KLUB

ATARI

2

91

602

ZE ZÁPISNÍKU PROGRAMÁTORA

TURBO BASIC MATH PACKAGE

Jedná se o čtyři soubory částečně obsahující matematické rutiny pohyblivé řádové čárky "vypreparované" pomocí TEXGENu 1.1 z TBASICu a částečně mnou doplněné do použitelného kompletu (je jich více než sto). Čtvrtý soubor je demonstrační.

Všechny čtyři soubory jsou zdrojové texty do MAC/65 uložené příkazem "SAVE", jejich celková délka přesahuje 40kB. Z knihovny těchto TBM rutin si můžete vybrat jen ty potřebné, a pak je přikompilovat do vašeho vlastního programu. Návodem, jak tuto knihovnu ovládat a používat, a popisem jednotlivých rutin se zabývá tento příspěvek.

Všechny tři knihovní soubory se kompilují pomocí "INCLUDE"

Soubor TBM1 ovládá celou knihovnu a doplňuje všechny potřebné rutiny k rutinám vámi vybraným. Kompiluje se jen při prvním průchodu, proto vyžaduje, abyste si ve vašem programu zavedli proměnnou PASS obsahující číslo průchodu kompilací (1 nebo 2) (viz manuál k MAC/65, kap.8.3). TBM1 musí být zkompilováno před kompilací souborů TBM2 a TBM3. Na pořadí těchto druhých dvou souborů nezáleží. Soubory TBM2 a TBM3 jsou vlastní bankou rutin.

Jednotlivé rutiny vyberete nastavením odpovídajícího bitu v řídících proměnných TBM.COM1 až TBM.COM15 (viz tabulka). Hodnota jim musí být přiřazena před kompilací souboru TBM1. Proměnným, u kterých žádný bit nenastavujete, musíte přiřadit nulu. Je dobré si vybírané rutiny napřed označit na papíře. Předejdete tak chybám při naplňování TBM.COM - proměnných. Např. chcete-li si vybrat jen rutiny násobení a sčítání, nastavíte TBM.COM1 na $09 a ostatní TBM.COM na nulu.

Rutiny jsou psány tak, aby byly co NEJRYCHLEJŠÍ, zkompilované proto zabírají dost místa. Několik bytů se dá ušetřit, povolíte-li odvolávat se při běhu programu na MATH-ROM, kde jsou uloženy různé konstanty. K tomu slouží proměnná TBM.COMMAND, jejíž bity mají tento význam:

0. bit .. smí se užívat MATH-ROM
1. bit .. MATH-ROM je při volání vypnuta
2. bit .. je-li užita rutina VAL.R, je vlastní (viz pozn.3).

Pro TBM.COMMAND platí totéž co pro ostatní TBM.COMy.

Celá knihovna (včetně konstant) zabírá po zkompilování 5-6kB. Jak již bylo uvedeno, nemusíte ji kompilovat celou, stačí si vybrat.

Matematické rutiny používají při výpočtech pomocné registry a proměnné, původně uložení v nulté stránce (ZPAGE). Při volbě jejich názvů jsem vycházel z přílohy ZAK "Důležité adresy systému a jejich použití 1-2". Ne všechny proměnné musí v ZPAGE zůstat. Některé můžete umístit jinam podle potřeby. Ne však všechny, jinak může při kompilaci dojít k chybě "BRANCH RANGE". V ZPAGE

nepoužívá všechny tyto proměnné zároveň, podrobněji se zde o tom nebudu rozepisovat. Dalšími proměnnými, které rutiny využívají jsou (uvedu původní adresy podle TB):

ZSU1=$F0, ZSU2=$F1, SCOUNT=$EF (původně se překrývají s CHARFLG,DIGRT a ESIGN)
LBUFF (adresa textového bufferu)
SR0=$5E0, SR1=$5E6, SR2=$5EC (po šesti bytech; SR0 musí být z nich první, SR1-SR0<=$FA, totéž SR2)
SBSIDE1 až SBSIDE6 (po osmi bytech).

Obecně matematické rutiny vracejí výsledky v registru FR0, první parametr pro výpočet je v FR0, je-li třeba druhého parametru, je v FR1. Registry FR0 a FR1 jsou po 12-ti bytech, naplňuje se však jenom prvních 6 bytů. Např. pro dělení je :

FR0=FR0/FR1.

Nyní již popis rutin. Jednotlivé popisy jsou odděleny středníkem (názvy shodné s TB nekomentuji).

TBM.COM1:
pomocná; dělení 10-ti (/10); *10; /2; sčítání; odčítání; dělení; násobení
TBM.COM2:
pomocná; DIV; MOD; desítkový exponenciál (10^FR0); EXP; druhá odmocnina; obecná mocnina (FR0^FR1); druhá mocnina (POZOR! v TB se takto nazývá odmocnina)
TBM.COM3:
SGN; adresa konstanty pravého úhlu, čtěte ji s posunutím podle RADFLG; FR0=-1; v FR0 vrátí jedničku s exponentem a se znaménkem podle akumulátoru (ten obsahuje 1.byte čísla, viz pozn.1); FR0=1; =2; =3; =0
TBM.COM4:
RAND; RND; RAD; DEG; FR0==+1; FR0==-1; ABS; FR0=-FR0
TBM.COM5:
upraví FR0 do "počítatelného" tvaru, používá se poté co do čísla zasahujete jinak než rutinama z knihovny; totéž pro 7-mi bytové FR0; VAL (viz pozn.3); STR$ (musí být povolena MATH-ROM, nebo si rutinu musíte napsat sami); převede HEX řetězec na CARD číslo (v TB se nazývá DEC); totéž jako HEX, pracuje však s číslem typu CARD (viz pozn.6); převede FR0 na CARD číslo; převede CARD číslo na reálné (opak předchozího)
TBM.COM6:
FRAC; INT; TRUNC; obecný logaritmus FR1 o základu FR0; CLOG; LOG; pomocná; výpočet polynomu v bodě FR0, koeficienty (od nejvyššího stupně) jsou uloženy od adr. z registrů Y (:H) a X (:L), jejich počet je v akumulátoru (min.2)
TBM.COM7:
cyklometrické a goniometrické funkce, počítají se v radiánech nebo ve stupních (podle RADFLG)
TBM.COM8:
hyperbolometrické a hyperbolické funkce
TBM.COM9:
přiřazovací rutiny LD do od, nepotřebují snad komentáře ("P"

adr. FLPTR na adr. FLPTR2
TBM.COM10:
vymění čísla v FR0 a SR2; vymění FR0 a FR1; přiřazovací rutiny
TBM.COM11:
*2; porovná FR0 s FR1, vrátí nastavené praporky procesoru "Z a"CY jako instrukce CMP; v akumulátoru (AC) vrátí výsledek testu FR0=FR1 (0 nebo 1); FR0>=FR1; ostatní podobně
TBM.COM12:
1/FR0; 1-FR0; přiřazování;;; v AC vrátí znaménko FR0 (0,1 nebo -1); AC=1; AC=0 (pomocné)
TBM.COM13:
nepoužito;;; v INBUFF přeskočí mezery; nastaví INBUFF na $580; přečte do AC znak z INBUFF, je-li "CY=0,převede jej na číslo; je znak v INBUFF číslice? (vrátí "CY); je znak šesnáctková číslice? (-"-)
TBM.COM14 a TBM.COM15 jsou nepoužity, nula se jim však přiřadit musí.

Poznámka 1:

Zde používaná reálná čísla jsou totožná s reálnými čísly z MATH-ROM. Tedy: jsou dlouhá 6B, v 1. bytu je exponent se znaménkem a v 7. bitu znaménko celého čísla. Dalších 5 bytů je vlastní číslo v BCD formátu.

Poznámka 2:

Při výpočtech platí:
-příliš malá čísla (menší než cca 1E-95) jsou zaokrouhlena na nulu, například EXP(-200)=0
-příliš velká čísla (větší než cca 1E95) jsou chybou
-některé rutiny vracejí oznámení, zda nedošlo k chybě při výpočtech, v tabulce jsou označeny "#". Je-li u nich po návratu "CY=1, chyba nastala, pro "CY=0 je výsledek v pořádku. Musím upozornit, že rutina POWER.R zde nerozlišuje, na rozdíl od TB, typ chyby, která nastala.

Poznámka 3:

Rutina VAL.R má zde trochu odlišné vlastnosti od rutiny z MATH-ROM:
-při chybě umístí kursor (CIX) na místo, kde chyba nastala
-je chybou např. 1E100 (na druhé nule se zastaví), MATH-ROM rutina přečte 1E10 a druhá nula v textovém bufferu zbyde
-má-li převáděné číslo lichý exponent, je možno převést celých 10 platných cifer, ne 9 jako v MATH-ROM rutině. Pro sudý exponent zůstává maximální počet cifer stále 9.

Poznámka 4:

Celá knihovna tvoří jednu lokální oblast (.LOCAL)

Poznámka 5:

Používáte-li rutiny z ROM, nesmíte měnit původní adresy návěští, které tyto rutiny užívají (týká se hlavně VAL.R a

STR.R).

Poznámka 6:

CARD čísla jsou dvoubytová čísla bez znaménka uložená na FR0,FR0+1.

A nakonec ještě popis demonstračního programu (soubor č.4 -"BASIC.M65"). Jedná se o ukázku použití uvedené aritmetiky v ATARI BASICu. Běh programu v BASICu, sám o sobě, se zrychlí minimálně, výrazně se však urychlí matematické výpočty, což si můžete ověřit např. tzv. Benchmark testy (Olomoucký zpravodaj 1-2/88).

Tento demo program modifikuje BASIC -ROM. Jednotlivé jeho části se nahrávají přímo do kopie BASICu. Zbytek matematických rutin zůstává uložen nad DOSem (od $2000).

Abyste mohli program přeložit, musíte:

1. Použít DOSu s disketou nebo TT-DOSu s RAM -DISKem.
2. Udělat několik úprav v souborech TBM2 a TBM3 (jsou popsané na začátku souboru BASIC.M65). Tyto úpravy ale nedělejte v originálních souborech knihovny, nejlépe, pokud máte rozšířenou paměť, je provádějte jen v RAM -DISKu.

Pokud budete knihovnu používat ve vašem vlastním programu, nic v ní měnit nemusíte. V tomto speciálním případě se jedná o modifikaci již existujícího programu (BASICu). Rozhodně vám nedoporučuji provádět v knihovně vlastní změny bez důkladné znalosti jejího obsahu.

3. Po provedení změn (je jich málo a jsou jednoduché) nahrajte do MAC/65 soubor č.4 a upravte si zařízení v řádcích s INCLUDE podle toho, kde máte uloženy knihovní (upravené) soubory. Pokud máte pouze kazetu bez RAM -DISKu, budete mít velké problémy nebo spíš smůlu.

Přeložený program pak nahrajte a spusťte z DOSu příkazem "L", nastaví se BASIC. Chcete-li teď používat ROM aritmetiku, zadejte POKE 54017,253, POKE 54017,255 nebo RESET nastaví aritmetiku zrychlenou.

Martin Plechšmíd

Tabulka TBM rutin:

TBM.COM\bit 7	6	5	4	3	2	1	0
1 ?LSH4.FR0	DIVIDE10.R	MUL10.R	DIVIDE2.R	#ADD.R	#SUB.R	#DIVIDE.R	#MUL.R
2 ?SCAT.FR1	#DIV.R	#MOD.R	#CEXP.R	#EXP.R	#SQRT.R	#POWER.R	#SQR.R
3 SIGN.R	RGHT.ANGLE	R.CNST.N1	R.CNST.?1	R.CNST.1	R.CNST.2	R.CNST.3	R.CNST.0
4 RAND.R	RND.R	SET.RAD	SET.DEG	ICREASE.R	DCREASE.R	ABS.R	MINUS.R
5 #SH.FR0	#SH.FR0E	#VAL.R	STR.R	#VAL.H	HEX.C	#CARD.R	REAL.C
6 FRAC.R	INT.R	TRUNC.R	#LOG.R	#L6.R	#LN.R	?EVAL.FR0.P	#POLYNOMIAL.R
7 #ARCCOTG.R	#ARCTG.R	#ARCCOS.R	#ARCSIN.R	#COTG.R	#TG.R	#COS.R	#SIN.R
8 #ARGCOTGH.R	#ARGTGH.R	#ARGCOSH.R	#ARGSINH.R	#COTGH.R	#TGH.R	#COSH.R	#SINH.R
9 LD.SR2.FR0	LD.SR1.FR0	LD.SR0.FR0	LD.FR1.FR0	LD.P.FR0	LD.FR1.P	LD.FR0.P	ASSIGN.R
10 CHANGE.FR0.SR2	CHANGE.FR0.FR1	LD.FR1.SR2	LD.FR1.SR1	LD.FR1.SR0	LD.FR0.SR2	LD.FR0.SR1	LD.FR0.SR0
11. MUL2.R	COMPARE.R	EQUAL.R	GREAT.EQ.R	GREATER.R	LESS.R	NON.EQ.R	LESS.EQ.R
12 #INVERSE.R	#SUB1.R	LD.SR2.FR1	LD.SR1.FR1	LD.SR0.FR1	SIGN?.R	?B.CNST.1	?B.CNST.0
13	nepouzito		SKIP.SP	SET.INBUFF	GET.FIG	FIGURE?	HFIGURE?
14		nepouzito					
15		nepouzito					

UŽIVATELSKÉ PROGRAMY

VERIFY II a MIKRONOTES

Mnohé ze čtenářů již jistě napadla myšlenka, použít data získaná pomocí programu VERIFY II jako základ kartotéky programů v databance MIKRONOTES. Bohužel, snadné realizaci této myšlenky brání několik překážek:

1. Zařízení "T:" programu VERIFY II není v pořádku. "Uřezává" konce souboru, takže několik záznamů, které program původně načetl, chybí.

2. Program MIKRONOTES "nesnáší" některé znaky, které se v souborech z VERIFY II vyskytují. Jedná se o znak EOF ($9B), který odděluje jednotlivé záznamy, dále o znaky jako ESC, "*", znaky inverzní a pod., které v názvech programů občas nalézáme.

3. Struktura souboru z VERIFY II neodpovídá struktuře kartotéky. Jde o názvy kazet a záhlaví tabulky.

Naštěstí nejde o překážky nepřekonatelné. První překážku odstraníme úpravou programu VERIFY II, druhé dvě pomocí programu, který soubor z VERIFY II upraví tak, aby mohl být využit v programu MIKRONOTES.

Úprava VERIFY II

V našem klubu kolují dva programy VERIFY II označené jako "VERIFY II+" a "VERIFY BT1". První z nich po nahrání požaduje zadat kontrolní kód. Druhý kontrolní kód nevyžaduje a má "zabudován" obslužný program pro tiskárnu BT 100. Existuje údajně ještě verze s obslužným programem pro zapisovač ALFIGRAF, avšak v našem klubu ji nemáme k dispozici.

K úpravě programů použijeme monitor TM 2004+. Klávesou <H> přepneme na hexadecimální tvar čísel. Pomocí funkce "TBOOT" (klávesa <O>) nahrajeme upravovaný program. VERIFY II+ je umístěn od adresy $0900, dlouhý je $1A00 bajtů a startovní adresu má $0C10. VERIFY BT1 je také umístěn od $0900, je však dlouhý $1BC0 bajtů a startovní adresu má $0F05. Tyto hodnoty je vhodné si před vlastní úpravou programu ověřit. Pokud nesouhlasí, může se jednat o nějakou další verzi a dále popsaný způsob opravy nemusí platit.

Vlastní úpravu provedeme pomocí funkce "MONITOR" (klávesa <M>). Postupně měníme podle dále uvedené tabulky obsahy adres. Změna hodnot na adresách $1C53, $1C54, $1BE8 a $1BE9 je nejdůležitější. Odstraňuje zmíněné "uřezávání" konce souboru. Úprava na adresách $1D99, $1DB8 a $1DF8 zlepšuje čitelnost nahrávek "turbo". Nezbytně nutná není ani změna hodnot na adresách $0C1E až $0C28. Na těchto adresách je umístěna ve VERIFY II+ žádost o zadání vstupního kódu, kterou změnou hodnot odstraníme. Po úpravě již vstupní kód nebude požadován. U VERIFY BT1 tato úprava samozřejmě odpadá.
Tabulka úprav programů VERIFY II+ a VERIFY BT1:

Na adrese	VERIFY II+ je	VERIFY II+ změnit na	I	VERIFY BT1 je	VERIFY BT1 změnit na
0C1E:	FF	C8	I	--	--
0C1F:	8D	EA	I	--	--
0C20:	FC	EA	I	--	--
0C21:	02	EA	I	--	--
0C22:	AD	EA	I	--	--
0C23:	FC	EA	I	--	--
0C24:	02	EA	I	--	--
0C25:	C9	EA	I	--	--
0C26:	FF	EA	I	--	--
0C27:	F0	EA	I	--	--
0C28:	F9	EA	I	--	--
1C53:	5A	37	I	22	FF
1C54:	56	42	I	1E	0A
1BE8:	5A	37	I	22	FF
1BE9:	56	42	I	1E	0A
1D99:	43	BB	I	0B	83
1DB8:	43	BB	I	0B	83
1DF8:	43	BB	I	0B	83

Opravený program znovu uložíme na kazetu pomocí funkce "TURBOSAVE" (klávesa <R>). Bude vhodné mírně změnit jméno programu v hlavičce nahrávky, abychom rozeznali opravené verze od chybných. Navrhuji VERIFY II+ změnit na "VERIFY II#" a VERIFY BT1 na "VERIFY BT#".

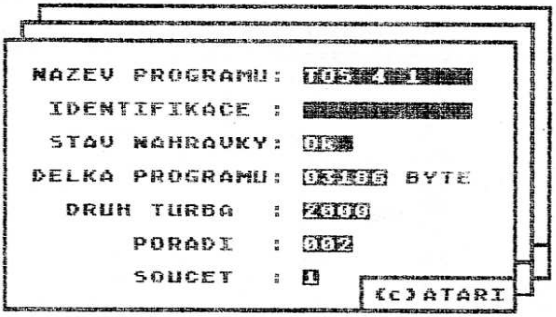
Úprava dat z VERIFY II

Data z VERIFY II musíme před použitím v MIKRONOTESu upravit. K tomu je velmi dobře použitelný program "VERIFY II => MIKRONOTES". Jeho autorem je pan Milan Hájek. Jedná se o převodník, který do paměti "natáhne" soubor uložený programem VERIFY II, odstraní z něj nežádoucí znaky a řetězce znaků a takto upravená data znovu uloží na kazetu jako soubor "T:" Mikronotesu (hlavička a jeden blok dat). Tento soubor je možné nahrát přímo do databanky.

Použití je velmi jednoduché. Po spuštění se jednou ozve bzučák. Připravíme v magnetofonu kazetu s upravovaným souborem dat, spustíme magnetofon a stlačíme kteroukoli klávesu. Program začne natahovat soubor. Po natažení každého bloku dat se motor magnetofonu na chvíli zastaví. V těchto okamžicích vždy stiskneme tlačítko PAUSE, aby se do mgf. pásku nevytlačil "drop out". Tlačítko PAUSE samozřejmě uvolníme jakmile chce program nahrávat další blok. Je-li soubor nahrán a upraven, nabídne nám program tři možnosti. Stiskem <T> zvolíme návrat do TOSu, stiskem <S> nebo <D> volíme uložení upraveného souboru na kazetu. Volbou "Soubor" <S> uložíme upravená data jako soubor Mikronotesu, tj. data i se vstupním formulářem, viz. obrázek. Volbou "Data" <D>

budou uložena data bez vstupního formuláře, tj. jako data Mikronotesu.

Obrázek "zabudovaného" vstupního formuláře:



Při používání programu je nutné respektovat tato omezení:

- program pro svou práci vyžaduje prostředí TOSu 4.1.
- upravená data jsou ukládána ve formátu "T:" Mikronotesu.
- při prohlížení obsahu kazet programem VERIFY II je nutné ukládat data každé kazety jako samostatný soubor. Před prohlížením další kazety je nutné data o předchozí kazetě vymazat.

-js-

Výpis programu VERIFY II => MIKRONOTES:

```
EJ 1 REM ************************************
PN 2 REM *  VERIFY II => MIKRONOTES  *
PZ 3 REM *  (C) Milan HAJEK 1990  OK602  *
CR 4 REM *==============================*
UO 5 REM * PRACUJE POUZE POD TOSem 4.1 *
ED 6 REM * ---------------------------- *
EO 7 REM ************************************
NN 8 REM
KK 10 TRAP 110
RE 15 DIM A$(25000),HL$(20),5$(152)
PI 20 5$(1,76)="┌──────────────────────────┐ |KONVERTOR
   DAT MEZI VERIFY II A MI-|  "
ID 25 5$(77,152)="|KRONOTESEM                  |  └──────
   ──────(c)1990 Milan HAJEK┘"
DV 30 GRAPHICS 2:POSITION 0,4:? #6;"PRIPRAV MAGNETOFON SNAHRAVKOU
    DAT Z PRO-GRAMU verify ii"
MF 35 POKE 710,192:POKE 712,192:POKE 752,1:? 5$;
UI 40 GOSUB 295
PX 45 I=USR(ADR("hhh [CTRL]♦"),1)
BN 50 GRAPHICS 2:POSITION 2,4:? #6;"stiskni tlacitko       - PAUS
   E -          na magnetofonu"
MH 55 POKE 710,192:POKE 712,192:POKE 752,1:? 5$;
MZ 60 OPEN #1,4,0,"D:"
VJ 65 FOR I=1 TO 90:GET #1,A:NEXT I:I=0
MA 70 I=I+1:GET #1,A
WZ 75 IF I=21 OR I=25 OR I=31 OR I=36 OR I=40 THEN 70
OC 80 IF I=41 THEN I=0
XY 85 IF A=155 THEN A=49
ZA 90 IF A=46 THEN A=32
```

```
FR 95 IF A)127 THEN A=A-128
UT 100 A$(LEN(A$)+1)=CHR$(A)
RZ 105 GOTO 70
NK 110 IF PEEK(195)<>136 THEN GRAPHICS 0:POKE 710,192:? :? :? "CH
       YBA CISLO ";PEEK(195):END
FT 115 A$=A$(1,466+(INT((LEN(A$)-466)/36)*36))
UZ 120 GOSUB 210:GOSUB 250
KV 125 A$(LEN(A$)+1)=CHR$(155)
KB 130 HL$(14,14)=CHR$(LEN(A$)-256*INT(LEN(A$)/256))
ME 135 HL$(15,15)=CHR$(INT(LEN(A$)/256))
GS 140 ZAC=ADR(HL$):KON=ZAC+LEN(HL$):GOSUB 180
JE 145 GRAPHICS 2:POSITION 1,4:? #6;"PRIPRAV MAGNETOFON      NA NA
       HRAVANI"
RO 150 POKE 710,192:POKE 712,192:POKE 752,1:? 5$;
RZ 155 I=USR(ADR("hhh [illegible]"),2)
MD 160 I=USR(ADR("hhh +| ♦"),0)
EK 165 ZAC=ADR(A$):KON=ZAC+LEN(A$):GOSUB 180
NK 170 I=USR(ADR("hhh +| ♦"),255)
NF 175 GOTO 120
IR 180 REM ■■■ NAHRAVANI ■■■
VO 185 POKE 50,ZAC-256*INT(ZAC/256)
ZR 190 POKE 51,INT(ZAC/256)
NK 195 POKE 52,KON-256*INT(KON/256)
HU 200 POKE 53,INT(KON/256)
ZN 205 RETURN
NO 210 REM ■■■■■ POCET ZAZNAMU ■■■■■
KK 215 P=(LEN(A$)-466)/36
PU 220 H=INT(P/100):P=P-H
NN 225 LH=INT(P/10):LL=P-LH*10
MV 230 L=16*LH+LL
GD 235 A$(448,448)=CHR$(L)
DE 240 A$(449,449)=CHR$(H)
ZV 245 RETURN
AJ 250 REM ■■■■■ SOUBOR/DATA ■■■■■
GF 255 GRAPHICS 2:CLOSE #2:OPEN #2,4,0,"K:"
OK 260 POSITION 0,3:? #6;"CHCETE NAHRAT":? #6;"              sOUBO
       R          NEBO dATA ?"
OT 265 ? #6:? #6;"            ( tOS )"
RT 270 POKE 710,192:POKE 712,192:POKE 752,1:? 5$;
UR 275 GET #2,A:IF A=83 THEN HL$="[illegible]T:SOUB.MKN♥      ":GOSUB 210:RE
       TURN
RO 280 IF A=84 THEN I=USR(PEEK(10)+256*PEEK(11))
PL 285 IF A<>68 THEN 275
EA 290 A$=A$(467,LEN(A$)):HL$="[illegible]T:DATA.MKN♥      ":RETURN
RT 295 A$(1,90)="♥-QRRRRRRRRRRRRRRRRRRRRRRRRRRRRE♥◢QXRRRRRRRRRRR
       RRRRRRRRRRRRRRRRRE|♥\QXRRRRRRRRRRRRRRRRRR"
QR 300 A$(91,180)="RRRRRRRRRE||♥\|♥↓|||♥\|♥├.!:%6♥├02/'2!-5└♥[illegible]♥
       ||||♥\|♥↓|||♥\|♥ )$%.4)&)+!#%♥├└♥[illegible]♥ ||||♥\|"
EC 305 A$(181,270)="♥↓|||♥\|♥ |34!6♥├.!(2!6+9└♥[illegible]♥◢|||♥\|♥↓|||♥\|
       ♥├$%,+!♥├02/'2!-5└♥[illegible]♥[illegible]94%♥ ||||♥\|♥↓|||♥\|♥"
WB 310 A$(271,360)="┘$25(♥├452[illegible]!♥ |└♥[illegible]♥◢|||♥\|♥↓|||♥\|♥\0/2!$)♥|
       └♥[illegible]♥◢|AD♥\|♥↓AD|♥\|♥\3/5#%4♥ |└♥[illegible]├♥ |QRRRRRR"
EX 315 A$(361,450)="RRDAC♥\|♥●|◢c ◢!4!2)AC♥◢ZRRRRRRRRRRRRRRRRRRRX
       RRRRRRRRC♥V\[illegible]┤[illegible]┘[illegible]┐[illegible]┤[illegible]┘[illegible]├6♥[illegible]♥♥"
UN 320 A$(451,466)="♥├♥ |♥┘♥┤♥┐♥/♥\[illegible]├"
YL 325 A$(250,250)=CHR$(34)
BK 330 A$(281,281)=CHR$(34)
ZU 335 RETURN
```

HARDWARE

PSACÍ STROJ CONSUL (2)

V minulém čísle zpravodaje jsme popsali realizaci řídící jednotky, jejímž prostřednictvím počítač komunikuje s elektrickým psacím strojem CONSUL řady 2xx EC. Také jsme uvedli, že k řídící jednotce je ještě třeba napsat obslužný program, tzv. "handler" nebo také někdy "driver", který nám teprve umožní psací stroj používat jako výstupní zařízení našeho ATARI. Takový handler si nyní popíšeme.

Kód kláves

Programové řešení handleru přímo závisí na kódu jednotlivých kláves, který vysílá kombinátor po stisku klávesy na konektor KI. Proto je dále uveden kód, který byl experimentálně zjištěn u autorova exempláře. (Opět vás upozorňujeme, že jednotlivé typy psacích strojů se mohou dost výrazně lišit. Proto je nutné všechny dále uváděné skutečnosti u vašeho stroje ověřit a případně zkorigovat!). Hvězdičkou označené kódy jsou ty, které jsou využitelné v našem případě aplikace - tiskárna počítače.

G	F	D	C	B	A	dec		znak	G	F	D	C	B	A	dec		znak
0	0	0	0	0	0	0		P-	1	0	0	0	0	0	32		VYP
0	0	0	0	0	1	1	*	TAB-	1	0	0	0	0	1	33	*	TAB
0	0	0	0	1	0	2		SP	1	0	0	0	1	0	34		D2
0	0	0	0	1	1	3	*	1...	1	0	0	0	1	1	35		NPK
0	0	0	1	0	0	4	*	!	1	0	0	1	0	0	36	*	,
0	0	0	1	0	1	5	*	A...	1	0	0	1	0	1	37	*	?
0	0	0	1	1	0	6	*	I	1	0	0	1	1	0	38	*	Z
0	0	0	1	1	1	7	*	H	1	0	0	1	1	1	39	*	Y
0	0	1	0	0	0	8	*	G	1	0	1	0	0	0	40	*	X
0	0	1	0	0	1	9	*	F	1	0	1	0	0	1	41	*	W
0	0	1	0	1	0	10	*	E	1	0	1	0	1	0	42	*	V
0	0	1	0	1	1	11	*	D	1	0	1	0	1	1	43	*	U
0	0	1	1	0	0	12	*	C	1	0	1	1	0	0	44	*	T
0	0	1	1	0	1	13	*	B	1	0	1	1	0	1	45	*	S
0	0	1	1	1	0	14	*	A	1	0	1	1	1	0	46	*	:
0	0	1	1	1	1	15	*	+	1	0	1	1	1	1	47	*	0
0	1	0	0	0	0	16		FP	1	1	0	0	0	0	48		DAT
0	1	0	0	0	1	17	*	TAB+	1	1	0	0	0	1	49		PS
0	1	0	0	1	0	18		L-	1	1	0	0	1	0	50		SPK
0	1	0	0	1	1	19		D1	1	1	0	0	1	1	51		NP
0	1	0	1	0	0	20	*	/	1	1	0	1	0	0	52		STOP
0	1	0	1	0	1	21	*	háček	1	1	0	1	0	1	53	*	(
0	1	0	1	1	0	22	*	R	1	1	0	1	1	0	54	*	9
0	1	0	1	1	1	23	*	Q	1	1	0	1	1	1	55	*	8
0	1	1	0	0	0	24	*	P	1	1	1	0	0	0	56	*	7
0	1	1	0	0	1	25	*	O	1	1	1	0	0	1	57	*	6
0	1	1	0	1	0	26	*	N	1	1	1	0	1	0	58	*	5

G	F	D	C	B	A	dec	znak	G	F	D	C	B	A	dec	znak
0	1	1	0	1	1	27	* M	1	1	1	0	1	1	59	* 4
0	1	1	1	0	0	28	* L	1	1	1	1	0	0	60	* 3
0	1	1	1	0	1	29	* K	1	1	1	1	0	1	61	* 2
0	1	1	1	1	0	30	* J	1	1	1	1	1	0	62	* 1
0	1	1	1	1	1	31	* "	1	1	1	1	1	1	63	* mez.

Na první pohled je patrné, že tento kód je naprosto jiný, než kód ASCII. Může vás zarazit, že neobsahuje malá písmena. Na psacím stroji se ale malá a velká písmena píší stejnou klávesou, ovšem se stisknutým nebo uvolněným přeřazovačem. Malá písmena jsou tedy zbytečná. Horší však je, že kód neobsahuje množinu speciálních znaků, které jsou využívány ve zdrojových textech programů. Jsou to např. znaky *, $, #, , a pod. Autor tento problém vyřešil tak, že tiskne háček či čárku nad ty znaky, kde se normálně nepoužívají. Např. "*" je "x" s háčkem, ">" je "v" s háčkem, "<" je "m" s háčkem, "$" je "s" s čárkou, a pod. Bohužel, u autorova stroje nejde programově ovládat posun válce o jeden znak zpět, a tedy není možné mnohé ze speciálních znaků tisknout přetiskem.

Handler

Obslužné programy periferií na ATARI mají většinou stejnou základní strukturu, aby mohli být standardně využívány, tj. aby normálně pracovaly se všemi I/O instrukcemi jak z Basicu, tak z jakýchkoli jiných programů. Každý takový handler se skládá z těchto částí:

1. tabulka handleru (handler entry point table) - je to tabulka, která obsahuje dvoubajtové adresy šesti základních obslužných rutin, zmenšené o 1.
2. šest základních obslužných rutin: OPEN, CLOSE, GET BYTE (READ), PUT BYTE (WRITE), GET STATUS a SPECIAL. Tyto rutiny realizují vlastní obsluhu periferie.
3. inicializační rutina - zařazuje handler do operačního systému.

Operační systém volá všechny periferie přes tabulku obslužných programů (handler table). V této tabulce jsou uvedeny jednak názvy zařízení (P, C, E, S, K) a jednak od každého zařízení adresa tabulky jeho handleru. Tabulka začíná na adrese $031A. Zařazení handleru do operačního systému - inicializace - tedy spočívá v nalezení volného místa v tabulce obslužných programů a v zapsání názvu zařízení a adresy jeho tabulky handleru.

Také náš handler psacího stroje je řešen výše popsaným způsobem. Jeho zdrojový text je uveden na konci článku.
Cílový strojový kód zabírá šestou stránku paměti a část páté.
Rutina INIT (INITB) provádí inicializaci, tj. zařazení

se pak již nepodílí. Při inicializaci se na obrazovce vypíše hlášení "P:ok".

Podprogram OPEN jednak kontroluje, zda je IOCB otvíráno pouze pro zápis. Pokud ne, vrací rutinou GETB chybový kód 131. Dále kontroluje, zda je psací stroj připojen. Pokud není, vrací prostřednictvím TIMEO chybový kód 138. Pokud je vše v pořádku, nastaví všechny bity joystick portu na výstup a provede na psacím stroji operaci "návrat válce" . Následně v Y-registru vrátí "1", což je pro operační systém příznak, že je vše "OK".

CLOSE rutina slouží k uzavření IOCB. Vrací v Y-registru "1". Mohla by také uvádět joystick port do původního stavu, avšak pro úsporu místa od toho bylo upuštěno.

GETB - jak již bylo uvedeno, tato rutina vrací v Y-registru chybový kód 131.

GET STATUS a SPECIAL. Tyto rutiny jsou v našem případě zbytečné a proto jsou realizovány instrukcí "RTS" - GETS.

Nejdůležitější rutinou je PUTB. Ta provádí vlastní manipulaci s psacím strojem prostřednictvím podprogramu ZNAK. Tento podprogram v akumulátoru očekává kód, který má poslat přes joystick port řídící jednotce. Nejdříve kontroluje, zda psací stroj může přijmoout znak, tj. zda je signál "BUSY" na vstupu TRIG1 logická nula. Pokud není čeká. Pokud je, nastaví bity jostick portu P0 až P6 podle obsahu akumulátoru. Poté pošle signál STROBE nastavením bitu P7 na logickou "1" a vrací se do PUTB. Ukončení signálu STROBE, tj. nastavení P7 na "0" je prováděno při zpracování dalšího znaku. (Všechny operační kódy jsou menší než 128).

Také rutina PUTB v akumulátoru očekává kód znaku, který má zpracovat. Nejdříve testuje zda se nejedná o kód tabulátoru. Pokud ano, vytiskne tolik mezer, kolik odpovídá číslu na adrese $C9. Dále testuje, jde-li o návrat válce. Jestliže ano, provede jej, zvýší čítač řádků o jednu a kontroluje, je-li vytisknuta celá stránka srovnáním čítače řádků COUNT s obsahem adresy 43 ($2B) RADKY (ICAX2Z). Je-li stránka vytisknuta čítač řádků je vynulován. Počítač 3x zapípá a čeká na stisk jakékoli klávesy. V této době uživatel založí nový papír do stroje a poté stiskne klávesu. Po stisku klávesy je proveden opět návrat válce.

Jestliže kód není ani návrat válce, ani tabulátor, odfiltruje rutina 7. bit kódu, čímž jsou případné inverzní znaky převedeny na normální. Grafické znaky se ignorují a nahražují mezerami, takže je možné je do vytisknutého textu doplnit. Jde-li o písmena, jsou malá převedena na velká odečtením 32 od kódu a stroj nastaven na malá písmena (kód "1...", při velkých písmenech je poslán kód "A..."). Nyní se ještě testuje, zda jde o písmeno s háčkem nebo čárkou (náhrada speciálních znaků ATASII). V případě kladného výsledku je vytisknuto příslušné znaménko. Naposled je vytisknut i žádaný znak. Po vytisknutí znaku rutina vrací v Y-registru "1".

Tabulka TAB slouží k převodu kódu ATASCII na kód psacího stroje. Pokud ji budete detailněji studovat, zjistíte, že ATASCII kódy jsou převáděny na kód inverzní ke kódu psacího stroje. Důvod je prostý. Autor při vývoji řídící jednotky použil k jejímu oddělení od počítače obvod 8287, který však signály invertuje. Původní verze popisovaného obslužného programu tedy byla vyvinuta pro invertované signály, a proto byly všechny kódy posílané na

port inverzní, což se odrazilo i v převodní tabulce. Pro námi popisovanou verzi řídící jednotky však byly použity cenově i odběrově výhodnější obvody MH74ALS08, které však signály neinvertují. Abychom nemuseli předělávat celý program, byl upraven podprogram ZNAK, který nyní invertuje signály před vysláním na port, čímž jsme dosáhli ekvivalentní funkce. Jedná se o řádky 1565 a 1580.

Poznámka:

Protože při inicializaci není ukládána žádná hodnota na adresu RADKY, je nutné ji uložit před tiskem buď příkazem POKE, nebo příkazem pro otevření kanálu:

```
OPEN #1,8,x,"P:"
```

kde x je počet řádků. Spolehlivější by však bylo (pokud bychom handler umístili do jiné oblasti a nemuseli šetřit místem) rezervovat pro tuto proměnnou adresu někde v oblasti obslužných rutin a ukládat do ní max. počet řádků při inicializaci zařízení rutinou INIT. Také by se tím zabránilo jejímu náhodnému přepsání při otevírání dalšího IOCB. Operační systém totiž tuto adresu používá pro dočasné uložení parametru ICAX2. V tom případě by bylo vhodné doplnit rutinu CLOSE o část, která by uvedla joystick port do původního stavu.

Velmi účelné by také bylo zařadit handler do některého z kazetových či diskových operačních systémů. Tak by byl univerzálně použitelný.

Tomáš Bělík, Jiří Skála

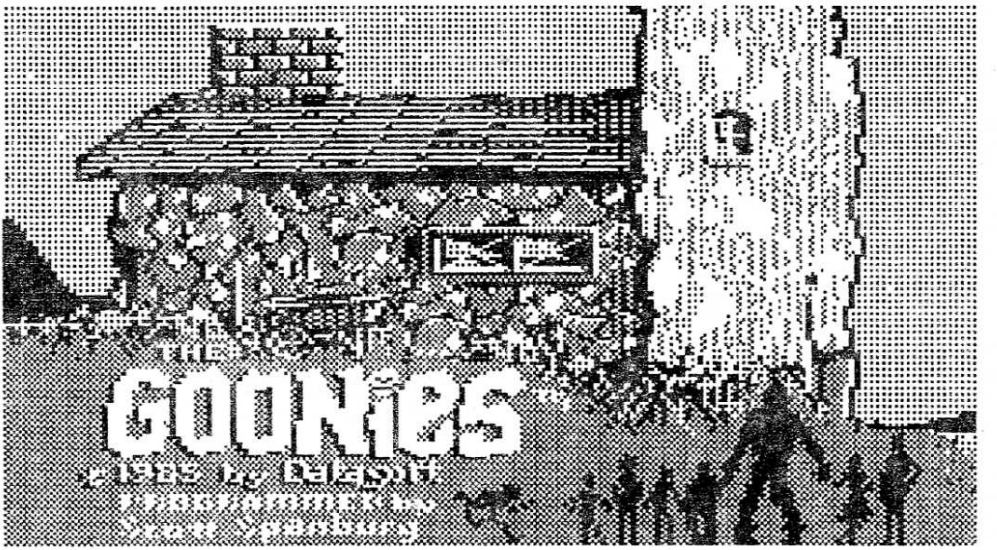
Výpis obslužného programu:

```
                1000 ; HANDLER PRO PSACÍ STROJ P: V1.4C
                1005 ; (Tomáš Bělík, 1988)
                1010 ;
                1020 ;Systémové adresy:
                1030 ;
=031A           1040 HATABS = $031A  ;tabulka vektorů zařízení
=002A           1050 ICAX1Z = $2A    ;pomocný registr ICAX1
=FDFC           1060 BELL =  $FDFC   ;zapípá a čeká na stisk klávesy
=C642           1070 DISPLAY = $C642 ;vytiskne řetězec znaků
=D300           1080 PORTA = 54016
=D302           1090 PORTCNTL = 54018
=D010           1100 TRIG0 = $D010   ;ps READY (připojen):TRIG0=0
=D011           1110 TRIG1 = $D011   ;PS přijme další znak:TRIG1=0
=00C9           1120 TABULATOR = $C9 ;o kolik mezer se posune tisk
                1125                 ;při TAB
                1130 ;
                1140 ;Pomocné adresy:
                1150 ;
=001F           1160 COUNT = $1F     ;čítač řádků
=002B           1170 RADKY = $2B     ;(ICAX2Z) počet tištěných
                1175                 ;řádek na stránku
                1180 ;
```

```
0000            1190     *=  $05F8
                1200 ;
                1210 ;Instalace zařízení "P:"
                1220 ;
05F8 68         1230 INITB PLA       ;pouze při inicializaci z Basicu
05F9 A941       1240 INIT LDA # <HAND ;uložení adresy handleru
05FB 8D1B03     1250     STA HATABS+1 ;na HATABS
05FE A906       1260     LDA # >HAND
0600 8D1C03     1270     STA HATABS+2
0603 20B206     1280     JSR START   ;vypíše text
0606 60         1290     RTS
                1300 ;
                1310 ;Rutina pro otevření kanálu
                1320 ;
0607 A52A       1330 OPEN LDA ICAX1Z ;?použití pouze pro zápis
0609 C908       1340     CMP #8
060B D01E       1350     BNE GETB
060D AC10D0     1360     LDY TRIG0   ;ano, ?je PS připojen
0610 D01C       1370     BNE TIMEO
0612 841F       1380     STY COUNT   ;ano, COUNT:=0
0614 A2FF       1390     LDX #255    ;nastavení portu A na OUT
0616 A938       1400     LDA #56
0618 8D02D3     1410     STA PORTCNTL
061B 8E00D3     1420     STX PORTA
061E A93C       1430     LDA #60
0620 8D02D3     1440     STA PORTCNTL
0623 A940       1450 VALINIT LDA #64 ;inicializace válce (na začátek)
0625 203106     1460     JSR ZNAK
0628 A001       1470 CLOSE LDY #1    ;vše OK
062A 60         1480     RTS
062B A083       1490 GETB LDY #131   ;pouze pro zápis
062D 60         1500 GETS RTS
062E A08A       1510 TIMEO LDY #138  ;PS neodpovídá
0630 60         1520     RTS
                1530 ;ZNAK: vytiskne znak na PS
                1540 ;
0631 AC11D0     1550 ZNAK LDY TRIG1  ;?přijme PS další znak
0634 D0FB       1560     BNE ZNAK    ;ne,tak čekej
0636 49FF       1565     EOR #255,   ;viz poznámka
0638 8D00D3     1570     STA PORTA   ;ano,pošli znak na PORTA
063B 297F       1580     AND #127    ;aktivuj "STROBE",tj P7=0
063D 8D00D3     1590     STA PORTA
0640 60         1600     RTS
                1610 ;
                1620 ;Tabulka handleru
                1630 ;
0641 0606       1640 HAND .WORD OPEN-1
0643 2706       1650     .WORD CLOSE-1
0645 2A06       1660     .WORD GETB-1
0647 4C06       1670     .WORD PUTB-1
0649 2C06       1680     .WORD GETS-1
064B 2C06       1690     .WORD GETS-1 ;SPEC=GETSTATUS - zde zbytečné
                1700 ;
                1710 ;PUTB: vyšle bajt na zařízení
                1720 ;
064D C97F       1730 PUTB CMP #127   ;test na tabulátor
```

```
064F D00C       1740     BNE NETAB   ;není-li TAB pokračuj
0651 A2C9       1750     LDX #TABULATOR ;do X ulož kolik mezer je TAB
0653 A910       1760 TISKTAB LDA #16 ;tisk mezery
0655 203106     1770     JSR ZNAK
0658 CA         1780     DEX         ;decrement počtu mezer,které se
                1785                 ;tisknou místo TAB
0659 D0F8       1790     BNE TISKTAB ;?uz jsou všechny,ne pokračuj
                1795                 ;v tisku
065B F0CB       1800     BEQ CLOSE   ;jsou všechny tak konec
065D C99B       1810 NETAB CMP #155  ;?návrat válce
065F D018       1820     BNE NERET   ;ne, pokračuj
0661 A940       1830     LDA #64     ;ano,proveď návrat válce
0663 203106     1840     JSR ZNAK
0666 E61F       1850     INC COUNT   ;COUNT:=COUNT+1
0668 A92B       1860     LDA #RADKY
066A C51F       1870     CMP COUNT   ;?je už celá stránka
066C D0BA       1880 SKOKCLOSE BNE CLOSE
066E A900       1890     LDA #0      ;ano,vynuluj čítač
0670 851F       1900     STA COUNT
0672 A903       1910     LDA #3
0674 20FCFD     1920     JSR BELL    ;zazvoň a čekej na stisk
                1925                 ;klávesy (libovolné)
0677 D0AA       1930     BNE VALINIT
0679 297F       1940 NERET AND #127  ;ignorace inverzních znaků
067B E91F       1950     SBC #31     ;ignorace gr.symbolů tj.
067D 1002       1960     BPL POKRA   ;jsou jako mezera a celý kód
                1965                 ;se posune
067F 2900       1970     AND #0      ;o 31,mezera=0,!=1 atd.
0681 AA         1980 POKRA TAX       ;uschování A
0682 F023       1990     BEQ TISK    ;? mezera (není třeba
                1995                 ;nastavovat velká/malá)
0684 2940       2000     AND #64     ;ne,?jde o malá písmena
0686 F006       2010     BEQ POKRB
0688 8A         2020     TXA         ;ano,převeď je na velká
0689 E920       2030     SBC #32
068B AA         2040     TAX
068C D005       2050     BNE MALA    ;ale PS nastav na malá
068E BDBF06     2060 POKRB LDA TAB,X ;?je znak z tabulky
                2065                 ;malé/velké písmeno
0691 3004       2070     BMI VELKA
0693 A93C       2080 MALA LDA #60    ;malé,nastav příslušné
                2085                 ;PS na "1..."
0695 D002       2090     BNE INVEL
0697 A93A       2100 VELKA LDA #58   ;velké, nastav na PS "A..."
0699 203106     2110 INVEL JSR ZNAK
069C BDBF06     2120     LDA TAB,X   ;?je písmeno s háčkem
                2125                 ;(čárkou) nebo ne
069F 4A         2130     LSR A
06A0 9005       2140     BCC TISK
06A2 A90A       2150     LDA #10     ;ano,tak vytiskni háček(čárku)
06A4 203106     2160     JSR ZNAK
06A7 BDBF06     2170 TISK LDA TAB,X  ;vlastní tisk znaku
06AA 4A         2180     LSR A       ;odsuň bit indikující
                2185                 ;háček (čárku)
06AB 293F       2190     AND #63     ;vymaskuj horní dva bity
06AD 203106     2200     JSR ZNAK    ;a vytiskni
```

```
06B0 D0BA       2210     BNE SKOKCLOSE ;navrat
06B2 A2BA       2220 START LDX # <TEXT ;výpis textu
06B4 A006       2230     LDY # >TEXT
06B6 2042C6     2240     JSR DISPLAY
06B9 60         2250     RTS
                2260 ;
                2270 ;Text "P:ok"
                2280 ;
06BA 503A6F6B   2290 TEXT .BYTE "P:ok"
06BE 9B         2300     .BYTE $9B
                2310 ;
                2320 ;Převodní tabulka - upravený ATASCII na
                2325 ;inverzní kód PS
                2330 ;
06BF A0D68061   2340 TAB .BYTE 160,214,128,97
06C3 E502C380   2350     .BYTE 229,2,195,128
06C7 B4346FE0   2360     .BYTE 180,52,111,224
06CB F63656B6   2370     .BYTE 246,54,86,182
06CF C08284A6   2380     .BYTE 192,130,132,166
06D3 88AAAC8E   2390     .BYTE 136,170,172,142
06D7 90B2E254   2400     .BYTE 144,178,226,84
06DB 29604BD4   2410     .BYTE 41,96,75,212
06DF 10C2C4E6   2420     .BYTE 16,194,196,230
06E3 C8EAECCE   2430     .BYTE 200,234,236,206
06E7 D0F2A2A4   2440     .BYTE 208,242,162,164
06EB 86A88A8C   2450     .BYTE 134,168,138,140
06EF AEB092E4   2460     .BYTE 174,176,146,228
06F3 C6E8CACC   2470     .BYTE 198,232,202,204
06F7 EEF0D2B5   2480     .BYTE 238,240,210,181
06FB B735AF62   2490     .BYTE 183,53,175,98
```



ATARI, technický zpravodaj pro mikroelektroniku a výpočetní techniku. Redaktor: Jiří Skála. Adresa redakce: Klub 602 Čs. spol. elektroniků, Slezská 130, 130 00 Praha 3. Vydává Čs. hifiklub, Wintrova 8, 160 41 Praha 6. Povoleno ÚVTEI pod ev. č. 87006. Vytiskla tiskárna Čs. hifiklubu, Praha 6, Wintrova 11.
Březen 1991

Igi/2024