezahrnuje písmena české abecedy s jejími dia-

kritickými znaménky. A tak nezbylo, než nějak si pomoci.

Skoro každý, kdo zařazení české abecedy ve svém slovním procesoru potřeboval mít, řešil věc obdobně - u Taswordu jsou písmenům s diakritickými znaménky většinou přiřazeny ASCII kódy 80H-8FH (grafické znaky ZX Spectra). To má však své nevýhody. Jednak je před napsáním takového písmenka nutno přejít do grafického módu a po jeho napsání opět přejít do módu normálního. Největší nevýhoda tohoto řešení však vyvstane v okamžiku, kdy text z takto upraveného Taswordu chceme vytisknout na tiskárně. Jsme připraveni o možnost průběžně měnit výsledný tisk pomocí grafických značek jakožto definovatelných řídících kódů tiskárny umisťovaných přímo do textového souboru. Lze nad tím sice mávnout rukou, ale máme-li tiskárnu, která má řadu tiskových funkcí, není příliš chytré jich nevyužívat.

Problematikou rozmístění písmen české či slovenské abecedy na klávesnici ZX Spectra se ještě budeme zabývat. V tomto článku si povšimnu řešení výše uvedeného problému. Tedy - co s textem, jehož některé znaky mají přiřazen jiný kód, než je tomu u našeho slovního procesoru? Po načtení takového souboru se obrazovka hemží všelijakými grafickými značkami, místo číslic se mohou objevit písmena apod. Pokud přímo nevíme, jak byl proveden původní převod, nevadí. Po krátkém detektivním pátrání brzy jeho zákonitost objevíme. Na papír si zapíšeme, jaká písmena, číslice či jiné symboly se skrývají pod "zmatenými" znaky na obrazovce. Potom zařadíme jejich kódy do konverzní tabulky (viz příklad níže) tak, že ke kódu, který má znak v našem slovním procesoru (je v páru vždy druhý) přiřadíme kód, který má tentýž znak ve zpracovávaném textovém souboru (v páru je první). Takto sestavené páry zapíšeme do paměti počítače od první adresy konverzní tabulky popř. obměníme jen ty, kterých se nějaká změna týká).

Níže připojený konverzní program ve strojovém kódu vlivem sestavy konverzní tabulky zpracovává znaky s kódy 80H-8FH (ostatních si "nevšímá"). Každý znak původního textového souboru, jehož kód je v uvedeném intervalu, je postupně porovnáván s kódy v konverzní tabulce. Když je kód nalezen, je v textovém souboru ihned nahrazen jeho "párovým sousedem" z konverzní tabulky. Konverze textu v délce 22 kB trvá přibližně 5-7 vteřin. Pochopitelně, že konverzní tabulka může obsahovat jakékoli kódy v jakémkoli pořadí. Dodržet musíme jen pravidlo o tom, že každý měněný kód původního textu bude v tabulce následován kódem, který jej bude nahrazovat. Jedná se tedy o prosté řazení párů, mezi nimiž je konverzní vztah.

Ve výpisu programu nejsou uvedeny adresy umístění instrukcí, protože program je plně relokatibilní (ve volné paměti jej můžete umístit kamkoli). Je to zajištěno nejen tím, že v něm nejsou žádné instrukce s absolutními adresami, ale i tím, že první adresa konverzní tabulky je vypočítána dvěma prvními instrukcemi programu - offset je v podstatě délka programu, která je přičtena (ADD HL,BC) k adrese volání rutiny z Basicu (tehdy reg. BC vždy obsahuje volací adresu). Jinými slovy - umístíme-li program od adresy 30100 (např.načtením z pásku příkazem LOAD "" CODE 30100) a zavoláme jej příkazem RANDOMIZE USR 30100, pak ihned po zavolání bude reg. BC obsahovat číslo 30100.

Jediné, co musíme v programu v případě potřeby nebo změn upravit, je první adresa textového souboru (zde 32768) a počet konverzních párů (zde 16). Pokud byste program jakkoli upravovali, nezapomeňte změnit i offset.

Vlastní funkce programu je zřejmá z komentáře. "Propust" kódů 80H-8FH instrukcemi porovnávání je umístěna pro urychlení práce programu. V případě, že měníme jiný číselný interval kódů, můžeme parametry porovnání změnit, resp. do propusti zařadit i jiné intervaly nebo jednotlivé kódy.

Délka vlastního programu (bez tabulky) je 57 bajtů, rozsah tabulky závisí na tom, kolik konverzních párů do ní zařadíme.

Pro provedení konverze je vhodné celý program umístit mezi konec Basicu a RAMTOP slovního procesoru, kde obvykle bývá ještě pár desítek volných bajtů. Pokud by se vám tam nevešel, můžete BASIC slovního procesoru o něco zkrátit některým z mnoha známých způsobů.

Tento (vlastně dekódovací) program můžeme použít i v opačném smyslu - pro zakódování jakéhokoli textu. Provedeme-li kombinovanou několikanásobnou konverzi všech (nebo skoro všech) znaků našeho textového souboru, těžko se kdy komu podaří jej rozluštit. Pokud ovšem ztratíme informaci o postupu jeho kódování, nerozluštíme jej už nikdy ani my sami.

Vyskytnou se i případy, kdy budeme muset překódovat svůj vlastní textový soubor. Dokud se nevžije nějaký standard kódování znaků s naší abecedou, můžeme si být jisti, že budeme stále narážet na větší či menší diference, které budou konverzi vyžadovat. Prakticky vzato - změnám kódů při zařazování písmen české nebo slovenské abecedy do souboru znaků textových editorů se nevyhneme, dokud nebudou běžně dostupné počítače s klávesnicí zahrnující písmena s diakritickými znaménky při splnění podmínky plně zajištěného softwarového zázemí.

Výpis programu konverze znaků textového souboru

```
        offset:EQU 57          ;Délka programu (bez tabulky)
        adrtex:EQU 32768       ;1.adr.text.souboru
        deltex:EQU 22528       ;Délka textového souboru Spectral
                               ;Writeru (u Taswordu je kratší)
213900  start: LD HL,offset    ;Do HL offset pro výpočet 1.adressy
09             ADD HL,BC       ;konverzní tabulky
E5             PUSH HL         ;Uložení 1.adr.konv.tab.do zásobníku
110080         LD DE,adrtex    ;Do DE 1.adr.textového souboru a
D5             PUSH DE         ;její uložení do zásobníku
21FFFF  LD HL,0 LD HL,FFFFH    ;Inicial.pro čítač počtu znaků text.
110058         LD DE,deltex    ;souboru (v DE je délka souboru)
B7             OR A            ;Nastavení CY na log.0
ED52           SBC HL,DE
EB             EX DE,HL        ;DE má teď funkci čítače počtu znaků
C1             POP BC          ;1.adr.text.souboru do BC
E1             POP HL          ;1.adr.konv.tabulky do HL

E5      další: PUSH HL         ;a její uložení do zásobníku
3E10           LD A,16         ;Do reg.A počet konverzních párů a
F5      porov: PUSH AF         ;uložení do zásobníku
0A             LD A,(BC)       ;Do reg.A kód znaku z text.souboru
FE80           CP 80H          ;Když je kód menší než 80H
3812           JR C,nic
FE90           CP 90H          ;nebo větší než 8FH,
300E           JR NC,nic       ;pak vždy skok na adr.nic-bez konv.
BE             CP (HL)         ;Hledání kódu v konv.tabulce
2808           JR Z,konv       ;Když nalezen, skok na adr.konv
```

```
23              INC HL          ;Nenalezen, zvýšení HL o 2
23              INC HL
F1              POP AF          ;Obnovení čítače počtu konv.párů a
3D              DEC A           ;jeho snížení o 1
20ED            JR NZ,porov     ;Když není nulový, skok na adr.porov
1804            JR test         ;Když je nulový, skok na adr.test

23      konv:   INC HL          ;Zvýšení adr.tab.o 1 (nový kód)
7E              LD A,(HL)       ;Nový kód přes reg.A
02              LD (BC),A       ;na adr.původního kódu v text.soub.

F1      nic:    POP AF          ;"Odhoz" 4 bajtů ze zásobníku pro
E1      test:   POP HL          ;uvolnění adresy event.návratu
03              INC BC          ;Zvýšení adr.text.souboru o 1
13              INC DE          ;Zvýšení čítače počtu znaků o 1
7A              LD A,D          ;Byly porovnány všechny znaky?
B3              OR E
C8              RET Z           ;Když ANO, návrat
18DC            JR další        ;Když NE, skok na adr.další

8030    DEFB:   80H,30H         ;Definované bajty převodní tabulky
8132    DEFB:   81H,32H         ;(konverzní páry - 1.bajt páru obsa-
8233    DEFB:   82H,33H         ;huje původní kód znaku, 2.bajt kód,
8334    DEFB:   83H,34H         ;kterým bude původní nahrazen)
8435    DEFB:   84H,35H
8537    DEFB:   85H,37H
8638    DEFB:   86H,38H
8739    DEFB:   87H,39H
882B    DEFB:   88H,2BH
893D    DEFB:   89H,3DH
8A31    DEFB:   8AH,31H
8B3C    DEFB:   8BH,3CH
8C2D    DEFB:   8CH,2DH
8D60    DEFB:   8DH,60H
8E20    DEFB:   8EH,20H
8F36    DEFB:   8FH,36H
```

- elzet-

BAJTEK - KOMPUTER - MIKROKLAN - IKS

tvoří čtveřici polských měsíčníků cele se věnujících malé výpočetní technice.

BAJTEK se na 32 stranách "mladosvětského" formátu zabývá malou výpočetní technikou. Mikropočítače, jímž Bajtek věnuje svou pozornost, patří mezi nejrozšířenější i u nás (ZX Spectrum, Amstrad). Pochopitelně se věnuje i domácím výrobkům (např. Meritum).

Jaký je obsah Bajtku? Vybrali jsme přehled článků z jeho letošního č.7: ZX

Spectrum - Prezentace, 64 znaků na řádce (program), Vlastní písmena (program), Spojení s tiskárnou DZM 180, Amstrad - programové finesy; Meritum - Odhalení tajemství, Nové funkce klávesnice; Co se hraje (informace o nových hrách, jejich recenze a rozbory herních situací, "pouky", čtenářský žebříček nejoblíbenejších); Trh (kde, co a za kolik); Netypické mikropočítače - New Brain; Kurs programování v jazyku LOGO (už 3. lekce); Listárna a další zajímavé materiály.

Protože Bajtek, Komputer a Mikroklan se zabývají mnohým z toho, co zajímá i nás všechny, již od tohoto Zpravodaje uvádíme jejich obsah v rubrice Mikrostránky. Budou v ní stručné informace i o obsahu jiných časopisů a-knih, které se nám podaří prolistovat. Chceme tak přispět vaší orientaci v literatuře, která by vám ve vaší zájmové i profesní činnosti mohla být nápomocná.

Abychom uspokojili zvědavost náruživých hráčů počítačových her, uvádíme žebříček Bajtku (od 1. do 10. místa): Knight Lore, Superchess, Booty, Underwurlde, Cyclone, Mugsy, Exploding Fist, Pyjamarama, Match Point, Jumping Jack.

Leccos zajímavého jsme vyčetli z reklamních stránek Bajtku. Např. ELKOR z Poznaně nabízí různé kazety, na každé pět původních polských programů. Hry jsou lacinější (890,- Pzl.), užitkové programy dražší (1390,-Pzl. za kazetu). Ve Varšavě působí zásilková půjčovna programů ENTER Computing. Jiné firmy nabízejí diskové jednotky, interfacy a joysticky pro ZX Spectrum (ceny neuvedeny). Volně v prodeji je celá řada mikropočítačů, ovšem za ceny dost vysoké (v tisících Pzl./: ZX Spectrum 122, ZX Sp. Plus 170, QL 380, C64 s mgf 250, C128 400, Atari 800XL s mgf 170, Atari 130XE 210, Schneider CPC464 se zel. mon. 370, CPC 6128 se z.m. 880. Někde mohou být ceny ještě vyšší. Např. BOMIS v Poznani nabízí CPC6128 se z. m. za 1,091 a s bar. m. za 1,42 milionu Pzl. K dostání jsou i malé "sejkoši" a dobrá periferní zařízení. Široká je nabídky softwaru, zabývá se jím řada malých firem po celém Polsku.

Bajtek vychází jako bratříček Sztandaru Mlodych (obdoba naší Mladé Fronty) a stojí 100 Pzl. O podobně zaměřenou sestřičku Mladé Fronty by byl určitě velký zájem i u nás.

Zbývající tři časopisy vycházejí ve formátu A4. **KOMPUTER** na svých 48 stranách podává zevrubnější informace o problematice světa mikropočítačů než je tomu u Bajtku. A to nejen po stránce soft-hardwarové, ale i v oborech, které nasazení počítačů jakkoli ovlivňuje. Je tedy vhodnější pro ty, kteří se chtějí dozvědět něco víc.

MIKROKLAN má mezi svými kolegy poněkud výjimečné postavení. Vznikl na základě spolupráce mezi polským vydavatelstvím měsíčníku Informatyka a rakouským vydavatelem NOT-Sigma, který jej pro Polsko tiskne. Vychází na 36 stranách křídového papíru s pěknou grafickou úpravou, v nákladu 100000 kusů. Vedle původních domácích materiálů je jich většina převzata ze západoněmeckého časopisu MICRO, jehož rakouskou mutaci vydává právě NOT-Sigma. Tím je zajištěno, že se v Mikroklanu budou objevovat i čerstvé, tedy aktuální překlady z tohoto měsíčníku. Přes to všechno je Komputer zatím na o něco vyšší úrovni než Mikroklan, jehož 1. číslo se objevilo teprve nedávno. Na jeho další vývoj si tedy ještě počkáme.

IKS jsou první písmena slov Informatyka, Komputery, Systemy. IKS je bohužel graficky na velmi slabé úrovni. Rovněž papír, na němž je jeho 32 stran tisknuto, nijak nepřidává na dojmu. Obsahově je zaměřen především na otiskování jednodušších basicových programů pro mikropočítače jako ZX Spectrum, C64, Amstrad, Sharp a polské

Meritum. Hlubší problematikou se zabývá jen okrajově. Oproti tomu však v něm vycházel např. seriál výuky jazyka LOGO. Je určen především mládeži ve věku kolem 15 let. Vznikl jako příloha Časopisu Zolnierz Wolnosci.

Hodnoceno celkově, nezbývá, než našim polským kolegům zdravě závidět. Nebereme-li v úvahu členské zpravodaje Mikrobáze, nevychází u nás zatím jediný časopis, který by se mikropočítačům plně věnoval. V roce 1987 se má objevit první, až příliš dlouho slibovaná Mikroelektronika. Jenže vzniká otázka - když zůstane je u ní, nebude to pro rozvoj tohoto širokého oboru přece jen trochu málo?

-elzet-

Mikrostránky

BAJTEK 8/86 (PLR)

LOGO (4.část) * Amstrad 6128 * Programy pro Amstrad * ZX Spectrum (interface Kempston pro joystick a programy) * ASCII a RADIX-50 * "Pouky" pro hry * Co se hraje * Mapa hry Roland´s Rat Race * Hry * Commodore 64 a kopírování * Databáze * Ceny hardwaru * Basic pro nejmenší

BAJTEK 9/86 (PLR)

Gemini 10X * Commodore 128 * Commodore a jeho paměť, programy * Hra pro Amstrad * Magnetofon a Amstrad * Co se hraje * Mapa hry Panama Joe * Atari a obrazovka, program * ZX Spectrum (kompilátor, stavba joysticku) * Laser 128 * Basic pro nejmenší * Ceny hardwaru * Homo Intelligens

KOMPUTER 6/86 (PLR)

Ochrana paměti před vloupáním * 80K RAM pro ZX Spectrum (návod) * Spectrum a magnetofon (1) * RAM 576K pro Amstrad * Disketové jednotky obecně a VC-1541 pro Commodore * Diskety s radiálním záznamem * Optické paměti * Mapa hry Nodes of Yesod * O hře Spellbound * Grafické zařízení PACO * Nové slovní procesory pro ZX Spectrum * Program písmena pro ZX Spectrum * IBM PC/AT * Propojování různých počítačů * RS232C * Ceny hardwaru

MIKROKLAN 1/86 (PLR)

Basic pro začátečníky * Mikro v úřadě * Diskety 3,5 palce * Počítačová síť (Jeden pro všechny) * Amstrad story * Programátor EPROMek (námět) * Operační systémy (1) - UNIX * 2048 portů pro ZX Spectrum * Programy pro ZX Spectrum * Ceny IO a počítačů

-elzet-

INDEX

O této nově připravované službě jsme vás informovali v minulém zpravodaji Mikrobáze. Bohužel se nám zatím oddálil sen o 16tibitovém počítači, jehož kvality jsou nezbytné pro zavedení této služby z hlediska potřeby rozsáhlé interní i externí paměti. Na jeho získání dále pracujeme, můžete pomoci i vy. Protože nechceme nikoho z vás uvádět do omylu, vracíme se k potřebným informacím pro rozběhnutí INDEXU ihned, jakmile bude počítač instalován. Nabídky programů nám však můžete zasílat průběžně.

Programová nabídka MIKROBÁZE

Jak jsme uvedli v minulém čísle zpravodaje Mikrobáze, budou v její programové nabídce uváděny pouze původní programy. Jiné programy, než v nabídce uvedené, neobjednávejte. K získání nebo výměně zahraničních nebo jinak specifických programů slouží svazarmovské kluby uživatelů jednotlivých počítačů. Tyto kluby se poměrně čile množí. Pokud ve vašem okolí žádný takový klub není, nic nestojí v cestě tomu, abyste jej založili třeba právě vy. Nápomocna v zaopatřování programů z různých oblastí a pro různé počítače vám bude i nově připravovaná členská služba INDEX, o níž jsme informovali v minulém zpravodaji. Nyní již k samotné programové nabídce:

ZX Spectrum

DrMG

Na bázi známé kombinace programů GENS3 a MONS3 postavená úprava, která umožňuje např. jednodušší spolupráci mezi oběma částmi programu, odpadají starosti se studenými a teplými starty, lze měnit začátek pracovní oblasti, při disassemblování se monitor neptá na adresu, kam překlad uložit, ale sám si vyhledá konec zdrojového textu generátoru, uloží překlad za něj a upraví příslušné parametry generátoru,

dále je přidáno tolik potřebné "pípání" tlačítek, průvodní texty jsou slovenské, přidaný modul provádí přepočty mezi různými číselnými soustavami atd. Původní funkce obou základních programů zůstávají zachovány. Doplněno dvěma svazky bohatého manuálu. DrMG je jedinou kompilací původního materiálu se zahraničním vzhledem k jeho určení pro průběžné doplňování znalostí assembleru Z80 v návaznosti na didaktickou činnost Mikrobáze směrovanou na její členskou základnu.

DIAPEN

Slovní procesor pro editaci textu v českém a slovenském jazyce. Název je složen ze dvou slov - PEN je dnes již mezinárodním označením mnoha druhů pisátek, DIA je zkratkou slova diakritický (rozlišovací), které ve spojení se znaménky označuje čárky, háčky, tečky, kroužky apod. u písmen mnoha abeced různých jazyků. DIAPEN počítá i s velmi jednoduchou úpravou pro editaci jakékoli abecedy mnoha dalších jazyků (od azbuky po norštinu). Na rozdíl od všelijakých úprav anglických editorů pro editaci diakritických znamének dosahuje DIAPEN zvláštní úpravou toho, že všechny původní znaky ASCII kódu zůstávají zachovány a vůbec se nemění jejich pozice na klávesnici. Výhoda tohoto řešení je m.j. v tom, že na DIAPENU napsaná např. česká slova se na jiném editoru promítnou nikoli jako změť nesrozumitelných znaků, ale budou u nich chybět jen dia-znaménka. Rovněž pro tisk českého textu z jiného editoru nebude třeba provádět žádné úpravy - text se vytiskne jen bez dia-znamének. Manuál DIAPENu bude obsahovat instruktáž provedení tisku písmen s těmito znaménky na tiskárnách, které mají buď grafický mód, nebo tzv. down-load do volné paměti tiskárny. Přímo na kazetě budou softwarové bloky pro práci s některými typy tiskáren a interfaců. DIAPEN bude obsahovat všechny funkce pro práci s textem, jak je tomu např. u Taswordu nebo Spectral Writeru, a některé funkce nové; ovládání tiskárny je rozšířeno. Obecně lze říci, že DIAPEN vyplňuje výraznou mezeru, která zeje v oblasti editace češtiny a slovenštiny z klávesnic zahraničních mikropočítačů. Doplněno bohatým manuálem.

DATALOG

Databanka, spojující rychlost programů ve strojovém kódu s variabilitou známého programu Master File. Tvůrce programu je veden snahou o co nejjednodušší řešení ovládání databanky (zvláště při tvorbě formy a formátu výpisů záznamů) při zachování vysoké variability struktury DATALOGU. Prohledávání v DATALOGU tedy nebude shodné s běžným hledáním v knižním katalogu, ale položka u jakéhokoli hesla se kdykoli bude moci stát heslem pro hledání dat daného klíče. Jednou z prakticky velmi cenných funkcí DATALOGU bude možnost provádění úprav a oprav jeho záznamů bez složitého "kurzorování", "šiftování", či přepisování opravovaných záznamů. Program umožní i snadný přepis dat na tiskárny všech typů, což mnoho ze známých databázových programů neumožňuje. DATALOG najde uplatnění při sestavování jakékoli informační databanky (od katalogu známek po seznamy členů organizací) s vysokou operativností při vyhledávání a aktualizaci dat. Zápis záznamů bude pochopitelně možný v češtině nebo slovenštině, rovněž abecední řazení proběhne podle dané abecedy. Doplněno opět bohatým manuálem.
Nově je programová nabídka rozšířena o následující dva programy:

µB-PASCAL

První původní mikropočítačový překladač jazyka PASCAL. Oproti známým verzím PASCALu fy HISOFT má µB-Pascal zavedeno několik nových funkcí. Jinak zcela odpovídá obecně užívanému jazykovému standardu. Program bude doprovázen obsáhlým manuálem a příklady. Nápomocí i podnětem pro začlenění PASCALu do rejstříku vašeho jazykového vybavení bude i v tomto čísle zahájený seriál rychlokursu PASCALu, bez nějž se při vážné práci s mikrcpočítačem nelze obejít hlavně (ale nejen) při tvorbě programů provádějících matematické výpočty všeho druhu - od elektroniky až po statistiku. Po skončení kursu se zpravodaj Mikrobáze bude PASCALu věnovat i nadále. Přednosti PASCALu vůči BASICu znalcům netřeba připcmínat, začínající pascalisté se o nich přesvědčí hned u svých prvních vlastních programů.

mikROMkód

Velmi žádaný, kompletní přehled rutin ROMky ZX Spectra 48K (tedy i ZX Spectra+ a ROMky, která je funkční v módu 48K u nových verzí ZX Spectra 128K). Kazeta bude obsahovat jednoduché rutiny, které budou voláním subrutin ROMky vykonávat požadované funkce i díky možnosti vkládání různých vstupních parametrů do hlavních rutin. Jedná se tedy o obdobu známého programu Supercode, ovšem s tím, že struktura rutin bude zcela odlišná, funkčně variabilnější a bude se plně soustředit na využití rutin paměti ROM. Celé dílo bude korunováno plným, komentovaným assemblerovým výpisem celých 16K ROMky. Pochopitelně nebude chybět ani popis hlavních programových rutin kazety. Oproti mnohým z vás známému originálnímu výpisu ROMky bude mikROMkód doplněn informacemi o možnostech práce s jejími stěžejními částmi (především kalkulátorem, rutinami obrazovky, klávesnice, ovládáním kanálů, zvuku, tisku, atd.). Program mikROMkód se svým velice rozsáhlým manuálem stane nezbytností každému, kdo bude chtít plně využít schopností svého počítače - především znalcům assembleru mikroprocesoru Z80. Ale i stoupenci jiných jazyků budou moci využít znalostí, které jim mikROMkód poskytne.

KAREL

Tento populární programovací jazyk vám Mikrobáze nabízí hned ve třech provedeních pro tři různé mikropočítače. Z toho verze pro ZX Spectrum je zcela novou modifikací programu. Programový manuál se zabývá nejen programovacími povely, jeho rozsah je významně rozšířen o celou metodiku programování s tímto typem jazyka. Ve své objednávce nezapomeňte uvést, pro jaký typ počítače program objednáváte. Dále si krátce připomeneme základní charakteristiku KARLA.

KAREL je mikropočítačový program, který je současně zábavnou hrou i seriózní učební pomůckou. Pochází ze Stanfordské univerzity, kde byl původně koncipován jako předstupeň výuky programovacího jazyka Pascal. V naší implementaci je do jisté míry setřena jednostranná orientace na Pascal a přidání některých nových prvků činí z

tohoto programu univerzální prostředek pro počáteční fáze výuky moderního programování.

Niklaus Wirth, autor programovacího jazyka Pascal, uvádí, že program - algoritmy + datové struktury. V systému KAREL jsou datové struktury potlačeny, což umožňuje snadnější chápání základních pojmů a postupů používaných při sestavování algoritmických struktur programů. Získané poznatky a dovednosti jsou pak využitelné ve většině ostatních programovacích jazyků.

KAREL má své opodstatnění i při přípravě k programování v jazyce Basic, který má nejširší uplatnění v oblasti mikropočítačů. Je holou skutečností, že Basic nemá dostatek prvků podporujících tvorbu strukturovaných a modulárních programů. Kdo však zvládnul KARLA, má i v jazyce Basic předpoklady k vytváření účinných, přehledných a dobře modifikovatelných programů. Mikropočítače a roboty sice směřují k takové dokonalosti, že je nebude třeba programovat speciálními programovacími jazyky, ale algoritmizace je dovednost využitelná obecně, nejen při programování počítačů.

V našem programu je Karel jméno robota, který je nakreslen na obrazovce mikropočítače. Karel má na obrazovce svoje město ohraničené zdí a v tomto městě vykonává funkci dopravní služby. Může se přesouvat z křižovatky na křižovatku a ukládat i sbírat na křižovatkách dopravní kužely, značky.

Program KAREL pro mikropočítač je dělán tak, aby sám dával návod k další činnosti obsluhy; lze s ním pracovat, aniž by uživatel potřeboval jakoukoliv příručku. V případech, kdy pro stručnost není jeho pokyn jednoznačný, lze správný postup nalézt metodou pokusů a omylů. Přesto je vhodné, či spíše nutné, doplnit práci s programováním Karla také vysvětlením základních pojmů strukturovaného a modulárního programování. K tomu slouží tištěný instrukční materiál.

Karlovi se dávají povely běžnými českými slovy. Pochopení nových poznatků proto není ztěžováno současným učením anglických slov. Karel je také úslužný robot, všechno, co si může domyslet, udělá nebo napíše sám. S formální stránkou svého učení v nejvyšší míře sám napomáhá. A tak se stává z trpělivého žáka ještě trpělivějším učitelem. (A pro zkušené programátory se z učitele stává zábavný společník!)

● ● ●

Další informace se týká aktuálního vývoje ve tvorbě programů, tedy i předpokládaného termínu jejich distribuce:

Dr.MG - autoři jej již odevzdali včetně manuálu, který bude předán do tisku v lednu. Reálný předpoklad zahájení distribuce spadá do února t.r.

µB-Pascal - autoři jej v prosinci odevzdali k redačnímu posouzení. Po některých drobných vylepšeních a odevzdání manuálu se dá předpokládat zahájení distribuce na přelomu února a března t.r.

DATALOG a DIAPEN - autoři oba programy průběžně cizelují se snahou o prezentaci vysoké profesionální úrovně. Vedle µB-PASCALU se jedná o nejkomplikovanější programy, u nichž je nutno počítat se zahájením distribuce pravděpodobně v dubnu t.r.

mikROMkód - překlad ROMky je již hotov, následuje tvorba série řídících rutin a rozšíření manuálu o jejich popis. Autoři předpokládají zakončení práce na dobu májovou. Protože manuál si svým rozsahem vyžádá velmi náročnou redakční práci i delší dobu tisku, distribuce je předvídána na konec června.

KAREL - nový manuál bude dán do tisku v únoru, na jehož konci bude zahájena distribuce.

Pokyny k objednávání programů

Samozřejmě, v platnosti zůstávají "pravidla hry" zveřejněná jak v Amatérském radiu (naposled v č. 5. ročníku 1985), tak v tiskovině s organizačními pokyny, kterou dostal každý, kdo projevil korespondenčním lístkem zájem o členství v Mikrobázi. Pro jistotu otiskujeme vzory vyplnění líce i rubu korespondenčního lístku k objednání programu z naší nabídky znovu.

Odesílatel:

Ing. Jan Novák
Jablonecká 56
Liberec

4 6 0 0 1

MIKROBÁZE
520214/0134
(rodné číslo)

Vyhrazeno pro služební nálepky a údaje pošty

602. ZO Svazarmu
Wintrova 8
Praha 6

1 6 0 4 1

Rub lístku budete vyplňovat v řádcích 1, 5 a 15 (viz vzor A). Do řádku 1 napíšete OBJEDNÁVKA PROGRAMU, do řádku 5 označení programu (bloku programů) podle naší nabídky.

POZOR! Programy ještě nemají přesná katalogová označení, v nichž budou v budoucnu zakódovány typy počítačů. Proto zatím programy popisujte v této formě:

(řádek č. 5) 5. typ počítače číslo a název programu (bloku) dle nabídky

Rub korespondenčního lístku pro objednávku programu:

1. OBJEDNÁVKA PROGRAMU
2.
3.
4.
5. Sord M5 7. Dissassembler
6.
7.
8.
9.
10.
11.
12.
13.
14.
15. Novák 8/12/1985

Za typ počítače budete proto podle svých potřeb vypisovat Spectrum, IQ 151, SORD M5, PMD 85 nebo ZX 81, číslo a název programu opište z nabídky přesně. Je pochopitelné, že programy jsou mezi různými typy počítačů nepřenositelné. Nemůžete proto například požadovat program 2. PASCAL pro počítač IQ 151, ale jen pro Spectrum, tak jak je nabízen, jakkoli byste ho pro IQ 151 potřebovali a chtěli. (Na to jsou jiná "pravidla hry" - viz organizační pokyny expedované dříve.) Nu, a nezapomeňte v řádku 15 na podpis a datum. Jinak bude objednávka neplatná.

Slovo k náhodným čtenářům

Dostal se vám tento zpravodaj Mikrobáze do rukou a zaujala vás aktivita rozvíjená v programových a technických službách pro uživatele osobních mikropočítačů? Redakce časopisu Amatérské radio jako iniciátor Mikrobáze a 602. ZO Svazarmu v Praze 6 jako realizátor vás zvou k členství v několikatisícovém kolektivu zájemců o výpočetní techniku a její aplikace. Jako člen Mikrobáze Svazarmu budete dostávat tyto zpravodaje (od roku 1986 čtyřikrát ročně), budete moci využívat programových nabídek a dalších plánovaných služeb.

Mikrobáze je službou pro mikropočítačovou techniku a ve své organizaci tuto techniku každodenně účelně a efektivně využívá. Proto je třeba i v případě, kdy teprve projevujete zájem o členství, dodržet určitou administrativní konvenci. O bližší informace o celém komplexu Mikrobáze a přihlašovací materiály je třeba požádat výhradně korespondenčním lístkem. Jeho líc vyplňte (zásadně strojem) podle vzoru č. 1. na straně 80. Rub korespondenčního lístku musí obsahovat v horní části čtyři číslované řádky s obsahem podle příkladu ve vzoru č. 2 (1. PŘIHLÁŠKA UŽIVATELE, 2. Jméno a příjmení, 3. Ulice, číslo, obec, poštovní směrovací číslo, okres, 4. Povolání/podnik, popřípadě škola). V dolní části lístku se podepište a uveďte datum vyplnění. Pak už stačí jen vhodit lístek do poštovní schránky a čekat na podrobné informační a přihlašovací materiály Mikrobáze. Dostanete je obratem.

1. PŘIHLÁŠKA UŽIVATELE
2. Ing. Jan Novák
3. Jablonecká 56, Liberec,460 01, Liberec
4. Programátor analytik /Textilana

31. 12. 1985 Novák

UPOZORNĚNÍ:

Znovu připomínáme, že Mikrobáze je služba jednotlivcům, členům Svazarmu. Nemůžeme proto vyřizovat žádosti organizací, škol a podniků.

OPRAVY TISKOVÝCH CHYB VE ZPRAVODAJI MIKROBÁZE č. 3

Prosíme vás, abyste si v minulém čísle zpravodaje opravili tyto chyby:

V článku Komprese a expanze obrazových dat ve výpisu programu na str. 23 má být na adr. FE18 správně 3222FE (nikoli 32FE18). Místo chybně zapsané adresy FE18 má být správně FE1E.

V tomtéž článku na str. 22, v jejím pravém sloupci, 11. řádce zdola chybí mezi uvozovkami znak * (správně má tedy být "*").

Za provedení oprav děkuje redakce.