MIKROBÁZE

07
- PASCAL (3)
- paralelní přenos dat (2)
- paralelní interfejs (2)
- Recenze : μB - PASCAL
  DATALOG

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

μB
AR-602

OBSAH

Mikrobáze, technický zpravodaj pro mikroelektroniku a výpočetní techniku. Vydává 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 ÚVTEI pod evidenčním číslem 87 007. Náklad 4000 výtisků.

Praha, červen 1987

K aktuálním úkolům masového rozvoje zájmové branné činnosti

7. zasedání ÚV Svazarmu, které se konalo v listopadu 1986, věnovalo svoji stěžejní pozornost aktuálním úkolům masového rozvoje zájmové branné činnosti ve Svazarmu. Charakterizovalo masovou zájmovou brannou činnost jako důležitý prostředek branně výchovného působení a konstatovalo, že v obsahu zájmové branné činnosti má rozhodující místo jednota politického a odborného působení. Ve všech sférách činnosti vychází organizace Svazarmu ze zákonitosti, že základem branné výchovy je morálně politický faktor, což plně platí i o zájmové branné činnosti, která patří k nejširší a nejpřitažlivější oblasti života a činnosti celé organizace.

Dále 7. zasedání ÚV Svazarmu konstatovalo, že je důležité zajišťovat realizaci zájmů jednotlivců a skupin v jednotě s celospolečenskými potřebami. Upozornilo na skutečnost, že zájmovou brannou činností prolíná polytechnická výchova, využívání progresívní techniky a technická propaganda. Ve všech činnostech je tak spojována odborná činnost s výchovou k tvůrčí práci. Zasedání dále upozornilo, že je doposud jen méně využíváno základní branné přípravy v odbornostech elektroniky, radioamatérství, ale i motorismu, a vyzdvihlo stále se zvyšující zájem zejména o odbornost elektronika. Za mimořádně důležité pokládá požadavek na vytvoření účelného systému doškolování nezbytného počtu výchovných kádrů, jejich spravedlivého oceňování z hlediska potřeb masového rozvoje činnosti i řešení málo uspokojivého materiálně technického a ekonomického zabezpečení činnosti technických odborností. Je také žádoucí rozšířit počty občanů a zejména mládeže v odborných klubech ZO Svazarmu. Ve všech místech, ústředními orgány Svazarmu počínaje, je také v tomto duchu třeba vytvářet příznivější podmínky k dalšímu masovému rozvoji zájmové branné činnosti.

Vysoká účelnost, hospodárnost, aktivnější hledání a využívání vlastních zdrojů bude i nadále naléhavým požadavkem na všech místech naší organizace. Navíc bude třeba dbát o větší preferenci podmínek pro masový rozvoj branně technických a branně sportovních činností a zejména pak dosáhnout účelnější koordinace možností se školami a všemi partnery jednotného systému branné výchovy obyvatelstva ČSSR.

7. zasedání ÚV Svazarmu tedy řešilo vpravdě strategické úkoly zájmové branné činnosti, které odpovídají vážnosti poslání zájmové branné činnosti v branné výchově.

Kopisté versus Mikrobáze

Je konec ledna 1987. Přede mnou leží sloupečky korespondenčních lístků s objednávkami jednotlivých titulů z programové nabídky Mikrobáze. Sečítám - Dr. MG 92 kusů, Datalog a Diapen po 80, Karel 32. Kolik že činí odhad počtu Specter v ČSSR? Prý něco kolem sta tisíc. Čísla, která nutně vedou k zamyšlení. Vůbec teď nemyslím

na všechny ideové a rétorické souboje, které předcházely přechodu Mikrobáze na platformu původnosti, či na kvantum práce, která musela být vykonána pro to, aby se dokázalo, že vybojované má právo na existenci, že je opodstatněné, smysluplné a také realizovatelné. Nakonec o myšlence samotné dnes už žádný rozumný člověk nepochybuje.

Sloupečky objednávek vedou mé myšlenky jinam. Tam, kde leží zakopaný pes. Stalo se jistým "zvykovým právem", že programy se vyměňují. Samozřejmě zdarma. Mám-li deset nových her, které jsem s někým vyměnil za jiné, aniž mne to cokoli stálo, proč bych je nenatočil někomu jinému za dalších deset kousků, které ještě nemám. A tak dále, pořád dokola. Divoké výměny hromad softwaru se u nás zabydlely. Taky jak jinak se k němu dostat, když program zatím nikde koupit nelze?

Při tom všem se však vytratilo, vlastně ani nevzniklo povědomí o tom, že za každým programem je lidská práce. Nejen programátora, ale celého produkčního týmu, který se neobejde nejen bez předpokládaných schopností, ale ani bez nijak laciného přístrojového a materiálního vybavení. Jistě - jsou programy, za které by člověk nedal ani zlámanou grešli. Ale jsou i takové, bez nichž by se mnozí z nás dnes už neobešli. A právě takové bude Mikrobáze chtít nabízet i v budoucnu.

Hledím na ty sloupečky a přímo cítím, jak je v nich zakódováno poselství uvedeného zvykového práva. Jak už si ti, kteří ve sloupečcích chybějí, brousí magnetofonové hlavy na řetězové kopírování programů Mikrobáze. Skoro je slyším: "Ale co, ono to k nám v proudu ostatních programů časem přiteče taky, tak na co se vyplňovat s objednávkami a ještě za to platit?"

Zvyk je železná košile. Mikrobáze však od září loňského roku v žádné obrněné košili nevězí, ale "nese svou kůži na trh" se vším všudy. I s tím, že pokud nebude ekonomicky rentabilní, zavře se nad její programovou nabídkou voda.

Přemítám o tom, jak tomu předejít. Kdyby člověk nabízel rohlíky, tam by problém nebyl. Rohlík není na koukání, ale k snědku. Jinak ztvrdne a není k ničemu. Specifika softwaru je v tomto ohledu (bohužel?) jiná. Takže jak? Zatajovat programy na nejvyšší míru? U některých to samozřejmě bude možné. Ale ne u všech - přece jsme dali slovo, že programy budou uživatelům přístupné pro jejich vlastní úpravy. A jak znám tuzemské dešifrátory, žádné utajení před nimi nakonec stejně neobstojí. A stačí jeden, jediný prostup programu do výměnných sítí, a za pár týdnů má Mikrobáze o pár tisíc bezúplatných fanoušků víc.

Zavírám oči a v hlavě se mi odvíjí děj bizarního filmu o likvidaci všeho softwaru. Jednoho dne ve všech novinách na světě vychází inzerát, v němž někdo uvádí, že jednotlivé programy bude ode všech firem kupovat jen on sám a komukoli je pak poskytne zdarma. Firmy, jedna za druhou, se ještě ten den odpoledne pokládají.

Přepínám na jiný kanál. Do svého klubu výpočetní techniky přichází nadšený odběratel nového programu Mikrobáze a ihned jím napouští lokální počítač. Všichni kolem se zájmem sledují a podvědomě sahají po svých kazetách, jež mají zavěšeny proklatě nízko. Skupina obdivovatelů se postupně mění ve skupinu nátlakovou - my jsme ti dali tamto a tohle a ty nám teď ten program nechceš dát? Buď majitel programu podlehne, nebo opustí místnost za zvuků tříštícího se skla okenních tabulek.

Vypínám vnitřní studio a vracím se do skutečnosti. Sloupečky se krčí na svém místě. Ano, už středověký pragmatik Machiavelli upozorňoval vladaře na nutnost po-

čítat se zvyky národů. Takže co? Vzdát to? Ani omylem! Ale kde najít řešení? Suma sumárum jsou vlastně jen dvě:

1) **Široká** propagace, která zajistí, aby odběr programů dosáhl alespoň hranice rentability jejich produkce.

2) **Apelace** na členy Mikrobáze, aby si uvědomili, že jedině přímým odběrem programů (nikoli "oklikou") mohou zajistit kontinuitu její programové nabídky. Nikdo nikomu nenutí nic, co by nechtěl, či snad dokonce nemohl používat. Základní struktura programové nabídky byla sestavena tak, aby zasáhla oblasti nejširších aplikací. Nelze se ovšem domnívat, že programy budou vznikat i tehdy, nebudou-li investice návratné. Ale nejde jen o návratnost samu. Jde i o to, aby členská úhrada za jednotlivé programy byla co možná nejnižší. A to mohou zajistit opět jen členové Mikrobáze sami. Kvalita programů je plně zajištěna, není třeba se obávat o její výši - v mnohém jsou tyto programy na vyšším stupni kvality, než jaká byla dosud na mikropočítačích prezentována domněle špičkovými zahraničními produkty. A také -- nejsme snad Češi a Slováci? Budeme nadale vsecko cist a psat bez hacku a carek?

Rovněž je nutno vzít v úvahu, že bez objemného manuálu si s užitkovými programy nikdo příliš neužije. A manuál je tou součástí softwaru, která je u nás plně chráněna před neoprávněným užitím. Tuto oblast bude Mikrobáze pochopitelně sledovat a jen nerada sáhne k prostředkům, které by při porušení zákona souvisejícího musela použít. Ale radši se nestrašme, o to nám přece nejde.

Vydrží-li Mikrobáze svůj první souboj s kopisty, je velmi pravděpodobné, že časem zapříčiní vznik nového zvykového práva uživatelů softwaru. Charakterizovat by jej bylo možno asi takto: "Tenhle (perfektní) program mám od Mikrobáze. Dal jsem za něj tolik a tolik. Za stejný obnos jej obdržíte tamtéž. Lituji, ale na výměnu mám vyhrazeny jiné programy." Průkopníci nového zvykového práva budou nechápajícími jedinci tu a tam označeni za sysly, lakomce, hamouny a škrty. Takovému klidně odtušte: "Abyste si o mně nemyslel nic zlého, počkejte tu moment, já vám skočím koupit svetr za dvě stovky, tak to vyjde skoro nastejno. Máte radši bílý nebo červený proužek na rukávu?" Slušný člověk pochopí aspoň tohle. Pokud však neodbytný jedinec odpoví, že radši bílý proužek, snadno si pak už poradíte sami.

Nechtěl jsem v tomto fejetonu nijak mentorovat. Vyšel jsem jen z předpokladu, že jako členové Mikrobáze zřejmě nemáte žádný zájem na tom, aby naše společná (a první toho druhu vůbec) programová nabídka "šla na buben". V podstatě jde jen o to, aby naše programy nevyužívalo padesát tisíc lidí, a sama Mikrobáze byla nucena pro "nezájem" své softwarové zřídlo zrušit. Dosáhnout této zkázy můžete velmi snadno. Stejně tak tomu můžete zabránit. Vše máte ve svých rukou jen vy sami. A naší povinností bylo na tuto okolnost upozornit. Abychom jednou nestáli nad rozlitým mlékem, nad nímž bývá pozdě brečet.

-elzet-

PASCAL

3.část

Zatím jsme se seznámili se čtyřmi jednoduchými typy: BOOLEAN, INTEGER, CHAR, REAL. Dále probereme strukturální typy: ARRAY (řetězec), RECORD (záznam) a SET OF (bázový typ).

ARRAY a RECORD jsou typy pole (podrobněji viz Programování v jazyku Pascal, SNTL). Pomocí řetězců můžeme deklarovat a využívat pole o libovolném počtu rozměrů. Např. jednorozměrnou proměnnou Tabule o deseti elementech můžeme deklarovat takto:

```
var Tabule : array [1..10] of integer;
```

Obecně platí:

typ-pole = "array" "[" typ-indexu {","typ-indexu} "]" "of" typ-složky

typ-indexu = ordinální-typ
typ složky = označení-typu

Obdobně deklarujeme pole o více rozměrech:

```
var Šachovnice : array [1..8, 1..8] of integer;

StranaTextu : array [0..31, 0..63] of char;

RubikKostka: array [-1..1, -1..1, -1..1] of boolean;
```

Všiměte si, že můžeme volit libovolný typ elementů pole, které můžeme strukturovat ještě hlouběji a vytvářet tak pole multidimenzionální.

```
var Iks : array [1..100] of array [1..3] of boolean;
```

Pro identifikaci elementu pole mohou být typem-indexu i znaky:

```
var Šachovnice : array ['A'..'H', 1..8] of integer;
```

Dejme tomu, že chceme přidělit všem elementům pole Šachovnice hodnotu 0. Provedeme to následovně:

```
  for Znak := 'A' to 'Z' do

    for Počet := 1 to 8 do

      Šachovnice [Znak, Počet] := 0;
```

Proměnná znak je typu CHAR, Počet typu INTEGER. Tento příklad ukazuje, jakým způsobem se v programu odvoláváme na určitý element pole.

Pole, jejichž elementy jsou znaky, umožňují pracovat s nimi i jako s proměnnými. Do této chvíle jsme uměli přidělit proměnné hodnotu výlučně jednoho znaku. Srovnejte:

```
const DélkaSlova = 12;
var Slovo : array [1.. DélkaSlova] of char;
```

Následující program demonstruje probranou látku:

```
program Odzadu;

{ Program cte znaky slova z klavesnice
a pak slovo vypisuje odzadu }

const DelkaSlova = 20;

var Vyraz    : array [1..DelkaSlova] of char;
    Index,k : integer;
    Znak     : char;

begin
{ Vlozeni slova }
  writeln ('Zapis slovo (jen velka pismena): ');
  Index :=1;
  read (Znak);
  while (Znak >= 'A')
        and (Znak <= 'Z')
            and (Index <= DelkaSlova) do
    begin
      Vyraz [Index] :=Znak;
      Index :=Index+1;
      read (Znak)
    end;
{ Psani odzadu }
  writeln;
  for k :=Index downto 1 do
    write (Vyraz [k]);
  writeln;
end.
```

Starší verze Pascalu nedovolují užití řetězce znaků s proměnnou délkou. V posledním programu je nutno "přebývající" elementy za vepsaným slovem vyplnit mezerami. Novější verze tento nedostatek řeší nově standardizovaným typem STRING. Pro osvojení práce s ním je nutno nahlédnout do manuálu vašeho Pascalu.

Další program ukazuje využití znakových polí pro šifrování textu. Klíčem šifry je řetězec 26 znaků.

```
program Sifra;

{ Program sifruje text podle zadaneho
  klice obsahujiciho retezec 26 znaku }

var Klic     : array ['A'..'Z'] of char;
    Text     : array [1..40] of char;
    Index,k : integer;
    Znak     : char;

begin
{ Vlozeni klice sifry }
  writeln ('Vloz klic 26 znaku ');
  for Znak :='A' to 'Z' do
    read (Klic [Znak]);
{ Ulozeni jedne radky textu }
  Index :=1
  while (Znak<>chr (13)) and (Index <=40) do
    begin
      read (Text [Index]);
      Index :=Index+1
    end;
{ Vypis zasifrovaneho textu }
  writeln;
  for k :=1 to Index-1 do
    if Text [k] = ' ' then
      write (' ')
    else
      write (Klic [Text [k]]);
end.
```

V tomto programu musejí být všechny elementy pole stejného typu. To je často nevýhodné. Např. chceme-li vytvořit kartotéku se jmény a telefonními čísly (jednotlivá pole jednoho záznamu budou tedy obsahovat různé typy), musíme vytvořit tomu odpovídající strukturu. Taková struktura se jmenuje RECORD, česky záznam. Skládá se z určitého množství polí, identifikovaných nikoli indexem (jako u řetězců), ale jménem. Každé pole má definovaný určitý typ. Obecně:

typ-záznam = "record" seznam-položek "end".

seznam-položek = [(pevná-část[";" var.-část] | var.-část)[";"]].

pevná-část = sekce-záznamu {";"sekce-záznamu} .

sekce-záznamu = seznam-identifikátorů ":" označení-typu.

variantní-část = "case" selektor-varianty "of" varianta{ ";" varianta} .

selektor-varianty = [rozlišovací-položka ":"] rozlišovací-typ.

rozlišovací-položka = identifikátor.

varianta = seznam-rozliš.-konstant ":""("seznam-položek")".

rozlišovací-typ = identifikátor-ordinálního-typu.

seznam-rozliš.-konstant = rozliš. konst.{ "," rozliš. konstanta} .

rozlišovací-konstanta = konstanta.

Každá strana naší plánované kartotéky se jmény a tel.čísly může být definována takto:

```
var Strana : record
               Jméno : array [1..12] of char;
               Příjmení : array  [1..12]  of char;
               Telefon  : 0.. 999999
             end;
```

Po slovu RECORD následují názvy polí, resp. položek záznamu spolu s jejich typy. Definice končí slovem END.

Na jednotlivá pole záznamu se můžeme odvolat např. takto:

```
Strana.Telefon :=211205;
Strana.Jméno [1]:='B';
for k :=1 to 12 do
   read (Strana.Příjmení [k]);
```

Anebo:

```
with Strana do

   begin
     Telefon :=211205;
     Jméno [1] := 'B';
     for  k :=1 to 12 do
     read (Příjmení [k])
end;
```

Zde užitá instrukce WITH může být interpretována jako "se záznamem Strana vykonej následující:". Při užití WITH není již nutné uvádět název záznamu při odvolávání se na jeho pole.

Naši kartotéku, obsahující třeba 100 stran, můžeme definovat jako matici záznamů:

```
var Kartotéka : array [1..100] of record;
                   Jméno : array [1..12] of char;
                   Příjmení : array [1..12] of char;
                   Telefon :  0..999999
                 end;
```

Odvolávka např. na telefon na straně 1 bude nyní vyžadovat delší zápis:

```
Kartotéka [1] . Telefon :=211205;
```

Anebo:

```
with Kartotéka [1] do
  Telefon :=211205;
```

RECORDy jsou velmi užitečné při tvorbě databázových programů v Pascalu. Problematika věci pochopitelně přesahuje rámec tohoto "rychlokursu", proto nelze, než odkázat na odbornou literaturu.

Dalším typem, na který zaměříme pozornost, je množina (resp. datové objekty typu množina). Jde o celou třídu konkrétních typů. V každém konkrétním typu množiny je pevně stanoven typ prvků množiny. Popis typu množiny má tvar:

```
set of bázový typ
```

Set je česky sada, soubor. Bázovým typem mohou být např. jednotlivé prvky (barevné odstíny) množiny typu Barva; schématicky:

```
type Barva = (Červená, Žlutá, Modrá);
     Odstín=  set of Barva;
```

Deklarace proměnné Alfa bude např. takováto:

```
var Alfa : set of integer;
```

Bázové elementy musejí být prostého typu (nestrukturovaného). Omezen bývá rovněž počet elementů. Při práci s bázovým typem můžeme provádět operace s množinami.

Např.:

```
Alfa := [];
```

Proměnná Alfa zde reprezentuje prázdnou množinu.

```
Alfa :=Alfa+[0, 2, 3, 4];
```

V tomto případě proměnná Alfa nabývá hodnoty sjednocení množin Alfa a [1, 2, 3, 4]. Znaménko + zde tedy nemá běžný algebraický význam. Dále * symbolizuje průnik množin, - rozdíl, = rovnost, <> nerovnost, < = je obsažena v, > = obsahuje.

Operátorem IN můžeme zjišťovat, zda daný element je prvkem množiny. Např.:

```
if X in Alfa then
  writeln ('X je prvkem množiny Alfa ')
else
  writeln ('X není prvkem množiny Alfa');
```

Je-li splněna uvedená podmínka (true), X je různé od nuly. A naopak.

V dalším si ukážeme, jak můžeme zjednodušit deklaraci typů dat. Porovnejte si naši předchozí definici kartotéky s touto:

```
const DélkaVýrazu = 12;
      PočetStran  = 100;
type Výraz = array [1..DélkaVýrazu] of char;
     Strana = record
                Jméno :  Výraz;
                Příjmení :  Výraz;
                Telefon  :  0..999999;
              end;
var Kartotéka : array [1..PočetStran] of Strana;
```

Obě definice činí v podstatě totéž. Na první pohled se může zdát, že ta poslední je zbytečně dlouhá. Všimněte si však, že nám tu odpadly definice typu ARRAY přímo u znakových polí. Byly nahrazeny návaznými definicemi konstanty DélkaVýrazu a řetězcového typu Výraz. Jistě si umíte představit, že při větším počtu polí v záznamu nám tento způsob definice značně ulehčí práci.

Podobně nám může někdy přijít vhod tzv. výčtový typ. Např.:

```
type DEN = (Pondělí, Úterý, Středa, Čtvrtek, Pátek, Sobota, Neděle);
```

Jednotlivé názvy dnů jsou identifikátory konstant typu DEN. Uspořádání přiřazených hodnot je dáno pořadím těchto identifikátorů v deklaraci. Tak Pondělí je přiřazeno 0, Úterý 1, nakonec Neděli 6 (ordinální čísla). Z toho vyplývá, že zde můžeme pracovat i s rovností a nerovnostmi.

Každý výčtový typ je ordinální (jeho ordinální číslo je reprezentováno standardní funkcí ord). Podobně můžeme při operacích s výčtovým typem použít funkce succ a pred (viz minulé pokračování):

```
ord(Středa)=2
succ(Pondělí)=Úterý
pred(Neděle)=Sobota
```

V našem případě succ (Neděle) a pred(Pondělí) není definováno.

Na závěr si ověříme některé nabyté poznatky na herním programu Převracečka.

```
program Prevracecka;

{ Cilem hry je prebarvit vsechna pole sachovnice 5 x 5 z
puvodni cerne barvy na bilou. Urcenim souradnic pole se vzdy
prevrati barva tohoto pole i ctyr poli stranove sousedicich z
cerne na bilou, resp.naopak. }

type Sachovnice = array [1..5, 1..5] of boolean;

var Plocha : Sachovnice;
    i, j   : integer;
    Konec  : boolean;

begin
{ Pocatecni stav plochy sachovnice }
  for j :=1 to 5 do
    for i :=1 to 5 do
      Plocha [i, j] :=false;
  Konec :=false;
  while not Konec do
    begin
{ Tisk plochy }
      writeln;
      writeln (' +---+---+---+---+---+ ');
      for i :+1 to 5 do
        begin
          write (' | ');
          for j :=1 to 5 do
            if not Plocha [i, j] then
              write ('X | ');
            else
              write ('  | ');
          writeln;
          writeln (' +---+---+---+---+---+ ');
        end;
{ Ulozeni souradnic pole sachovnice }
      writeln (' Zadej souradnice ');
      writeln (' (radka/sloupec 1..5/1..5): ');
      read (i, j);
{ Provedeni tahu }
      Plocha [i, j] :=not Plocha [i, j];
      if j+1<6 then
        Plocha [i, j+1] :=not Plocha [i, j+1];
      if j-1>0 then
        Plocha [i, j-1] :=not Plocha [i, j-1];
      if i+1<6 then
        Plocha [i+1, j] :=not Plocha [i+1, j];
      if i-1>0 then
        Plocha [i-1, j] :=not Plocha [i-1, j];
{ Test, zda hra konci }
      Konec :=true;
      for i :=1 to 5 do
        for j :=1 to 5 do
          if not Plocha [i, j] then
            Konec :=false;
    end;
  writeln;
  writeln (' Gratuluji, vyhral jste! ');
end.
```

Podle Bajtku 12/86 a
Programování v jazyku Pascal, SNTL 1986
zpracoval -elzet-

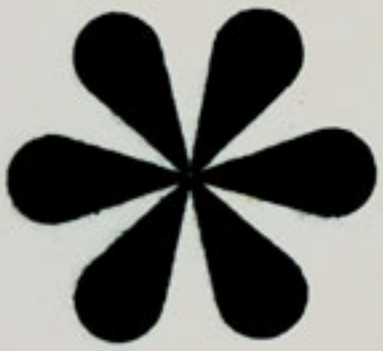
Mikroprocesor Z80

AUTONOMNÍ ASSEMBLEROVÉ RUTINY
(5. část)

Jedním z důležitých nástrojů pro odlaďování programů ve strojovém kódu je tzv. krokování, resp. trasování programu. Při krokování (single stepping) jdeme programem postupně po jedné instrukci (např. po stisknutí tlačítka se provede vždy jen jedna instrukce adresovaná reg. PC). Tak můžeme na monitoru paměti a registrů mikroprocesoru sledovat, jak se program odvíjí. Trasování (tracing) je výrazně zpomalený běh programu. Rychlost trasování lze programově určit. Profesionální odlaďovač programů (debugger, analyzer) obvykle disponuje četnými možnostmi provádění testů vývoje (analýzy) programu během jeho trasování.

Následující rutina NEWPC je základní kostrou odlaďovače programu ve strojovém kódu. Jejím principem je předběžné určení obsahu programového čítače (reg. PC). Je tedy určován obsah, jaký bude tento registr obsahovat po provedení právě realizované instrukce (anticipace obsahu reg. PC). Základní funkce se dělí na dvě hlavní části. U instrukcí, které nemění programové řízení skokově, se k předchozímu obsahu reg. PC připočte pouze délka prováděné instrukce (počet bajtů). U skokových instrukcí (JP,JR, CALL, RET, DJNZ, RST) je nutné obsah reg. PC vypočítat podle odchylky dané tou kterou instrukcí. Většina z těchto instrukcí může ještě obsahovat podmínku. Rutina proto musí zjistit, zda je podmínka v daném momentu platná či nikoli.

NEWPC netestuje blokové instrukce (např. LDIR, CPIR, OTIR apod.) a případy, kdy instrukce DJNZ "skáče sama na sebe" na jednom programovém řádku assembleru. Uvedené musíte ošetřit vlastní řídící rutinou, z níž budete do NEWPC vstupovat. Řídící rutina musí m.j. určovat tzv. bod přerušení (breakpoint), během nějž se do zásobníku na níže uvedené adresy budou ukládat průběžné obsahy registrů testovaného programu. Touto rutinou budete určovat i mód krokování nebo trasování (a jeho rychlost). Do rutiny bodu přerušení se budete vracet na adresu, kterou uložíte na adresy SP a SP+1. Do cílového (krokovaného) programu budete vstupovat adresou SP+0EH a SP+0FH (viz níže). Součástí NEWPC je i subrutina LENGTH pro výpočet délky instrukce. LENGTH v uvedeném výpisu není, najdete ji ve zpravodaji Mikrobáze č. 4.



```
**************************************************************
* NEWPC        Předurčení obsahu programového čítače (reg.PC)   *
**************************************************************
:Cinnost       Výpočet obsahu reg.PC, jaký bude po provedení
               právě vykonávané instrukce. Toto předurčení obsahu
               reg.PC umožní provádět trasování a krokování
               programy ve strojovém kódu.
:Akce          Určení počtu bajtů prováděné instrukce
               IF NOT PC ovlivněno instrukcí skoku:
               THEN: (přičti délku instrukce)
               ELSE: (vypočti nový obsah PC)
--------------------------------------------------------------
:CPU           Z80
:Hardware      Paměť obsahující cílový (object) program
:Software      LENGTH - subrutina pro výpočet délky instrukce
               CTEST - nastavuje CY na log.1, když je splněna
               podmínka uvedená v instrukci krokovaného programu
--------------------------------------------------------------
:Vstup         HL adresuje 1.bajt prováděné instrukce
               Zásobník na vstupu musí obsahovat:
               SP+0,1: PC  (adr.návratu do rutiny bodu přerušení)
               SP+2,3: AF  |     - v pořadí FA (odbobně CB,ED,LH)
               SP+4,5: BC  |   !Uložení obsahů registrů
               SP+6,7: DE  |---!cílového programu do zásobníku
               SP+8,9: HL  |   !rutinou bodu přerušení
               SP+A,B: IX  |
               SP+C,D: IY  |
               SP+E,F: PC  (adr.návratu do cílového programu)
:Výstup        HL - předurčený obsah reg.PC
               E - délka prováděné instrukce
               obsah AF, BC, D je změněn
:Chyby         Žádné
:Registry      AF, BC, DE, HL
:Zásobník      6 (včetně CALL LENGTH)
:RAM           -
:Délka         184
:Cykly         Neuvedeny
--------------------------------------------------------------
:Třída 2       -diskrétní      *přerušitelná    *promovatelná
-***-*         *opakovatelná   -relokovatelná   *robustní
**************************************************************
;
NEWPC   PUSH  HL        ;Ulož ukazatel instrukce a     E5
        CALL  LENGTH    ;najdi její délku              CD lo hi
        POP   HL        ;Ukaz.instr.zpět do HL         E1
        LD    A,(HL)    ;1.bajt instr.do reg.A         7E
        INC   HL        ;Ukaz.instr.zvyš o 1           23
        CP    18H       ;Je instr.relat.skokem?        FE 18
        JR    NZ,DJUMP  ;Když NE, skok na DJUMP        20 0B
;
RJUMP   LD    B,0       ;B=0 pro vyšší bajt offsetu    06 00
        LD    C,(HL)    ;Offset relat.skoku do reg.C   4E
        INC   HL        ;Ukaz.instr. zvyš o 1          23
        BIT   7,C       ;Dvojk.kompl.test (Sign)       CB 79
        JR    Z,POS     ;a převod na                   28 01
        DEC   B         ;"16-bitový offset"            05
POS     ADD   HL,BC     ;Určení nového obsahu          09
        RET             ;reg.PC pro rel.skok           C9
;
DJUMP   CP    10H       ;Je instr.DJNZ?                FE 10
        JR    NZ,AJUMP  ;Když NE, skok na AJUMP        20 0B
        PUSH  HL        ;Uschovej ukaz.instr.          E5
        LD    HL,5      ;Dále převeď pův.obsah reg.B   21 05 00
        ADD   HL,SP     ;krokovaného programu          39
        LD    B,(HL)    ;do reg.B                      46
        POP   HL        ;Obnov ukaz.instr.             E1
        DJNZ  RJUMP     ;Imituj DJNZ, dokud není Z=1   10 E8
        INC   HL        ;Zvyš ukaz.instr.              23
        RET             ;Reg.PC pro další instrukci    C9
;
AJUMP   CP    0C3H      ;Jde o nepodm.JP nebo CALL?    FE C3
        JR    Z,RDADR   ;Když ANO, přečti jejich       28 19
        CP    0CDH      ;2.a 3.bajt                    FE CD
        JR    Z,RDADR                                  28 15
        CP    0C9H      ;Jde o nepodm.RET?             FE C9
        LD    BC,14     ;Indexuj pův.PC pomocí BC      01 0E 00
        JR    Z,STADR   ;a utvoř nový PC               28 0B
        CP    0EDH      ;Jde o instrukci ED?           FE ED
        JR    NZ,JPHL   ;Když NE, skok na JPHL         20 0F
        LD    A,(HL)                                   7E
        AND   0F7H      ;Je to RETI nebo RETN?         E6 F7
        CP    45H       ;(PC dtto jako u RET)          FE 45
        JR    NZ,NOTFR  ;Když NE, skok na NOTFR        20 1F
;
STADR   LD    H,B       ;Zásobníkový index             60
        LD    L,C       ;převeď z BC do HL, přičti SP  69
        ADD   HL,SP     ;SP teď adresuje reg.v zás.    39
```

```
RDADR   LD      A,(HL)      ;Do HL nové slovo pro          7E
        INC     HL          ;nový obsah PC                 23
        LD      H,(HL)      ;(alias LD HL,(HL))            66
        LD      L,A                                        6F
        RET                 ;Návrat s novým reg.PC         C9
;
JPHL    CP      0E9H        ;Pro instr.JP (HL) podobný     FE E9
        LD      C,8         ;indexový postup s pův.HL      0E 08
        JR      Z,STADR                                    28 F2
        CP      0DDH        ;Indexování pův.IX             FE DD
        LD      C,10        ;Jedná se o JP (IX)?           0E 0A
        JR      Z,JPINDX    ;Když ANO, skok na JPINDX      28 06
        CP      0FDH        ;Totéž pro IY                  FE FD
        LD      C,12                                       0E 0C
        JR      NZ,JRCC     ;Skok na JRCC, když neindex.   20 09
;
JPINDX  LD      A,(HL)      ;2.bajt index.instr.do reg.A   7E
        CP      0E9H        ;Když jde o index.skok,        FE E9
        JR      Z,STADR     ;jdi určit nový PC             28 E1
;
NOTFR   LD      C,E         ;Žádný skok, tak přičti délku  4B
        DEC     HL          ;instrukce k orig.hodnotě      2B
        ADD     HL,BC       ;ukazatele (=nový PC)          09
        RET                 ;a vrať se                     C9
;
JRCC    LD      D,A         ;1.bajt rel.skoku do reg.D     57
        AND     0E7H        ;Test, zda je skok podmíněn    E6 E7
        CP      20H                                        FE 20
        JR      NZ,RSTN     ;Když NE, skok na RSTN         20 09
        RES     5,D         ;Test Z nebo CY (ne V a S)     CB AA
        CALL    CTEST       ;Výsledek uložen do CY         CD lo hi
        JR      C,RJUMP     ;Při nepodm.skoku na RJUMP     38 9B
        INC     HL          ;Jinak přeskoč offset          23
        RET                 ;a vrať se                     C9
;
RSTN    AND     0C7H        ;Test instrukce RST            E6 C7
        CP      0C7H        ;Není-li to RST,               FE C7
        JR      NZ,JPCC     ;skok na JPCC                  20 06
        LD      A,D         ;Použij bity 3 a 5 op.kódu     7A
        AND     38H         ;pro určení adresy nulové      E6 38
        LD      H,B         ;stránky. 00H je vyšší bajt    60
        LD      L,A         ;z reg.B, do L nižší bajt z A  6F
        RET                 ;HL je nový PC; návrat         C9
;
JPCC    CP      0C2H        ;Je instr.podm. JP?            FE C2
        JR      Z,ABSCC     ;Když ANO, skok na ABSCC       28 10
        CP      0C4H        ;Je to podm. CALL?             FE C4
        JR      Z,ABSCC     ;Když ANO, skok na ABSCC       28 0C
        CP      0C0H        ;Jde o podmíněný RET?          FE C0
        JR      NZ,NOTFR    ;Když NE, skok na NOTFR        20 D4
        CALL    CTEST       ;Test podmínky u RET           CD lo hi
        RET     NC          ;HL=nový PC při CY=0           D0
        LD      C,14        ;Při CY=1 indexuj pův.PC       0E 0E
        JR      STADR       ;a urči jej                    18 AD
;
ABSCC   CALL    CTEST       ;Test podmínky u JP a CALL     CD lo hi
        JR      C,RDADR     ;Když CY=1, ber vše jako JP    38 AB
        JR      NOTFR       ;Jinak bez transferu           18 C5
;
;...Testování podmínky obsažené v instrukci;
;...CY=1, když podmínka platí; CY=0, když neplatí.
;
CTEST   PUSH    HL          ;Uschovej ukaz.instrukce       E5
        LD      HL,2        ;Indexuj pův.stav.indikátory   21 02 00
        ADD     HL,SP       ;reg.F                         39
        LD      A,(HL)      ;a ulož je do reg.A            7E
        POP     HL          ;Obnov ukaz.instr.             E1
        BIT     3,D         ;Když je podmínka splněna      CB 5A
        JR      NZ,HBIT     ;pro log.0,                    20 01
        CPL                 ;invertuj všechny bity         2F
;
HBIT    BIT     5,D         ;Rozdělení na testy            CB 6A
        JR      NZ,LBIT     ;Z!CY nebo S!V                 20 07
        BIT     4,D         ;Rozdělení na testy            CB 62
        JR      NZ,CARY     ;Z nebo CY                     20 09
;
ZERO    RLA                 ;Kopie Z do kopie S            17
SIGN    RLA                 ;Kopie S do indik.CY           17
        RET                 ;Výsledek v CY                 C9
;
LBIT    BIT     4,D         ;Rozdělení na V nebo S         CB 62
        JR      NZ,SIGN     ;Při Z=0 skok na SIGN          20 FA
;
PRTY    RRA                 ;Kopie P!V                     1F
        RRA                 ;do kopie CY                   1F
CARY    RRA                 ;Kopie CY do indik.CY          1F
        RET                 ;Výsledek v CY                 C9
```

****

NEWPC testuje instrukce, zapříčiňující odchylku v programovém řízení, v tomto pořadí:

1	JR	dis	18	
2	DJNZ	dis	10	
3	JP	addr	C3	
4	CALL	addr	CD	
5	RET		C9	
6	RETI,	RETN	ED 0100 X101	
7	JP	(HL)	E9	
8	JP	(IX)	DD E9	
9	JP	(IY)	FD E9	
10	JR	cc,dis	001X X000	(cc je podmínka)
11	RST	page 0	11XX X111	
12	JP	cc,addr	11XX X010	
13	CALL	cc,addr	11XX X100	
14	RET	cc	11XX X000	

Test se zastaví v případě, že po kódu ED, DD nebo FD nenásleduje platný kód skoku. Vhodnou úpravou rutiny NEWPC byste mohli trasovat i programy, které obsahují nestandardní (tzv. ilegální) instrukce Z80.

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

Paralelní přenos dat

Část 2.

V minulé části byly popsány obvody používající LSTTL (nízkovýkonovou Schottky TT-logiku) a ALSTTL (rozvinutou LSTTL). Často však bývá jednodušší a výhodnější použít jeden interfejsový adaptér typu LSI namísto mnoha menších pouzder.

Tyto adaptéry mají mnoho podob. Pracují jako diskoví "dispečeři", "regulátory" CRT, ACIA (asynchronní komunikační interfejsové adaptéry), USARTy a různé paralelní I/O adaptéry. Všechna tato zařízení jsou programovatelná. Softwarově můžete měnit jejich funkce a způsob jejich provádění (ovšem v rozumných mezích).

Následující materiál se věnuje paralelním I/O interfejsovým adaptérům. Nicméně se nepokouší o kompletní funkční popis zařízení - to by zabralo příliš mnoho místa. Místo toho jsou zdůrazněny některé jejich významější odlišnosti. Pokud potřebujete ještě detailnější informace, nahlédněte do technických informací výrobců.

PŘEMÍRA ZKRATEK

Literatura zabývající se paralelními periferními interfejsovými adaptéry obsahuje skutečnou houšť zkratek:

6820, 6821 a 6520 jsou PIA (periferní interfejsové adaptéry)

6522 je VIA (univerzální interfejsový adaptér)

6822 je IIA (industrial, tedy průmyslový interfejsový adaptér)
8255 je PPI (programovatelný periferní interfejs)
Z8420 z rodiny Z80 je PIO (paralelní Input/Output, tedy vstupně/výstupní řadič)

Schémata v tomto článku jsou funkčními blokovými diagramy a zahrnují popis výstupů pouzder pěti nejpopulárnějších paralelních interfejsových čipů typu LSI.

I když jsou si uvedené obvody v mnohém podobné, každý má své zvláštní dispozice, jimiž vyhovuje určitým aplikacím lépe než ostatní. Všechny mohou být naprogramovány pro "jednopouzdrové řešení" funkcí klávesového či tiskového interfejsu z minulé části.

Interfejsové adaptéry používáme z několika důvodů. Během výstupu fungují jako buffery přenosu dat ze systémové sběrnice. Jejich poměrná zatížitelnost je značná, avšak jejich mnohem důležitější funkcí je ochrana - adaptér izoluje systém od vnějšího světa (nepřenáší poruchy atd. do systému). Dojde-li k závažným poruchám zařízení, může se sice poškodit adaptér, počítačový systém však zůstane funkční. Kdyby vnější zařízení byla připojena přímo na sběrnici, tato ochrana by pochopitelně neexistovala. Porucha zařízení by mohla vážně poškodit mnoho součástí mikropočítače.

Propojovací adaptéry mohou být používány pro vyrovnání časových rozdílů mezi systémem počítače a pomalejšími vnějšími zařízeními jako jsou tiskárny nebo relativně pomalé integrované obvody pro speciální účely. Univerzálnost těchto adaptérů umožňuje předefinování funkcí vstupu a výstupu dle potřeby a bez jakýchkoli hardwarových úprav. Vstupy se mohou stát výstupy a naopak. Způsob funkce systému tak může být rychle změněn při změně požadavků na něj kladených. Použití jednoho pouzdra místo mnoha čipů typu SSI a MSI rovněž zjednodušuje rozvržení plošných spojů a zkracuje dobu jejich navrhování i výroby.

Z8420 PIO

Mikroprocesor Z8420 PIO, pocházející z rodiny Z80, se liší od ostatních PIA probíraných v tomto článku tím, že všechny přenosy dat jím procházejících jsou řízeny přerušením. Jeho dva I/O porty jsou označeny A a B. Každý se skládá z osmi datových linek a dvou linek pro handshaking. Čtyři různé operační módy určují konfiguraci portů.

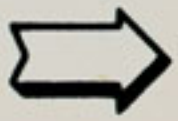
V módu 0 je všech osm datových vývodů portu výstupních; handshaking má jeden výstup a jeden vstup.

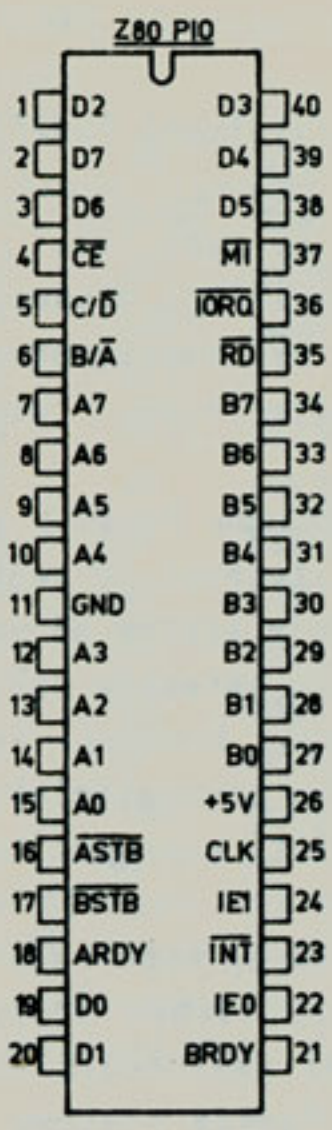
V módu 1 je všech osm datových vývodů portu vstupních; linky pro handshaking jsou stejné jako v módu 0.

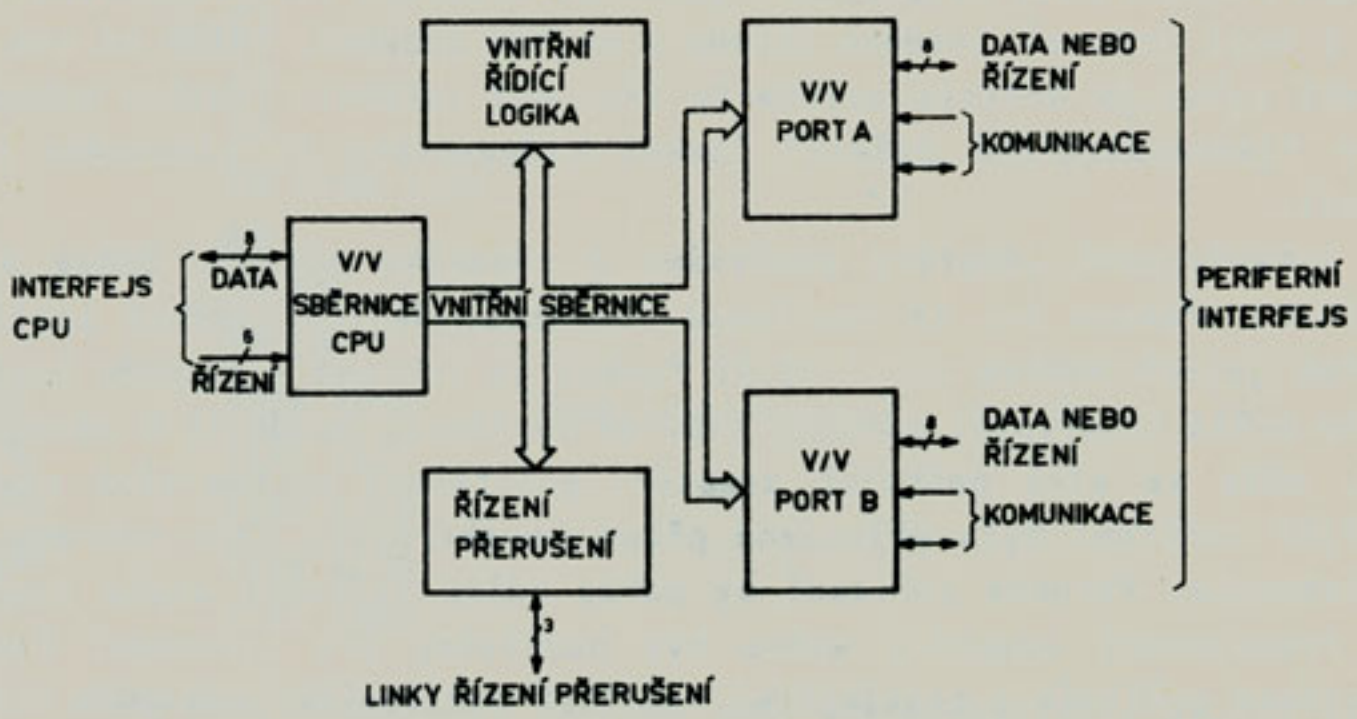
Mód 2 využívá datové linky portu A a handshaking obou portů pro obousměrný datový přenos.

V módu 3 je každý jednotlivý datový I/O vývod definován buď jako vstup nebo jako výstup. Maskovací registr určuje, zda je na vstupu generováno přerušení.

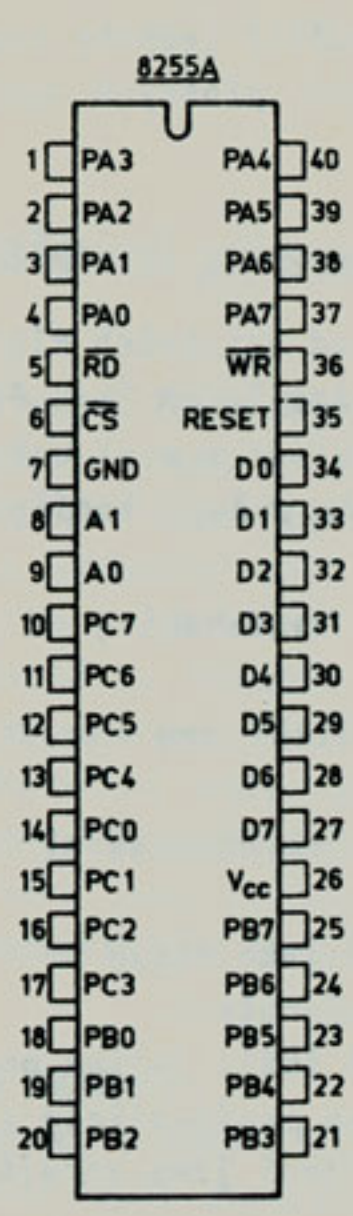
Vzhledem k univerzálnosti použití přerušení, včetně rozsáhlé škály jejich priorizace, je tento integrovaný obvod předurčen pro práci se systémem používajícím mikroprocesor Z80 především v případě, kdy je obsluhována řada zařízení jako zdrojů přerušení s požadavkem na jejich řazení podle daných priorit.

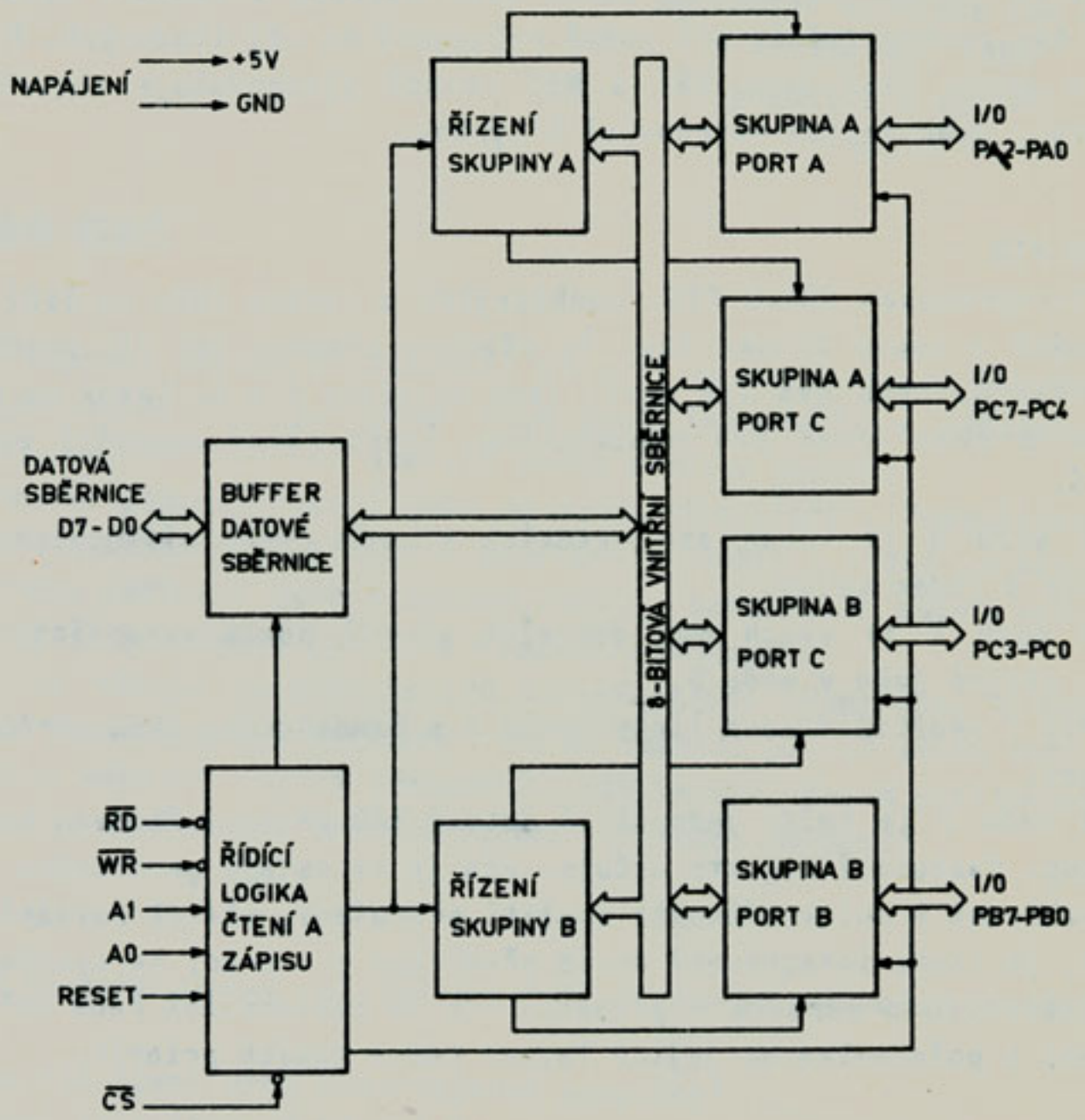






Z8420 PIO





8255 PPI

8255 PPI

Interfejsový adaptér 8255 PPI od firmy Intel patří k prvním, které se objevily na trhu. Vyrábějí jej rovněž firmy National Semiconductor a NEC. Původně byl navržen pro použití v systémech 8008 a 8080A (což prozrazuje jeho stáří).

24 vývodů je seskupeno do tří portů, označených A, B, C. Tři operační módy určují funkci portů. Obecně jsou porty A a B datové, port C může být buď datový nebo řídící, což závisí na zvoleném operačním módu. Porty A a B mohou být programovány jako celovstupové nebo celovýstupové (nelze je programovat vývod po vývodu, což lze u portů některých jiných adaptérů).

V módu 0 je každý port rozdělen do dvou skupin po čtyřech vývodech; každá skupina je definovatelná buď jako vstupní nebo výstupní.

V módu 1 mohou být porty A a B definovány individuálně jako vstupní nebo výstupní. V tomto módu horní polovina portu C realizuje handshaking portu A, dolní portu B.

Mód 2 používá osm linek portu A pro obousměrný přenos a pět linek z portu C slouží funkci handshaking a řízení.

I/O vývody tohoto integrovaného obvodu jsou funkčně podobné portu B ze série 6500 a 6800 PIA. N-kanálové tranzistory spojují jak zem, tak kladné napájení, když je vývod konfigurován jako výstup. Proto se při vysoké úrovni výstupy dostávají do TTL úrovní (2,4 V minimálně) i bez zdvíhacích rezistorů. Ale pro získání prahových hodnot CMOS je nutné rezistory připojit (obvykle 10 kiloohmů).

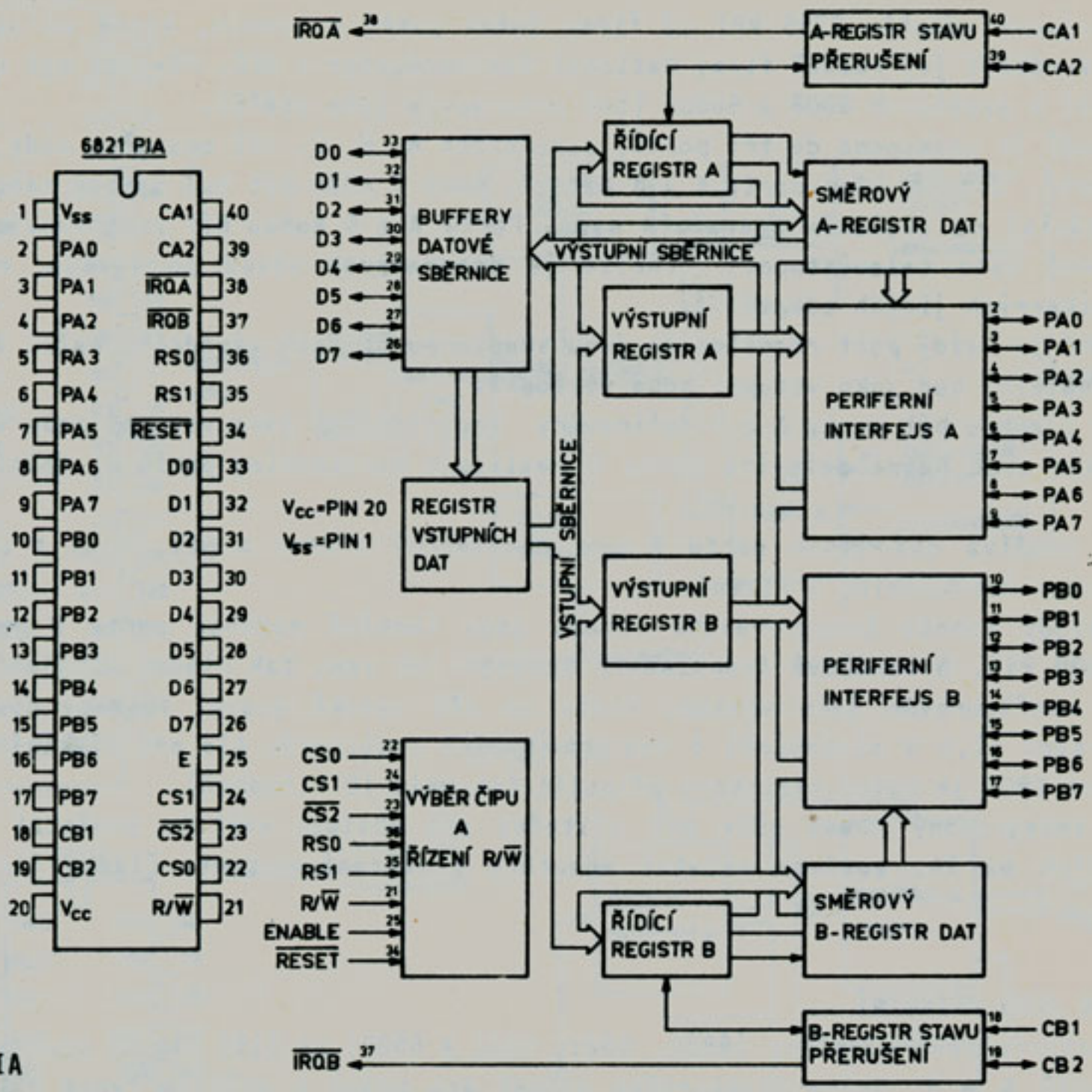
Tento integrovaný obvod může být užitečný při obsluze velkého množství linek, protože jich má 24, zatímco ostatní adaptéry probírané v tomto článku jich mají pouze 20.

6821 PIA A JEHO PŘÍBUZNÍ

Tyto interfejsové adaptéry (6820, 6821, 6822 a 6520) se liší hlavně charakteristikami portů A a B. Vyjma stanoveného určení pro práci s určitým typem externích zařízení jsou tyto adaptéry téměř identické.

Všechny obsahují šest registrů zabírajících čtyři adresy. Každý port má po jednom řídícím, směrovém a datovém registru. Směrový registr každého portu se dělí o adresu s datovým registrem portu. Aktivní registr se volí podle stavu bitu řídícího registru portu. Když je bit nulový, je akční směrový registr; je-li nastaven, je zpřístupněn regist datový. Směrový registr portu A je obvykle značen DDRA, portu B DDRB. Stav (0 nebo 1) každého individuálního bitu směrového registru určuje, zda odpovídající vývod datcvého registru je vstupní nebo výstupní. Je-li bit=0, odpovídající vývod datového registru je vstupní. Když je bit=1, korespondující vývod datového registru je výstupní. Tato programovatelnost linku po lince poskytuje značnou univerzálnost.

Každý port má dva vývody pro handshaking. Vývody portu A jsou označeny CA1 a CA2; portu B CB1 a CB2. Vývody CA1 a CB1 mohou být použity pouze jako vstupní. Aktivní přechod na každém z nich nastaví stavový indikátor v odpovídajícím řídícím registru a způsobí, že přerušovaný výstup IRQ-A nebo IRQ-B půjde na úroveň log.0 (je-li toto přerušení umožněno). Každý z vývodů CA2 a CB2 může být vstupní nebo



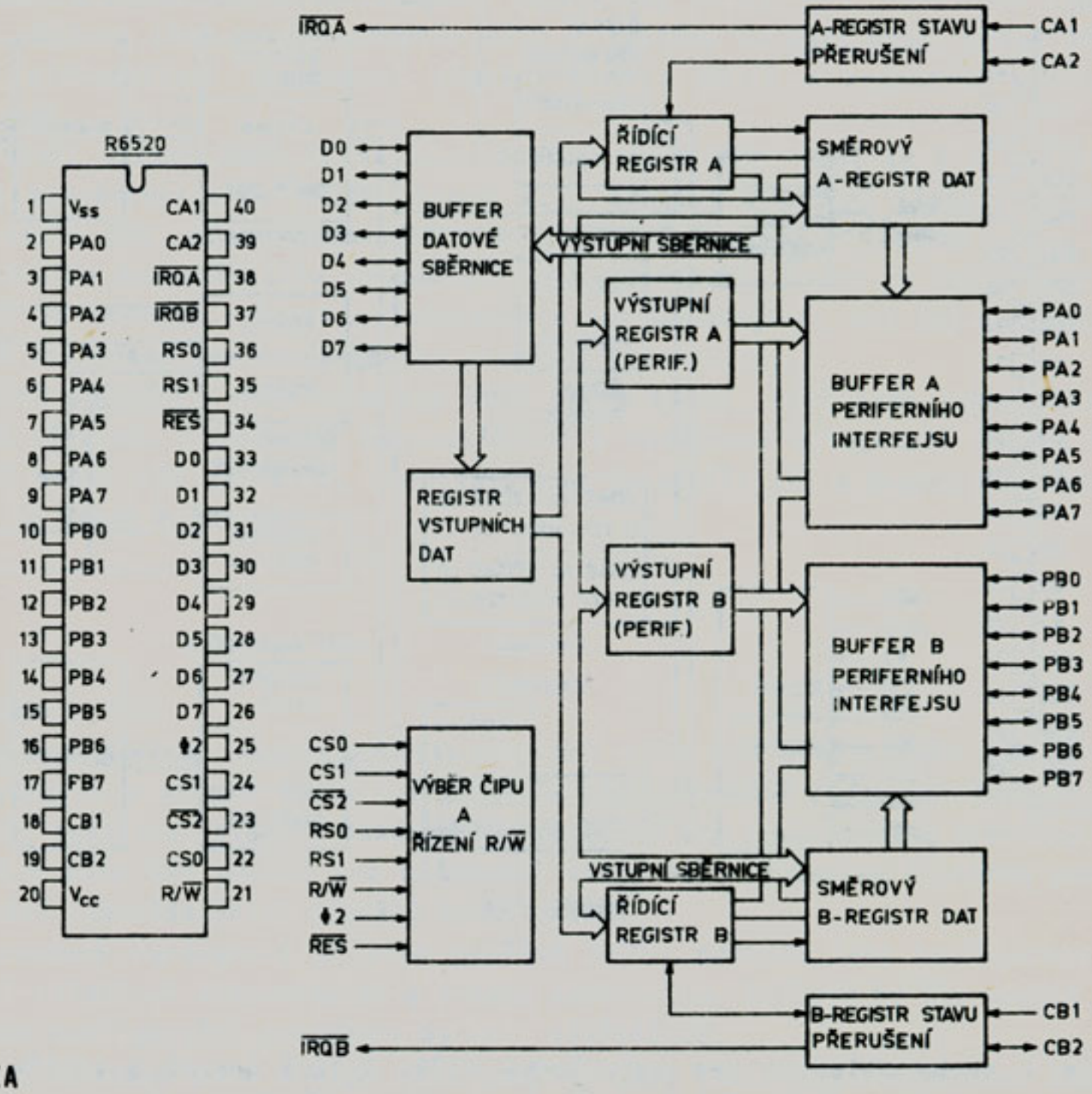
6821 PIA

výstupní. Oba jsou schopny generovat přerušení, jsou-li použity jako vstupní. Konfigurace řídících vývodů je určena nastavením individuálních bitů v řídících registrech.

Tyto dva porty PIA, i když jsou v mnohém podobné, se významně liší svým provedením. Každá linka portu A má n-kanálový tranzistor mezi vývodem a zemí a pasívní zdvíhací rezistor mezi vývodem a kladným napájením. To portu (když je použit jako výstupní) umožňuje řídit obvody CMOS bez vnějších zdvíhacích rezistorů. Průtok proudu je přitom minimální. Port A proto není vhodný pro řízení NPN tranzistorů nebo jiných prvků vyžadujících zdroj proudu.

Port B má aktivní tranzistor mezi vývodem a zemí a další mezi vývodem a kladným napájením. I když vývody portu B mohou dodávat proud pro řízení například NPN tranzistorů, budou na výstupu max. na úrovni TTL. Jsou-li požadovány vyšší úrovně, např. pro řízení CMOS, je nutno použít externí zdvíhací rezistory.

Na PIA 6820 a 6821 je často pohlíženo jako na záměnné, ale v lecčem se významně liší. 6821 je jenom jeden ze dvou mikroprocesorů série 6800 firmy Motorola. Každý vývod portu A mikroprocesoru 6821 je schopný řídit dvě TTL zátěže při použití jako



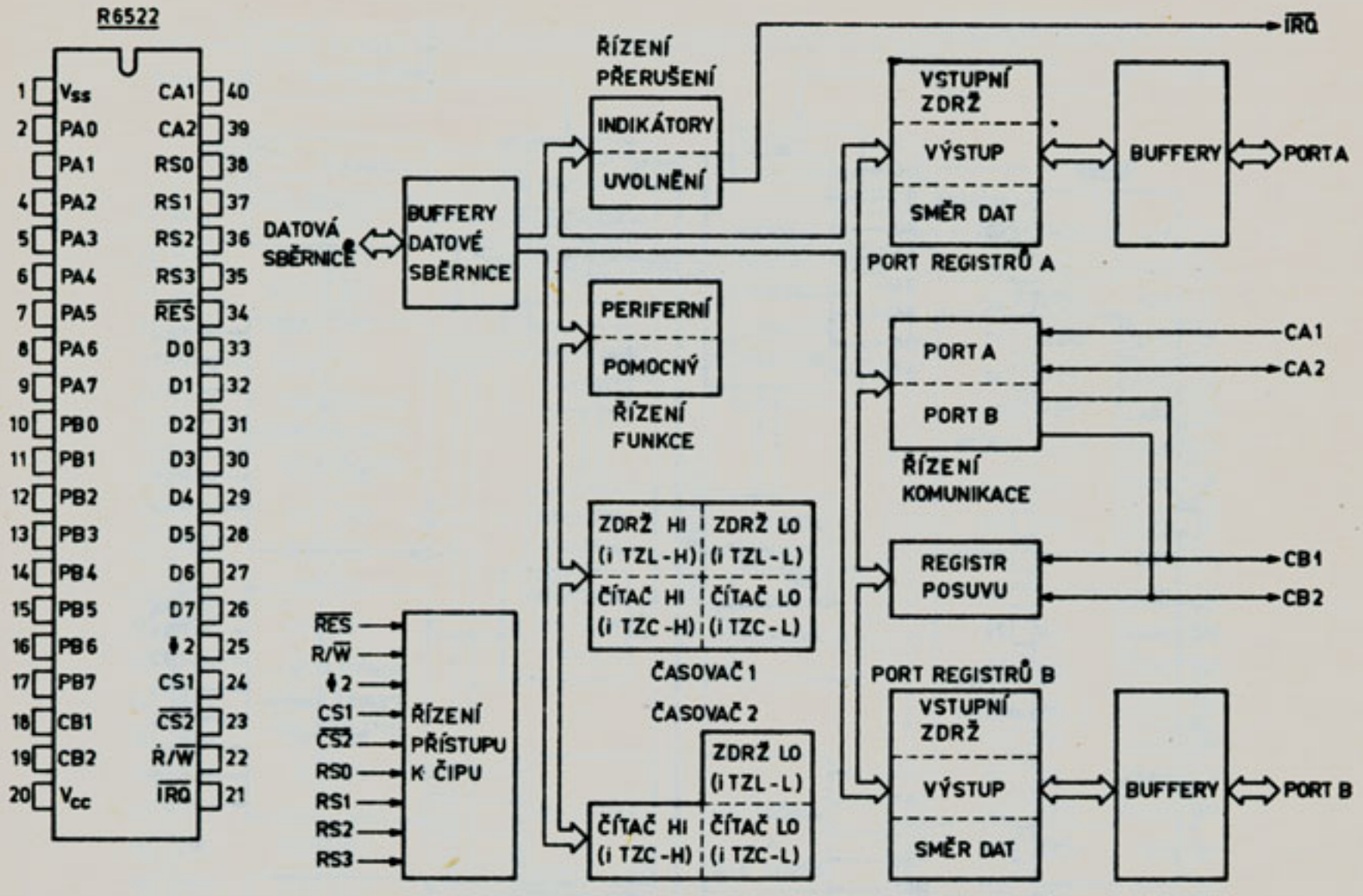
R6520 PIA

výstup a být řízen max. jednou a půl TTL záteže při vstupu. Odpovídající údaje pro 6820 jsou mnohem nižší. Takže 6820 je vhodnější, když prvek s omezenou řídící schopností řídí port A. Pro vyšší řídící schopnost výstupu by měl být použit 6821.

6822 je podobný 6821, až na to, že všechny výstupy jsou typu otevřeného kolektoru, schopné vydržet napětí až 18 V (ačkoli Motorola udává maximum normálního pracovního napětí 15 V). IIA tedy může být použit jako interfejs pro zařízení, operující s podstatně vyšším napětím než je standardní 5 V logika.

Mnoho specializovaných laboratorních přístrojů a průmyslových obráběcích strojů má výstupní BCD data s úrovněmi kolísajícími od napětí blízkého nule až do 12 nebo 14 V. Jak naznačuje označení "průmyslový interfejsový adaptér", schopnost zpracovat vysoké napětí spolu s velkou šumovou tolerancí této součástky může být značně užitečná při použití v typickém průmyslovém prostředí s vysokým šumem a interferencí.

6520 se od 6821 liší jen málo. Výstupy 6520 jsou specifikovány pro řízení jedné TTL zátěže, zatímco 6821 může řídit dvě. Vstupy portu A 6520 využijí max. jednu TTL zátěž řídícího obvodu oproti jednomu a půl násobku portu A 6821. Jinak jsou tyto dvě součástky v podstatě identické.



R6522 VIA

6522 VIA

6522 je v mnoha ohledech jen vylepšeným 6520. Řídící úrovně a zátěže portů A a B jsou v podstatě stejné, včetně značení vývodů portů. 6522 má 16 vnitřních registrů okupujících 16 adres a používá čtyři registrem volitelné linky (na rozdíl od dvou 6520). Má pouze jeden chip-select (s vysokou úrovní) a jeden výstup pro přerušení namísto dvou.

Přídavné registry řídí řadu dalších funkcí: T-1, 16-bitový časovač/čítač (který se snižuje s fází 2 systémových hodin) a druhý čítač, T-2, který se může snížit s každou časovou prodlevou čítače T-1 nebo s každým negativním přechodem na vývodu PB6.

Čítač T-2 dokáže počítat pulzy z vnějšího zdroje. Čip může generovat obdélníkový signál na vývodu PB7, jeho frekvence je určena čítačem T-1. 8-bitový obousměrně posuvný registr umožňuje jednoduchý sériový přenos dat. Programovatelné dispozice funkcí handshaking jsou rozšířeny; 6522 dokáže střádat vstupní data pod kontrolou vnějšího zařízení.

APLIKACE

Nejjednodušší je uzpůsobit interfejsový adaptér mikroprocesoru ze stejné čipové série. U Z80 použijete Z8420 PIO, u 6809 to bude 6821 a tak dále. Mikroprocesory

sérií 6500- a 6800- mohou být obvykle zapojovány volně, bez časovacích nebo jiných problémů. Můžete však použít adaptér 8255 v systému na bázi 6502, pokud dekódujete oddělené nízkoaktivní strobovací signály READ a WRITE a zajistíte vysokoaktivní RESET signál.

V počítači Victor 9000 PC na bázi 8088 je 6522 VIA, jenž je i v počítači Mac-Intosh s mikroprocesorem 68000, který m.j. používá i duální sériový interfejsový čip ze série Z80. V podstatě máte v propojování lecčeho s lecčím velkou svobodu, pokud dodržíte kompatibilitu obvodových charakteristik. Často je lepší použít známý čip, když je schopen vykonat požadovanou práci uspokojivě, než jej nahradit neznámým, jímž dosáhneme jen nevýznamného zlepšení vlastností.

V rodině 6500/6800 použijete typ 6522 pouze tehdy, když bude jeho efekt oproti 6821/6520 výrazný. Zabudovaný časovač/čítač a rozšířené schopnosti funkce handshaking 6522 mohou být užitečné, ale tento typ je složitější a dražší než 6821 a jeho časování je velmi kritické.

INTERFEJSING s 6821

Jakýkoli z těchto interfejsových adaptérů může být použit jako náhrada obvodů LSTTL uvedených v minulé části (**obrázky 2 - 5**) s funkčně podobnými výsledky. Mohou automaticky generovat signály handshaking potřebné pro interfejsing tiskárny. Kterýkoli z nich může být použit pro klávesový vstup nebo výstup na tiskárnu pomocí naprogramování odpovídajících signálů handshaking.

Dále uvedený hardwarový příklad používá 6821, protože jeho konfigurace je pro tento účel relativně srozumitelná a snadno popsatelná. Zajisté by šlo použít i Z8420 a 8255, ale konfigurační procedura by byla složitější.

Schéma 1 ukazuje port A čipu 6821 použitý jako interfejs klávesnice. Datový výstup klávesnice řídí linky PA0-PA7, konfigurované jako vstupy. Klávesnicový datový strobovací signál je připojen na CA1 a CA2 (pokud je to žádoucí, může být připojen i na klávesový vstup). Přechod na CA1 nastaví stavový indikátor řídícího registru - může být použit pro přerušení procesoru přes výstup IRQ-A. Když přerušovací výstup IRQ-A zůstane nezapojený, klávesový I/O může být obsluhován dotazováním. Detaily dekódování byly demonstrovány v minulé části ve schématu 2.

Schéma 1 ukazuje port B PIA, který je použit pro paralelní řízení tiskárny. PB0-PB7 řídí její datové linky. CB2 generuje (nízkoaktivní) strobovací signál DATA po jeden kmit časového pulzu po napsání dat do portu B. ACKnowledge z tiskárny překlápí CB1. To nastaví stavový indikátor řídícího registru portu B - pokud je IRQ-B připojen na přerušovací linku procesoru, ACK z tiskárny způsobí přerušení procesoru. Výstup na tiskárnu může být řízen buď přerušovací subrutinou nebo dotazováním.

Po realizování těchto funkcí lze použít i 8255. Pak port A funguje jako klávesový vstup se čtyřmi vyššími vývody portu C použitými pro handshaking. Portem B vystupují data na tiskárnu a čtyři nižší vývody slouží handshakingu. Adaptér 8255 bude pracovat v módu 1.

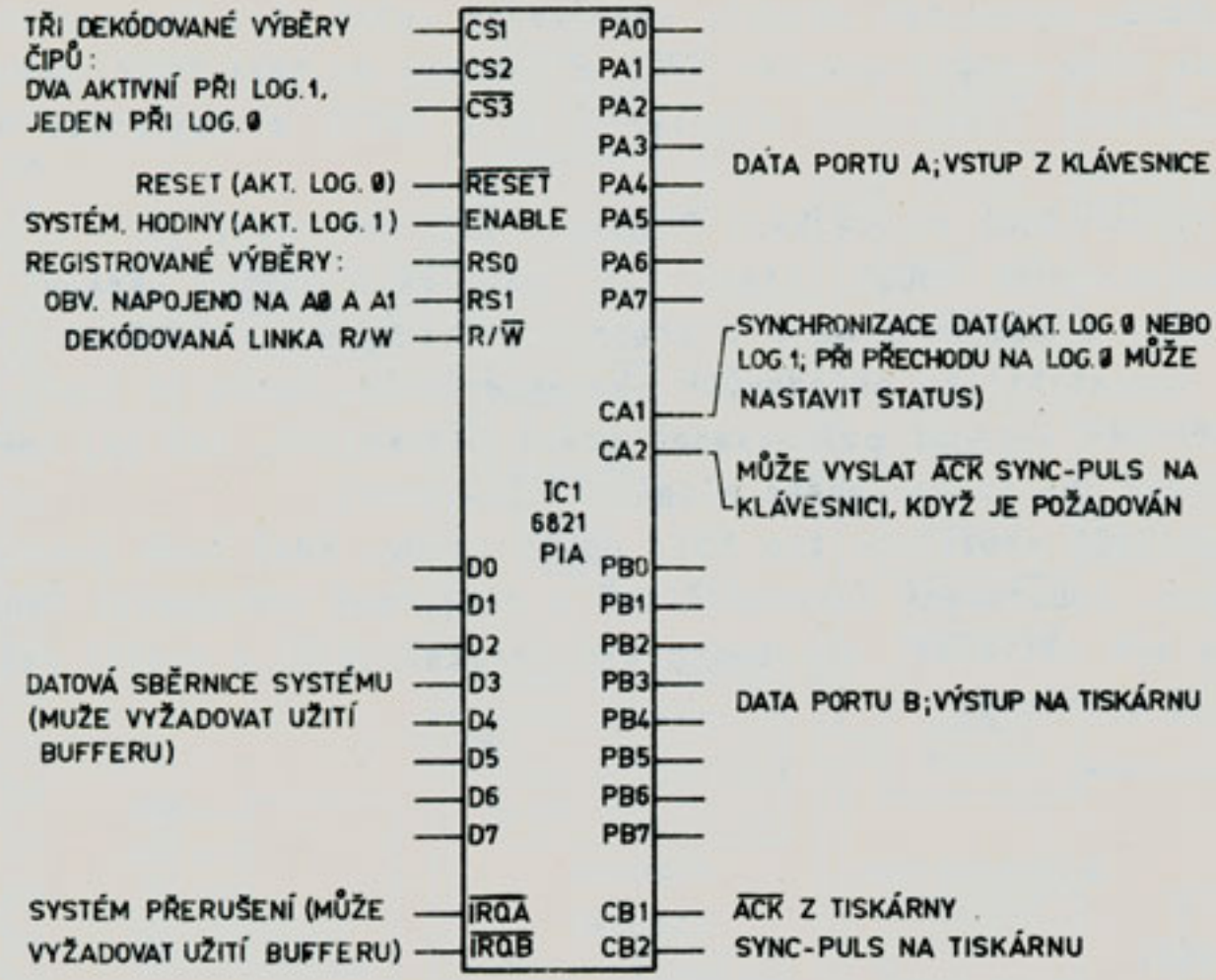


Schéma 1 - Užití PIA jako vstupního (klávesnice) a výstupního (tiskárna) interfejsu

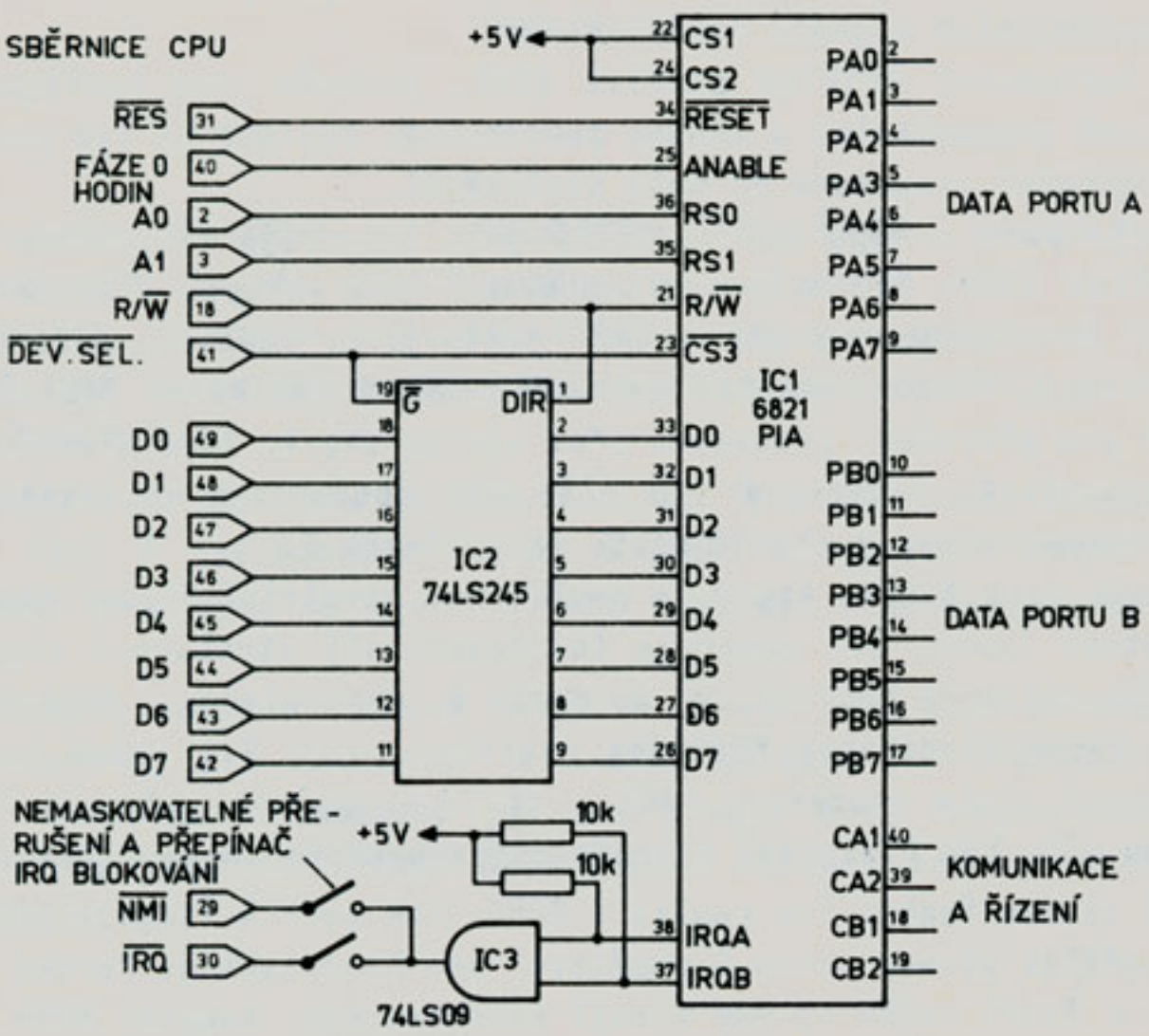


Schéma 2 - PIA interfejs (Apple II)

DVOJITÝ PIA INTERFEJS PRO APPLE II

Schéma 2 ukazuje základní prvky PIA připojeného ke sběrnici mikropočítače Apple II. Dekódování chip-select může být provedeno různými způsoby.

Periferní konektor Apple II má přiděleno 16 vyhražených I/O adres. Vývod 41 konektoru je na úrovni log.0 při práci s kteroukoli z oněch 16 adres. Na schématu 3a jeden PIA zabírá nejnižší čtyři adresy v 16-bajtovém rozsahu, druhý PIA pak čtyři vyšší adresy.

Signál vývodu 41 DEVICE SELECT (volba zařízení) sběrnice mikroprocesoru částečně aktivuje oba PIA, když jde k nule - spouští nízkoaktivní vstup chip-select na každém

IC	5V	GND
IC1	20	10
IC2	14	7
IC3	20	1
IC4	20	1

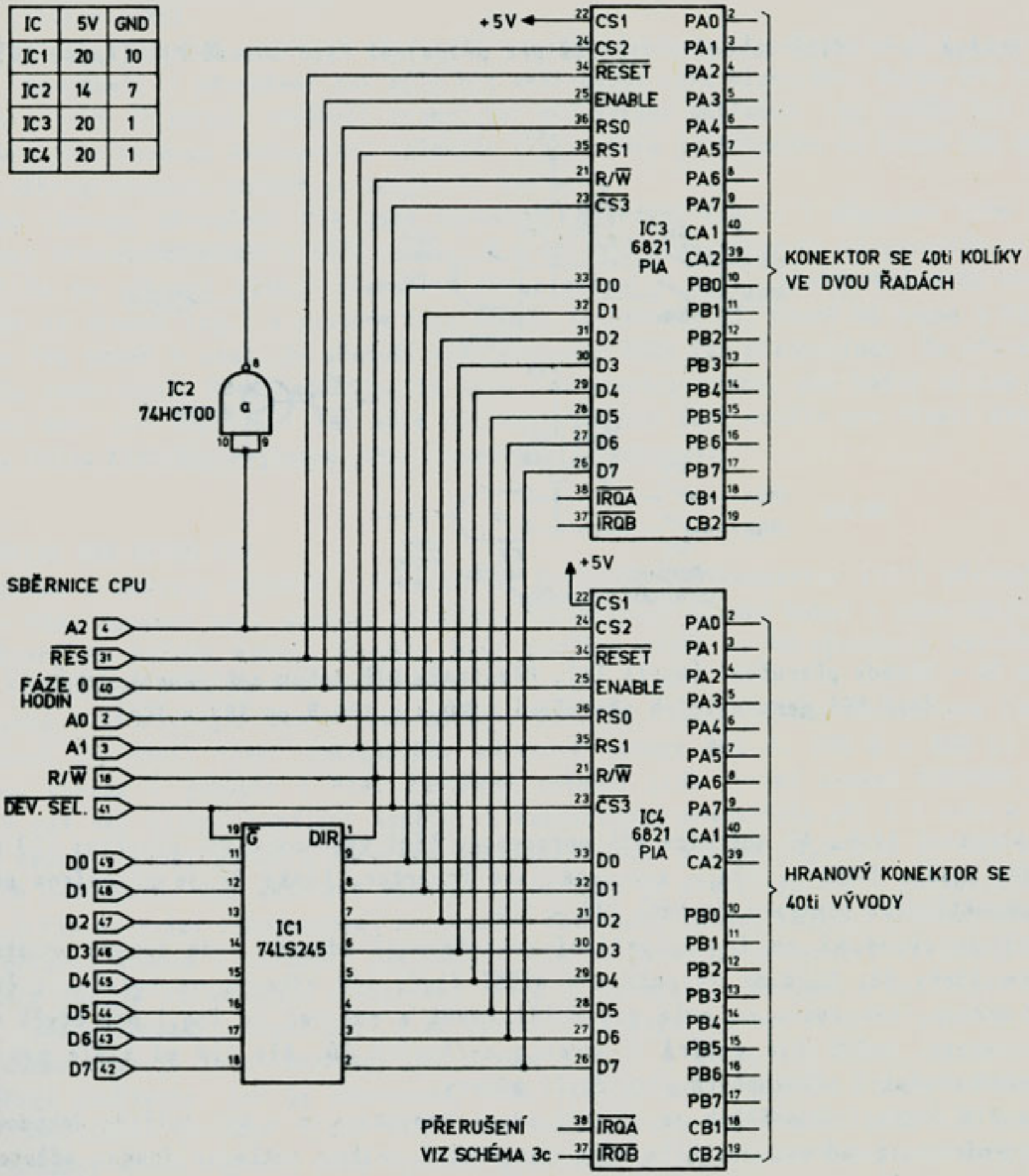


Schéma 3a - Dvojitý interfejs PIA (Apple II)

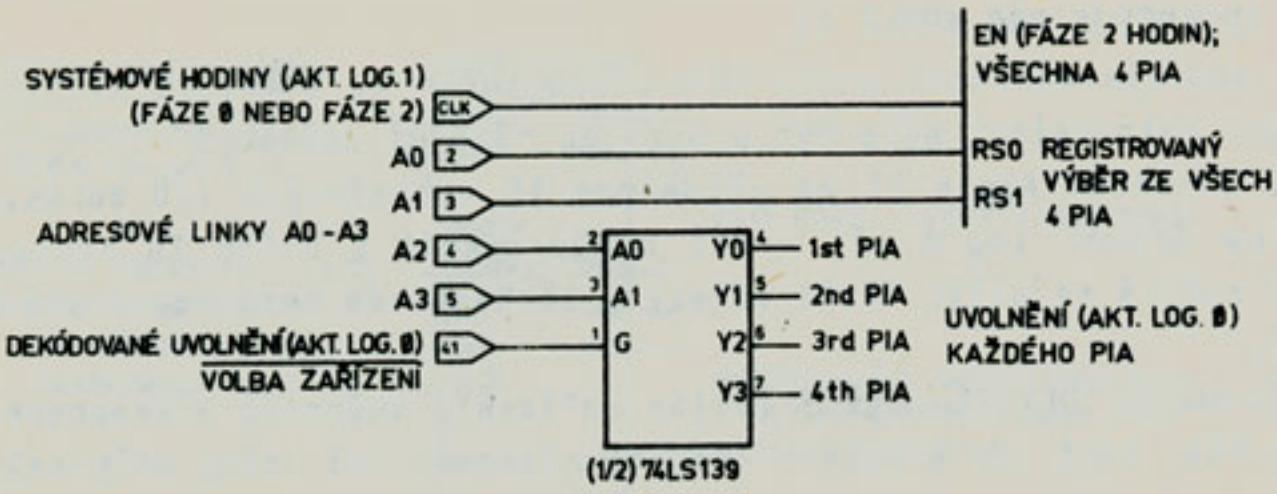


Schéma 3b - Užití dekodéru 74LS139 pro připojení čtyř obvodů PIA (Apple II)

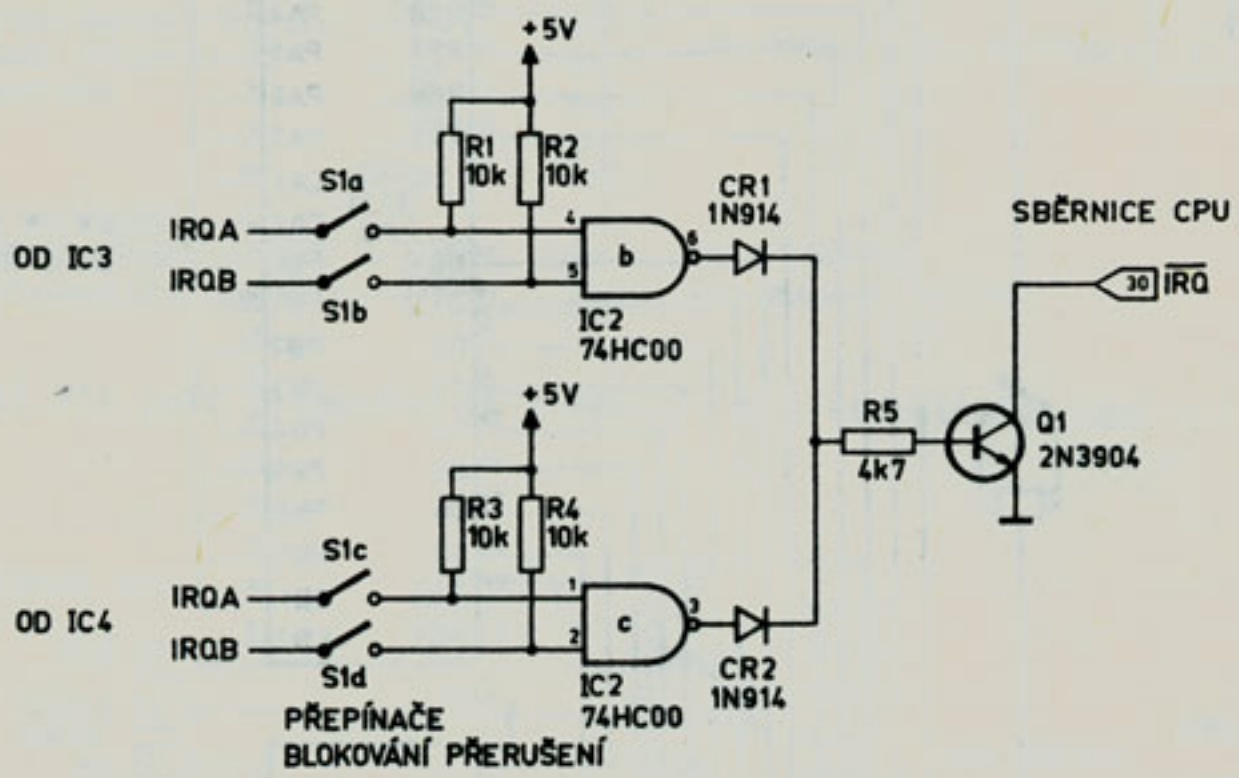


Schéma 3c - Obvody přerušení (Apple II). Přepínače DIP mohou zablokovat jakékoli ze čtyř PIA generujících přerušení (IRQ-A a IRQ-B od IC3 a IC4)

PIA. Adresová linka A2 sběrnice mikroprocesoru řídí vysokoaktivní otevření IC3 přes 74HCT00 hradlo NAND - IC2a - zapojené jako invertor. Linka A2 je připojena přímo k vysokoaktivnímu otevření druhého PIA - IC4.

Když je vývod 41 na log.0, aktivní interfejsový adaptér bude zvolen prostřednictvím linky A2. A2 bude na nule pro nižší čtyři I/O adresy, na log.1 pro čtyři vyšší adresy, pak znovu na nule pro další čtyři a nakonec na log.1 pro čtyři nejvyšší adresy. Každý PIA zabírá 8 adres z celkových 16. Ale čip má stále pouze 6 vnitřních registrů přístupných přes čtyři adresy.

Oba PIA okupují 8 nižších ze 16 I/O adres konektoru a - bez dalšího dekódování - i horních osm adres. Jestliže máte dostatečně velkou základní desku, můžete na ni umístit čtyři PIA a použít polovinu dekodéru 74LS139 pro rozdělení 16-bajtového adresového rozsahu na čtyři 4-bajtové rozsahy a každý PIA přiřadit ke každému bloku

(viz schéma 3b). Transceiver 74LS245 (IC1 na schématu 3a), který ukládá datové linky do vyrovnávací paměti, je spouštěn přímo vývodem 41 sběrnice mikroprocesoru a je vždy aktivní při práci s jakoukoli I/O adresou konektoru.

Pro generování přerušení jsou použity dvě z hradel NAND pouzdra 74HC00 (IC2b a IC2c na schématu 3c) jako zdroje proudu báze tranzistoru 2N3904 přes 1N914 nebo podobné spínací diody (CR1 a CR2). Tranzistor 2N3904 pak řídí vývod IRQ na sběrnici - když je aktivní, úroveň vývodu jde k nule. Každý interfejsový adaptér má dva výstupy pro přerušení: IRQ-A a IRQ-B, oba nízkoaktivní, s otevřeným kolektorem. Každý je udržován na vysoké úrovni 10-kiloohmovým rezistorem (R1-R4) a řídí vstup hradla NAND (IC2b a IC2c).

Přepínače S1a-S1d umožňují uživateli odpojit výstupy přerušení na PIA z hradel NAND. Kterékoli z přerušení tak může být aktivováno nebo blokováno. Volba NPN tranzistoru není důležitá. Jakýkoli tranzistor NPN vykoná potřebné. Typ 2N3904 byl zvolen pro jeho snadnou dostupnost. Kolektor tranzistoru je připojen na linku IRQ sběrnice mikroprocesoru, emitor je spojen se zemí.

Oba výstupy přerušení PIA jsou normálně na úrovni log.1. V důsledku toho jsou výstupy hradel NAND IC2b a IC2c na log. Ø a udržují tranzistor Q1 ve vypnutém stavu. Jestliže se některý výstup přerušení PIA překlopí k nule, jím řízené NAND hradlo přejde na úroveň log.1 a proudem báze otevře tranzistor Q1, který se zapne a stáhne linku IRQ téměř k zemi. To přeruší mikroprocesor 6502 (za předpokladu, že přerušení nebylo maskováno). Přepínače S1a-S1d jsou instalovány proto, aby každý výstup přerušení mohl být odpojen - tak může být odpovídající port použit pro řízení dotazováním (bitu řídícího registru jako stavového indikátoru).

INTERFEJS PRO DESKU PIA

Schéma 4 ukazuje ADC0809 - 8-kanálový, 8-bitový A/D konvertor (IC1) přidaný na desku PIA. Spojení mezi deskami zajišťuje 40-vodičový plochý kabel s odpovídajícími konektory na každém konci. Obvykle bude použito desky se 40-vývodovým zakončením nebo paticového zásuvkového konektoru.

Deska PIA a vnější deska jsou spojeny tak, že řídící vývody konvertoru (ALE - adresové otevření; START CONVERSION; adresové vstupy ADD A, ADD B a ADD C) jsou řízeny portem A jednoho z PIA. Zaručený vstupní práh vysoké úrovně ADC0809 je o 1,5 V níž než kladné napájecí napětí. S konvertorem pracujícím při 5 V nebo o něco výše nemají vývody portu B zaručeno dosažení tohoto minima (3,5-3,62 V) bez zdvíhacích rezistorů. Osm datových výstupů konvertoru je proto spojeno s portem B 6821. Port A PIA se svými zabudovanými zdvíhacími rezistory snadno zajistí prahové minimum konvertoru. Schéma 5 ukazuje, jak port A PIA se svými zabudovanými zdvíhacími rezistory (RA je na schématu znázorněn) může být použit ve spojení s optickými spojkami jako izolovaný vstupní port s až deseti 1-bitovými vstupy (PA0-PA7, CA1 a CA2).

Schéma ukazuje fototranzistor bez předpětí (Q1) a diodu LED (CR1). Mohou to být oddělené součástky, jak je zobrazeno, nebo kombinované v jednom z komerčně dostupných izolačních pouzder (jako 4N26 nebo 4N28). Toto uspořádání může být použito pro snímání stavů zapnutí nebo vypnutí přepínačů a pro monitorování událostí pomalejšího průběhu při nekompatibilních napěťových úrovních.

"Přepínačem" může být fóliové okno v bezpečnostním systému, může to být i další

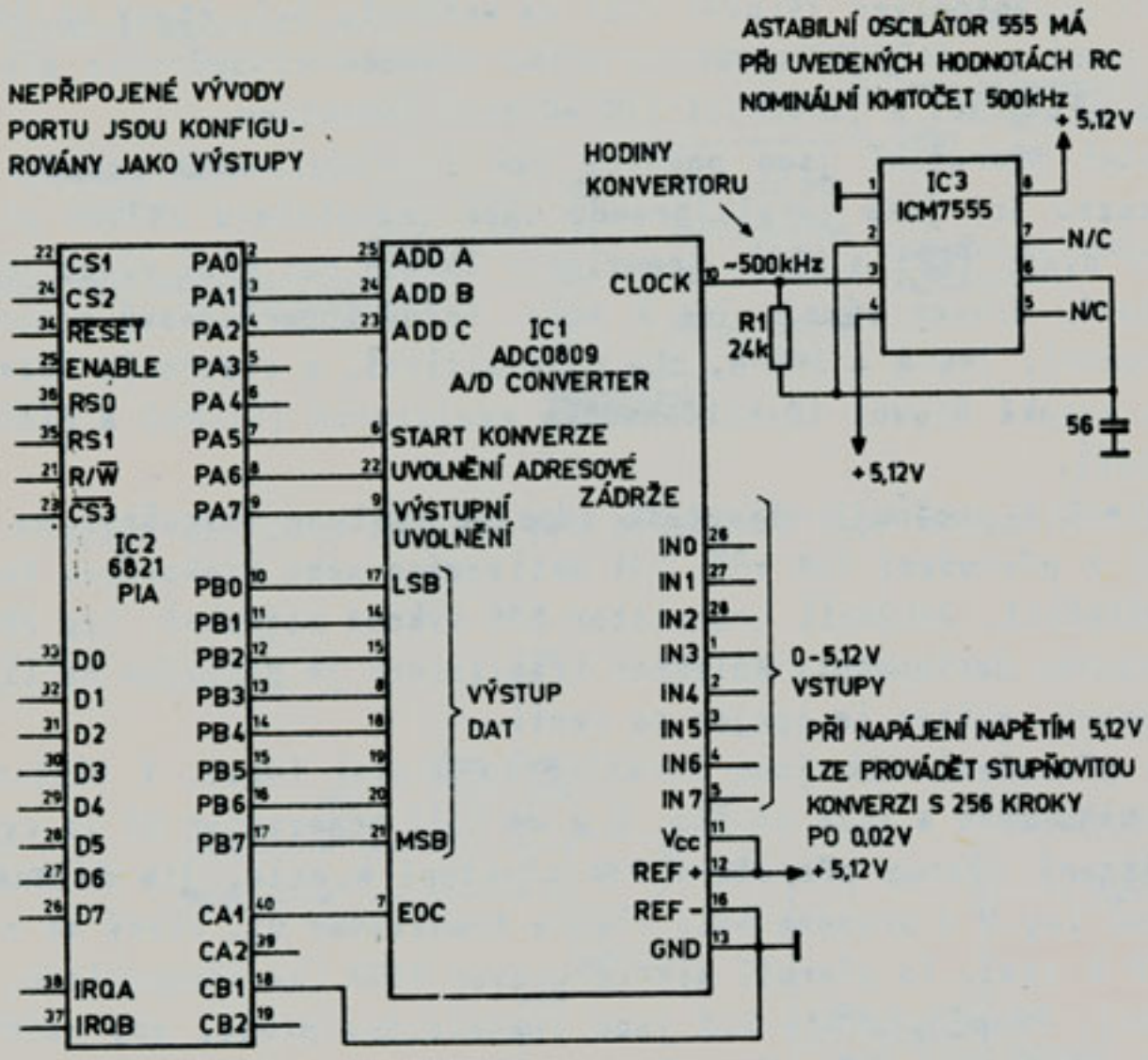


Schéma 4 - A/D konvertor ADC0809 s dvojitým interfejsem PIA

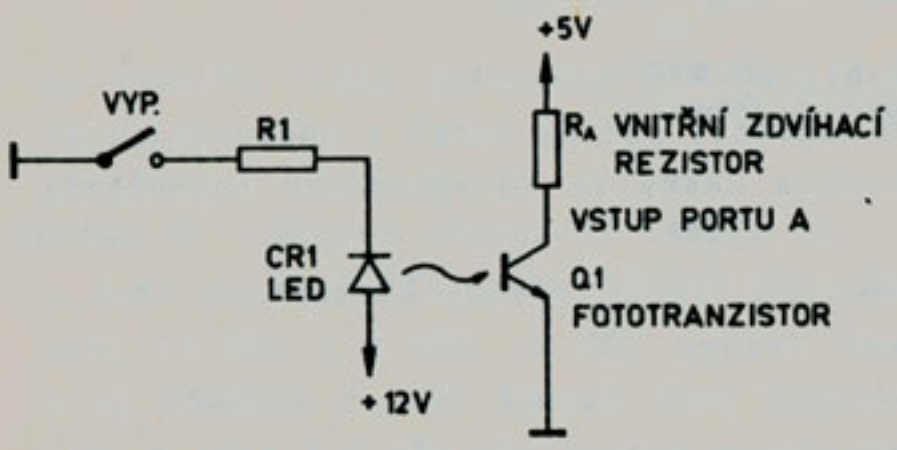


Schéma 5 - Světelná vazba přenosu dat na port A PIA

fototranzistor, osvětlený svou vlastní LED diodou, která je částí poplašného systému proti nežádoucímu vstupu apod.

Pro tento typ interfejsu jistě najdete řadu dalších aplikací. Schopnost dodávat proud, kterou má port B, může být využita při řízení NPN tranzistorů, které pak mohou ovládat relé nebo solenoidy. Speciální reléové ovládače, jako NE5090, mohou být řízeny oběma porty. Zabudované zdvíhací rezistory portu A mohou zajistit, že žádné relé nebude aktivní při uvedení zařízení do chodu.

SHRNUTÍ

Interfejsing v podstatě není žádnou velkou, natož tajemnou vědou. Chce to jen trochu praxe, ostatní pak už "bude přicházet samo". Rozhodně je však dobré se orien-

tovat v tom, jaké dispozice nám které obvody nabízejí. Žádná znalost není nikdy k zahození. Možných obvodových kombinací pro přenos dat je více než dost. Čím více budeme o jednotlivých obvodech vědět, tím méně nás budou naše "broučí" desky trápit. A tím efektivněji budou fungovat.

BYTE 8/86
Přeložil Rad. PLETKA

Paralelní interfejs

(2. část)

V minulém čísle jsme se dostali do fáze, v které jsme schopni přenést jeden znak z počítače do periférie. I nadále budeme jako periférii uvažovat tiskárnu. Na ní lze dobře demonstrovat, jakým způsobem může takový přenos probíhat. V minulé části byl na obr. 8 znázorněn časový průběh přenosu jednoho bajtu ve standardu Centronics. Osm datových vodičů je doplněno třemi řídícími vodiči:

výstupní: STROBE - potvrzuje platnost dat
vstupní : BUSY - informuje o zaměstnanosti periférie
ACK - informuje o tom, že data byla přejata

Signály STROBE a ACK jsou aktivní v log.0, BUSY v log.1.

Rutina pro přenos jednoho bajtu může být napsána např. takto (viz i 1. část):

```
INIT    LD      A,131       ; hodnota CW do akumulátoru
        OUT     (127),A     ; přenos do CWR
        LD      A,9         ; bitová operace (PC4 nastavíme)
        OUT     (127),A     ; STROBE neaktivní
        RET                 ; obvod inicializován
BYTE    LD      C,A         ; úschova v bajtu z reg.A do reg.C
B2      IN      A,(95)      ; čtu z portu C
        BIT     3,A         ; je BUSY? (log.1=ano)
        JR      NZ,B2       ; musím čekat
        LD      A,C         ; bajt zpět do akumulátoru
        OUT     (31),A      ; vyslání na bránu A
        LD      A,8         ; bitová operace (PC4 vynulujeme)
        OUT     (127),A     ; STROBE=log.0 (data platná)
        LD      A,9         ; bitová operace (PC4 nastavíme)
        OUT     (127),A     ; STROBE=log.1 (konec impulsu)
        RET                 ; bajt byl přenesen
```

V první části je rutina INIT, která naprogramuje obvod 8255. Následuje nastavení výstupního vodiče STROBE do neaktivního stavu (neplatná data). Rutina BYTE čeká na připravenost periférie, pak připojí data a na STROBE udělá puls.

Pokud bude naším úkolem vytvořit přijímač, můžeme postupovat analogicky: bránu A obvodu 8255 naprogramujeme jako vstupní, bránu C opět rozdělíme na poloviny (dolní

polovina výstupní, horní vstupní) a program může vypadat následovně (opět bez použití vodiče ACK):

```
INIT    LD      A,152       ; určení režimu
        OUT     (127),A     ; PA,PCh vstup, PB,PCl výstup
        LD      A,7         ; BUSY do log.1 (nepřipraven)
        OUT     (127),A     ; nastavení bitu 3 PC na log.1
        RET                 ; návrat
BYTE-I  LD      A,6         ; vynulování bitu 3 PC (BUSY)
        OUT     (127),A     ; jsme připraveni
B2      IN      A,(95)      ; čtení stavu
        BIT     4,A         ; STROBE?
        JR      NZ,B2       ; ne
        IN      A,(31)      ; data do ACC
        LD      C,A         ; uschování do reg. C
        LD      A,7         ; nastavení bitu 3 PC (BUSY) na log.1
        OUT     (127),A     ; nejsme připraveni
        LD      A,C         ; data do ACC
        RET                 ; návrat
```

Nyní předpokládejme, že je tiskárna připojena a my vytváříme program, který ji má uvést do chodu. Nejprve si musíme uvědomit, jak se procesor dostane k naší rutině.

Zpravidla jsou adresy všech vstupních a výstupních rutin zapsány v tzv. kanálových informací. Zde se mikroprocesor "dozví", kde začíná jaká rutina. Do nich vstupuje většinou s jedním znakem, u systémů pracujících s mikroprocesory 8080, 8085, Z80 bývá tento znak zpravidla uložen v registru A. Výstup může probíhat následovně:

1) do reg.A uložíme znak určený pro výstup a skočíme na hlavní výstupní rutinu (společná pro všechny periférie),
2) v kanálových informacích procesor najde startovací adresu rutiny
3) ...a skočí na ni.

Před vlastním vysíláním nebo přijímáním dat musí procesor ještě určit, který kanál bude používat.

U ZX Spectra jsou kanálové informace uloženy v paměti RAM (sem jsou přeneseny z ROM při inicializaci systému počítače). První adresu jejich uložení obsahuje systémová proměnná CHANS (viz i článek Kanály ZX Spectra ve zpravodaji Mikrobáze č. 4). Informace jsou organizovány v pěticích bajtů, kde 1.+2. udávají adresu začátku výstupní rutiny tohoto kanálu, 3.+4. udávají konec a bajt poslední je jménem kanálu - S=obrazovka (screen), K=editační zóna (keyboard), R=pracovní prostor a P=tiskárna (printer). Pokud by vstupní rutina kanálu neměla smysl (např. u tiskárny), zamíří bajty 3.+4. na adresu chybové rutiny (objeví se chybové hlášení).

Kanály jsou v pořadí K, S, R a P. Po připojení některých periférií - jako je microdrive nebo disková jednotka - se po jejich inicializaci přidávají ještě další kanály s jinými jmény (některé mají i větší délku). Kanály K, S, R, P jsou ale vždy umístěny od začátku kanálových informací.

Pokud chceme informaci pro tiskárnu změnit a namířit adresu výstupní rutiny kanálu 'P' na naši vlastní rutinu, stačí zadat:

```
POKE (PEEK 23631+256*PEEK 23632+15),ADR
```

kde ADR je vstupní adresa naší rutiny (tomu odpovídá druhá část rutiny INIT-P).

A nyní vlastní program pro výstup na tiskárnu. Nejprve bychom si měli ujasnit, jaké znaky nám může procesor poslat. Pro konkrétní příklad použiji opět ZX Spectrum:

a) kódy menší než 32	**řídící kódy**
b) kódy 32 až 127	**abecední znaky a symboly**
c) kódy 128 až 143	**grafické znaky**
d) kódy 144 až 164	**uživatelská grafika**
e) kódy 165 až 255	**klíčová slova**

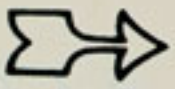
Vlastní program bude rozdělen na bloky a) až e), stejně jako jsou děleny kódy přicházející do naší rutiny.

Pokud jde o tiskárnu, budu se orientovat na EPSON FX-80 (a všechny s ní kompatibilní), která se ve své třídě stává světovým standardem. Tato tiskárna umí (kromě jiného) tisk znaků s kódy 32 až 126 (127 je v ASCII DELete), umí CR (návrat vozíku), LF (posun válce), umí podtrhávat znaky současně při tisku, umí TABelovat a také má možnost grafického výstupu (parametrem je počet bajtů, který bude následovat). To je vše, co budeme potřebovat.

Příkazy se tiskárně předávají jako posloupnost bajtů, stejně jako data pro tisk.

A nyní si zapíšeme celý program, který používá při běhu dvě systémové proměnné COL, COLMAX a jeden bajt jako FLAG (jsou popsány přímo v programu):

```
INIT-P  LD      A,131       ; hodnota CW do akumulátoru
        OUT     (127),A     ; přenos do CWR
        LD      A,9         ; bitová operace (PC4 na log.1)
        OUT     (127),A     ; STROBE neaktivní

        LD      HL,(23631)  ; systémová proměnná CHANS
        LD      DE,15       ; CHANS+15 je 'P' kanál
        ADD     HL,DE       ;
        LD      DE,PRINT    ;nový obsah = adresa naší rutiny
        LD      (HL),E      ;
        INC     HL          ;
        LD      (HL),D      ; nový obsah v kanálových informacích

        LD      (IY+69),80  ; COLMAX (šířka válce) na 80 znaků

        LD      (IY+70),0   ; bit 0 funguje jako flag při tisku
                            ; CHR (23)  (1=čekám na parametr)
        CALL    OUTSTR      ; tiskárnu nastavíme na širší válec
        DEFB    27, '0'     ; a jeho konec si budeme hlídat sami
        DEFB    200,255     ;
        RET                 ; návrat
```



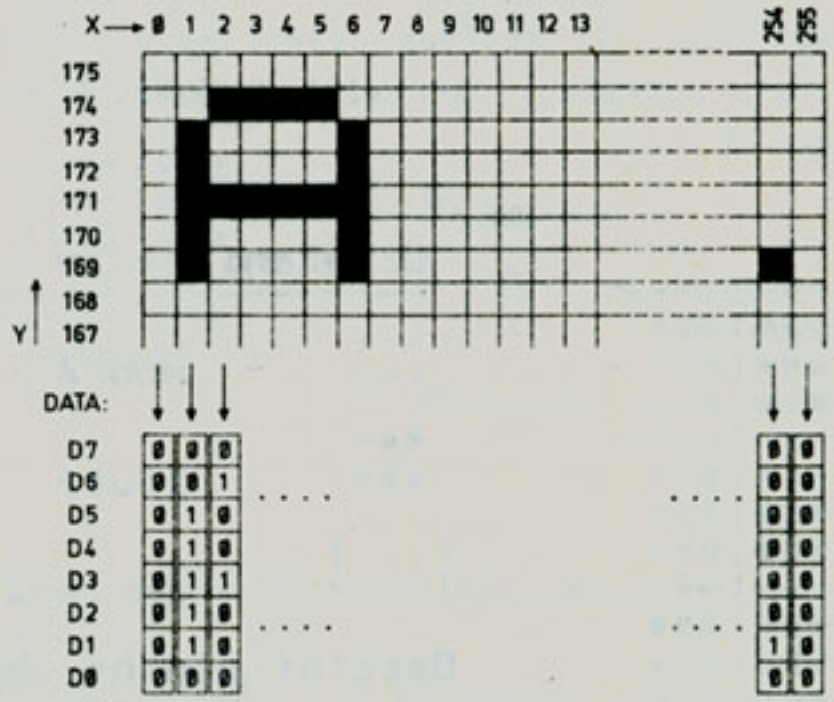
```
PRINT   LD      HL,COL      ; po celou dobu tisku bude v HL ukazatel na proměnnou
                            ; (současná pozice)
        BIT     0,(IY+70)   ; je to parametr pro tabelátor ?
        JR      NZ,TABEX    ; ano, jdi udělat TAB A

; testování kódů menších než 32-blok a)
        CP      6           ; posun válce o polovinu ?
        JR      NZ,P1       ; když ne, skok dále
        LD      B,(IY+69)   ; max. šířka válce
        SRL     B           ; (max.š.válce)/2
        LD      A,(HL)      ; aktuální pozice hlavy
        CP      B           ; jsme před polovinou
        JR      NC,LFOUT    ; když ne, skok na novou řádku
        JR      CHR23       ; TAB ((max.š.válce)/2)

P1      CP      13          ; je to kód CR?
        JR      Z,LFOUT     ; když ano, potom skok na novou řádku
        CP      23          ; je to TAB?
        JR      NZ,P2       ; když ne, pokračuj
        SET     1,(IY+70)   ; nastav 'čekám na parametr'
        RET                 ; návrat

P2      CP      26          ; tento kód zvolen pro COPY
        JR      Z,COPY      ; když je to COPY, pak skok
        CP      32          ; je kód menší než 32?
        RET     C           ; když ano, návrat

; testování 32..127-blok b)

        CP      127         ; je to (c)?
        JR      NZ,P3       ; když ne, skok dále
        LD      A, "C"      ; ano, nahradíme jej písmenem 'c'

P3      CP      NC,P4       ; když ano, skok vpřed

PRNT    CALL    OUTCHA      ; pro interval 32..126 rovnou tisk
        LD      A, (COLMAX) ; max.š.válce do akumulátoru
        INC     (HL)        ; inkrementuj COL
        CP      (HL)        ; jsme na hodnotě stejné s COLMAX?
        RET     NZ          ; když ne, návrat
        JR      LFOUT       ; jsme na konci, pak návrat

; testování kódů 128..143-blok c)
P4      SUB     144         ; jsme v intervalu 128..143?
        JR      C,P5        ; když ano, skok

; testování kódů 143..164-blok d) a 165..255-blok e)
        CP      21          ; jsme v U.D.G. ?
        JR      C,P6        ; když ano, skok na P6
```

```
        SUB     21          ; posun (165-->0,....,255- ->90)
        JP      #0C10       ; skok na rutinu PO-TOKENS, která klíčové slovo roz-
                            ; dělí na jednotlivé znaky (a opět volá rutinu PRINT)
P5      LD      A,"?"       ; znaky 128..143 tisknout nebudeme
        JR      PRINT       ; a nahradíme je třeba otazníkem

; znaky U. D. G. budeme tisknout jako A..U, ale podtržené
P6      SUB     79          ; znaky U. D. G. posuneme o 79 dolů
        PUSH    AF          ; tzn. gr. 'A' --> 'A'
        CALL    OUTSTR      ;      gr. 'U' --> 'U'
        DEFB    27,"-"      ; zapneme podtrhávání
        DEFB    1,255       ;
        POP     AF          ; původní znak zpět do akumulátoru
        CALL    PRNT        ; znak vytiskneme podtržený
        CALL    OUTSTR      ; zrušíme podtrhávání
        DEFB    27,45       ;
        DEFB    0,255       ;
        RET                 ; návrat

CHR23   LD      (HL),A      ; rutina pro TABelátor
        LD      (LBL1),A    ; parametry pro OUTSTR zmodifikujeme
        CALL    OUTSTR      ; (to lze jen v RAM) a pošleme tiskárně příkaz TAB A
        DEFB    27,68       ;
LBL1    DEFB    0,0         ;
        DEFB    9,255       ;
        RES     0,(IY+70)   ; zrušíme vlajku 'čekám na parametr'
        RET                 ; návrat

LFOUT   XOR     A           ; rutina 'posun válce'
        LD      (HL),A      ; COL inicializujeme
        LD      A,10        ; kód LF (Line Feed)

;  vyslání znaku do tiskárny
OUTCHA  PUSH    BC          ; uschování reg. BC do zásobníku
        LD      C,A         ; přenos bajtu z reg.C do reg.A
01      IN      A,(95)      ; čtu z portu C
        BIT     3,A         ; je BUSY? log.1=ano
        JR      Z,02        ; když ne, skok na vysílání znaku
        CALL    #1F54       ; test klávesy BREAK
        JR      C,01        ;   jediná možnost zastavení, pokud by
        RST     #08         ;   nepřišel signál o připravenosti
        DEFB    #0C         ; při BREAKu skok na Error-rutinu
                            ;   jinak znovu na 01
02      LD      A,C         ; bajt zpět do akumulátoru
        OUT     (31),A      : vyslání na bránu A
        LD      A,8         ; bitová operace (PC4 vynulujeme)
```

```
          OUT     (127),A     ; STROBE = log. 0 (data platná)
          LD      A,9         ; bitová operace (PC4 nastavíme)
          OUT     (127),1     ; STROBE = log.1 (konec impulsu)
          POP     BC
          RET                 ; bajt byl přenesen

; rutina pro vyslání série bajtů za CALL OUTSTR
   OUTSTR EX      (SP),HL     ; ukazatel na bajt
          LD      A,(HL)      ; bajt do akumulátoru
          INC     HL          ; inkrementace HL
          EX      (SP),HL     ; (SP)< -- >HL
          CP      255         ; je to poslední bajt
          RET     Z           ; když ano, pak návrat
          CALL    OUTCHA      ; když ne, tisk
          JR      OUTSTR      ;    a skok na OUTSTR

   ; rutina pro tisk kopie obrazovky ZX Spectra
   COPY   LD      BC,#AF00    ; nastavení čítačů Y=B=vertikální
   C1     CALL    OUTSTR      ;                   X=C=horizontální
          DEFB    27? 'J',24  ; nastavení menšího posuvu válce
          DEFB    13          ; vyslání CR
          DEFB    27,'K',0    ; zapnutí grafického režimu:
          DEFB    1,255       ;    budu vysílat 256 bajtů (1 řádek)
   C2     PUSH    BC          ; uschování čítačů
          LD      PE,#0B00    ; D=čítač linek ve znaku, v E generuji
                              ;   bajt (musím vysílat svislé bajty,
                              ;   horní bit má největší váhu, obr.1)
C3        PUSH    BC          ; znovu uschování čítačů
          CALL    #22AA       ; rutina pro výpočet adresy bajtu
                              ;   s pozicí C,B
          LD      B,A         ; v HL vrací adresu, v A pozici
          INC     B           ; pozice do B
          LD      A,(HL)      ; do akumulátoru žádaný bajt
C4        RLA                 ; B*posun
          DJNZ    C4          ; opakuj B*
          RL      E           ; posun bodu přes CY do E
          POP     BC          ; inicial. souřadnic
          DEC     B           ; zmenšení o jednu linku
          DEC     D           ; zmenšení čítače linek
          JR      NZ,C3       ; ještě jich nebylo 8, skok na C3
          LD      A,E         ; vygenerovaný bajt do akumulátoru
          CALL    OUTCHA      ; a přenos do tiskárny
          POP     BC          ; inicializace souřadnic
          INC     C           ; inkrementace souřadnice Y
          JR      NZ,CZ       ; ještě nejsme na konci, skok na C2
          LD      A,B         ; dekrementace horizontálního čítače
```

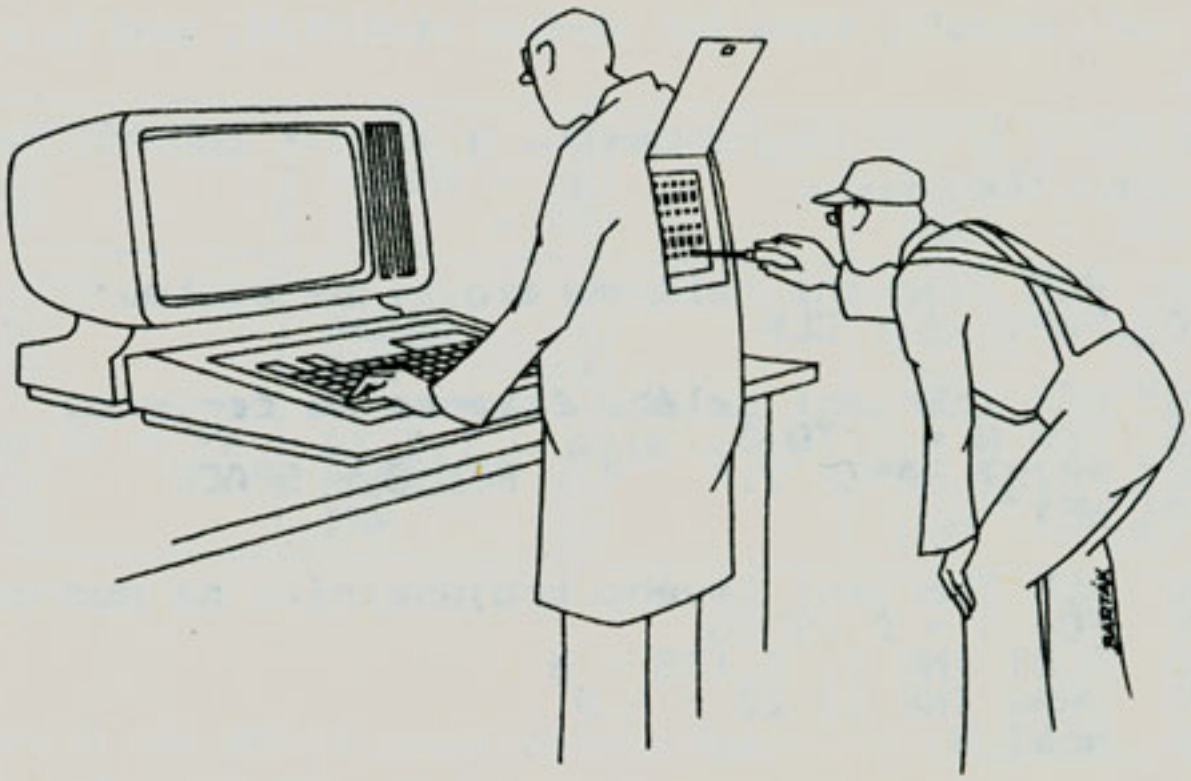
```
        SUB     8           ;   o jeden řádek (8linek)
        LD      B,A         ;
        JR      NC,C1       ; nejsme-li dole, pak skok na C1
        JR      LFOUT       ; na konci ještě LF

COL     EQU     23680       ; proměnná udávající pozici tisku
COLMAX  EQU     23679       ; maximální šířka válce
FLAG    EQU     23681       ; vlajka pro tisk CHR (23)
```

To je tedy stručný program pro obsluhu tiskárny počítačem pomocí obvodu 8255. Program pouze demonstruje, jak může vypadat taková jeho základní kostra. Po menších modifikacích jej lze používat i s jiným hardwarovým zapojením nebo i s jinou tiskárnou.



Obr. 1 Způsob generace bajtů v příkazu COPY

Richard LUKEŠ



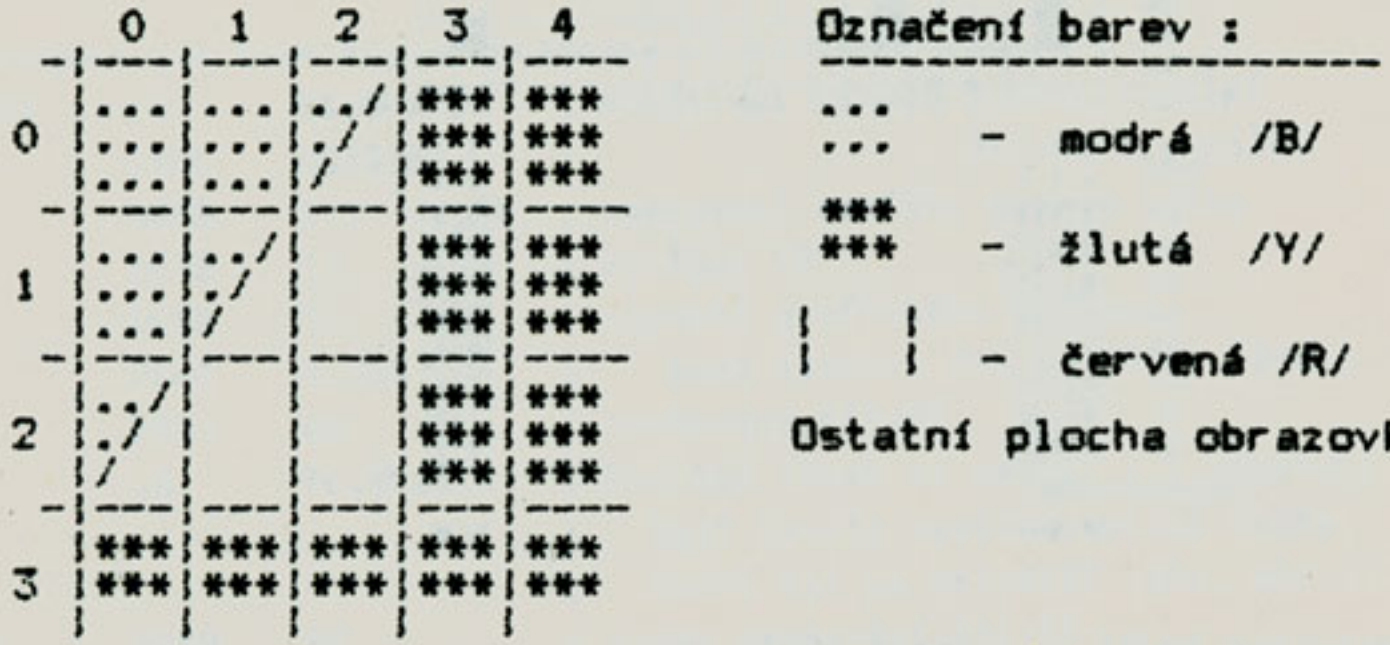
Informační sběrnice

Obrazová paměť ZX Spectra

(4. část)

Kresba šikmé barevné plochy

Dostáváme se k tomu, co jsem slíbil - jak nakreslit šikmou barevnou plochu, aby na ní nebyly schody vytvořené atributovými čtverci. Ten nejběžnější způsob, který používají i profesionální firmy, si nejlépe osvětlíme na příkladu. Dejme tomu, že v levém horním rohu obrazovky potřebuji nakreslit obrázek, který by vypadal přibližně takhle:



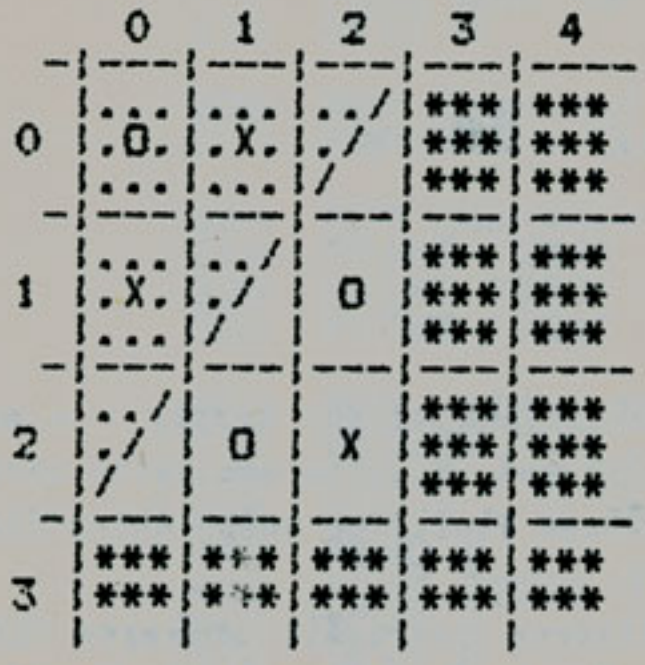
Trochu jsme si to zjednodušili - jednak jsme obrazec umístili do levého horního rohu obrazovky a jednak ona šikmá dělící čára jde přesně po úhlopříčce - ale pro demonstraci to zcela postačí. Kdyby byl obrazec složitější, méně by vynikl samotný princip kreslení.

Nakreslit takovouto plochu je jednoduché. Je několik způsobů, jak to udělat. Jeden z nich může být třeba tento:

```
100  REM Zbarvení celé obrazovky na žlutou
110  PAPER 6 : CLS

120  REM Zbarvení celého čtverce na červenou
130  FOR N = 1 TO 3
140  PRINT PAPER 2;"    " : REM 3 x SPACE
150  NEXT N

160  REM Zbarvení levého trojúhelníku na modrou
170  FOR N = 0 TO 23
180  PLOT INK 1; 0,175 - N
190  DRAW INK 1; 23 - N,0
200  NEXT N
```

Vidíme, že roli dělící přímky v šikmých barevných plochách zde na sebe vzala hranice mezi barvou papíru a inkoustu.

Potíž s kresbou nastane, když obrazec, který potřebujeme nakreslit, bude modifikován takto:



Označení barev :

... - modrá /B/

*** - žlutá /Y/

| | - červená /R/

Písmena :

0 - bílá /W/
X - zelená /G/

Ostatní plocha obrazovky žlutá

Teď už jde skutečně o poněkud bizarní obrazec; ale to mi musíte odpustit, demonstračně se nám velice hodí.

Kdybychom použili metodu kreslení z předchozí ukázky, nemohli bychom nakreslit tu část obrazce, která je vlevo nad úhlopříčkou, protože tam se nám kříží požadavky na barvu inkoustu a papíru. Při konstrukci takového obrazce již musíme uvažovat nad každým znakovým čtvercem samostatně. Abychom se v nich vyznali, zůstaneme u našeho osvědčeného číslování - budeme je značit, jako bychom je chtěli vytisknout pomocí povelu PRINT AT. Použijeme tedy atributovou tabulku a zapíšeme si, jaké atributy budeme v jednotlivých čtvercích potřebovat:

AT 0,0 - ATTR = 15 PAPER - /B/ INK - /W/	AT 0,1 - ATTR = 12 PAPER - /B/ INK - /G/	AT 0,2 - ATTR = ?? PAPER - /?/ INK - /?/
AT 1,0 - ATTR = 12 PAPER - /B/ INK - /W/	AT 1,1 - ATTR = ?? PAPER - /?/ INK - /?/	AT 1,2 - ATTR = 23 PAPER - /R/ INK - /W/
AT 2,0 - ATTR = ?? PAPER - /?/ INK - /?/	AT 2,1 - ATTR = 23 PAPER - /R/ INK - /W/	AT 2,2 - ATTR = 20 PAPER - /R/ INK - /G/

Vidíme, že potíže nám dělají právě znakové čtverce v úhlopříčce, které by měly být v jedné polovině modré (B) a ve druhé červené (R). Protože jediná možnost, jak kombinovat dvě barvy v jednom znakovém čtverci, je kombinace papír - inkoust, musíme

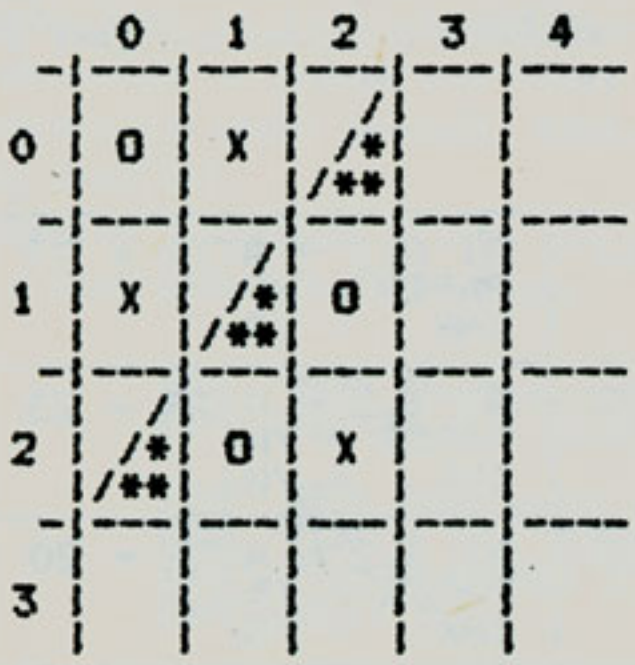
úhlopříčné rozdělení zajistit přes tisk obrazce tak, aby kresba vytvářela onen požadovaný obrazec, neboli ono požadované šikmé rozdělení:

Teď bude stačit, aby papír měl barvu jedné požadované strany kresby - v našem případě modrou (B) a inkoust barvu druhé - tedy červené (R). Tím nám vyšlo atributové číslo 10. Takto vyrobený trojúhelníček teď můžeme vložit do oněch úhlopříčných čtverců s tiskovými posicemi AT 0,2; AT 1,1; AT 2,0.

Příslušný trojúhelníček můžeme do znakového čtverce zakreslit příkazem PLOT - třikrát na různých místech. Nebo můžeme téhož výsledku dosáhnout poněkud elegantněji tak, že si tento obrazec nadefinujeme jako vlastní USR znak a budeme jej tisknout přímo na požadované místo v grafickém modu.

Jak se to dělá, je detailně a dobře popsáno v manuálu ke SPECTRU - tak budeme mlčky předpokládat, že jsme si tento trojúhelníček nadefinovali jako grafické "A".

Můžeme tedy vytisknout obrazec, který bude (bez stanovení atributů - ty zatím necháme na základní hodnotě 56) vypadat asi takto:



Označení barev :

***
*** - inkoust

| | |
| | | - papír

Písmena :

O - inkoust
X - inkoust

Ostatní plocha obrazovky - papír

Do takovéhoto obrazce můžeme klidně vepsat atributy podle předchozí tabulky; samozřejmě s tím, že úhlopříčná pole budou mít atribut 10; a dostaneme původně požadovaný obrazec.

Praktická realizace programu, který by to splnil, by mohla vypadat třeba takto:
("A" na řádcích 310 - 330 je znak A v grafickém modu
** na řádku 120 je druhá mocnina)

```
100  REM Definování USR znaku A
110  FOR N = 0 TO 7
120  POKE USR "A" + N,2 ** ( N + 1 ) - 1
130  NEXT N

200  REM Zbarvení obrazovky na žlutou
210  PAPER 6 : CLS

300  REM Nakreslení základní kresby
310  PRINT "OXA"
320  PRINT "XAO"
330  PRINT "AOX"

400  REM Určení atributů kresby
410  RESORE 500
420  FOR M = 0 TO 64 STEP 32
430  FOR N = 0 TO 2
440  READ B
450  POKE 22528 + N + M , B
460  NEXT N
470  NEXT M

500  REM Tabulka hodnot atributů
510  DATA 15,12,10
520  DATA 12,10,23
530  DATA 10,23,20
```

Program snad není nutno nijak zvlášť popisovat, je dost jednoduchý a jednotlivé poznámky jej jednoznačně rozčleňují. Znovu podotýkám, že to, co jsme uvedli, není ani jediné, ani nejkratší řešení daného úkolu.

Tím jsme probrali stručnou ukázku v jazyce BASIC, takže nám zbývá demonstrační ukázka ve strojovém kódu.

Kresba pomocí strojového kódu

Zvolíme opět nějaký jednoduchý příklad. Dejme tomu, že pro zobrazení výsledků nějakého programu potřebujeme tisknout žlutě na modrém pozadí. Současně chceme, aby naše výsledková tabule byla orámována jednoduchou tenkou linkou. Výsledky budeme tisknout na všech 24 řádcích, tedy i v editační zóně - v ní Basicem však rámeček neuděláme.

Nezbývá než jít na to jinak. Nabízí se sice řešení, pomocí nějž bychom mohli vše, co by šlo, udělat Basicem a zbytek dokreslit a dobarvit strojovým kódem. Ale z demonstračních důvodů se rozhodneme, že tento postup není "čistý" a celý obrazec uděláme strojovým kódem.

Nejdříve malý rozbor; úkol rozdělíme na dva samostatné celky:

1 - provedení kresby rámečku
ta se opět rozdělí na čtyři samostatné části:
a - horní vodorovná
b - levá svislá
c - pravá svislá
d - dolní vodorovná

2 - změna atributů

Protože už víme jak na to, bude samotný rozbor úlohy poměrně jednoduchý. Nejdříve kresba:

Horní vodorovná čára se nakreslí tak, že do nultého bodového řádku zapíšeme do všech lokací 255, protože to je všech osm bodů, které budou postupně tvořit linku rámečku. První adresa je 16384 a ve vodorovné lince je takovýchto adres přesně tolik, kolik je znakových čtverců v jednom znakovém řádku, tedy 32. Níže najdete výpis strojového kódu a tam je tato část zapsána pod řádky 270 až 330.

Dolní vodorovná čára je typově stejná - jen těch 255 budeme zapisovat do posledního bodového řádku, t.j. do řádku, kterým obrazovka končí. Její adresa bude:

konec obrazové paměti - 32 (to je 22528 - 32 = 22496)

Počet lokací paměti je stejný, tedy 32. V programové části je zapsána na řádcích 30 až 80.

Svislá levá linka je poněkud komplikovanější, ale ne o moc. Budeme zapisovat takové číslo, které v bitu, jenž je zcela vlevo - tedy v bitu 7, bude mít zapsáno 1. Ve skutečnosti je to číslo 128, ale to nás "jako" nezajímá. První lokace, na které toto číslo bude zapsáno, je začátek obrazovky, tedy 16384. Další lokace bude o 32 větší - přeskok zbytku řádku na začátek dalšího řádku - to budeme (třeba na přeskáčku) opakovat celkem tolikrát, kolik je bodových řádků, tedy 192. V tomto případě vůbec nezáleží na tom, aby se body kreslily popořádku. Důležité je pouze to, aby se nakreslily všechny. Rychlost kreslení je tak velká, že se vůbec nedá postřehnout, takže z hlediska estetického o nic nepřijdeme.

Tak se nám ovšem přemažou již nakreslené čárky na horní i dolní lince. Je samozřejmě jedno, zda zkrátíme svislou či vodorovnou linku. V programu je to uděláno obojím způsobem - nejdřív nakreslíme dolní vodorovnou, potom obě boční od horní linky (o jeden bod zkrácené tak, aby nepřemázly dolní linku) a nakonec linku horní, která může přemazat obě boční. Takže délka svislé bude o jeden bod kratší, čili 191. V programu je tato část zapsána na řádcích 100 až 170.

Svislá pravá je stejná, jen počáteční adresu má o 31 větší. Vše ostatní, včetně délky, je shodné. Tato část je zapsána na řádcích 200 až 260.

Nakonec zbývá přebarvit atributy. Požadavkem je žluté písmo (Y) na modrém pozadí (B) - podle tabulky atributů má atributové číslo hodnotu 14. Toto číslo potřebujeme zapsat do celé paměti atributů, t.zn. od adresy 22528, v délce 768 - právě tolik je atributových čtverců. Tato poslední část programu je zapsána na řádcích 350 až 400.

Máme shromážděny všechny požadavky a můžeme psát vlastní assemblerový program:

```
 10          ORG   60000       ; adresa začátku
 20 ; vodorovná dolní
 30          ld    hl,22496    ; startovací  adresa
 40          ld    de,22497    ; cílová adresa
 50          ld    bc,31       ; počítadlo
 60          ld    a, 255      ; co se zapisuje
 70          ld    (22496),a   ; zápis do první lokace
 80          ldir              ; přepis celého bloku
 90 ; svislá levá
100          ld    de,32       ; velikost skoku
120          ld    b,191       ; počítadlo
130          ld    hl,16384    ; startovací adresa
140 skok1    set   7,(hl)      ; zápis bitu do lokace
150          add   hl,de       ; adresový posun
160          dec   b           ; zmenšení počítadla
170          jr    nz,skok1    ; není-li počítadlo na
180                            ; nule - skoč na návěstí
190 ; svislá pravá
200          ld    b,191       ; počítadlo
210          ld    hl,16415    ; startovací adresa
220 skok2    set   0,(hl)      ; zápis bitu do lokace
230          add   hl,de       ; adresový posun
240          dec   b           ; zmenšení počítadla
250          jr    nz,skok2    ; není-li počítadlo na
260                            ; nule - skoč na návěstí
270 ; vodorovná horní
280          ld    hl,16384    ; startovací  adresa
290          ld    de,16385    ; cílová adresa
300          ld    bc,31       ; počítadlo
310          ld    a, 255      ; co se zapisuje
320          ld    (16384),a   ; zápis do první lokace
330          ldir              ; přepis celého bloku
340 ; zápis attributů
350          ld    hl,22528    ; startovací  adresa
360          ld    de,22529    ; cílová adresa
370          ld    bc,768      ; počítadlo
380          ld    a, 14       ; zápis attribut. čísla
390          ld    (22528),a   ; zápis do první lokace
400          ldir              ; přepis celého bloku
410          ret               ; návrat do BASIC
```

Začátek tohoto programu byl (nahodile) dán na adresu 60000. Program má tu výhodu, že je relokovatelný - je možno jej bez úprav. umístit kamkoli do paměti beze změny jeho funkcí. Navíc nesmaže obrazovku (ponechá původní kresbu); proto byl použit příkaz SET na řádcích 140 a 220. Program se spouští od startovní adresy:

RANDOMIZE USR 60000

Nezbývá než opět dodat, že toto není ani jediné, ani nejkratší řešení zadaného úkolu. Ale zde nešlo o žádnou minimalizaci.

V článku, jako je tento, samozřejmě nelze zdaleka probrat všechno. Autor musí vždy zjednodušovat a v lecčem ustupovat, aby se do článku aspoň část problematiky vůbec vešla. Doufám, že všem, kterým uvedené informace přišly vhod, pomohly nejen pochopit organizaci obrazovky, ale poskytly i podněty k vlastním experimentům.

Jiří POBŘÍSLO

Datalog

(Recenze původního programu Mikrobáze)

Konečně! Konečně je na světě databanka, která umí česky, slovensky i mnohak jinak -sky a cky! Ale to není zdaleka vše, co nového přináší uživatelům (zatím jen) Specter.

Když se objeví nová variace na určité téma, přirozeně nutí ke srovnání s variacemi předchozími. V případě tématu databázového jsou to zahraniční programy VU-FILE a MASTER FILE pro Spectrum.

První radostný rozdíl je v tom, že DATALOG vznikl v naší domovině. Je originálem, žádnou záplatovanou odvozeninou. Další potěchu a velkou úlevu přinese uživatelům DATALOGu způsob jeho komunikace s obsluhou. V zahraničí se tato komunikace nazývá "menu-driven". Vyjadřuje tolik, že všechny funkce programu volíme posuvem kurzoru po srozumitelných názvech funkcí zobrazovaných v různých menu, jimiž podle potřeby prostupujeme. Tedy žádné zadávání číselných parametrů řádek a sloupců obrazovky, žádné (obvyklou sklerózou narušované) zapamatovávání si "anonymních" funkcí několika desítek tlačítek! Obsluha DATALOGu jde jako na drátku.

Vzpomínám si, jak dlouho mi trvalo, než jsem se doslova propracoval manuálem programu MASTER FILE, a jak plynuly hodiny při návrhu struktury zapisovaných dat (zpráv a položek) a jejich výpisu na obrazovku. Jak jsem chtěl leckdy udělat nový návrh pro novou aplikaci, ale když jsem si představil, co mne čeká, radši jsem to vzdal. Znám několik lidí, kteří to vzdali vůbec a zůstali při stařičké databázi VU-FILE, přestože má ve srovnání s MASTER FILEm četné neduhy, byť je svižnější.

Když mi v únoru autor DATALOGu přinesl k posouzení svůj produkt i s manuálem, říkal jsem si; "Jen počkej, Petře Adámku, teď uvidíš, zač je toho loket!" Schválně jsem se do manuálu ani nepodíval a rovnou jsem začal operovat s programem. Když má sloužit všem, tedy i nezkušeným uživatelům, tak žádný manuál přece není podmínkou zdárné obsluhy!

Po načtení programu se objevilo první menu. Po pár vteřinách oťukávání klávesnice jsem pochopil, čím se jak hýbe a "pálí". "No, tak tohle ti vyšlo, uvidíme dál." Nadefinoval jsem pár zpráv a postoupil do dalšího menu. Aha, tak odsud se vkládají názvy položek. Tak jich tam dám třeba deset, ať to stojí zato. Je libo některou opravit, přidat, vymazat? Nic snažšího! Ťuk, ťuk a je to. "Ty kluku jeden lišácká! Ale teď už s tebou bude ámen, protože jdu na definici struktury a výpisu dat!" Ťuk, ťuk... a je tu další menu s pár nápisy. Po dvou pokusech se strefuji do černého. Další menu. Tak kam dám položku Jméno? Třeba doprostřed obrazovky. Tři ťuky a je tam. Následují další položky, všech deset. Šup z jednoho menu do druhého a zpět a do třetího a zpět... Vše se velmi jasně zobrazuje, orientace okamžitá. Kteroukoli definici lze kdykoli jakkoli opravit, zrušit, přidat. Ihned vybarvit, zjasnit, zvolit znaky v módu výpisu 64 či 32 na řádce v kterékoli položce. Je libo, aby byly položky zjasněny v celém jejich předdefinovaném poli, či jen v té části, kde jsou znaky? Ťuk, ťuk a hotovo. Kurzor lítá rychlostí blesku v názvech menu a při návrhu výpisu po celé obrazovce. Nastavím kurzor, "odpálím" jej... a už je zase jedna definice hotová. Chci ji opravit? Nastavím kurzor, "bum"...a mám ji přesně

tam, kde chci, a stará definice je fuč. A co takhle grafika? Příslušné menu tvrdí, že umí úsečky a...okna! A skutečně - kurzor ve tvaru grafického bodu mi maluje úsečku z kurzorem definovaného výchozího do konečného bodu. A nejen svisle či horizontálně. Kterýmkoli směrem. Při malování se úsečka hezky "natahuje", abych průběžně viděl, jak šlechtí výpis položek. Volím tvorbu okna. Nastavím výchozí pozici a "kurzoruji". Okno se hezky rozevírá a průběžně ukazuje svůj tvar. Začínám ztrácet punc sveřepého kritika. Na jeho místo nastupuje dobrý pocit.

Přeskočím do menu zápisu dat. A už tam bez problému sypu óěščřžýáíé atd. Vida, jak to česky vypadá hned jinak, než "anglicko-česky" psaná Alzbeta Novakova či Staromestske namesti. Zkouším funkci abecedního řazení podle zvolené položky. Obrazovka se vymazává a roste na ní řada malinkých čtverečků. Aha, autor si posloužil obrazovou pamětí jako bufferem pro tuto operaci. Aspoň vidím, že řazení probíhá. Prohlížím seřazená data - slova začínající písmenem ř jsou skutečně za r. To budou mít třeba takoví botanici radost, že konečně budou mít řeřichu za růží, zelináři česnek za cibulí atd. atp. Řadit mohu i podle čísel. Podobně - podle mnou zadaného řetězce číselného či znakového - mohu celý soubor dat prohledávat (větší či menší než, roven tomu či onomu), a tak si vytvářet soubor vybraných dat.

Proskakuji pár menu do toho, v němž si volím možnost zápisu dat na vnější paměťové médium. Je tu microdrive a magnetofon. Mám wafadrive, proto volím vstup do Basicu...třeba tam přímo budou příkazy pro komunikaci. Jsou! Přepisuji syntax microdrivu na Wafadrive (lze přepsat prakticky na cokoli) a vracím se zpět do menu. Vidím, že v něm mohu zapsat i doplňující informaci k názvu souboru dat. Dobrý nápad, těch obvyklých pouhých 10 znaků názvu často, přečasto, nestačí pro orientaci v záplavě souborů všeho druhu. Vykonám potřebné (pár ťuknutí), připravím kazetku pro zápis souboru a stisknu tlačítko. A ejhle - ono se vynořilo ještě jedno malinké "menýčko"! Ptá se, zda chci nahrát jen vybraná data, nebo všechna. Začínám tušit, co to znamená. Zkouším vybraná. A hned nato na Wafadrive ještě "natáčím" všechna. Následuje RESET počítače a načtení "čistého" DATALOGU. Do něj ihned zkouším načíst vybraná data. Moje tušení se nemíjí se skutečností - vybraná data vytvořila samostatný, ihned přístupný soubor dat. Napadá mne připojit k nim předtím nahraná všechna data. Skutečně se připojila. Tak tam mám sice vybrané dvakrát, ale to proto, že jiná jsem v ten moment po ruce neměl. DATALOG má tedy další, a to velmi významnou schopnost - dělení a spojování souborů podle výběru, který si uživatel sám stanoví!!! Sbohem VU-FILE a MASTER-FILE! Mám chuť jít si k sousedům vypůjčit klobouk, abych jej mohl před DATALOGem smeknout. V duchu se Petrovi Adámkovi omlouvám - není to "žádnej kluk lišácká", on prostě umí.

Zkusil jsem si "stopnout", jak dlouho mi (napodruhé) bude trvat založení databanky, která bude obsahovat 3 položky (jméno, adresu, telefon). I s grafikou to bylo 12 minut. Pokud si vzpomínám na podobný úkon u MASTER FILU, tam si zabral několik hodin.

Ale aby, k naší újmě, autora nepostihl pád z přílišné pýchy, pár výhrad:

Těm, kteří nepíší rychlostí korespondentek, bude programátorem zpracovaný test stisknutých tlačítek vyhovovat. Mně ne. Jde o to, že stisknu-li ihned po tlačítku třeba T tlačítko třeba H, zobrazí se místo H znova znak T. Což mne nutí zpomalit. A to mi vadí.

Při opravách zapsaných dat (znaků) v položkách a měnitelných názvech v menu musím kurzorem vymazat nechtěné znaky, protože DATALOG je při zápisu znaků stále v módu jejich vkládání (INSERT). Uvítal bych možnost volby módu přepisu. (OVERWRITE). Mazání funkcí DELETE zdržuje. Znaky, které by po přepisu zbyly, bych radši mazal mezerníkem (jako ve slovním procesoru).

Znaky položek se na tiskárně tisknou, jak jsou sestaveny na obrazovce (nikoli však coby kopie obrazovky, jako u výše jmenovaných zahraničních databází, ale velmi výhodně jako kódy ASCII, přizpůsobené čs. normě KOI8 - tisk je díky tomu rychlý). Bohužel DATALOG neumožňuje dodatečné rozvržení výpisu na tiskárně za použití jejích řídících kódů. Tak třeba nemohu vyplnit složenku, do níž se zapisuje čtyřikrát ta samá adresa do čtyř kolonek vedle sebe. Apod. Pro takový účel si budu muset vyrobit svůj vlastní "stroják".

Konec výhrad. Uvědomuji si, že se jedná o první verzi tohoto programu. A na to, že je první, je - bez jakéhokoli fandění - těchto drobných výhrad více než málo. Určitě bude mnoho nových uživatelů DATALOGu požadovat, aby Mikrobáze distribuovala převodník dat z MASTER-FILU do DATALOGu (Petr už přislíbil jeho tvorbu). Když nový program viděl jeden z dalších nadšených (MUDr. J. Kofránek), ihned ho napadlo, že by Mikrobáze pro DATALOG mohla vytvářet užitečné soubory dat, které, jakožto počítačoví frenetici, při své práci využijeme. To by Mikrobáze určitě mohla a měla.

Pro zápis znaků s diakritickými znaménky autor zvolil netradiční formu (svůj díl "viny" na tom nesu i já). Každý uživatel však tomuto nóvu po krátké době rychle přivykne. Tak např. chcete-li napsat písmenko č, podržíte tlačítko c o chvilku déle. To platí i při psaní velkých písmen. Zápis v obou "úrovních" je rozlišen zvukovou signalizací. Tímto časováním je počítač využit jako počítač, není pouhou simulací mechanického psacího stroje. Jako vždy, všechno má své plus i mínus. V tomto případě však nejde o nic jiného než o zvyk. Neprofesionální "datlové" (je nás zdrcující většina) toto řešení přivítají i proto, že si nebudou muset pamatovat separátní rozmístění písmenek s dia-znaménky na klávesnici (provázené jejich věčným hledáním, ev. i obtížným, stejně nekonečným "šiftováním").

K programu je připojen velmi podrobný dvoudílný manuál (jak se vyjádřil jeden z redaktorů-recenzentů - "ukecaný" manuál). Podle mého náhledu je právě detailní rozvedení práce s programem velmi důležité pro uživatele, kteří nemají žádnou zkušenost s tímto typem programu (resp. chtějí využívat počítač převážně jako nástroj pro třídění dat a jejich aktualizaci). Ti musejí být do problematiky uvedeni se vším všudy. Toto hledisko první část manuálu splňuje. Druhá, kratší, je věnována programátorům. Jsou v ní uvedeny i některé systémové proměnné programu, jejichž změnou můžeme "tvarovat" výsledky některých funkcí programu dle našeho přání. Uvedena je i řada dalších důležitých informací o DATALOGu a programové příklady pro provedení speciálních změn. V tomto směru Mikrobáze (ve spolupráci s autorem) plní svůj slib neformálního, nekonvenčního přístupu ke svým členům.

Závěrem chci podotknout, že v recenzi nejsou uvedeny všechny dispozice, jimiž se DATALOG vyznačuje. Recenze není manuál. A i když se vám třeba může zdát, že díky neuvěřitelně snadné obsluze programu se bez manuálu obejdete, je to pravda jen zčásti. Až si jej přečtete, uvidíte sami.

Člověk si nikdy nemá hrát na proroka. Ale přesto si myslím, že architektura

DATALOGu bude u tohoto typu programu na malých osmibitových počítačích už těžko překonána. Možná ale snad, kdyby...Petře, co říkáš, nešlo by pro další verzi počítat se zařazením funkcí file-processoru? To by nás pak už nikdo nepředhonil.

Ladislav ZAJÍČEK

µB - PASCAL

(Recenze původního programu Mikrobáze)

Jazykem, který máme k dispozici ihned po zapnutí Spectra, je Basic. Přesněji řečeno dialekt ZX-Basicu. Přes všechno (snad až příliš módní) brojení proti tomuto jazyku si Basic stále drží svou pozici. Jeho obliba vyplývá především z jednoduché, snadno zapamatovatelné syntaxe jazyka a z možnosti interaktivního ladění programů.

Při řešení složitějších úloh v Basicu však narážíme na mnohé překážky. Pomalost interpreteru je tou "nejmenší" z nich. Lze ji snadno odstranit kompilátorem (překladačem). Hůře se však obcházejí chybějící základní programové konstrukce (procedury s parametrem a lokálními proměnnými, možnost rekurze, složitější datové struktury, dynamická alokace paměti apod.). Některé z nich (např. zmíněné parametry u procedur) jsou vyřešeny u složitějších dialektů Basicu. Mezi "spectristy" nalezly oblibu tři z nich: Beta-Basic, Mega-Basic a Sigma-Basic.

Nicméně tyto složitější dialekty Basicu přesto samy o sobě neodstraňují hlavní nectnost Basicu, kterou je samotný způsob programování. Orientace příkazů jazyka do jednotlivých očíslovaných řádek, na které lze v programu libovolně skákat, svádí ke vzniku programových monster, ve kterých se záhy nevyzná ani jejich tvůrce.

Mikrobáze přichází s nabídkou jazyka založeného na jasných a srozumitelných konstrukcích, respektujících všechny zásady strukturovaného programování a umožňujících uživateli vytvářet celou plejádu nejrůznějších datových struktur. Tímto programovacím jazykem, vytvořeným původně pro potřeby výuky programování, je Pascal.

Dnes, více než patnáct let po jeho vzniku, je možno konstatovat, že se Pascal osvědčil nejen jako vynikající prostředek pro výuku programování, ale stal se i oblíbeným jazykem pro vytváření nejrůznějších aplikačních (převážně nenumerických) programů.

Autoři µB-Pascalu - Dr. Januš Drózd a Dr. Petr Couf - vytvořili implementaci i pro československé mikropočítače PP-01 a IQ-151 (pod operačním systémem AMOS). To znamená, že programy jsou na úrovni zdrojového textu na tyto, v našem školství používané počítače, bez větších problémů přenositelné.

Na mikropočítačích Spectrum je rozšířen překladač Pascalu firmy HiSoft a jeho ing. Petrem Adámkem modifikovaná verze. Proto ve své recenzi budu přednosti a nedostatky µB-Pascalu srovnávat s touto mezi "spectristy" značně rozšířenou implementací.

Po nahrání µB-Pascalu do počítače okamžitě přejdeme do režimu řídícího jazyka, ve kterém je možno zapisovat nebo načítat zdrojový text na magnetofon či microdrive, nastavovat parametry překladače, spouštět překlad zdrojového textu programu apod. V tomto režimu také překladač podává uživateli nejrůznější zprávy (o souborech na vnějších paměťových médiích, o průběhu překladu, o chybách, zjištěných během překla-

du nebo běhu programu atd.). Na rozdíl od HiSoft-Pascalu je škála těchto hlášení podstatně bohatší a navíc probíhá v českém jazyce.

Z řídícího režimu je možno přecházet do režimu celostránkového editoru (64 znaků na řádku, 24 řádek na stránku), s jehož pomocí vytváříme vlastní zdrojový text programu. Z tohoto hlediska představuje µB-Pascal integrovaný programový systém, zahrnující prostředky pro editaci, překlad, ladění a spouštění programů vytvářených v jazyce Pascal.

Rozdělení paměti je následující:

Nad systémovými proměnnými a úsekem určeným pro microdrivovou mapu a kanálové informace je oblast paměti, vyhrazená pro ukládání zdrojového textu programu. Tato oblast je shora omezena adresou, uloženou v systémové proměnné RAMTOP. Pod RAMTOPem je uložen strojový zásobník. Pokud se zdrojový text přiblíží (asi na 80 bajtů) k vrcholu zásobníku, je ohlášena chyba a systém již z klávesnice nebo z vnějšího paměťového média žádné další příkazy zdrojového textu nepřijme.

Strojový kód vytvářený při překladu se ukládá do pracovní oblasti překladače, uložené nad RAMTOPem. Rozdělení paměti mezi úsek vyhrazený pro ukládání zdrojového textu a úsek pro generovaný kód můžeme změnou obsahu systémové proměnné RAMTOP měnit podle potřeby.

Při dalších programech je možno zdrojový text (celý nebo pouze jeho část) zapsat v sektorech (po 512 bajtech) na magnetofon nebo microdrive a tyto úseky pak při překladu postupně načíst a překládat. Tak je minimalizován prostor vyhrazený pro zdrojový text programu ve prospěch oblasti určené pro uložení generovaného strojového kódu.

Oblast pro generovaný kód je shora omezena - adresa jejího konce je v systémové proměnné PRAMT. Nejvyšší adresa, kam smí zasahovat kód přeloženého programu (kdy ovšem přepíšeme část kódu řídícího systému), je 0D32H. Z této oblasti je možno snížením PRAMT ještě "vykousnout" úsek, kam je případně možno uložit uživatelem vytvořenou rutinu ve strojovém kódu, kterou je možno z µB-Pascalu volat.

Vytvořený strojový kód programu je možno zapsat na magnetofon nebo microdrive, nebo jej z magnetofonu či microdrivu do počítače načíst.

Zde je třeba upozornit na to, že ve stávající verzi µB-Pascalu nelze vytvářet programy samostatně spustitelné mimo systém µB-Pascal. Generovaný kód totiž využívá některé procedury uložené ve speciální knihovně, která je součástí systému µB-Pascal (tzv. knihovna provozních, neboli runtimových procedur).

Překladač firmy HiSoft tvorbu samostatně spustitelných programů umožňuje - na magnetofon či microdrive se zapisuje program ve strojovém kódu včetně celé rutinové knihovny. HiSoft-Pascal má pro překlad rozsáhlých programů i možnost tzv. destruktivního překladu. V tomto režimu se překladač přepíše generovaným programem, který se na závěr zapíše na magnetofon či microdrive.

Na druhé straně však, překládáme-li v HiSoft-Pascalu program v nedestruktivním režimu překladu a nestačí-li místo pro uložení vytvářeného kódu, překlad se zastaví. Jako varování se ohlásí zpravidla nesmyslná syntaktická chyba. Stiskneme-li poté omylem klávesu pokračování v překladu, systém se zhroutí. V některých případech se ale zhroutí i bez tohoto varování. µB-Pascal má tyto případy dokonale ošetřeny.

V zájmu objektivity je třeba upozornit na následující nedostatek µB-Pascalu. Generovaný strojový kód využívá pouze podmnožinu instrukcí mikroprocesoru Z80, která je kompatibilní s podmnožinou instrukcí INTEL 8080. To vede k vytváření relativně delších strojových programů, než je tomu u HiSoft-Pascalu.

Abychom si udělali představu o rychlosti vytvářeného programu, proveďme test s programem (viz výpis 1), který zjišťuje počet prvočísel v rozsahu 1 až 16383. Napíšeme-li obdobný program v BASICu (výpis 2), pak se doba výpočtu bude pohybovat okolo 6 minut a 10 sekund. µB-Pascal je s výpočtem hotov za 9 sekund. Program přeložený HiSoft-Pascalem se s výpočtem vypořádá za 4 sekundy.

µB-Pascal byl m.j. vytvářen pro účely výuky programování. Proto se autoři zaměřili na hlídání mnoha možných syntaktických i běhových chyb. To pochopitelně trochu zpomaluje rychlost tvorby generovaného kódu i jeho běh. Kontrolu však můžeme vypnout pomocí nastavení parametrů překladu. Pak se doba běhu uvedeného testovacího programu zkrátí z 9 na 3 sekundy! Vypneme-li obdobné kontroly (kterých je ale méně) v HiSoft-Pascalu, zrychlíme program rovněž na 3 sekundy.

Počet chybových hlášení při překladu je u HiSoft-Pascalu 68, u µB-Pascalu téměř dvojnásobek - 126, což podstatně zlepšuje detekci chyb. Navíc - některé chyby HiSoft Pascal detekuje nedokonale. Stačí např. v programu z výpisu 1 změnit konstantu Rozměr z 8190 na 30000 a program v režimu nedestruktivního překladu přeložit. Překladač HiSoft-Pascalu (na rozdíl od µB-Pascalu) neohlásí, že vektor dlouhý 30000 bajtů se do vyhrazeného místa nevejde. Překlad proběhne zdánlivě bez chyb, avšak spustíme-li přeložený program, systém se zhroutí.

Záludnější než syntaktické chyby, odhalitelné už při překladu, jsou chyby, které se projeví až při běhu programu (např. dělení nulou apod.). HiSoft-Pascal hlídá 11 běhových chyb, µB-Pascal má implementováno celkem 37 různých hlášení o chybách při běhu programu. Navíc na obrazovku či tiskárnu vypisuje mnohem dokonalejší protokol o zastavení programu než HiSoft-Pascal, který uvede nanejvýš obsah čítače adres. Velice užitečné je hlídání běhových chyb při práci s dynamickými proměnnými. Chceme-li vytvořit dynamickou proměnnou a není-li již pro ní místo, µB-Pascal nás na to upozorní; v HiSoft-Pascalu se "zatmí obrazovka" a naskočí "osudový" nápis "Sinclair Research..."

µB-Pascal se v hlavních rysech drží mezinárodní normy Pascalu ISO 7185. Podstatná je pouze jedna odchylka - pro rozlišení dvou identifikátorů je významných pouze prvních šest znaků. Pro zlepšení čitelnosti lze kamkoli do identifikátorů umístit znak podtržítka, který se při rozlišování identifikátoru ignoruje.

HiSoft-Pascal má - oproti normě jazyka - následující základní omezení, která jsou dosti nepříjemná:

1. - příkaz CASE nesmí mít variantní část
2. - procedury a funkce nemohou být formálními parametry
3. - chybí procedura DISPOSE, uvolňující nepotřebné dynamické proměnné do volné paměti
4. - chybí datový typ Soubor (FILE)

µB-Pascal tato omezení nemá! Oproti normě má navíc následující zásadní rozšíření:

1. - příkaz CASE může být ukončen klíčovým slovem ELSE, které se provede v případě, kdy hodnota selektoru větvení se nerovná žádné z uvedených konstant
2. - funkce PEEK - vrací obsah zvoleného místa v paměti
3. - funkce POKE - zapisuje zadaný obsah na zvolenou adresu v paměti
4. - funkce INP a OUT čte, resp. posílá příslušný bajt z/na příslušnou bránu
5. - funkce ENABLE a DISABLE - povoluje a zakazuje přerušení
6. - procedura CALL provede strojový podprogram uložený na specifikované adrese; procedurou je podprogram ev. možno předat parametry do reg. BC, DE a zásobníku
7. - funkce FCALL, na rozdíl od předchozí procedury, navíc vrátí obsah reg. DE
8. - funkce REF vrátí adresu specifikované proměnné
9. - grafické procedury LINE, PLOT, ARC a CIRCLE

HiSoft-Pascal má zhruba analogická rozšíření. Neumožňuje (pomocí standardní procedury, obdobné procedurám CALL a FCALL z µB-Pascalu) přímo předat parametry uživatelskému programu, zapsanému ve strojovém kódu. Navíc má prostředky, jak zaznamenat na microdrive či magnetofon definovaný obsah paměti (ale nezná datový typ Soubor).

Manuál dodávaný k µB-Pascalu obsahuje podrobný popis způsobu práce s integrovaným programovým systémem Pascal a popis všech implementačních zvláštností.

ZÁVĚREČNÉ SHRNUTÍ

Největšími nedostatky stávající verze µB-Pascalu ve srovnání s HiSoft-Pascalem je chybějící režim destruktivního překladu a nemožnost generovat programy samostatně spustitelné mimo systém µB-Pascal. Mezi negativa můžeme zařadit i to, že kód je generován v instrukcích 8080, nikoli Z80.

Největší předností µB-Pascalu je mnohem podrobnější detekce syntaktických a běhových chyb, dokonalejší ochrana před fatálními chybami způsobujícími zhroucení systému a implementaci odpovídající požadavkům normy jazyka (HiSoft-Pascal nemá implementovány některé důležité konstrukce).

Hledáme-li pro oba překladače oblast jejich převážného použití, pak v případě tvorby rozsáhlých programů je (vzhledem ke kratšímu kódu a možnosti destruktivního překladu) výhodnější HiSoft-Pascal. Pro tvorbu kratších aplikačních programů a zejména pro účely výuky je ovšem nesporně mnohem vhodnější µB-Pascal.

Jak sdělili autoři µB-Pascalu, připravují pro zpravodaj Mikrobáze materiál, pojednávající o tom, jak umožnit, aby překladačem vygenerovaný kód pracoval samostatně mimo µB-Pascal (pouze s "vypreparovanou" runtimovou knihovnou v délce cca 7K).

Zvláště těm z vás, kteří se Pascal teprve hodláte učit (jeho studium vřele doporučuji), µB-Pascal ušetří celou řadu problémů, s nimiž byste se potýkali v případě HiSoft-Pascalu.

MUDr. KOFRÁNEK, CSc.

V domovině Specter

Jak vyplývá z anglických časopisů, spectristé jsou pro tamní výrobce povážlivě nezajímavou skupinou zákazníků. Snad do trosek zmírajícího království sira Sinclaira vnese trochu života amstradovské Spectrum +2. Takže, upřímně řečeno, není až tak z čeho vybírat. Týká se to především hardwaru.

Firma Saga nabízí několik "náhradních" klávesnic pro ZX Spectrum v cenách od 40 do 70 Lstg. Zvláště ty dražší jsou pěkné, ale cenově dnes přesahují cenu samotného počítače.

Pestrá nabídka je na trhu joysticků. Hry jsou hlavní potravou domácích Specter. Test měsíčníku Sinclair User vyhodnotil joysticky takto (maximum je 100 procent): Mach 1 (95), Speed King (91), Elite (89), Kraft (85), 125+ (82) atd. až po Master a Gun Shot (po 65). Mach stojí 15 Lstg, nejlacinější je za 7 Lstg.

Několik malých firem vyrábí pro ZX Spectrum myši. Z testu čtyř (ve stejném časopise) vyšla vítězně myška AMX. Spolu s programy AMX Art, Mouse Control, Icon Designer, Calculator a Puzzle stojí 70 Lstg. Je kompatibilní s grafickými programy Art Studio a Artist II. Spodní cenová hranice u "slabších" myší je 50 Lstg.

Již před nějakým časem se na spectristickém trhu objevily diskové jednotky. Velké popularitě se těší Opus Discovery. Je konstruován pro diskety 3,5´´ o obsahu 250K neformátovaných. Její černá skříňka obsahuje (kromě diskového) další interfejsy pro joystick, tiskárnu a monitor s video-vstupem. Má vlastní operační systém BK ROM a zabudovaný zdroj, kterým napájí i ZX Spectrum. Prodává se s programem, který převede váš software z pásku na disk. Pro bývalé uživatele microdrivů je příjemné, že má shodnou syntaxi ovládacích příkazů. Vzhledem k celkové vybavenosti je její cena 100 Lstg velmi nízká.

Dumpingovou cenou Opusu Discovery byl výrobce diskové jednotky Beta přinucen ke snížení cen. Přesto jeho diskový interfejs Beta 128 stojí celých 70 Lstg. Koupíte -li si však spolu s ním i jednotku, interfejs vám do ceny "není započítán" a spolu vás vše přijde na 160 Lstg u 3,5´´, resp. na 180 Lstg u 5,25´´. Obě jednotky obsáhnou po 1 MB záznamu, což už stojí za to. Dvojče přijde na 260, resp. 300 Lstg. Interfejs k tiskárně si musíte koupit zvlášť za 20 Lstg. Beta je oboustranná, pracuje s dvojitou hustotou záznamu na 80ti stopách. Má vlastní zdroj a je kompatibilní se ZX1.

Microdrive s interfejsem ZX1 se prodává v průměru za 50 Lstg. Kupujete-li více microdrivů najednou, přijde pak každý na cca 20 Lstg. Cena kazetky nepřesahuje 2 Lstg. Wafadrivy mají značný cenový rozptyl - od 60 až přes 100 Lstg.

Objevují se varianty již dlouho známých interfejsů pro komunikaci s tiskárnami. Oblíbený ZX LPRINT III, který býval k dostání jen pro paralelní přenos, je obohacen o sériový port. Umí vykonávat přímé příkazy LLIST, LPRINT a COPY. Stojí 30 Lstg. Za stejnou cenu se prodává např. i Tasman Printer Interface, který komunikuje s dlouhou řadou tiskáren. S určitou inovací přišla firma Ram Electronics. Její Ram-Print má (ve vlastní ROMce) implikovaný slovní procesor RamWrite, který lze kdykoli zavolat. Výhodou je, že pro zapisovaný text můžete využít celé volné paměti, nemusíte nic předem nahrávat. Nevýhodou je, že slovní procesor ve srovnání s Taswordem,

Spectral Writerem či Last Wordem nestojí za moc, navíc je nepřístupný programovým úpravám, který kutilové u tohoto typu programů tak rádi provádějí. Co lze považovat za neslušnost vůči zákazníkům, je jeho ubohý, odflinknutý manuál. RamPrint má ještě dva vstupy pro joysticky. Stojí 40 Lstg.

Firma Romantic Robot upravila svůj stařičký Multiface One při nezměněné ceně 40 Lstg. Má svůj operační systém 8K, přičemž dalších 8K poskytuje uživateli jako volnou RAMku. Tak vlastně rozšiřuje rozsah volné paměti na 56K. Darovaných 8K RAM můžete využít (např. přidaným MULTITOOLKITem) pro monitorování a úpravy her, které vám interfejs kdykoli zastaví. Umožňuje i přenos jakéhokoli programu z a na m-drive, Wafadrive, pásek, disketové jednotky Beta, Opus a Kempston. Lze připojit i joystick. Zajímavostí je, že u přenášených programů provádí současně kompresi dat, takže zkracuje jejich celkovou záznamovou plochu i čas nahrávky. S tiskárnami nekomunikuje, ale je "průchodný". Interfejsům, které mají implementovánu funkci COPY, můžete při práci s Multifacem dát příkaz k tisku stávající obrazovky.

Ceny tiskáren, které šly před časem prudce dolů, svůj pokles zpomalily. Nejlacinější je tiskárnička Alphacom, za pouhých 20 Lstg. Seikosha SP 1000 přijde na cca 200 Lstg, podobně jako Brother M1109. Epson LX86 stojí cca 230, FX85 400, Kaga Taxan KP-810 270 Lstg. A pro zajímavost - laserové tiskárny, které tisknou 300-600 řádek za minutu, stojí kolem 2000 $.

Pracovat s monitorem je určitě snem každého spectristy. Několik firem začalo vyrábět interfejsy, jež umožňnují připojení některých typů monitorů. RGB interfejs stojí kolem 45 Lstg. Monochromatické monitory se nabízejí v cenách do 100 Lstg, lacinejší barevné kolem 200 Lstg.

Zajímavým produktem je Dataskip Videoface, který provede za 0,27 sec. digitalizaci TV obrazu z videokamery, videomagnetofonu nebo videovýstupu televizoru v rozsahu 256x 192x 4 bity. Pomocí dodaného softwaru lze obraz dodatečně upravovat. Uchová 6 obrazovek pro případnou tvorbu animace. Je kompatibilní s m-drivem a diskovou jednotkou Beta. Cena 70 Lstg. O 10 víc stojí Volex TTX2000S, který spolupracuje se ZX Spectrem při příjmu a zpracování teletextu.

Na tomto místě je vhodné uvést, že uvedené katalogové ceny nezahrnují kabely, které jsou značně drahé; jejich cena se pohybuje kolem 10 Lstg. Katalogové ceny jsou rovněž vždy o něco vyšší, než za jaké lze výrobky sehnat při běhání po obchodech.

Poslední hardwarové výrobky, které stojí za povšimnutí, jsou modemy. Jedním z typických představitelů moderního modemu pro spojení se ZX Spectrem je VOYAGER za 80 Lstg. Při dokoupení potřebného softwaru a interfejsu vás vše přijde na něco přes 100 Lstg.

O něco živější je softwarový trh. To se však týká zábavných programů. U užitkových (až na drobné výjimky) je to bída. Ty z vás, kteří si oblíbili slovní procesor Tasword, bude zajímat, jaké dodatky k němu firma dodává. TASWIDE upravuje zobrazení znaků při práci s Basicem na šířku 64 na řádce. TASPRINT obsahuje 5 typů písmen pro tisk na tiskárnu v grafickém módu. TASCOPY vytiskne obrazovku. TASMERGE přenáší do Taswordu záznamy z Masterfilu. Novinkou je TASWORD THREE, který je dodáván pouze na kazetkách m-drivu. Je kompatibilní se systémem Opusu Discovery. Obsahuje některá

přilepšení, jako např. mail merge. Pro nové Spectrum 128 je připravena verze TASWORD 128.

Objevily se tři nové programy pro práci s grafikou Spectra. Firma Softcat nabízí ANIMATOR 1, který má být stejně snadno obsluhovatelný, ale schopnější než výborné Art Studio. Lze jej doplnit modulem Poster Machine, jímž můžete tisknout plakáty, sestavené z většího počtu "obrazovek". Mimochodem, nemáte jej už někdo? Tato otázka platí i pro ARTIST II. Ten se skládá ze tří modulů - základního, pro tvorbu stránek (jejich grafiku je pak možno tisknout společně s textem ze slovního procesoru Writer firmy Softechnics!) a pro kreace sprajtů. Vzhledem k tomu, že se jedná o obohacenou verzi předchozího programu Artist, lze očekávat opravdu vynikající výsledky. Do třetice - LORIGRAPH, obdoba Art Studia co do snadnosti obsluhy i schopností, které jsou mírně rozšířeny. Pnogram pracuje pouze s černobílou kresbou (barevné úpravy je nakonec možno dodělat jiným programem). Hlavní přednost LORIGRAPHu spočívá v tom, že stanovené části kresby pojímá pseudoprostorově. Tak je možno jimi otáčet podle uživatelem dané osy. Co tato možnost přináší do tvorby kresby, si dokáže představit každý, kdo se computerové grafice věnuje častěji.

Závislé uživatele Basicu potěší svou solidností známá firma HiSoft programem BASIC COMPILER, který převede basicový program na strojový kód. Výsledné zrychlení běhu zkompilovaného programu se v průběhu pohybuje mezi násobky 4 až 10 (podle četnosti užití "hůře kompilovatelných" příkazů Basicu). Při užití těch "snadněji" převoditelných může být program rychlejší až 150 krát! Jako všechny předchozí kompilery, má i tento svá omezení, avšak jen malá. Neporadí si se "systémovými" příkazy SAVE, LOAD apod. a některými vyjádřeními při operacích s řetězci. Program má několik předností, které se projevují zvláště při práci s číselnými proměnnými. Jsou-li jimi celá čísla, ukládá je v úsporném formátu. Pokud mu předem nesdělíte, že tak má učinit, dokáže je při kompilaci najít a na tuto možnost vás upozorní. Rovněž vyhledává dělení a násobení dvěmi (kdo zná assembler, ví proč). Kompilace probíhá ve třech průbězích, během nichž obrazovka slouží jako buffer. Verze pro 48K je dlouhá cca 10K. Tak vám pro Basic zbyde kolem 30K. Verze pro Spectrum 128 +2 zabírá asi 500 bajtů v basicové oblasti, zbytek je ve druhé "paměťové bance" 64K.

Absolutním hitem systémových programů je LASER GENIUS (editor-assembler a monitor-analyzátor) fy Oasis. Jeho popis by zabral mnoho stran; tedy jen něco z toho mnohého, co nabízí - macro s proměnnými parametry, Phoenix s rozšířenou sadou pseudoinstrukcí pro tvorbu především matematických programů, analyzér se syntaxí Forthu, možnosti přenosu návěští z a do částí tvořeného programu, resp. při připojování subrutin z knihovny, volbu řady činností - např. trasování, definici několika pracovních oblastí pro uložení dat z analyzovaného programu či uživatelské obrazovky atd. To však zdaleka není vše, co s tímto prvním vývojovým systémem pro ZX Spectrum můžeme činit. Manuál obsahuje i příkazy pro Amstrad a jejich modifikace pro Spectrum 128.

Herní software, který je vlastně tím hlavním tržním produktem světa Specter, stupňuje svou rafinovanost. Vedle hromad příšer a hrůz všeho druhu se objevují myšlenkově i profesionálně dokonalé prcgramy, které jsou omezeny už jen rozsahem paměti ZX Spectra a jeho, pro dnešní dobu už příliš limitovanými možnostmi grafického vyjádření. Šachoví fanoušci jistě uvítají první prostorové šachy PSI CHESS. K simu-

lacím se řadí TT RACER, který vzdáleně připomíná výborný Full Throttle, HARDBALL simuluje baseball, BUMP SET SPIKE volejbal, ROOM TEN je squash pro astronauty (hraje se v prostoru s malou gravitací), XENO je zvláštním hokejem se dvěma "disko-hokejisty". Tvůrci skvělé strategické hry Arnhem přišli s další specialitkou z dějin bitev - NAPOLEONem a jeho bojem u Eylau. Bojové letecké operace jsou hned ve dvou vydáních. HARRIER se rozhodně řadí k nejzdařilejším simulacím tohoto typu. Konečně je tu zase vzlet a přistání se vším všudy. Startovat lze dokonce třemi způsoby, i vertikálně (tím čtvrtým je samozřejmě rozbití letadla). Ovládání není jednoduché. Vzpomínáte na počet ovládacích tlačítek programu Fighter Pilot? HARRIER jich má 30. Výborně zpracovaný manuál vás zavádí do vysoké školy letectví. Oproti tomu ACE je příliš zjednodušenou simulací řízení letu (nahoru-dolů-vlevo-vpravo-pal). Akční. Leč vnitřní chudoba brzy unaví. REBEL STAR se řadí ke strategickým hrám - jedna báze na měsíci útočí, druhá se brání; hraje jeden proti počítači, nebo dva proti sobě. Pokud jste si před léty oblíbili hru Football Manager, padne vám do noty HEAD COACH. Coby manažer týmu amerického fotbalu jej vedete k metám nejvyšším. Stále se objevuje množství her, vycházejících z "filosofie" programu Knight Lore (říkám jim "rohovky" podle umístění krychlových prostorů jedním rohem proti hráči). Mezi ty nové patří např. NEXOR, GLIDER RIDER či GREAT ESPACE, který se však od obvyklé schématičnosti "rohovek" odlišuje svým dějem. K této skupině patří i TRAIL OF DARKNESS, který je označován za Fairlight II. Kombinací "rohovky" s akční i dialogovou hrou je PRODIGY. Populární Subre Wulf našel svého dokonalejšího pokračovatele ve hře FIRELORD. Pokud už jste si zahráli na vesmírného obchodníka-bojovníka ve hře Elite, pak vás musí nadchnout novinky, která čerpá z podobného provedení animace, ale na vyšší úrovni - STAR GLIDER. Jestli jste měli trpělivost s hrou Tau Ceti, čeká vás další oříšek - ACADEMY. Jde o značné rozšíření dějovosti Tau Ceti, které ACADEMY zůstává věrná i po grafické stránce. Mezi výjimečné tvůrce, jehož hry nelze zařadit do nějaké škatulky, patří autor skvělých, strategií nabitých Lords of Midnight. Jeho nový výtvor DARK SCEPTRE je ideovou obdobou Lordů. Provedení se však výrazně liší animací. Mezi originality s náporem na logické myšlení v akci patří TRAPDOOR s půvabnými strašidélky. A co ještě stojí za zmínku? URIDIUM - akční hra, která slavila úspěchy na Commodoru. Klonem Shadow Fire a Fairlight je STRIKEFORCE COBRA. Ale to není ani zdaleka vše! Naši spectrističtí dekodéři a odtajovači zakletých programů si už určitě masírují buňky. Jen do toho, chlapci! Jsou nás tisíce, kteří už se těšíme na výsledky vašich probdělých nocí, abychom je pak stejně nerozumně probdívali také.

Sinclair User 9-12/86
vybral -elzet-

Programová nabídka MIKROBÁZE

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 písátek, 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.

/uB-PASCAL

Integrovaný systém umožňující editaci, překlad a provádění programů v jazyce PASCAL. Obrazovkový systém editoru pracuje se 64 znaky na řádce. Použitá verze jazyka PASCAL je velmi blízká mezinárodní normě ISO 7185 (úroveň Ø) a implementacím DC-PASCALU na mikropočítačích IQ 151 a PP 01. Jazyk obsahuje řadu rozšíření. Překladač je navržen tak, aby byl vhodným prostředkem i pro výuku programování. Poskytuje detailní chybovou diagnostiku a možnost přísných běhových kontrol. Programy mohou pracovat na logické úrovni se soubory, které fyzicky vstupují nebo vystupují přes klávesnici, obrazovku, tiskárnu, magnetofon, microdrive. Přiložen přehledně zpracovaný manuál.

DATALOG

Svým uživatelským komfortem v mnoha směrech výrazně převyšuje obdobné databázové programy pro ZX Spectrum. Založení databanky a formátu výpisu všech zpráv a položek je snadné (řešeno přímým grafickým navrhováním formátu s průběžnou kontrolou jeho vzhledu na obrazovce) a velmi variabilní. To platí i pro provedení jakékoli změny nebo opravy. Uživatel je při práci s DATALOGEM veden jednoznačnou volbou funkcí z posloupnosti přehledných menu. Kterákoli položka zprávy může být vypisována ve formátu 64 nebo 32 znaků/ř. Novinkou, kterou uživatelé ZX Spectra ocení, je možnost dělení souborů dat databanky podle uživatelem stanoveného výběru s následným individuálním zápisem vybraných částí souborů na vnější paměťové médium. Takto vzniklé "dílčí soubory" mohou být do databanky načteny jak samy o sobě, tak i přičleněny k souboru v databance přítomnému, tedy spojovány. Komunikace se záznamovými zařízeními je zajištěna příkazy Basicu, které jsou uživateli přístupné, přizpůsobitelné jakémukoli záznamovému zařízení. Dodávaná verze obsahuje příkazy pro magnetofon a microdrive. Přenos dat na tiskárnu neprobíhá pomocí funkce COPY, ale ve formě znakových kódů. DATALOG pracuje s českou a slovenskou abecedou, implementována jsou i jinojazyčná písmena, vyskytující se např. v příjmeních. Velmi detailně zpracovaný manuál DATALOGu má dvě části. První je určena běžným uživatelům, druhá poskytuje programátorům informace především o možnosti provedení změn některých parametrů DATALOGu.

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.

PLOŠNÍK

Se stane vítanou pomůckou každého konstruktéra - elektronika. S jeho pomocí lze snadno a rychle vytvářet návrhy plošných spojů, osazených až 100 obvodových prvků (včetně integrovaných obvodů). Jako vstupní parametry jsou do PLOŠNÍKu vkládány typy součástek, očíslovaná místa jejich spojů atd. Na výstupu je zpracován grafický návrh plošného spoje, který je možno dodatečně optimalizovat jak programově, tak přímo graficky na obrazovce. Výsledný návrh lze vytisknout na tiskárně s grafickým módem nebo s pomocí definovaných znaků tiskárny (download). Uživatelé bez přístupu k tiskárně mohou použít cestu fotografické kopie obrazovky.

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.

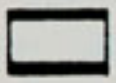
Karlovi se dávají povely běžnými českými slovy. Pochopení nových poznatků proto není ztěžováno současným učením anglických slov. Karel je také úslužný robot, všechno, co si může domyslet, udělá nebo napíše sám. S formální stránkou svého učení v nejvyšší míře sám napomáhá. A tak se stává z trpělivého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

AMSTRAD/SCHNEIDER

TRAN-SP-AM

Oboustranný převodník formátů zápisu a čtení dat mezi počítači ZX Spectrum a Amstrad/Schneider. Program např. umožní, aby text zapsaný na kterémkoli z obou počítačů bylo možno načíst "do jeho kolegy" a jakkoli s ním dále pracovat. Převod se provádí na počítači Amstrad/Schneider. To znamená, že TRAN-SP-AM umožní vzájemnou komunikaci mezi držiteli obou typů počítačů, stejně jako těm z vás, kteří ze Spectra přecházíte na Amstrad, nabídne možnost převodu všech vašich textů zapsaných na Spectru na záznamový formát vašeho nového počítače. Součástí programu je modul, který umožní konverzi znakových kódů, pokud jí bude třeba. Pokročilejší uživatelé výpočetní techniky správně tuší, že převádět lze nejen texty psané na slovních procesorech, ale i jakékoli jiné (zdrojový text assembleru, databázové záznamy apod.). Převod se provádí na počítači Amstrad/Schneider. Obsluha programu je vedena logicky řazenými dotazy menu, je proto velmi jednoduchá. Pro uživatele, kterým je formát dat ještě trochu hádankou, je připojen informativní manuál.

Všechny programy budou distribuovány pouze na kazetách.



Ve zpravodaji čís. 5 uvedená informace o programu ING.INK pro AMSTRAD/SCHNEIDER je zrušena po dohodě s autorem i klubem uživatelů tohoto typu počítače. Jeho členové se z důvodu potřeby rychlé aplikace česky a slovensky píšícího slovního procesoru rozhodli dát přednost úpravě zahraničního programu.

Tajemník klubů výpočetní techniky 602. ZO Svazarmu Dan Dočekal přišel s nabídkou přímé účasti členů klubu na převodu programů Mikrobáze na jiné typy počítačů. Protože programy jsou tvořeny v assembleru Z80, připadají v úvahu počítače se stejným typem mikroprocesoru. Jedná se především o SORD M3, SHARP MZ-821, Amstrad/Schneider. O tom, jak se tato spolupráce bude vyvíjet, vás budeme průběžně informovat.

Informace o aktuálním vývoji tvorby programů pro nabídku Mikrobáze a jejich distribuci:

Jak jste si mohli povšimnout, u některých programů dochází k časovým posunům. Programová tvorba je velmi náročnou tvůrčí činností. Stává se, že i když je program hotov, přijde nápad na vám prospěšnou změnu některé jeho funkce, či se objeví nepatrná "kosmetická" vada - a pak se musí počkat na realizaci potřebné úpravy. Velmi náročnou součástí nabízeného softwaru jsou manuály. V mnoha případech svým velmi hutným rozsahem překračují 100 stran strojopisu, což je už docela slušná kniha. Manuály jdou k redakčnímu zpracování a do výroby pochopitelně až po všech "vyladěních" programu. Jistě mnozí z vás víte, jaké problémy mají zahraniční výrobci užitkového softwaru ve vztahu k dodržení slíbených lhůt jejich distribuce. Oproti leckdy velmi odbytým manuálům těchto firem chceme, aby naše manuály byly tím, čím být mají - kvalitní, vyčerpávající informací o obsluze programu. Péče, kterou jim z tohoto hlediska věnujeme, si - i s ohledem na jejich rozsah - žádá svůj čas. Jakmile Mikrobáze dostane hotový program i manuál od tvůrce, je vše už věcí výroby, která, jak dokazuje pravidelnost vydávání zpravodaje, běží na plné obrátky. Co však nemůže nikdo urychlit (ani autoři sami), je tvůrčí proces vytvářených programů. Proto vás prosíme, abyste termíny uvedené u ještě nedistribuovaných programů brali jako plán, v němž může výše uvedenými vlivy dojít k posunu.

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.

Rub lístku budete vyplňovat v řádcích 1, 5 a 15. Do řádku 1 napíšete OBJEDNÁVKA PROGRAMU, do řádku 5 označení (název) programu podle nabídky.

POZOR! Programy ještě nemají přesná katologová označení, v nichž budou v budoucnu zakódovány typy počítačů. Proto zatím označení programu na řádku č. 5 uvádějte i s udáním typu počítače, pro který program objednáváte. Příklad:

5. Diapen/Sinclair Spectrum

Slovo k náhodným čtenářům

Dostal se vám tento zpravodaj Mikrobáze do rukou a zaujala vás aktivita rozvíjená v programových a technických službách pro uživatele 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 zpravodaje (v roce 1987 celkem pět čísel), budete moci využívat programových nabídek a dalších plánovaných služeb.

O bližší informace o celém komplexu služeb Mikrobáze a přihlašovací materiály je třeba požádat korespondenčním lístkem. Jeho líc vyplňte (zásadně strojem) podle vzoru na této straně, na rub napište sdělení: "Mám zájem o členství v Mikrobázi", připojte datum a podpis. Pak už stačí vhodit lístek do poštovní schránky a čekat na podrobné informační a přihlašovací materiály Mikrobáze. Dostanete je obratem. I když se naším členem stanete až v průběhu roku, máme pro vás připraveny všechny Zpravodaje Mikrobáze vydané od ledna 1987.