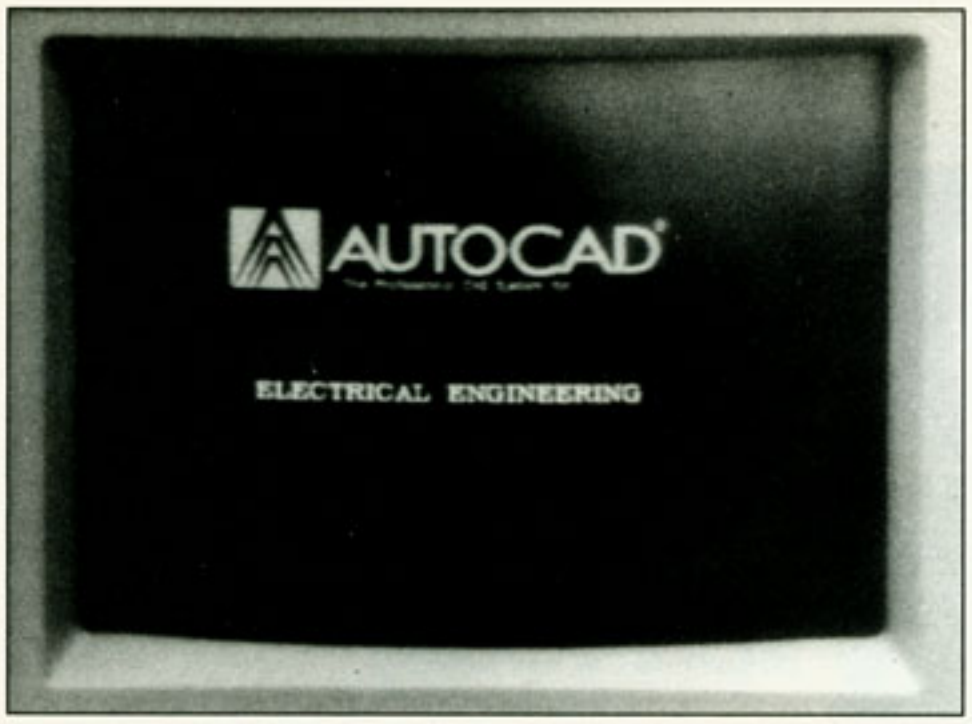
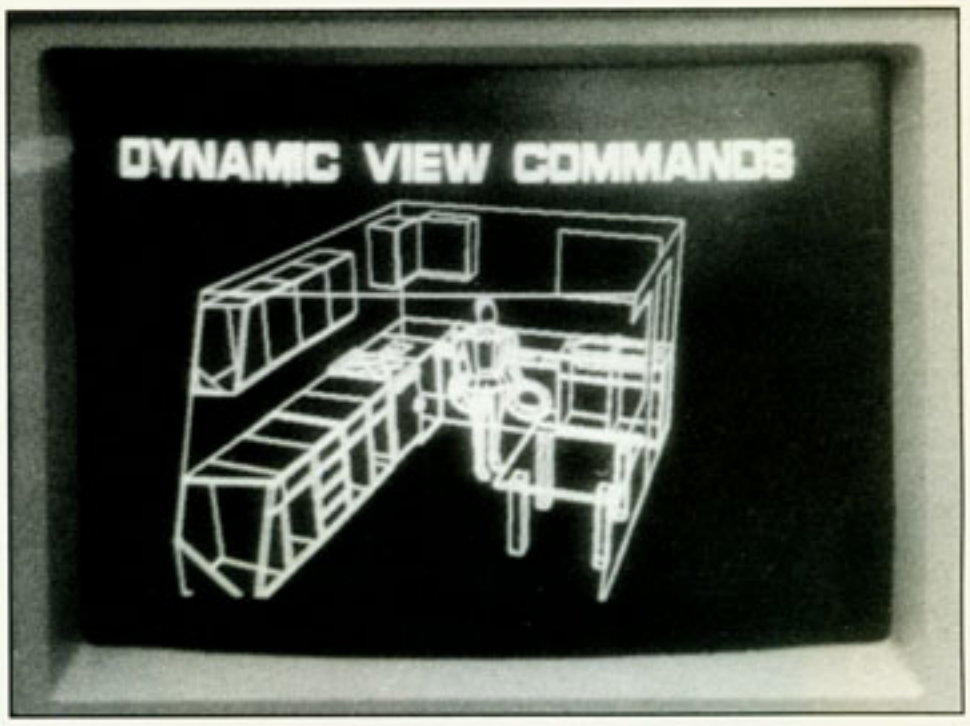
MIKRO BÁZE

1989 9

technický zpravodaj pro zájemce o mikropočítače

Cena 12 Kčs





AUTOCAD EXPO PRAGUE

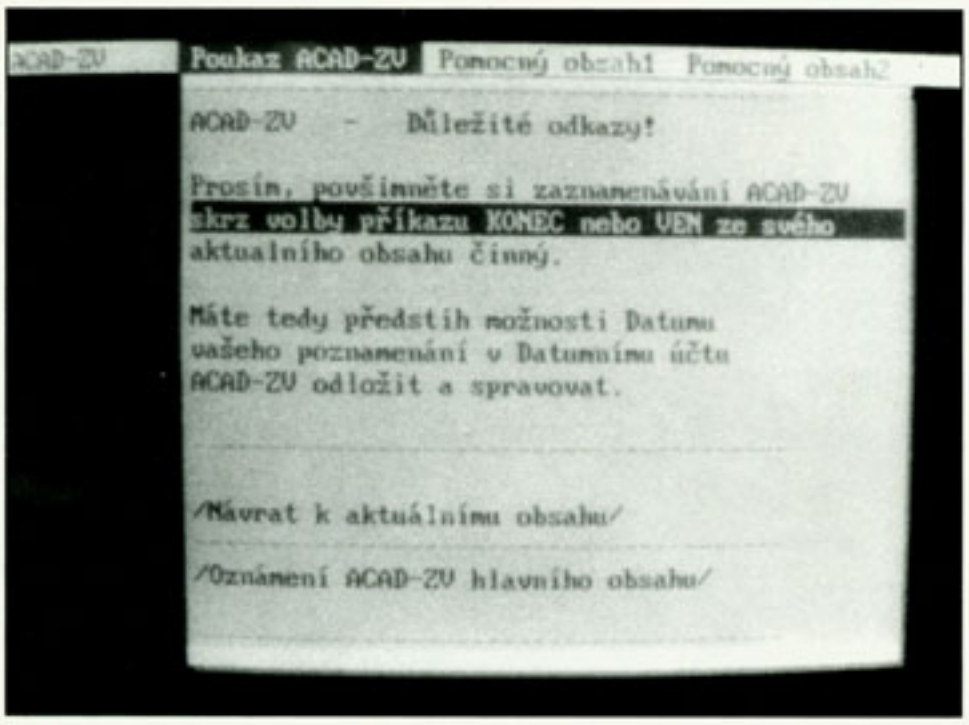
Tak se jmenovala výstava, kterou ve dnech 4. - 8.9.1989 pořádala britská firma Autodesk Ltd, ve spolupráci se Zenitcentrem SSM. Zúčastnilo se jí 20 zahraničních a 10 tuzemských vystavovatelů.

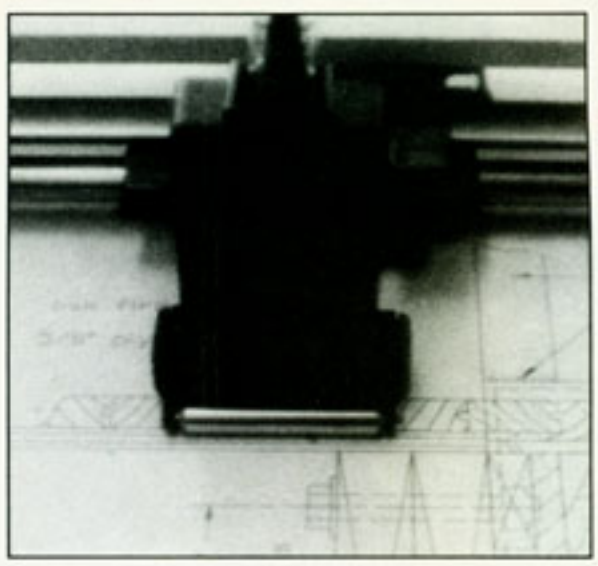
Sučástí výstavy bylo i školení uživatelů AutoCadu a vyhodnocení soutěže 2000+1 AIP. Podrobnosti přineseme v některém z příštích čísel Mikrobáze.

Mimochodem, při prohlížení přiložených fotografií vás asi napadne, že by bylo praktičtější, kdyby české verze programů připravovali Češi.

AUTODESK LTD
LONDON

ZENITCENTRUM





Text i foto: D. Meca

MIKRO BÁZE

1989/9

OBSAH

Technický zpravodaj Svazarmu pro zájemce o mikropočítače. Vydává 602. ZO Svazarmu ve spolupráci s redakcí časopisu Amatérské radio. Povoleno ÚVTEI pod evidenčním číslem 87 007. Sestavil vedoucí redaktor Daniel Meca. Obálka ak. grafik Jiří Blažek a Daniel Meca, grafická úprava Zdeňka Perglerová. Sekretářka redakce Zdeňka Válková. Redakční rada: Petr Horský, ing. Jan Klabal, ing. Petr Kratochvíl, Josef Kroupa, Rudolf Mach, Daniel Meca, ing. Alois Myslík, ing. Josef Truxa. Za původnost a správnost příspěvků ručí autoři. Ročně vyjde 10 čísel. Cena výtisku 12 Kčs podle ČCÚ a SCÚ č. 1030/202/86. Roční předplatné 120 Kčs. Objednávky přijímá a zpravodaj rozšiřuje 602. ZO Svazarmu, Wintrova 8, 160 41 Praha 6.

PODNIKEJ, ALE ZŮSTAŇ ČLOVĚKEM

K této malé úvaze mě přiměl nečekaný dopad drobného podnikání na mezilidské vztahy. Oč jde? Tak třeba o to, že před rokem, či před dvěma se v klubech výpočetní techniky stále něco kutilo, jednotlivé i kolektivně se ledacos vymýšlelo a když někdo vymyslel zajímavý program, či hardwarový doplněk, hned se v klubu pochlubil. A nejen to, program si mohli všichni nakopírovat, dokumentaci k hardwaru dal každý autor k dispozici a ještě pomohl ostatním při stavbě a oživení. Jednotliví autoři se předháněli ve snaze publikovat, takže na stránkách odborných časopisů byla přímo tlačenice.

V poslední době je ale všchno jinak. Přijdu například do klubu (libovolného), protože jsem se doslechl, že ten a ten vymyslel zajímavý doplněk k počítači. Myslím si, že by bylo pro naše čtenáře zajímavé uveřejnit příslušný stavební návod. Autor mi však místo žádané dokumentace dá do ruky ceník svých výrobků a služeb, které provádí na základě povolení ONV. Schema zapojení mi nemůže ukázat, natož pak abych ho otiskl - to by přišel o "kšeft".

Ano, zájmová činnost se stala předmětem obchodu. To samo by ještě nebylo nic zlého, pokud by se jednalo skutečně o obchod a ne o "kšeftaření", či dokonce o šmelinu. Získat za výsledek své práce určité finanční ohodnocení, to je jistě v pořádku. Jenže ono se často obchoduje také s výsledky cizí práce. A tak třeba, zatímco Jirka Lamač čeká až 602. ZO začne prodávat jeho novou CP/M pro Sharp, aby dostal zaslouženou odměnu za svou práci, jistý T. M. (prozatím uvádím jen iniciály) v klidu prodává tento program četným zájemcům po 300,- Kčs.

Řada drobných podnikatelů se snaží znemožnit okopírování hardwarových konstrukcí tak, že smažou hodnoty součástek. To lze označit za poněkud zpozdilé. Komplikuje to opravy, přitom však odborník si hodnoty součástek na dobrém kusu změří a postaví si kopii. Naopak člověk bez příslušných znalostí a vědomostí, nebo i ten kdo prostě na vlastní stavbu nemá čas, si výrobek rád koupí hotový. Ostatně se málokdy jedná o zcela originální zapojení, které by si zasloužilo právní ochranu. Navíc bývá výrobce málokdy autorem zapojení. Autor z toho většinou nemá nic. Řada zahraničních výrobců klidně zveřejňuje zapojení svých výrobků aniž by tím zchudla.

Nejde ovšem jen o schemata zapojení. V rámci podnikání se řada lidí nerada dělí s ostatními o své zkušenosti a praktické poznatky. Zřejmě mají pocit, že je lépe uplatní při různých placených službách. O své zkušenosti se však neradi dělí především ti, kteří jich mají sami málo. Skutečný odborník klidně každému rád poradí, aniž by ho nějak ubylo. Mohu to demonstrovat třeba na příkladu ing. V. Daněčka, autora BOBO 64. Žádnou ze svých konstrukcí nikdy před nikým netajil. Připravuje k uveřejnění popis řady unikátních servisních pomůcek, které používá při svém drobném podnikání. Konkurence ani úbytku zákazníků, kteří si pomohou sami se nebojí. Nápadů má prostě dost.

Přístup některých drobných podnikatelů někdy komplikuje život opravdovým amatérům a nadšencům. Pozoroval jsem před časem na burze jednoho počítačového nadšence, který ze všech sil smlouval o ceně diskového řadiče. Když už byli s prodávajícím dohodnuti na ceně 900,- Kčs, přihnal se jak velká voda jeden drobný podnikatel a hned nabídl 1200,- Kčs s tím, že jich vezme dvacet i více. Prý se mu to bohatě vyplatí. Amatér pochopitelně odešel s nepořízenou. Bude si asi muset koupit od některého podnikatele hotový finální výrobek, ve kterém ovšem za řadič zaplatí ještě mnohem víc, tak aby se to tomu podnikateli vyplatilo.

A to ještě není všechno. Existuje "podnikatel", který za celkem slušnou cenu (kolem 300,- Kčs) dodává paralelní interface s I8255. Má to však háček. Zapojení je na oboustranném prokoveném plošném spoji. Kdyby takový plošný spoj (který doma udělat nelze) skutečně koupil, musel by tenhle podnikatel na každém kusu nejméně 100,- Kčs prodělávat. Už tušíte na co narážím? No jistě - řada lidí podniká tak, že si ledacos potřebného prostě přinese z práce. Jiní sice různé díly koupí, jenže často zase od těch, kteří si je přinesli z práce, nebo je v práci vyrobili.

Z výše uvedeného je jasné, že drobné podnikání přináší vedle všeho pozitivního také řadu úskalí. Věřím však, že se postupem času všechno urovná a zvítězí v lidech to lepší. A tak apeluji - **podnikej, ale zůstaň člověkem.**

Daniel Meca

ZX SPECTRUM AKO TERMINÁL

Michal Kročka

Súčasným moderným trendom je spájanie výpočtových systémov do väčších celkov. Završením týchto trendov sú počítačové siete ako napríklad ETHERNET, DECNET a iné.

Celé toto úsilie má praktické opodstatnenie. Od jednoduchého prenášania dát až po zdielanie výkonných systémových prostriedkov, akými sú veľkokapacitné pamäte, alebo tlačiarne.

Takto možno vytvoriť veľké databanky do ktorých má prostredníctvom sietí prístup ľubovolný užívateľ.

Menšie mikropočítače sa využívajú ako inteligentné terminály na zber údajov.

Tento príspevok sa zaoberá pripojením počítača ZX-Spectrum ako jednoduchého terminálu.

Nevýhodou Spectra je, že nemá žiaden vstupno-výstupný port (ak nerátame magnetofónový vstup). Preto je potrebné zhotoviť jednoduchý sériový port RS-232C. Je možné použiť aj zapojenie uverejnené v AR-2/88.

Ďalšou nevýhodou je, že Spectrum zobrazuje len 22 riadkov po 32 znakoch. Tento problém sa bez zhoršenia čitateľnosti nedá odstrániť, a preto som sa ním nezaoberal.

Iným problémom je, že klávesnica má len 40 tlačítok, pričom negeneruje rôzne dôležité kódy ako ESC, ^C, ^S a iné. Preto je nutné programovo zmeniť funkcie niektorých kláves. Nové funkcie sú uvedené v tejto tabuľke:

key	kod	funkcia
DELETE	127	DEL
GRAPHICS	3	^C
RIGHT	9	TAB
UP	11	VT
DOWN	10	LF
LEFT	8	BS
INV.VIDEO	17	^Q
TRUE VIDEO	19	^S
CAPS LOCK	-	CAPS
EDIT	-	EXIT
<=	26	^Z
<>	27	ESC
>=	18	^R
AND	91	[
OR	93	[
STOP	126	~
NOT	124	:
STEP	92	\
TO	123	(
THEN	125	)

Samotný prenos znakov je uskutečňovaný protokolom XON/XOFF. To znamená, že vyslaním znaku ^S (19 dekadicky) sa oznamuje zariadeniu, že nie som schopný prijať dáta. Vyslaním znaku ^Q (17 dekadicky) sa oznamuje pripravenosť prijať znak.

Na ošetrovanie klávesnice je využitý pôvodný systém Spectra. Tento, počas prerušenia vyvolaného obvodom ULA, zisťuje stlačenie klávesy a ASCII kód uloží do systémovej premennej LAST_K. Stlačenie novej klávesy sa oznamuje nastavením šiesteho bitu v systémovej premennej FLAGS.

Na zobrazovanie je využitá systémová rutina, do ktorej sa vstupuje inštrukciou RST 16. Táto rutina zobrazí znak, ktorého ASCII kód je v registry A. Treba len nulovať systémovú premennú SCR_CT, aby pri popísaní celej obrazovky nedošlo k výpisu "scroll ?".

Samotné zobrazovanie je u Spectra veľmi pomalé. Preto je algoritmus programu upravený tak, že po prijatí znaku sa okamžite vyšle ^S. To by malo blokovať vysielanie ďalších znakov. Ale vzhľadom na konečnú odozvu systému sa tak nestane a prišlé znaky sa ukládajú do bufra. Ak počas doby, určenej premennou T_OUT, nepríde znak, tak nasleduje zobrazenie znakov z bufra. Vyšle sa ^Q a program sa vracia do slučky čakania na znak a kontroly stlačenia klávesy.

Zobrazenie nie je rýchle ako u terminálov určených na tento účel, ale je spoľahlivé aj pri vyšších prenosových rýchlostiach (9600 Bd).

Program neinterpretuje žiadne riadiace znaky okrem CR a TAB.

Takto upravený Sinclair používam ako terminál v systéme SM 52/11. V spojení s KERMITOM ho používam na prenos údajov na Sinclair a späť.

Výpis programu

```
                10 ;************************
                20 ;*                      *
                30 ;*  E M U L A T O R     *
                40 ;* T E R M I N A L U    *
                50 ;*                      *
                60 ;************************
                70
                80 ;Krocka Michal
                90
5C8C           100 SCR_CT EQU       23692
5C88           110 S_POSN EQU       23688
5C08           120 LAST_K EQU       23560
5C3B           130 FLAGS  EQU       23611
5C6A           140 FLAGS2 EQU       23658
00FF           150 T_OUT  EQU       #FF
001B           160 E_MAR: EQU       27
0003           170 ^C     EQU       3
0013           180 ^S     EQU       19
0012           190 ^R     EQU       18
0011           200 ^Q     EQU       17
001A           210 ^Z     EQU       26
               220
               230 ;adresy i8251
001B           240 DATA   EQU       #1B
005B           250 STATUS EQU       #5B
               260
7530           270        ORG       30000
7530           280        ENT       $
               290
               300 ;programovanie i8251
7530 AF        310        XOR       A
7531 D35B      320        OUT       (STATUS),A
7533 D35B      330        OUT       (STATUS),A
7535 D35B      340        OUT       (STATUS),A
7537 3E40      350        LD        A,64
7539 D35B      360        OUT       (STATUS),A
753B 3E4E      370        LD        A,78
753D D35B      380        OUT       (STATUS),A
753F 3E37      390        LD        A,55
7541 D35B      400        OUT       (STATUS),A
               410 ;otvorenie kanalu 2
7543 3E02      420        LD        A,2
7545 CD0116    430        CALL      #1601
               440 ;odlozenie stacku
7548 ED735676  450        LD        (SP_BU),SP
```

```
                        460
                        470 ;zobrazenie kurzora
754C 216A5C             480 CONN:  LD       HL,FLAGS2
754F CDC975             490        CALL     RES1
                        500
                        510 ;zobrazenie promptu
7552 3E0D               520        LD       A,13
7554 215475             530 START: LD       HL,START
7557 E5                 540        PUSH     HL
7558 D31B               550        OUT      (DATA),A
                        560
                        570 ;testovacia slucka
755A DB5B               580 STAR0: IN       A,(STATUS)
755C E602               590        AND      2
755E C2F975             600        JP       NZ,CODE
7561 3A3B5C             610        LD       A,(FLAGS)
7564 E620               620        AND      %100000
7566 28F2               630        JR       Z,STAR0
                        640
                        650 ;spracovanie KEY
7568 3A3B5C             660 KEY:   LD       A,(FLAGS)
756B CBAF               670        RES      5,A
756D 323B5C             680        LD       (FLAGS),A
7570 3A0B5C             690        LD       A,(LAST_K)
                        700 ;najdenie znaku v tabulke
7573 0610               710        LD       B,SIZE/3
7575 218375             720        LD       HL,TABU
7578 BE                 730 FIND:  CP       (HL)
7579 2806               740        JR       Z,FIND2
757B 23                 750        INC      HL
757C 23                 760        INC      HL
757D 23                 770        INC      HL
757E 10F8               780        DJNZ     FIND
7580 C9                 790 FIND1: RET
7581 23                 800 FIND2: INC      HL
7582 E9                 810        JP       (HL)
                        820
7583 04                 830 TABU:  DEFB     4
7584 182E               840        JR       CTR_S
7586 05                 850        DEFB     5
7587 182F               860        JR       CTR_Q
7589 06                 870        DEFB     6
758A 1832               880        JR       C_LOCK
758C 07                 890        DEFB     7
758D 1843               900        JR       BREAK
758F 0C                 910        DEFB     12
7590 1846               920        JR       DEL_K
7592 0F                 930        DEFB     15
7593 1846               940        JR       CTR_C
7595 C3                 950        DEFB     195
7596 1846               960        JR       BAR
7598 C5                 970        DEFB     197
7599 1846               980        JR       R_BR
759B C6                 990        DEFB     198
759C 1846              1000        JR       L_BR
759E C7                1010        DEFB     199
759F 1846              1020        JR       CTR_Z
75A1 C8                1030        DEFB     200
75A2 1846              1040        JR       CTR_R
75A4 C9                1050        DEFB     201
75A5 1846              1060        JR       ESC
75A7 CB                1070        DEFB     203
75A8 1846              1080        JR       R_ANG
75AA CC                1090        DEFB     204
75AB 1846              1100        JR       L_ANG
75AD CD                1110        DEFB     205
75AE 1846              1120        JR       B_SLA
75B0 E2                1130        DEFB     226
75B1 3E7E              1140        LD       A,"~"
75B3 C9                1150 TABE:  RET
0030                   1160 SIZE:  EQU      TABE-TABU
                       1170
75B4 3E13              1180 CTR_S: LD       A,^S
75B6 1802              1190        JR       CTR_X
75B8 3E11              1200 CTR_Q  LD       A,^Q
75BA 324576            1210 CTR_X: LD       (QQ+1),A
75BD C9                1220        RET
                       1230
                       1240 ;stlacenie CAPS LOCK
                       1250 ;zmena kurzora C<>L
75BE 216A5C            1260 C_LOCK LD       HL,FLAGS2
75C1 CB5E              1270        BIT      3,(HL)
75C3 CBDE              1280        SET      3,(HL)
75C5 3E43              1290        LD       A,"C"
75C7 2804              1300        JR       Z,CHANG
75C9 CB9E              1310 RES1   RES      3,(HL)
75CB 3E4C              1320        LD       A,"L"
                       1330 ;zmena kurzora
75CD 325A76            1340 CHANG: LD       (CRR),A
75D0 186F              1350        JR       W_HL2
                       1360
                       1370 ;navrat do BASICU
75D2 ED785676          1380 BREAK  LD       SP,(SP_BU)
75D6 FB                1390        EI
75D7 C9                1400        RET
75D8 3E7F              1410 DEL_K: LD       A,127
75DA C9                1420        RET
75DB 3E03              1430 CTR_C: LD       A,^C
75DD C9                1440        RET
75DE 3E7C              1450 BAR:   LD       A,"|"
75E0 C9                1460        RET
75E1 3E5D              1470 R_BR:  LD       A,"]"
75E3 C9                1480        RET
75E4 3E5B              1490 L_BR:  LD       A,"["
75E6 C9                1500        RET
75E7 3E1A              1510 CTR_Z: LD       A,^Z
75E9 C9                1520        RET
75EA 3E12              1530 CTR_R: LD       A,^R
75EC C9                1540        RET
75ED 3E1B              1550 ESC:   LD       A,27
75EF C9                1560        RET
75F0 3E7D              1570 R_ANG: LD       A,"}"
75F2 C9                1580        RET
75F3 3E7B              1590 L_ANG: LD       A,"{"
75F5 C9                1600        RET
75F6 3E5C              1610 B_SLA: LD       A,"\"
75F8 C9                1620        RET
                       1630
75F9 3E13              1640 CODE:  LD       A,^S
75FB D31B              1650        OUT      (DATA),A
75FD 216176            1660        LD.      HL,BUFER
7600 DB1B              1670 COD1:  IN       A,(DATA)
7602 77                1680        LD       (HL),A
7603 23                1690        INC      HL
7604 06FF              1700        LD       B,T_OUT
7606 DB5B              1710 COD2:  IN       A,(STATUS)
7608 E602              1720        AND      2
760A 20F4              1730        JR       NZ,COD1
760C 10F8              1740        DJNZ     COD2
760E 361B              1750        LD       (HL),E_MAR
                       1760
                       1770 ;vypis bufra
7610 215F76            1780        LD       HL,PREBU
7613 7E                1790 W_HL:  LD       A,(HL)
7614 23                1800        INC      HL
7615 FE1B              1810        CP       E_MAR
7617 2828              1820        JR       Z,W_HL2
7619 FE09              1830        CP       9
761B 280F              1840        JR       Z,W_TAB
761D FE08              1850        CP       8
761F 38F2              1860        JR       C,W_HL
7621 FE0A              1870        CP       10
7623 28EE              1880        JR       Z,W_HL
7625 D7                1890 W_HL1: RST      #10
7626 AF                1900        XOR      A
7627 328C5C            1910        LD       (SCR_CT),A
762A 18E7              1920        JR       W_HL
762C 3A885C            1930 W_TAB: LD       A,(S_POSN)
762F 3D                1940        DEC      A
7630 3D                1950        DEC      A
7631 E607              1960        AND      7
7633 28DE              1970        JR       Z,W_HL
7635 47                1980        LD       B,A
7636 3E20              1990 W_TA1: LD       A," "
7638 D7                2000        RST      #10
7639 AF                2010        XOR      A
```

```
763A 328C5C    2020          LD        (SCR_CT),A
763D 10F7      2030          DJNZ      W_TA1
763F 18D2      2040          JR        W_HL
7641 CD4776    2050 W_HL2:   CALL      KURZ
7644 3E11      2060 QQ:      LD        A,^Q
7646 C9        2070          RET
               2080
7647 215876    2090 KURZ:    LD        HL,K_STR
764A 7E        2100 W_STR:   LD        A,(HL)
764B FE1B      2110          CP        E_MAR
764D C8        2120          RET       Z
764E D7        2130          RST       #10
764F AF        2140          XOR       A
7650 328C5C    2150          LD        (SCR_CT),A
7653 23        2160          INC       HL
7654 18F4      2170          JR        W_STR
               2180
7656           2190 SP_BU:   DEFS      2
7658 1201      2200 K_STR:   DEFB      18,1
765A 4C        2210 CRR:     DEFB      "L"
765B 1200081B  2220          DEFB      18,0,8,E_MAR
765F 2008      2230 PREBU:   DEFB      32,8
7661 00        2240 BUFER:   DEFB      0
```

Pozn. red.: Škoda že nové významy tlačítek jsou přiřazeny jinak, než u velice rozšířené Lamačovy implementace CP/M. Ostatně rutiny pro obsluhu klávesnice a výpis na obrazovku z Lamačovy CP/M by šly velmi dobře použít pro zde popisovanou aplikaci.

DOPISY - OHLASY - DOPISY - OHLASY - DOPISY - OHLASY - DOPISY - OHLASY
DOPISY - OHLASY - DOPISY - OHLASY - DOPISY - OHLASY - DOPISY - OHLASY
DOPISY - OHLASY - DOPISY - OHLASY - DOPISY - OHLASY - DOPISY - OHLASY

ZVÝŠENÉ POŽIADAVKY NA UČITEĽA

(Meca) v spravodaji 3/89 pod jednou z hviezdičiek * dokonca na dvoch miestach konštatuje, že len obťažne získavate informácie o mimopražskom dianí a vyzýva ...(To dvojí opakování zprávy způsobili v tiskárně - pozn. red.)

Taktiež chcem reagovať na príspevok v rubrike Metakomunikácia - elzet - v článku Počítač vážňom učiteľov. Chcem podporiť konštatovanie, že všeobecne na školách naplatia v článku uvedené výroky pedagogických pracovníkov: "Žiaci sú puberťaci", "držanie počítačov pod kľúčom" a pod.

V našom mestečku sa neuzatvárame do seba, o čom svedčí bohatá spolupráca širokého okolia pri zapájaní detí a mládeže do VTR. Dobre známym je vydarený I. festival vedy a techniky v ČSSR, ktorý sa uskutočnil v Šali. Z nedávno odvysielaných televíznych aktualít (28.8.) bolo zrejmé, že spolupracujeme aj so susednými okresmi. Veď Martin Pasztor a Miro Petrásek pod vedením pedagóga zo SPŠ Nové Zámky sú tvorcami školského robota Oskar, ktorý bol vystavený v Paríži v predvečer osláv 200-ročnice Francúzskej revolúcie. (Martin tam dostal odmenu 6 000 frankov.)

Dobrá je spolupráca s SOUCH, DPM a vôbec s ľuďmi, ktorí sledujú nove trendy v technike a snažia sa ich obetavo aplikovať na československé podmienky. V poslednom dvojtýždeníku o vede a technike ZENIT č. 23-24 uverejňuje RNDr. M.Kečkés reportáž: Priemer doslat k.o. - je to O.K. Uvádza tu riešenia, ako v malom mestečku (Šaľa) sa snažia ísť dopredu.

V oblasti vedy a techniky mladých najlepšie poznám situáciu na SPŠCH v Šali.

Po dohode CŠV SZM s riaditeľstvom školy bol na Celoškolskej konferencii SZM dňa 8. decembra 1986 schválený štatút klubu vedeckotechnickej činnosti detí a mládeže, plán práce a rozpočet. Registráciou klubu na OV SZM v Galante bol schválený jeho názov ORBITÁL.

Úlohou klubu je najmä úzko nadväzovať na školskú výuku, obohacovať a rozvíjať získané vedomosti v škole s dôrazom na praktické riešenie problémov, ďalej napomáhať i intenzifikácii a racionalizácii šk. výuky, k urychľovaniu kvalitného materiálnotechnického vybavenia školy modernou technikou, informáciami a metodickými materiálmi.

Pohnútkou k vytvoreniu klubu technického zamerenia bolo sledovanie záujmu (spočiatku len snaživejších) detí a mládeže o plnšie poznanie a zvládnutie niektorých prírodovedných a technických okruhov, než sú tieto premietnuté do širokokoncipovaných uč. plánov škôl, a tiež čiastočne obmedzených možností diferencovaného prístupu v súčasných mnohopočetných triedách.

Organizátori si boli vedomí toho, že zavádzaním klubov vedecko-technickej činnosti sa vyrovnáva hlboký rozpor medzi možnosťami školy a záujmom niektorých žiakov, znižuje sa znevažovanie poslania školy v očiach mládeže, vyrovnáva sa oneskorovanie realizácie požiadaviek vonkajšieho okolia a vnútorných podmienok žiakov v škole.

Každodenná bohatá činnosť zanietených ped. pracovníkov a žiakov sa za 4 roky rozrástla ďaleko za teritórium školy.

Ďalším záujemcom radi odovzdajú svoje skúsenosti a vážnym záujemcom o založenie takého klubu pošlú jednak Štatút a tiež Plán práce s pravidelnou i príležitostnou činnosťou, ktorý je od minulého roku vďaka riaditeľovi školy rozšírený o vedeckotechnickú prázdninovú činnosť.

Už pred desiatimi narodeninami mala SPŠCH Šaľa koncepčne dobre rozpracovaný systém podchytenia každého nadaného alebo aspoň aktívnejšieho svojho žiaka k hlbšej zainteresovanosti do plnenia vyučovacích i mimorozvrhových cieľov. K 1. októbru 1985 bol schválený a rozbehnutý Program rozvíjania účasti detí a mládeže vo vedecko-technickom rozvoji na podmienky SPŠCH v Šali.

V troch samostatných častiach, zámermi a obsahom však prelínajúcich sa, je konkrétna a terminovaná zainteresovanosť jednotlivých pracovníkov školy a domova mládeže:

I. rozvrhovaná činnosť v jednotlivých učebných predmetoch a študijných skupinách
II. záujmová činnosť individuálna a kolektívna - prírodovedné, technické a odborné krúžky, SOČ, školenia, inštruktáže, aktívy, konzultácie, propagácia VTR, výstavky, súťaže
III. pomoc a spolupráca školy s ďalšími organizáciami v smere rozvíjania účasti detí a mládeže do VTR.

Priebežné hodnotenie ukazuje, že koncepcia začleňovania žiakov do VTR je správna, realizácia zámerov je hodnotná, o aktuálnej priorite akcií sa rozhoduje na pravidelných gremiálnych poradách, a tak i popri spoločensko-vedných, kultúrnych i športovo-branných akciách nenastávajú kolízie.

Za pozornosť stojí obohatenie Programu ... o prázdninovú činnosť - letnú pracovnú aktivitu v n.p. Duslo Šaľa. Na prázdninovú prax do podniku nastúpilo niekoľko žiakov využiť (i získať) vedomosti k podnikovým počítačom. Niektorí programy či údaje iba vkladali do najnovších strojov, iní sa podielali aj na rozbore úloh a tvorbe programov.

Pre potreby vyučovania i pre záujmovú činnosť zabezpečovalo riaditeľstvo školy vhodnú prístrojovú a didaktickú techniku priebežne. Klasické elektronické obvody so súčiastkami II. generácie postupne nahrádzajú modernejšie zariadenia. V podmienkach školy sa vytvorilo za posledné 3 roky 17 elektrotechnických pomôcok pre podporu vyučova-

nia i pomôcky charakteru spotrebnej elektroniky. Nedostatok mnohých súčiastok, hoci cenove dostupných, obmedzuje progresívnu aplikovateľnosť elektroniky a brzdí potrebnú údržbu zariadení. Štyri druhy stavebnicovo riešených pomôcek boli zabezpečené prostredníctvom organizácie Zväzarm. Sú to jednak generátory, zdroje, ale i stavebnice číslicovej techniky.

Pravidelne pracujú krúžky záujmovej činnosti rôzneho zamerenia. V rámci vedecko-technickej činnosti sú usporadúvané odborné semináre a exkurzie. Pedagogickí pracovníci a žiaci školy majú možnosť využívať osobné počítače v laboratóriách školy. Podľa rozdeľovníka strojového času je v krúžkovej a individuálnej činnosti každoročne zapájaných 60 až 110 členov. Venujú sa tu aplikáciám elektroniky v zložitejších zariadeniach, najmä v základných a prídavných prostriedkoch výpočtovej techniky, prehlbovaniu základov programovania v jazyku Basic, tvorbe programov v assembleri, oboznamovaniu sa so štruktúrou zložitejších obvodov, než to predpisovali učebné osnovy a stavba jednoduchých obvodov.

V dielni tvorivosti, ktorá je sprístupnená záujemcom po celý deň, vzniklo 42 programov pre mikropočítače.

Pracovník školy získal ocenenie na celoslovenskej prehliadke v Martine za spoluautorstvo námetového videoprogramu PUM (Programátorská univerzita mládeže), ktorý bol vytvorený so zámerom zefektívniť snahu v odstraňovaní počítačovej negramotnosti detí, mládeže i dospelých.

V živej klubonej činnosti sú dvere otvorené pre pionierov, kde sú k dispozícii rôzne technické návody, príručky, časopisy a denná tlač.

Na tunajšej škole sa viackrát konal prázdninový Krajský kurz programátorov - pionierov. Na troch pracoviskách (SPŠCH, na miestnom gymnáziu, DPM) sa o postupoch v jazyku Žofka rozhodovalo aj v tomto roku.

Vo využívaní školských robotov sú len vzácne výsledky. V Šali už pred piatimi rokmi použili detskú stavebnicu Merkur a poznatky o mikroprocesore 8080 aplikovali na prvých školských mikropočítačoch PMI-80. K tomu žiaci využili na ovládanie posuvov aj krokový motor z n.p. MEZ Náchod.

Naznačil som niektoré možnosti ako podchytiť nadaných jednotlivcov so záujmom o najnovší trend vo vede a technike, ako sa môže realizovať požiadavka spojenia viacerých odborností, ako možno v našich podmienkach podnecovať k sústavnému využívaniu modernej techniky a tak mladým vytvoriť predpoklady pre ich úspešnú budúcu prax.

Pedagogická činnosť je namáhavá, kde však vedenie školy vytvorí optimálne podmienky, je aj práca väčšinou radostná.

Ing. Pavol Ružanský
Novomeského 18
927 00 Šaľa

Vážená redakce, Světlá nad Sázavou 18.8.1989

po obdržení poslední zásilky se zpravodajem Mikrobáze (tj. čísly 3 a 4) jsem si s velkým zájmem pročítal jednotlivé příspěvky. Až do okamžiku, než jsem narazil na článek Počítač vězněm učitele podepsaný zkratkou - elzet. Až do této chvíle jsem považoval všechny příspěvky podepsané právě touto zkratkou za jedny z nejlepších v celé Mikrobázi. Ovšem s myšlenkami obsaženými v tomto příspěvku nelze souhlasit.

Sám jsem učitel, který má ve škole výpočetní techniku na starosti. Učím na SOU zemědělském, ve kterém je vybavení na poměrně slušné úrovni. Naše počítače jsou kromě výuky využity po čtyři odpoledne v týdnu pro kroužky. Ve výuce využíváme každou možnost, kterou nám poskytují osnovy, k tomu, abychom žáky seznámili s obsluhou počítače a základy programování.

Ale abych nemluvil pouze za naše učiliště. Protože mám přehled, co se v této oblasti děje prakticky na všech zemědělských učilištích v ČSR, můžu to do jisté míry posoudit. Téměř všude, kde mají alespoň základní vybavení počítači, se najde nadšený učitel, který se ve svém vlastním volnu věnuje žákům, ale i sám se pokouší vytvářet programy. Toto všechno se děje bez nároku na jakoukoliv odměnu. Najdou se samozřejmě i tací, kteří o práci s počítači nemají zájem. Nelze je však odsuzovat, protože společenské ocenění práce učitele dnes oproti minulosti silně pokulhává. Tím samozřejmě nemám na mysli jenom to, že učitel po 25 letech praxe je platově pod průměrem republiky. (Ani nemluvě o těch mladších!).

Dalším z problémů, které uvádí zmíněný článek, je tzv. zamřížování počítače. Ovšem to vám potvrdí prakticky každý učitel (alespoň učitel z učiliště), že ve škole najde jenom velice málo žáků, které by mohl klidně pustit samotné k počítači a být si jistý, že se nic nestane (žáku ani počítači). Připočteme-li k tomu všeobecné známou poruchovost počítače IQ 151, nelze se těmto učitelům divit. Autor by potom asi nechtěl jít za tohoto učitele "sedět".

Dalšími problémy jsou i příprava učitele na výuku programování a samotné programové vybavení jednotlivých druhů počítačů. První problém úzce souvisí s nedostatkem literatury na našem trhu, i s tím, že není mnoho učitelů, kteří se s programováním setkali při svém studiu.

Takže závěrem bych chtěl říci, že se domnívám, že autor -elzet- zevšeobecňuje naprosto neoprávněně. Bylo by vhodné, když si chce hrát na "počítačového Komenského", aby přišel s konstruktivní kritikou a poradil jakým způsobem nejlépe ve školách využívat počítače.

S pozdravem

Otakar Březina
Na Bradle 957
582 91 Světlá nad S.

ZE SVĚTA

Budoucnost v americké nemocnici

Woodhull Medical and Mental Health Center předznamenává systém pro rozpoznávání hlasu, který podstatně ulehčuje práci radiologům zmíněné nemocnice. Náklady na konpletní systém včetně programového vybavení jsou 25000 USD. Podle amerického časopisu MIS Week je systém vybaven slovníkem 1000 slov a jeho spolehlivost je 75 - 95%. Systém na základě diktátu radiologa vypracuje kompletní zprávu, kterou lze vytisknout na běžné tiskárně. Lze definovat jednoduché výrazy, které však systém ve zprávě interpretuje jako dlouhé, často se vyskytující fráze. Například slova Srdce, Normal, která radiolog nadiktuje systému, jsou v textu nahražena libovolně dlouhým popisem srdce v normálním stavu. Zpráva je ihned k dispozici ošetřujícím lékařům bez zbytečného čekání na písařky a následnou opravu chyb. Podobných systémů instaloval výrobce, americká firma Kurtzweil, již 60 kusů.

Opět Desktop Publishing

Britský časopis Personal Computer World vydává, zatím jako přílohu, novou publikaci DESKTOP PUBLISHING WORLD, která je věnována nejprogresivnější oblasti aplikace výpočetní techniky - publikační činnosti pomocí počítače. Obsahem přílohy jsou zkušenosti uživatelů s hard i softwarovými produkty zahraničních firem. Dostupnost: středisko VTI Svazarmu, Martinská ul.

Assembler 8086

Britský časopis Personal Computer World (PCW) zveřejňuje pravidelně na straně 194 v rubrice SubSet informace o assembleru pro mikroprocesor 8086 a krátké rutiny spolu s radami "jak na to". Všechny zveřejněné rutiny jsou přehledně zdokumentovány. Za rubrikou SubSet následuje rubrika Program File se zajímavými programy čtenářů pro různé typy počítačů.

Variace na téma
ROZŠÍŘENÍ PAMĚTI ZX SPECTRA

Jakub Vaněk

V poslední době se na stránkách Amatérského rádia, Sdělovací techniky, Mikrobáze a jiných odborných časopisů objevovaly a stále ještě objevují články, popisující různé úpravy a rozšíření paměti oblíbeného počítače ZX Spectrum. Dvě z těchto úprav se dostaly do popředí zájmu. To proto, že obě umožnují připojit pamět RAM v celém prostoru 64kB operační paměti. Díky tomu na ně mohl být implementován operační systém CP/M, který je na osmibitových počítačích vůbec nejrozsířenější.

První z nich, "Úprava adresování a zvětšení rozsahu paměti počítače ZX Spectrum" autorů ing. Pavla Trollera a Petra Císaře, uveřejněná v ST 11/87 [1], byla koncipována jako vnější přídavná jednotka, řešící přepínání paměti stylem "on/off", bez ohledu na stav datových vodičů.

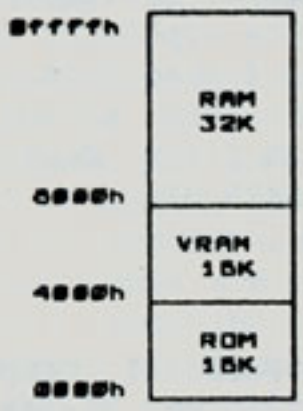
Druhá, "80K RAM pro ZX Spectrum" od Jiřího Lamače, prvně publikovaná v Mikrobázi č. 6/87 [2], řeší rozšíření jako maximálně jednoduchou vnitřní úpravu. Dalším rozšířením možností této úpravy se zabýval článek autorů Lamač, Meca, Vaněk, uveřejněný pod názvem "CP/M na ZX Spectrum" v Amatérském rádiu č. 9/88 [3].

Protože vývoj jde dál a od napsání posledně jmenovaného článku už uplynula dlouhá doba (redakci AR byl předán v lednu 1988), vracíme se ještě jednou k této problematice několika variantami zapojení. Vznik těchto variant byl ovlivněn především nečekaným značným vzrůstem cen pamětí 256kb na začátku loňského roku. Tehdy se totiž pro řadu uživatelů staly verze 272/528kB cenově nedostupné. Zato se objevily dotazy typu: "Co s demontovanými pamětmi 32k? - Nešla by udělat verze s dvěma sadami pamětí 64k?" a pod. A protože téměř všechno jde, když se ví jak - zde jsou žádané varianty a ještě něco navíc.

U všech popsaných modifikací je zajištěna vzájemná kompatibilita zdola nahoru.

POPIS ZAPOJENÍ

Nejprve si stručně připomeneme, jak je v ZX Spectru adresována paměť a jaké jsou možnosti rozšíření. Mapa paměti standardního ZX Spectra je na obr. 1a. Od adresy 0000h do 3FFFh je standardně pamět ROM. V oblasti 4000h až 7FFFh je fyzicky oddělená pamět RAM, ve které je od 4000h do 5AFFh umístěna obrazová paměť, obsluhovaná obvodem ULA. V dalsí části textu bude tato paměť nazývána VRAM. Od 8000h do 0FFFFh je hlavní pamět RAM.



obr. 1a
ZX Spectrum 48k
1 x 32kB

Výběrové obvody pro ROM a VRAM jsou zaintegrovány v obvodu ULA, pro RAM jsou u verzí 2 až 4 tvořeny obvody 74LS32 (IC23) a 74LS00 (IC24), u verze 5 a 6 (ZX Spectrum +) obvodem ULA2 (IC27). (Pro obvody uvnitř ZX Spectra bude dále použito označení ICxx, obvody popisovaného zapojení budou značeny jako IOxx).

Činnost výběrových obvodů si popíšeme tabulkou:

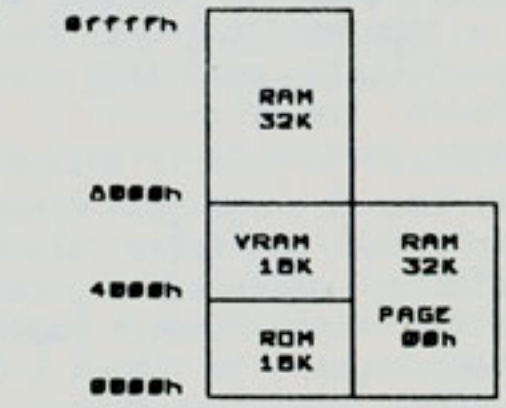
A15	A14	ROM	VRAM	RAM
0	0	aktivní	odpojená	odpojená
0	1	odpojená	aktivní	odpojená
1	x	odpojená	odpojená	aktivní

V jednotlivých zapojeních se využívá stav, kdy A15 se do výběrových obvodů přivádí překódovaná v závislosti na jednotlivých módech činnosti.

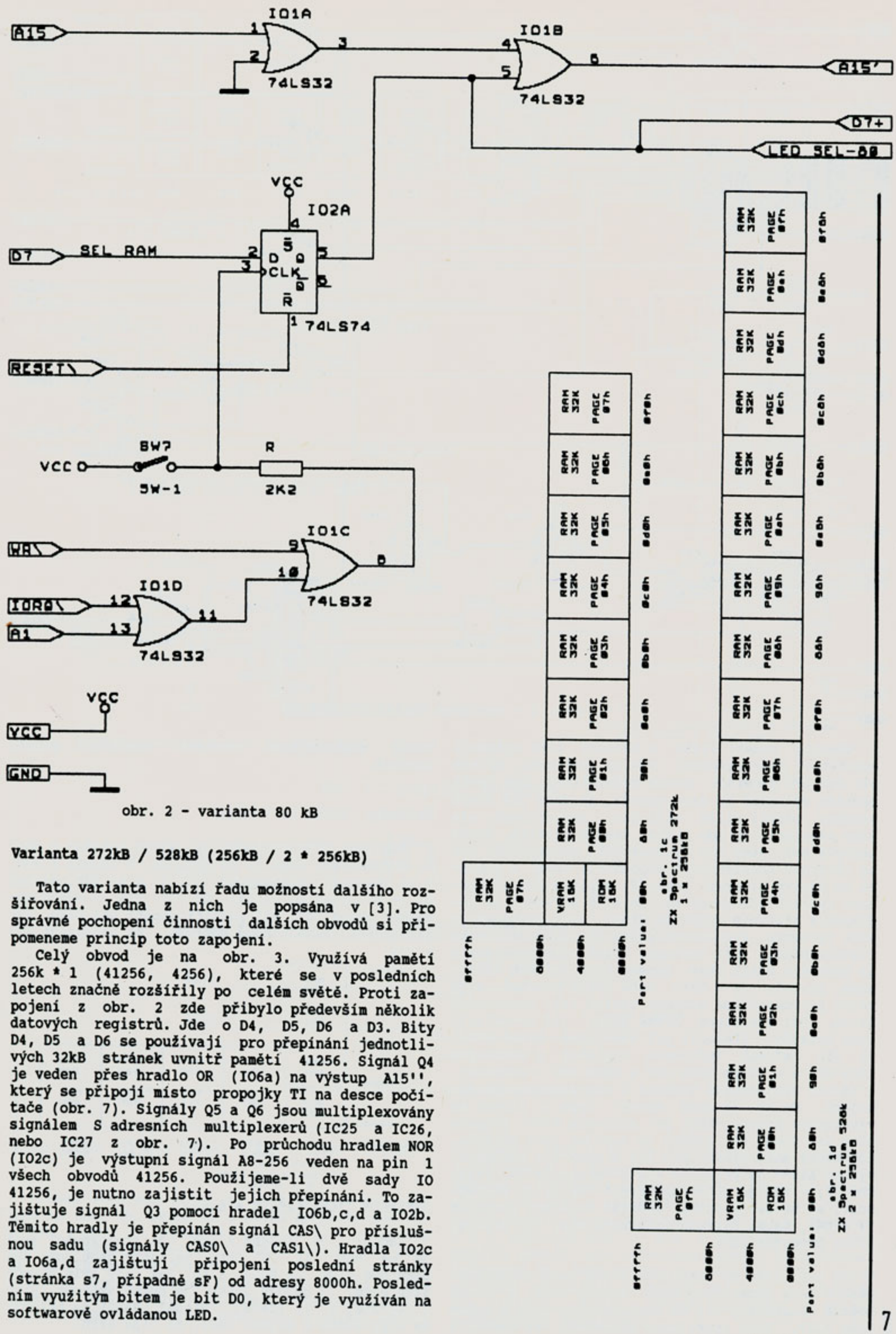
Pro řízení stránkování je použit port na adrese 253 (0FDh), který je u Spectra 128 standardně použit ke stránkování a není používán žádnou továrně vyráběnou periferií s vyjímkou Betadisku (s tím si však všechna popsaná zapojení rozumí - podrobnosti najdete v návodu na stavbu Betadisku). Tato volba umožňuje lineární adresaci pomocí adresy A1, což pronikavě zjednodušuje zapojení.

Varianta 80kB

Princip tohoto zapojení je použit ve všech dalších variantách. Jedná se o obvod na obr. 2, popsaný v [2], určený pro použití pamětí 64k * 1 (4164, 2164, K565RU2). Ten v sobě sdružuje výběrovou a stránkovací část. Výběrová část je tvořena dvěma hradly OR (IO1c,d). Při aktivním signálu IORQ\=0 a A1=0 je vzestupnou hranou signálu WR\ zapsána informace z datové sběrnice do registru D (IO5). Rezistor, zařazený mezi výstupem výběrového obvodu (IO1c) a hodinovým vstupem registru (IO5 pin 9), slouží spolu se spínačem k zablokování zápisu do registru. Při jeho sepnutí se počítač chová jako bez úpravy. Signál RESET\ zajišťuje po připojení napájecího napětí, nebo po resetu počítače nastavení výchozího stavu, tj. módu 48k. Z registru, do kterého byl zapsán stav D7, je signál (Q7) veden do hradla OR (IO1b). Na druhý vstup hradla je vedena posílená A15. Při signálu Q7=log.0 odpovídá A15' (výstup hradla IO1b) stavu A15. Pokud ovšem zapíšeme do registru D7=log.1 => Q7=log.1, je výstup A15'=log.1. Tím je po připojení A15' místo A15 na obvod ULA, na výběrové obvody RAM a na systémový konektor, zajištěn přístup do paměti RAM od 0000h do 0FFFFh, tj. celých 64kB (viz výše uvedená tabulka). Mapa paměti je na obr. 1b.



obr. 1b
ZX Spectrum 60k
1 x 64kB

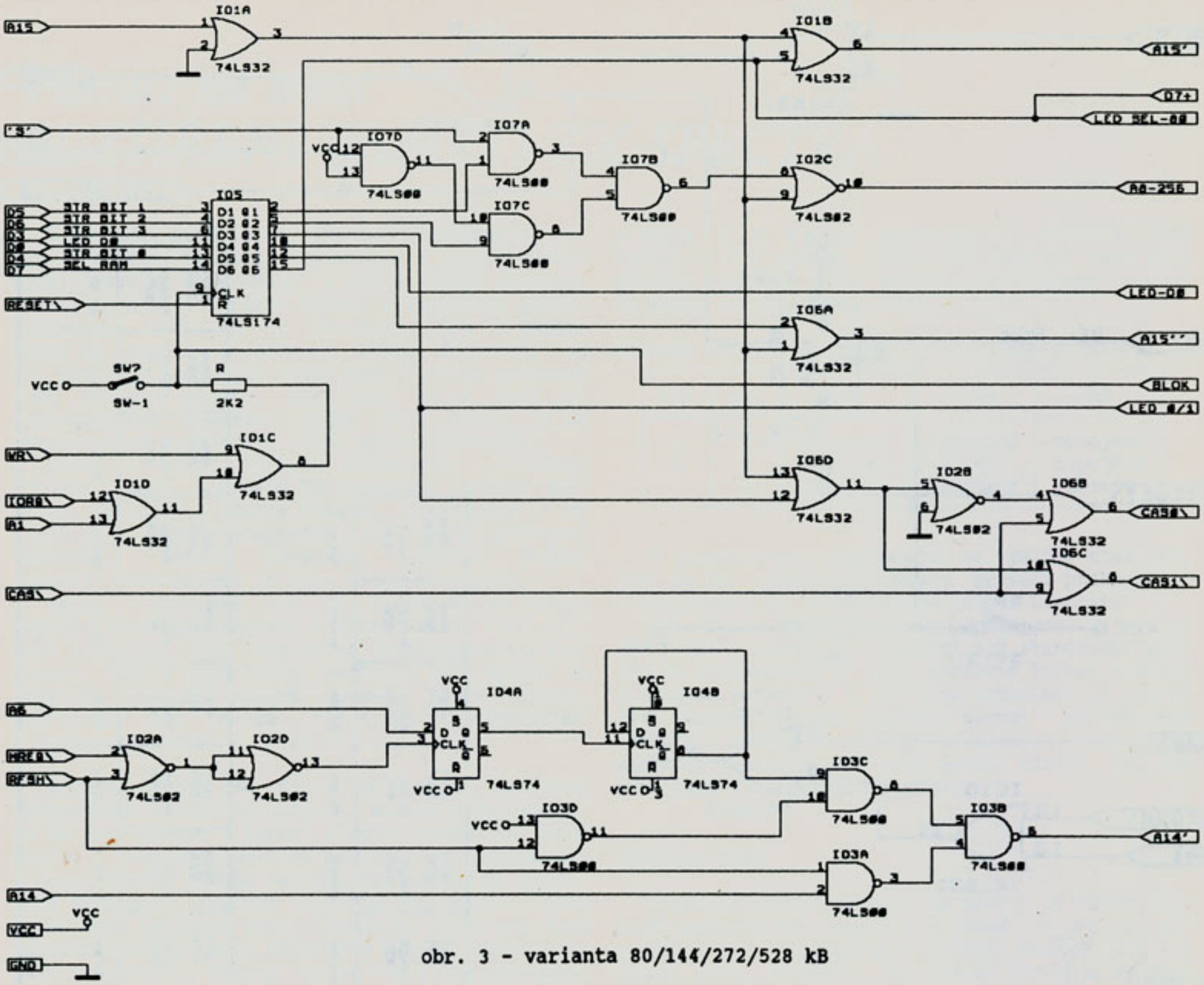


obr. 2 - varianta 80 kB

Varianta 272kB / 528kB (256kB / 2 * 256kB)

Tato varianta nabízí řadu možností dalšího rozšiřování. Jedna z nich je popsána v [3]. Pro správné pochopení činnosti dalších obvodů si připomeneme princip toto zapojení.

Celý obvod je na obr. 3. Využívá pamětí 256k * 1 (41256, 4256), které se v posledních letech značně rozšířily po celém světě. Proti zapojení z obr. 2 zde přibylo především několik datových registrů. Jde o D4, D5, D6 a D3. Bity D4, D5 a D6 se používají pro přepínání jednotlivých 32kB stránek uvnitř pamětí 41256. Signál Q4 je veden přes hradlo OR (IO6a) na výstup A15'', který se připojí místo propojky TI na desce počítače (obr. 7). Signály Q5 a Q6 jsou multiplexovány signálem S adresních multiplexerů (IC25 a IC26, nebo IC27 z obr. 7). Po průchodu hradlem NOR (IO2c) je výstupní signál A8-256 veden na pin 1 všech obvodů 41256. Použijeme-li dvě sady IO 41256, je nutno zajistit jejich přepínání. To zajišťuje signál Q3 pomocí hradel IO6b,c,d a IO2b. Těmito hradly je přepínán signál CAS\ pro příslušnou sadu (signály CAS0\ a CAS1\). Hradla IO2c a IO6a,d zajišťují připojení poslední stránky (stránka s7, případně sF) od adresy 8000h. Posledním využitým bitem je bit D0, který je využíván na softwarové ovládanou LED.



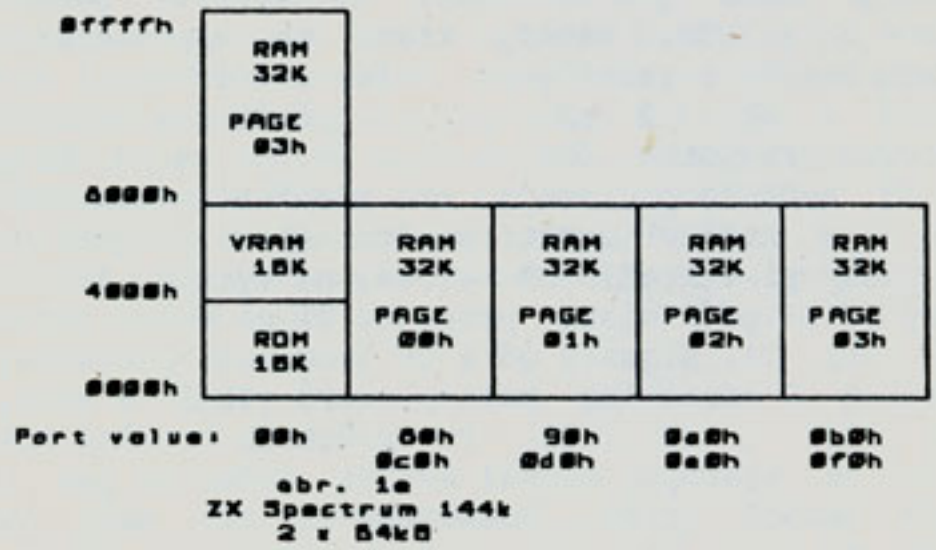
obr. 3 - varianta 80/144/272/528 kB

Všechny paměti 41256 a některé 4164 vyžadují osmibitový občerstvovací cyklus (refresh), přitom ale Z80 CPU generuje pouze sedmibitový. Jak tento problém vyřešit? Je sice pravda, že dnes již skoro všechny paměti 41256 mají tzv. autorefresh, ale jeho zapojení by zde vyšlo značně složité. Proto nezbývá než osmý bit vygenerovat hardwarově. Zde se většinou konstruktéři dopouštějí chyb. Z pohledu na časový diagram (obr. 8) vyplývá, že přítomnost refresh informace na dolních adresách je indikována aktivním signálem RFSH\. Její platnost je však výrobcem zaručována pouze po dobu aktivního signálu MERQ\. Při sestupné hraně tohoto signálu je nevhodné měnit stav osmého bitu refresh informace, protože je v této době čtena pamětmi a je velmi pravděpodobná možnost vzniku tzv. hazardních stavů. Při vzestupné hraně MREQ\ již nevznikají tyto hazardy. (Pozor!! při vzestupné hraně RFSH\ je již refresh informace neplatná u některých Z80A CPU a u všech Z80B CPU a Z80H CPU - ověřeno). Z těchto skutečností vychází i zde publikované zapojení. Do KO (klopný obvod) typu D (IO4a) se vzestupnou hranou MREQ\ pri aktivním RFSH\ přepíše stav A6. Druhý KO (IO4b) je zapojen jako dělička. IO3 tvoří multiplexer pro A14 a osmý bit refreshe. Výstup A14' je připojen místo propojky OKI (viz, obr. 7). Zde je nutno upozornit na další skutečnost, a to že je nutné dodržet zapojení propojek TI a OKI, neboť refresh informace musí být přítomná na propojce OKI.

Na jaře 1988 se stalo ve světě cosi neočekávaného - ceny pamětových čipů začaly růst, a již na podzim dosahovaly tří až šestinásobku ceny z konce roku 1987. Tato nepříjemná skuteňost si vynutila vznik úspornějších variant rozšíření paměti Spectra.

Varianta 144kB (2 * 64kB)

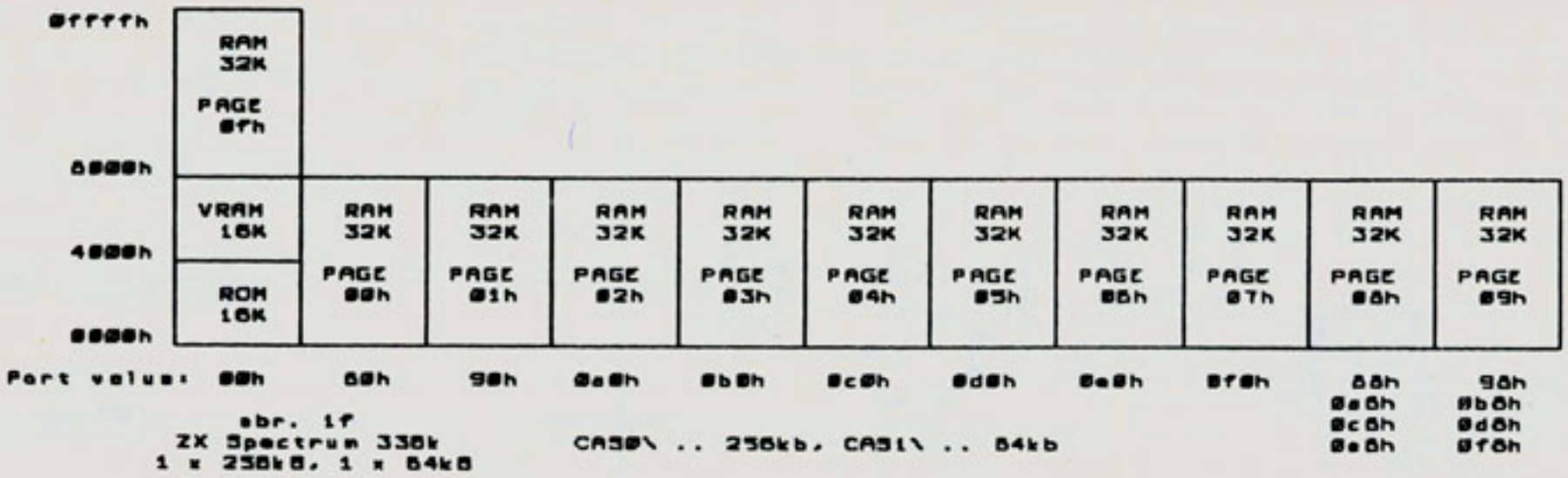
Zapojení této varianty je skoro totožné s obr. 3. Stránkovací mapa je na obr. 1e. V obvodu z obr. 3 se nezapojí signály D5, D6, S a A8-256, signál D3 je připojen na signál D5 CPU. Tím je možno vynechat obvod IO7. Při použití pamětí RAM 64kb, vyžadujících sedmibitový refresh, se neosazují IO3 a IO4.



obr. 1e
ZX Spectrum 144k
2 x 64kB

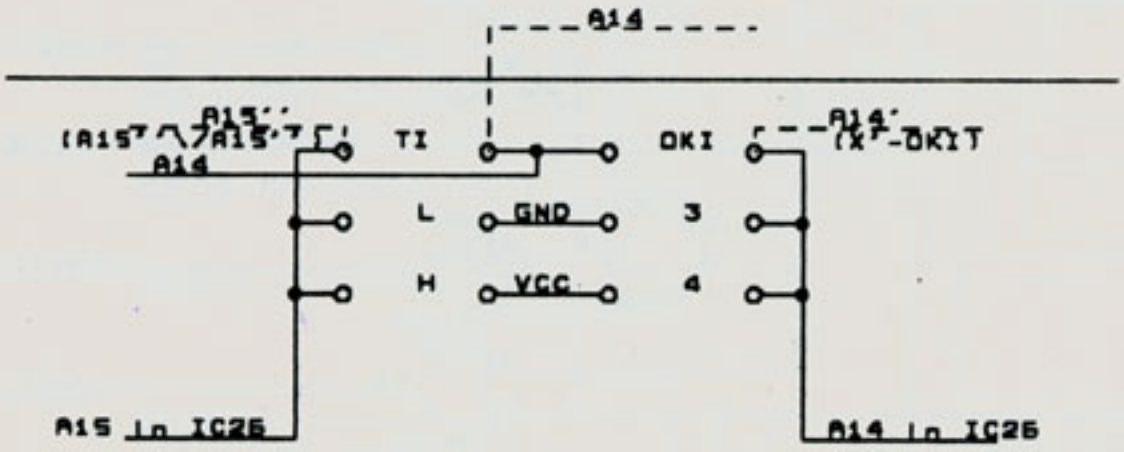
Varianta 336kB (256kB + 64kB)

Zapojení je shodné s obr. 3; stránkovací mapa je na obr. 1f. U této varianty jsou opět dva paměťové čipy na sobě, a to tak, aby byly obvody 64kb dole. U obvodů 256kb je nutno vyhnout piny 1 (A8-256) a 15 (CAS0\). Signál CAS0\ se připojí na čipy 256kb a signál CAS1\ na čipy 64kb.

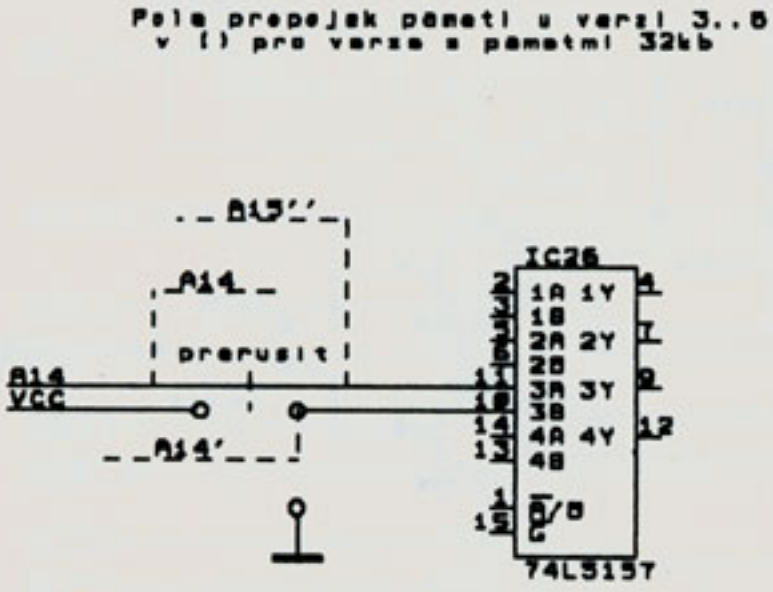


obr. 1f
ZX Spectrum 336k
1 x 256kB, 1 x 64kB

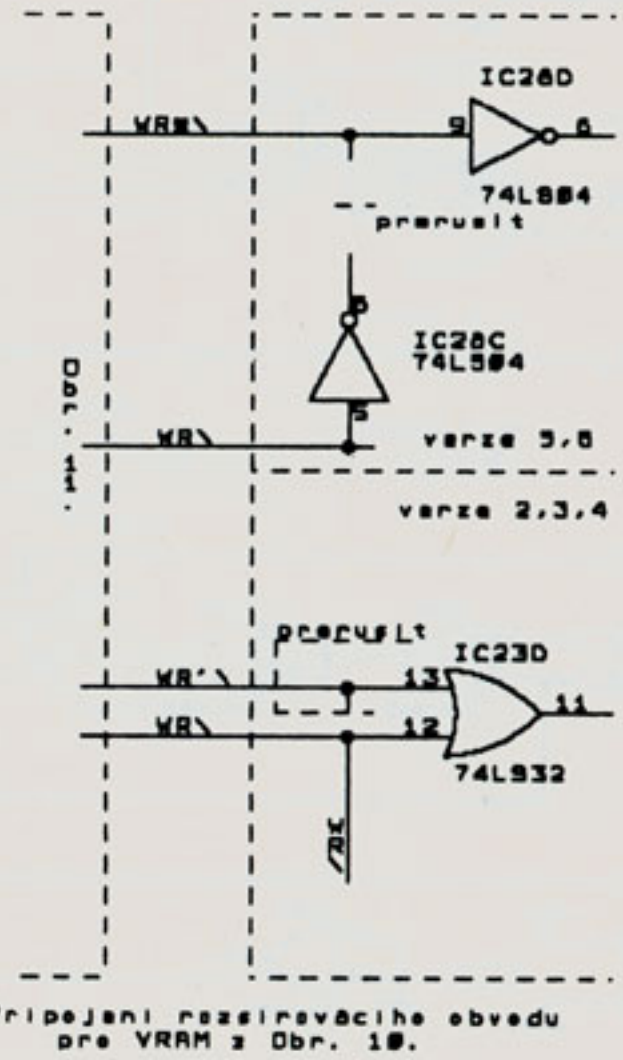
CAS0\ .. 256kb, CAS1\ .. 64kb



Pole propojek pameti u verzi 3..6
v () pro verze s pametmi 32kb

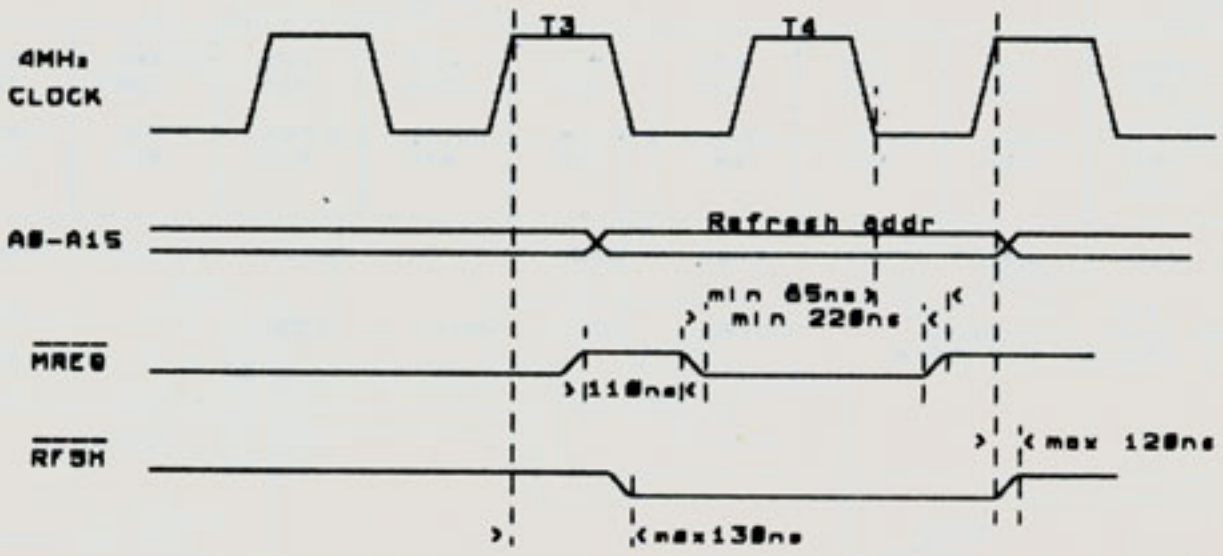


Pole propojek pameti u verze 2



Pripojeni rozsirovaciho obvodu
pro VRAM z Obr. 10.

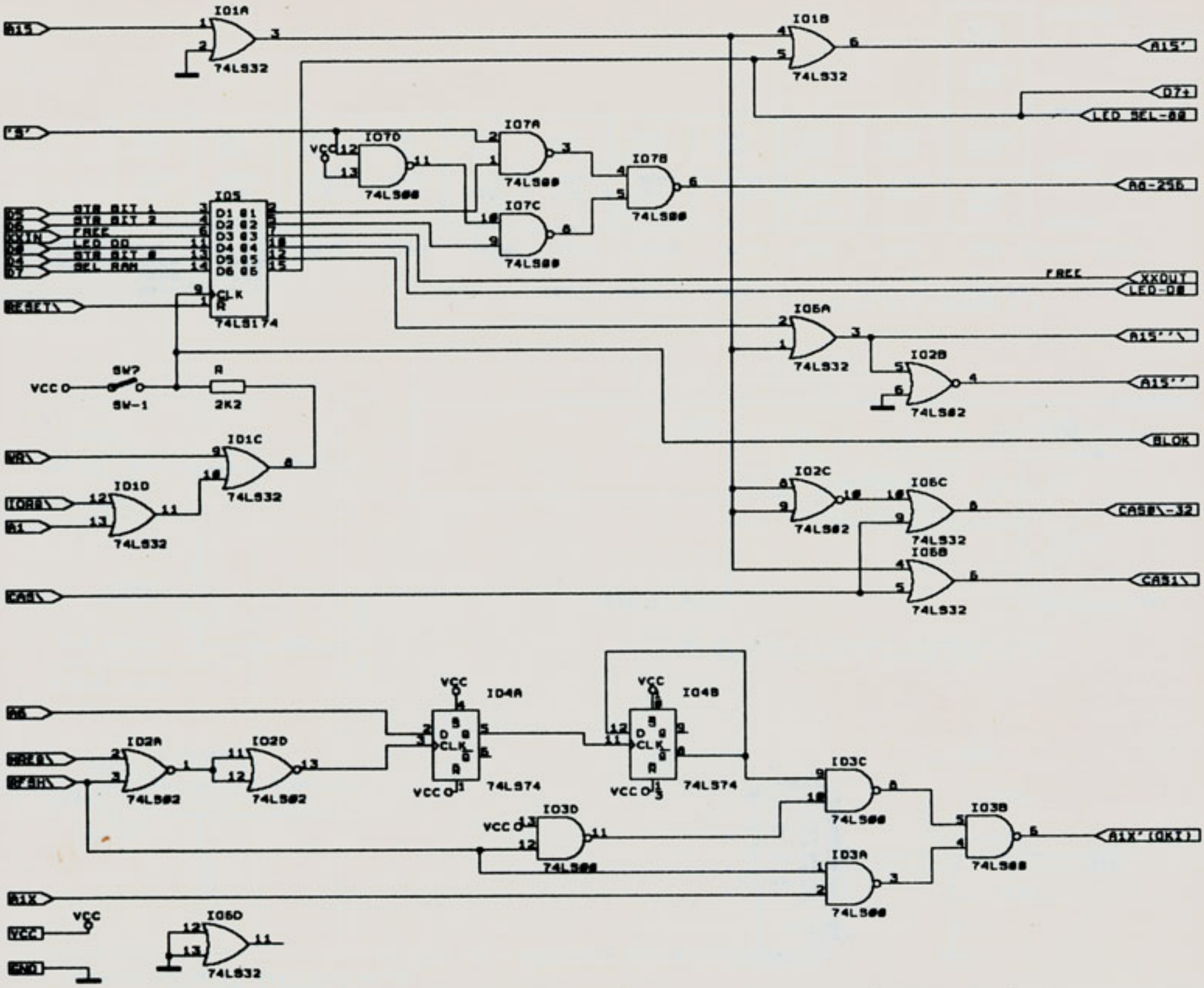
obr. 7 - propojky, připojení



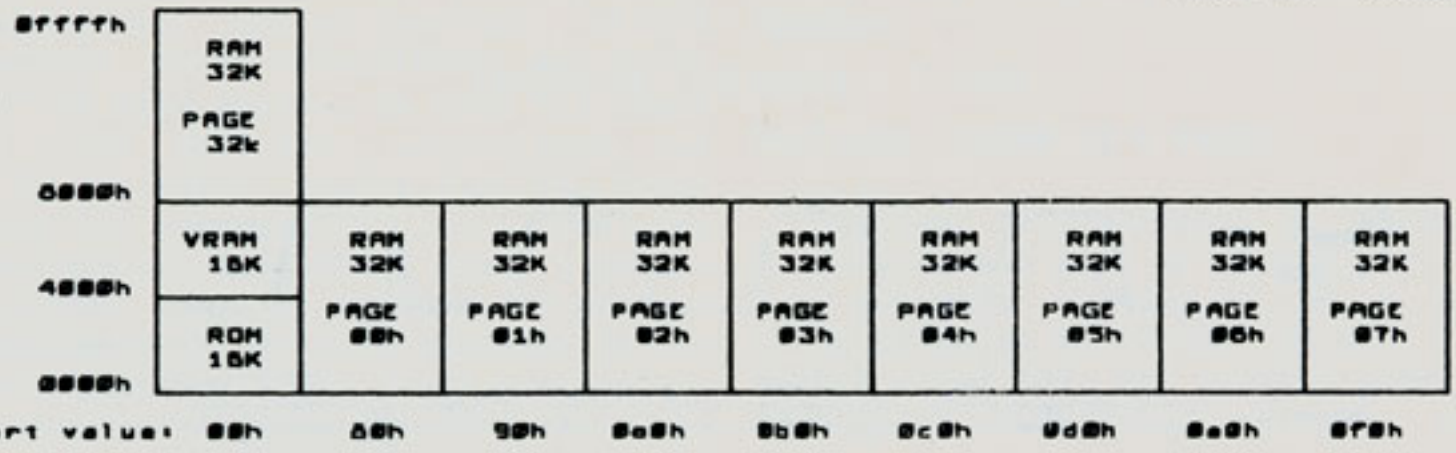
obr. 8 - časování signálů procesoru

Verze 112kB / 304kB (32kB + 64KB / 32kB + 256kB)

Schema zapojení je na obr. 4 a jeho stránkovací mapa na obr. 1h. Jedná se o mírně modifikované zapojení z obr. 3. Vytváření signálu A15' je shodné jako u předešlých variant. Signál A8-256 jsou pouze multiplexované výstupy Q5 a Q6. Při variantě 112kB je vypuštěn obvod IO7 a signál A8-256 se nevytváří. Zcela jiný je princip vytváření signálů CAS0\ a CAS1\. Signálem CAS0\ jsou vybírány původní obvody 4532 a je aktivní pouze při A15=log.1 a CAS\=log.0. Signál CAS1\ je aktivní při A15=log.0, Q7=log.1 a CAS\=log.0. Vazba Q7 je nepřímo přes A15' a výběrové obvody. Signál A15'' je nutno umistit místo propojek L, H, 3, 4. Propojce L a 3 odpovídá A15''\, H a 4 odpovídá A15''. Při použití obvodu pro vytváření osmého bitu refresh infornace je jeho výstup na propojce OKI a vstup odpovídá propojkám v pořadí OKI, 3, 4 signálům A14, A15''\, A15''. Zbytek zapojení je shodný s předešlými variantami.



obr. 4 - varianta 112/304 kB

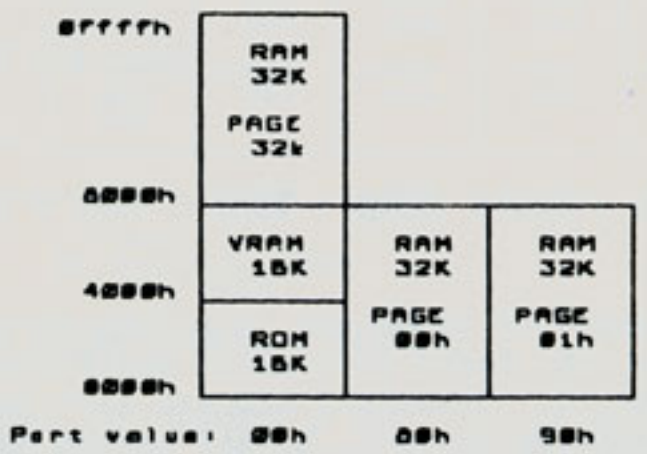


obr. 1h
ZX Spectrum 304k
1 x 256kB, 1 x 32kB
CAS0\ .. 32kb, CAS1\ .. 256kb

Uvedeným zapojením bylo a je vytýkáno, že neumožňují nahradit ROM pamětí RAM se zakázaným zápisem, do které by bylo možno nahrát např. obsah ISOROM, LECROM a jiných [4]. Následující dvě zapojení však tento problém řeší.

Napřed zapojení pro ty, kteří se spokojí s velikostí paměti 80kB. Jeho schema je na obr. 6. Obvod umožňuje připojení paměti RAM misto ROM a zákaz zapisu do ní.

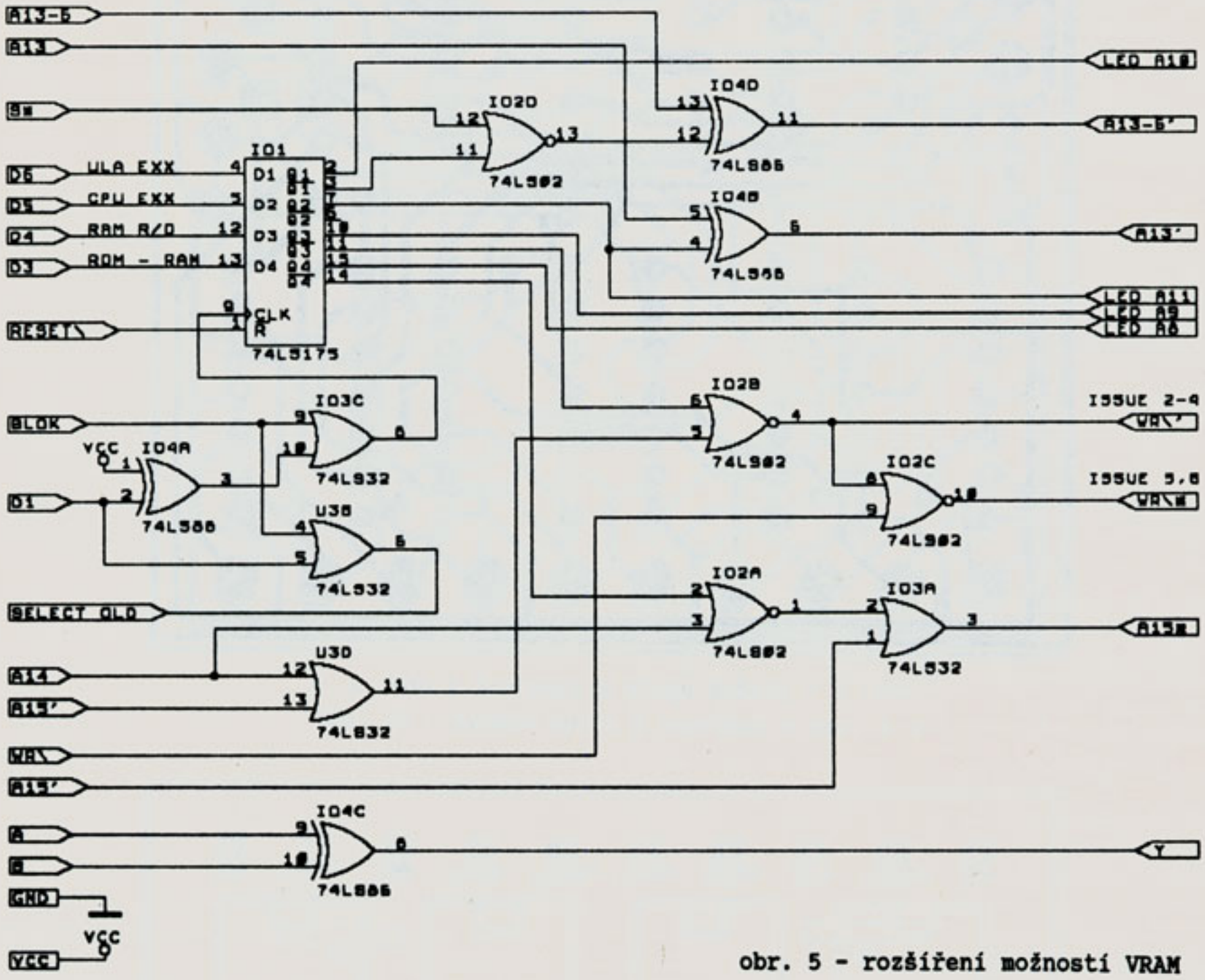
Druhý obvod, podle obr. 5, slouží k rozšíření výše uvedených zapojení s větší pamětí. Jsou zde, kromě dvou již popsaných módů, ještě další, umožňující provádět některé obrazové efekty, známé ze ZX Spectra 128. Umožňují totiž přístup do dvou oblastí videoram, jejichž zobrazení je nožné prohazovat. Bit D6 prohazuje z hlediska obvodů ULA



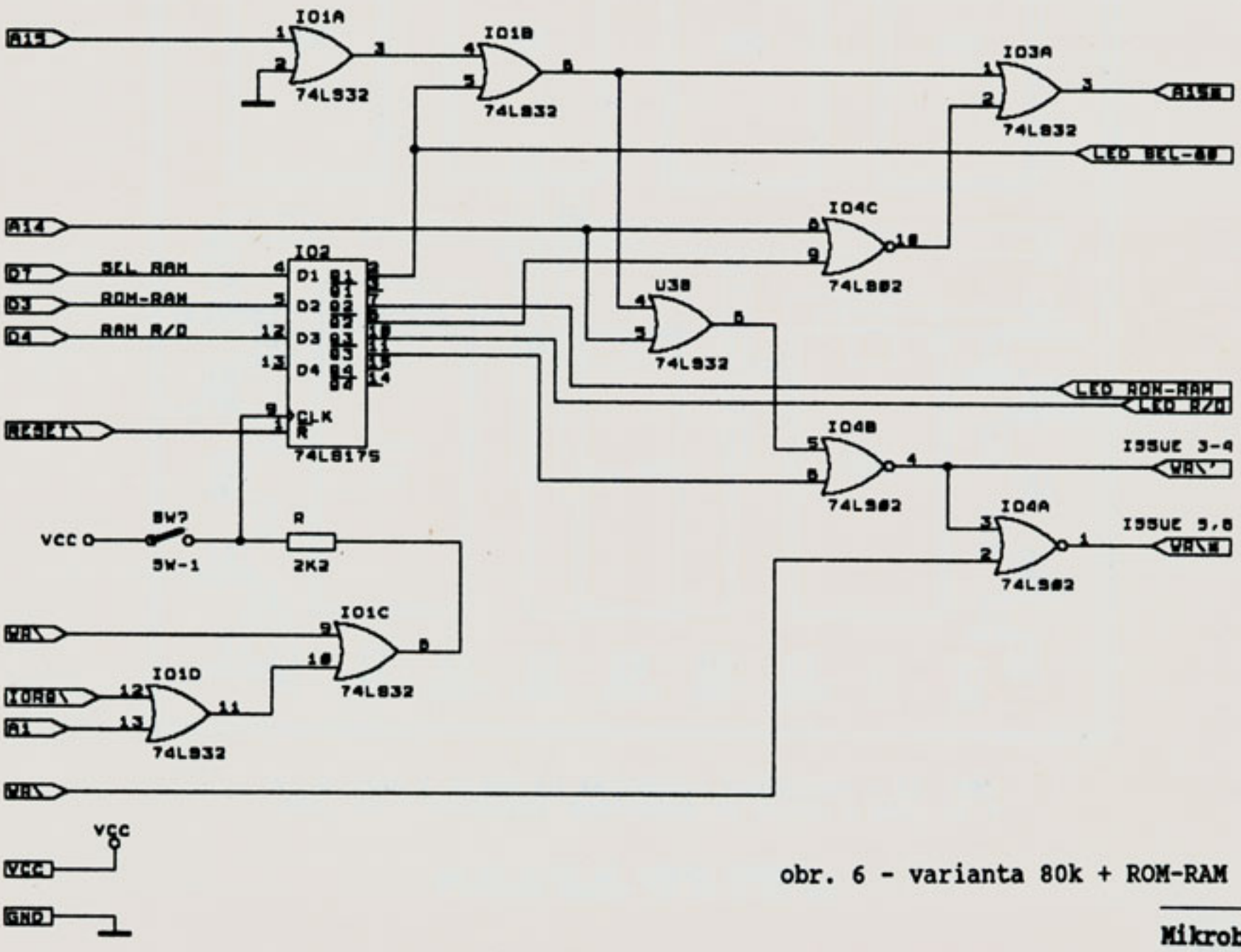
obr. 1g
ZX Spectrum 112k
1 x 64kB, 1 x 64kB
CAS0\ .. 32kb, CAS1\ .. 64kb

i CPU začátek obrazovky 4000h <=> 6000h (prohození půlek paměti VRAM) a bit D5 tuto záměnu provádí pouze z hlediska CPU. Tak je možné zobrazovat jednu obrazovku a ve druhé může zatím probíhat kreslení jiného obrázku. Protože počet potřebných ovládacích bitů již přesáhl osm, je zvolen bit D1 za rozhodující - při úrovni L je zachována původní stránkovací funkce portu a při H na D1 je zvolen registr nových módů.

Návrh desek s plošnými spoji je na obr. 9, 10 a 11. Rozmístění součástek je zřejmé z obr. 12, 13, a 14. Zapojení z obr. 5 doplňuje zapojení z obr. 3, nebo 4. Celek je pak řešen jako dvě oddělené destičky. Jejich umístění v klasickém "gumovém" ZX Spectru je už velmi problematické. Velká je určena do oblasti mezi CPU a ULA, malá by se mohla vejít vedle obvodu ULA, před modulátor. S umistěním v ZX Spectru+ problém není.

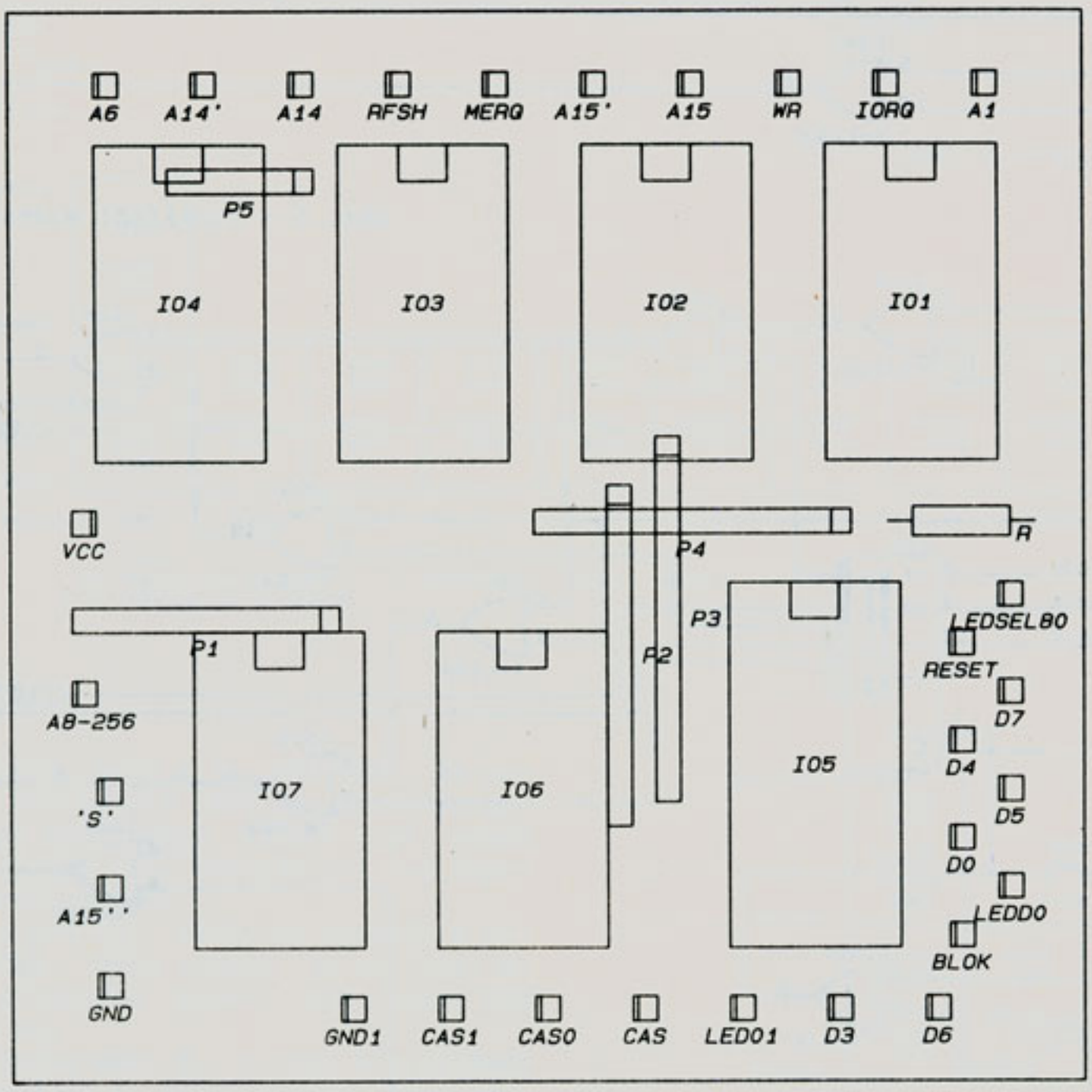


obr. 5 - rozšíření možností VRAM



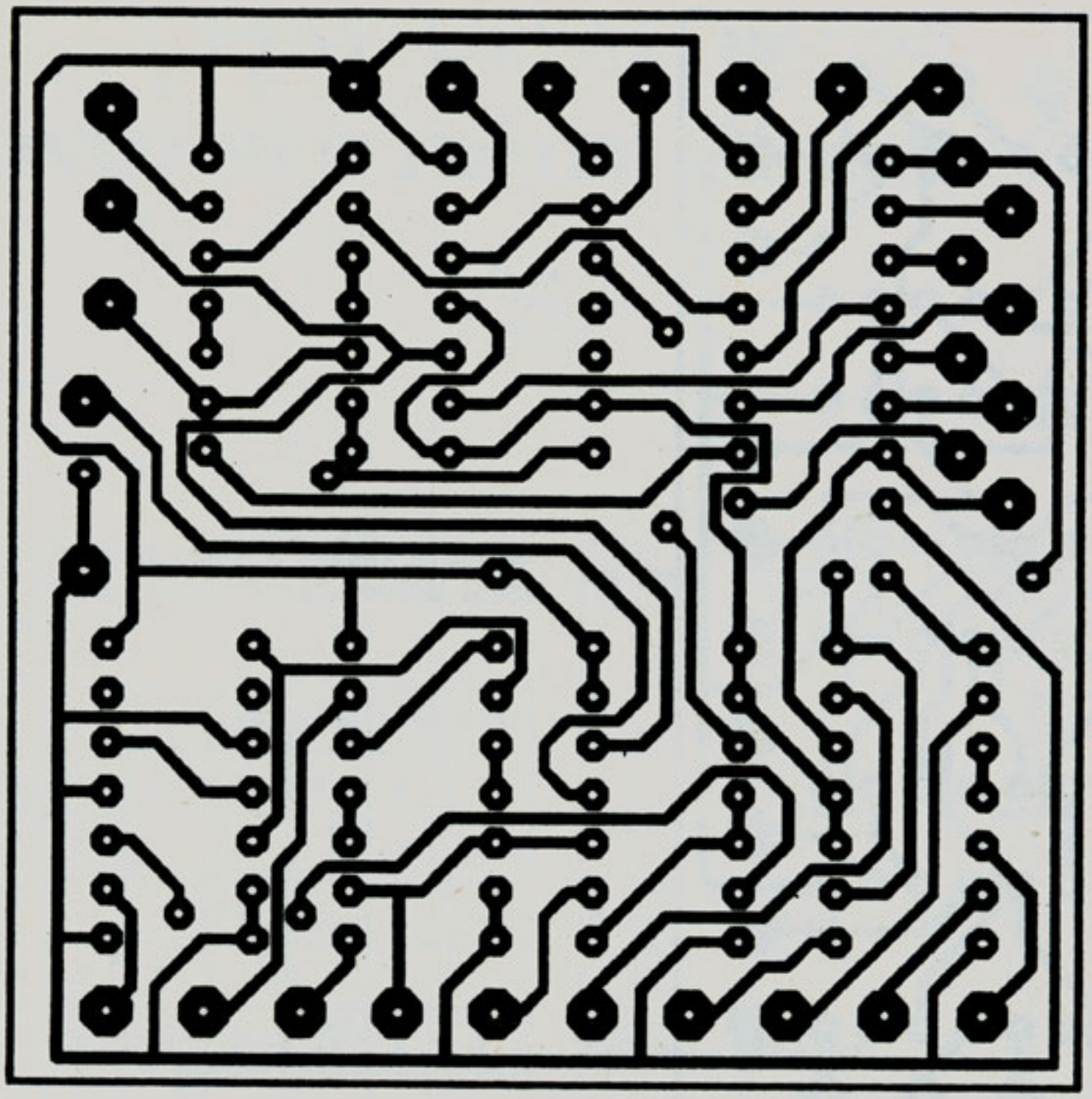
obr. 6 - varianta 80k + ROM-RAM

obr. 9 - plošné spoje k zapojení z obr. 3 *

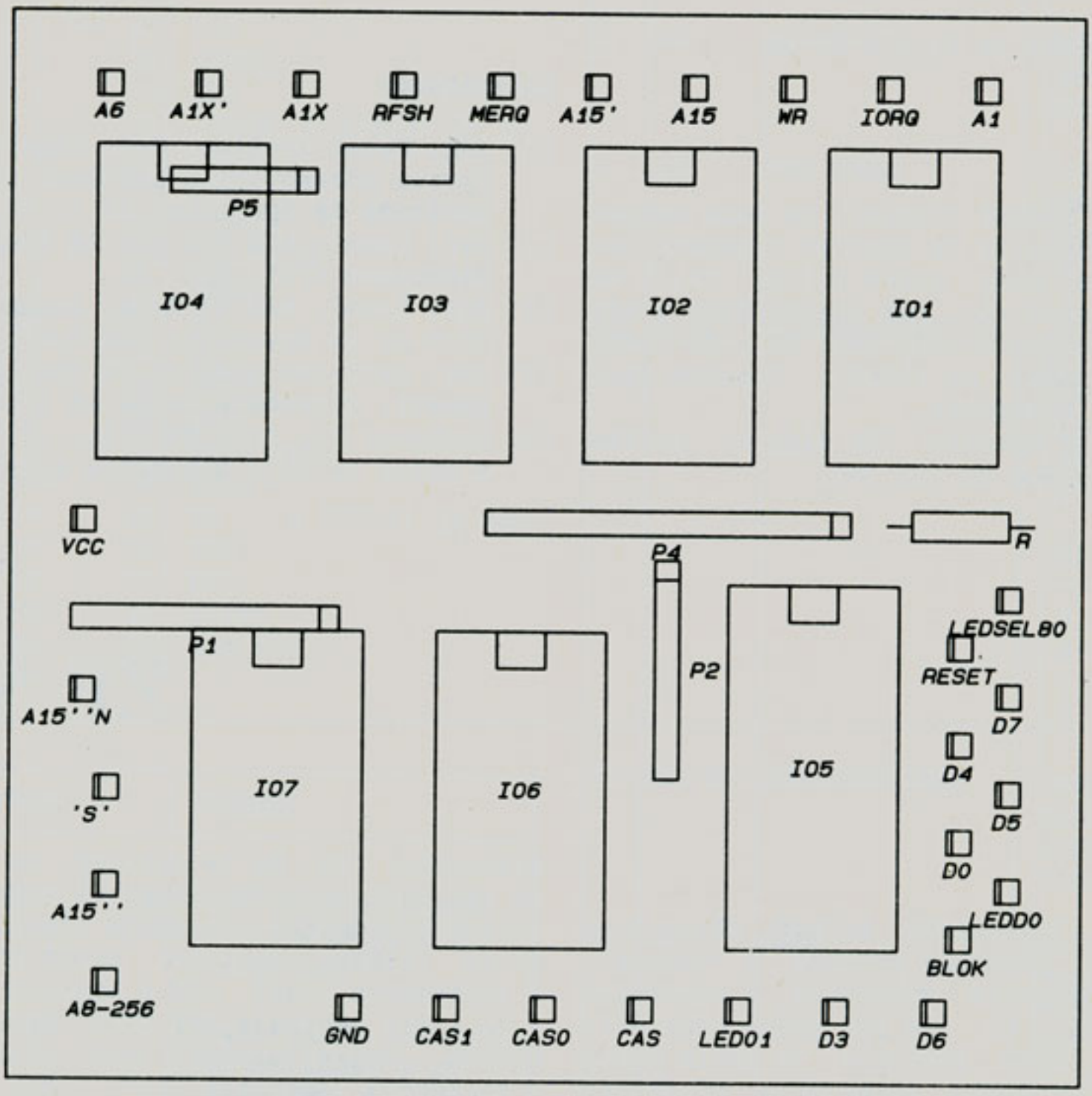


obr. 12 - rozložení součástek na desce z obr. 9 *

* rozměr desky je 55 x 55 mm

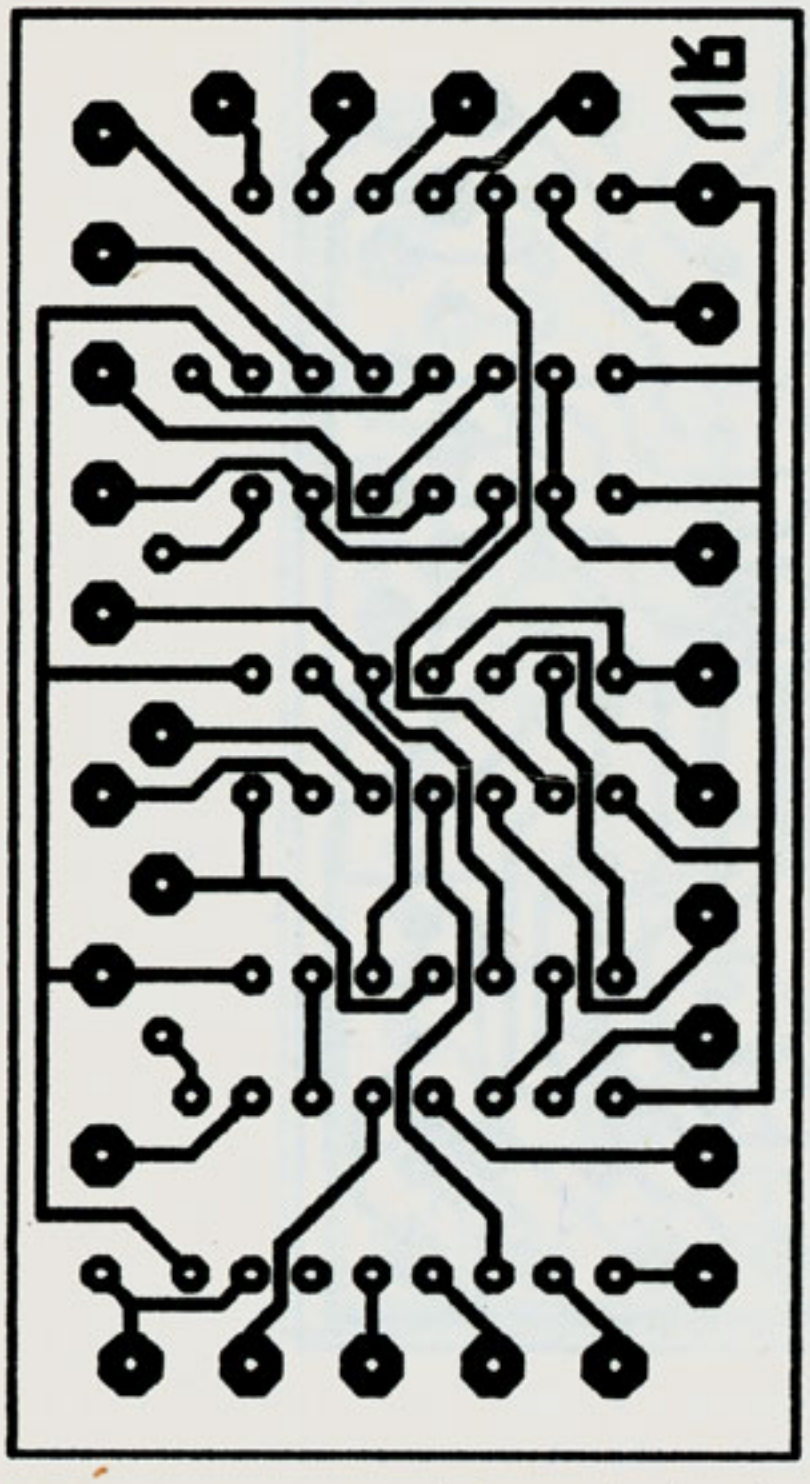


obr. 10 - plošné spoje k zapojení z obr. 4 *



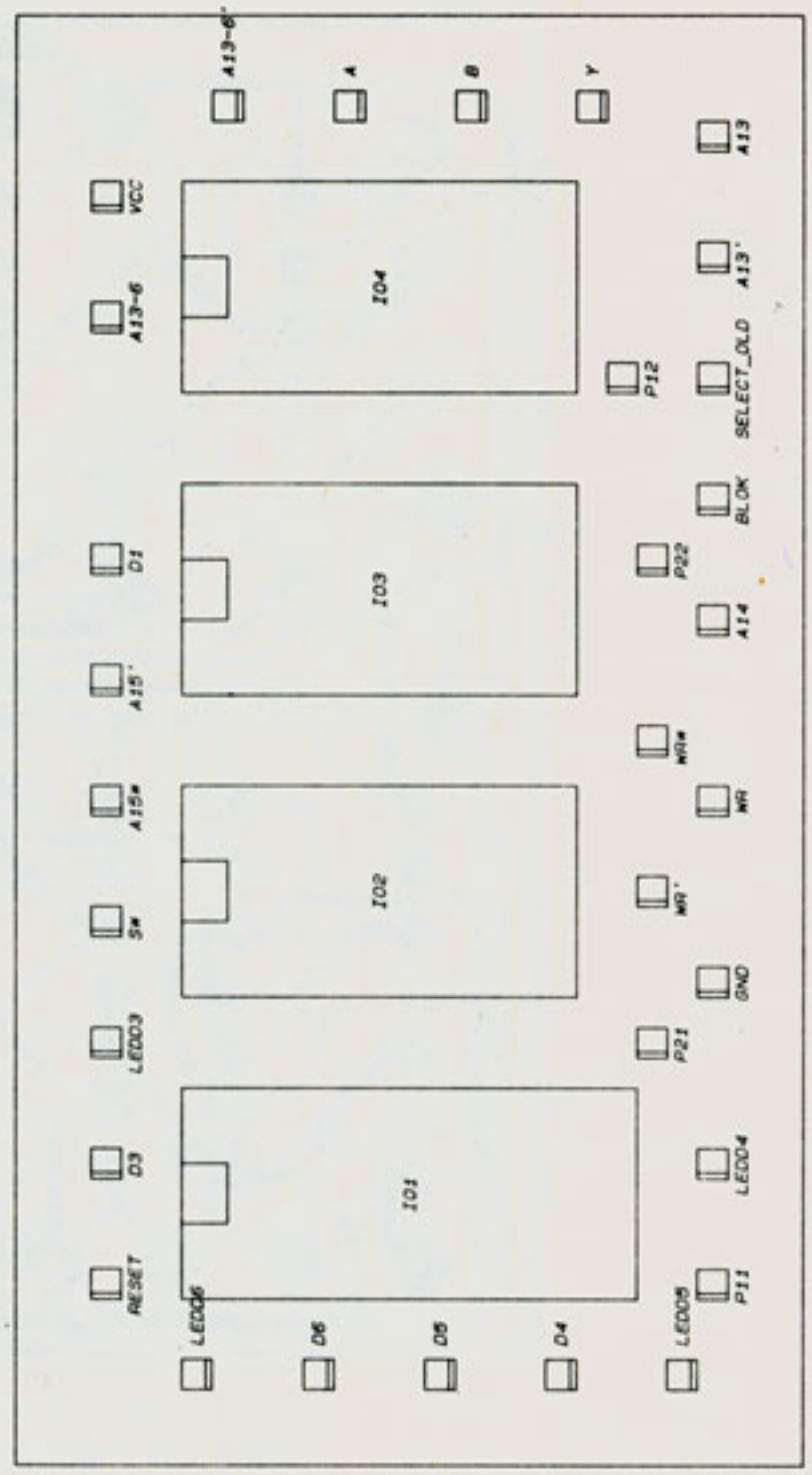
obr. 13 - rozložení součástek na desce z obr. 10 *

* rozměr desky je 55 x 55 mm



obr. 11 - plošné spoje k zapojení z obr. 5

rozměr desky je 32,5 x 60 mm



obr. 14 - rozložení součástek na desce z obr. 11

Co říci závěrem? Lidské fantazii se meze nekladou - již existuje několik ZX Specter s pamětí neuvěřitelných 1040kB, tedy právě tolik, kolik ná Atari Mega ST a vlastně víc, než značná část u nás se vyskytujících počitačů řady IBM PC (těm končí většinou operační paměť na 640kB). Otázkou však je, zda to již není absurdita.

Jak připojit k ZX Spectru 1Mb čipy si povíme zas někdy jindy (ověřeno to již je).

Literatura:

[1] ing. Pavel Troller, Petr Císař: Úprava adresování a zvětšení rozsahu paměti počítače ZX Spectrum; ST 11/87, str. 417
[2] Jiří Lamač: 80K RAM pro ZX Spectrum; Mikrobáze 6/87, str. 41
[3] Jiří Lamač, Daniel Meca, Jakub Vaněk: CP/M na ZX Spectrum; ARA 9/88, str. 337
[4] ing. Ladislav Sieger: ZX Spectrum s rozšířenou pamětí; Mikrobáze 10/88 až 3/89
[5] rozhovor "Je toho 80 kilo..."; Mikrobáze 10/88 a ohlas na něj; Mikrobáze 4/89

..... ZE SVĚTA

DynaBook

První definice "dynamické knihy" Alana Kaye zněla: malý, lehký, přenosný počítač s "mocným", inteligentním programovým vybavením a obrovskou záznamovou kapacitou - to vše pro průměrného uživatele. DYNABOOK firmy Scenario je zatím pouze náznakem cesty k produktu, splňujícím představy Alana Kaye beze zbytku. Na první pohled vypadá jako přenosný počítač třídy PC, avšak na rozdíl od klasického "portejblu", nemá obrazovku pevně spojenou se základní jednotkou. Nemá ani klávesnici - displej typu LCD (720 x 400) nahrazuje totiž klávesnici tím, že je citlivý na dotek a proto je na základní jednotku napojen volně pomocí kabelu. Základní jednotka obsahuje klasický mikroprocesor 80286 (10MHz), operační paměť RAM 640 kilobajtů a systémovou paměť ROM s kapacitou 256 kilobajtů - ta totiž obsahuje AMI BIOS, MS-DOS a drajvry pro CD-ROM. Nejzajímavější jsou však záznamová média - disketová jednotka 3.5" / 720 KB, pevný disk 3.5" s kapacitou 20 megebajtů a nakonec lahůdka - CD-ROM Hitachi se záznamovou kapacitou 550 megabajtů. Tyto jednotky lze připojit celkem čtyři (!). Popisovaným parametrům odpovídá i cena: 3750 GBP. Popis: PCW 5/89 - str. 152.

CP/M · CP/M · CP/M

Postavte si s námi diskový řadič

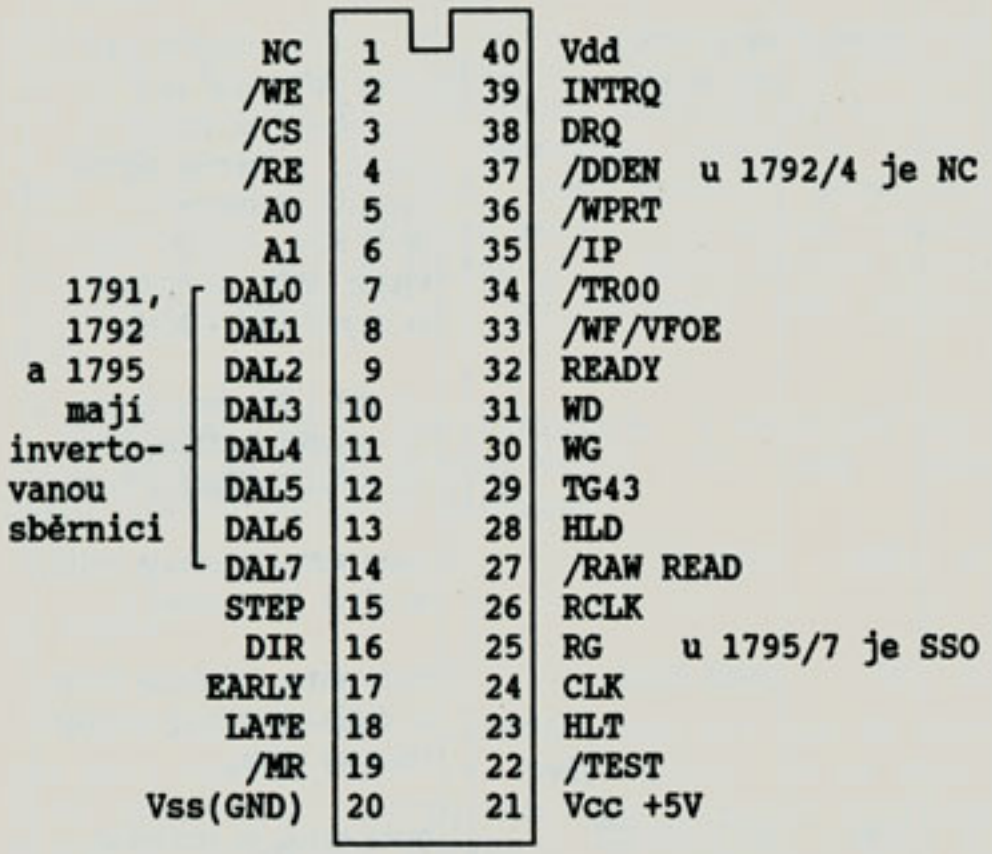
/5/

(pokračování)

Daniel Meca

Abychom si mohli lépe povídat o funkci diskového řadiče, uvedu napřed část katalogového listu obvodů řady WD 17xx:

FEATURES	1791	1792	1793	1794	1795	1797
Single Density FM	X	X	X	X	X	X
Double Density MFM	X		X		X	X
True Data Bus			X	X		X
Inverted Data Bus	X	X			X	
Write Precomp	X	X	X	X	X	X
Side Sel. Output					X	X



PIN	SYMBOL	NÁZEV VÝVODU		KOMENTÁŘ
1	NC	NO CONNECT		Vývod je vnitřně propojen se zdrojem předpětí. Musí zůstat volný!
19	/MR	MASTER RESET	I	Logická nula na tomto vstupu (po dobu alespoň 50 µs) způsobí reset řadiče a zapíše $03 do příkazového registru. Not ready status (bit 7 ve Status registru) je resetován během trvání resetovacího impulzu. Po skončení tohoto impulzu provede řadič RESTORE a zapíše do registru sektoru 1.
		NAPÁJENÍ		
20	Vss	GROUND	I	
21	Vcc	+5V ±5%	I	
40	Vdd	+12V ±5%	I	
		KOMUNIKACE S POČÍTAČEM		
2	/WE	WRITE ENABLE	I	Logická nula na tomto vstupu způsobí přenos dat z DAL do zvoleného registru, pokud je i CS na nule.
3	/CS	CHIP SELECT	I	Logická nula na tomto vstupu způsobí výběr čipu a umožní komunikaci s počítačem.
4	/RE	READ ENABLE	I	Logická nula na tomto vstupu způsobí přenos dat ze zvoleného registru na DAL, pokud je i CS na nule.
5,6	A0,A1	REGISTER SELECT LINES	I	Tyto vstupy určují které registry budou připojeny pro čtení a zápis na DAL. Směr toku dat je řízen vstupy /RE a /WE. Způsob adresování je tento: /CS A1 A0 /RE /WE 0 0 0 Status Command 0 0 1 Track Track 0 1 0 Sector Sector 0 1 1 Data Data
7-14	DAL0-DAL7	DATA ACCESS LINES	B	Osmibitová oboustranná sběrnice. Přenáší data, řízení a status. Směr toku dat je řízen vstupy /RE a /WE.
24	CLK	CLOCK	I	Na tento vstup musí být přiveden bodinový kmitočet se střídou 50% pro účely vnitřního časování. Pro mechaniky 8" je zapotřebí kmitočet 2MHz ±1%, pro 5¼" 1MHz ±1%.
38	DRQ	DATA REQUEST	O	Tento výstup s otevřeným kolektorem indikuje, že jsou v DATA registru připravena

PIN	SYMBOL	NÁZEV VÝVODU		KOMENTÁŘ
				data při operaci čtení, nebo že je DATA registr prázdný při operaci zápis. Signál je resetován, když je řadič obsloužen počítačem čtením, případně zápisem do datového registru.
39	INTRQ	INTERRUPT REQUEST	O	Tento výstup s otevřeným kolektorem je nastaven při ukončení každé operace a je resetován, když je přečten STATUS registr, nebo je zapsáno do řídícího registru.
KOMUNIKACE S MECHANIKOU FLOPPYDISKU:				
15	STEP	STEP	O	Na tomto výstupu se objevují impulzy pro každý krok.
16	DIRC	DIRECTION	O	Na tomto výstupu se objeví log. 1, když se má krokovat směrem dovnitř a log. 0, pro směr ven.
17	EARLY	EARLY	O	Log. 1 na tomto výstupu indikuje, že má být zapisovaný pulz posunut prekompenzačními obvody tak, aby přišel dříve.
18	LATE	LATE	O	Log. 1 na tomto výstupu indikuje, že má být zapisovaný pulz posunut prekompenzačními obvody tak, aby přišel později.
22	/TEST	TEST	I	Tento vstup se používá jen pro testovací účely. Normálně je připojen na +5V.
23	HLT	HEAD LOAD TIMING	I	Když je na tomto vstupu log. 1, předpokládá se, že je hlava přiklopena. Obvykle se odvozuje monostabilním klopným obvodem z HLD.
25	RG	READ GATE (1791,1792,1793,1794)	O	Tento výstup se používá pro synchronizaci externího datového separátoru. Po přečtení dvou (single density), nebo čtyř (double density) nulových bajtů za sebou se aktivuje.
25	SSO	SIDE SELECT OUTPUT (1795,1797)	O	Úroveň na tomto výstupu je přímo ovládána bitem "U" v příkazech typu II a III. Když je U=0, bude na SSO také 0 a naopak. Stav SSO je porovnáván s informací o straně v adresové značce. Při nesouhlasu je nastaven bit 4 ve Status registru, tedy příznak RNF. Stav SSO je změněn jen na začátku příkazů typu II a III. Při MASTER RESET je nulován.
26	RCLK	READ CLOCK	I	Na tento vstup musí být přiveden hodinový signál odvozený ze čtených dat. Je důležitá fáze vůči RAW READ.
27	/RAW READ	RAW READ	I	Vstup dat z mechaniky. Při hraně signálu sem musí být zaveden pulz log. 0.
28	HLD	HEAD LOAD	O	Výstup řídící přiklápění hlav na záznamové medium.
29	TG43	TRACK GREATER THEN 43	O	Tento výstup informuje mechaniku o tom, že hlava je na stopě s číslem větším než 43. Údaj je platný pouze během provádění příkazů Čtení nebo Zápis.
30	WG	WRITE GATE	O	Výstup je nastaven při zápisu na disketu.
31	WD	WRITE DATA	O	Pulzy 200ns (MFM), nebo 500ns (FM), určující okamžik změny magnetizace média. Obsahují adresové značky, data i bodinové pulzy pro oba formáty záznamu.
32	READY	READY	I	Tento vstup zjišťuje před začátkem vykonávání příkazů Čtení, či Zápis, zda je mechanika připravena. Pokud není, operace se neprovede a je generováno přerušení. Operace typu I jsou provedeny bez ohledu na stav vstupu READY. Stav tohoto vstupu se odráží v invertované formě v bitu 7 Status registru.
33	/WF/VFOE	WRITE FAULT VFO ENABLE	I O	Je-li WG=1, funguje jako vstup /WF. Pokud se v té době objeví na vstupu log. 0, je operace zápisu okamžitě přerušena. Je-li WG=0, funguje jako výstup /VFOE. Na výstupu se objeví log. 0 když: a) HLT i HLD jsou na log. 1 b) uplynul Settling time c) řadič čte data z disku
34	/TR00	TRACK 00	I	Tento vstup informuje řadič, že hlava je na stopě 0.
35	/IP	INDEX PULSE	I	Tento vstup informuje řadič, že čidlem mechaniky prochází indexový otvor.
36	/WPRT	WRITE PROTECT	I	Tento vstup je vzorkován při přijmutí příkazu k zápisu. Pokud je zde log. 0, je příkaz ukončen a je nastaven bit 6 ve Status registru.
37	/DDEN	DOUBLE DENSITY	I	Volba single, nebo double density. Při log. 1 je zvolena jednoduchá hustota, při log. 0 je zvolena dvojitá hustota záznamu. U 1792/4 musí tento vývod zůstat volný!

Pozn. - I=vstup O=výstup B=obousměrný

(pokračování příště)

CHIWRITER – *textový procesor s otazníkem*

RNDr. Miroslav Mandula

1. Úvod

Jednou z poměrně častých činností, při kterých použití počítače přináší své neoddiskutovatelné výhody, je zpracování textu. Lze asi tvrdit, že na téměř každém osobním počítači libovolné kategorie je implementován nějaký textový procesor. Některé z nich vlastně ani nemají právo si tak vznešeně říkat a spíše je možno je označit za lepší editory, a na druhé straně jsou takové, jejichž schopnosti se blíží k DTP (desk-top publishing) systémům.

V poslední době se u nás na počítačích pracujících pod operačním systémem MS-DOS poměrně hodně rozšířil textový procesor ChiWriter firmy Horstmann Software Design Corporation. Jedná se o jednoduchý vícefontový textový procesor pracující v grafice, vhodný zejména pro vědecké a technické texty. Na rozdíl od běžných textových procesorů umožňuje ChiWriter snadno a rychle psát různé matematické výrazy, vzorečky, tabulky, jednoduchá schémata apod., které se musely donedávna dopisovat ručně či jinou technikou.

Paradoxem ale je, že obliba ChiWriteru (můžeme-li to takto nazývat) není založena na vlastnostech, kterými je jedinečný a kvůli nimž - alespoň podle slov svého autora - vznikl, ale na faktu, že na něm lze velmi jednoduchým způsobem implementovat češtinu. Využití grafiky je základním předpokladem pro používání češtiny bez nutnosti předefinovat znakový generátor (pokud je v RAM, jako např. u karty EGA) nebo vyměnit paměť ROM v hardwarovém generátoru znaků. To ovšem nemění nic na faktu, že ChiWriter je i jinak docela dobrý textový procesor, který si zaslouží naši pozornost.

ChiWriter se u nás vyskytuje v několika verzích; dále se budu věnovat verzi 3.01, tedy v době psaní článku poslední dostupné.

2. Popis

ChiWriter je tzv. WYSIWYG (what you see is what you get) systém, drží se tedy zásady, že na monitoru prakticky vidíte to, co nakonec vytiskne tiskárna. To osobně považuji za výhodu, i když se v obci programátorské vyskytují četní odpůrci tohoto systému. Vidíte-li totiž na monitoru prakticky výsledný tvar dokumentu, značně to usnadňuje jeho případné opravy a celkovou grafickou úpravu. Ovšem zároveň je zřejmé, že od určité kvality tisku, kdy rozlišovací schopnosti tiskárny a monitoru přestávají být srovnatelné, je nutno tuto vlastnost oželet.

Co vlastně pro provozování ChiWriteru potřebujete? Jednak osobní počítač IBM PC nebo kompatibilní se dvěma disketovými jednotkami nebo s jednou disketovou jednotkou a pevným diskem, MS-DOS verze 2.0 nebo pozdější, jednu z dodávaných grafických karet a k ní příslušný monitor, alespoň 256 kB paměti a maticovou tiskárnu. V základní verzi umí ChiWriter obsluhovat CGA a devítijehličkové tiskárny kompatibilní s tiskárnami firem Epson, OKI a IBM. Pokud vlastníte grafickou kartu s vyšším rozlišením (EGA, VGA, AT&T/Olivetti, Toshiba 3100, Hercules) anebo lepší tiskárny (24 jehličkové či laserové), můžete si dokoupit diskety obsahující příslušné drivery a fonty. K základní verzi patří ještě program pro úpravy a návrh fontů (Font Designer) a integrovaný program pro kontrolu pravopisu anglicky psaných textů (spelling checker).

Aby mohl ChiWriter pracovat i s jinými znaky, než které obsahuje znakový generátor počítače, pracuje v grafickém režimu. Z tohoto důvodu musí mít zobrazované znaky nějak popsány. To je u ChiWriteru vyřešeno pomocí tabulek, které obsahují zobrazení znaků v určitém rastru, takže v rámci rozlišení daného rastrem může ChiWriter zobrazovat nejenom obyčejná písmena, ale libovolný hieroglyf, či cokoli jiného.

Každá tabulka obsahuje 94 znaků, kterým jsou přiřazeny ASCII kódy mezi 33 a 126. Výsledný znak, který se vám zobrazí na monitoru, je určen jednak příslušnou tabulkou, jednak kódem znaku, nebo jinak řečeno pozicí znaku v dané tabulce. Těchto tabulek může ChiWriter najednou použít až 20, přičemž při psaní textu je můžete libovolně "přepínat". Pokud vám to vyhovuje více, je možno si představit, že ChiWriter používá současně až 20 různých znakových generátorů po 94 znacích.

Obdobné tabulky znaků, jako pro zobrazování na monitoru, potřebuje ChiWriter i pro tiskárnu. Ty se liší interně pouze rozlišením a maximálními rozměry rastru. Obsahy tabulek nejsou součástí kódu ChiWriteru, ale tvoří samostatné soubory, které jsou rozlišeny jednak jménem, jednak extenzí (tj. částí kompletního jména souboru v MS-DOSu). Pro správnou práci ChiWriteru potřebujete vždy dvojice tabulek se stejným jménem (jednu pro monitor a jednu pro tiskárnu), ve kterých jsou pod stejnými kódy stejné znaky. (Pokud máte pořadí znaků v jedné tabulce oproti druhé přeházeno, máte smůlu a text, který vypadá na monitoru normálně, se vám vytiskne v poněkud zašifrované podobě.) Tyto tabulky - soubory jsou v rámci dvojice rozlišeny extenzí.

Tak se konečně dostáváme k tomu, co se v ChiWriteru chápe pod pojmem font. Fontem tedy nazvěme sadu znaků popsaných v příslušných tabulkách. Popisu fontu jsem věnoval tolik místa záměrně, neboť koncepce fontu je jednak účinné řešení problému nestandardních znaků, jednak je pro pochopení funkce ChiWriteru nezbytný.

V základní sadě je dodáváno několik alfanumerických fontů obsahujících různé typy písma, jako např. tučné, kurzívu, skript, gotické a podtržené písmo, písmena národních abeced aj., ale také azbuku a řeckou abecedu, a několik symbolických fontů umožňujících např. orámování tabulek, vytváření matematických symbolů (integrační a sumační znaménka a závorky různých velikostí, relační znaménka, aj.), či obsahující speciální znaky, jako např. znak copyrightu, měnové symboly apod.

Je logické, že alfanumerické fonty mají kódy přiřazeny k jednotlivým znakům tak, aby byly maximálné kompatibilní s kódem ASCII. Tato kompatibilita může být absolutní pouze u fontů, které obsahují stejné znaky jako ASCII pouze v jiné typografické sazbě (tedy tučný, kurzíva apod.). U ostatních je možno tuto kompatibilitu dodržet jen částečně nebo vůbec ne, a uživateli nezbývá než si pamatovat, že např. horní část velkého integračního znaménka má ve fontu s názvem MathII, umístěnou pod písmenem "i" (tedy s kódem stejným jako má ASCII "i").

Zároveň, díky této koncepci fontů, je velmi snadné implementovat na ChiWriter češtinu (tzn. písmenka s diakritickými znaménky). Stačí např. k standardnímu fontu doplnit další, upravený přidáním čárek a háčků. Obvykle se ale čeština implementuje v ChiWriteru do dvou upravených fontů, protože ke kompletnímu pokrytí velkých a malých písmen české abecedy potřebujete 82 znaků a tak vám zbývá v jednom fontu pouze 12 znaků pro číslice, tečku, čárku atd., což je málo. Běžně se tedy volí uspořádání takové, že jeden font vypouští oproti standardnímu ASCII číslice a některé další znaky a nahrazuje je malými českými háčkovanými a čárkovanými písmeny (stejně jako na psacím stroji), a druhý font obsahuje česká písmena, která i na psacím stroji píšete odděleným úhozem "mrtvých kláves" pro háček nebo čárku.

ChiWriter umožňuje vytvořit si jako znak pouze část požadovaného symbolu a ten potom složit z více znaků "napsaných" na vhodných místech, tedy z příslušně nadefinovaných znaků dělat "skládačky" symbolů, jako se toho využívá např. u dodávaných matematických fontů pro psaní různě velkých závorek či integrálů. Je tedy možno vytvořit si jako čtyři znaky značku benzenového jádra nebo další potřebné značky a pomocí tohoto fontu "kreslit" chemické vzorce složitých organických sloučenin. ChiWriter samozřejmě není program na kreslení schémat, ale někdy se tato možnost vytvořit si prakticky libovolný obrazec může hodit. Základním omezením při tvorbě znaků je fakt, že znaky fontu jsou zadávány v rastru s pevnými maximálními rozměry (např. pro devítijehličkové tiskárny v rastru 16x24 bodů). Kdybychom proto chtěli vytvořit např. písmeno A 3x vyšší a 2x širší než standardní, spotřebovali bychom na vytvoření tohoto A šest znaků, což mimo jiné znamená, že celou abecedu složenou z těchto maxipísmen do jednoho fontu neumístíme.

K fontům ještě dvě poznámky. Celkový počet 20 fontů, které můžete současně použít v jednom dokumentu, považuji osobně za plně postačující a připadá-li to někomu jako omezení, odkazuji ho na DTP systémy.

Druhá poznámka se týká možné námitky, že taková benevolence v přiřazování kódů jednotlivým znakům povede k přišernému zmatku. To také bohužel je v praxi pozorovaný jev. Proto lze uživateli pouze doporučit, aby si v případě, že mu nevyhovují již používané fonty, vytvořil vlastní sadu fontů a pokud bude někomu předávat dokument napsaný pomocí vlastních fontů, poskytnout tyto fonty společně s textem.

3. Psaní textu

Každý textový procesor, tedy i ChiWriter, umožňuje s textem provádět kromě běžného psaní i další operace. Tyto funkce lze u ChiWriteru vyvolávat dvěma způsoby. Předně je to pomocí menu, které je obdobné jako menu jiného rozšířeného programu Lotus 1-2-3. Menu je vhodné především pro uživatele, kteří v ChiWriteru nepíší příliš často, nebo se s ním teprve učí zacházet. Nevýhodou je, že některé funkce se vyvolávají až ve druhém či třetím submenu a tím se uživatel zbytečně zdržuje. Většinu příkazů, tj. všechny častěji používané, lze u ChiWriteru vyvolat také pomocí kombinace určité klávesy (většinou první písmeno názvu funkce) s klávesami Ctrl nebo Alt. Takto lze pracovat s ChiWriterem velmi efektivně a rychle.

Pro ty, kteří si nejsou nějakou funkcí jisti, ev. pro začátečníky, kteří se s ChiWriterem teprve seznamují, je tu možnost vyvolat soubor Help, který je zpracován dostatečně dobře na to, abyste mohli s ChiWriterem uspokojivě pracovat, aniž byste si předtím přečetli manuál.

Psaní textu je ovlivněno třemi přepínači, kterými lze nastavit různé režimy (módy). Jednak je to obvyklý insert mód, který zapříčiňuje to, že se znaky "vsouvají" na pozici kurzoru do textu a text napravo od kurzoru se automaticky posouvá doprava. Při vypnutém insert módu se text přepisuje.

Další mód nám zabezpečuje automatické zarovnávání pravého okraje textu (justification). Aby mohl ChiWriter automaticky formátovat text, vkládá do něj tzv. měkké mezery a měkká odřádkování (soft spaces, soft returns), které se při změně textu nebo formátu zase samovolně vypouštějí. Psaní textu vypadá pak asi následovně; slovo, které se již nevejde celé na jeden řádek, ChiWriter automaticky přetáhne na další (word-wrapping). Přitom provede měkké odřádkování. Při zapnutém zarovnávání pravého okraje jsou současně mezi slova zarovnávané řádky doplněny měkké mezery tak, že poslední slovo na řádce je zarovnáno k pravé zarážce. Jak již bylo naznačeno, měkké mezery a odřádkování může ChiWriter při přeformátování odstavce podle potřeby dodávat nebo ubírat. Tvrdé odřádkování (hard return) ukončuje obvykle odstavec a pro ChiWriter signalizuje ukončení formátovací operace. Proto se nevyplácí odřádkovávat během odstavce "ručně", neboť potom v případě potřeby, např. po provedení oprav, odstavec řádně nezformátujete. Stejně tak není vhodné zarovnávat pravý okraj tvrdými mezerami, tj. těmi, které napíšete stisknutím mezerníku.

Pohyb kurzoru je možno řídit obvyklým způsobem, tedy po sloupcích a řádcích, po slovech, po stránkách a na začátek a konec textu.

Obdobně jako na psacím stroji lze u ChiWriteru libovolně nastavovat levou a pravou zarážku textu. Nastavení zarážek nemusí být stálé v celém dokumentu, čímž si můžete např. vynechat místo na obrázek v textu tak, že v příslušné části textu vhodně zmenšíte šířku dokumentu. ChiWriter neumožňuje psát vícesloupcový text. Pokud se to zdá být omezením, připomínám, že ChiWriter je textový procesor umožňující snadné psaní matematických textů, nikoliv DTP systém.

Vhodnou kombinací nastavení zarážek a tabulátorů, které můžete nastavit na libovolný sloupec textu, můžete bez problémů dělat prakticky vše, co si zamanete, včetně vícenásobného odsazování textu u výčtu položek.

Kromě zarovnávání pravého okraje umožňuje ChiWriter text centrovat a zarovnávat vámi určený znak slova na určitý sloupec. To je velmi užitečné, když např. zarovnáváte zprava čísla v tabulce, nebo u nich zarovnáváte desetinnou čárku.

Pro psaní tabulek je také užitečný tzv. box mód, který umožňuje pomocí pohybu kurzoru kreslit vodorovné a svislé čáry různých typů, což vám umožní danou tabulku velmi snadno orámovat. ChiWriter přitom sám dosazuje na příslušné pozice v textu znaky z odpovídajícího symbolického fontu. Rámovat nemusíte samozřejmě pouze tabulky, ale např. i vzorečky, důležitá upozornění apod.

Další jedinečná věc, kterou ChiWriter umožňuje, je možnost používat prakticky libovolné množství indexových úrovní. Tento rys je svázán se stavbou řádky v ChiWriteru. Nejmenší přístupnou jednotkou ve vertikálním směru není řádka, ale tzv. řada. Každá řádka se skládá ze dvou či více řad, přičemž při standardním řádkování (jako na psacím stroji) odpovídá 2, 3, 4, a 6 řad na řádek řádkování 1, 1 1/2, 2 a 3. Každou řádku lze ovšem "natáhnout" na libovolný počet řad a vytvořit tak požadované indexové úrovně. To umožňuje psát v ChiWriteru i složité matematické výrazy, protože pokud výraz přesahuje svou výškou normální znaky, je rozdělen do indexových úrovní, v nichž jsou dané symboly složeny z příslušných částí, umístěných nad sebou v sousedících řadách.

Pro editaci takovýchto výrazů má ChiWriter tzv. nesynchronizovaný mód, kdy se prováděné operace (např. vypuštění znaku) týkají pouze řady obsahující kurzor. V synchronizovaném módu se týkají operace celé řádky, tedy všech indexových úrovní. Proto při vypuštění znaku jsou v tomto případě vypuštěny znaky v příslušném sloupci i ve všech zbývajících řadách řádky.

Členění textu do stránek je prováděno dvěma způsoby. Jednak je možno "ručně" umístit do textu na požadovaných místech příkazy k odstránkování, jednak ChiWriter provádí automatické stránkování podle nastaveného počtu řad na stránku. Sympatické je, že lze zakázat odstránkování za určitým řádkem, takže je možno se vyvarovat takových nepříjemností, jako je např. odstránkování mezi názvem a vlastním začátkem kapitoly.

Přepínání jednotlivých fontů je vyřešeno velmi jednoduše - stačí stisknout příslušnou funkční klávesu, přičemž pokud ji stisknete pouze jednou, je ve zvoleném fontu napsán jen následující znak a pokračujete v psaní v původním fontu. Pro trvalé přepnutí musíte stisknout příslušnou funkční klávesu dvakrát. To je užitečné, pokud např. prokládáte běžný text řeckými písmeny. Pokud byste náhodou zapomněli, v kterém fontu nebo na které klávese máte nadefinovaný příslušný znak, je možno i zde vyvolat Help a na obrazovce se vám zobrazí klávesnice s příslušnými nadefinovanými znaky.

Blokové operace jsou v ChiWriteru prováděny přes tzv. paste buffer. Označenou oblast je možno buď přenést nebo okopírovat do tohoto bufferu a potom ji podle potřeby vložit na jinou pozici. Obsah bufferu se maže až při dalším zápisu do něj. Tím je možno realizovat všechny požadované operace, jako je mazání, kopírování a přemisťování bloků včetně možnosti opravy nechtěného vymazání bloku (Undo). V označených oblastech lze navíc provádět změny fontů, měnit řádkování, tisknout je na tiskárně či uložit na disk.

Pokud potřebujete najít v textu výskyt určité posloupnosti znaků nebo nahradit jednu poslupnost druhou, ChiWriter tuto možnost poskytuje. Samozřejmě je možno zadávat jako hledanou či nahrazující posloupnost všechny znaky včetně změn fontů, takže je možno např. nahradit všechna kurzívou psaná "ahoj" jejich tučnou verzí. Musíte si dát ovšem pozor na případ, kdy máte např. písmeno "ó" nadefinováno ve dvou fontech pod různými kódy. Potom, píšete-li např. slovo "kód" s písmenem "ó" střídavě z jednoho a druhého fontu, vyhledávací rutina najde pouze ty výskyty slova, které jsou totožné se zadaným. To je také jeden z důsledků takto pojaté koncepce fontů. Při zapnutém synchronizovaném módu jsou prohledávány pouze základní řady, v nesynchronizovaném i všechny indexové úrovně.

Psaní poznámek pod čarou je v ChiWriteru velmi jednoduché. V místě, ke kterému chcete poznámku vztáhnout, pouze umístíte odkaz, čímž se vám obrazovka rozdělí na dvě poloviny a v té dolní můžete psát text poznámky. Po ukončení editace textu poznámky se vrátíte na původní místo v dokumentu. Poznámky jsou číslovány automaticky.

Často opakovaná slova či činnosti je možno v ChiWriteru nadefinovat jako posloupnosti kláves. S ChiWriterem dostanete již nadefinované posloupnosti vypisující např. různě velké závorky a odmocniny. Je také např. možno definovat si posloupnost zarovnávající desetinnou čárku čísla nebo vypisující vaši adresu. Posloupnosti kláves mohou volat jedna druhou.

Integrovaný spelling checker je již upraven "na míru", tedy např. ignoruje změny fontů, které by jiným dělaly potíže. Umožňuje také používat pomocné slovníky. Slovník k příslušnému dokumentu vytvoříte tak, že neznámá slova do něj zapíšete příkazem Write spelling checkeru. Těchto pomocných slovníků můžete v daném okamžiku používat více a díky tomu, že se jedná o obyčejné ASCII soubory, je možno (např. i pomocí ChiWriteru) je spojovat, a vytvořit si tak z několika malých slovníků, patřících k příslušným dokumentům, jeden univerzální velký slovník k dané oblasti. Slovník spelling checkeru obsahuje americkou angličtinu, proto se nesmíte divit, že některé správné anglické tvary slov nebude znát. Použití českého spelling checkeru (snad v dohledné době bude realizován) je komplikováno již zmíněným důsledkem koncepce fontu, tedy prakticky libovolným přiřazováním kódů jednotlivým znakům.

Jedním ze základních údajů o textovém procesoru je maximální velikost editovaného dokumentu. Tu u ChiWriteru není možno jednoduše určit, neboť např. časté změny fontů zvyšují faktickou velikost daného textu. Velice hrubým odhadem se dá říci, že při 640 kB paměti a standardních stránkách lze v ChiWriteru editovat zhruba 70-ti stránkový text. Pokud potřebujete více, nezbývá vám než rozdělit text do více částí.

Z diskových operací ovládá ChiWriter kromě zápisu a přečtení textu i změnu a zobrazení adresáře, vymazání určitého souboru a připojení dokumentu (Merge). Dále je možno zaznamenávat a číst dokumenty jako obyčejné ASCII soubory, ovšem za cenu ztráty všech vymožeností jako jsou změny fontů, indexy aj.; tato funkce není prakticky použitelná ani pro matematické, ani pro české texty. Toho lze ale využít např. pro přenos běžného anglického textu mezi různými programy nebo k jednoduché editaci konfiguračních souborů, i když si ChiWriter nedělá nárok na náhradu dobrého programátorského editoru. Je možno si také nechat pravidelně (s volitelným intervalem 1 až 9999 minut) zaznamenávat text do pomocného souboru. Ačkoliv se lze takto velmi snadno vyhnout nepříjemným překvapením, ať už v důsledku vlastní chyby, kdy si např. nechtěně smažete část dokumentu, nebo výpadku napájení počítače, zaráží mne, jak málo uživatelů v mém okolí si tuto možnost zvyklo využívat.

4. Tisk

Efektivnost práce s tiskárnou závisí na tzv. driveru tiskárny. To je ovšem ve skutečnosti pouhý textový soubor, kterým je ChiWriter informován o tom, co tiskárna umí a jak se toho dosáhne. Jelikož je tisk v grafickém režimu značně pomalý (na devítijehličkových tiskárnách se většinou jedna řádka tiskne na čtyři průchody hlavy, a tak tisk jedné normalizované stránky českého textu obnáší podle rychlosti tiskárny cca 5 minut), je výhodné maximálně využívat textového režimu tiskárny. Ty znaky, které neumí tiskárna tisknout v textovém režimu, jsou tisknuty v režimu grafickém a jsou brány z příslušných souborů. Chceme-li ovšem, aby český text vypadal jednotně, není možno kombinovat grafický a textový režim tiskárny a český text je nutno tisknout celý v grafice, přestože písmen s diakritickými znaménky v něm je menšina. Pokud se vám ovšem podaří vytvořit nějakým způsobem (např. pomocí down-loadu) takové fonty, že tvar písmen s diakritikou je tentýž jako tvar původních znaků tiskárny, ztrácí výše uvedená námitka smysl.

Tisk jako takový lze volit ve třech různých roztečích, Pica (10 znaků/palec), Elite (12 znaků/ palec) a proporcionální, a ve třech různých kvalitách tisku, Draft, Letter a Special. Special zde označuje něco mezi Draft a Letter a většinou se realizuje nastavením tiskárny do módu emphasized. Fonty, které se tisknou graficky (tzn. např. písmena řecké abecedy), vycházejí ovšem samozřejmě vždy v kvalitě Letter. Není ale problémem napsat driver a navrhnout příslušné fonty, které tisknou češtinu a ostatní nestandardní znaky na 2 průchody vozíku, ovšem za cenu snížení kvality tisku zhruba na úroveň Draft módu, což nemusí být vždy na závadu. Ostatně tento driver je již používán. Za zmínku stojí snad i ten fakt, že je k dispozici driver umožňující dosti efektivní spolupráci ChiWriteru s tiskárnami Robotron.

Při vlastním tisku může ChiWriter využívat tzv. mikromezer. To znamená, že měkké mezery, které byly ChiWriterem dodány do textu při zarovnávání okraje, jsou rozpočítány mezi tvrdé mezery, čímž je dosaženo toho, že na jedné řádce jsou všechny mezery stejné, což opticky značně zkvalitňuje výstup. Ten by měl být nejlepší v proporcionálním módu, který ale může někdy dělat právě díky proměnné šířce znaku potíže při přesném výpočtu mikromezer a pozice znaku potřebné např. pro zarovnávání na určitý sloupec. Pokud se člověk navíc zamyslí nad tím, že proporcionální mód zachovává obsah řádek (nedokáže při dostatku místa na nějaké řádce vytisknout slovo ze řádky následující, tedy vlastně přeformátovat odstavec), zjistí, že je proporcionální jen částečně.

Při tisku je možno nastavovat kromě konstantního odsazení textu (takto lze např. získat požadovaný levý okraj) i horní a dolní okraj. Mezi horním okrajem a textem lze tisknout hlavičku textu, mezi dolním okrajem a textem "patičku" (footer, český ekvivalent mi není znám). Součástí hlavičky nebo patičky může být mj. i číslo stránky. Hlavičky a patičky je možno definovat pro prvních devět stran a pro sudé a liché stránky. Číslování stránek, stejně jako poznámek pod čarou, je automatické. ChiWriter je dostatečně chytrý, aby si dokázal spočítat, co se mu vlastně vejde na stránku, když tam kromě horního a dolního okraje má být i hlavička a patička a několik poznámek pod čarou, takže si nemusíte dělat starosti se zalamováním stránek. Lze samozřejmě tisknout vybrané stránky textu a tisk směřovat do souboru a vytisknout ho dodatečně pouhým zkopírováním na tiskárnu. (Má-li ovšem být text tištěn s podstatným podílem grafického módu tiskárny, je velikost tohoto souboru značná, řádově stovky KB.) Tak je možno tisknout jednak na tiskárnách, ke kterým nemáte běžně přístup, jednak můžete využít programů, které umí tisknout v pozadí, tj. při běhu jiného programu. To je v první řadě služební program Print MS-DOSu (který však v této souvislosti pracuje u verze 3.30 s chybami).

Pokud máte texty vytvořené na počítači s velkou pamětí a jste nuceni tisknout je na počítači s pamětí menší, je vám umožněno tisknout tyto texty přes ChiWriter přímo ze souboru. Jestliže však i v tomto případě potřebujete text editovat, nezbývá vám než rozdělit tyto dokumenty do více částí. Tisk takového dokumentu nečiní žádné potíže, protože čítače stránek a poznámek pod čarou je možno nastavit na libovolnou počáteční hodnotu, čímž můžete zařídit, že čísla stránek i poznámek pod čarou budou tištěna správně.

5. Modifikace

Základním způsobem modifikace ChiWriteru je nastavení konfigurace. Konfigurační soubor obsahuje informace např. o tom, kolik a které fonty se budou používat, nastavení tabulátorů a zarážek, počáteční hodnoty rozteče a kvality tisku, název driveru pro tiskárnu, aj. Zvláštní konfigurační soubor je určen pro zobrazení na monitoru. Ten obsahuje název driveru a údaje o rozlišovací schopnosti grafické desky. U desek s vyšším rozlišením (např. EGA, VGA, Hercules) můžete využívat dvou módů. První využívá speciální fonty se znaky, které jsou velké a dobře čitelné. Nevýhodou je to, že na monitoru vidíte pouze cca 2/3 toho, co při použití druhého módu. Ten využívá standardních fontů (určených pro kartu CGA), takže znaky jsou menší a vejde se vám tedy na obrazovku více řad, ale čitelnost se sníží na hranici ještě pohodlného čtení. Mezi oběma módy se můžete přepínat změnou konfiguračního souboru pro monitor. Barevné desky EGA a VGA je možno inicializovat pomocí dodávaných programů a nastavit si na nich požadované kombinace barev. Tady máte možnost experimentování, nemyslím však, že příliš užitečného.

Další možnosti modifikace jsou v driveru tiskárny. Ten si můžete vytvořit tak, aby optimálně využíval možnosti vaší tiskárny, mj. i downloadu. V manuálu ChiWriteru naleznete dostatek informací o tom, jak má takový driver vypadat. Pokud jste dostatečně zkušený programátor a nelíbí se vám spolupráce ChiWriteru s tiskárnou (např. díky tomu, že vaše tiskárna má poněkud obskurně optimalizovaný pohyb hlavičky nebo nestandardní rozlišení), můžete si vytvořit pomocí assembleru vlastní driver (nyní míním opravdu program, obsluhující zařízení), který bude komunikovat s tiskárnou podle vašich požadavků. Podrobnější informace naleznete opět v manuálu.

Na závěr ještě poznámku o přenositelnosti souborů. Soubory ChiWriteru jsou svým vnitřním formátem neslučitelné s formátem všech známých programů. Přenos textu pomocí ASCII souborů (s omezeními uvedenými již dříve) je omezen v podstatě pouze na běžné anglické texty. Češtinu, matematické výrazy a různé jiné speciální symboly není možno vlastně efektivně přenášet vůbec. Zkušenější programátor by ale měl umět napsat konverzní program mezi ChiWriterem a jinými programy (vnitřní formát ChiWriteru je podrobně popsán v manuálu), ale díky libovůli v přiřazování kódů znakům fontu vám takový program bude fungovat právě s jednou sadou fon-

tů (1), a složité matematické výrazy obecně nelze transformovat vůbec. To vidím jako jedno ze základních omezení ChiWriteru, protože text, který lze v něm většinou snadno psát, nemůžete použít jako zdroj pro nějaký výkonnější program, např. DTP systém.

6. Závěr

Co říci závěrem? Pokud se budete dívat na ChiWriter jako na jednoduchý textový procesor, umožňující kromě psaní technických textů i snadné použití češtiny, budete jeho schopnostmi příjemně překvapeni. Pokud budete požadovat výstup na profesionální úrovni a s možnostmi profesionální sazby, asi budete naopak zklamáni. Osobně se domnívám, že ChiWriter je velmi zdařilý produkt (v našich krajích hovořit o velmi výhodné ceně asi nemá příliš smysl), který si u nás udělal jméno snadnou implementací češtiny na prakticky libovolném počítači kompatibilním s IBM PC, přestože jeho původní určení je jiné. Přes nejednotnosti v definici českých fontů začíná svým rozšířením již působit jako určitý standard. Stojíte-li proto před problémem psaní českých nebo technických textů a nemáte příliš náročné požadavky, mohu vám ChiWriter vřele doporučit.

---

(1) Programy pro převod českého textu (čeština Kamenických) z a do formátu editoru ChiWriter ovšem existují. PC-DIR šíří jejich zdrojový text, takže je lze jednoduše modifikovat pro různá rozložení českých kláves (pozn. red.).

Obr. 1: Příklady dodávaných fontů.

Italic
Bold
Script
Gothic
small

ORATOR

Greek (α β γ Γ Δ)
Symbol (¶ ® ©)
Foreign (ä ß Å æ)
Linedraw (┬ → │)
Math I (≈ ± ∞ ≤ ≠ ∈)
Math II (∫ { Σ)

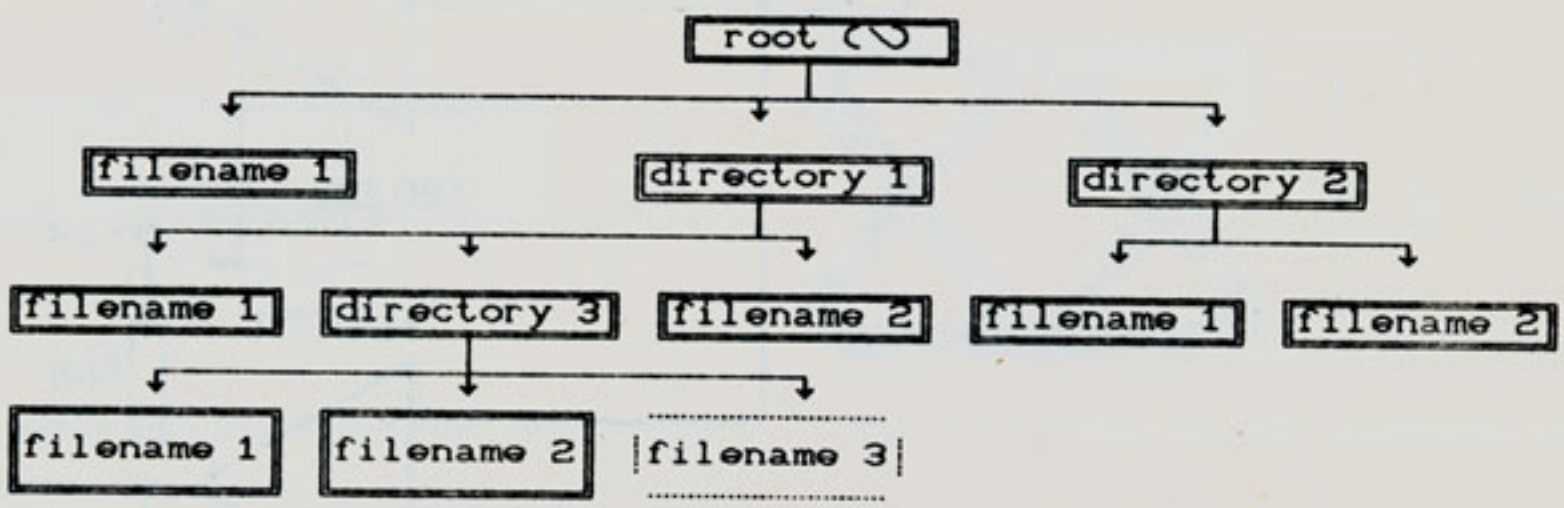
Obr. 2: Ukázka symbolů skládaných z více znaků.

$$\Sigma\ \Sigma\ \sum\ \sum\ \sum\ \sum\ [\,]\ \left[\,\right]\ (\,)\ \left(\,\right)\ \{\,\}\ \int\ \int\ \oint\ \cup\ \cap\ \sqcap\ \sqcup\ \times\ \sqrt{\sqrt{}}$$

Obr. 3: Vzorečky lze v ChiWriteru psát velmi snadno...

$$[D^{n}_{r,kl}]^{-1} \equiv \int_{0}^{R} 2\pi r dr \int_{z_n - \frac{1}{2}h_n}^{z_n + \frac{1}{2}h_n} [\rho_k^{n*}(r)][D^{-1}(r,z)][\rho_l^{n}(r)],$$

Obr. 4: Ukázka použití box módu.



Obr. 5: Tabulka vytvořená pomocí automatického zarovnávání.

Zleva	Deset. tečka	Zarovnej ±	Zprava
jedna	1.0	1.0 ± 0.1	jedna
dvě	22.00	22.00 ± 0.22	dvě
tři	333.000	333.000 ± 0.333	tři

Příjem teletextu pomocí osobního počítače

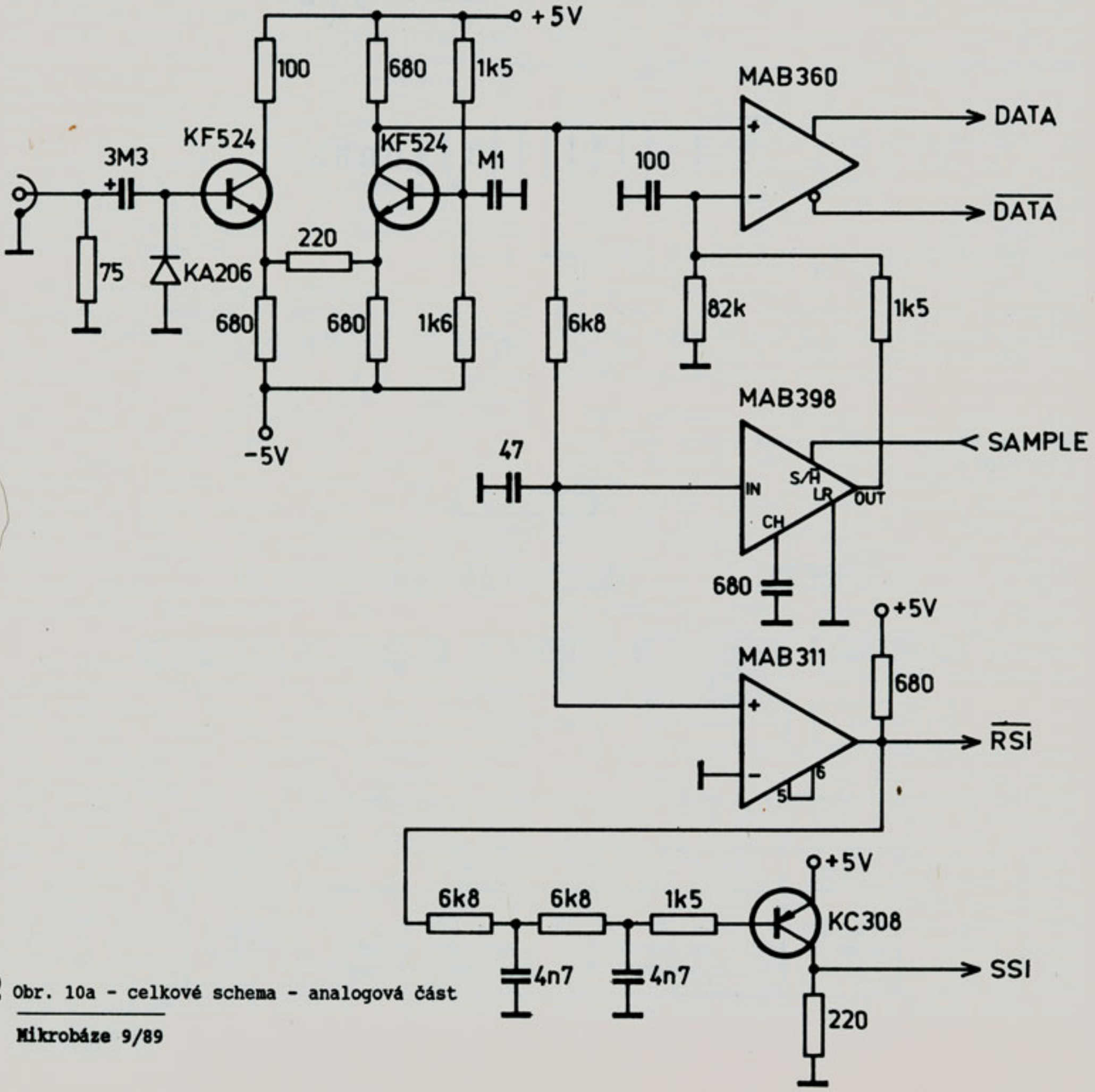
/3/

Tomáš Laška, autor hardware

Oživení a seřízení adaptéru

Na obr. 10 je celkové schema. Z něho je dobře patrná struktura uvedená na blokovém schematu. Oživování je velmi jednoduché protože zapojení obsahuje jen jeden nastavovací prvek a to proměnnou indukčnost v obvodu oscilátoru. V praxi se ukázalo, že k nastavování dokonce není nutné použít čítač ani osciloskop. Nastavení lze provést tak, že navolíme stránku 891 "Zkouška synchronizace dekodéru" a otáčením jádra cívky se snažíme nastavit co nejmenší počet chyb. Tato stránka obsahuje symboly "š" a plný čtvereček, což je v hexadecimálním vyjádření #FE a #7F, takže je v signálu čtrnáct jedniček a dvě nuly. To je nejhorší možná povolená kombinace, která dokonale prověří nastavení oscilátoru, protože synchronizační impulsy jsou generovány při změně signálu DATA. Samozřejmě musí být perfektní signál, aby bylo zaručeno, že chyby pocházejí od špatného nastavení a ne od chyb v signálu. Po nastavení oscilátoru je vhodné cívku zakápnout lakem. Takto nastavený adaptér můžeme připojit na libovolný zdroj normovaného videosignálu obsahujícího Teletext a nemusíme již nic jiného nastavovat.

Zatím jsme se nesetkali se zahraničním Teletextem, který by obsahoval takové užitečné stránky na testování dekodérů, jaké distribuje ČST a za to jí patří dík.

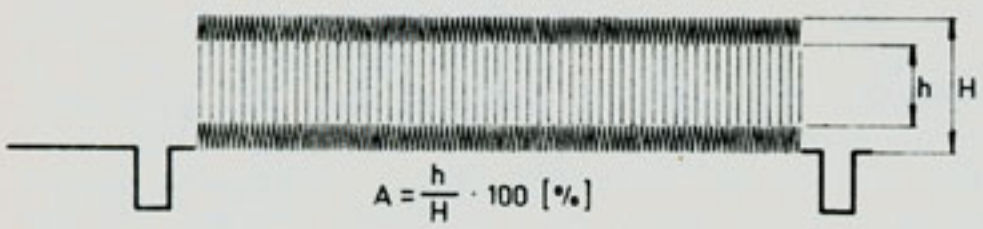


Obr. 10a - celkové schema - analogová část

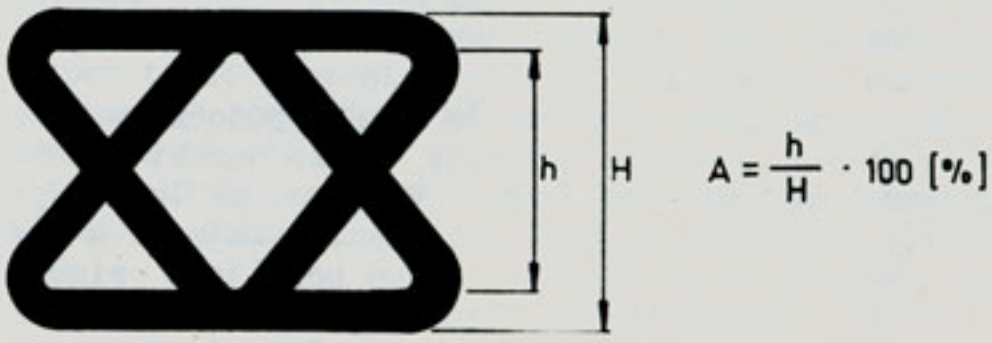
Obr. 10b - celkové schema - digitální část
* poznámka k celkovému schematu: rozvod napájení a blokovací kondenzátory nejsou zakresleny

Kvalitativní parametry signálu Teletext:

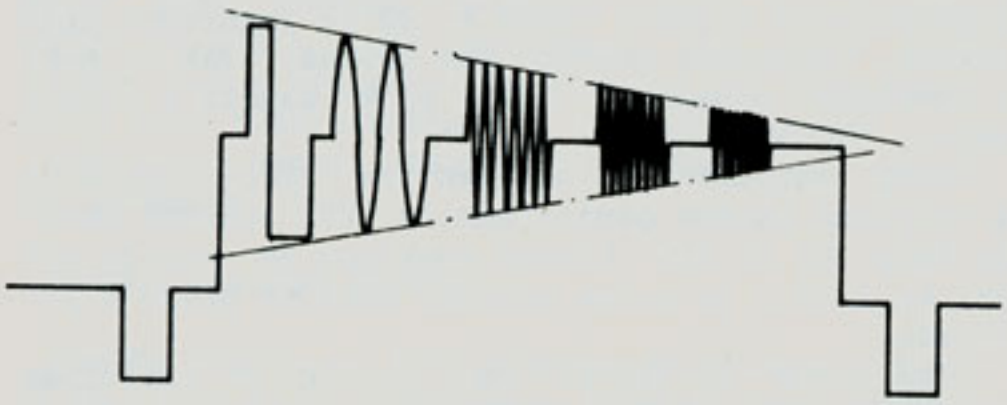
Jedním z nejdůležitějších parametrů je tzv. výška oka, udávaná v procentech. Výškou oka lze zejména zjistit frekvenční a fázovou charakteristiku. Je to vlastně rozdíl mezi minimální hodnotou maximální amplitudy a maximální hodnoty minimální amplitudy k celkové amplitudě oka. Orientačně můžeme výšku oka určit osciloskopem při lineárním vychylování podle obr. 11. Korektně se měření provádí tak, že na vstup X osciloskopu přivedeme sinusový signál o kmitočtu 6.9375 MHz/4 a na vstup Y osciloskopu přivedeme datový signál Teletextu. Výsledný obrazec oka je na obr. 12. Kvalitu signálu a vstupních obvodů TV přijímače můžeme posoudit také podle řádku č. 18 "kmitočtové svazky" (viz obr. 13).



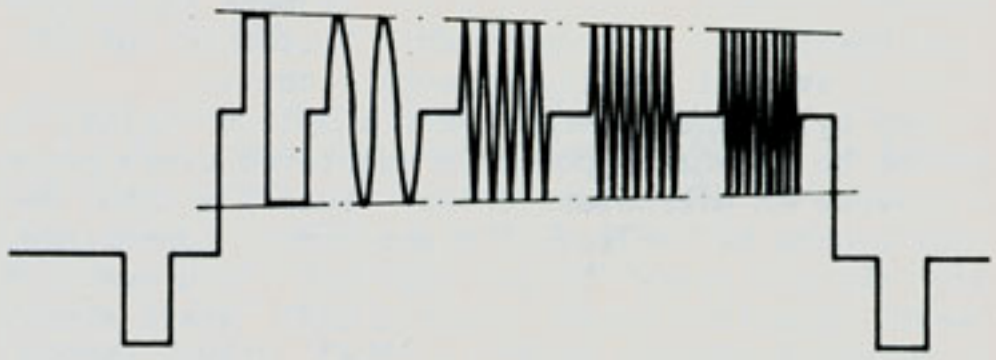
Obr. 11 - výška oka při lineárním vychylování



Obr. 12 - výška oka při sinusovém vychylování



špatná kmitočtová charakteristika



dobrá kmitočtová charakteristika

Obr. 13 - řádek č. 18, kmitočtové svazky

Jak již bylo řečeno, přijímat Teletext není jako chytat Prahu na středních vlnách. Proto bychom chtěli čtenáře seznámit s parametry našeho adaptéru a se zkušenostmi z příjmu. Měření a zkušenosti jsme nasbírali provozem adaptéru a zejména ve Výzkumném ústavu spojů. Tímto bychom chtěli poděkovat za neocenitelnou pomoc ing. Vladimíru Kameníkovi z VUS, člověku, který se problematikou Teletextu zabývá od zrodu tohoto systému.

Teletextový signál je degradován zejména odrazy (vícecestným šířením signálu) a křížovou modulací. Z toho je jasné, že problémy vznikají zejména v husté městské zástavbě, ve společných televizních rozvodech, kde vlivem impedančního nepřizpůsobení vznikají odrazy, signál je často konvertován do nižších pásem atd. Příjem je ztížen v oblastech se silným průmyslovým rušením a u vysílačů pracujících na blízkých kmitočtech. Velmi záleží také na konkrétním vysílači. V Praze je na tom nejhůře stařičký vysílač na Petříně (kolem 60% výšky oka) a nejlépe Buková hora (kolem 80%), měřeno na Jižním městě. Dá se říci, že i ve městě se pohybuje výška oka v průměru nad 65%. Samozřejmě záleží na lokalitě a přijímaném vysílači.

Dosažené výsledky:

Do směrové antény byl zařazen útlumový a šumový článek a bylo prováděno měření oka při prahu chybovosti tj. když se na kontrolní stránce č. 891 začnou vyskytovat chyby. U našeho adaptéru se tato hodnota pohybovala kolem 37% ve srovnání s profesionálním výrobkem fy PHILIPS kde jsme naměřili 35%. Dá se tedy říci že náš adaptér je s tímto dekodérem plně srovnatelný. Při uvedené výšce oka se v textu projevují drobné výpadky které neporuší význam textu.

Při praktickém provozu se takové extrémní podmínky nevyskytují. Při provozu z přenosného BTV Mánes na prutovou anténu z vysílače Petřín (výstupní výška oka 52%) lze příjem označit za bezchybný. Ověřili jsme i příjem ze společného televizního rozvodu na Jižním městě v devítipodlažním panelovém domě, kde je program PLR a druhý program ČSR konvertován do 1. TV pásma a získali jsme překvapující výsledky. 2. program ČST byl bezchybný a na PLR, kde byla patrná křížová modulace (Votice) se vyskytly ojedinělé chyby. Výborné výsledky jsme dosáhli při příjmu z geostacionárních satelitů. Tam je signál perfektní a tudíž i příjem Teletextu je bezchybný. Pokud se na některém slabším transpondéru objeví v obraze drop-outy, projeví se to v Teletextu jako shluky chyb. V tomto případě by ani nejlepší dekodér nevyrobil signál vlastně z "ničeho".

K praktickému provozu je nutno podotknout, že subjektivně nejlépe naladěný obraz nemusí ještě znamenat nejlepší Teletext. Většinou nejlepší výsledky dosáhneme drobným rozladěním, zejména u méně kvalitních přijímačů. Pokud použijeme signál z tuneru videorekordéru, laděného kmitočtovou syntézou, obdržíme zpravidla perfektní Teletext nebot'tyto tunery bývají kvalitní.

Před zavedením vysílání Teletextu bylo nutné provést korekci vysílačů a to zejména kmitočtové charakteristiky a skupinového zpoždění, což přineslo i zvýšení kvality barevného obrazu.

Aby bylo zhodnocení našeho adaptéru úplné, je vhodné ho srovnat s podobným adaptérem uveřejněným v příloze AR mikroelektronika, autorů z Brna. Neradi bychom se snižovali k pomlouvání konkurence, ale některé prvky jejich zapojení jsou svérázné (např. kapacitní zátěž signálu DATA, která slouží k "doladění signálu DATA" a jiné). Hlavní nevýhodou jejich zapojení je nutnost nastavovat rozhodovací úroveň komparátoru v obvodu regenerace Teletextu individuálně podle zdroje signálu. Je to dáno použitím vrcholového detektoru a trimru pro komparační úroveň, čímž vzniká značná závislost na signálu. Poněkud složitá je i komunikace adaptéru s počítačem.

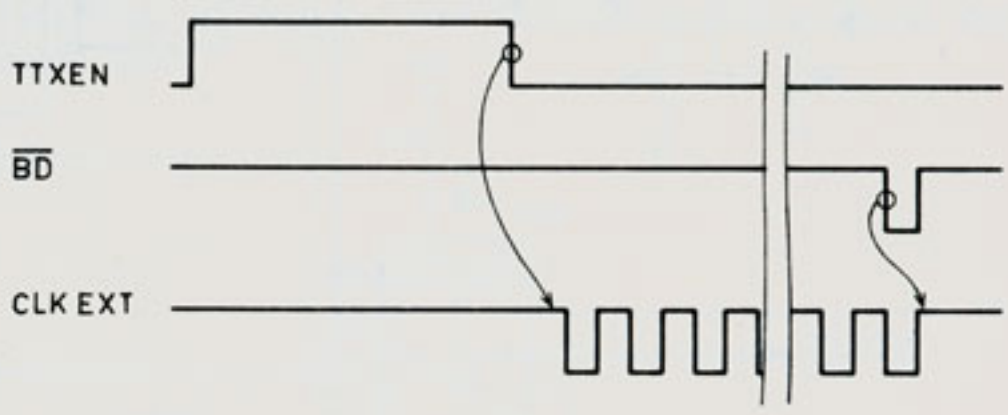
Náš adaptér obsahuje 13 integrovaných obvodů TTL, 3 analogové IO, 3 tranzistory a 1 nastavovací prvek - brněnský 22 IO, 7 tranzistorů a 7 nastavovacích prvků.

Náš adaptér nemá žádné nároky na řadu TTL; v praxi byly zkoušeny různé kombinace obvodů řady standard, LS a ALS, přičemž zapojení vždy fungovalo na první pokus.

Připojení k počítači

Komunikace s počítačem se děje přes paralelní rozhraní např. s obvodem 8255, nebo Z80 PIO. Adaptér vyžaduje 10 vstupních linek (8 dat, 2 handshake) a 1 výstupní (clock external). V této konfiguraci je adaptér nezávislý na hardware počítače. Pokud bychom se vázali více na hardware (využití interruptu), dal by se počet linek redukovat na 8 vstupních - datových. Tím by ovšem adaptér ztratil na univerzálnosti.

Komunikace s počítačem probíhá takto (obr. 9):



Obr. 9 - komunikace adaptéru s počítačem

software testuje signál TTXEN. Pokud je aktivní, znamená to, že adaptér ukládá TTX data do své paměti od adresy nula. Po skončení aktivity TTXEN začne počítač přesouvat TTX data do své operační paměti. To se provádí tak, že se vodičem CLKEXT generují hodinové impulzy. Po vygenerování každého impulzu je jeden datový bajt uložen do RAM počítače. Současně se testuje stav signálu BO, který udává zda-li jsou všechna TTX data přesunuta (jde v podstatě o odčítání do adresy 0). Toto řešení je výhodné tím, že po skončení aktivity signálu TTXEN se čítače zastaví na adrese dané počtem Teletextových dat v půlsnímku. Jejich odčítáním do nuly je přesunuto jen nezbytné množství dat. Při programovém generování hodinových impulzů zase nejsou kladeny nároky na hodinový takt. Instrukční soubor mikroprocesoru pouze musí stihnout přenést TTX data do konce půlsnímku, respektive do další aktivity signálu TTXEN.

Zrychlení přenosu by bylo možno provést připojením chip selectu vstupního portu na vstup CLKEXT a připojením signálů TTXEN a BO na interrupt procesoru, případně použít DMA přenos. Pak by mohl v adaptéru odpadnout celý blok pamětí a čítačů adres, tj. 5 IO.

Celý adaptér je napájen ze zdroje +5V/0.7A a -5V/20mA (možno použít měnič). Při osazení obvody řady LS či ALS klesne spotřeba ze zdroje +5V asi na 0.2A a je možno adaptér napájet přímo z počítače.

Teď zbývá už jen si něco povědět o obslužném programu. To udělá příště jeho autor, Michal Matyska.

(pokračování příště)

DŘU, DŘEŠ, DŘEME... CÉČKO /9/

Struktury

Představte si, že chcete založit velmi jednoduchou databázi obsahující jména lidí, jejich adresy a telefonní čísla. Pochopitelně byste použili pole - pro 100 záznamů třeba takováto:

```
char jména[100][30];
char adresy[100][50];
double telef[100];
```

Pro takový případ Céčko nabízí strukturu:

```
struct seznam
 (char jména[30];
  char adresy[50];
  double telef;
 ) početstr[100];
```

Jde o obdobu záznamů (records) v Pascalu nebo Module-2. Uvedená struktura má jmenovku **seznam** a název své proměnné **početstr** s přímým určením počtu struktur [100]. Je to pole 100 struktur. Interní pole **jména**, pole **adresy** a proměnná **telef** jsou jednotlivými prvky každé struktury. Výsledkem této deklarace, resp. definice je rezervování 100*(30+50+4)=8400 bajtů paměti pro zápis dat. Do takto definované struktury můžete rovnou vkládat svá data. Pro přístup k prvkům struktury má Céčko dva speciální operátory. Pro náš jednoduchý případ je přístupovým operátorem . (**tečka**). První zleva se vždy píše název proměnné struktury s ev. indexem:

```
početstr[55].telef=1234567;
```

znamená, že si zjednáváme přístup k elementu **telef** v 56. struktuře pole struktur (jistě si pamatujete, že číslování polí jde od nuly). Tomuto elementu je současně přiřazena uvedená číselná hodnota.

Proměnná **početstr** tu má podobný význam jako v případě jiných proměnných:

```
int pole.100.;
```

vyhrazuje paměť pro 100 čísel typu int. Když si podle výše uvedené definice místo **int** dosadíte **struct** a místo **pole početstr**, pak nejde o nic jiného, než o vyhrazení paměti pro 100 (kombinovaných) polí typu **struktura** s jmenovkou **soupis**.

Deklarace nemusí obsahovat ani jmenovku, ani název proměnné; např.:

```
struct (seznam prvků);
```

Tomuto zápisu se říká **šablona struktury**. V paměti se vůbec nijak neprojeví. Když přidáte jmenovku

```
struct seznam (seznam prvků);
```

nestane se rovněž nic. Ale na jmenovku struktury se můžete odvolávat v programu. Třeba:

```
struct seznam
 (char pole[20];);

main()
(struct seznam početstr[22];
...)
```

Ve funkci **main()** je tak definováno pole 22 struktur sestavených podle šablony struktury s jmenovkou **seznam**. **početstr** je tu opět názvem proměnné typu **struktura**. Můžeme se o tom snadno přesvědčit:

```
typedef struct seznam
        (int i;
         char pole[10];
         char a;
        ) S;

main()
(printf("%d",sizeof(S));)
```

Výpis:

```
13
```

Pro strukturu S bylo vyhrazeno 13 bajtů. V případě S[100] by to bylo 1300 bajtů (u operátoru **sizeof** by opět byl tvar (S) - nikoli (S[100]), protože definovaným typem je S!). Kdybychom v seznamu prvků struktury místo definice **char pole[10]** použili deklaraci **char (*pole)[10]** (což je ukazatel na pole o 10 znacích), ve struktuře by se pro tento element nevyhradilo 10 bajtů, ale jen 2 (samotný ukazatel má u osmibitových počítačů rozměr 2 bajtů).

Tak jako můžeme obecně deklarovat víc proměnných najednou, třeba:

```
int a,b,c,d,*uknaint;
```

jde to i u struktur:

```
struct seznam (seznam prvků) i,j,k,l,*uknastr;
```

Ovšem pozor na obvyklý rozdíl - zatímco struktury i,j,k,l jsou definovány (všem čtyřem je přímo vyhrazena paměť jako u int **a,b,c,d**) a můžeme se tedy přímo obracet k prvkům struktur přes názvy jejich proměnných (např.: j.prvek), v případě ukazatele to hned nejde. Ukazatel musíme napřed nějak inicializovat. Po své prvotní deklaraci ukazatel **"neukazuje nikam"** - obvykle ukazuje na adresu 0 (pokud ji tam kompilátor nedává, je tam "smetí"). Tak např. budeme-li chtít, aby uvedený ukazatel

směřoval na strukturu **k**, provedeme toto přiřazení:

```
uknastr=&k;
```

Pro přístup k prvkům struktur přes ukazatele má Céčko jiný operátor než je tečka. Je to **šipka** vpravo složená ze dvou znaků **->**. Tedy:

```
uknastr->prvek;
```

Což je podle pravidel hry s ukazateli totéž jako:

```
(*uknastr).prvek;
```

Závorky jsou tu nutné, protože tečka má vyšší prioritu.

Při práci se strukturami se často užívá operátor **typedef**. Radši ještě jednou připomenu, že typedef slouží k uživatelské definici typu (typy jako char, int apod. jsou předdefinovány). Při zvykání si na typedef buďte obezřetní, abyste si jím orientaci v programu spíš nezatemnili. Tak např. po deklaraci:

```
typedef int Číslo;
```

můžeme jakoukoli jinou proměnnou typu int deklarovat takto:

```
Číslo a,b,c,d;
```

Všechny proměnné a,b,c,d budou typu int. Nic zvláštního, že? O něco horší to bude s naší začátečnickou orientací třeba už jen v takovéto směsi deklarací:

```
typedef char *Jméno;
Jméno kluk,holka;
```

- jde de facto o deklaraci:

```
char *kluk,*holka;
```

Při malé zkušenosti a v zápalu programovací hry se nám snadno stane, že na hvězdičku zapomeneme a staneme se hrdinou komedie plné omylů.

Povšimněte si, že definované typy píšu s velkým počátečním písmenem, abych na tento typ byl při prohlížení textu upozorněn (a aby se mi nepletl s preprocesorovými definicemi psanými jen velkými písmeny).

Podívejte se zpět na typedef z prográmku pro výpis rozsahu paměti struktury. Podle předchozího schématu můžeme deklarovat a definovat další struktury se stejným seznamem prvků (podle šablony struktury s jmenovkou **seznam**):

```
s *uknastr,jednastr,desetstr[10];
```

K užití typedef struktur závěrem cvičný příklad:

```
typedef struct str
        (char *pole;
        ) Struktura;

main()
(int x;
static char *p="ahoj";
Struktura s[2];
for (x=0;x<2;x++)
 (s[x].pole=p;
  printf("%u %u %s\n",&s[x].pole,s[x].pole,
                        s[x]pole);
 )
)
```

Výpis:

```
65522 48252 ahoj
65524 48252 ahoj
```

Podle šablony struktury **str** jako základu pro definici typu **Struktura** je určen rozsah i obsah každé ze dvou struktur, čili pole struktur s názvem **s**. Prvkem každé struktury je tu jen jeden ukazatel na typ char. Tak dopředu nevíme, kam a na co ukazuje (zpočátku "neukazuje nikam"). Příkaz:

```
static char *p="ahoj";
```

inicializuje ukazatele **p** na typ char i řetězec, na nějž ukazuje. Kompilátor umístí řetězec do volné paměti - jak z výpisu vidíme, řetězec začíná od adresy 48252, což je zároveň hodnota ukazatele **p**. Na tutéž hodnotu se cyklicky incializuje ukazatel pole v každé z obou struktur příkazem:

```
s[x].pole=p;
```

Levý sloupec výpisu informuje o adresách, na nichž jsou sídla ukazatelů pole - jsou to dva sousedící páry adres. Oba odkazují na tentýž řetězec "ahoj".

Zkusme si ještě jednu malou hříčku s ukazatelem - ve struktuře teď nebude ukazatel na typ char, ale přímá definice rozměru znakového pole:

```
struct str
 (char pole[10];
 ) s;

main()
(int x;
static char *depo="ahoj";
for(x=0;*depo;x++)
  s.pole[x]=*depo++;
printf("%c\n%s",*(depo-4),s.pole);
)
```

Výpis:

```
a
ahoj
```

Struktura **s** je definována externě, proto se na ni ve funkci můžeme odvolávat přímo. Podmínka ukončení cyklu ***depo** je totéž, co ***depo!=0**. Ta bude splněna, až cyklus narazí na nulu ukončující řetězec "ahoj". Do té doby se jeden znak po druhém kopíruje do znakového pole ve struktuře. Protože se každým průchodem cyklu zvyšuje ukazatel o 1, po skončení cyklu ukazuje za řetězec na nulu, která cyklus zastaví. Ve výpisu je pomocí adresové aritmetiky ukazatel vrácen na svou původní hodnotu (opět ukazuje na první znak "a"). A jak vidíme, znakové pole struktury převzalo všechny znaky řetězce "ahoj".

Prográmek má však dva nedostatky, kterých se musíte běžně vyvarovat. Jednak neukládá potřebnou nulu za poslední znak pole ve struktuře a cyklus se nestará o to, jestli kopírovaný řetězec není delší, než kolik se vejde do znakového pole struktury (tedy než 9 znaků, protože 10. pak musí být zakončovací binární nula).

Nyní jedno zamyšlení nad rozdílností obou posledních deklarací znakových polí coby prvků struktur. Deklarace ukazatele zabrala 2 bajty paměti. V posledním případě je však přímo stanovena maximální dovolená délka řetězce a podle ní je mu vyhrazena paměť ve struktuře. Tento rozdíl má zásadní význam při práci s poli struktur. Protože řetězce mívají různou délku, užitím šablony podle posledního programu přijde hodně paměti nazmar. Zato budou všechny manipulace se záznamy databáze snadnější (výmazy, vkládání nových záznamů apod. - a to nejen v paměti počítače, ale i při komunikaci s diskem). Použití ukazatelů přímo předurčuje aplikaci funkcí pro práci s dynamickou pamětí. Tato komplexní problematika jde mimo rámec výkladu

(mohl bych se k ní vrátit po skončení naší céčkové seznamky).

Protože je seriál určen naprostým začátečníkům, uvedu ještě dva podobné krátké prográmky, abyste měli práci se strukturami a ukazateli pod nezbytným drobnohledem:

```
struct str (char *znaky;) s;
main()
(static char *uk="ahoj";
s.znaky=uk;
printf("%u %u %u\n",s.znaky,uk,s);
printf("%s %s",s.znaky,uk);
}
```

Výpis:

```
48259 48259 48259
ahoj ahoj
```

A s použitím ukazatele na strukturu:

```
struct str (char *znaky) t,*s;
main()
(static char *uk="ahoj";
s=&t;
s->znaky=uk;
printf("%u %u %u\n",s->znaky,uk,s);
printf("%s %s",s->znaky,uk);
}
```

Výpis:

```
48259 48259 65534
ahoj ahoj
```

V posledním případě je adresa 65534 adresou, na níž leží proměnná t. Ta je přiřazena ukazateli s, protože obsah této adresy (neboli *s) ukazuje na strukturu. Pro odkazy na prvek struktury přes jejího ukazatele je použita šipka vpravo.

Zatím poznané vztahy rozvedu na trochu větší a praktičtější ploše - data budeme vkládat z klávesnice. Přitom se seznámíte s další knihovní funkcí *gets(). Program si postupně rozebereme ve třech velmi příbuzných alternativách. První je nejjednodušší:

```
  5 #define EOF -1
 10 struct lidi
 20  (char jméno[10];
 30   int roknar;);
 40 #define POCET 3
 50
 60 main()
 70 (char *gets(); /* deklarace fce vracející
 80 int x;            ukazatele na typ char */
 90 struct lidi soupis[POCET];
100 for(x=0;x<POCET;x++)
110  (printf("Jméno:\n");
120   gets(soupis[x].jméno);
130   printf("Rok narození:\n");
140   scanf("%d",&soupis[x].roknar);
150   while(getchar!='\n') ;
160  }
170 for(x=0;x<POCET;x++)
180   printf("%s %d\n",soupis[x].jméno,
                       soupis[x].roknar);
190 }
200
210 char *gets(s)
220 char *s;
230 (static int;
240 static char *cs;
250 cs=s;
260 while((c=getchar())!=EOF && c!='\n')
270   *cs++=c;
280 *cs=0;
290 return ((c==EOF && cs==s)?0:s);
300 )
```

Šablona struktury je zřejmá. Definice struktury její proměnnou proběhne na řádce 70 (vyhradí se paměť pro pole tří struktur). Uložení dat do jejich znakových polí jméno probíhá prostřednictvím funkce char *gets(), která vrací ukazatele na typ char (gets znamená get string - dostaň řetězec). Protože jde o funkci, která vrací ukazatele na typ char, musíme ji ve funkci, odkud ji voláme, deklarovat (ř.70). To, že jsme se s takovou deklarací příliš nesetkávali, bylo proto, že jsme zatím používali převážně funkce typu int (jejich zamlčená deklarace "říkala", že jde právě a jen o funkce tohoto typu). Argumentem funkce *gets() je ukazatel, v našem případě adresa prvku jméno, které je znakovým polem. Argumentem je tedy adresa prvního prvku znakového pole jméno, které je prvkem struktury. Funkce *gets() si tuto adresu převede do svých dvou ukazatelů na typ char (viz deklarace *s a *cs). Znaky do řetězce ukládá pomocí *cs (ř. 260 a 270). Ukazatel s je napřed okopírován do ukazatele cs, jehož obsah se v cyklu for postupně zvyšuje o 1. Proto se hodnota ukazatele na začátek řetězce chrání v s. Na ř. 280 se za zapsaný řetězec vloží zakončovací nula. Řádka 290 vrátí buď nulu nebo hodnotu ukazatele s podle toho, je-li obsah závorky pravdivý či naopak (pravdivý je jen tehdy, když jediným vloženým znakem byl EOF čili end of file, konec souboru, který nejčastěji mívá hodnotu -1).

Funkce scanf() uloží číslo do prvku roknar. Připomínám, že tato funkce ve svém argumentu musí mít formát a adresu uložení vkládaných dat. Záhadná řádka 50 slouží k "zabrzdění" zvláštních účinků scanf(). Kdyby tam nebyla, "prohučeli" bychom řádkami 100-130, do pole jméno by se nic neuložilo a zastavili bychom se zase na scanf().

Druhou alternativou programu by mohlo být užití ukazatele na typ char místo znakového pole jméno[10]. Ukazatel nazvěme:

```
char *jméno;
```

Pak se v celém programu změní jen první argument na řádce 180:

```
printf(".....",*soupis[x].jméno,.....);
```

Jde tedy jen o přidání hvězdičky. Zápis:

```
soupis[x].*jméno    /* CHYBA! */
```

by byl z hlediska záměru chybný. Vše, co spojují referenční operátory sestupné odkazující na prvek struktury, tvoří - vzhledem k onomu prvku - nedělitelný celek (viz rovněž řádka 140). Tím ovšem není řečeno, že bychom nemohli použít operátory i "uvnitř" referenčního výrazu. Máme-li takovouto strukturu:

```
struct (int číslo;
        int *ukaz;
       ) *uk;
```

pak pro následující výrazy platí uvedené vztahy:

```
++uk->prvek     inkrementace prvku
(++uk)->prvek   inkrementace uk před přístupem
                k prvku
(uk++)->prvek   inkrementace uk po přístupu
                k prvku
```

```
*uk->ukaz       obsah místa, kam ukazuje ukaz
*uk->ukaz++     inkrementace ukaz po přístupu
                k místu, kam ukaz ukazuje
(*uk->ukaz)++   inkrementace obsahu místa, kam
                ukazuje ukaz
*uk++->ukaz     inkrementace uk po přístupu
                k místu, kam ukazuje ukaz
```

Operátory . a -> mají (spolu s kulatými a hranatými závorkami) nejvyšší prioritu. Proto tam, kde těsnost vazby na referenční operátory potřebujeme přerušit, použijeme kulaté závorky.

Vraťme se zpět k našemu programu. Místo deklarace char *jméno, která předem nijak nevymezuje délku řetězce, bychom adekvátně definici char jméno[10] mohli použít char *jméno[1][10], což je jeden ukazatel na znakové pole o délce 10 bajtů. Deklarace char *jméno [10] by znamenala něco jiného - bylo by to 10 ukazatelů na typ char.

Poslední alternativa přinese opět nový poznatek. Struktury můžeme vnořovat do sebe. Strukturu lidi si "rozdělíme" na dvě:

```
struct lidiA
 (char jméno[10];);
struct lidiB
 (struct lidiA názevstr;
  int roknar;);
```

Struktura lidiA názevstr obsahuje jeden prvek char jméno[10] a je součástí struktury s jmenovkou lidiB. Na prvky vnořených struktur se dostaneme stupňovitými odkazy podle počtu vnoření. To se projeví i nutnými změnami na uvedených řádkách:

```
ř. 90  struct lidiB soupis[POCET];
ř.120  gets(soupis[x].názevstr.jméno);
a první argument na ř. 180:
       soupis[x].názevstr.jméno,
```

K prvku jméno struktury lidiA se tedy dostáváme postupné přes názvy dvou proměnných - soupis a názevstr. Celá záležitost je přísně logická a není v ní žádná záludnost. Jen je třeba dát pozor na odkazování do vnořených struktur, kde se střídají prosté názvy s ukazateli - viz hned další příklad.

Obsah struktur můžeme incializovat i přímo, podobné jako pole:

```
#define DÉLKA 20
struct jméno
 (char křestní[DELKA];
  char příjmení[DELKA];
 );
struct člověk
 (struct jméno nahoře;
  char hobby[DÉLKA];
  char zaměst[DÉLKA];
  char příjem[DÉLKA];
  int početočí;
 );
main()
(static struct člověk tady[2]=
 (("Jan",
   "Zižka"
  )"palcát",
   "programátor husitů",
   "plné kádě",
   1
 ),
 (("-el",
   "zet-"
  )"nevícodřív",
   "akdemuhlavastojí",
   "1 prázdná, 2.děravá",
   2
 );
struct člověk *dva;
dva=&tady[0]; /* inic.ukazatele na strukturu */
printf("%d %d\n",dva->početočí,(*dva).početočí);
++dva;
printf("%s, %s, %s",dva->zaměst,dva->příjem,
                    dva->jméno.příjmení);
)
```

Výpis:

```
1 1
nevícodřív, 1 prázdná 2.děravá, zet-
```

Povšimněte si, že obsah vnořené struktury je oddělen ještě jednými svorkami. Jednou z nejdůležitějších věcí na celém tomto konglomerátu je příkaz:

```
++dva;   /* totéž, co dva+=1; */
```

Jde o zvýšení ukazatele v rámci adresové aritmetiky (tedy nikoli o 1!!!). Každá struktura zabere 102 bajtů. K momentální hodnotě ukazatele se tedy přičte právě 102 bajtů. Ukazatel byl původně inicializován na hodnotu začátku prvního seznamu prvků. Po zvýšení bude ukazovat na druhý seznam. Proto poslední printf() vypisuje ze seznamu druhé struktury. Pokud byste chtěli zvýšit ukazatele skutečně jen o 1, museli byste použít konverze nebo kopie hodnoty do jiné proměnné a po jejím zvýšení zase zpět. Adresovou aritmetikou tak můžete v paměti skákat z jedné struktury do druhé (pokud jsou seřazené vedle sebe a stejné dlouhé).

S tím souvisejí potíže, jaké mají majitelé ZX Spectra se svým Hisoftem C. V něm nelze zjistit rozměr určité struktury z jejich pole, když mají rozdílnou délku. Slušné kompilátory umožňují použít operátor sizeof pro jakýkoli objekt. Hisoft C to umí jen pro (před)definovaný typ, proto se u něj musejí používat různé komplikované "obchvaty", součty a odečty.

Všimněte si tvaru posledního argumentu posledního příkazu programu. Je příkladem kombinovaného odkazu do vnořené struktury (první odkaz jde přes ukazatel, druhý přímo).

Poslední zajímavé vztahy, které platí pro daný program, jsou:

```
dva==&tady[0]
*dva==tady[0]
tady[0].hobby==(*dva).hobby==dva->hobby
```

Pomocí struktur můžeme sestavovat např. binární stromy. V takovém případě struktura odkazuje sama na sebe. Kernighan a Ritchie ve své učebnici uvádějí základní tvar takové struktury:

```
struct prvekstromu
 (char *slovo;
  int count;
  struct prvekstromu *levý;
  struct prvekstromu *pravý;
 );
```

U struktur není dovoleno, aby obsahovaly samy sebe. V uvedené rekurzívní deklaraci jde o odkaz pomocí ukazatele, což je něco jiného. Pro podobné datové konstrukce je nezbytné znát jejich algoritmy a funkce, což překračuje stanovené meze výkladu. Po skončení tohoto seriálu bych se k aplikaci Céčka pro takové případy mohl vrátit.

Uniony

Mezi strukturami a uniony je velmi těsná souvislost s jedinou odlišností - zatímco struktury

jsou objekty s fyzickou posloupností deklarovaných prvků šablony, uniony v každém momentu fyzicky obsahují jen jeden z deklarovaných prvků šablony. Union vždy zabírá paměť v rozsahu nejdelšího deklarovaného prvku. Např.:

```
union prvky
 (int číslo;
  double numero;
  char znak;
 ) un;
```

Tento union bude stále zabírat 4 bajty paměti, protože typ double je nejdelší. Když do tohoto unionu uložíme znak, přemaže se, co v něm bylo zapsáno předtím, a uloží se do něj jeden bajt. Paměťový rozsah 4 bajtů unionu se ale nezmění. Z toho vyplývá, že uniony slouží jako určitá "šatna" či "odložna" pro proměnné, které by jinak zbytečně "visely" v paměti počítače. To má velký význam při šetření paměti, když průběžně potřebujeme vždy jen jednu z mnoha deklarovaných proměnných bez ohledu na stav ostatních. Představte si, že bez unionu byste všechny uvedené prvky museli mít natrvalo uložené v nějakém poli struktur! Jinak se s uniony zachází stejně jako se strukturami. Uniony mohou být prvky struktur a naopak.

Název union nepovažuji za příliš šťastný. Do češtiny byl přeložen jako sjednocení. Oba názvy nijak neevokují podstatu, protože v obecném smyslu sjednocením do unie nic nemizí. Slovo union by se spíš hodilo pro strukturu a pro union snad něco jako selekton. Ale otcové Céčka to tak pojmenovali, tož budiž.

Když jsem poprvé narazil na uniony, bylo mi velkou záhadou, k čemu - kromě šetření paměti - mohou být. Dlouho jsem bádal v literatuře, ale nic zvlášť úchvatného jsem v ní nenašel. Dále tedy aspoň něco z toho, co stojí za uznamenání.

```
#define KRUŽNICE    1
#define OBDÉLNIK    2
#define TROJÚHELNÍK 3
typedef struct
        (float plocha;
         int druh;
         union
          (float průměr; /* kružnice */
           float a[2];   /* obdélník */
           float b[3];   /* trojúhelník */
          ) rozměry;
        ) obrazec;

main()
(obrazec obr; /* definice struktury obr */
...)
```

Když máme pole takovýchto struktur, můžeme každé určit, k jakému geometrickému obrazci se bude vztahovat přidělením čísla 1, 2 nebo 3 do prvku druh pomocí definovaného výrazu preprocesoru a současným přiřazením základních rozměrů obrazce. Např.:

```
obr.druh=KRUŽNICE;
obr.rozměry.průměr=6.8;
```

Když pak chceme provést výpočet plochy obrazce pro každou strukturu (s uložením výsledku do prvku plocha), můžeme použít např. výběr:

```
switch(obrazec.druh)
 (case KRUŽNICE: ...výpočet...; break;
  case OBDÉLNÍK: ...výpočet...; break;
  case TROJÚHELNÍK: ...výpočet...; break;
  default: ...chyba...;
 )
if(!chyba) obr.plocha=...výsledekvýpočtu...;
```

Takovéto konstrukci struktury se říká variant structure neboli variantní, tedy proměnlivá, alternativní struktura (obdoba variant record v Pascalu). Proměnná druh ve struktuře se jmenuje active component tag (jmenovka aktivní složky), protože s její pomocí programově určujeme, který z prvků unionu bude aktivní.

Příklad vnořené struktury do unionu, který je sám vnořen do struktury:

```
struct
(char jméno[25];
 int věk;
 char pohlaví;
 stav st; /* jmenovka aktivní složky */
 union
  (struct
    (char datumsňatku[8];
     char jménodruha[25];
     int početdětí;
    ) rodinainfo;
   char datumrozvodu[8];
  ) matrikainfo;
) základinfo;
```

Aktivní složka může být určena pomocí "očíslování" prvku st, jak je nabízí nový standard Céčka ANSI:

```
typedef enum(SVOBOD,SŇATEK,ROZVOD)stav;
```

Když jsme při ukládání dat programem vyzváni k volbě jedné ze tří možností a zvolíme třeba SŇATEK, do proměnné st se uloží číslo 1 (SVOBOD je 0, ROZVOD 2). Bez enum() musíme použít definice preprocesoru jako v minulém příkladu.

Pomocí typedef je stanoven uživatelský typ stav, který je využit při deklaraci proměnné st. Podle jejího obsahu pak program automaticky nabízí jednotlivé prvky k zápisu, resp. ke čtení. To znamená, že pro svobodné se nebude nabízet k zápisu ani vypisovat cokoli z unionu.

Všimněte si, že struktury ani union nemají jmenovky, protože jsou definovány najednou, bez potřeby následného odkazování na jejich šablony (tehdy bychom se bez jmenovky samozřejmě neobešli).

Příklady odkazů na prvky:

```
základinfo.jméno
základinfo.matrikainfo.datumrozvodu
základinfo.matrikainfo.rodnainfo.datumsňatku
```

Union matrikainfo má dva prvky - strukturu rodinainfo a znakové pole datumrozvodu.

Oba poslední příklady jsem převzal z knihy Naraina Gehaniho "C: An Advanced Introduction" (AT.T Bell Labs, Murray Hill, New Jersey; Computer Science Press, 1985) v ruském překladu N. Džechani: Programmirovanije na jazyke Si (Moskva, Radio i svjaz, 1988). Tato kniha je určena zkušenějším programátorům.

Na závěr mám pro vás jeden hlavolam, jehož řešením příště začnu. Pochází přímo z knihovny Hisoftu C a týká se vstupní deklarace struktury pro práci s V/V soubory. Zkuste vyluštit, jak to s onou strukturou a následnými deklaracemi vlastně je:

```
struct header
 (struct header *ptr;
  unsigned size;
 );
typedef struct header Header, *Headerptr;
Header base, *allocp;
```

Můžete si samozřejmě pomoci analýzou na počíta-

či. Ale jinak jste už vybaveni vším, co pro rozluštění "od oka" potřebujete.

A propos - k složitosti céčkových programů. Nedávno jsem viděl profesionální (a hodně dlouhý) výpis céčkového programu. Ke svému údivu jsem v něm nenašel žádnou velkou složitost jako třeba funkce vracející ukazatel na pole ukazatelů na funkci vracející ukazatel na typ char a podobné šílenosti. I jednohvězdičkových ukazatelů bylo použito s mírou. A hlavně - vše bylo perfektně přehledné a čitelné. A to prosím šlo o editor/kompilátor Céčka!

Vzal jsem si z toho jedno ponaučení - používat instrumentář Céčka co nejjednodušším a nejpřehlednějším způsobem bez ztráty efektivity. Podpořil mě v tom i jeden ze článků pravidelného přispěvatele měsíčníku BYTE, Jerry Pournella. Autor v něm úsměvně obhajuje opovrhovaný Basic (nikoli ten spectrovský) a říká, že když jsou s nějakým softwarem problémy, tak byl určitě napsán v Céčku.

A teď se něčeho přidržte, abyste odolali náporu následujícího zvěstování - Céčko máte za sebou! Už v něm nic víc není! Ale tak úplně pravda to samozřejmě není. Znát slovíčka jazyka a napsat román je přece jen rozdíl. Navíc u Céčka je ten "problém", že se v něm ponejvíce programuje knihovnami, tedy dovedným voláním knihovních podprogramů, jichž je stále rostoucí přehršel.

Proto se sejdeme ještě nad jedním, závěrečným pokračováním, ve kterém se dozvíte aspoň to nejdůležitější o nejdůležitějších funkcích céčkové knihovny zvané **stdio.h** - plus něco navrch.

-elzet-

MODUL KEMPSTON

Ing. Jiří Vacek

1. Popis zapojení

Obvodové schema modulu je na obr. 1. Modul umožňuje připojit ke Spectru pomocí tohoto interface dva ovladače - joysticky, označované podle typu připojení (zejména adresování) jako Kempston joystick. Zapojení je převzato z firemního výrobku a je upraveno jednak o možnost připojení tlačítka RESET, jednak konstrukčně pro zařazení do sady našich ZX modulů.

Zapojení obsahuje pouze 3 integrované obvody, 17 odporů a konstrukční součástky (konektory, resp. tlačítko RESET).

Jádrem jsou obvody 74LS244, sloužící jako oddělovač sběrnice ZX Spectra od signálů ze dvou ovladačů. Výběr obvodů 74LS244 je prováděn pomocí obvodu 74LS32. První ovladač je vybírán pomocí signálů A5 a IORQnon. Jsou-li oba signály v úrovni L, je tato úroveň i na výstupu 8/74LS32, a tím je převzat stav prvního ovladače ze VSTUPu A obvodem IO3 (74LS244) na datovou sběrnici Spectra. Data přečtená programem instrukcí IN 31 jsou předána programu, který je tak informován o poloze ovladače.

Vstupy obvodů 74LS244 jsou přes odpory R1 až R16 (10 k) připojeny k 0V. Ovladač obsahuje celkem 5 spínacích kontaktů. 4 z nich jsou určeny pro základní směry vlevo (left), vpravo (right), nahoru (up) a dolů (down). Pátý je určen pro střelbu (fire) či schvalování. Kontakty ovladače mají společný vodič (common), který je připojen přes R17 (1 k) na napájení +5V.

Jsou-li kontakty ovladače v rozepnutém stavu, jsou vstupy obvodu 74LS244 staženy přes R1 až R16 k 0V a programem je přečtena 0. Nasměrujeme-li ovladač, můžeme pomocí instrukcí Basicu zjistit jeho správnou funkci takto:

```
1 PRINT AT 0,0; IN 31, IN 32,: GOTO 1
```

SMĚRY	VÝPIS
VPRAVO	1
VLEVO	2
DOLŮ (k sobě)	4
NAHORU (od sebe)	8
STŘELBA	16
VPRAVO DOLŮ	5
VLEVO DOLŮ	6
VPRAVO NAHORU	9
VLEVO NAHORU	10

Obdobně funguje i druhý ovladač, který je vybírán signály A4 a IORQnon. Jsou-li oba v úrovni L, je tato úroveň i na výstupu 6/74LS32, a tím je vybrán obvod IO2 typu 74LS244 pro VSTUP B, tedy pro druhý ovladač (adresace A4=0, např. 32).

Tlačítko RESET je připojeno klasickým způsobem, a to přímo na sběrnici ZX (43/K3) a 0V.

2. Stavba a oživení

Návrh oboustranného plošného spoje je na obr. 2 a obr. 3. Rozměr desky je 65 x 92,5 mm. Konstrukčně je modul navržen tak, aby po připojení k počítači ležel plochou na stole. Tlačítko RESET a konektory pro ovladače jsou přístupné shora, modul musíme zařadit jako poslední, neobsahuje totiž pokračovací konektor. Osazení spojové desky modulu je na obr. 4.

Před osazením desky zkontrolujeme nejprve kvalitu spojů a obdobně jako u předchozích modulů pracujeme při osazování s rozmyslem. Většinou máme spoje bez prokovených otvorů pro součástky, proto musíme součástky pájet z obou stran, a to zejména při prostupech signálů na druhou stranu desky.

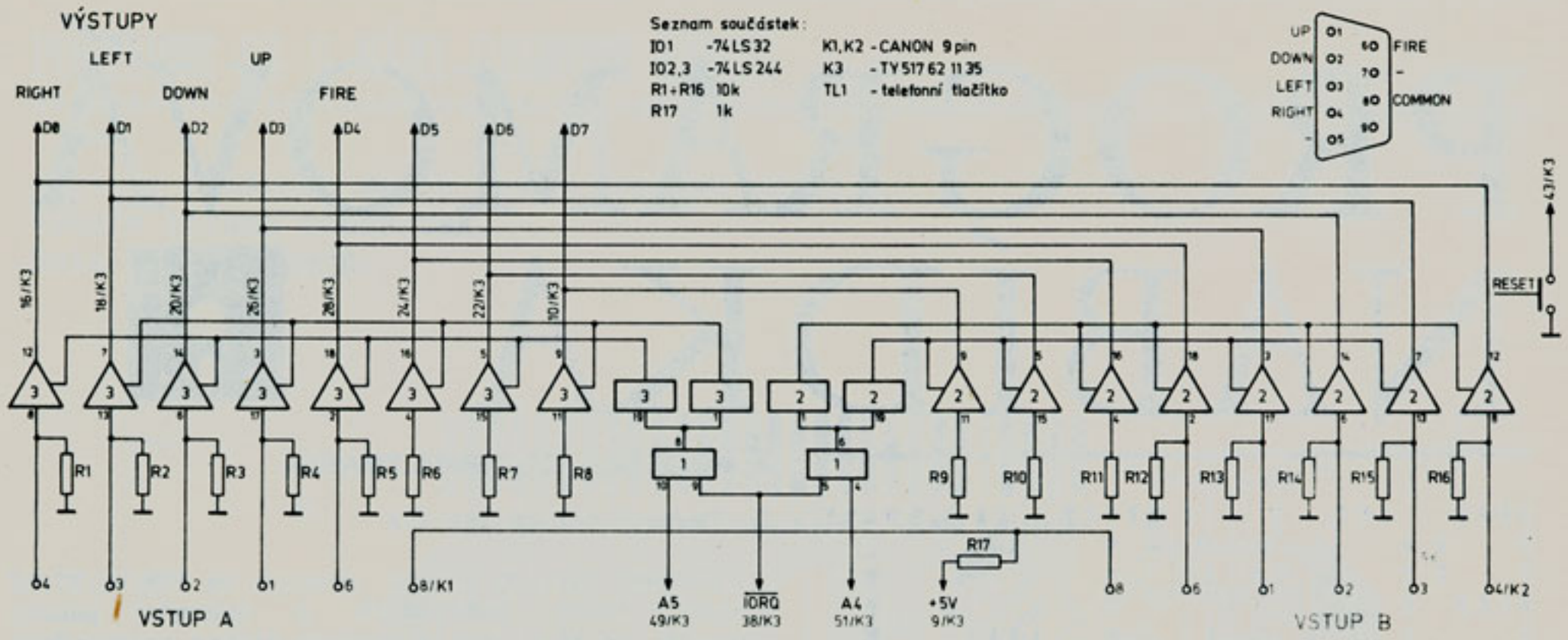
Před připojením ke Spectru zachováme ověřený postup. Proměříme možnost zkratů sousedních pinů konektoru K3, modul připojíme ke zdroji +5V a změříme odběr. Bude cca 50 mA, což je jedna z velkých výhod tohoto zapojení. Obdobný interfejs pro jeden ovladač s MH 3212 má odběr cca 150 mA.

Potom připojíme modul při vypnutém Spectru ke sběrnici ZX. Po zapnutí počítače musí proběhout inicializace dle zvyklostí. Zbývá již jen napsat program pro zkoušku správné funkce modulu a ovladačů (viz předchozí kapitolu). Pak již můžeme vyzkoušet hru, která má v menu volbu pro KEMPSTON JOYSTICK, resp. provést instalaci programu pro náš ovladač (např. Art Studio).

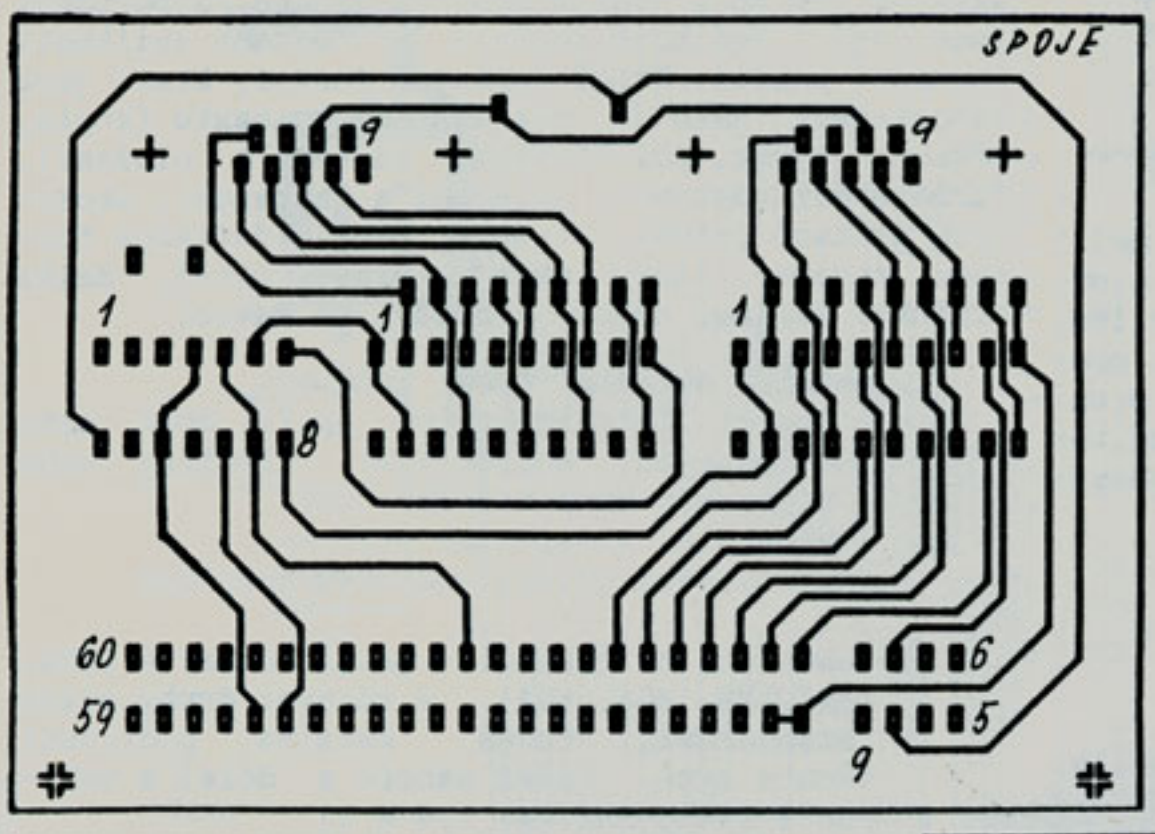
3. Seznam použitých součástek

IO 1	74 LS 32
IO 2,3	74 LS 244
R 1÷16	10 kiloohmů (miniaturní na výšku)
R 17	1 kiloohm
K 1,2	konektor CANON 9 pin
K 3	konektor FRB 62 pin - TY 517 62 11 35
T1 1	tlačítko (např. telefonní)

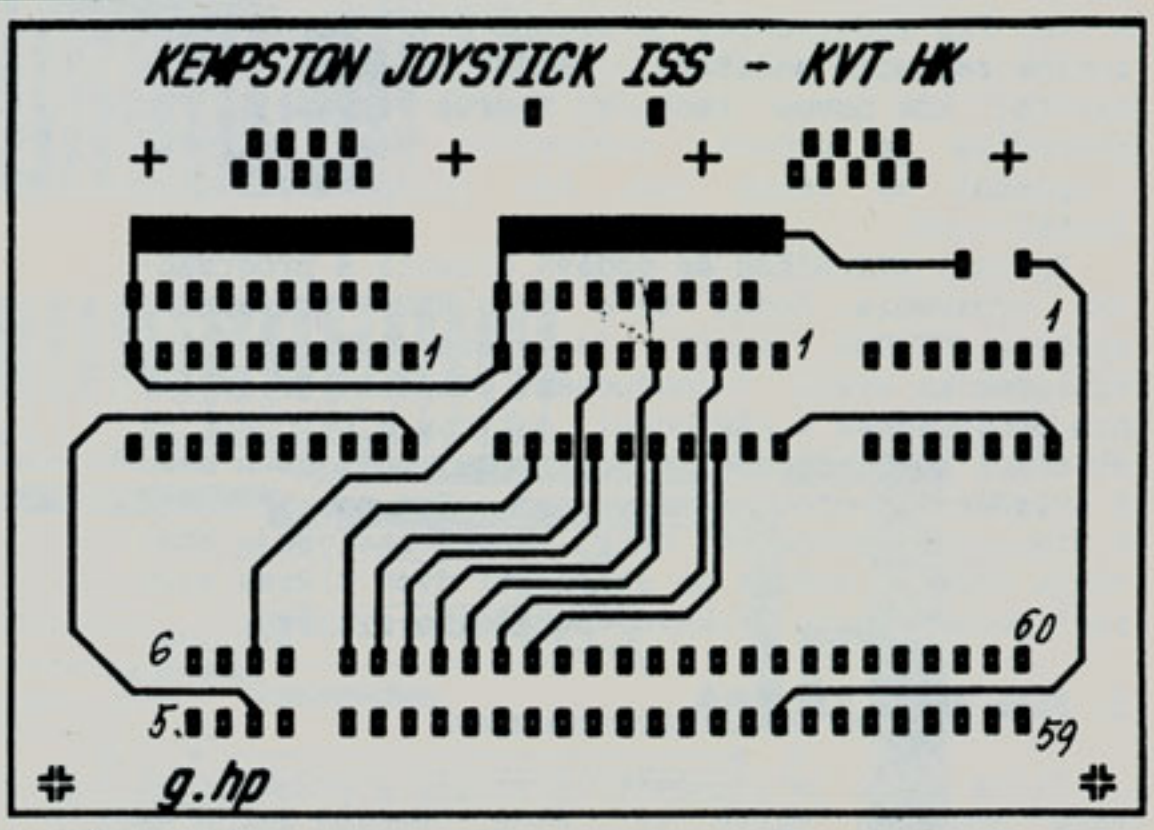
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>



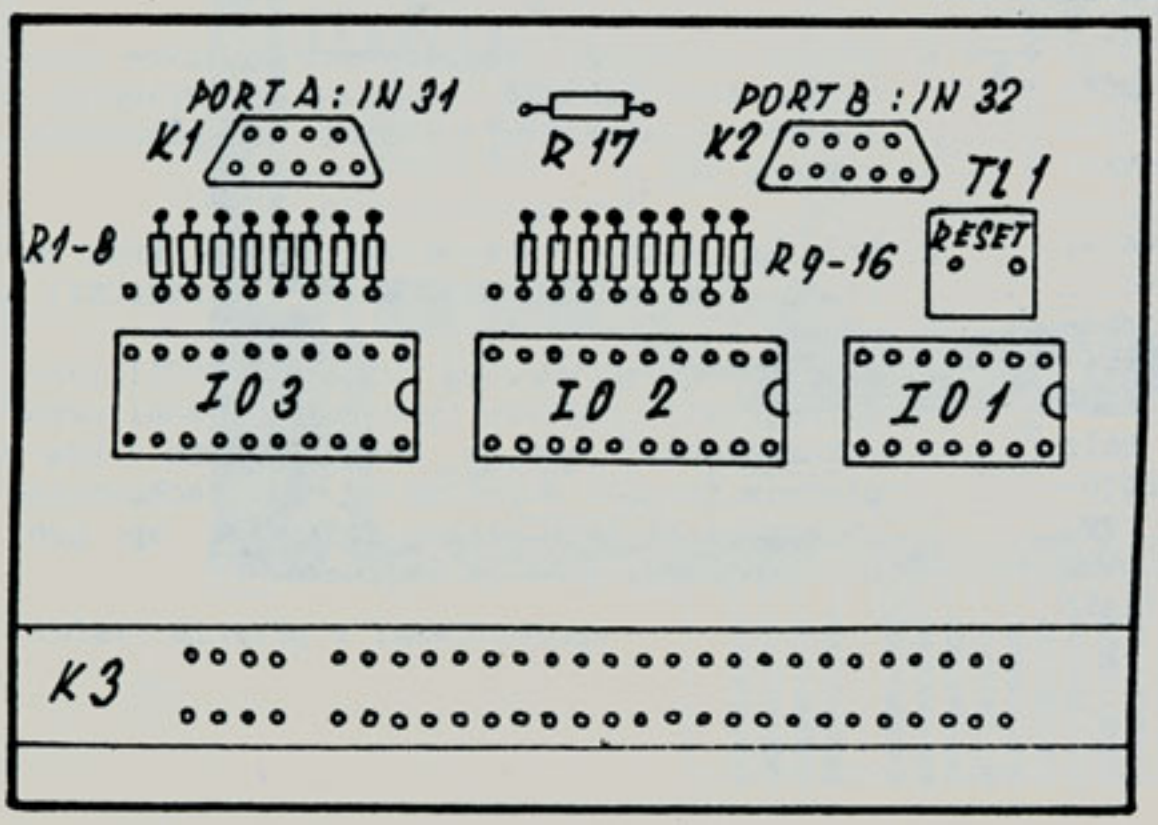
Obr. 1 - schema interfejsu



Obr. 2 - návrh plošných spojů (strana A)

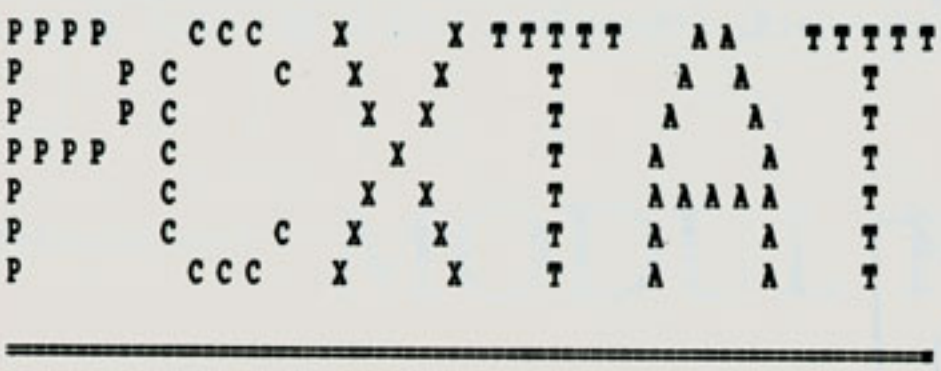


Obr. 3 - návrh plošných spojů (strana B)



Obr. 4 - rozložení součástek

PROGRAMOVÁ NABÍDKA



Naše programové vybavení pro IBM PC lze dělit na tvorbu interaktivních kursů (v nich máme dlouhou tradici - jistě znáte dálkové interaktivní kursy číslicové techniky), dále na aplikační programové vybavení (na trh přichází skutečně první "velký" ryze původní software - textový editor Text602) a konečně na zakázkové programové vybavení (databázové systémy typu dBASE).

A. Interaktivní kursy

1. Kurs MS-DOS

Obsáhlá příručka pro začátečníky i pokročilé, která seznamuje s operačním systémem MS-DOS 3.2 a 3.3. Pro začátečníky je určena první část, která začíná zapnutím počítače - najde uplatnění v podnicích, kde novou techniku teprve zavádějí. Pro pokročilé je určena referenční část příručky - uplatní se všude tam, kde již počítače jsou instalované.

Spolu s příručkou se dodává disketa s programovým vybavením. Jedná se o tzv. HELP operačního systému MS-DOS, který je určen začátečníkům. Vzhledem ke svému jednoduchému ovládání je ideálním prostředkem k překonání ostychu a osvojení si ovládání počítače. Je řízen pomocí pull down menu a ovládán kurzorovými klávesami. Pro pokročilé je k dispozici rezidentní HELP, který analyzuje stavovou řádku a nabízí po stisknutí dvou kláves syntaxi požadovaného příkazu operačního systému.

2. Kurs dBASE III Plus

Cílem kursu je seznámit uživatele s podstatnými rysy programu dBASE III Plus. Výklad kursu je soustředěn kolem výstavby databanky, obsahující informace o pracovnících malého podniku. Příručka je dělena do logických celků - ASSISTENT a jeho použití, popis skupiny příkazů SET, programování a referenční popis příkazů dBASE III Plus. Výklad je důsledně provázen řešenými příklady - uživatel s jejich pomocí postupně buduje svoji databanku.

Spolu s příručkou je dodávána disketa se soubory, na kterých jsou dokumentovány příklady. Uživatel tak může nejen experimentovat, ale i přizpůsobovat si databanku pro svůj konkrétní případ. Dále jsou k dispozici programy pro formátování zdrojových textů v dBASE a pro zrychlení provádění .PRG souborů. Uživatelé Turbo Pascalu uvítají knihovnu pro práci s datovými a indexovými soubory dBASE III Plus.

B. Aplikační programové vybavení

1. Textový editor Text602

Původní textový editor pracuje na IBM/PC/XT/AT kompatibilních počítačích s operační pamětí alespoň 384 KB a jedním pružným diskem (doporučené jsou dva). Velikost souboru je omezena velikostí operační paměti. Kromě běžných funkcí, které jsou samozřejmé, jako je práce s bloky textu (kopie, přesun, výmaz, formátování, načtení a uložení), formátování odstavců, nalezení a nahrazení, skokových příkazů (řádka, stránka, blok), definice formátu stránky (levý okraj, pravý okraj, délka stránky, záhlaví nahoře a dole), má navíc:

1. možnost ovládání třemi způsoby:
   - pomocí inteligentních pull down menu ovládaných kurzorovými tlačítky nebo Microsoft kompatibilní myší
   - vlastními zkrácenými příkazy
   - zkrácenými příkazy editoru WordStar

2. zobrazení na jakékoliv grafické kartě (CGA, HERCULES, EGA, VGA), s různými druhy písma (standardní, tučné, kurzíva, podtržení těchto typů, index nahoře a dole) a velikostí řádkování (1, 1.5 a 2)

3. několik druhů klávesnic: ČSR a SSR totožné s psacím strojem CONZUL, SPC pro speciální symboly (čárová grafika, matematické symboly a znaky západoevropských abeced), ČSa pro programátory a IBM standardní počítačová klávesnice

4. tisk je možný:
   - se standardně dodávanými grafickými drivery na jakékoliv EPSON/IBM kompatibilní maticové tiskárně (9/24 jehel)
   - s drivery pro download na EPSON/IBM/STAR/NEC kompatibilní maticové tiskárně (9/24 jehel)
   - s drivery pro download na HP LaserJet II kompatibilní laserové tiskárně
   - s drivery které si může definovat uživatel

5. dělení slov při formátování odstavce nebo bloku (automatické nebo s potvrzením), včetně přetahování spojek a předložek z konce vět

6. možnost volby vstupně/výstupního kódu ve třech normách (KOI-8čs, LATIN II, KEYBCS2)

7. export souborů ve dvou formátech ASCII (nedokumentační editory typu Turbo Pascal nebo dokumentační, jako je MS-WORD) nebo v kompletním formátu WordStar 3.3 se zachováním informace o typu písma, řádkování ap. pro DTP programy, jako je PageMaker

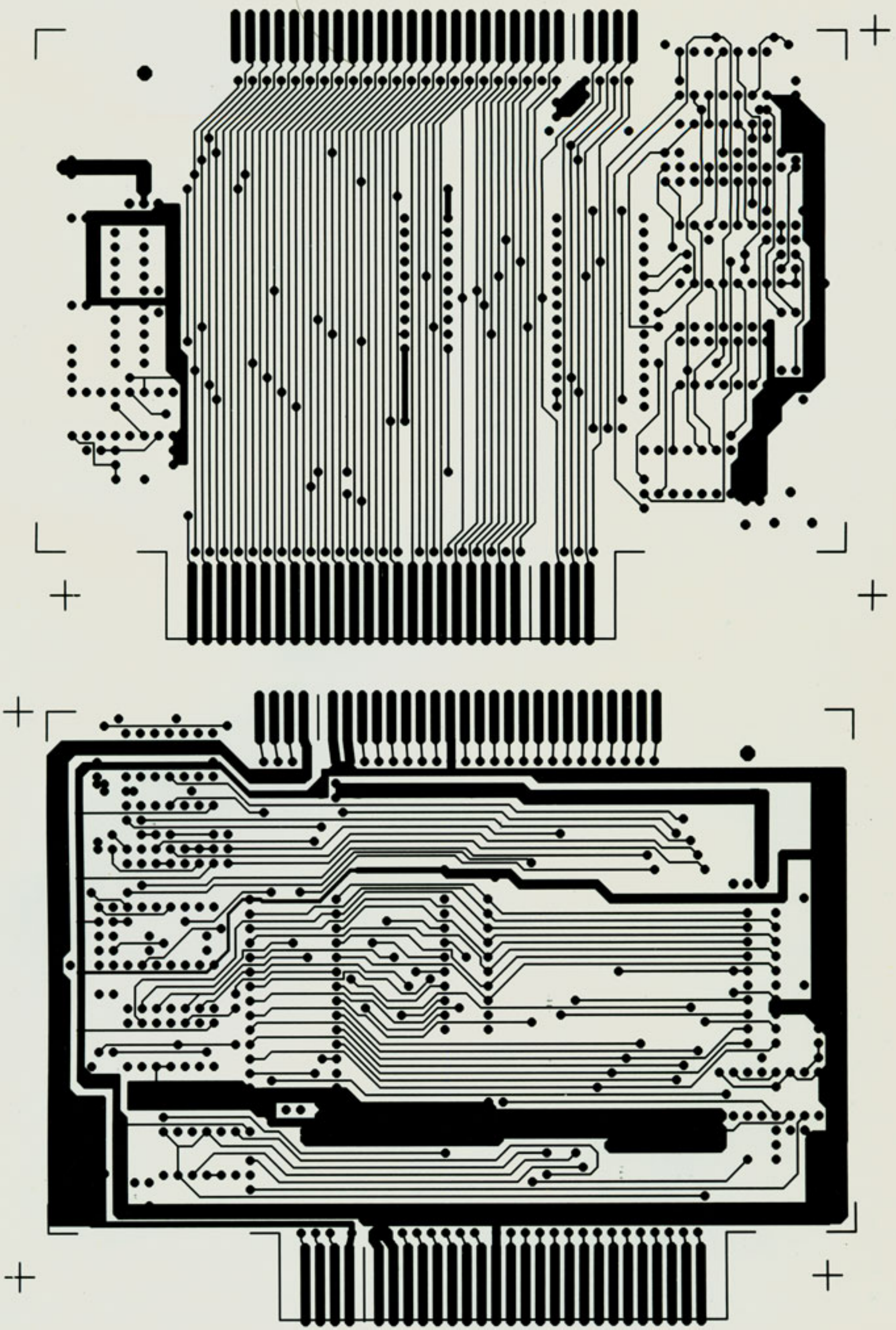
(pokračování v příštím čísle)

****

POSTAVTE SI S NÁMI DISKOVÝ ŘADIČ

****

Obrazec plošných spojů desky systému v5.03.
Nahoře strana součástek.
Dole strana spojů.



* poznámka: v čísle 6 je na straně 11 ve schematu chybně označen vývod PGM EPROM (pin 27) jako NC. Správně má být spojen s +5V, tj. s pinem 28. V obrazci plošných spojů tato chyba není.

ERA 89

TRENČÍN

17.- 26.11.1989

21. CELOŠTÁTNA PREHLIADKA TECHNICKEJ TVORIVOSTI V ELEKTRONIKE A RÁDIOAMATÉRSTVE